aws

開發人員指南

Amazon Elastic Container Service

Amazon Elastic Container Service: 開發人員指南

Table of Contents

什麼是 Amazon Elastic Container Service？

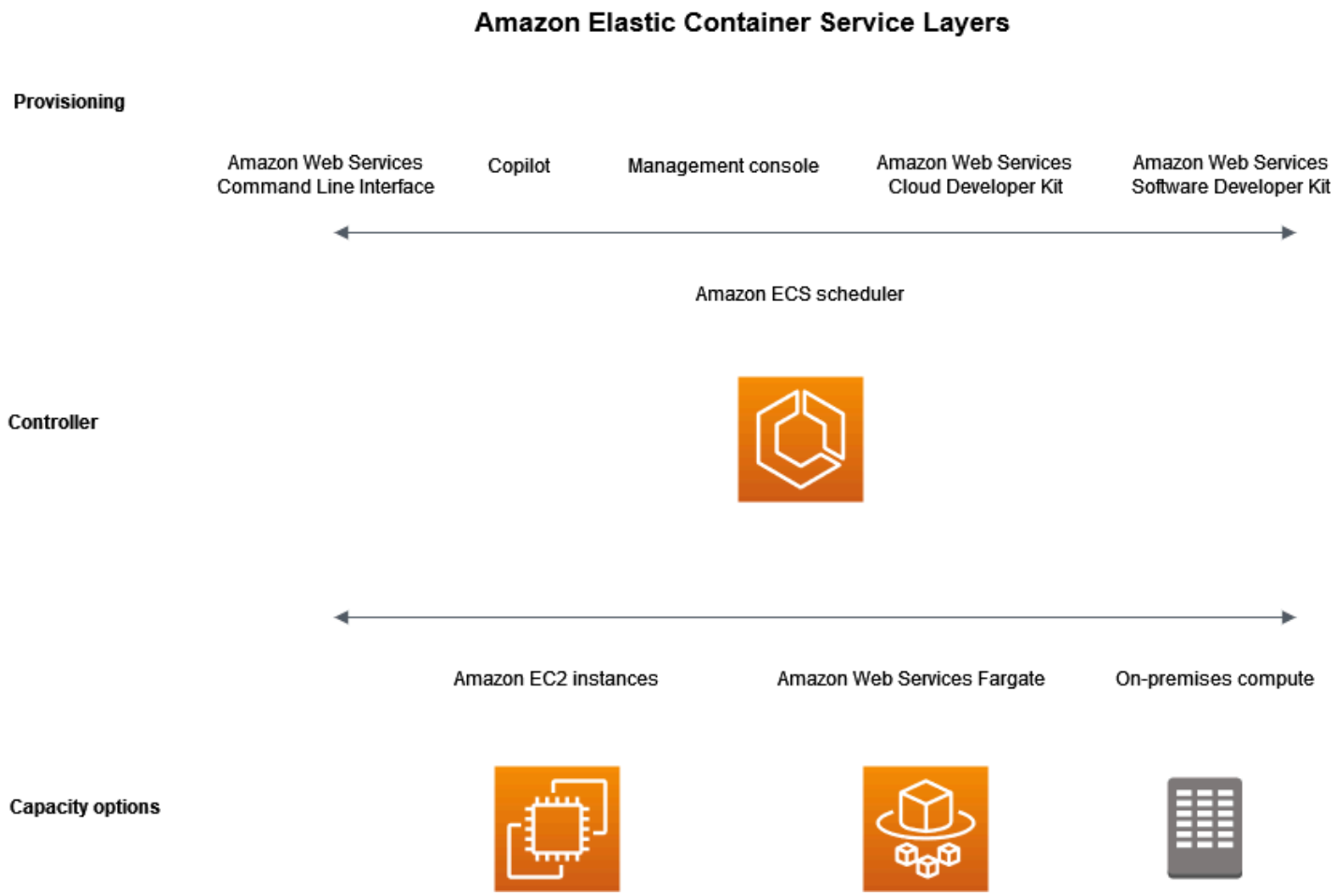
Amazon Elastic Container Service (Amazon ECS) 是全受管容器協調服務，可協助您輕鬆部署、管理和擴展容器化應用程式。Amazon 是全受管服務，ECS內建 AWS 組態和操作最佳實務。它與 Amazon Elastic Container Registry 等 AWS 工具以及 Docker 等第三方工具整合。這種整合可讓團隊能夠更輕鬆地專注於建置應用程式，而無需為環境分心。您可以在雲端和內部部署 AWS 區域 中跨 執行和擴展容器工作負載，而無須複雜的控制平面管理。

Amazon ECS術語和元件

Amazon 有三個圖層ECS：

- 容量 - 您的容器執行所在的基礎設施
- 控制器 - 部署和管理在容器上執行的應用程式
- 佈建 - 可用於與排程器連接以部署和管理應用程式和容器的工具

下圖顯示 Amazon ECS層。



Amazon ECS容量

Amazon ECS容量是容器執行所在的基礎設施。以下是容量選項概觀：

- AWS 雲端中的 Amazon EC2執行個體

  您可以選擇執行個體類型、執行個體數量，以及管理容量。

- AWS 雲端中的無伺服器 (AWS Fargate)

  Fargate 是無伺服器運算 pay-as-you-go引擎。搭配使用 Fargate，您無需管理伺服器、處理容量規劃，或出於安全性而隔離容器工作負載。

- 內部部署虛擬機器或伺服器

  Amazon ECS Anywhere 支援將內部部署伺服器或虛擬機器 (VM) 等外部執行個體註冊到您的 Amazon ECS叢集。

容量可以位於下列任何 AWS 資源中：

- 可用區域
- 本機區域
- Wavelength 區域
- AWS 區域
- AWS Outposts

Amazon ECS控制器

Amazon ECS排程器是管理您的應用程式的軟體。

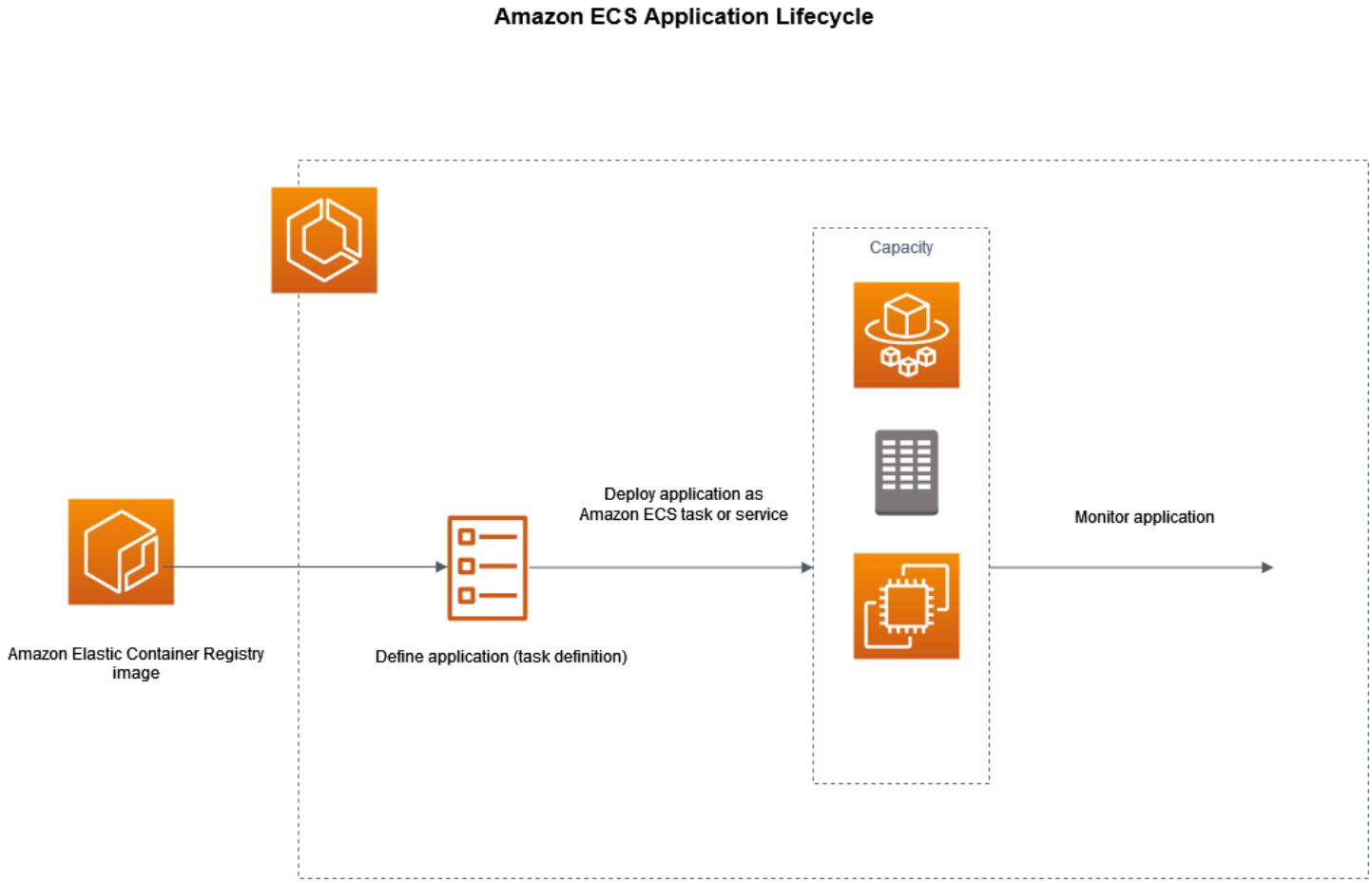
Amazon ECS佈建

佈建 Amazon 有多個選項ECS：

- AWS Management Console — 提供 Web 界面，供您用來存取 Amazon ECS 資源。
- AWS Command Line Interface (AWS CLI) — 為廣泛的 AWS 服務提供命令，包括 Amazon ECS。Windows、Mac 和 Linux 均支援此介面。如需詳細資訊，請參閱AWS Command Line Interface。
- AWS SDKs — 提供特定語言APIs，並負責許多連線詳細資訊。包括計算簽章、處理請求重試和錯誤處理。如需詳細資訊，請參閱AWS SDKs。
- Copilot — 為開發人員提供開放原始碼工具，以在 Amazon 上建置、發行和操作生產就緒的容器化應用程式ECS。如需詳細資訊，請參閱 GitHub網站上的 Copilot。
- AWS CDK - 提供開放原始碼軟體開發架構，您可以使用該架構來建模，並使用熟悉的程式設計語言來佈建您的雲端應用程式資源。 AWS CDK 會透過 AWS CloudFormation以安全、可重複的方式佈建您的資源。

應用程式生命週期

下圖顯示應用程式生命週期及其與 Amazon ECS元件搭配使用的方式。



您必須建構您的應用程式，以便它們可以在容器上執行。容器是軟體開發的標準化單元，可容納軟體應用程式執行所需的所有一切。這包括相關代碼、執行時間、系統工具和系統庫。容器依據稱為映像的唯讀範本而建立。映像通常由 Dockerfile 建立。Dockerfile 是純文字檔案，其中包含建置容器的指示。建置這些映像之後，這些映像會存放在登錄檔中，例如可從ECR中下載的 Amazon。

建立並存放映像之後，您會建立 Amazon ECS任務定義。任務定義是您應用程式的藍圖。它是 JSON 格式的文字檔案，描述參數和一個或多個構成您應用程式的容器。例如，您可以使用其來指定作業系統的映像和參數、要使用的容器、要為您的應用程式開啟的連接埠，以及任務中的容器要使用的資料磁碟區。任務定義可用的特定參數取決於您特定應用程式的需求。

定義任務定義之後，您可以將其部署為叢集上的服務或任務。叢集是在註冊至叢集的容量基礎結構上執行的任務或服務的邏輯群組。

任務是在叢集內將任務定義執行個體化。您可以執行獨立任務，也可以將任務作為服務的一部分執行。您可以使用 Amazon ECS服務，在 Amazon ECS叢集中同時執行和維護所需數量的任務。運作方式是，如果您的任何任務因任何原因失敗或停止，Amazon ECS服務排程器會根據您的任務定義啟動另一個執行個體。這樣就可以取代該任務，從而在服務中保持所需的任務數量。

容器代理程式會在 Amazon ECS叢集內的每個容器執行個體上執行。代理程式會將目前執行中的任務和資源使用率的相關資訊傳送至 Amazon ECS。它會在收到來自 Amazon 的請求時啟動和停止任務 ECS。

部署任務或服務後，您可以使用下列任何工具來監控部署和應用程式：

- CloudWatch
- 執行期監控

Amazon ECS 相關資訊

以下相關資源可協助您使用此服務。

- AWS Fargate – Fargate 功能概觀。
- Windows on AWS – Windows AWS 工作負載和服務概觀。
- Linux from AWS – 現代 Linux 作業系統的產品組合 AWS。

開發人員工具

- AWS App2Container – 將 .NET 和 Java 應用程式現代化為容器化應用程式的命令列工具。
- 適用於 Amazon Web Services 的工具 – 用於 AWS SDKs管理各種程式設計語言的 Amazon ECS 資源和操作。
- AWS CloudFormation – 使用自動化部署來定義和管理環境中的所有 AWS 資源。
- 使用 AWS Copilot 命令列介面建立 Amazon ECS 資源 – End-to-end 開發人員工作流程，用於建立、發行和操作符合基礎設施 AWS 最佳實務的容器應用程式。
- AWS Cloud Development Kit (AWS CDK) – Infrastructure-as-Code(IAC) 架構，您可以使用您選擇的程式設計語言來定義 AWS 雲端基礎設施。

開發人員教學課程

- AWS Compute Blogs – 有關新功能、深入探討功能、程式碼範例和最佳實務的資訊。

AWS re:Post

- AWS re:Post – AWS 受管問題和答案 (Q & A) 服務，為您的技術問題提供群眾來源、專家審查的答案。

定價

- Amazon ECS定價 – Amazon 的定價資訊ECS。
- AWS Fargate 定價 – Fargate 定價資訊。

一般 AWS 資源

下列一般資源可協助您使用 AWS。

- 課程與研討會 – 連結至以角色為基礎的特殊課程，以及自行安排進度的實驗室，以協助強化您的 AWS 技能並取得實際經驗。
- AWS 開發人員中心 – 探索教學課程、下載工具，並了解 AWS 開發人員事件。
- AWS 開發人員工具 – 開發人員工具、、SDKsIDE工具組和命令列工具的連結，用於開發和管理 AWS 應用程式。
- 入門資源中心 – 了解如何設定您的 AWS 帳戶、加入 AWS 社群，以及啟動您的第一個應用程式。
- 實作教學課程 – 遵循 step-by-step教學課程來啟動您的第一個應用程式 AWS。
- AWS 白皮書 – 技術 AWS 白皮書的完整清單連結，涵蓋架構、安全和經濟等主題，並由 AWS Solutions Architects 或其他技術專家撰寫。
- AWS 支援 中心 – 建立和管理 AWS 支援 案例的中樞。也包含其他實用資源的連結，例如論壇、技術 、FAQs服務運作狀態 和 AWS Trusted Advisor。
- 支援 – 有關 的資訊的主要網頁 支援，快速 one-on-one回應的支援管道，可協助您在雲端中建置和執行應用程式。
- 聯絡我們 – 查詢有關 AWS 帳單、帳戶、事件、濫用與其他問題的聯絡中心。
- AWS 網站條款 – 有關我們的著作權和商標、您的帳戶、授權和網站存取，以及其他主題的詳細資訊。

了解如何建立和使用 Amazon ECS 資源

下列指南提供可存取 Amazon ECS 的工具簡介，以及執行容器的入門程序。Docker 基本概念會逐步引導您透過基本步驟建立 Docker 容器映像並將其上傳到 Amazon ECR 私有儲存庫。入門指南會逐步引導您使用 AWS Copilot 命令列界面和 AWS Management Console 來完成在 Amazon ECS 和 上執行容器的常見任務 AWS Fargate。

目錄

設定以使用 Amazon ECS。

如果您已經註冊 Amazon Web Services (AWS) 且已在使用 Amazon Elastic Compute Cloud (Amazon EC2)，您也幾乎可使用 Amazon ECS 了。這兩項服務的設定程序很類似。下列指南將協助您為啟動您的首個 Amazon ECS 叢集做好準備。

完成下列任務，為 Amazon ECS 進行設定。

AWS Management Console

AWS Management Console 是以瀏覽器為基礎的介面，用於管理 Amazon ECS 資源。主控台提供服務的可視概觀，讓您無需使用其他工具即可輕鬆探索 Amazon ECS 功能和函數。提供許多相關的教學課程和演練，可以引導您使用主控台。

如需主控台的指導教學，請參閱 了解如何建立和使用 Amazon ECS 資源。

一開始，許多客戶偏好使用 主控台，因為它會提供即時視覺化意見回饋，說明他們採取的動作是否成功。熟悉 AWS 的客戶 AWS Management Console可以輕鬆管理負載平衡器和 Amazon EC2 執行個體等相關資源。

從 開始 AWS Management Console。

註冊 AWS 帳戶

如果您沒有 AWS 帳戶，請完成下列步驟來建立一個 。

註冊 AWS 帳戶

1. 開啟 https://portal.aws.amazon.com/billing/signup。
2. 請遵循線上指示進行。

   部分註冊程序需接收來電，並在電話鍵盤輸入驗證碼。

   當您註冊 時 AWS 帳戶，AWS 帳戶根使用者會建立 。根使用者有權存取該帳戶中的所有 AWS 服務 和資源。作為安全最佳實務，請將管理存取權指派給使用者，並且僅使用根使用者來執行需要根使用者存取權的任務。

AWS 會在註冊程序完成後傳送確認電子郵件給您。您可以隨時登錄 https://aws.amazon.com/ 並選擇我的帳戶，以檢視您目前的帳戶活動並管理帳戶。

建立具有管理存取權的使用者

註冊 後 AWS 帳戶，請保護 AWS 帳戶根使用者、啟用 AWS IAM Identity Center和建立管理使用者，以免將根使用者用於日常任務。

保護您的 AWS 帳戶根使用者

1. 選擇根使用者並輸入 AWS 帳戶 您的電子郵件地址，以帳戶擁有者AWS Management Console身分登入 。在下一頁中，輸入您的密碼。

   如需使用根使用者登入的說明，請參閱 AWS 登入 使用者指南中的以根使用者身分登入。
2. 若要在您的根使用者帳戶上啟用多重要素驗證 (MFA)。

   如需說明，請參閱《IAM 使用者指南》中的為您的 AWS 帳戶 根使用者 （主控台） 啟用虛擬 MFA 裝置。

建立具有管理存取權的使用者

1. 啟用 IAM Identity Center。

如需指示，請參閱《AWS IAM Identity Center 使用者指南》中的啟用 AWS IAM Identity Center。

2. 在 IAM Identity Center 中，將管理存取權授予使用者。

   如需使用 IAM Identity Center 目錄 做為身分來源的教學課程，請參閱AWS IAM Identity Center 《 使用者指南》中的使用預設值設定使用者存取權 IAM Identity Center 目錄。

以具有管理存取權的使用者身分登入

- 若要使用您的 IAM Identity Center 使用者簽署，請使用建立 IAM Identity Center 使用者時傳送至您電子郵件地址的簽署 URL。

  如需使用 IAM Identity Center 使用者登入的協助，請參閱AWS 登入 《 使用者指南》中的登入 AWS 存取入口網站。

指派存取權給其他使用者

1. 在 IAM Identity Center 中，建立一個許可集來遵循套用最低權限的最佳實務。

   如需指示，請參閱《AWS IAM Identity Center 使用者指南》中的建立許可集。

2. 將使用者指派至群組，然後對該群組指派單一登入存取權。

   如需指示，請參閱《AWS IAM Identity Center 使用者指南》中的新增群組。

建立 Virtual Private Cloud

您可以使用 Amazon Virtual Private Cloud (Amazon VPC) 將 AWS 資源啟動至您定義的虛擬網路。強烈建議您在 VPC 中啟動您的容器執行個體。

如果您有預設 VPC，可以跳過本節，並進行下一個任務「建立安全群組」。若要判斷您是否有預設 VPC，請參閱《Amazon VPC 使用者指南》中的使用預設 VPC 和預設子網路。否則，您可以利用下面的步驟，在帳戶中建立非預設 VPC。

如需有關如何建立 VPC 的資訊，請參閱《Amazon VPC 使用者指南》中的建立 VPC，並使用下表來決定要選取的選項。

選項	Value
要建立的資源	僅 VPC
名稱	可以選擇為 VPC 提供名稱。
IPv4 CIDR 區塊	IPv4 CIDR 手動輸入 CIDR 區塊大小必須為介於 /16 和 /28 之間的大小。
IPv6 CIDR 區塊	無 IPv6 CIDR 區塊
租用	預設

如需有關 Amazon VPC 的詳細資訊，請參閱《Amazon VPC 使用者指南》中的什麼是 Amazon VPC？。

建立安全群組

安全群組的功能為相關聯容器執行個體的防火牆，可在容器執行個體層級控制傳入和傳出流量。您可以新增規則至安全群組，讓您從您的 IP 地址使用 SSH 連接到您的容器執行個體。您也可以新增允許任何位置之傳入和傳出 HTTP 和 HTTPS 存取的規則。依您的任務所需在開放連接埠新增任何規則。容器執行個體需要外部網路存取，才可以與 Amazon ECS 服務端點通訊。

如果您打算在多個區域中啟動容器執行個體，則需要在每個區域中建立安全群組。如需詳細資訊，請參閱《Amazon EC2 使用者指南》中的區域和可用區域。

> **Tip**
>
> 您需要本機電腦的公有 IP 地址 (可以使用服務來取得)。例如，我們提供下列服務：http://checkip.amazonaws.com/ 或 https://checkip.amazonaws.com/。若要尋找其他能夠提供您 IP 地址的服務，請使用搜尋片語 "what is my IP address" (我的 IP 地址為何)。如果您透過網際網路服務供應商 (ISP) 或是經由不具靜態 IP 地址的防火牆連線，則必須找出用戶端電腦使用的 IP 地址範圍。

如需如何建立安全群組的詳細資訊，請參閱《Amazon EC2 使用者指南》中的為您的 Amazon EC2 執行個體建立安全群組，並使用下表來決定要選取的選項。 Amazon EC2

選項	Value
區域	您建立金鑰對的同一個區域。
名稱	一個您易記的名稱 (例如 ecs-instances-default-cluster)。
VPC	預設 VPC (有「預設」標記)。 **Note** 如果您的帳戶支援 Amazon EC2 Classic，請選取您在先前任務中建立的 VPC。

如需針對您的使用案例新增傳出規則的相關資訊，請參閱《Amazon EC2 使用者指南》中的不同使用案例的安全群組規則。

Amazon ECS 容器執行個體不需要開啟任何傳入連接埠。不過建議您新增 SSH 規則，以登入容器執行個體，並以 Docker 命令檢查任務。如果您希望您的容器執行個體託管執行 Web 伺服器的任務，則也可以新增 HTTP 和 HTTPS 的規則。而容器執行個體需要外部網路存取，才可以與 Amazon ECS 服務端點通訊。完成下列步驟，以新增這些選用的安全群組規則。

將下列三個傳入規則新增至您的安全群組。如需如何建立安全群組的相關資訊，請參閱《Amazon EC2 使用者指南》中的設定安全群組規則。

選項	Value
HTTP 規則	類型：HTTP 來源：Anywhere (`0.0.0.0/0`) 此選項會自動新增 0.0.0.0/0 IPv4 CIDR 區塊作為來源。通常在測試環境中短暫進行此

選項	Value	
	操作是沒有問題的，但用在生產環境則不安全。在生產環境中，建議您只授權特定 IP 地址或特定範圍的地址存取您的執行個體。	
HTTPS 規則	Type (類型)：HTTPS 來源：Anywhere (`0.0.0.0/0`) 通常在測試環境中短暫進行此操作是沒有問題的，但用在生產環境則不安全。在生產環境中，建議您只授權特定 IP 地址或特定範圍的地址存取您的執行個體。	

選項	Value
SSH 規則	Type (類型)：SSH 來源：Custom (自訂)，並以 CIDR 表示法來指定您的電腦或網路的公有 IP 位址。若要以 CIDR 表示法指定個別 IP 地址，請新增路由前綴 /32。例如，如果您的 IP 地址為 `203.0.113.25` ，請指定 `203.0.113.25/32` 。如果您的公司會分配某個範圍的地址，請指定整個範圍 (例如 `203.0.113.0/24`)。 **Important** 基於安全考量，不建議您允許從所有 IP 地址 (`0.0.0.0/0`) 對您執行個體進行 SSH 存取，除非僅為短時間測試。

建立憑證來連線至 EC2 執行個體

針對 Amazon ECS，只有當您想要使用 EC2 啟動類型時，才需要金鑰對。

AWS 使用公有金鑰密碼編譯來保護執行個體的登入資訊。Linux 執行個體 (例如 Amazon ECS 容器執行個體) 沒有密碼用於 SSH 存取。您需要使用金鑰對，以安全地登入執行個體。當您啟動容器執行個體時，要指定金鑰對名稱，再於使用 SSH 登入時，提供私有金鑰。

如果您尚未建立金鑰對，可以使用 Amazon EC2 主控台來建立。如果您打算在多個區域中啟動執行個體，則需要在每個區域中建立金鑰對。如需區域的詳細資訊，請參閱《Amazon EC2 使用者指南》中的區域和可用區域。

建立一組金鑰對

- 使用 Amazon EC2 主控台來建立金鑰對。如需建立金鑰對的詳細資訊，請參閱《Amazon EC2 使用者指南》中的建立金鑰對。

如需如何連線至執行個體的詳細資訊，請參閱《Amazon EC2 使用者指南》中的連線至 Linux 執行個體。

安裝 AWS CLI

AWS Management Console 可用於使用 Amazon ECS 手動管理所有操作。不過，您可以在 AWS CLI 本機桌面或開發人員方塊中安裝 ，以便建置指令碼來自動化 Amazon ECS 中的常見管理任務。

若要 AWS CLI 搭配 Amazon ECS 使用 ，請安裝 AWS CLI 最新版本。如需安裝 AWS CLI 或將其升級至最新版本的詳細資訊，請參閱AWS Command Line Interface 《 使用者指南》中的安裝或更新至最新版本的 AWS CLI。

The AWS Command Line Interface (AWS CLI) 是統一的工具，可用來管理 AWS 服務。僅使用此單一工具，您就可以透過指令碼來控制多個 AWS 服務並自動化這些服務。中的 Amazon ECS 命令 AWS CLI 是 Amazon ECS API 的反射。

AWS CLI 適合偏好 且用來編寫指令碼和與命令列工具互動的客戶，並確切知道他們想要在 Amazon ECS 資源上執行哪些動作。對於想要熟悉 Amazon ECS APIs 的客戶 AWS CLI 也很有幫助。客戶可以使用 AWS CLI 直接從命令列界面對 Amazon ECS 資源執行多項操作，包括建立、讀取、更新和刪除操作。

AWS CLI 如果您正在或想要熟悉 Amazon ECS APIs 和對應的 CLI 命令，並想要撰寫自動化指令碼，並在 Amazon ECS 資源上執行特定動作，請使用 。

AWS 也提供命令列工具 AWS Tools for Windows PowerShell。如需詳細資訊，請參閱《AWS Tools for Windows PowerShell 使用者指南》https://docs.aws.amazon.com/powershell/latest/userguide/。

使用 Amazon ECS 的後續步驟

安裝 後 AWS CLI，您可以繼續使用 Amazon ECS 時，使用許多不同的工具。下列連結說明這些工具有哪些，並提供如何搭配 Amazon ECS 使用它們的範例。

- 使用 Docker 建立您的第一個容器映像，並將其推送至 Amazon ECR，以用於您的 Amazon ECS 任務定義。

- 了解如何為 Fargate 啟動類型建立 Amazon ECS Linux 任務.
- 了解如何為 Fargate 啟動類型建立 Amazon ECS Windows 任務.
- 了解如何為 EC2 啟動類型建立 Amazon ECS Windows 任務.
- 使用您偏好的程式設計語言，利用 使用 建立 Amazon ECS 資源 AWS CDK 將基礎設施或架構定義為程式碼。
- 使用 自動部署來定義和管理環境中的所有 AWS 資源使用 建立 Amazon ECS 資源 AWS CloudFormation。
- 使用完整的使用 AWS Copilot 命令列介面建立 Amazon ECS 資源end-to-end開發人員工作流程來建立、發行和操作符合基礎設施 AWS 最佳實務的容器應用程式。

建立在 Amazon ECS 上使用的容器映像

Amazon ECS 使用任務定義中的 Docker 映像來啟動容器。Docker 是一種技術，可為您提供建置、執行、測試和部署分散式應用程式所需的工具。

此處概述的步驟旨在逐步引導您建立第一個 Docker 映像，並將該映像推送到 Amazon ECR (即容器登錄檔)，以便在 Amazon ECS 任務定義中使用。此引導過程假定您對 Docker 的含義和其運作方式有基本的了解。如需 Docker 的詳細資訊，請參閱什麼是 Docker？ 和 Docker 文件。

必要條件

開始之前，請務必先達成以下先決條件。

- 確保您已完成 Amazon ECR 設定步驟。如需詳細資訊，請參閱《Amazon Elastic Container Registry 使用者指南》中的在 Amazon ECR 中移動映像的生命週期。
- 您的使用者已具備存取和使用 Amazon ECR 服務所需的 IAM 許可。如需詳細資訊，請參閱 Amazon ECR 受管政策。
- 您已安裝 Docker。如需有關 Amazon Linux 2 Docker 的安裝步驟，請參閱 在 AL2023 上安裝 Docker。如需其他作業系統的相關資訊，請參閱 Docker Desktop 概觀中的 Docker 文件。
- 您已 AWS CLI 安裝並設定 。如需詳細資訊，請參閱AWS Command Line Interface 《 使用者指南》中的安裝或更新至最新版本的 AWS CLI。

若您沒有或不需要本機開發環境，而且您偏好透過 Amazon EC2 執行個體使用 Docker，我們提供以下步驟來使用 Amazon Linux 2 啟動 Amazon EC2 執行個體並安裝 Docker Engine 和 Docker CLI。

在 AL2023 上安裝 Docker

Docker 可在多個不同的作業系統上使用，包括大部分的現代 Linux 發行版本，例如 Ubuntu，甚至是 macOS 和 Windows。如需如何在特定作業系統上安裝 Docker 的詳細資訊，請前往「Docker 安裝指南」。

您不需要本機開發系統，就能使用 Docker。如果您已在使用 Amazon EC2，則可以啟動 Amazon Linux 2023 執行個體，並安裝 Docker 以開始使用。

如果您已經安裝 Docker，請跳到「建立 Docker 映像」。

使用 Amazon Linux 2023 AMI 在 Amazon EC2 執行個體上 安裝 Docker

1. 使用最新版 Amazon Linux 2023 AMI 啟動執行個體。如需詳細資訊，請參閱《Amazon EC2 使用者指南》中的使用 主控台中的啟動執行個體精靈啟動 EC2 執行個體。 Amazon EC2
2. 連線到您的執行個體。如需詳細資訊，請參閱《Amazon EC2 使用者指南》中的連線至 EC2 執行個體。 Amazon EC2
3. 更新已安裝的套裝服務，並在執行個體上封裝快取。

   ```
   sudo yum update -y
   ```

4. 安裝最新的 Docker Community Edition 套裝服務。

   ```
   sudo yum install docker
   ```

5. 啟動 Docker 服務。

   ```
   sudo service docker start
   ```

6. 將 ec2-user 新增至 docker 群組，讓您可以在不使用 sudo 的情況下執行 Docker 命令。

   ```
   sudo usermod -a -G docker ec2-user
   ```

7. 登出並重新登入，以取得新的 docker 群組許可。關閉目前的 SSH 終端機視窗，即可完成此操作，並在新的 SSH 終端機視窗中重新連接至執行個體。新的 SSH 工作階段將會有適當的 docker 群組許可。
8. 驗證 ec2-user 可以在不使用 sudo 的情況下執行 Docker 命令。

   ```
   docker info
   ```

> **Note**
>
> 在某些情況下，您可能需要重新啟動執行個體，才能提供 ec2-user 存取 Docker 常駐程式的許可。如果您看到下列錯誤，請嘗試重新啟動執行個體：
>
> ```
> Cannot connect to the Docker daemon. Is the docker daemon running on this
>  host?
> ```

建立 Docker 映像

Amazon ECS 任務定義使用 Docker 映像，藉以啟動您叢集中之容器執行個體上的容器。在本節中，您將建立簡單 Web 應用程式的 Docker 映像，並在本機系統或 Amazon EC2 執行個體上進行測試，然後將映像推送至 Amazon ECR 容器登錄檔，讓您可以在 Amazon ECS 任務定義中使用。

建立簡單 Web 應用程式的 Docker 映像

1. 建立稱為 Dockerfile 的檔案。Dockerfile 是一種資訊清單，說明用於您 Docker 映像的基本映像，以及您要安裝並在其上執行的項目。如需 Dockerfile 的詳細資訊，請前往「Dockerfile 參考」。

   ```
   touch Dockerfile
   ```

2. 編輯您剛建立的 Dockerfile，並新增下列內容。

   ```
   FROM public.ecr.aws/amazonlinux/amazonlinux:latest

   # Update installed packages and install Apache
   RUN yum update -y && \
    yum install -y httpd

   # Write hello world message
   RUN echo 'Hello World!' > /var/www/html/index.html

   # Configure Apache
   RUN echo 'mkdir -p /var/run/httpd' >> /root/run_apache.sh && \
    echo 'mkdir -p /var/lock/httpd' >> /root/run_apache.sh && \
    echo '/usr/sbin/httpd -D FOREGROUND' >> /root/run_apache.sh && \
    chmod 755 /root/run_apache.sh
   ```

```
EXPOSE 80

CMD /root/run_apache.sh
```

該 Dockerfile 使用在 Amazon ECR 公共上託管的 Amazon Linux 2 映像。RUN 指令會更新套件快取，並安裝 Web 伺服器的一些軟體套件服務，然後寫入 "Hello World!" 內容至 Web 伺服器文件根目錄。此EXPOSE指示表示容器上的連接埠 80 是正在接聽的連接埠 80，而CMD指示會啟動 Web 伺服器。

3. 從 Dockerfile 建置 Docker 映像。

> **Note**
>
> 在下列命令中，有些 Docker 版本可能需要 Dockerfile 的完整路徑，而不是下面所示的相對路徑。
> 如果您在 MacOS M1/Mx-chip 上執行 命令，請使用 --platform 選項 "--platform linux/amd64"。

```
docker build -t hello-world .
```

4. 列出您的容器映像。

```
docker images --filter reference=hello-world
```

輸出：

```
REPOSITORY          TAG                 IMAGE ID            CREATED
 SIZE
hello-world         latest              e9ffedc8c286        4 minutes ago
 194MB
```

5. 執行新建置的映像。-p 80:80 選項會將容器上的公開連接埠 80 映射至主機系統上的連接埠 80。

```
docker run -t -i -p 80:80 hello-world
```

> **Note**
>
> Apache Web 伺服器中的輸出會顯示在終端機視窗中。您可以忽略 "`Could not reliably determine the fully qualified domain name`" 訊息。

6. 開啟瀏覽器，然後指向執行 Docker 並託管容器的伺服器。

   - 如果您使用的是 EC2 執行個體，則這是伺服器的「公有 DNS」值，這是您使用 SSH 來連線至執行個體的同個地址。請確定您執行個體的安全群組允許連接埠 80 上的入站流量。
   - 如果您在本機執行 Docker，請將瀏覽器指向 http://localhost/。
   - 如果您在 Windows 或 Mac 電腦上使用 docker-machine，則請使用 docker-machine ip 指令找到託管 Docker 之 VirtualBox VM 的 IP 地址，其中將 *`machine-name`* 替代為您所使用之 Docker 機器的名稱。

     ```
     docker-machine ip machine-name
     ```

   您應該會看到網頁，內含您的 "Hello World!" 陳述式。

7. 輸入 Ctrl + c，以停止 Docker 容器。

推送映像至 Amazon Elastic Container Registry

Amazon ECR 是受管 Docker AWS 登錄服務。您可以使用 Docker CLI 在 Amazon ECR 儲存庫中推送、提取與管理映像。如需 Amazon ECR 產品詳細資訊、特色客戶案例研究和常見問答集，請參閱 Amazon Elastic Container Registry 產品詳細資訊頁面。

標記映像並將之推送至 Amazon ECR

1. 建立 Amazon ECR 儲存庫，以便存放 `hello-world` 映像。請記下輸出中的 `repositoryUri`。

   `region`使用 取代 AWS 區域，例如 `us-east-1`。

   ```
   aws ecr create-repository --repository-name hello-repository --region region
   ```

   輸出：

   ```
   {
   ```

```
    "repository": {
        "registryId": "aws_account_id",
        "repositoryName": "hello-repository",
        "repositoryArn": "arn:aws:ecr:region:aws_account_id:repository/hello-
repository",
        "createdAt": 1505337806.0,
        "repositoryUri": "aws_account_id.dkr.ecr.region.amazonaws.com/hello-
repository"
    }
}
```

2. 為 hello-world 映像標記上一步中的 repositoryUri 值。

```
docker tag hello-world aws_account_id.dkr.ecr.region.amazonaws.com/hello-repository
```

3. 執行 aws ecr get-login-password 命令。指定您要驗證的登錄 URI。如需詳細資訊，請參閱《Amazon Elastic Container Registry 使用者指南》中的登錄檔身分驗證。

```
aws ecr get-login-password --region region | docker login --username AWS --
password-stdin aws_account_id.dkr.ecr.region.amazonaws.com
```

輸出：

```
Login Succeeded
```

> **Important**
>
> 若您收到錯誤，請安裝或升級至最新版本的 AWS CLI。如需詳細資訊，請參閱AWS Command Line Interface 《 使用者指南》中的安裝或更新至最新版本的 AWS CLI。

4. 使用先前步驟中的 repositoryUri 值，將映像推送至 Amazon ECR。

```
docker push aws_account_id.dkr.ecr.region.amazonaws.com/hello-repository
```

清除

若要繼續建立 Amazon ECS 任務定義並使用容器映像啟動任務，請跳到 後續步驟。在您試驗完 Amazon ECR 映像後，即可刪除儲存庫，這樣就不會向您收取映像儲存的費用。

```
aws ecr delete-repository --repository-name hello-repository --region region --force
```

後續步驟

任務定義需要任務執行角色。如需詳細資訊，請參閱Amazon ECS 任務執行 IAM 角色。

建立容器映像並將其推送至 Amazon ECR 之後，您可以在任務定義中使用該映像。如需詳細資訊，請參閱下列其中一個項目：

- the section called “了解如何為 Fargate 啟動類型建立 Linux 任務”
- the section called “了解如何為 Fargate 啟動類型建立 Windows 任務”
- 使用 為 Fargate 啟動類型建立 Amazon ECS Linux 任務 AWS CLI

了解如何為 Fargate 啟動類型建立 Amazon ECS Linux 任務

Amazon Elastic Container Service (Amazon ECS) 是具高可擴展性且快速的容器管理服務，可讓您輕鬆執行、停止和管理容器。您可以在 AWS Fargate上啟動服務或任務，以在受 Amazon ECS 管理的無伺服器基礎設施上託管容器。如需 Fargate 的詳細資訊，請參閱 AWS Fargate 適用於 Amazon ECS。

在 Amazon ECS 支援 AWS Fargate Fargate 的 區域中，使用 AWS Fargate 啟動類型的任務，在 上開始使用 Amazon ECS。

完成下列步驟，以在 AWS Fargate上開始使用 Amazon ECS。

必要條件

開始之前，請完成 中的步驟，設定以使用 Amazon ECS。 以及您的 AWS 使用者具有 IAM `AdministratorAccess` 政策範例中指定的許可。

主控台會嘗試自動建立任務執行 IAM 角色，該角色是 Fargate 任務的必要項目。若要確保主控台可成功建立此 IAM 角色，下列其中一項必須為 True：

- 您的使用者具有管理員存取。如需詳細資訊，請參閱設定以使用 Amazon ECS。 。
- 您的使用者具有建立服務角色的 IAM 許可。如需詳細資訊，請參閱建立角色以將許可委派給 AWS 服務。
- 具備管理員存取的使用者已手動建立任務執行角色，讓其可在帳戶上提供使用。如需詳細資訊，請參閱Amazon ECS 任務執行 IAM 角色。

Important

使用任務定義建立服務時選取的安全群組，必須為入站流量開放連接埠 80。將下列傳入規則新增至安全群組。如需有關如何建立安全群組的資訊，請參閱《Amazon EC2 使用者指南》中的為您的 Amazon EC2 執行個體建立安全群組。 Amazon EC2

- Type (類型)：HTTP
- Protocol (通訊協定)：TCP
- 連接埠範圍：80
- 來源：Anywhere (`0.0.0.0/0`)

步驟 1：建立叢集

建立使用預設 VPC 的叢集。

在開始之前，請指派適當的 IAM 許可。如需詳細資訊，請參閱the section called “Amazon ECS 叢集範例”。

1. 開啟主控台，網址為 https://console.aws.amazon.com/ecs/v2。
2. 從導覽列中選取要使用的「區域」。
3. 在導覽窗格中，選擇叢集。
4. 在 Clusters (叢集) 頁面上，選擇 Create cluster (建立叢集)。
5. 在 Cluster configuration (叢集組態) 下的 Cluster name (叢集名稱) 中，輸入唯一的名稱。

   名稱可以包含最多 255 個字母 (大小寫)、數字與連字號。
6. (選用) 若要開啟 Container Insights，請展開 Monitoring (監控)，然後開啟 Use Container Insights (使用 Container Insights)。
7. (選用) 為協助識別您的叢集，請展開 Tags (標籤)，然後設定標籤。

   [新增標籤] 選擇新增標籤，並執行下列動作：

   - 對於 Key (金鑰)，輸入金鑰名稱。
   - 對於 Value (值)，進入金鑰值。

   [移除標籤] 選擇標籤「金鑰」和「值」右側的移除。

8. 選擇 Create (建立)。

步驟 2：建立任務定義

任務定義就像您應用程式的藍圖。每次在 Amazon ECS 中啟動任務時，您都必須指定任務定義。服務接著會知道要為容器使用哪個 Docker 映像、要在任務中使用多少個容器，以及每個容器的資源配置。

1. 在導覽窗格中，選擇 Task Definitions (任務定義)。
2. 選擇 Create new Task Definition (建立新任務定義)，以及 Create new revision with JSON (使用 JSON 建立新修訂版)。
3. 複製下列任務定義範例並貼到方塊中，然後選擇 Save (儲存)。

```
{
    "family": "sample-fargate",
    "networkMode": "awsvpc",
    "containerDefinitions": [
        {
            "name": "fargate-app",
            "image": "public.ecr.aws/docker/library/httpd:latest",
            "portMappings": [
                {
                    "containerPort": 80,
                    "hostPort": 80,
                    "protocol": "tcp"
                }
            ],
            "essential": true,
            "entryPoint": [
                "sh",
  "-c"
            ],
            "command": [
                "/bin/sh -c \"echo '<html> <head> <title>Amazon ECS Sample
 App</title> <style>body {margin-top: 40px; background-color: #333;} </style> </
head><body> <div style=color:white;text-align:center> <h1>Amazon ECS Sample App</
h1> <h2>Congratulations!</h2> <p>Your application is now running on a container in
 Amazon ECS.</p> </div></body></html>' >  /usr/local/apache2/htdocs/index.html &&
 httpd-foreground\""
            ]
        }
    ],
```

```
    "requiresCompatibilities": [
        "FARGATE"
    ],
    "cpu": "256",
    "memory": "512"
}
```

4. 選擇 Create (建立)。

步驟 3：建立服務

使用任務定義來建立服務。

1. 在導覽窗格中，選擇 Clusters (叢集)，然後選取您在 步驟 1：建立叢集 中建立的叢集。
2. 在 Services (服務) 索引標籤上，選擇 Create (建立)。
3. 在 Deployment configuration (部署組態)，指定應用程式的部署方式。
   a. 在 Task Definitions (任務定義) 中，選擇您在 步驟 2：建立任務定義 中建立的任務定義。
   b. 針對 Service name (服務名稱)，輸入服務的名稱。
   c. 在 Desired tasks (所需任務) 中，請輸入 1。
4. 在 Networking (聯網) 下，您可以為您的任務建立新的安全群組或選擇現有的安全群組。請確定您使用的安全群組已在 必要條件 下方列出傳入規則。
5. 選擇 Create (建立)。

步驟 4：檢視服務

1. 開啟主控台，網址為 https://console.aws.amazon.com/ecs/v2。
2. 在導覽窗格中，選擇叢集。
3. 選擇您執行服務的叢集。
4. 在 Services (服務) 索引標籤中，Service name (服務名稱) 下，選擇您在 步驟 3：建立服務 中建立的服務。
5. 選擇 Tasks (任務) 索引標籤，然後選擇服務中的任務。
6. 在任務頁面 Configuration (組態) 區段的 Public IP (公有 IP) 中，選擇 Open address (開放地址)。

步驟 5：清除

完成使用 Amazon ECS 叢集時，您應該清除與其相關的資源，以免未使用的資源產生費用。

某些 Amazon ECS 資源 (例如任務、服務、叢集和容器執行個體) 可使用 Amazon ECS 主控台進行清除。其他資源，例如 Amazon EC2 執行個體、Elastic Load Balancing 負載平衡器和 Auto Scaling 群組，必須在 Amazon EC2 主控台中手動清除或刪除建立它們的 AWS CloudFormation 堆疊。

1. 在導覽窗格中，選擇叢集。
2. 在 Clusters (叢集) 頁面，選取您為本教學課程建立的叢集。
3. 選擇 Services (服務) 索引標籤。
4. 選取服務，然後選擇 Delete (刪除)。
5. 在確認提示中，輸入 delete (刪除)，然後選擇 Delete (刪除)。或者，您可以使用 `Force delete`選項，讓 Amazon ECS 在刪除服務之前代您縮減服務規模。

   等到刪除該服務。
6. 選擇 Delete Cluster (刪除叢集)。在確認提示中，輸入 delete ***`cluster-name`***，然後選擇 Delete (刪除)。刪除叢集會清除與叢集一起建立的相關聯資源，包括 Auto Scaling 群組、VPC 或負載平衡器。

了解如何為 Fargate 啟動類型建立 Amazon ECS Windows 任務

在 Amazon ECS 支援 AWS Fargate Fargate 的 區域中，使用 AWS Fargate 啟動類型的任務，在 上開始使用 Amazon ECS。

完成下列步驟，以在 AWS Fargate上開始使用 Amazon ECS。

必要條件

開始之前，請完成 中的步驟，設定以使用 Amazon ECS。 以及您的 AWS 使用者具有 IAM `AdministratorAccess` 政策範例中指定的許可。

主控台會嘗試自動建立任務執行 IAM 角色，該角色是 Fargate 任務的必要項目。若要確保主控台可成功建立此 IAM 角色，下列其中一項必須為 True：

- 您的使用者具有管理員存取。如需詳細資訊，請參閱設定以使用 Amazon ECS。 。
- 您的使用者具有建立服務角色的 IAM 許可。如需詳細資訊，請參閱建立角色以委派許可給 AWS 服務。

- 具備管理員存取的使用者已手動建立任務執行角色，讓其可在帳戶上提供使用。如需詳細資訊，請參閱Amazon ECS 任務執行 IAM 角色。

> **Important**
>
> 使用任務定義建立服務時選取的安全群組，必須為入站流量開放連接埠 80。將下列傳入規則新增至安全群組。如需有關如何建立安全群組的資訊，請參閱《Amazon EC2 使用者指南》中的為您的 Amazon EC2 執行個體建立安全群組。 Amazon EC2
>
> - Type (類型)：HTTP
> - Protocol (通訊協定)：TCP
> - 連接埠範圍：80
> - 來源：Anywhere (`0.0.0.0/0`)

步驟 1：建立叢集

您可以建立使用預設 VPC 且名為 windows 的全新叢集。

使用 建立叢集 AWS Management Console

1. 開啟主控台，網址為 https://console.aws.amazon.com/ecs/v2。
2. 從導覽列中選取要使用的「區域」。
3. 在導覽窗格中，選擇叢集。
4. 在 Clusters (叢集) 頁面上，選擇 Create cluster (建立叢集)。
5. 在 Cluster configuration (叢集組態) 下的 Cluster name (叢集名稱) 中，輸入 windows。
6. (選用) 若要開啟 Container Insights，請展開 Monitoring (監控)，然後開啟 Use Container Insights (使用 Container Insights)。
7. (選用) 為協助識別您的叢集，請展開 Tags (標籤)，然後設定標籤。

   [新增標籤] 選擇新增標籤，並執行下列動作：

   - 對於 Key (金鑰)，輸入金鑰名稱。
   - 對於 Value (值)，進入金鑰值。

   [移除標籤] 選擇標籤「金鑰」和「值」右側的移除。

8. 選擇 Create (建立)。

步驟 2：註冊 Windows 任務定義

您必須先註冊任務定義，才能在您的 Amazon ECS 叢集中執行 Windows 容器。以下任務定義範例會在具有 `mcr.microsoft.com/windows/servercore/iis` 容器映像之容器執行個體的連接埠 8080 顯示一個簡單的網頁。

向 註冊範例任務定義 AWS Management Console

1. 在導覽窗格中，選擇 Task Definitions (任務定義)。
2. 選擇 Create new task definitio (建立新任務定義)、Create new task definition with JSON (使用 JSON 建立新的任務定義)。
3. 複製下列任務定義範例並貼到方塊中，然後選擇 Save (儲存)。

```
{
    "containerDefinitions": [
        {
            "command": ["New-Item -Path C:\\inetpub\\wwwroot\\index.html
 -Type file -Value '<html> <head> <title>Amazon ECS Sample App</title>
 <style>body {margin-top: 40px; background-color: #333;} </style> </head><body>
 <div style=color:white;text-align:center> <h1>Amazon ECS Sample App</h1>
 <h2>Congratulations!</h2> <p>Your application is now running on a container in
 Amazon ECS.</p>'; C:\\ServiceMonitor.exe w3svc"],
            "entryPoint": [
                "powershell",
                "-Command"
            ],
            "essential": true,
            "cpu": 2048,
            "memory": 4096,
            "image": "mcr.microsoft.com/windows/servercore/iis:windowsservercore-
ltsc2019",
            "name": "sample_windows_app",
            "portMappings": [
                {
                    "hostPort": 80,
                    "containerPort": 80,
                    "protocol": "tcp"
                }
            ]
```

```
        }
    ],
    "memory": "4096",
    "cpu": "2048",
    "networkMode": "awsvpc",
    "family": "windows-simple-iis-2019-core",
    "executionRoleArn": "arn:aws:iam::012345678910:role/ecsTaskExecutionRole",
    "runtimePlatform": {"operatingSystemFamily": "WINDOWS_SERVER_2019_CORE"},
    "requiresCompatibilities": ["FARGATE"]
}
```

4. 驗證您的資訊，然後選擇 Create (建立)。

步驟 3：以您的任務定義建立服務

註冊您的任務定義之後，即可用以放置任務到您的叢集中。下列程序會以您的任務定義建立一項服務，並在您的叢集放置一項任務。

使用主控台從您的任務定義建立服務

1. 在導覽窗格中，選擇 Clusters (叢集)，然後選取您在 步驟 1：建立叢集 中建立的叢集。
2. 在 Services (服務) 索引標籤上，選擇 Create (建立)。
3. 在 Deployment configuration (部署組態)，指定應用程式的部署方式。
   a. 在 Task Definitions (任務定義) 中，選擇您在 步驟 2：註冊 Windows 任務定義 中建立的任務定義。
   b. 針對 Service name (服務名稱)，輸入服務的名稱。
   c. 在 Desired tasks (所需任務) 中，請輸入 1。
4. 在 Networking (聯網) 下，您可以建立新的安全群組，或選擇現有的安全群組。請確定您使用的安全群組已在 必要條件 下方列出傳入規則。
5. 選擇 Create (建立)。

步驟 4：檢視服務

您的服務已於您的叢集中啟動任務後，您可以在瀏覽器中檢視服務並開啟 IIS 測試頁面，確認容器正在執行。

> **Note**
>
> 您的容器執行個體約需 15 分鐘來下載和解壓縮 Windows 容器基礎層。

檢視服務

1. 開啟主控台，網址為 https://console.aws.amazon.com/ecs/v2。
2. 在導覽窗格中，選擇叢集。
3. 選擇您執行服務的叢集。
4. 在 Services (服務) 索引標籤中，Service name (服務名稱) 下，選擇您在 步驟 3：以您的任務定義建立服務 中建立的服務。
5. 選擇 Tasks (任務) 索引標籤，然後選擇服務中的任務。
6. 在任務頁面 Configuration (組態) 區段的 Public IP (公有 IP) 中，選擇 Open address (開放地址)。

步驟 5：清除

完成使用 Amazon ECS 叢集時，您應該清除與其相關的資源，以免未使用的資源產生費用。

某些 Amazon ECS 資源 (例如任務、服務、叢集和容器執行個體) 可使用 Amazon ECS 主控台進行清除。其他資源，例如 Amazon EC2 執行個體、Elastic Load Balancing 負載平衡器和 Auto Scaling 群組，必須在 Amazon EC2 主控台中手動清除或刪除建立它們的 AWS CloudFormation 堆疊。

1. 在導覽窗格中，選擇叢集。
2. 在 Clusters (叢集) 頁面，選取您為本教學課程建立的叢集。
3. 選擇 Services (服務) 索引標籤。
4. 選取服務，然後選擇 Delete (刪除)。
5. 在確認提示中，輸入 delete (刪除)，然後選擇 Delete (刪除)。

   等到刪除該服務。
6. 選擇 Delete Cluster (刪除叢集)。在確認提示中，輸入 delete ***cluster-name***，然後選擇 Delete (刪除)。刪除叢集會清除與叢集一起建立的相關聯資源，包括 Auto Scaling 群組、VPC 或負載平衡器。

了解如何為 EC2 啟動類型建立 Amazon ECS Windows 任務

透過註冊任務定義、建立叢集以及在主控台中建立服務，使用 EC2 啟動類型開始使用 Amazon ECS。

完成下列步驟，以使用 EC2 啟動類型來開始使用 Amazon ECS。

必要條件

開始之前，請完成 中的步驟，設定以使用 Amazon ECS。 以及您的 AWS 使用者具有 IAM `AdministratorAccess` 政策範例中指定的許可。

主控台會嘗試自動建立任務執行 IAM 角色，該角色是 Fargate 任務的必要項目。若要確保主控台可成功建立此 IAM 角色，下列其中一項必須為 True：

- 您的使用者具有管理員存取。如需詳細資訊，請參閱設定以使用 Amazon ECS。 。
- 您的使用者具有建立服務角色的 IAM 許可。如需詳細資訊，請參閱建立角色以委派許可給 AWS 服務。
- 具備管理員存取的使用者已手動建立任務執行角色，讓其可在帳戶上提供使用。如需詳細資訊，請參閱Amazon ECS 任務執行 IAM 角色。

> **Important**
>
> 使用任務定義建立服務時選取的安全群組，必須為入站流量開放連接埠 80。將下列傳入規則新增至安全群組。如需如何建立安全群組的詳細資訊，請參閱《Amazon EC2 使用者指南》中的為您的 Amazon EC2 執行個體建立安全群組。 Amazon EC2
>
> - Type (類型)：HTTP
> - Protocol (通訊協定)：TCP
> - 連接埠範圍：80
> - 來源：Anywhere (`0.0.0.0/0`)

步驟 1：建立叢集

Amazon ECS 叢集是任務、服務和容器執行個體的邏輯分組。

下列步驟將逐步引導您使用一個已註冊至叢集的 Amazon EC2 執行個體來建立叢集，以便讓我們在該叢集上執行任務。若未提及特定欄位，請保留預設主控台值。

建立新叢集 (Amazon ECS 主控台)

在開始之前，請指派適當的 IAM 許可。如需詳細資訊，請參閱the section called “Amazon ECS 叢集範例”。

1. 開啟主控台，網址為 https://console.aws.amazon.com/ecs/v2。
2. 從導覽列中選取要使用的「區域」。
3. 在導覽窗格中，選擇叢集。
4. 在 Clusters (叢集) 頁面上，選擇 Create cluster (建立叢集)。
5. 在 Cluster configuration (叢集組態) 下的 Cluster name (叢集名稱) 中，輸入唯一的名稱。

   名稱可以包含最多 255 個字母 (大小寫)、數字與連字號。
6. (選用) 若要變更任務和服務啟動所在的 VPC 和子網路，請在 Networking (聯網) 下，執行下列任一操作：

   - 若要移除子網路，請在 Subnets (子網路) 下，對您要移除之每一個子網路選擇 X。
   - 若要變更為非 default (預設) VPC，請在 VPC 下，選擇現有的 VPC，然後在 Subnets (子網路) 下選擇各個子網路。
7. 若要將 Amazon EC2 執行個體新增至叢集，展開 Infrastructure (基礎設施)，然後選取 Amazon EC2 執行個體。接下來，設定作為容量提供者的 Auto Scaling 群組：

   a. 要使用現有 Auto Scaling 群組，請從 Auto Scaling group (ASG) (Auto Scaling 群組 (ASG)) 中選取該群組。

   b. 若要建立 Auto Scaling 群組，請從 Auto Scaling group (ASG) (Auto Scaling 群組 (ASG)) 中選取 Create new group (建立新群組)，然後提供有關該群組的下列詳細資訊：

      - 針對 Operating system/Architecture (作業系統/架構)，為 Auto Scaling 群組執行個體選擇 Amazon ECS 最佳化 AMI。
      - 對於 EC2 instance type (EC2 執行個體類型)，選擇適合您工作負載的執行個體類型。如需不同執行個體類型的詳細資訊，請參閱 Amazon EC2 執行個體。

        如果 Auto Scaling 群組使用相同或類似的執行個體類型，則受管擴展效果最佳。
      - 對於 SSH key pair (SSH 金鑰對)，選擇在連線到執行個體時證明您身分的金鑰對。
      - 對於 Capacity (容量)，輸入 Auto Scaling 群組中要啟動的最小執行個體數和最大執行個體數。Amazon EC2 執行個體在您的 AWS 資源中存在時會產生成本。如需詳細資訊，請參閱 Amazon EC2 定價。

8. (選用) 若要開啟 Container Insights，請展開 Monitoring (監控)，然後開啟 Use Container Insights (使用 Container Insights)。
9. (選用) 若要管理叢集標籤，請展開 Tags (標籤)，然後執行下列其中一項操作：

   [新增標籤] 選擇新增標籤，並執行下列動作：

   - 對於 Key (金鑰)，輸入金鑰名稱。
   - 對於 Value (值)，進入金鑰值。

   [移除標籤] 選擇標籤「金鑰」和「值」右側的移除。
10. 選擇建立。

步驟 2：註冊任務定義

向 註冊範例任務定義 AWS Management Console

1. 在導覽窗格中，選擇 Task Definitions (任務定義)。
2. 選擇 Create new task definitio (建立新任務定義)、Create new task definition with JSON (使用 JSON 建立新的任務定義)。
3. 複製下列任務定義範例並貼到方塊中，然後選擇 Save (儲存)。

```
{
    "containerDefinitions": [
        {
            "command": ["New-Item -Path C:\\inetpub\\wwwroot\\index.html
 -Type file -Value '<html> <head> <title>Amazon ECS Sample App</title>
 <style>body {margin-top: 40px; background-color: #333;} </style> </head><body>
 <div style=color:white;text-align:center> <h1>Amazon ECS Sample App</h1>
 <h2>Congratulations!</h2> <p>Your application is now running on a container in
 Amazon ECS.</p>'; C:\\ServiceMonitor.exe w3svc"],
            "entryPoint": [
                "powershell",
                "-Command"
            ],
            "essential": true,
            "cpu": 2048,
            "memory": 4096,
            "image": "mcr.microsoft.com/windows/servercore/iis:windowsservercore-
ltsc2019",
```

```
            "name": "sample_windows_app",
            "portMappings": [
                {
                    "hostPort": 443,
                    "containerPort": 80,
                    "protocol": "tcp"
                }
            ]
        }
    ],
    "memory": "4096",
    "cpu": "2048",
    "family": "windows-simple-iis-2019-core",
    "executionRoleArn": "arn:aws:iam::012345678910:role/ecsTaskExecutionRole",
    "runtimePlatform": {"operatingSystemFamily": "WINDOWS_SERVER_2019_CORE"},
    "requiresCompatibilities": ["EC2"]
}
```

4. 驗證您的資訊，然後選擇 Create (建立)。

步驟 3：建立服務

Amazon ECS 服務可協助您在 Amazon ECS 叢集中同時執行並維持指定數目的任務定義執行個體。如果您的有任務因為任何原因而故障或停止，Amazon ECS 服務排程器就會啟動任務定義的另一個執行個體取代之，以維護服務中所需的任務數量。如需服務的詳細資訊，請參閱 Amazon ECS 服務。

建立服務

1. 在導覽窗格中，選擇叢集。
2. 選取您在 步驟 1：建立叢集 中建立的叢集。
3. 在 Services (服務) 標籤上，選擇 Create (建立)。
4. 在 Environment (環境) 區段中，執行以下動作：
   a. 在 Compute options (運算選項) 中，選擇 Launch type (啟動類型)。
   b. 針對 Launch type (啟動類型)，選取 EC2
5. 在 Deployment configuration (部署組態) 區段中，執行以下操作：
   a. 在 Family (系列) 中，選擇您在 步驟 2：註冊任務定義 中建立的任務定義。
   b. 針對 Service name (服務名稱)，輸入服務的名稱。

c. 在 Desired tasks (所需任務) 中，請輸入 1。

6. 檢閱選項，然後選擇建立。
7. 選擇 View service (檢視服務) 以檢閱服務。

步驟 4：檢視服務

服務是一種 Web 應用程式，您可以使用 Web 瀏覽器檢視其容器。

1. 開啟主控台，網址為 https://console.aws.amazon.com/ecs/v2。
2. 在導覽窗格中，選擇叢集。
3. 選擇您執行服務的叢集。
4. 在 Services (服務) 索引標籤中，Service name (服務名稱) 下，選擇您在 步驟 3：建立服務 中建立的服務。
5. 選擇 Tasks (任務) 索引標籤，然後選擇服務中的任務。
6. 在任務頁面 Configuration (組態) 區段的 Public IP (公有 IP) 中，選擇 Open address (開放地址)。下面的螢幕擷取畫面是預期的輸出。

步驟 5：清除

完成使用 Amazon ECS 叢集時，您應該清除與其相關的資源，以免未使用的資源產生費用。

某些 Amazon ECS 資源 (例如任務、服務、叢集和容器執行個體) 可使用 Amazon ECS 主控台進行清除。其他資源，例如 Amazon EC2 執行個體、Elastic Load Balancing 負載平衡器和 Auto Scaling 群組，必須在 Amazon EC2 主控台中手動清除或刪除建立它們的 AWS CloudFormation 堆疊。

1. 在導覽窗格中，選擇叢集。

2. 在 Clusters (叢集) 頁面，選取您為本教學課程建立的叢集叢集。
3. 選擇 Services (服務) 索引標籤。
4. 選取服務，然後選擇 Delete (刪除)。
5. 在確認提示中，輸入 delete (刪除)，然後選擇 Delete (刪除)。

   等到刪除該服務。
6. 選擇 Delete Cluster (刪除叢集)。在確認提示中，輸入 delete ***cluster-name***，然後選擇 Delete (刪除)。刪除叢集會清除與叢集一起建立的相關聯資源，包括 Auto Scaling 群組、VPC 或負載平衡器。

使用 建立 Amazon ECS 資源 AWS CDK

AWS Cloud Development Kit (AWS CDK) 是 Infrastructure-as-Code (IAC) 架構，您可以使用您選擇的程式設計語言來定義 AWS 雲端基礎設施。若要定義您自己的雲端基礎設施，要先編寫包含一個或更多堆疊的應用程式 (使用 CDK 支援的其中一種語言)。然後，您將它合成到 AWS CloudFormation 範本，並將您的資源部署到 AWS 帳戶。請依照本主題中的步驟，使用 Amazon Elastic Container Service (Amazon ECS) 和 Fargate AWS CDK 上的 部署容器化 Web 伺服器。

隨附於 CDK 的 AWS 建構程式庫提供模組，可用於建立 AWS 服務 所提供資源的模型。針對熱門服務，程式庫會提供具有智能預設和最佳實務的彙整建構。其中的模組，特別是 `aws-ecs-patterns`，提供了高階抽象概念，讓您可以僅憑幾行程式碼行定義您的容器化服務和所有必要的支援資源。

本主題使用 `ApplicationLoadBalancedFargateService` 建構。此建構會在 Fargate 上一個 Application Load Balancer 的後方部署 Amazon ECS 服務。`aws-ecs-patterns` 模組還包含採用 Network Load Balancer 並在 Amazon EC2 上執行的建構。

開始此任務之前，請先設定您的 AWS CDK 開發環境，然後執行下列命令 AWS CDK 來安裝 。如需如何設定開發 AWS CDK 環境的說明，請參閱 AWS CDK - 先決條件入門。

```
npm install -g aws-cdk
```

> **Note**
>
> 這些說明假設您使用 AWS CDK v2。

主題

步驟 1：設定您的 AWS CDK 專案

為您的新 AWS CDK 應用程式建立目錄並初始化專案。

TypeScript

```
mkdir hello-ecs
cd hello-ecs
cdk init --language typescript
```

JavaScript

```
mkdir hello-ecs
cd hello-ecs
cdk init --language javascript
```

Python

```
mkdir hello-ecs
cd hello-ecs
cdk init --language python
```

專案啟動後，請啟用專案的虛擬環境，並安裝 AWS CDK的基準相依性。

```
source .venv/bin/activate
python -m pip install -r requirements.txt
```

Java

```
mkdir hello-ecs
cd hello-ecs
cdk init --language java
```

將此 Maven 專案匯入您的 Java IDE。例如，在 Eclipse 中，使用檔案 > 匯入 > Maven > 現有的 Maven 專案。

C#

```
mkdir hello-ecs
cd hello-ecs
cdk init --language csharp
```

Go

```
mkdir hello-ecs
cd hello-ecs
cdk init --language go
```

> **Note**
>
> AWS CDK 應用程式範本使用專案目錄的名稱來產生來源檔案和類別的名稱。在此範例中，目錄名為 `hello-ecs`。若使用不同的專案目錄名稱，您的應用程式將與這些說明不相符。

AWS CDK v2 包含單一套件 AWS 服務 中所有 的穩定建構，稱為 `aws-cdk-lib`。在初始化專案時，此套件會安裝為相依性套件。使用某些程式設計語言時，套件會在您第一次建置專案時安裝。本主題涵蓋了如何使用 Amazon ECS 模式建構，該建構提供了用於 Amazon ECS 的高階抽象概念。此模組依靠 Amazon ECS 建構和其他建構，來佈建您 Amazon ECS 應用程式所需的來源。

用於將這些程式庫匯入 CDK 應用程式的名稱可能會略有不同，具體取決於您使用的程式設計語言。以下是在每種支援的 CDK 程式設計語言中使用的名稱，供您參考。

TypeScript

```
aws-cdk-lib/aws-ecs
aws-cdk-lib/aws-ecs-patterns
```

JavaScript

```
aws-cdk-lib/aws-ecs
aws-cdk-lib/aws-ecs-patterns
```

Python

```
aws_cdk.aws_ecs
aws_cdk.aws_ecs_patterns
```

Java

```
software.amazon.awscdk.services.ecs
software.amazon.awscdk.services.ecs.patterns
```

C#

```
Amazon.CDK.AWS.ECS
Amazon.CDK.AWS.ECS.Patterns
```

Go

```
github.com/aws/aws-cdk-go/awscdk/v2/awsecs
github.com/aws/aws-cdk-go/awscdk/v2/awsecspatterns
```

步驟 2：使用 AWS CDK 在 Fargate 上定義容器化 Web 伺服器

使用來自 DockerHub 的容器映像 amazon-ecs-sample。此映像包含在 Amazon Linux 2 上執行的 PHP Web 應用程式。

在您建立的 AWS CDK 專案中，編輯包含堆疊定義的檔案，以類似下列其中一個範例。

> **Note**
>
> 堆疊是部署的單位。所有資源必須在同一堆疊中，而一個堆疊中的所有資源都要部署在一起。若某資源無法部署，任何已部署的其他資源將會復原。 AWS CDK 應用程式可以包含多個堆疊，而一個堆疊中的資源可以參考另一個堆疊中的資源。

TypeScript

更新 `lib/hello-ecs-stack.ts` 使其相似於以下內容。

```
import * as cdk from 'aws-cdk-lib';
import { Construct } from 'constructs';
```

```
import * as ecs from 'aws-cdk-lib/aws-ecs';
import * as ecsp from 'aws-cdk-lib/aws-ecs-patterns';

export class HelloEcsStack extends cdk.Stack {
  constructor(scope: Construct, id: string, props?: cdk.StackProps) {
    super(scope, id, props);

    new ecsp.ApplicationLoadBalancedFargateService(this, 'MyWebServer', {
      taskImageOptions: {
        image: ecs.ContainerImage.fromRegistry('amazon/amazon-ecs-sample'),
      },
      publicLoadBalancer: true
    });
  }
}
```

JavaScript

更新 `lib/hello-ecs-stack.js` 使其相似於以下內容。

```
const cdk = require('aws-cdk-lib');
const { Construct } = require('constructs');
const ecs = require('aws-cdk-lib/aws-ecs');
const ecsp = require('aws-cdk-lib/aws-ecs-patterns');

class HelloEcsStack extends cdk.Stack {
  constructor(scope = Construct, id = string, props = cdk.StackProps) {
    super(scope, id, props);

    new ecsp.ApplicationLoadBalancedFargateService(this, 'MyWebServer', {
      taskImageOptions: {
        image: ecs.ContainerImage.fromRegistry('amazon/amazon-ecs-sample'),
      },
      publicLoadBalancer: true
    });
  }
}

module.exports = { HelloEcsStack }
```

Python

更新 `hello-ecs/hello_ecs_stack.py` 使其相似於以下內容。

```
import aws_cdk as cdk
from constructs import Construct

import aws_cdk.aws_ecs as ecs
import aws_cdk.aws_ecs_patterns as ecsp

class HelloEcsStack(cdk.Stack):

    def __init__(self, scope: Construct, construct_id: str, **kwargs) -> None:
        super().__init__(scope, construct_id, **kwargs)

        ecsp.ApplicationLoadBalancedFargateService(self, "MyWebServer",
            task_image_options=ecsp.ApplicationLoadBalancedTaskImageOptions(
                image=ecs.ContainerImage.from_registry("amazon/amazon-ecs-sample")),
            public_load_balancer=True
        )
```

Java

更新 src/main/java/com.myorg/HelloEcsStack.java 使其相似於以下內容。

```
package com.myorg;

import software.constructs.Construct;
import software.amazon.awscdk.Stack;
import software.amazon.awscdk.StackProps;

import software.amazon.awscdk.services.ecs.ContainerImage;
import
 software.amazon.awscdk.services.ecs.patterns.ApplicationLoadBalancedFargateService;
import
 software.amazon.awscdk.services.ecs.patterns.ApplicationLoadBalancedTaskImageOptions;

public class HelloEcsStack extends Stack {
    public HelloEcsStack(final Construct scope, final String id) {
        this(scope, id, null);
    }

    public HelloEcsStack(final Construct scope, final String id, final StackProps
 props) {
        super(scope, id, props);

        ApplicationLoadBalancedFargateService.Builder.create(this, "MyWebServer")
```

```
          .taskImageOptions(ApplicationLoadBalancedTaskImageOptions.builder()
            .image(ContainerImage.fromRegistry("amazon/amazon-ecs-sample"))
            .build())
          .publicLoadBalancer(true)
          .build();
    }
}
```

C#

更新 src/HelloEcs/HelloEcsStack.cs 使其相似於以下內容。

```
using Amazon.CDK;
using Constructs;
using Amazon.CDK.AWS.ECS;
using Amazon.CDK.AWS.ECS.Patterns;
namespace HelloEcs
{
    public class HelloEcsStack : Stack
    {
        internal HelloEcsStack(Construct scope, string id, IStackProps props =
 null) : base(scope, id, props)
        {
            new ApplicationLoadBalancedFargateService(this, "MyWebServer",
                new ApplicationLoadBalancedFargateServiceProps
                {
                    TaskImageOptions = new ApplicationLoadBalancedTaskImageOptions
                    {
                        Image = ContainerImage.FromRegistry("amazon/amazon-ecs-
sample")
                    },
                    PublicLoadBalancer = true
                });
        }
    }
}
```

Go

更新 hello-ecs.go 使其相似於以下內容。

```
package main
```

```
import (
 "github.com/aws/aws-cdk-go/awscdk/v2"
 // "github.com/aws/aws-cdk-go/awscdk/v2/awssqs"
 "github.com/aws/aws-cdk-go/awscdk/v2/awsecs"
 "github.com/aws/aws-cdk-go/awscdk/v2/awsecspatterns"
 "github.com/aws/constructs-go/constructs/v10"
 "github.com/aws/jsii-runtime-go"
)

type HelloEcsStackProps struct {
 awscdk.StackProps
}

func NewHelloEcsStack(scope constructs.Construct, id string, props
 *HelloEcsStackProps) awscdk.Stack {
 var sprops awscdk.StackProps
 if props != nil {
  sprops = props.StackProps
 }
 stack := awscdk.NewStack(scope, &id, &sprops)

 // The code that defines your stack goes here

 // example resource
 // queue := awssqs.NewQueue(stack, jsii.String("HelloEcsQueue"),
 &awssqs.QueueProps{
 //  VisibilityTimeout: awscdk.Duration_Seconds(jsii.Number(300)),
 // })
 res := awsecspatterns.NewApplicationLoadBalancedFargateService(stack,
 jsii.String("MyWebServer"),
  &awsecspatterns.ApplicationLoadBalancedFargateServiceProps{
   TaskImageOptions: &awsecspatterns.ApplicationLoadBalancedTaskImageOptions{
    Image: awsecs.ContainerImage_FromRegistry(jsii.String("amazon/amazon-ecs-
sample"), &awsecs.RepositoryImageProps{}),
   },
  },
 )
 awscdk.NewCfnOutput(stack, jsii.String("LoadBalancerDNS"),
 &awscdk.CfnOutputProps{Value: res.LoadBalancer().LoadBalancerDnsName()})

 return stack
}

func main() {
```

```
 defer jsii.Close()

 app := awscdk.NewApp(nil)

 NewHelloEcsStack(app, "HelloEcsStack", &HelloEcsStackProps{
  awscdk.StackProps{
   Env: env(),
  },
 })

 app.Synth(nil)
}

// env determines the AWS environment (account+region) in which our stack is to
// be deployed. For more information see: https://docs.aws.amazon.com/cdk/latest/
guide/environments.html
func env() *awscdk.Environment {
 // If unspecified, this stack will be "environment-agnostic".
 // Account/Region-dependent features and context lookups will not work, but a
 // single synthesized template can be deployed anywhere.
 //---------------------------------------------------------------------------
 return nil

 // Uncomment if you know exactly what account and region you want to deploy
 // the stack to. This is the recommendation for production stacks.
 //---------------------------------------------------------------------------
 // return &awscdk.Environment{
 //  Account: jsii.String("123456789012"),
 //  Region:  jsii.String("us-east-1"),
 // }

 // Uncomment to specialize this stack for the AWS Account and Region that are
 // implied by the current CLI configuration. This is recommended for dev
 // stacks.
 //---------------------------------------------------------------------------
 // return &awscdk.Environment{
 //  Account: jsii.String(os.Getenv("CDK_DEFAULT_ACCOUNT")),
 //  Region:  jsii.String(os.Getenv("CDK_DEFAULT_REGION")),
 // }
}
```

前面的簡短程式碼片段包括以下內容：

- 服務的邏輯名稱：MyWebServer。
- 從 DockerHub 取得的容器映像：amazon/amazon-ecs-sample。
- 其他相關資訊，例如負載平衡器具有公有地址，而且可以從網際網路存取的事實。

AWS CDK 將建立部署 Web 伺服器所需的所有資源，包括下列資源。此範例中省略了這些資源。

- Amazon ECS 叢集
- Amazon VPC 和 Amazon EC2 執行個體
- Auto Scaling 群組
- Application Load Balancer
- (IAM) 角色和政策

部分自動佈建的資源是由堆疊中定義的全部 Amazon ECS 服務進行共享。

保存來源檔案，然後在應用程式的主目錄中執行 cdk synth 指令。會 AWS CDK 執行應用程式並從中合成 AWS CloudFormation 範本，然後顯示範本。範本是大約 600 行的 YAML 檔案。檔案的開頭如此處所示。您的範本可能與此範例不同。

```
Resources:
  MyWebServerLB3B5FD3AB:
    Type: AWS::ElasticLoadBalancingV2::LoadBalancer
    Properties:
      LoadBalancerAttributes:
        - Key: deletion_protection.enabled
          Value: "false"
      Scheme: internet-facing
      SecurityGroups:
        - Fn::GetAtt:
            - MyWebServerLBSecurityGroup01B285AA
            - GroupId
      Subnets:
        - Ref: EcsDefaultClusterMnL3mNNYNVpcPublicSubnet1Subnet3C273B99
        - Ref: EcsDefaultClusterMnL3mNNYNVpcPublicSubnet2Subnet95FF715A
      Type: application
    DependsOn:
      - EcsDefaultClusterMnL3mNNYNVpcPublicSubnet1DefaultRouteFF4E2178
      - EcsDefaultClusterMnL3mNNYNVpcPublicSubnet2DefaultRouteB1375520
    Metadata:
```

```
      aws:cdk:path: HelloEcsStack/MyWebServer/LB/Resource
  MyWebServerLBSecurityGroup01B285AA:
    Type: AWS::EC2::SecurityGroup
    Properties:
      GroupDescription: Automatically created Security Group for ELB
 HelloEcsStackMyWebServerLB06757F57
      SecurityGroupIngress:
        - CidrIp: 0.0.0.0/0
          Description: Allow from anyone on port 80
          FromPort: 80
          IpProtocol: tcp
          ToPort: 80
      VpcId:
        Ref: EcsDefaultClusterMnL3mNNYNVpc7788A521
    Metadata:
      aws:cdk:path: HelloEcsStack/MyWebServer/LB/SecurityGroup/Resource
# and so on for another few hundred lines
```

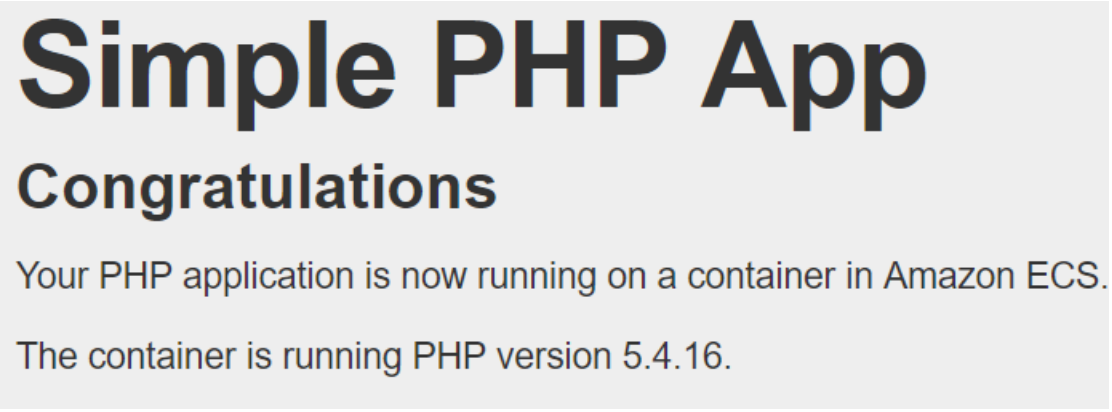
若要在 中部署 服務 AWS 帳戶，請在應用程式的主目錄中執行 cdk deploy命令。您需要核准 AWS CDK 產生的 IAM 政策。

部署需要幾分鐘的時間，在此期間 會 AWS CDK 建立數個資源。部署輸出的最後幾行包含負載平衡器的公開主機名稱，和您的新 Web 伺服器 URL。如下所示：

```
Outputs:
HelloEcsStack.MyWebServerLoadBalancerDNSXXXXXXX = Hello-MyWeb-ZZZZZZZZZZZZZ-
ZZZZZZZZZZ.us-west-2.elb.amazonaws.com
HelloEcsStack.MyWebServerServiceURLYYYYYYYY = http://Hello-MyWeb-ZZZZZZZZZZZZZ-
ZZZZZZZZZZ.us-west-2.elb.amazonaws.com
```

步驟 3：測試 Web 伺服器

從部署輸出複製 URL，並貼到您的 Web 瀏覽器。此時會顯示下列來自 Web 伺服器的歡迎使用訊息。



步驟 4：清理

在您完成使用 Web 伺服器之後，於應用程式的主目錄中執行 `cdk destroy` 指令來用 CDK 結束服務。這樣做可以防止您未來意外產生任何費用。

後續步驟

若要進一步了解如何使用 開發 AWS 基礎設施 AWS CDK，請參閱 AWS CDK 開發人員指南。

如需以您選擇的語言撰寫 AWS CDK 應用程式的資訊，請參閱以下內容：

TypeScript

- 以 TypeScript 使用 AWS CDK

JavaScript

- 以 JavaScript 使用 AWS CDK

Python

- 在 Python AWS CDK 中使用

Java

- 在 Java AWS CDK 中使用

C#

- 在 C# AWS CDK 中使用

Go

- 在 Go AWS CDK 中使用

如需本主題中使用的 AWS 建構程式庫模組的詳細資訊，請參閱下列 AWS CDK API 參考概觀。

- aws-ecs
- aws-ecs-patterns

使用 建立 Amazon ECS 資源 AWS CloudFormation

Amazon ECS 已與 整合 AWS CloudFormation，這項服務可讓您使用定義的範本來建立和設定 AWS 資源的模型。如此一來，您可以花更少的時間建立並管理資源和基礎設施。使用 AWS CloudFormation，您可以建立範本來描述您想要的所有 AWS 資源，例如特定的 Amazon ECS 叢集。然後， AWS CloudFormation 會負責佈建和設定這些資源。

使用 時 AWS CloudFormation，您可以重複使用範本，以一致且可重複的方式設定 Amazon ECS 資源。您會描述您的資源一次，然後在多個 AWS 帳戶 和 之間再次佈建相同的資源 AWS 區域。

AWS CloudFormation 範本

若要為 Amazon ECS 和相關服務佈建和設定資源，請確定您熟悉 AWS CloudFormation template. AWS CloudFormation templates 是 JSON 或 YAML 格式的文字檔案，其描述您想要在 AWS CloudFormation 堆疊中佈建的資源。如果您不熟悉 JSON 或 YAML 格式，或兩者都熟悉，您可以使用 AWS CloudFormation 設計工具開始使用 AWS CloudFormation 範本。如需詳細資訊，請參閱AWS CloudFormation 《 使用者指南》中的什麼是 AWS CloudFormation 設計工具？。

Amazon ECS 支援在 AWS CloudFormation中建立叢集、任務定義、服務和任務集。下列範例示範如何使用 AWS CLI來以範本建立資源。您也可以使用 AWS CloudFormation 主控台建立這些資源。如欲進一步了解如何使用 AWS CloudFormation 主控台建立資源，請參閱 AWS CloudFormation 使用者指南。

範例範本

使用獨立堆疊建立 Amazon ECS 資源

下列範例示範如何為每個資源使用獨立堆疊來建立 Amazon ECS 資源。

任務定義

您可以使用下列範本來建立 Fargate Linux 任務。

JSON

```
 {
   "AWSTemplateFormatVersion": "2010-09-09",
   "Resources": {
        "ECSTaskDefinition": {
           "Type": "AWS::ECS::TaskDefinition",
           "Properties": {
               "ContainerDefinitions": [
                   {
                       "Command": [
                           "/bin/sh -c \"echo '<html> <head> <title>Amazon ECS
 Sample App</title> <style>body {margin-top: 40px; background-color: #333;} </style>
 </head><body> <div style=color:white;text-align:center> <h1>Amazon ECS Sample App</
h1> <h2>Congratulations!</h2> <p>Your application is now running on a container in
 Amazon ECS.</p> </div></body></html>' >  /usr/local/apache2/htdocs/index.html &&
 httpd-foreground\""
                       ],
                       "EntryPoint": [
                           "sh",
                           "-c"
                       ],
                       "Essential": true,
                       "Image": "httpd:2.4",
                       "LogConfiguration": {
                           "LogDriver": "awslogs",
                           "Options": {
                               "awslogs-group": "/ecs/fargate-task-definition",
                               "awslogs-region": "us-east-1",
                               "awslogs-stream-prefix": "ecs"
                           }
                       },
                       "Name": "sample-fargate-app",
                       "PortMappings": [
                           {
                               "ContainerPort": 80,
                               "HostPort": 80,
                               "Protocol": "tcp"
                           }
                       ]
                   }
               ],
               "Cpu": 256,
```

```
                "ExecutionRoleArn": "arn:aws:iam::aws_account_id:role/
ecsTaskExecutionRole",
                "Family": "task-definition-cfn",
                "Memory": 512,
                "NetworkMode": "awsvpc",
                "RequiresCompatibilities": [
                    "FARGATE"
                ],
                "RuntimePlatform": {
                    "OperatingSystemFamily": "LINUX"
                }
            }

        }
        }
    }
```

YAML

```
AWSTemplateFormatVersion: 2010-09-09
Resources:
  ECSTaskDefinition:
    Type: 'AWS::ECS::TaskDefinition'
    Properties:
      ContainerDefinitions:
        - Command:
            - >-
              /bin/sh -c "echo '<html> <head> <title>Amazon ECS Sample
              App</title> <style>body {margin-top: 40px; background-color:
              #333;} </style> </head><body> <div
              style=color:white;text-align:center> <h1>Amazon ECS Sample
              App</h1> <h2>Congratulations!</h2> <p>Your application is now
              running on a container in Amazon ECS.</p> </div></body></html>' >
              /usr/local/apache2/htdocs/index.html && httpd-foreground"
          EntryPoint:
            - sh
            - '-c'
          Essential: true
          Image: 'httpd:2.4'
          LogConfiguration:
            LogDriver: awslogs
            Options:
              awslogs-group: /ecs/fargate-task-definition
```

```
            awslogs-region: us-east-1
            awslogs-stream-prefix: ecs
        Name: sample-fargate-app
        PortMappings:
          - ContainerPort: 80
            HostPort: 80
            Protocol: tcp
    Cpu: 256
    ExecutionRoleArn: 'arn:aws:iam::aws_account_id:role/ecsTaskExecutionRole'
    Family: task-definition-cfn
    Memory: 512
    NetworkMode: awsvpc
    RequiresCompatibilities:
      - FARGATE
    RuntimePlatform:
      OperatingSystemFamily: LINUX
```

叢集

您可以使用下列範本來建立一個空叢集。

JSON

```
{
    "AWSTemplateFormatVersion": "2010-09-09",
    "Resources": {
        "ECSCluster": {
            "Type": "AWS::ECS::Cluster",
            "Properties": {
                "ClusterName": "MyEmptyCluster"
            }
        }
    }
}
```

YAML

```
AWSTemplateFormatVersion: 2010-09-09
Resources:
  ECSCluster:
    Type: 'AWS::ECS::Cluster'
    Properties:
```

```
      ClusterName: MyEmptyCluster
```

使用 AL2023 Amazon ECS-Optimized-AMI 建立叢集

定義使用容量提供者在 Amazon EC2 上啟動 AL2023 執行個體的叢集。

> **Important**
>
> 如需取得最新的 AMI ID，請參閱《Amazon Elastic Container Service 開發人員指南》中的 Amazon ECS 最佳化 AMI。

JSON

```
{
    "AWSTemplateFormatVersion": "2010-09-09",
    "Description": "EC2 ECS cluster that starts out empty, with no EC2 instances
 yet. An ECS capacity provider automatically launches more EC2 instances as required
 on the fly when you request ECS to launch services or standalone tasks.",
    "Parameters": {
        "InstanceType": {
            "Type": "String",
            "Description": "EC2 instance type",
            "Default": "t2.medium",
            "AllowedValues": [
                "t1.micro",
                "t2.2xlarge",
                "t2.large",
                "t2.medium",
                "t2.micro",
                "t2.nano",
                "t2.small",
                "t2.xlarge",
                "t3.2xlarge",
                "t3.large",
                "t3.medium",
                "t3.micro",
                "t3.nano",
                "t3.small",
                "t3.xlarge"
            ]
        },
```

```
        "DesiredCapacity": {
            "Type": "Number",
            "Default": "0",
            "Description": "Number of EC2 instances to launch in your ECS cluster."
        },
        "MaxSize": {
            "Type": "Number",
            "Default": "100",
            "Description": "Maximum number of EC2 instances that can be launched in
 your ECS cluster."
        },
        "ECSAMI": {
            "Description": "The Amazon Machine Image ID used for the cluster",
            "Type": "AWS::SSM::Parameter::Value<AWS::EC2::Image::Id>",
            "Default": "/aws/service/ecs/optimized-ami/amazon-linux-2023/
recommended/image_id"
        },
        "VpcId": {
            "Type": "AWS::EC2::VPC::Id",
            "Description": "VPC ID where the ECS cluster is launched",
            "Default": "vpc-1234567890abcdef0"
        },
        "SubnetIds": {
            "Type": "List<AWS::EC2::Subnet::Id>",
            "Description": "List of subnet IDs where the EC2 instances will be
 launched",
            "Default": "subnet-021345abcdef67890"
        }
    },
    "Resources": {
        "ECSCluster": {
            "Type": "AWS::ECS::Cluster",
            "Properties": {
                "ClusterSettings": [
                    {
                        "Name": "containerInsights",
                        "Value": "enabled"
                    }
                ]
            }
        },
        "ECSAutoScalingGroup": {
            "Type": "AWS::AutoScaling::AutoScalingGroup",
            "DependsOn": [
```

```
                "ECSCluster",
                "EC2Role"
            ],
            "Properties": {
                "VPCZoneIdentifier": {
                    "Ref": "SubnetIds"
                },
                "LaunchTemplate": {
                    "LaunchTemplateId": {
                        "Ref": "ContainerInstances"
                    },
                    "Version": {
                        "Fn::GetAtt": [
                            "ContainerInstances",
                            "LatestVersionNumber"
                        ]
                    }
                },
                "MinSize": 0,
                "MaxSize": {
                    "Ref": "MaxSize"
                },
                "DesiredCapacity": {
                    "Ref": "DesiredCapacity"
                },
                "NewInstancesProtectedFromScaleIn": true
            },
            "UpdatePolicy": {
                "AutoScalingReplacingUpdate": {
                    "WillReplace": "true"
                }
            }
        },
        "ContainerInstances": {
            "Type": "AWS::EC2::LaunchTemplate",
            "Properties": {
                "LaunchTemplateName": "asg-launch-template-2",
                "LaunchTemplateData": {
                    "ImageId": {
                        "Ref": "ECSAMI"
                    },
                    "InstanceType": {
                        "Ref": "InstanceType"
                    },
```

```
                    "IamInstanceProfile": {
                        "Name": {
                            "Ref": "EC2InstanceProfile"
                        }
                    },
                    "SecurityGroupIds": [
                        {
                            "Ref": "ContainerHostSecurityGroup"
                        }
                    ],
                    "UserData": {
                        "Fn::Base64": {
                            "Fn::Sub": "#!/bin/bash -xe\n echo ECS_CLUSTER=
${ECSCluster} >> /etc/ecs/ecs.config\n yum install -y aws-cfn-bootstrap\n /opt/aws/
bin/cfn-init -v --stack ${AWS::StackId} --resource ContainerInstances --configsets
 full_install --region ${AWS::Region} &\n"
                        }
                    },
                    "MetadataOptions": {
                        "HttpEndpoint": "enabled",
                        "HttpTokens": "required"
                    }
                }
            }
        },
        "EC2InstanceProfile": {
            "Type": "AWS::IAM::InstanceProfile",
            "Properties": {
                "Path": "/",
                "Roles": [
                    {
                        "Ref": "EC2Role"
                    }
                ]
            }
        },
        "CapacityProvider": {
            "Type": "AWS::ECS::CapacityProvider",
            "Properties": {
                "AutoScalingGroupProvider": {
                    "AutoScalingGroupArn": {
                        "Ref": "ECSAutoScalingGroup"
                    },
                    "ManagedScaling": {
```

```
                    "InstanceWarmupPeriod": 60,
                    "MinimumScalingStepSize": 1,
                    "MaximumScalingStepSize": 100,
                    "Status": "ENABLED",
                    "TargetCapacity": 100
                },
                "ManagedTerminationProtection": "ENABLED"
            }
        }
    },
    "CapacityProviderAssociation": {
        "Type": "AWS::ECS::ClusterCapacityProviderAssociations",
        "Properties": {
            "CapacityProviders": [
                {
                    "Ref": "CapacityProvider"
                }
            ],
            "Cluster": {
                "Ref": "ECSCluster"
            },
            "DefaultCapacityProviderStrategy": [
                {
                    "Base": 0,
                    "CapacityProvider": {
                        "Ref": "CapacityProvider"
                    },
                    "Weight": 1
                }
            ]
        }
    },
    "ContainerHostSecurityGroup": {
        "Type": "AWS::EC2::SecurityGroup",
        "Properties": {
            "GroupDescription": "Access to the EC2 hosts that run containers",
            "VpcId": {
                "Ref": "VpcId"
            }
        }
    },
    "EC2Role": {
        "Type": "AWS::IAM::Role",
        "Properties": {
```

```
                "AssumeRolePolicyDocument": {
                    "Statement": [
                        {
                            "Effect": "Allow",
                            "Principal": {
                                "Service": [
                                    "ec2.amazonaws.com"
                                ]
                            },
                            "Action": [
                                "sts:AssumeRole"
                            ]
                        }
                    ]
                },
                "Path": "/",
                "ManagedPolicyArns": [
                    "arn:aws:iam::aws:policy/service-role/
AmazonEC2ContainerServiceforEC2Role",
                    "arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore"
                ]
            }
        },
        "ECSTaskExecutionRole": {
            "Type": "AWS::IAM::Role",
            "Properties": {
                "AssumeRolePolicyDocument": {
                    "Statement": [
                        {
                            "Effect": "Allow",
                            "Principal": {
                                "Service": [
                                    "ecs-tasks.amazonaws.com"
                                ]
                            },
                            "Action": [
                                "sts:AssumeRole"
                            ],
                            "Condition": {
                                "ArnLike": {
                                    "aws:SourceArn": {
                                        "Fn::Sub": "arn:${AWS::Partition}:ecs:
${AWS::Region}:${AWS::AccountId}:*"
                                    }
```

```
                                },
                                "StringEquals": {
                                    "aws:SourceAccount": {
                                        "Fn::Sub": "${AWS::AccountId}"
                                    }
                                }
                            }
                        }
                    ]
                },
                "Path": "/",
                "ManagedPolicyArns": [
                    "arn:aws:iam::aws:policy/service-role/
AmazonECSTaskExecutionRolePolicy"
                ]
            }
        }
    },
    "Outputs": {
        "ClusterName": {
            "Description": "The ECS cluster into which to launch resources",
            "Value": "ECSCluster"
        },
        "ECSTaskExecutionRole": {
            "Description": "The role used to start up a task",
            "Value": "ECSTaskExecutionRole"
        },
        "CapacityProvider": {
            "Description": "The cluster capacity provider that the service should
 use to request capacity when it wants to start up a task",
            "Value": "CapacityProvider"
        }
    }
}
```

YAML

```
AWSTemplateFormatVersion: '2010-09-09'
Description: EC2 ECS cluster that starts out empty, with no EC2 instances yet. An
 ECS capacity provider automatically launches more EC2 instances as required on the
 fly when you request ECS to launch services or standalone tasks.
Parameters:
  InstanceType:
```

```
    Type: String
    Description: EC2 instance type
    Default: t2.medium
    AllowedValues:
      - t1.micro
      - t2.2xlarge
      - t2.large
      - t2.medium
      - t2.micro
      - t2.nano
      - t2.small
      - t2.xlarge
      - t3.2xlarge
      - t3.large
      - t3.medium
      - t3.micro
      - t3.nano
      - t3.small
      - t3.xlarge
  DesiredCapacity:
    Type: Number
    Default: '0'
    Description: Number of EC2 instances to launch in your ECS cluster.
  MaxSize:
    Type: Number
    Default: '100'
    Description: Maximum number of EC2 instances that can be launched in your ECS
 cluster.
  ECSAMI:
    Description: The Amazon Machine Image ID used for the cluster
    Type: AWS::SSM::Parameter::Value<AWS::EC2::Image::Id>
    Default: /aws/service/ecs/optimized-ami/amazon-linux-2023/recommended/image_id
  VpcId:
    Type: AWS::EC2::VPC::Id
    Description: VPC ID where the ECS cluster is launched
    Default: vpc-1234567890abcdef0
  SubnetIds:
    Type: List<AWS::EC2::Subnet::Id>
    Description: List of subnet IDs where the EC2 instances will be launched
    Default: subnet-021345abcdef67890
Resources:
  ECSCluster:
    Type: AWS::ECS::Cluster
    Properties:
```

```
      ClusterSettings:
        - Name: containerInsights
          Value: enabled
  ECSAutoScalingGroup:
    Type: AWS::AutoScaling::AutoScalingGroup
    DependsOn:
      - ECSCluster
      - EC2Role
    Properties:
      VPCZoneIdentifier: !Ref SubnetIds
      LaunchTemplate:
        LaunchTemplateId: !Ref ContainerInstances
        Version: !GetAtt ContainerInstances.LatestVersionNumber
      MinSize: 0
      MaxSize: !Ref MaxSize
      DesiredCapacity: !Ref DesiredCapacity
      NewInstancesProtectedFromScaleIn: true
    UpdatePolicy:
      AutoScalingReplacingUpdate:
        WillReplace: 'true'
  ContainerInstances:
    Type: AWS::EC2::LaunchTemplate
    Properties:
      LaunchTemplateName: asg-launch-template-2
      LaunchTemplateData:
        ImageId: !Ref ECSAMI
        InstanceType: !Ref InstanceType
        IamInstanceProfile:
          Name: !Ref EC2InstanceProfile
        SecurityGroupIds:
          - !Ref ContainerHostSecurityGroup
        UserData: !Base64
          Fn::Sub: |
            #!/bin/bash -xe
            echo ECS_CLUSTER=${ECSCluster} >> /etc/ecs/ecs.config
            yum install -y aws-cfn-bootstrap
            /opt/aws/bin/cfn-init -v --stack ${AWS::StackId} --resource
ContainerInstances --configsets full_install --region ${AWS::Region} &
        MetadataOptions:
          HttpEndpoint: enabled
          HttpTokens: required
  EC2InstanceProfile:
    Type: AWS::IAM::InstanceProfile
    Properties:
```

```
      Path: /
      Roles:
        - !Ref EC2Role
  CapacityProvider:
    Type: AWS::ECS::CapacityProvider
    Properties:
      AutoScalingGroupProvider:
        AutoScalingGroupArn: !Ref ECSAutoScalingGroup
        ManagedScaling:
          InstanceWarmupPeriod: 60
          MinimumScalingStepSize: 1
          MaximumScalingStepSize: 100
          Status: ENABLED
          TargetCapacity: 100
        ManagedTerminationProtection: ENABLED
  CapacityProviderAssociation:
    Type: AWS::ECS::ClusterCapacityProviderAssociations
    Properties:
      CapacityProviders:
        - !Ref CapacityProvider
      Cluster: !Ref ECSCluster
      DefaultCapacityProviderStrategy:
        - Base: 0
          CapacityProvider: !Ref CapacityProvider
          Weight: 1
  ContainerHostSecurityGroup:
    Type: AWS::EC2::SecurityGroup
    Properties:
      GroupDescription: Access to the EC2 hosts that run containers
      VpcId: !Ref VpcId
  EC2Role:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Statement:
          - Effect: Allow
            Principal:
              Service:
                - ec2.amazonaws.com
            Action:
              - sts:AssumeRole
      Path: /
      ManagedPolicyArns:
        - arn:aws:iam::aws:policy/service-role/AmazonEC2ContainerServiceforEC2Role
```

```
        - arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore
  ECSTaskExecutionRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Statement:
          - Effect: Allow
            Principal:
              Service:
                - ecs-tasks.amazonaws.com
            Action:
              - sts:AssumeRole
            Condition:
              ArnLike:
                aws:SourceArn: !Sub arn:${AWS::Partition}:ecs:${AWS::Region}:
${AWS::AccountId}:*
              StringEquals:
                aws:SourceAccount: !Sub ${AWS::AccountId}
      Path: /
      ManagedPolicyArns:
        - arn:aws:iam::aws:policy/service-role/AmazonECSTaskExecutionRolePolicy
Outputs:
  ClusterName:
    Description: The ECS cluster into which to launch resources
    Value: ECSCluster
  ECSTaskExecutionRole:
    Description: The role used to start up a task
    Value: ECSTaskExecutionRole
  CapacityProvider:
    Description: The cluster capacity provider that the service should use to
 request capacity when it wants to start up a task
    Value: CapacityProvider
```

部署 服務

下列範本定義使用容量提供者來請求 AL2023 容量以執行的服務。容器會在上線時啟動至 AL2023 執行個體：

JSON

```
{
  "AWSTemplateFormatVersion": "2010-09-09",
```

```
 "Description": "An example service that deploys in AWS VPC networking mode on
EC2 capacity. Service uses a capacity provider to request EC2 instances to run
on. Service runs with networking in private subnets, but still accessible to the
internet via a load balancer hosted in public subnets.",
 "Parameters": {
     "VpcId": {
         "Type": "String",
         "Description": "The VPC that the service is running inside of"
     },
     "PublicSubnetIds": {
         "Type": "List<AWS::EC2::Subnet::Id>",
         "Description": "List of public subnet ID's to put the load balancer in"
     },
     "PrivateSubnetIds": {
         "Type": "List<AWS::EC2::Subnet::Id>",
         "Description": "List of private subnet ID's that the AWS VPC tasks are in"
     },
     "ClusterName": {
         "Type": "String",
         "Description": "The name of the ECS cluster into which to launch
capacity."
     },
     "ECSTaskExecutionRole": {
         "Type": "String",
         "Description": "The role used to start up an ECS task"
     },
     "CapacityProvider": {
         "Type": "String",
         "Description": "The cluster capacity provider that the service should use
to request capacity when it wants to start up a task"
     },
     "ServiceName": {
         "Type": "String",
         "Default": "web",
         "Description": "A name for the service"
     },
     "ImageUrl": {
         "Type": "String",
         "Default": "public.ecr.aws/docker/library/nginx:latest",
         "Description": "The url of a docker image that contains the application
process that will handle the traffic for this service"
     },
     "ContainerCpu": {
         "Type": "Number",
```

```
            "Default": 256,
            "Description": "How much CPU to give the container. 1024 is 1 CPU"
        },
        "ContainerMemory": {
            "Type": "Number",
            "Default": 512,
            "Description": "How much memory in megabytes to give the container"
        },
        "ContainerPort": {
            "Type": "Number",
            "Default": 80,
            "Description": "What port that the application expects traffic on"
        },
        "DesiredCount": {
            "Type": "Number",
            "Default": 2,
            "Description": "How many copies of the service task to run"
        }
    },
    "Resources": {
        "TaskDefinition": {
            "Type": "AWS::ECS::TaskDefinition",
            "Properties": {
                "Family": {
                    "Ref": "ServiceName"
                },
                "Cpu": {
                    "Ref": "ContainerCpu"
                },
                "Memory": {
                    "Ref": "ContainerMemory"
                },
                "NetworkMode": "awsvpc",
                "RequiresCompatibilities": [
                    "EC2"
                ],
                "ExecutionRoleArn": {
                    "Ref": "ECSTaskExecutionRole"
                },
                "ContainerDefinitions": [
                    {
                        "Name": {
                            "Ref": "ServiceName"
                        },
```

```
                    "Cpu": {
                        "Ref": "ContainerCpu"
                    },
                    "Memory": {
                        "Ref": "ContainerMemory"
                    },
                    "Image": {
                        "Ref": "ImageUrl"
                    },
                    "PortMappings": [
                        {
                            "ContainerPort": {
                                "Ref": "ContainerPort"
                            },
                            "HostPort": {
                                "Ref": "ContainerPort"
                            }
                        }
                    ],
                    "LogConfiguration": {
                        "LogDriver": "awslogs",
                        "Options": {
                            "mode": "non-blocking",
                            "max-buffer-size": "25m",
                            "awslogs-group": {
                                "Ref": "LogGroup"
                            },
                            "awslogs-region": {
                                "Ref": "AWS::Region"
                            },
                            "awslogs-stream-prefix": {
                                "Ref": "ServiceName"
                            }
                        }
                    }
                }
            ]
        }
    },
    "Service": {
        "Type": "AWS::ECS::Service",
        "DependsOn": "PublicLoadBalancerListener",
        "Properties": {
            "ServiceName": {
```

```
                "Ref": "ServiceName"
            },
            "Cluster": {
                "Ref": "ClusterName"
            },
            "PlacementStrategies": [
                {
                    "Field": "attribute:ecs.availability-zone",
                    "Type": "spread"
                },
                {
                    "Field": "cpu",
                    "Type": "binpack"
                }
            ],
            "CapacityProviderStrategy": [
                {
                    "Base": 0,
                    "CapacityProvider": {
                        "Ref": "CapacityProvider"
                    },
                    "Weight": 1
                }
            ],
            "NetworkConfiguration": {
                "AwsvpcConfiguration": {
                    "SecurityGroups": [
                        {
                            "Ref": "ServiceSecurityGroup"
                        }
                    ],
                    "Subnets": {
                        "Ref": "PrivateSubnetIds"
                    }
                }
            },
            "DeploymentConfiguration": {
                "MaximumPercent": 200,
                "MinimumHealthyPercent": 75
            },
            "DesiredCount": {
                "Ref": "DesiredCount"
            },
            "TaskDefinition": {
```

```
                "Ref": "TaskDefinition"
            },
            "LoadBalancers": [
                {
                    "ContainerName": {
                        "Ref": "ServiceName"
                    },
                    "ContainerPort": {
                        "Ref": "ContainerPort"
                    },
                    "TargetGroupArn": {
                        "Ref": "ServiceTargetGroup"
                    }
                }
            ]
        }
    },
    "ServiceSecurityGroup": {
        "Type": "AWS::EC2::SecurityGroup",
        "Properties": {
            "GroupDescription": "Security group for service",
            "VpcId": {
                "Ref": "VpcId"
            }
        }
    },
    "ServiceTargetGroup": {
        "Type": "AWS::ElasticLoadBalancingV2::TargetGroup",
        "Properties": {
            "HealthCheckIntervalSeconds": 6,
            "HealthCheckPath": "/",
            "HealthCheckProtocol": "HTTP",
            "HealthCheckTimeoutSeconds": 5,
            "HealthyThresholdCount": 2,
            "TargetType": "ip",
            "Port": {
                "Ref": "ContainerPort"
            },
            "Protocol": "HTTP",
            "UnhealthyThresholdCount": 10,
            "VpcId": {
                "Ref": "VpcId"
            },
            "TargetGroupAttributes": [
```

```
                {
                    "Key": "deregistration_delay.timeout_seconds",
                    "Value": 0
                }
            ]
        }
    },
    "PublicLoadBalancerSG": {
        "Type": "AWS::EC2::SecurityGroup",
        "Properties": {
            "GroupDescription": "Access to the public facing load balancer",
            "VpcId": {
                "Ref": "VpcId"
            },
            "SecurityGroupIngress": [
                {
                    "CidrIp": "0.0.0.0/0",
                    "IpProtocol": -1
                }
            ]
        }
    },
    "PublicLoadBalancer": {
        "Type": "AWS::ElasticLoadBalancingV2::LoadBalancer",
        "Properties": {
            "Scheme": "internet-facing",
            "LoadBalancerAttributes": [
                {
                    "Key": "idle_timeout.timeout_seconds",
                    "Value": "30"
                }
            ],
            "Subnets": {
                "Ref": "PublicSubnetIds"
            },
            "SecurityGroups": [
                {
                    "Ref": "PublicLoadBalancerSG"
                }
            ]
        }
    },
    "PublicLoadBalancerListener": {
        "Type": "AWS::ElasticLoadBalancingV2::Listener",
```

```
            "Properties": {
                "DefaultActions": [
                    {
                        "Type": "forward",
                        "ForwardConfig": {
                            "TargetGroups": [
                                {
                                    "TargetGroupArn": {
                                        "Ref": "ServiceTargetGroup"
                                    },
                                    "Weight": 100
                                }
                            ]
                        }
                    }
                ],
                "LoadBalancerArn": {
                    "Ref": "PublicLoadBalancer"
                },
                "Port": 80,
                "Protocol": "HTTP"
            }
        },
        "ServiceIngressfromLoadBalancer": {
            "Type": "AWS::EC2::SecurityGroupIngress",
            "Properties": {
                "Description": "Ingress from the public ALB",
                "GroupId": {
                    "Ref": "ServiceSecurityGroup"
                },
                "IpProtocol": -1,
                "SourceSecurityGroupId": {
                    "Ref": "PublicLoadBalancerSG"
                }
            }
        },
        "LogGroup": {
            "Type": "AWS::Logs::LogGroup"
        }
    }
}
```

YAML

```
AWSTemplateFormatVersion: '2010-09-09'
Description: >-
  An example service that deploys in AWS VPC networking mode on EC2 capacity.
  Service uses a capacity provider to request EC2 instances to run on. Service
  runs with networking in private subnets, but still accessible to the internet
  via a load balancer hosted in public subnets.
Parameters:
  VpcId:
    Type: String
    Description: The VPC that the service is running inside of
  PublicSubnetIds:
    Type: 'List<AWS::EC2::Subnet::Id>'
    Description: List of public subnet ID's to put the load balancer in
  PrivateSubnetIds:
    Type: 'List<AWS::EC2::Subnet::Id>'
    Description: List of private subnet ID's that the AWS VPC tasks are in
  ClusterName:
    Type: String
    Description: The name of the ECS cluster into which to launch capacity.
  ECSTaskExecutionRole:
    Type: String
    Description: The role used to start up an ECS task
  CapacityProvider:
    Type: String
    Description: >-
      The cluster capacity provider that the service should use to request
      capacity when it wants to start up a task
  ServiceName:
    Type: String
    Default: web
    Description: A name for the service
  ImageUrl:
    Type: String
    Default: 'public.ecr.aws/docker/library/nginx:latest'
    Description: >-
      The url of a docker image that contains the application process that will
      handle the traffic for this service
  ContainerCpu:
    Type: Number
    Default: 256
    Description: How much CPU to give the container. 1024 is 1 CPU
  ContainerMemory:
```

```
    Type: Number
    Default: 512
    Description: How much memory in megabytes to give the container
  ContainerPort:
    Type: Number
    Default: 80
    Description: What port that the application expects traffic on
  DesiredCount:
    Type: Number
    Default: 2
    Description: How many copies of the service task to run
Resources:
  TaskDefinition:
    Type: 'AWS::ECS::TaskDefinition'
    Properties:
      Family: !Ref ServiceName
      Cpu: !Ref ContainerCpu
      Memory: !Ref ContainerMemory
      NetworkMode: awsvpc
      RequiresCompatibilities:
        - EC2
      ExecutionRoleArn: !Ref ECSTaskExecutionRole
      ContainerDefinitions:
        - Name: !Ref ServiceName
          Cpu: !Ref ContainerCpu
          Memory: !Ref ContainerMemory
          Image: !Ref ImageUrl
          PortMappings:
            - ContainerPort: !Ref ContainerPort
              HostPort: !Ref ContainerPort
          LogConfiguration:
            LogDriver: awslogs
            Options:
              mode: non-blocking
              max-buffer-size: 25m
              awslogs-group: !Ref LogGroup
              awslogs-region: !Ref AWS::Region
              awslogs-stream-prefix: !Ref ServiceName
  Service:
    Type: AWS::ECS::Service
    DependsOn: PublicLoadBalancerListener
    Properties:
      ServiceName: !Ref ServiceName
      Cluster: !Ref ClusterName
```

```
      PlacementStrategies:
        - Field: 'attribute:ecs.availability-zone'
          Type: spread
        - Field: cpu
          Type: binpack
      CapacityProviderStrategy:
        - Base: 0
          CapacityProvider: !Ref CapacityProvider
          Weight: 1
      NetworkConfiguration:
        AwsvpcConfiguration:
          SecurityGroups:
            - !Ref ServiceSecurityGroup
          Subnets: !Ref PrivateSubnetIds
      DeploymentConfiguration:
        MaximumPercent: 200
        MinimumHealthyPercent: 75
      DesiredCount: !Ref DesiredCount
      TaskDefinition: !Ref TaskDefinition
      LoadBalancers:
        - ContainerName: !Ref ServiceName
          ContainerPort: !Ref ContainerPort
          TargetGroupArn: !Ref ServiceTargetGroup
  ServiceSecurityGroup:
    Type: 'AWS::EC2::SecurityGroup'
    Properties:
      GroupDescription: Security group for service
      VpcId: !Ref VpcId
  ServiceTargetGroup:
    Type: 'AWS::ElasticLoadBalancingV2::TargetGroup'
    Properties:
      HealthCheckIntervalSeconds: 6
      HealthCheckPath: /
      HealthCheckProtocol: HTTP
      HealthCheckTimeoutSeconds: 5
      HealthyThresholdCount: 2
      TargetType: ip
      Port: !Ref ContainerPort
      Protocol: HTTP
      UnhealthyThresholdCount: 10
      VpcId: !Ref VpcId
      TargetGroupAttributes:
        - Key: deregistration_delay.timeout_seconds
          Value: 0
```

```
  PublicLoadBalancerSG:
    Type: 'AWS::EC2::SecurityGroup'
    Properties:
      GroupDescription: Access to the public facing load balancer
      VpcId: !Ref VpcId
      SecurityGroupIngress:
        - CidrIp: 0.0.0.0/0
          IpProtocol: -1
  PublicLoadBalancer:
    Type: 'AWS::ElasticLoadBalancingV2::LoadBalancer'
    Properties:
      Scheme: internet-facing
      LoadBalancerAttributes:
        - Key: idle_timeout.timeout_seconds
          Value: '30'
      Subnets: !Ref PublicSubnetIds
      SecurityGroups:
        - !Ref PublicLoadBalancerSG
  PublicLoadBalancerListener:
    Type: 'AWS::ElasticLoadBalancingV2::Listener'
    Properties:
      DefaultActions:
        - Type: forward
          ForwardConfig:
            TargetGroups:
              - TargetGroupArn: !Ref ServiceTargetGroup
                Weight: 100
      LoadBalancerArn: !Ref PublicLoadBalancer
      Port: 80
      Protocol: HTTP
  ServiceIngressfromLoadBalancer:
    Type: 'AWS::EC2::SecurityGroupIngress'
    Properties:
      Description: Ingress from the public ALB
      GroupId: !Ref ServiceSecurityGroup
      IpProtocol: -1
      SourceSecurityGroupId: !Ref PublicLoadBalancerSG
  LogGroup:
    Type: 'AWS::Logs::LogGroup'
```

在同一堆疊中建立多個 Amazon ECS 資源

您可以使用下列範例範本，在同一堆疊中建立多個 Amazon ECS 資源。範本會建立一個名為 CFNCluster 的 Amazon ECS 叢集。此叢集包含一個用於設定 Web 伺服器的 Linux Fargate 任務定義。範本也會建立名為 cfn-service 的服務，該服務會啟動並維護由任務定義所定義的任務。在使用此範本之前，請確保服務的 NetworkConfiguration 中的子網路和安全群組 ID 皆屬於相同的 VPC，且安全群組具有必要的規則。如需安全群組規則的詳細資訊，請參閱《Amazon VPC 使用者指南》中的安全群組規則。

JSON

```
{
    "AWSTemplateFormatVersion": "2010-09-09",
    "Resources": {
        "ECSCluster": {
            "Type": "AWS::ECS::Cluster",
            "Properties": {
                "ClusterName": "CFNCluster"
            }
        },
        "ECSTaskDefinition": {
            "Type": "AWS::ECS::TaskDefinition",
            "Properties": {
                "ContainerDefinitions": [
                    {
                        "Command": [
                            "/bin/sh -c \"echo '<html> <head> <title>Amazon ECS
 Sample App</title> <style>body {margin-top: 40px; background-color: #333;} </style>
 </head><body> <div style=color:white;text-align:center> <h1>Amazon ECS Sample App</
h1> <h2>Congratulations!</h2> <p>Your application is now running on a container in
 Amazon ECS.</p> </div></body></html>' >  /usr/local/apache2/htdocs/index.html &&
 httpd-foreground\""
                        ],
                        "EntryPoint": [
                            "sh",
                            "-c"
                        ],
                        "Essential": true,
                        "Image": "httpd:2.4",
                        "LogConfiguration": {
                            "LogDriver": "awslogs",
                            "Options": {
                                "awslogs-group": "/ecs/fargate-task-definition",
```

```
                            "awslogs-region": "us-east-1",
                            "awslogs-stream-prefix": "ecs"
                        }
                    },
                    "Name": "sample-fargate-app",
                    "PortMappings": [
                        {
                            "ContainerPort": 80,
                            "HostPort": 80,
                            "Protocol": "tcp"
                        }
                    ]
                }
            ],
            "Cpu": 256,
            "ExecutionRoleArn": "arn:aws:iam::aws_account_id::role/
ecsTaskExecutionRole",
            "Family": "task-definition-cfn",
            "Memory": 512,
            "NetworkMode": "awsvpc",
            "RequiresCompatibilities": [
                "FARGATE"
            ],
            "RuntimePlatform": {
                "OperatingSystemFamily": "LINUX"
            }
        }
    },
    "ECSService": {
        "Type": "AWS::ECS::Service",
        "Properties": {
            "ServiceName": "cfn-service",
            "Cluster": {
                "Ref": "ECSCluster"
            },
            "DesiredCount": 1,
            "LaunchType": "FARGATE",
            "NetworkConfiguration": {
                "AwsvpcConfiguration": {
                    "AssignPublicIp": "ENABLED",
                    "SecurityGroups": [
                        "sg-abcdef01234567890"
                    ],
                    "Subnets": [
```

```
                            "subnet-abcdef01234567890"
                        ]
                    }
                },
                "TaskDefinition": {
                    "Ref": "ECSTaskDefinition"
                }
            }
        }
    }
}
```

YAML

```
AWSTemplateFormatVersion: 2010-09-09
Resources:
  ECSCluster:
    Type: 'AWS::ECS::Cluster'
    Properties:
      ClusterName: CFNCluster
  ECSTaskDefinition:
    Type: 'AWS::ECS::TaskDefinition'
    Properties:
      ContainerDefinitions:
        - Command:
            - >-
              /bin/sh -c "echo '<html> <head> <title>Amazon ECS Sample
              App</title> <style>body {margin-top: 40px; background-color:
              #333;} </style> </head><body> <div
              style=color:white;text-align:center> <h1>Amazon ECS Sample
              App</h1> <h2>Congratulations!</h2> <p>Your application is now
              running on a container in Amazon ECS.</p> </div></body></html>' >
              /usr/local/apache2/htdocs/index.html && httpd-foreground"
          EntryPoint:
            - sh
            - '-c'
          Essential: true
          Image: 'httpd:2.4'
          LogConfiguration:
            LogDriver: awslogs
            Options:
              awslogs-group: /ecs/fargate-task-definition
              awslogs-region: us-east-1
```

```
              awslogs-stream-prefix: ecs
          Name: sample-fargate-app
          PortMappings:
            - ContainerPort: 80
              HostPort: 80
              Protocol: tcp
      Cpu: 256
      ExecutionRoleArn: 'arn:aws:iam::aws_account_id:role/ecsTaskExecutionRole'
      Family: task-definition-cfn
      Memory: 512
      NetworkMode: awsvpc
      RequiresCompatibilities:
        - FARGATE
      RuntimePlatform:
        OperatingSystemFamily: LINUX
  ECSService:
    Type: 'AWS::ECS::Service'
    Properties:
      ServiceName: cfn-service
      Cluster: !Ref ECSCluster
      DesiredCount: 1
      LaunchType: FARGATE
      NetworkConfiguration:
        AwsvpcConfiguration:
          AssignPublicIp: ENABLED
          SecurityGroups:
            - sg-abcdef01234567890
          Subnets:
            - subnet-abcdef01234567890
      TaskDefinition: !Ref ECSTaskDefinition
```

使用 從範本 AWS CLI 建立資源

下列指令會使用名為 ecs-template-body.json 的範本內文檔，建立一個名為 ecs-stack 的堆疊。請確定範本內文檔為 JSON 或 YAML 格式。檔案的位置已指定於 --template-body 參數。在此情況下，範本內文檔位於目前的目錄。

```
aws cloudformation create-stack \
      --stack-name ecs-stack \
      --template-body file://ecs-template-body.json
```

若要確保正確建立資源，請檢查 Amazon ECS 主控台，或使用下列指令：

- 下列指令可列出所有任務定義。

```
aws ecs list-task-definitions
```

- 下列指令可列出所有叢集。

```
aws ecs list-clusters
```

- 下列指令可列出叢集 *CFNCluster* 中定義的所有服務。將 *CFNCluster* 替換為您要在其中建立服務的叢集名稱。

```
aws ecs list-services \
      --cluster CFNCluster
```

進一步了解 AWS CloudFormation

若要進一步了解 AWS CloudFormation，請參閱下列資源：

- AWS CloudFormation
- AWS CloudFormation 使用者指南
- AWS CloudFormation 命令列界面使用者指南

使用 AWS Copilot 命令列介面建立 Amazon ECS 資源

AWS Copilot 命令列界面 (CLI) 命令可簡化從本機開發環境在 Amazon ECS 上建置、發佈和操作生產就緒容器化應用程式。 AWS Copilot CLI 與支援現代應用程式最佳實務的開發人員工作流程保持一致：從使用基礎設施做為程式碼，到建立代表使用者佈建的 CI/CD 管道。使用 AWS Copilot CLI 作為日常開發和測試週期的一部分，作為 的替代方案 AWS Management Console。

AWS Copilot 目前支援 Linux、macOS 和 Windows 系統。如需 AWS Copilot CLI 最新版本的詳細資訊，請參閱發行。

> **Note**
>
> AWS Copilot CLI 的原始碼可在 GitHub 上取得。我們建議您提交問題，並為想要進行的變更提取請求。但是，Amazon Web Services 目前不支援執行 AWS Copilot 程式碼的已修改複本。

透過在 Gitter 或 GitHub 上與我們連線來報告 AWS Copilot 的問題，您可以在其中開啟問題、提供意見回饋和報告錯誤。

如需安裝 AWS Copilot CLI 的詳細資訊，請參閱安裝 AWS Copilot CLI。如需部署範例應用程式的詳細資訊，請參閱 使用 AWS Copilot CLI 部署範例 Amazon ECS 應用程式。 AWS Copilot AWS 網站提供 Copilot CLI 的其他文件。

安裝 AWS Copilot CLI

您可以使用 Homebrew 或手動下載二進位檔，並依照下列步驟安裝 AWS Copilot CLI。

使用 Homebrew

以下命令用於使用 Homebrew 在 macOS 或 Linux 系統上安裝 AWS Copilot CLI。在安裝之前，應先安裝 Homebrew。如需詳細資訊，請參閱 Homebrew。

```
brew install aws/tap/copilot-cli
```

下載二進位

除了 Homebrew 之外，您也可以在 macOS、Windows 或 Linux 系統上手動安裝 AWS Copilot CLI。使用下列命令讓作業系統下載二進位檔。macOS 和 Linux 範例也包含將執行許可套用至二進位檔的命令，並列出說明選單以確認安裝是否正常運作。

macOS

macOS：

```
sudo curl -Lo /usr/local/bin/copilot https://github.com/aws/copilot-cli/releases/
latest/download/copilot-darwin \
   && sudo chmod +x /usr/local/bin/copilot \
   && copilot --help
```

macOS ARM 系統：

```
sudo curl -Lo /usr/local/bin/copilot https://github.com/aws/copilot-cli/releases/
latest/download/copilot-darwin-arm64 \
   && sudo chmod +x /usr/local/bin/copilot \
   && copilot --help
```

Linux

對於 Linux x86 (64 位元) 系統：

```
sudo curl -Lo /usr/local/bin/copilot https://github.com/aws/copilot-cli/releases/
latest/download/copilot-linux \
   && sudo chmod +x /usr/local/bin/copilot \
   && copilot --help
```

對於 Linux ARM 系統：

```
sudo curl -Lo /usr/local/bin/copilot https://github.com/aws/copilot-cli/releases/
latest/download/copilot-linux-arm64 \
   && sudo chmod +x /usr/local/bin/copilot \
   && copilot --help
```

Windows

使用 Powershell，執行下列命令：

```
New-Item -Path 'C:\copilot' -ItemType directory; `
  Invoke-WebRequest -OutFile 'C:\copilot\copilot.exe' https://github.com/aws/
copilot-cli/releases/latest/download/copilot-windows.exe
```

(選用) 使用 PGP 簽章驗證手動安裝的 AWS Copilot CLI

AWS Copilot CLI 可執行檔使用 PGP 簽章進行密碼編譯簽署。PGP 簽章可用來驗證 AWS Copilot CLI 可執行檔的有效性。使用 GnuPG 工具，透過下列步驟來驗證簽章。

1. 下載並安裝 GnuPG。如需詳細資訊，請參閱 GnuPG 網站。

   macOS

   建議您使用 Homebrew。參照 Homebrew 網站上的說明，以執行安裝程序。如需詳細資訊，請參閱 Homebrew。Homebrew 安裝完畢後，請從 macOS 終端機使用下列命令。

   ```
   brew install gnupg
   ```

   Linux

   請使用套件軟體管理工具在 Linux 上安裝 gpg。

Windows

從 GnuPG 網站下載 Windows 簡易安裝程式並以管理員身分進行安裝。安裝 GnuPG 後，請關閉並重新打開管理員 PowerShell。

如需詳細資訊，請參閱「下載 GnuPG」。

2. 驗證 GnuPG 路徑是否已新增到您的環境路徑。

macOS

```
echo $PATH
```

如果未在輸出中發現 GnuPG 路徑，請執行以下命令將其新增到路徑中。

```
PATH=$PATH:<path to GnuPG executable files>
```

Linux

```
echo $PATH
```

如果未在輸出中發現 GnuPG 路徑，請執行以下命令將其新增到路徑中。

```
export PATH=$PATH:<path to GnuPG executable files>
```

Windows

```
Write-Output $Env:PATH
```

如果未在輸出中發現 GnuPG 路徑，請執行以下命令將其新增到路徑中。

```
$Env:PATH += ";<path to GnuPG executable files>"
```

3. 建立本機純文字檔案。

macOS

在終端機中，輸入：

```
touch <public_key_filename.txt>
```

使用 TextEdit 開啟檔案。

Linux

在 gedit 等文字編輯器中建立文字檔案。另存為 public_key_filename.txt

Windows

在記事本等文字編輯器中建立文字檔案。另存為 public_key_filename.txt

4. 新增下列 Amazon ECS PGP 公有金鑰內容，並儲存檔案。

```
-----BEGIN PGP PUBLIC KEY BLOCK-----
Version: GnuPG v2

mQINBFq1SasBEADliGcT1NVJ1ydfN8DqebYYe9ne3dt6jqKFmKowLmm6LLGJe7HU
jGtqhCWRDkN+qPpHqdArRgDZAtn2pXY5fEipHgar4CP8QgRnRMO2fl74lmavr4Vg
7K/KH8VHlq2uRw32/B94XLEgRbGTMdWFdKuxoPCttBQaMj3LGn6Pe+6xVWRkChQu
BoQAhjBQ+bEm0kNy0LjNgjNlnL3UMAG56t8E3LANIgGgEnpNsB1UwfWluPoGZoTx
N+6pHBJrKIL/1v/ETU4FXpYw2zvhWNahxeNRnoYj3uycHkeliCrw4kj0+skizBgO
2K7oVX8Oc3j5+ZilhL/qDLXmUCb2az5cMM1mOoF8EKX5HaNuq1KfwJxqXE6NNIcO
lFTrT7QwD5fMNld3FanLgv/ZnIrsSaqJOL6zRSq8O4LN1OWBVbndExk2Kr+5kFxn
5lBPgfPgRj5hQ+KTHMa9Y8Z7yUc64BJiN6F9Nl7FJuSsfqbdkvRLsQRbcBG9qxX3
rJAEhieJzVMEUNl+EgeCkxj5xuSkNU7zw2c3hQZqEcrADLV+hvFJktOz9Gm6xzbq
lTnWWCz4xrIWtuEBA2qE+MlDheVd78a3gIsEaSTfQq0osYXaQbvlnSWOoc1y/5Zb
zizHTJIhLtUyls9WisP2s0emeHZicVMfW61EgPrJAiupgc7kyZvFt4YwfwARAQAB
tCRBbWF6b24gRUNTIDxlY3Mtc2VjdXJpdHlAYW1hem9uLmNvbT6JAhwEEAECAAYF
AlrjL0YACgkQHivRXs0TaQrg1g/+JppwPqHnlVPmv7lessB8I5UqZeD6p6uVpHd7
Bs3pcPp8BV7BdRbs3sPLt5bV1+rkqOlw+0gZ4Q/ue/YbWtOAt4qY0OcEo0HgcnaX
lsB827QIfZIVtGWMhuh94xzm/SJkvngml6KB3YJNnWP61A9qJ37/VbVVLzvcmazA
McWB4HUMNrhd0JgBCo0gIpqCbpJEvUc02Bjn23eEJsS9kC7OUAHyQkVnx4d9UzXF
4OoISF6hmQKIBoLnRrAlj5Qvs3GhvHQ0ThYq0Grk/KMJJX2CSqt7tWJ8gk1n3H3Y
SReRXJRnv7DsDDBwFgT6r5Q2HW1TBUvaoZy5hF6maD09nHcNnvBjqADzeT8Tr/Qu
bBCLzkNSYqqkpgtwv7seoD2P4n1giRvDAOEfMZpVkUr+C252IaH1HZFEz+TvBVQM
Y8OWWxmIJW+J6evjo3N1eO19UHv71jvoF8zljbI4bsL2c+QTJmOv7nRqzDQgCWyp
Id/v2dUVVTk1j9omuLBBwNJzQCB+72LcIzJhYmaP1HC4LcKQG+/f41exuItenatK
lEJQhYtyVXcBlh6Yn/wzNg2NWOwb3vqY/F7m6u9ixAwgtIMgPCDE4aJ86zrrXYFz
N2HqkTSQh77Z8KPKmyGopsmN/reMuilPdINb249nA0dzoN+nj+tTFOYCIaLaFyjs
Z0r1QAOJAjkEEwECACMFAlq1SasCGwMHCwkIBwMCAQYVCAIJCgsEFgIDAQIeAQIX
gAAKCRC86dmkLVF4T9iFEACEnkm1dNXsWUx34R3c0vamHrPxvfkyI1FlEUen8D1h
uX9xy6jCEROHWEp0rjGK4QDPgM93sWJ+s1UAKg214QRVzft0y9/DdR+twApA0fzy
uavIthGd6+03jAAo6udYDE+cZC3P7XBbDiYEWk4XAF9I1JjB8hTZUgvXBL046JhG
```

```
eM17+crgUyQeetkiOQemLbsbXQ40Bd9V7zf7XJraFd8VrwNUwNb+9KFtgAsc9rk+
YIT/PEf+YOPysgcxI4sTWghtyCulVnuGoskgDv4v73PALU0ieUrvvQVqWMRvhVx1
0X90J7cC1KOyhlEQQ1aFTgmQjmXexVTwIBm8LvysFK6YXM41KjOrlz3+6xBIm/qe
bFyLUnf4WoiuOplAaJhK9pRY+XEnGNxdtN4D26Kd0F+PLkm3Tr3Hy3b1Ok34FlGr
KVHUq1TZD7cvMnnNKEELTUcKX+1mV3an16nmAg/my1JSUt6BNK2rJpY1s/kkSGSE
XQ4zuF2IGCpvBFhYAlt5Un5zwqkwwQR3/n2kwAoDzonJcehDw/C/cGos5D0aIU7I
K2X2aTD3+pA7Mx3IMe2hqmYqRt9X42yF1PIEVRneBRJ3HDezAgJrNh0GQWRQkhIx
gz6/cTR+ekr5TptVszS9few2GpI5bCgBKBisZIssT89aw7mAKWut0Gcm4qM9/yK6
1bkCDQRatUmrARAAxNPvVwreJ2yAiFcUpdRlVhsuOgnxvs1QgsIw3H7+Pacr9Hpe
8uftYZqdC82KeSKhpHq7c8gMTMucIINtH25x9BCc73E33EjCL9Lqov1TL7+QkgHe
T+JIhZwdD8Mx2K+LVVVu/aWkNrfMuNwyDUciSI4D5QHa8T+F8fgN4OTpwYjirzel
5yoICMr9hVcbzDNv/ozKCxjx+XKgnFc3wrnDfJfntfDAT7ecwbUTL+viQKJ646s+
psiqXRYtVvYInEhLVrJ0aV6zHFoigE/Bils6/g7ru1Q6CEHqEw++APs5CcE8VzJu
WAGSVHZgun5Y9N4quR/M9Vm+IPMhTxrAg7rOvyRN9cAXfeSMf77I+XTifigNna8x
t/MOdjXr1fjF4pThEi5u6WsuRdFwjY2azEv3vevodTi4HoJReH6dFRa6y8c+UDgl
2iHiOKIpQqLbHEfQmHcDd2fix+AaJKMnPGNku9qCFEMbgSRJpXz6BfwnY1QuKE+I
R6jA0frUNt2jhiGG/F8RceXzohaaC/Cx7LUCUFWc0n7z32C9/Dtj7I1PMOacdZzz
bjJzRKO/ZDv+UN/c9dwAkllzAyPMwGBkUaY68EBstnIliW34aWm6IiHhxioVPKSp
VJfyiXPO0EXqujtHLAeChfjcns3I12YshT1dv2PafG53fp33ZdzeUgsBo+EAEQEA
AYkCHwQYAQIACQUCWrVJqwIbDAAKCRC86dmkLVF4T+ZdD/9x/8APzgNJF3o3STrF
jvnV1ycyhWYGAeBJiu7wjsNWwzMFOv15tLjB7AqeVxZn+WKDD/mIOQ45OZvnYZuy
X7DR0JszaH9wrYTxZLVruAu+t6UL0y/XQ4L1GZ9QR6+r+7t1Mvbfy7BlHbvX/gYt
Rwe/uwdibI0CagEzyX+2D3kTOlHO5XThbXaNf8AN8zha91Jt2Q2UR2X5T6JcwtMz
FBvZnl3LSmZyE0EQehS2iUurU4uWOpGppuqVnbi0jbCvCHKgDGrqZ0smKNAQng54
F365W3g8AfY48s8XQwzmcliowYX9bT8PZiEi0J4QmQh0aXkpqZyFefuWeOL2R94S
XKzr+gRh3BAULoqF+qK+IUMxTip9KTPNvYDpiC66yBiT6gFDji5Ca9pGpJXrC3xe
TXiKQ8DBWDhBPVPrruLIaenTtZEOsPc4I85yt5U9RoPTStcOr34s3w5yEaJagt6S
Gc5r9ysjkfH6+6rbi1ujxMgROSqtqr+RyB+V9A5/OgtNZc8llK6u4UoOCde8jUUW
vqWKvjJB/Kz3u4zaeNu2ZyyHaOqOuH+TETcW+jsY9IhbEzqN5yQYGi4pVmDkY5vu
lXbJnbqPKpRXgM9BecV9AMbPgbDq/5LnHJJXg+G8YQOgp4lR/hC1TEFdIp5wM8AK
CWsENyt2o1rjgMXiZOMF8A5oBLkCDQRatUuSARAAr77kj7j2QR2SZeOSlFBvV7oS
mFeSNnz9xZssqrsm6bTwSHM6YLDwc7Sdf2esDdyzONETwqrVCg+FxgL8hmo9hS4c
rR6tmrP0mOmptr+xLLsKcaP7ogIXsyZnrEAEsvW8PnfayoiPCdc3cMCR/lTnHFGA
7EuR/XLBmi7Qg9tByVYQ5Yj5wB9V4B2yeCt3XtzPqeLKvaxl7PNelaHGJQY/xo+m
V0bndxf9IY+4oFJ4blD32WqvyxESo7vW6WBh7oqv3Zbm0yQrr8a6mDBpqLkvWwNI
3kpJR974tg5o5LfDu1BeeyHWPSGm4U/G4JB+JIG1ADy+RmoWEt4BqTCZ/knnoGvw
D5sTCxbKdmuOmhGyTssoG+3OOcGYHV7pWYPhazKHMPm201xKCjH1RfzRULzGKjD+
yMLT1I3AXFmLmZJXikAOlvE3/wgMqCXscbycbLjLD/bXIuFWo3rzoezeXjgi/DJx
jKBAyBTYO5nMcth1O9oaFd9d0HbsOUDkIMnsgGBE766Piro6MHo0T0rXl07Tp4pI
rwuSOsc6XzCzdImj0Wc6axS/HeUKRXWdXJwno5awTwXKRJMXGfhCvSvbcbc2Wx+L
IKvmB7EB4K3fmjFFE67yolmiw2qRcUBfygtH3eL5XZU28MiCpue8Y8GKJoBAUyvf
KeM1rO8Jm3iRAc5a/D0AEQEAAYkEPgQYAQIACQUCWrVLkgIbAgIpCRC86dmkLVF4
T8FdIAQZAQIABgUCWrVLkgAKCRDePL1hra+LjtHYD/9MucxdFe6bXO1dQR4tKhhQ
P0LRqy6zlBY9ILCLowNdGZdqorogUiUymgn3VhEhVtxTOoHcN7qOuM01PNsRnOeS
```

```
EYjf8Xrb1clzkD6xULwmOclTb9bBxnBc/4PFvHAbZW3QzusaZniNgkuxt6BTfloS
Of4inq71kjmGK+TlzQ6mUMQUg228NUQC+a84EPqYyAeY1sgvgB7hJBhYL0QAxhcW
6m20Rd8iEc6HyzJ3yCOCsKip/nRWAbf0OvfHfRBp0+m0ZwnJM8cPRFjOqqzFpKH9
HpDmTrC4wKP1+TL52LyEqNh4yZitXmZNV7giSRIkk0eDSko+bFy6VbMzKUMkUJK3
D3eHFAMkujmbfJmSMTJOPGn5SB1HyjCZNx6bhIIbQyEUB9gKCmUFaqXKwKpF6rj0
iQXAJxLR/shZ5Rk96VxzOphUl7T90m/PnUEEPwq8KsBhnMRgxa0RFidDP+n9fgtv
HLmrOqX9zBCVXh0mdWYLrWvmzQFWzG7AoE55fkf8nAEPsalrCdtaNUBHRXA0OQxG
AHMOdJQQvBsmqMvuAdjkDWpFu5y0My5ddU+hiUzUyQLjL5Hhd5LOUDdewlZgIw1j
xrEAUzDKetnemM8GkHxDgg8koev5frmShJuce7vSjKpCNg3EIJSgqMOPFjJuLWtZ
vjHeDNbJy6uNL65ckJy6WhGjEADS2WAW1D6Tfekkc21SsIXk/LqEpLMR/0g5OUif
wcEN1rS9IJXBwIy8MelN9qr5KcKQLmfdfBNEyyceBhyVl0MDyHOKC+7PofMtkGBq
13QieRHv5GJ8LB3fclqHV8pwTTo3Bc8z2g0TjmUYAN/ixETdReDoKavWJYSE9yoM
aaJu279ioVTrwpECse0XkiRyKToTjwOb73CGkBZZpJyqux/rmCV/fp4ALdSW8zbz
FJVORaivhoWwzjpfQKhwcU9lABXi2UvVm14v0AfeI7oiJPSU1zM4fEny4oiIBXlR
zhFNih1UjIu82X16mTm3BwbIga/s1fnQRGzyhqUIMii+mWra23EwjChaxpvjjcUH
5ilLc5Zq781aCYRygYQw+hu5nFkOH1R+Z50Ubxjd/aqUfnGIAX7kPMD3Lof4KldD
Q8ppQriUvxVo+4nPV6rpTy/PyqCLWDjkguHpJsEFsMkwajrAz0QNSAU5CJ0G2Zu4
yxvYlumHCEl7nbFrm0vIiA75Sa8KnywTDsyZsu3XcOcf3g+g1xWTpjJqy2bYXlqz
9uDOWtArWHOis6bq8l9RE6xr1RBVXS6uqgQIZFBGyq66b0dIq4D2JdsUvgEMaHbc
e7tBfeB1CMBdA64e9Rq7bFR7Tvt8gasCZYlNr3lydh+dFHIEkH53HzQe6l88HEic
+0jVnLkCDQRa55wJARAAyLya2Lx6gyoWoJN1a6740q3o8e9d4KggQOfGMTCflmeq
ivuzgN+3DZHN+9ty2KxXMtn0mhHBerZdbNJyjMNT1gAgrhPNB4HtXBXum2wS57WK
DNmade914L7FWTPAWBG2Wn448OEHTqsClICXXWy9IICgclAEyIq0Yq5mAdTEgRJS
Z8t4GpwtDL9gNQyFXaWQmDmkAsCygQMvhAlmu9xOIzQG5CxSnZFk7zcuL60k14Z3
Cmt49k4T/7ZU8goWi8tt+rU78/IL3J/fF9+1civ1OwuUidgfPCSvOUW1JojsdCQA
L+RZJcoXq7lfOFj/eNjeOSstCTDPfTCL+kThE6E5neDtbQHBYkEX1BRiTedsV4+M
ucgiTrdQFWKf89G72xdv8ut9AYYQ2BbEYU+JAYhUH8rYYui2dHKJIgjNvJscuUWb
+QEqJIRleJRhrO+/CHgMs4fZAkWF1VFhKBkcKmEjLn1f7EJJUUW84ZhKXjO/AUPX
1CHsNjziRceuJCJYox1cwsoq6jTE50GiNzcIxTn9xUc0UMKFeggNAFys1K+TDTm3
Bzo8H5ucjCUEmUm9lhkGwqTZgOlRX5eqPX+JBoSaObqhgqCa5IPinKRa6MgoFPHK
6sYKqroYwBGgZm6Js5chpNchvJMs/3WXNOEVg0J3z3vP0DMhxqWm+r+n9zlW8qsA
EQEAAYkEPgQYAQgACQUCWuecCQIbAgIpCRC86dmkLVF4T8FdIAQZAQgABgUCWuec
CQAKCRBQ3szEcQ5hr+ykD/4tOLRHFHXuKUcxgGaubUcVtsFrwBKma1cYjqaPms8u
6Sk0wfGRI32G/GhOrp0Ts/MOkbObq6VLTh8N5Yc/53MEl8zQFw9Y5AmRoW4PZXER
ujs5s7p4oR7xHMihMjCCBn1bvrR+34YPfgzTcgLiOEFHYT8UTxwnGmXOvNkMM7md
xD3CV5q6VAte8WKBo/220II3fcQlc9r/oWX4kXXkb0v9hoGwKbDJ1tzqTPrp/xFt
yohqnvImpnlz+Q9zXmbrWYL9/g8VCmW/NN2gju2G3Lu/TlFUWIT4v/5OPK6TdeNb
VKJO4+S8bTayqSG9CML1S57KSgCo5HUhQWeSNHI+fpe5oX6FALPT9JLDce8OZz1i
cZZ0MELP37mOOQun0AlmHm/hVzf0f311PtbzcqWaE51tJvgUR/nZFo6Ta3O5Ezhs
3VlEJNQ1Ijf/6DH87SxvAoRIARCuZd0qxBcDK0avpFzUtbJd24lRA3WJpkEiMqKv
RDVZkE4b6TW61f0o+LaVfK6E8oLpixegS4fiqC16mFrOdyRk+RJJfIUyz0WTDVmt
g0U1CO1ezokMSqkJ7724pyjr2xf/r9/sC6aOJwB/lKgZkJfC6NqL7TlxVA31dUga
LEOvEJTTE4gl+tYtfsCDvALCtqL0jduSkUo+RXcBItmXhA+tShW0pbS2Rtx/ixua
KohVD/0R4QxiSwQmICNtm9mw9ydIl1yjYXX5a9x4wMJracNY/LBybJPFnZnT4dYR
```

```
z4XjqysDwvvYZByaWoIe3QxjX84V6MlI2IdAT/xImu8gbaCI8tmyfpIrLnPKiR9D
VFYfGBXuAX7+HgPPSFtrHQONCALxxzlbNpS+zxt9r0MiLgcLyspWxSdmoYGZ6nQP
RO5Nm/ZVS+u2imPCRzNUZEMa+dlE6kHx0rS0dPiuJ4O7NtPeYDKkoQtNagspsDvh
cK7CSqAiKMq06UBTxqlTSRkm62eOCtcs3p3OeHu5GRZF1uzTET0ZxYkaPgdrQknx
ozjP5mC7X+45lcCfmcVt94TFNL5HwEUVJpmOgmzILCI8yoDTWzloo+i+fPFsXX4f
kynhE83mSEcr5VHFYrTY3mQXGmNJ3bCLuc/jq7ysGq69xiKmTlUeXFm+aojcRO5i
zyShIRJZ0GZfuzDYFDbMV9amA/YQGygLw//zP5ju5SW26dNxlf3MdFQE5JJ86rn9
MgZ4gcpazHEVUsbZsgkLizRp9imUiH8ymLqAXnfRGlU/LpNSefnvDFTtEIRcpOHc
bhayG0bk51Bd4mioOXnIsKy4j63nJXA27x5EVVHQ1sYRN8Ny4Fdr2tMAmj2O+X+J
qX2yy/UX5nSPU492e2CdZ1UhoU0SRFY3bxKHKB7SDbVeav+K5g==
=Gi5D
-----END PGP PUBLIC KEY BLOCK-----
```

以下為 Amazon ECS PGP 公有金鑰的詳細資訊，以供參考：

```
Key ID: BCE9D9A42D51784F
Type: RSA
Size: 4096/4096
Expires: Never
User ID: Amazon ECS
Key fingerprint: F34C 3DDA E729 26B0 79BE AEC6 BCE9 D9A4 2D51 784F
```

5. 在終端機中透過下列命令使用 Amazon ECS PGP 公有金鑰匯入檔案。

```
gpg --import <public_key_filename.txt>
```

6. 下載 AWS Copilot CLI 簽章。簽章是 ASCII 分離的 PGP 簽章，存放於副檔名為 `.asc` 的檔案中。簽章檔案的名稱會與對應執行檔的名稱相同，並在名稱後加上 `.asc`。

   macOS

   對於 macOS 系統，請使用以下命令。

   ```
   sudo curl -Lo copilot.asc https://github.com/aws/copilot-cli/releases/latest/download/copilot-darwin.asc
   ```

   Linux

   對於 Linux x86 (64 位元) 系統，請執行以下命令。

```
sudo curl -Lo copilot.asc https://github.com/aws/copilot-cli/releases/latest/
download/copilot-linux.asc
```

對於 Linux ARM 系統，請執行以下命令。

```
sudo curl -Lo copilot.asc https://github.com/aws/copilot-cli/releases/latest/
download/copilot-linux-arm64.asc
```

Windows

使用 Powershell，執行下列命令。

```
Invoke-WebRequest -OutFile 'C:\copilot\copilot.asc' https://github.com/aws/
copilot-cli/releases/latest/download/copilot-windows.exe.asc
```

7. 使用以下命令驗證簽章。

   - macOS 和 Linux 系統：

     ```
     gpg --verify copilot.asc /usr/local/bin/copilot
     ```

   - Windows 系統：

     ```
     gpg --verify 'C:\copilot\copilot.asc' 'C:\copilot\copilot.exe'
     ```

   預期的輸出結果：

   ```
   gpg: Signature made Tue Apr  3 13:29:30 2018 PDT
   gpg:                using RSA key DE3CBD61ADAF8B8E
   gpg: Good signature from "Amazon ECS <ecs-security@amazon.com>" [unknown]
   gpg: WARNING: This key is not certified with a trusted signature!
   gpg:          There is no indication that the signature belongs to the owner.
   Primary key fingerprint: F34C 3DDA E729 26B0 79BE  AEC6 BCE9 D9A4 2D51 784F
        Subkey fingerprint: EB3D F841 E2C9 212A 2BD4  2232 DE3C BD61 ADAF 8B8E
   ```

> **Important**
>
> 輸出中有警告很正常，不會有問題。這是因為個人 PGP 金鑰 (如果有) 和 Amazon ECS PGP 金鑰之間沒有信任鏈。如需詳細資訊，請參閱「信任網路」。

8. 對於 Windows 安裝，在 Powershell 上執行下列命令，將 AWS Copilot 目錄新增至路徑。

```
$Env:PATH += ";<path to Copilot executable files>"
```

使用 AWS Copilot CLI 部署範例 Amazon ECS 應用程式

安裝 AWS Copilot CLI 之後，您可以依照下列步驟部署範例應用程式、驗證部署，以及清除資源。

必要條件

在開始前，請確定您符合以下先決條件：

- 安裝及設定 AWS CLI。如需詳細資訊，請參閱 AWS 命令列介面。
- 執行 `aws configure` 以設定 AWS Copilot CLI 用來管理應用程式和服務的預設設定檔。
- 安裝和執行 Docker。如需詳細資訊，請參閱 Docker 入門。

使用單一命令部署範例 Amazon ECS 應用程式

1. 使用以下命令部署從 GitHub 儲存庫複製的範例 Web 應用程式。如需 AWS Copilot `init`及其旗標的詳細資訊，請參閱 AWS Copilot 文件。

```
git clone https://github.com/aws-samples/aws-copilot-sample-service.git demo-app &&
 \
cd demo-app &&                                \
copilot init --app demo                       \
  --name api                                  \
  --type 'Load Balanced Web Service'          \
  --dockerfile './Dockerfile'                 \
  --port 80                                   \
  --tag  latest                               \
  --deploy
```

2. 部署完成後， AWS Copilot CLI 會傳回 URL，供您用來驗證部署。您也可以使用下列命令來驗證應用程式的狀態。

   - 列出所有 AWS Copilot 應用程式。

     ```
     copilot app ls
     ```

   - 顯示有關應用程式中環境和服務的資訊。

     ```
     copilot app show
     ```

   - 顯示有關您的環境的資訊。

     ```
     copilot env ls
     ```

   - 顯示有關服務的資訊，包括端點、容量和相關資源。

     ```
     copilot svc show
     ```

   - 應用程式中所有服務的清單。

     ```
     copilot svc ls
     ```

   - 顯示已部署服務的日誌。

     ```
     copilot svc logs
     ```

   - 顯示服務狀態。

     ```
     copilot svc status
     ```

3. 完成此示範後，請執行下列命令來清除相關聯的資源，並避免未使用的資源產生費用。

   ```
   copilot app delete
   ```

Amazon ECS最佳實務

您可以使用下列任一頁面，了解 Amazon ECS 網路最重要的操作最佳實務。

最佳實務概觀	進一步了解	
將應用程式連接至網際網路	將 Amazon ECS 應用程式連接至網際網路	
ECS 從網際網路接收 Amazon 的傳入連線。	ECS 從網際網路接收 Amazon 傳入連線的最佳實務	
將 Amazon ECS連接至 中的其他 AWS 服務 VPC	從 內部將 Amazon ECS 連線至 AWS 服務的最佳實務 VPC	
跨 AWS 帳戶 和 的網路服務 VPCs	跨 AWS 帳戶 和 聯網 Amazon ECS服務的最佳實務 VPCs	
網路問題疑難排解	AWS 適用於 Amazon ECS 網路故障診斷的 服務	

您可以使用下列任一頁面，了解 Fargate 在 Amazon 上最重要的操作最佳實務ECS。

最佳實務概觀	進一步了解	
Fargate 安全性	Amazon ECS 中的 Fargate 安全最佳實務	
Fargate 安全考量	Amazon ECS 的 Fargate 安全考量	
Fargate 容器映像提取行為上的 Linux 容器	Fargate 容器映像提取行為上的 Linux 容器適用於 Amazon ECS	

最佳實務概觀	進一步了解	
Fargate 容器映像提取行為上的 Windows 容器	Fargate 容器映像提取行為上的 Windows 容器適用於 Amazon ECS	
Fargate 任務淘汰	Amazon ECS AWS Fargate 的任務淘汰和維護	

您可以使用下列任一頁面，了解任務定義最重要的操作最佳實務。

最佳實務概觀	進一步了解	
容器映像	Amazon ECS 容器映像的最佳實務	
任務大小	Amazon ECS 任務大小的最佳實務	
磁碟區最佳實務	Amazon ECS 任務的儲存選項	

您可以使用下列任一頁面，了解叢集和容量最重要的操作最佳實務。

最佳實務概觀	進一步了解	
Fargate 安全性	Amazon ECS 中的 Fargate 安全最佳實務	
EC2 容器執行個體安全考量	Amazon ECS 的 Amazon EC2 容器執行個體安全考量	
叢集自動擴展	最佳化 Amazon ECS 叢集自動擴展	

您可以使用下列任一頁面，了解任務和服務最重要的操作最佳實務。

最佳實務概觀	進一步了解
最佳化任務啟動時間	最佳化 Amazon ECS 任務啟動時間
服務參數	Amazon ECS 服務參數的最佳實務
最佳化負載平衡器運作狀態檢查參數	最佳化 Amazon ECS 的負載平衡器運作狀態檢查參數
最佳化負載平衡器連線排放參數	最佳化 Amazon ECS 的負載平衡器連線耗盡參數
最佳化服務自動擴展	最佳化 Amazon ECS 服務自動擴展

您可以使用下列任何頁面來了解最重要的安全操作最佳實務。

最佳實務概觀	進一步了解
網路安全	Amazon ECS 的網路安全最佳實務
任務和容器安全	Amazon ECS 任務和容器安全最佳實務

AWS Fargate 適用於 Amazon ECS

AWS Fargate 是一項技術，可以與 Amazon ECS 搭配使用以執行容器，而不需管理 Amazon EC2 執行個體的伺服器或叢集。有了 AWS Fargate，就不再需要佈建、設定或擴展虛擬機器的叢集來執行容器。這樣一來即無須選擇伺服器類型、決定何時擴展叢集，或最佳化叢集壓縮。

當您使用 Fargate 啟動類型執行任務和服務時，將會在容器中封裝應用程式、指定 CPU 和記憶體需求、定義聯網和 IAM 政策，並啟動應用程式。每個 Fargate 任務都有自己的隔離界限，並且不會與另一個任務共用基礎核心、CPU 資源、記憶體資源或彈性網路界面。您可以將 `requiresCompatibilities` 任務定義參數設定為 `FARGATE`，以設定 Fargate 的任務定義。如需詳細資訊，請參閱啟動類型。

Fargate 提供適用於 Amazon Linux 2 （平台 1.3.0 版）、 Bottlerocket 作業系統 （平台 1.4.0 版） 和 Microsoft Windows 2019 Server Full 和 Core 版本的平台版本。除非另有說明，否則此頁面上的資訊適用於所有 Fargate 平台。

本主題說明不同的 Fargate 任務和服務元件，並注意搭配使用 Fargate 與 Amazon ECS 的特殊考量。

如需在 Fargate 上支援 Linux 容器的區域的資訊，請參閱the section called “AWS Fargate 上的 Linux 容器”。

如需在 Fargate 上支援 Windows 容器的區域的資訊，請參閱the section called “AWS Fargate 上的 Windows 容器”。

逐步解說

如需有關如何使用 主控台的資訊，請參閱：

- 了解如何為 Fargate 啟動類型建立 Amazon ECS Linux 任務
- 了解如何為 Fargate 啟動類型建立 Amazon ECS Windows 任務

如需有關如何使用 的資訊 AWS CLI，請參閱：

- 使用 為 Fargate 啟動類型建立 Amazon ECS Linux 任務 AWS CLI
- 使用 為 Fargate 啟動類型建立 Amazon ECS Windows 任務 AWS CLI

容量提供者

下列容量提供者可供使用：

- Fargate
- Fargate Spot - 相較於AWS Fargate價格，以折扣費率執行可容忍中斷的 Amazon ECS 任務。Fargate Spot 在備用運算容量上執行任務。當 AWS 需要恢復容量時，您的任務會受到兩分鐘警告而中斷。如需詳細資訊，請參閱Fargate 啟動類型的 Amazon ECS 叢集 。

任務定義

使用 Fargate 啟動類型的任務不支援所有可用的 Amazon ECS 任務定義參數。有些參數完全不予以支援，而其他參數對 Fargate 任務會有不同的行為。如需詳細資訊，請參閱任務 CPU 和記憶體。

平台版本

AWS Fargate 平台版本用於參考 Fargate 任務基礎設施的特定執行期環境。其結合了核心與容器執行時間版本。在執行任務或建立服務以維護許多相同的任務時，請選取平台版本。

為因應執行時間環境演進 (例如是否有核心或作業系統的更新、新功能、錯誤修正或安全性更新)，我們會不時發行新的平台版本修訂版。建立新的平台版本修訂版即可更新 Fargate 平台版本。每項任務在其生命週期期間都只會在一個平台版本修訂版上執行。如果想要使用最新的平台版本修訂版，必須啟動新任務。在 Fargate 上執行的新任務一律會在平台版本的最新修訂版上執行，以確保任務一律在已修補的安全基礎設施上啟動。

如果發現影響現有平台版本的安全問題， 會 AWS 建立新的平台版本修補修訂，並淘汰在易受攻擊修訂上執行的任務。在某些案例中，您可能會收到通知，告知您在 Fargate 上的任務已排程淘汰。如需詳細資訊，請參閱Amazon ECS AWS Fargate 的任務淘汰和維護 。

如需詳細資訊，請參閱 適用於 Amazon ECS 的 Fargate 平台版本。

服務負載平衡

AWS Fargate 上的 Amazon ECS 服務可以選擇性地設定為使用 Elastic Load Balancing，將您服務中任務的流量平均分配。

AWS Fargate 上的 Amazon ECS 服務支援 Application Load Balancer 和 Network Load Balancer 負載平衡器類型。Application Load Balancer 用於路由 HTTP/HTTPS (或 Layer 7) 流量。Network

Load Balancer 用於路由 TCP 或 UDP (或 Layer 4) 流量。如需詳細資訊，請參閱使用負載平衡來分發 Amazon ECS 服務流量。

當您為這些服務建立目標群組時，必須選擇 `ip` 作為目標類型，而不是選擇 `instance`。因為採用 `awsvpc` 網路模式的任務是與彈性網路界面相關聯，而非與 Amazon EC2 執行個體相關聯。如需詳細資訊，請參閱使用負載平衡來分發 Amazon ECS 服務流量。

當使用平台版本 1.4 或更高版本時，只支援使用 Network Load Balancer 將 UDP 流量路由到 AWS Fargate 上的 Amazon ECS 任務。

用量指標

您可以使用 CloudWatch 用量指標來提供您帳戶的資源用量可見度。使用這些指標，以 CloudWatch 圖表和儀表板視覺化目前的服務使用狀況。

AWS Fargate 用量指標對應至 AWS 服務配額。您可以設定警示，在您的用量接近服務配額時發出警示。如需 AWS Fargate 的服務配額詳細資訊，請參閱 AWS Fargate 服務配額。

如需 AWS Fargate 用量指標的詳細資訊，請參閱AWS Fargate用量指標。

Amazon ECS 何時使用 Fargate 啟動類型的安全考量

我們建議希望為其任務進行強隔離的客戶使用 Fargate。Fargate 會在硬體虛擬化環境中執行每個任務。這可確保這些容器化工作負載不會與其他任務共用網路介面、Fargate 暫時性儲存、CPU 或記憶體。如需詳細資訊，請參閱 的安全概觀 AWS Fargate。

Amazon ECS 中的 Fargate 安全最佳實務

建議您在使用 AWS Fargate時考量下列最佳實務。如需其他指引，請參閱 的安全概觀 AWS Fargate。

使用 AWS KMS 加密 Fargate 的暫時性儲存

您應該讓 AWS KMS 或您自己的客戶受管金鑰加密暫時性儲存。對於使用平台版本 `1.4.0` 或更新版本在 Fargate 上託管的任務，每個任務都會接收 20 GiB 的暫時性儲存。如需詳細資訊，請參閱客戶受管金鑰 (CMK)。您可以增加暫時性儲存的總量，最多可達 200 GiB，方法是在任務定義中指定 `ephemeralStorage` 參數。對於 2020 年 5 月 28 日或之後啟動的這類任務，暫時性儲存會使用 Fargate 管理的加密金鑰，以 AES-256 加密演算法加密。

如需詳細資訊，請參閱 Amazon ECS 任務的儲存選項。

範例：使用暫時性儲存加密在 Fargate 平台 1.4.0 版上啟動任務

下列命令將在 Fargate 平台 1.4 版上啟動任務。由於此任務是做為叢集的一部分啟動，因此會使用自動加密的暫時性儲存的 20 GiB。

```
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
```

使用 Fargate 進行核心 syscall 追蹤的 SYS_PTRACE 功能

由 Docker 提供從容器中新增或刪除的 Linux 功能預設組態。

在 Fargate 上啟動的任務僅支援新增 SYS_PTRACE 核心功能。

下列影片說明如何透過 Sysdig Falco 專案使用此功能。

#ContainersFromTheCouch - 使用 SYS_PTRACE 功能對 Fargate 任務進行故障診斷

在上一個影片中討論的程式碼可以在 GitHub 這裡找到。

搭配 Fargate 執行期監控使用 Amazon GuardDuty

Amazon GuardDuty 是一種威脅偵測服務，可協助保護您的帳戶、容器、工作負載和 AWS 環境中的資料。GuardDuty 使用機器學習 (ML) 模型，以及異常和威脅偵測功能，持續監控不同的日誌來源和執行時間活動，以識別環境中的潛在安全風險和惡意活動，並排定其優先順序。

GuardDuty 中的執行期監控透過持續監控 AWS 日誌和聯網活動來識別惡意或未經授權的行為，來保護 Fargate 上執行的工作負載。執行期監控使用輕量且全受管的 GuardDuty 安全代理程式，可分析主機上行為，例如檔案存取、程序執行和網路連線。這涵蓋的問題包括權限升級、使用公開的登入資料，或與惡意 IP 地址、網域的通訊，以及 Amazon EC2 執行個體和容器工作負載上存在惡意軟體。如需詳細資訊，請參閱《GuardDuty 使用者指南》中的 GuardDuty 執行期監控。 GuardDuty

Amazon ECS 的 Fargate 安全考量

每項任務都有專用的基礎架構容量，因為 Fargate 會在隔離的虛擬環境中執行每個工作負載。在 Fargate 上執行的工作負載不會與其他任務共用網路介面、暫時性儲存、CPU 或記憶體。您可以在任

務中執行多個容器，包括應用程式容器和附屬容器，或者僅是附屬。附屬是在 Amazon ECS 任務中與應用程式容器一起執行的容器。當應用程式容器執行核心應用程式程式碼時，在附屬中執行的程序可以增強應用程式。附屬可協助您將應用程式功能隔離到專用容器中，讓您更輕鬆地更新應用程式的各個部分。

屬於相同任務的容器會共用 Fargate 啟動類型的資源，因為這些容器將始終會在相同的主機上執行並共用計算資源。這些容器也共用 Fargate 提供的暫時性儲存。任務中的 Linux 容器會共用網路命名空間，包括 IP 地址和網路連接埠。在任務內部，屬於任務的容器可以透過本機主機進行相互通訊。

Fargate 中的執行期環境可防止您使用 EC2 執行個體支援的某些控制器功能。架構在 Fargate 上執行的工作負載時，請考量下列事項：

- 沒有特殊權限的容器或存取 - Fargate 目前無法提供特殊權限的容器或存取等功能。這將影響使用案例，例如在 Docker 中執行 Docker。
- 對 Linux 功能的有限存取 - 在 Fargate 上執行容器的環境已被鎖定。其他 Linux 功能 (例如：CAP_SYS_ADMIN 和 CAP_NET_ADMIN) 受到限制，以防止權限提升。Fargate 支援將 CAP_SYS_PSTRACE Linux 功能新增至任務中，以允許在任務中部署的可觀測性和安全工具來監控容器化應用程式。
- 無法存取基礎主機 - 客戶和 AWS 運算子都無法連線到執行客戶工作負載的主機。您可以使用 ECS Exec 執行命令，或為在 Fargate 上執行的容器取得 Shell。您可以使用 ECS Exec 來協助收集診斷資訊以進行偵錯。Fargate 也可防止容器存取基礎主機的資源，例如檔案系統、裝置、網路和容器執行期。
- 網路 - 您可以使用安全群組和網路 ACL 來控制傳入和傳出流量。Fargate 任務會從 VPC 中設定的子網路接收 IP 地址。

適用於 Amazon ECS 的 Fargate 平台版本

AWS Fargate 平台版本用於參考 Fargate 任務基礎設施的特定執行期環境。其結合了核心與容器執行時間版本。在執行任務或建立服務以維護許多相同的任務時，請選取平台版本。

為因應執行時間環境演進 (例如是否有核心或作業系統的更新、新功能、錯誤修正或安全性更新)，我們會不時發行新的平台版本修訂版。建立新的平台版本修訂版即可更新 Fargate 平台版本。每項任務在其生命週期期間都只會在一個平台版本修訂版上執行。如果想要使用最新的平台版本修訂版，必須啟動新任務。在 Fargate 上執行的新任務一律會在平台版本的最新修訂版上執行，以確保任務一律在已修補的安全基礎設施上啟動。

如果發現影響現有平台版本的安全問題， 會 AWS 建立新的平台版本修補修訂，並淘汰在易受攻擊修訂上執行的任務。在某些案例中，您可能會收到通知，告知您在 Fargate 上的任務已排程淘汰。如需詳細資訊，請參閱Amazon ECS AWS Fargate 的任務淘汰和維護 。

您可以在執行任務或部署服務時指定平台版本。

指定平台版本時，應考慮以下項目：

- 您可以指定特定版本編號，例如 `1.4.0`或 `LATEST`。

  LATEST Linux 平台版本為 `1.4.0`。

  LATEST Windows 平台版本為 `1.0.0` 。
- 如果要更新服務的平台版本，請建立部署。例如，假設您有在 Linux 平台版本 `1.3.0` 上執行任務的服務。若要變更服務以在 Linux 平台版本 上執行任務`1.4.0`，請更新您的服務並指定新的平台版本。您的任務會透過最新的平台版本和最新的平台版本修訂版重新部署。如需有關部署的詳細資訊，請參閱 Amazon ECS 服務。
- 當您沒有更新平台版本便擴展服務規模時，這些任務所獲得的平台版本，即為服務目前部署上指定的版本。例如，假設您有在 Linux 平台版本 `1.3.0` 上執行任務的服務。如果增加所需的服務數量，服務排程器會使用平台版本 `1.3.0` 的最新平台版本修訂版來啟動新任務。
- 新任務一律會在平台版本的最新版本上執行。這可確保任務一律位於安全且修補的基礎設施上。
- Fargate 上 Linux 容器和 Windows 容器的平台版本編號各自獨立。例如，Fargate 上 Windows 容器的平台版本 `1.0.0` 中使用的行為、功能和軟體與 Fargate 上 Linux 容器的平台版本 `1.0.0` 無法相比。
- 下列適用於 Fargate Windows 平台版本。

  Microsoft Windows Server 容器映像必須在特定版本的 Windows Server 中建立。執行任務或建立符合 Windows Server 容器映像的服務時，必須在 `platformFamily` 中選取相同版本的 Windows Server。此外，您可以在任務定義中提供相符的 `operatingSystemFamily`，以防止任務在錯誤的 Windows 版本上執行。如需詳細資訊，請參閱 Microsoft Learn 網站上的比對容器主機版本與容器映像版本。

在 Amazon ECS 上遷移至 Linux 平台 1.4.0 版

將 Fargate 任務上的 Amazon ECS 從平台版本 `1.0.0`、`1.1.0`、`1.2.0` 或 `1.3.0` 遷移至平台版本 `1.4.0` 時，應考慮以下項目。最佳實務是先確認您的任務在平台版本上正常運作，`1.4.0`再遷移任務。

- 進出任務的網路流量行為已更新。從平台 1.4.0 版開始，Fargate 上的所有 Amazon ECS 任務都會收到單一彈性網路介面（稱為任務 ENI），且所有網路流量都會流經 VPC 中的該 ENI。您可以透過 VPC 流程日誌查看流量。如需詳細資訊，請參閱 Fargate 啟動類型的 Amazon ECS 任務聯網選項。
- 如果您使用界面 VPC 端點，請考慮下列事項。
  - 對於使用 Amazon ECR 託管的容器映像，您需要下列端點。如需詳細資訊，請參閱 《Amazon Elastic Container Registry 使用者指南》中的 Amazon ECR 介面 VPC 端點 (AWS PrivateLink)。
    - com.amazonaws.***region***.ecr.dkr Amazon ECR VPC 端點
    - com.amazonaws.***region***.ecr.api Amazon ECR VPC 端點
    - Amazon S3 閘道端點
  - 當您的任務定義參考 Secrets Manager 秘密來擷取容器的敏感資料時，您必須為 Secrets Manager 建立介面 VPC 端點。如需詳細資訊，請參閱 《AWS Secrets Manager 使用者指南》中的搭配使用 Secrets Manager 與 VPC 端點。
  - 當您的任務定義參考 Systems Manager 參數存放區參數來擷取容器的敏感資料時，您必須建立 Systems Manager 的介面 VPC 端點。如需詳細資訊，請參閱AWS Systems Manager 《 使用者指南》中的使用適用於 Systems Manager 的 VPC 端點來改善 EC2 執行個體的安全性。
  - 與您的任務相關聯的彈性網路界面 (ENI) 的安全群組需要安全群組規則，才能允許任務與 VPC 端點之間的流量。

Fargate Linux 平台版本變更日誌

以下是可用的 Linux 平台版本。如需平台版本取代的相關資訊，請參閱AWS Fargate Linux 平台版本棄用。

1.4.0

以下是平台版本 `1.4.0` 的變更日誌。

- 從 2020 年 11 月 5 日開始，在 Fargate 上啟動的使用平台版本 `1.4.0` 的任何新 Amazon ECS 任務都將能夠使用以下功能：
  - 當您使用 Secrets Manager 來存放敏感資料時，您可以將特定的 JSON 金鑰或特定版本的秘密作為環境變數插入在日誌組態中。如需詳細資訊，請參閱將敏感資料傳遞至 Amazon ECS 容器。
  - 使用 `environmentFiles` 容器定義參數，指定大量環境變數。如需詳細資訊，請參閱將個別環境變數傳遞至 Amazon ECS 容器。
  - 在 VPC 中執行的任務以及為 IPv6 啟用的子網路將被指派一個私有 IPv4 地址和一個 IPv6 地址。如需詳細資訊，請參閱 Fargate 啟動類型的 Amazon ECS 任務聯網選項。

  - 任務中繼資料端點版本 4 提供有關任務和容器的其他中繼資料，包括任務啟動類型、容器的 Amazon Resource Name (ARN)，以及使用的日誌驅動程式和日誌驅動程式選項。查詢 `/stats` 端點時，您還會收到容器的網路速度統計資訊。如需詳細資訊，請參閱任務中繼資料端點第 4 版。
- 從 2020 年 7 月 30 日開始，在 Fargate 上啟動的使用平台版本 `1.4.0` 的任何新 Amazon ECS 任務將能夠使用 Network Load Balancer 將 UDP 流量路由到 Fargate 任務上的 Amazon ECS。如需詳細資訊，請參閱使用負載平衡來分發 Amazon ECS 服務流量。
- 自 2020 年 5 月 28 日起，在 Fargate 上使用平台版本啟動的任何新 Amazon ECS 任務`1.4.0`，都會使用 AWS 擁有的加密金鑰，以 AES-256 加密演算法加密其暫時性儲存體。如需詳細資訊，請參閱 Amazon ECS 的 Fargate 任務暫時性儲存 和 Amazon ECS 任務的儲存選項。
- 新增了對使用 Amazon EFS 檔案系統磁碟區的支援，以便實現持久性任務儲存。如需詳細資訊，請參閱搭配 Amazon ECS 使用 Amazon EFS 磁碟區。
- 暫時任務儲存已增加到每個任務至少 20 GB。如需詳細資訊，請參閱Amazon ECS 的 Fargate 任務暫時性儲存。
- 進出任務的網路流量行為已更新。從平台版本 1.4.0 開始，所有 Fargate 任務會收到單一彈性網路界面 (也稱為任務 ENI)，所有網路流量都會流經 VPC 中的 ENI，而且您可以透過 VPC 流程日誌查看。如需 Amazon EC2 啟動類型聯網的詳細資訊，請參閱 EC2 啟動類型的 Amazon ECS 任務聯網選項。如需 Fargate 啟動類型聯網的詳細資訊，請參閱 Fargate 啟動類型的 Amazon ECS 任務聯網選項。
- 任務 ENI 新增 Jumbo Frame 的支援。網路界面皆以最大傳輸單位 (MTU) 來設定，這是適合單一框架的最大酬載大小。MTU 越大，單一框架能容納的應用程式酬載越多，可降低每個框架的額外負荷並提高效率。當任務和目的地之間的網路路徑支援 Jumbo Frame (例如，VPC 中剩餘的所有流量) 時，支援 Jumbo Frame 會降低額外負荷。
- CloudWatch Container Insights 將包含 Fargate 任務的網路效能指標。如需詳細資訊，請參閱使用具有增強可觀測性的 Container Insights 監控 Amazon ECS 容器。
- 已新增任務中繼資料端點第 4 版的支援，該端點為您的 Fargate 任務提供額外的資訊，包括任務的網路統計資料，以及執行中任務所在的可用區域。如需詳細資訊，請參閱 Amazon ECS 任務中繼資料端點第 4 版 和 Fargate 上任務的 Amazon ECS 任務中繼資料端點第 4 版。
- 在容器定義中，新增對 SYS_PTRACE Linux 參數的支援。如需詳細資訊，請參閱Linux 參數。
- Fargate 容器代理會取代所有 Fargate 任務的 Amazon ECS 容器代理的使用。此變更通常不會對您的任務執行方式產生影響。
- 容器執行時間目前使用 Containerd，而非 Docker。此變更很可能不會對您的任務執行方式產生影響。您會注意到，容器執行時間產生的一些錯誤訊息會從提及 Docker 變成為較為一般的錯誤。如需詳細資訊，請參閱 Amazon ECS 已停止的任務錯誤訊息。

- 以 Amazon Linux 2 為基礎。

1.3.0

以下是平台版本 `1.3.0` 的變更日誌。

- 從 2019 年 9 月 30 日開始，啟動的任何新 Fargate 任務都支援 `awsfirelens` 日誌驅動程式。設定 FireLens for Amazon ECS 使用任務定義參數，將日誌路由到 AWS 服務或 AWS Partner Network (APN) 目的地，以進行日誌儲存和分析。如需詳細資訊，請參閱將 Amazon ECS 日誌傳送至 AWS 服務或 AWS Partner。
- 增加了對 Fargate 任務的任務回收機制，此過程將重新整理屬於 Amazon ECS 服務的各項任務。如需詳細資訊，Amazon ECS 上的 AWS Fargate 任務淘汰和維護。
- 從 2019 年 3 月 27 日開始，啟動的任何新的 Fargate 任務可以使用額外的任務定義參數，來定義代理組態、容器啟動和關機的相依性，以及每一容器的啟動和停止逾時值。如需詳細資訊，請參閱代理組態、容器相依性及容器逾時。
- 從 2019 年 4 月 2 日開始，啟動的任何新 Fargate 任務都支援將敏感資料注入您的容器，方法是將敏感資料存放在 AWS Secrets Manager 秘密或 AWS Systems Manager 參數存放區參數中，然後在容器定義中參考它們。如需詳細資訊，請參閱將敏感資料傳遞至 Amazon ECS 容器。
- 從 2019 年 5 月 1 日開始，啟動的任何新 Fargate 任務支援使用 `secretOptions` 容器定義參數，以參考容器日誌組態中的敏感資料。如需詳細資訊，請參閱將敏感資料傳遞至 Amazon ECS 容器。
- 從 2019 年 5 月 1 日開始，啟動的任何新 Fargate 任務除了支援 `awslogs` 日誌驅動程式外，還支援 `splunk` 日誌驅動程式。如需詳細資訊，請參閱儲存與記錄。
- 從 2019 年 7 月 9 日開始，啟動的任何新 Fargate 任務都支援 CloudWatch Container Insights。如需詳細資訊，請參閱使用具有增強可觀測性的 Container Insights 監控 Amazon ECS 容器。
- 從 2019 年 12 月 3 日開始，支援 Fargate Spot 容量提供者。如需詳細資訊，請參閱Fargate 啟動類型的 Amazon ECS 叢集 。
- 以 Amazon Linux 2 為基礎。

AWS Fargate Linux 平台版本棄用

此頁面列出 AWS Fargate 已棄用或已排定棄用的 Linux 平台版本。這些平台版本在其公佈的取代日期前都會保持可用。

對已排定取代的每個平台版本提供強制更新日期。在強制更新日期當天，任何使用 `LATEST` 平台版本 (已排定取代的平台版本) 的服務都將使用強制新部署選項進行更新。使用強制新部署選項更新服務時，

在排定取代的平台版本上執行的所有任務都會停止，並且會使用 LATEST 標籤當時指示的平台版本啟動新任務。已設定明確平台版本的獨立任務或服務不受強制更新日期的影響。

我們建議您更新您的服務和獨立任務，以使用最新的平台版本。如需遷移至最新平台版本的詳細資訊，請參閱在 Amazon ECS 上遷移至 Linux 平台 1.4.0 版。

平台版本到達取代日期後，平台版本將不再適用於新任務或服務。任何明確使用過時平台版本的獨立任務或服務，都會繼續使用該平台版本，直到任務停止為止。在取代日期之後，已被取代的平台版本將不再收到任何安全性更新或錯誤修正。

平台版本	強制更新日期	取代日期
1.0.0	2020 年 10 月 26 日	2020 年 12 月 14 日
1.1.0	2020 年 10 月 26 日	2020 年 12 月 14 日
1.2.0	2020 年 10 月 26 日	2020 年 12 月 14 日

如需目前平台版本的相關詳細資訊，請參閱適用於 Amazon ECS 的 Fargate 平台版本。

已棄用 Fargate Linux 版本變更日誌

1.2.0

以下是平台版本 `1.2.0` 的變更日誌。

> **Note**
>
> 平台版本 `1.2.0` 已無法使用。如需平台版本取代的相關資訊，請參閱AWS Fargate Linux 平台版本棄用。

- 新增使用 的私有登錄驗證支援 AWS Secrets Manager。如需詳細資訊，請參閱在 Amazon ECS 中使用非AWS 容器映像。

1.1.0

以下是平台版本 `1.1.0` 的變更日誌。

> **Note**
>
> 平台版本 `1.1.0` 已無法使用。如需平台版本取代的相關資訊，請參閱AWS Fargate Linux 平台版本棄用。

- 新增對 Amazon ECS 任務中繼資料端點的支援。如需詳細資訊，請參閱Amazon ECS 任務中繼資料可用於 Fargate 上的任務。
- 在容器定義中，新增對 Docker 運作狀態檢查的支援。如需詳細資訊，請參閱運作狀態檢查。
- 新增對 Amazon ECS 服務探索的支援。如需詳細資訊，請參閱使用服務探索將 Amazon ECS 服務與 DNS 名稱連線。

1.0.0

以下是平台版本 `1.0.0` 的變更日誌。

> **Note**
>
> 平台版本 `1.0.0` 已無法使用。如需平台版本取代的相關資訊，請參閱AWS Fargate Linux 平台版本棄用。

- 以 Amazon Linux 2017.09 為基礎。
- 初始版本。

Fargate Windows 平台版本變更日誌

以下是可用於 Windows 容器的平台版本。

1.0.0

以下是平台版本 `1.0.0` 的變更日誌。

- 下列 Microsoft Windows Server 作業系統上支援的初始版本：
  - Windows Server 2019 Full
  - Windows Server 2019 Core
  - Windows Server 2022 Full

- Windows Server 2022 Core

Fargate 容器映像提取行為上的 Linux 容器適用於 Amazon ECS

每個 Fargate 任務都會在自己的單次使用、單一租用戶執行個體上執行。當您在 Fargate 上執行 Linux 容器時，不會快取執行個體上的容器映像或容器映像層。因此，對於任務中定義的每個容器映像，需要從每個 Fargate 任務的容器映像登錄檔中提取整個容器映像。提取影像所需的時間與啟動 Fargate 任務所需的時間直接相關。

請考量下列事項，以最佳化映像提取時間。

容器映像鄰近

若要縮短下載容器映像所需的時間，請盡可能將資料放在接近運算的位置。透過網際網路或跨 提取容器映像 AWS 區域 可能會影響下載時間。建議您將容器映像存放在任務將執行的相同區域中。如果您在 Amazon ECR 中存放容器映像，請使用 VPC 介面端點來進一步縮短映像提取時間。如需詳細資訊，請參閱《Amazon ECR 使用者指南》中的 Amazon ECR 介面 VPC 端點 (AWS PrivateLink)。

容器映像大小縮減

容器映像的大小會直接影響下載時間。減少容器映像的大小或容器映像層的數量，可以減少下載映像所需的時間。輕量型基礎映像（例如最小的 Amazon Linux 2023 容器映像）可以明顯小於傳統作業系統基礎映像。如需最小映像的詳細資訊，請參閱《Amazon Linux 2023 使用者指南》中的 AL2023 最小容器映像。

替代壓縮演算法

容器映像層通常會在推送至容器映像登錄檔時壓縮。壓縮容器映像層可減少必須跨網路傳輸並存放在容器映像登錄檔中的資料量。容器映像層由容器執行時間下載至執行個體後，該層會解壓縮。使用的壓縮演算法和可供執行時間使用的 vCPUs數量會影響解除壓縮容器映像所需的時間。在 Fargate 上，您可以增加任務的大小，或利用效能更佳的 zstd 壓縮演算法來減少解壓縮所需的時間。如需詳細資訊，請參閱 GitHub 上的 zstd。如需有關如何實作 Fargate 映像的資訊，請參閱使用 zstd 壓縮容器映像縮短 AWS Fargate 啟動時間。

延遲載入容器映像

對於大型容器映像 (> 250mb)，最好是延遲載入容器映像，而不是下載所有容器映像。在 Fargate 上，您可以使用 Seekable OCI (SOCI) 來延遲從容器映像登錄檔載入容器映像。如需詳細資訊，請參閱 GitHub 上的社交快照，以及使用 Seekable OCI (SOCI) 的延遲載入容器映像。

Amazon ECS Fargate 考量事項上的 Windows 容器

以下是您在 AWS Fargate 上執行 Windows 容器時需要知道的差異和考量事項。

如果您需要在 Linux 和 Windows 容器上執行任務，則需要為每個作業系統建立個別的任務定義。

AWS 會處理作業系統授權管理，因此您不需要任何其他 Microsoft Windows Server 授權。

AWS Fargate 上的 Windows 容器支援下列作業系統：

- Windows Server 2019 Full
- Windows Server 2019 Core
- Windows Server 2022 Full
- Windows Server 2022 Core

AWS Fargate 上的 Windows 容器支援 awslogs 驅動程式。如需詳細資訊，請參閱the section called “將日誌傳送至 CloudWatch ”。

Fargate 上的 Windows 容器不支援下列功能：

- Amazon FSx
- ENI 中繼
- Windows 容器gMSAs
- 適用於任務的 App Mesh 服務和代理整合
- 適用於任務的 Firelens 日誌路由器整合
- EFS 磁碟區
- EBS 磁碟區
- 下列任務定義參數：
  - `maxSwap`
  - `swappiness`
  - `environmentFiles`
- Fargate Spot 容量提供者
- 映像磁碟區

  Dockerfile `volume` 選項會遭到忽略。而是在任務定義中使用綁定掛載。如需詳細資訊，請參閱搭配 Amazon ECS 使用繫結掛載。

Fargate 容器映像提取行為上的 Windows 容器適用於 Amazon ECS

Fargate Windows 會快取 Microsoft 提供的最新一個月伺服器核心基礎映像。這些映像符合每個修補程式星期二更新的 KB/Build 數字修補程式。例如，在 2024 年 4 月 9 日，Microsoft 為 Windows Server 2019 發行 KB5036896 (17763.5696)。上個月 KB 在 2024 年 3 月 12 日是 KB5035849 (17763.5576)。因此，針對 平台`WINDOWS_SERVER_2019_CORE`和`WINDOWS_SERVER_2019_FULL`下列容器映像已快取：

- `mcr.microsoft.com/windows/servercore:ltsc2019`
- `mcr.microsoft.com/windows/servercore:10.0.17763.5696`
- `mcr.microsoft.com/windows/servercore:10.0.17763.5576`

此外，在 2024 年 4 月 9 日，Microsoft 為 Windows Server 2022 發行 KB5036909 (20348.2402)。上個月 KB 在 2024 年 3 月 12 日是 KB5035857 (20348.2340)。因此，針對 平台`WINDOWS_SERVER_2022_CORE`和`WINDOWS_SERVER_2022_FULL`下列容器映像已快取：

- `mcr.microsoft.com/windows/servercore:ltsc2022`
- `mcr.microsoft.com/windows/servercore:10.0.20348.2402`
- `mcr.microsoft.com/windows/servercore:10.0.20348.2340`

Amazon ECS 的 Fargate 任務暫時性儲存

佈建時，在 上託管在 Linux 容器上的每個 Amazon ECS 任務都會 AWS Fargate 收到下列暫時性儲存，以供繫結掛載使用。這可以在任務定義中使用 `volumes`、`mountPoints` 和 `volumesFrom` 參數的容器之間進行掛載和共用。上的 Windows 容器不支援此功能 AWS Fargate。

Fargate Linux 容器平台版本

1.4.0 版或更新版本

根據預設，使用平台 `1.4.0` 版或更新版本的託管於 Fargate 上的所有 Amazon ECS 任務都會收到至少 20 GB 的暫時性儲存。暫時性儲存的總量可以增加，最多可達 200 GiB。您可在任務定義中指定 `ephemeralStorage` 參數來實現這一操作。

任務的提取、壓縮和未壓縮的容器映像都會存放在暫時性儲存中。若要判斷您的任務必須使用的暫時性儲存總量，您必須從為任務配置的暫時性儲存總量中減去容器映像使用的儲存量。

對於 2020 年 5 月 28 日或之後啟動並使用平台版本 1.4.0 或更新版本的任務，會使用 AES-256 加密演算法來加密暫時性儲存。此演算法使用 AWS 擁有的加密金鑰，或者您可以建立自己的客戶受管金鑰。如需詳細資訊，請參閱AWS Fargate 暫時性儲存的客戶受管金鑰。

若任務使用 2022 年 11 月 18 日或之後啟動的平台版本 1.4.0 或更高版本，暫時性儲存使用量會透過任務中繼資料端點回報。任務中的應用程式可以查詢任務中繼資料端點版本 4，以取得其暫時性儲存保留大小和使用量。

此外，如果您開啟 Amazon CloudWatch Container Insights，暫時性儲存預留大小和使用量會傳送至 Container Insights。

> **Note**
>
> Fargate 會在磁盤上保留空間。此空間僅由 Fargate 使用。我們不會向您收費。它不會顯示在這些指標中。但是，您可以在其他工具 (例如 df) 中看到此額外儲存空間。

1.3.0 版或更早版本

對於使用平台 1.3.0 版或更早版本的 Fargate 任務上的 Amazon ECS，每個任務都會收到下列暫時性儲存。

- 10 GB 的 Docker 層儲存體

  > **Note**
  >
  > 此數量包括壓縮和未壓縮的容器映像成品。

- 額外的 4 GB 磁碟區掛載。這可以在任務定義中使用 volumes、mountPoints 和 volumesFrom 參數的容器之間進行掛載和共用。

Fargate Windows 容器平台版本

1.0.0 版或更新版本

根據預設，使用平台 1.0.0 版或更新版本的託管於 Fargate 上的所有 Amazon ECS 任務都會收到至少 20 GB 的暫時性儲存。暫時性儲存的總量可以增加，最多可達 200 GiB。您可在任務定義中指定 ephemeralStorage 參數來實現這一操作。

任務的提取、壓縮和未壓縮的容器映像都會存放在暫時性儲存中。若要判斷您的任務必須使用的暫時性儲存總量，您必須從為任務配置的暫時性儲存總量中減去容器映像使用的儲存量。

如需詳細資訊，請參閱搭配 Amazon ECS 使用繫結掛載。

Amazon ECS 暫時性儲存的客戶受管金鑰 AWS Fargate

AWS Fargate 支援客戶受管金鑰加密存放在暫時性儲存體中的 Amazon ECS 任務資料，以協助法規敏感型客戶滿足其內部安全政策。客戶仍然可以獲得 Fargate 的無伺服器優勢，同時為合規稽核人員提供自我管理儲存加密的增強可見性。雖然 Fargate 預設具有 Fargate 受管暫時性儲存加密，但客戶也可以在加密財務或運作狀態相關資訊等敏感資料時，使用自己的自我管理金鑰。

您可以將自己的金鑰匯入 AWS KMS 或建立金鑰 AWS KMS。這些自我管理金鑰存放在 中 AWS KMS ，並執行標準 AWS KMS 生命週期動作，例如輪換、停用和刪除。您可以在 CloudTrail 日誌中稽核金鑰存取和用量。

根據預設，KMS 金鑰支援每個金鑰 50，000 個授予。Fargate 會針對每個客戶受管金鑰任務使用單一 AWS KMS 授權，因此支援最多 50，000 個同時執行的金鑰任務。如果您想要增加此數字，您可以要求提高限制，這會依case-by-case核准。

Fargate 使用客戶受管金鑰不會收取任何額外費用。您只需支付使用儲存和 API 請求 AWS KMS 金鑰的標準價格。

主題

- 為 Amazon ECS 的 Fargate 暫時性儲存建立加密金鑰
- 管理 Amazon ECS Fargate 暫時性儲存的 AWS KMS 金鑰

為 Amazon ECS 的 Fargate 暫時性儲存建立加密金鑰

建立客戶受管金鑰來加密儲存在 Fargate 暫時性儲存體上的資料。

> **Note**
>
> Fargate 暫時性儲存加密搭配客戶受管金鑰不適用於 Windows 任務叢集。
> Fargate 暫時性儲存加密搭配客戶受管金鑰在 `platformVersions` 之前無法使用`1.4.0`。
> Fargate 會保留暫時性儲存空間，而該儲存空間僅供 Fargate 使用，且您不需要支付該空間的費用。配置可能與非客戶受管金鑰任務不同，但總空間保持不變。您可以在 等工具中檢視此變更`df`。
> Fargate 暫時性儲存不支援多區域金鑰。

Fargate 暫時性儲存不支援 KMS 金鑰別名。

若要建立客戶受管金鑰 (CMK) 來加密 Fargate 中的暫時性儲存 AWS KMS，請遵循下列步驟。

1. 導覽至 https://console.aws.amazon.com/kms。
2. 請遵循 AWS Key Management Service 開發人員指南中建立金鑰的指示。
3. 建立 AWS KMS 金鑰時，請務必在金鑰政策中提供 Fargate 服務相關 AWS KMS 操作許可。政策中必須允許下列 API 操作，才能將客戶受管金鑰與 Amazon ECS 叢集資源搭配使用。

   - `kms:GenerateDataKeyWithoutPlainText` - 從提供的金鑰呼叫 `GenerateDataKeyWithoutPlainText` 以產生加密的資料 AWS KMS 金鑰。
   - `kms:CreateGrant` - 將授予新增至客戶受管金鑰。准許控制對指定 AWS KMS 金鑰的存取，允許存取 Amazon ECS Fargate 所需的准許操作。如需使用授與的詳細資訊，請參閱 AWS Key Management Service 開發人員指南。這可讓 Amazon ECS Fargate 執行下列動作：
     - 呼叫 `Decrypt` AWS KMS 以取得加密金鑰，以解密暫時性儲存資料。
     - 設定淘汰主體，以允許 服務至 `RetireGrant`。
   - `kms:DescribeKey` - 提供客戶受管金鑰詳細資訊，以允許 Amazon ECS 在對稱且已啟用時驗證金鑰。

   下列範例顯示您要套用至目標 AWS KMS 金鑰以進行加密的金鑰政策。若要使用範例政策陳述式，請以您自己的資訊取代*#########*。一如往常，只設定您需要的許可，但您需要 AWS KMS 提供許可給至少一個使用者，以避免發生錯誤。

```
{
      "Sid": "Allow generate data key access for Fargate tasks.",
      "Effect": "Allow",
      "Principal": { "Service":"fargate.amazonaws.com" },
      "Action": [
        "kms:GenerateDataKeyWithoutPlaintext"
      ],
      "Condition": {
        "StringEquals": {
          "kms:EncryptionContext:aws:ecs:clusterAccount": [
            "customerAccountId"
          ],
          "kms:EncryptionContext:aws:ecs:clusterName": [
             "clusterName"
```

```
          ]
        }
      },
      "Resource": "*"
    },
    {
      "Sid": "Allow grant creation permission for Fargate tasks.",
      "Effect": "Allow",
      "Principal": { "Service":"fargate.amazonaws.com" },
      "Action": [
        "kms:CreateGrant"
      ],
      "Condition": {
        "StringEquals": {
          "kms:EncryptionContext:aws:ecs:clusterAccount": [
            "customerAccountId"
          ],
          "kms:EncryptionContext:aws:ecs:clusterName": [
             "clusterName"
          ]
        },
       "ForAllValues:StringEquals": {
          "kms:GrantOperations": [
             "Decrypt"
          ]
       }
      },
      "Resource": "*"
    },
    {
      "Sid": "Allow describe key permission for cluster operator - CreateCluster
 and UpdateCluster.",
      "Effect": "Allow",
      "Principal": { "AWS":"arn:aws:iam::customerAccountId:role/customer-chosen-
role" },
      "Action": [
        "kms:DescribeKey"
      ],
      "Resource": "*"
    }
```

Fargate 任務使用 aws:ecs:clusterAccount和 aws:ecs:clusterName加密內容金鑰搭配金鑰進行密碼編譯操作。客戶應新增這些許可，以限制對特定帳戶和/或叢集的存取。當您指定叢集時，請使用叢集名稱，而不是 ARN。

如需詳細資訊，請參閱 AWS KMS 開發人員指南中的加密內容。

建立或更新叢集時，您可以選擇使用條件索引鍵 fargateEphemeralStorageKmsKeyId。此條件金鑰可讓客戶更精細地控制 IAM 政策。對fargateEphemeralStorageKmsKeyId組態的更新只會在新的服務部署上生效。

以下是允許客戶僅將許可授予一組特定核准 AWS KMS 金鑰的範例。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ecs:CreateCluster",
        "ecs:UpdateCluster"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "ecs:fargate-ephemeral-storage-kms-key": "arn:aws:kms:us-
west-2:111122223333:key/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111"
        }
      }
    }
  ]
}
```

接下來是拒絕嘗試移除已與叢集建立關聯的 AWS KMS 金鑰的範例。

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Deny",
    "Action": [
        "ecs:CreateCluster",
        "ecs:UpdateCluster"
```

```
    ],
    "Resource": "*",
    "Condition": {
      "Null": {
        "ecs:fargate-ephemeral-storage-kms-key": "true"
      }
    }
  }
}
```

客戶可以使用 `describe-cluster`、 或 `describe-services`命令 AWS CLI `describe-tasks`，查看其未受管任務或服務任務是否使用 金鑰加密。

如需詳細資訊，請參閱《 AWS KMS 開發人員指南》中的 的條件金鑰 AWS KMS。

AWS Management Console

1. 開啟主控台，網址為 https://console.aws.amazon.com/ecs/v2。
2. 在左側導覽中選擇叢集，然後在右上角建立叢集，或選擇現有叢集。針對現有叢集，選擇右上角的更新叢集。
3. 在工作流程的加密區段下，您可以選擇受管儲存和 Fargate 暫時性儲存下的 AWS KMS 金鑰。您也可以從這裡選擇建立 AWS KMS 金鑰。
4. 完成建立新叢集後，請選擇建立，如果您要更新現有的叢集，請選擇更新。

AWS CLI

以下是使用 建立叢集和設定 Fargate 暫時性儲存的範例 AWS CLI （使用您自己的值取代*##*值）：

```
aws ecs create-cluster --cluster clusterName \
--configuration '{"managedStorageConfiguration":
{"fargateEphemeralStorageKmsKeyId":"arn:aws:kms:us-
west-2:012345678901:key/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111"}}'
{
    "cluster": {
        "clusterArn": "arn:aws:ecs:us-west-2:012345678901:cluster/clusterName",
        "clusterName": "clusterName",
        "configuration": {
            "managedStorageConfiguration": {
                "fargateEphemeralStorageKmsKeyId": "arn:aws:kms:us-
west-2:012345678901:key/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111"
```

```
            }
        },
        "status": "ACTIVE",
        "registeredContainerInstancesCount": 0,
        "runningTasksCount": 0,
        "pendingTasksCount": 0,
        "activeServicesCount": 0,
        "statistics": [],
        "tags": [],
        "settings": [],
        "capacityProviders": [],
        "defaultCapacityProviderStrategy": []
    },
    "clusterCount": 5
}
```

AWS CloudFormation

以下是使用 建立叢集和設定 Fargate 暫時性儲存的範例範本 AWS CloudFormation （使用您自己的值取代*##*值）：

```
AWSTemplateFormatVersion: 2010-09-09
Resources:
  MyCluster:
    Type: AWS::ECS::Cluster
    Properties:
      ClusterName: "clusterName"
      Configuration:
        ManagedStorageConfiguration:
          FargateEphemeralStorageKmsKeyId: "arn:aws:kms:us-
west-2:012345678901:key/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111"
```

管理 Amazon ECS Fargate 暫時性儲存的 AWS KMS 金鑰

建立或匯入您的 AWS KMS 金鑰以加密 Fargate 暫時性儲存體之後，您會以與任何其他 AWS KMS 金鑰相同的方式進行管理。

自動輪換 AWS KMS 金鑰

您可以啟用自動金鑰輪換或手動輪換。自動金鑰輪換會每年為您輪換金鑰，方法是為金鑰產生新的密碼編譯材料。 AWS KMS 也會儲存所有先前版本的密碼編譯材料，因此您可以解密使用先前金鑰版本的任何資料。在您刪除金鑰 AWS KMS 之前， 不會刪除任何輪換的資料。

自動金鑰輪換是選用的，可以隨時啟用或停用。

停用或撤銷 AWS KMS 金鑰

如果您停用 中的客戶受管金鑰 AWS KMS，它不會對執行中的任務產生任何影響，並且會在其生命週期中繼續運作。如果新任務使用停用或撤銷的金鑰，任務會失敗，因為無法存取金鑰。您應該設定 CloudWatch 警示或類似警示，以確保不再需要停用的金鑰來解密已加密的資料。

刪除 AWS KMS 金鑰

刪除金鑰應一律是最後一個手段，而且只有在您確定不再需要刪除的金鑰時，才應完成。嘗試使用已刪除金鑰的新任務將會失敗，因為它們無法存取該金鑰。 AWS KMS 建議停用金鑰，而不是刪除該金鑰。如果您覺得有必要刪除金鑰，建議您先停用該金鑰，並設定 CloudWatch 警示，以確保不需要該金鑰。如果您確實刪除金鑰， AWS KMS 則 至少提供七天來改變主意。

稽核 AWS KMS 金鑰存取

您可以使用 CloudTrail 日誌來稽核對 AWS KMS 金鑰的存取。您可以檢查 AWS KMS 操作 `CreateGrant`、 `GenerateDataKeyWithoutPlaintext`和 `Decrypt`。這些操作也會顯示 `aws:ecs:clusterAccount`和 `aws:ecs:clusterName`，做為`EncryptionContext`登入 CloudTrail 的一部分。

以下是 `GenerateDataKeyWithoutPlaintext`、`GenerateDataKeyWithoutPlaintext (DryRun)`、`CreateGrant (DryRun)`、 和 的 CloudTrail 事件範例 `CreateGrantRetireGrant`（使用您自己的值取代*##*值）。

GenerateDataKeyWithoutPlaintext

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AWSService",
        "invokedBy": "ec2-frontend-api.amazonaws.com"
    },
    "eventTime": "2024-04-23T18:08:13Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "GenerateDataKeyWithoutPlaintext",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "ec2-frontend-api.amazonaws.com",
    "userAgent": "ec2-frontend-api.amazonaws.com",
    "requestParameters": {
```

```
        "numberOfBytes": 64,
        "keyId": "arn:aws:kms:us-west-2:account-id:key/a1b2c3d4-5678-90ab-cdef-
EXAMPLE11111",
        "encryptionContext": {
            "aws:ecs:clusterAccount": "account-id",
            "aws:ebs:id": "vol-xxxxxxx",
            "aws:ecs:clusterName": "cluster-name"
        }
    },
    "responseElements": null,
    "requestID": "a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",
    "eventID": "a1b2c3d4-5678-90ab-cdef-EXAMPLE33333",
    "readOnly": true,
    "resources": [
        {
            "accountId": "AWS Internal",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-west-2:account-id:key/a1b2c3d4-5678-90ab-cdef-
EXAMPLE11111"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "account-id",
    "sharedEventID": "a1b2c3d4-5678-90ab-cdef-EXAMPLEaaaaa",
    "eventCategory": "Management"
}
```


GenerateDataKeyWithoutPlaintext (DryRun)


```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AWSService",
        "invokedBy": "fargate.amazonaws.com"
    },
    "eventTime": "2024-04-23T18:08:11Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "GenerateDataKeyWithoutPlaintext",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "fargate.amazonaws.com",
    "userAgent": "fargate.amazonaws.com",
    "errorCode": "DryRunOperationException",
```

```
    "errorMessage": "The request would have succeeded, but the DryRun option is
 set.",
    "requestParameters": {
        "keyId": "arn:aws:kms:us-west-2:account-id:key/a1b2c3d4-5678-90ab-cdef-
EXAMPLE11111",
        "dryRun": true,
        "numberOfBytes": 64,
        "encryptionContext": {
            "aws:ecs:clusterAccount": "account-id",
            "aws:ecs:clusterName": "cluster-name"
        }
    },
    "responseElements": null,
    "requestID": "a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",
    "eventID": "a1b2c3d4-5678-90ab-cdef-EXAMPLE33333",
    "readOnly": true,
    "resources": [
        {
            "accountId": "AWS Internal",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-west-2:account-id:key/a1b2c3d4-5678-90ab-cdef-
EXAMPLE11111"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "account-id",
    "sharedEventID": "a1b2c3d4-5678-90ab-cdef-EXAMPLEaaaaa",
    "eventCategory": "Management"
}
```

CreateGrant

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AWSService",
        "invokedBy": "ec2-frontend-api.amazonaws.com"
    },
    "eventTime": "2024-04-23T18:08:13Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "CreateGrant",
    "awsRegion": "us-west-2",
```

```
    "sourceIPAddress": "ec2-frontend-api.amazonaws.com",
    "userAgent": "ec2-frontend-api.amazonaws.com",
    "requestParameters": {
        "keyId": "arn:aws:kms:us-west-2:account-id:key/a1b2c3d4-5678-90ab-cdef-
EXAMPLE11111",
        "granteePrincipal": "fargate.us-west-2.amazonaws.com",
        "operations": [
            "Decrypt"
        ],
        "constraints": {
            "encryptionContextSubset": {
                "aws:ecs:clusterAccount": "account-id",
                "aws:ebs:id": "vol-xxxx",
                "aws:ecs:clusterName": "cluster-name"
            }
        },
        "retiringPrincipal": "ec2.us-west-2.amazonaws.com"
    },
    "responseElements": {
        "grantId":
 "e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855",
        "keyId": "arn:aws:kms:us-west-2:account-id:key/a1b2c3d4-5678-90ab-cdef-
EXAMPLE11111"
    },
    "requestID": "a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",
    "eventID": "a1b2c3d4-5678-90ab-cdef-EXAMPLE33333",
    "readOnly": false,
    "resources": [
        {
            "accountId": "AWS Internal",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-west-2:account-id:key/a1b2c3d4-5678-90ab-cdef-
EXAMPLE11111"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "account-id",
    "sharedEventID": "a1b2c3d4-5678-90ab-cdef-EXAMPLEaaaaa",
    "eventCategory": "Management"
}
```

CreateGrant (DryRun)

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AWSService",
        "invokedBy": "fargate.amazonaws.com"
    },
    "eventTime": "2024-04-23T18:08:11Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "CreateGrant",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "fargate.amazonaws.com",
    "userAgent": "fargate.amazonaws.com",
    "errorCode": "DryRunOperationException",
    "errorMessage": "The request would have succeeded, but the DryRun option is
 set.",
    "requestParameters": {
        "keyId": "arn:aws:kms:us-west-2:account-id:key/a1b2c3d4-5678-90ab-cdef-
EXAMPLE11111",
        "granteePrincipal": "fargate.us-west-2.amazonaws.com",
        "dryRun": true,
        "operations": [
            "Decrypt"
        ],
        "constraints": {
            "encryptionContextSubset": {
                "aws:ecs:clusterAccount": "account-id",
                "aws:ecs:clusterName": "cluster-name"
            }
        }
    },
    "responseElements": null,
    "requestID": "a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",
    "eventID": "a1b2c3d4-5678-90ab-cdef-EXAMPLE33333",
    "readOnly": false,
    "resources": [
        {
            "accountId": "AWS Internal",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-west-2:account-id:key/a1b2c3d4-5678-90ab-cdef-
EXAMPLE11111"
        }
    ],
```

```
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "account-id",
    "sharedEventID": "a1b2c3d4-5678-90ab-cdef-EXAMPLEaaaaa",
    "eventCategory": "Management"
}
```

RetireGrant

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AWSService",
        "invokedBy": "AWS Internal"
    },
    "eventTime": "2024-04-20T18:37:38Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "RetireGrant",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "AWS Internal",
    "userAgent": "AWS Internal",
    "requestParameters": null,
    "responseElements": {
        "keyId": "arn:aws:kms:us-west-2:account-id:key/a1b2c3d4-5678-90ab-cdef-
EXAMPLE11111"
    },
    "additionalEventData": {
        "grantId":
 "e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855"
    },
    "requestID": "a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",
    "eventID": "a1b2c3d4-5678-90ab-cdef-EXAMPLE33333",
    "readOnly": false,
    "resources": [
        {
            "accountId": "AWS Internal",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-west-2:account-id:key/a1b2c3d4-5678-90ab-cdef-
EXAMPLE11111"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
```

```
    "recipientAccountId": "account-id",
    "sharedEventID": "a1b2c3d4-5678-90ab-cdef-EXAMPLEaaaaa",
    "eventCategory": "Management"
}
```

Amazon ECS AWS Fargate 的任務淘汰和維護

AWS 負責維護 AWS Fargate 的基礎基礎設施。 AWS 決定何時需要將平台版本修訂取代為基礎設施的新修訂。這稱為任務淘汰。在平台版本修訂淘汰時 AWS 傳送任務淘汰通知。我們會定期更新我們支援的平台版本，以引進新的修訂，其中包含 Fargate 執行期軟體的更新，以及作業系統和容器執行期等基礎相依性。較新的修訂版本推出後，我們會淘汰較舊的修訂版本，以確保所有客戶工作負載都執行於 Fargate 平台版本的最新修訂版本。當修訂版本淘汰時，在該修訂版本上執行的所有任務都會停止。

Amazon ECS 任務可以分類為服務任務和獨立任務。服務任務會部署為服務的一部分，並由 Amazon ECS 排程控制。如需詳細資訊，請參閱Amazon ECS 服務。獨立任務是由 Amazon ECS RunTask API 啟動的任務，無論是直接或外部排程器啟動，例如排程任務（由 Amazon EventBridge 啟動）AWS Batch，或 AWS Step Functions。您不需要採取任何動作來回應服務任務的任務淘汰，因為 Amazon ECS 排程器會自動取代任務。

對於獨立任務，您可能需要執行額外的處理，以回應任務淘汰。如需詳細資訊，請參閱Amazon ECS 可以自動處理獨立任務嗎？。

對於服務任務，您不需要採取任何動作來淘汰任務，除非您想要先取代這些任務 AWS 。當 Amazon ECS 排程器停止任務時，它會使用 `maximumPercent`並啟動新的任務，以嘗試維持服務所需的計數。為了將 AWS Fargate 任務淘汰的影響降至最低，您應該在部署工作負載時遵循 Amazon ECS 最佳實務。使用 REPLICA 服務排程器的服務`maximumPercent`預設值為 200%。因此，當 AWS Fargate 開始淘汰任務時，Amazon ECS 會先排程新任務，然後等待它執行，然後再淘汰舊任務。當您將`maximumPercent`值設定為 100% 時，Amazon ECS 會先停止任務，然後取代任務。

對於獨立任務淘汰， 會在任務淘汰日期或之後 AWS 停止任務。當任務停止時，Amazon ECS 不會啟動替代任務。如果您需要這些任務繼續執行，您需要在通知中指定的時間之前停止執行中的任務並啟動替代任務。因此，我們建議客戶監控獨立任務的狀態，並視需要實作邏輯以取代已停止的任務。

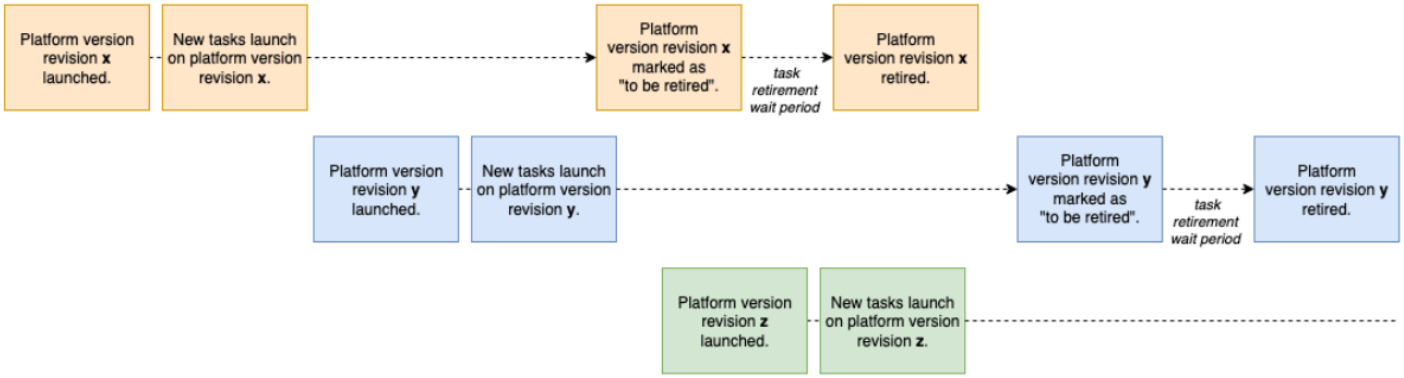
當任務在任何場景下停止時，您都可以執行 `describe-tasks`。回應中的 `stoppedReason` 是 `ECS is performing maintenance on the underlying infrastructure hosting the task`。

當有新的平台版本修訂需要取代為新的修訂時，任務維護即適用。如果基礎 Fargate 主機發生問題，Amazon ECS 會在沒有任務淘汰通知的情況下取代主機。

任務淘汰通知概觀

當 將平台版本修訂 AWS 標記為需要淘汰時，我們會識別所有區域中在該平台版本修訂上執行的所有任務。然後，我們會為每個區域傳送每個帳戶的通知，強調受影響的任務或服務，以及開始淘汰的日期。

下圖顯示 Fargate 平台版本修訂從新修訂啟動到平台修訂淘汰的生命週期。



以下資訊提供詳細資訊。

- 啟動新的平台版本修訂後，所有新任務都會排程在此修訂版本上。
- 已排程和執行的現有任務，會維持在最初在任務期間置放的修訂中，不會遷移至新的修訂。
- 新任務，例如作為服務或 Fargate 任務淘汰更新的一部分，會放置在啟動時可用的最新平台版本修訂中。

任務淘汰通知會透過 AWS Health Dashboard 和電子郵件傳送至已註冊的電子郵件地址，並包含下列資訊：

- 任務淘汰日期 - 任務將在此日期或之後停止。
- 若為獨立任務，則為任務的 ID。
- 若為服務任務，則為執行服務所在之叢集的 ID 以及服務的 ID。
- 您需要採取的後續步驟。

一般而言，我們會針對每個 中的服務和獨立任務傳送每個通知 AWS 區域。不過，在某些情況下，您可能會針對每個任務類型收到多個事件，例如，當有太多任務需要淘汰，而在我們的通知機制中超過限制時。

您可以透過以下方法確定已排程要淘汰的任務：

- 的 AWS Health Dashboard

  AWS Health 通知可以透過 Amazon EventBridge 傳送至封存儲存，例如 Amazon Simple Storage Service、執行 AWS Lambda 函數等自動動作，或其他通知系統，例如 Amazon Simple Notification Service。如需詳細資訊，請參閱使用 Amazon EventBridge 監控 AWS Health 事件。如需將通知傳送至 Amazon Chime、Slack 或 Microsoft Teams 的組態範例，請參閱 GitHub 上的 AWS Health Aware 儲存庫。

  以下是 EventBridge 事件範例。

```
{
    "version": "0",
    "id": "3c268027-f43c-0171-7425-1d799EXAMPLE",
    "detail-type": "AWS Health Event",
    "source": "aws.health",
    "account": "123456789012",
    "time": "2023-08-16T23:18:51Z",
    "region": "us-east-1",
    "resources": [
        "cluster|service",
        "cluster|service"
    ],
    "detail": {
        "eventArn": "arn:aws:health:us-east-1::event/ECS/
AWS_ECS_TASK_PATCHING_RETIREMENT/AWS_ECS_TASK_PATCHING_RETIREMENT_test1",
        "service": "ECS",
        "eventScopeCode": "ACCOUNT_SPECIFIC",
        "communicationId":
 "7988399e2e6fb0b905ddc88e0e2de1fd17e4c9fa60349577446d95a18EXAMPLE",
        "lastUpdatedTime": "Wed, 16 Aug 2023 23:18:52 GMT",
        "eventRegion": "us-east-1",
        "eventTypeCode": "AWS_ECS_TASK_PATCHING_RETIREMENT",
        "eventTypeCategory": "scheduledChange",
        "startTime": "Wed, 16 Aug 2023 23:18:51 GMT",
        "endTime": "Fri, 18 Aug 2023 23:18:51 GMT",
        "eventDescription": [
            {
                "language": "en_US",
                "latestDescription": "\\nA software update has been deployed to
 Fargate which includes CVE patches or other critical patches. No action is required
 on your part. All new tasks launched automatically uses the latest software
 version. For existing tasks, your tasks need to be restarted in order for these
 updates to apply. Your tasks running as part of the following ECS Services will
```

```
 be automatically updated beginning Wed, 16 Aug 2023 23:18:51 GMT.\\n\\nAfter Wed,
 16 Aug 2023 23:18:51 GMT, the ECS scheduler will gradually replace these tasks,
 respecting the deployment settings for your service. Typically, services should
 see little to no interruption during the update and no action is required. When AWS
 stops tasks, AWS uses the minimum healthy percent (1) and launches a new task in
 an attempt to maintain the desired count for the service. By default, the minimum
 healthy percent of a service is 100 percent, so a new task is started first before
 a task is stopped. Service tasks are routinely replaced in the same way when
 you scale the service or deploy configuration changes or deploy task definition
 revisions. If you would like to control the timing of this restart you can update
 the service before Wed, 16 Aug 2023 23:18:51 GMT, by running the update-service
 command from the ECS command-line interface specifying force-new-deployment for
 services using Rolling update deployment type. For example:\\n\\n$ aws ecs update-
service -service service_name \\\n--cluster cluster_name -force-new-deployment\
\n\\nFor services using Blue/Green deployment type with AWS CodeDeploy:\\nPlease
 refer to create-deployment document (2) and create new deployment using same task
 definition revision.\\n\\nFor further details on ECS deployment types, please
 refer to ECS Deployment Developer Guide (1).\\nFor further details on Fargate's
 update process, please refer to the AWS Fargate User Guide (3).\\nIf you have
 any questions or concerns, please contact AWS Support (4).\\n\\n(1) https://
docs.aws.amazon.com/AmazonECS/latest/developerguide/deployment-types.html\\n(2)
 https://docs.aws.amazon.com/cli/latest/reference/deploy/create-deployment.html\\n(3)
 https://docs.aws.amazon.com/AmazonECS/latest/userguide/task-maintenance.html\\n(4)
 https://aws.amazon.com/support\\n\\nA list of your affected resources(s) can be
 found in the 'Affected resources' tab in the 'Cluster/ Service' format in the AWS
 Health Dashboard. \\n\\n"
            }
        ],
        "affectedEntities": [
            {
                "entityValue": "cluster|service"
            },
            {
                "entityValue": "cluster|service"
            }
        ]
    }
}
```

- 電子郵件

  電子郵件會傳送至 AWS 帳戶 ID 的已註冊電子郵件。

如需如何準備任務淘汰的資訊，請參閱 準備在 Amazon ECS 上淘汰 AWS Fargate 任務 。

我可以選擇退出任務淘汰嗎？

否。作為 AWS 共同責任模型的一部分， AWS 負責管理和維護 的基礎基礎設施 AWS Fargate。這包括執行定期平台更新，以確保安全性和穩定性。這些更新由 自動套用 AWS ，而且不是客戶可以選擇退出的項目。與在 EC2 執行個體上執行工作負載 AWS Fargate 相比，這是使用 的主要優點，維護基礎平台的責任由 處理 AWS。此模型可讓您專注於應用程式，而非基礎設施維護。透過自動套用這些平台更新， AWS 能夠讓 Fargate 環境up-to-date狀態和安全，而無需您作為客戶採取任何動作。這有助於提供可靠且安全的容器化環境，以便在 Fargate 上執行工作負載。

我可以透過其他服務取得任務淘汰通知 AWS 嗎？

AWS 會將任務淘汰通知傳送至 ， AWS Health Dashboard 以及傳送至 上的主要電子郵件聯絡人 AWS 帳戶。 AWS Health Dashboard 提供多種與其他 AWS 服務的整合，包括 EventBridge。您可以使用 EventBridge 來自動化通知的可見性（例如， 將訊息轉送至 ChatOps 工具）。如需詳細資訊，請參閱解決方案概觀：擷取任務淘汰通知。

我可以在任務排程後變更任務淘汰嗎？

否。排程是根據任務淘汰等待時間，預設為 7 天。如果您需要更多時間，您可以選擇將等待期設定為 14 天。如需詳細資訊，請參閱步驟 2：擷取任務淘汰通知以提醒團隊並採取動作。此組態的變更會套用至未來排程的淘汰。目前排程的淘汰不會受到影響。如果您有任何其他疑慮，請聯絡 支援。

Amazon ECS 如何處理屬於服務一部分的任務？

對於服務任務，您不需要採取任何動作來回應任務淘汰，除非您想要先取代這些任務 AWS 。當 Amazon ECS 排程器停止任務時，它會使用運作狀態最低百分比，並啟動新的任務，以嘗試維持服務所需的計數。為了將 Fargate 任務淘汰的影響降至最低，工作負載應按照 Amazon ECS 最佳實務進行部署。例如，將無狀態應用程式部署為 Amazon ECS 服務時，例如 Web 或 API 伺服器，客戶應部署多個任務複本，並將 minimumHealthyPercent 設定為 100%。根據預設，服務的運作狀態最低百分比為 100%。因此，當 Fargate 開始淘汰任務時，Amazon ECS 會先排程新任務，並等待它執行，然後再淘汰舊任務。當您擴展服務、部署組態變更或部署任務定義修訂時，服務任務會定期取代為任務淘汰的一部分。若要準備任務淘汰程序，請參閱 準備在 Amazon ECS 上淘汰 AWS Fargate 任務 。

Amazon ECS 可以自動處理獨立任務嗎？

No. AWS 無法為由 RunTask、排程任務（例如透過 EventBridge 排程器） AWS Batch或 啟動的獨立任務建立替代任務 AWS Step Functions。Amazon ECS 只會管理屬於服務一部分的任務。

準備在 Amazon ECS 上淘汰 AWS Fargate 任務

為了準備任務淘汰，請執行下列操作：

1. 設定任務淘汰等待期。
2. 擷取任務淘汰通知以通知團隊成員。
3. 您無法控制任務淘汰的確切時間，但是，您可以使用強制部署選項更新服務來控制任務的取代。

步驟 1：設定任務等待時間

您可以設定 Fargate 開始任務淘汰的時間。對於需要立即套用更新的工作負載，請選擇立即設定 (0)。當您需要更多控制項時，例如當任務只能在特定時段中停止時，請設定 7 日 (7) 或 14 日 (14) 選項。

我們建議您選擇較短的等待期，以便更快獲得更新的平台版本修訂版。

透過執行 `put-account-setting-default`或 `put-account-setting` 做為根使用者或管理使用者來設定等待期。對 `name` 使用 `fargateTaskRetirementWaitPeriod` 選項，並將 `value` 選項設定為以下值之一：

- `0` - AWS 傳送通知，並立即開始淘汰受影響的任務。
- `7` - AWS 傳送通知，並等待 7 個日曆天，再開始淘汰受影響的任務。
- `14` - AWS 傳送通知，並等待 14 個日曆日，然後再開始淘汰受影響的任務。

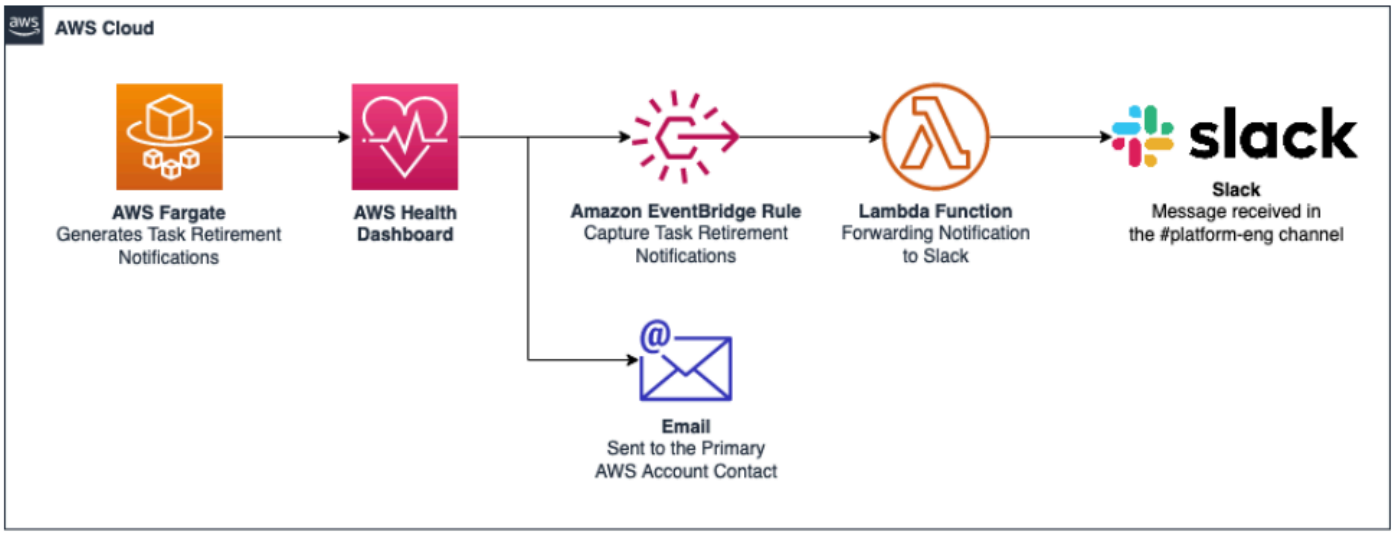
預設值是 7 天。

如需詳細資訊，請參閱《Amazon Elastic Container Service API 參考》中的 put-account-setting-default 和 put-account-setting。

步驟 2：擷取任務淘汰通知以提醒團隊並採取動作

當即將淘汰任務時， AWS 會將任務淘汰通知傳送至 AWS Health 儀表板，以及傳送至 上的主要電子郵件聯絡人 AWS 帳戶。 AWS Health 儀表板提供許多與其他 AWS 服務的整合，包括 Amazon EventBridge。您可以使用 EventBridge 從任務淘汰通知中建置自動化，例如透過將訊息轉送到 ChatOps 工具來提高即將淘汰的可見性。 AWS Health Aware 是一種資源，可顯示 AWS Health 儀表板的強大功能，以及如何將通知分佈到整個組織中。您可以將任務淘汰通知轉送到聊天應用程式，例如 Slack。

下圖顯示解決方案概觀。



以下資訊提供詳細資訊。

- Fargate 會將任務淘汰通知 AWS Health 傳送至儀表板。
- Dashboard AWS Health 會將郵件傳送至 上的主要電子郵件聯絡人 AWS 帳戶，並通知 EventBridge。
- EventBridge 具有擷取淘汰通知的規則。

  尋找具有事件詳細資訊類型的事件的規則：`"AWS Health Event" and the Event Detail Type Code: "AWS_ECS_TASK_PATCHING_RETIREMENT"`
- 此規則會觸發 Lambda 函數，使用 Slack 傳入 Webhook 將資訊轉送至 Slack。如需詳細資訊，請參閱傳入 Webhook。

如需程式碼範例，請參閱在 Github 上擷取 AWS Fargate 任務淘汰通知。

步驟 3：控制任務的取代

您無法控制任務淘汰的確切時間，但您可以定義等待時間。如果您想要控制依自己的排程取代任務，您可以擷取任務淘汰通知，先了解任務淘汰日期。然後，您可以重新部署服務以啟動替代任務，並同樣取代任何獨立任務。對於使用滾動部署的服務，您可以在淘汰開始時間之前使用 `update-service` 搭配 `force-deployment`選項來更新服務。

下列`update-service`範例使用 `force-deployment`選項。

```
aws ecs update-service –-service service_name \
```

```
    --cluster cluster_name \
     --force-new-deployment
```

對於使用藍/綠部署的服務，您需要在其中建立新的部署 AWS CodeDeploy。如需有關如何建立部署的資訊，請參閱 AWS Command Line Interface 參考中的建立部署。

支援 AWS Fargate 上 Amazon ECS 的區域

您可以使用下表來驗證區域對 AWS Fargate 上的 Linux 容器和 AWS Fargate 上的 Windows 容器的支援。

AWS Fargate 上的 Linux 容器

下列 AWS Fargate 支援 上的 Amazon ECS Linux 容器 AWS 區域。適用時，支援的可用區域 ID 會註明。

區域名稱	區域
美國東部 (俄亥俄)	us-east-2
美國東部 (維吉尼亞北部)	us-east-1
美國西部 (加利佛尼亞北部)	us-west-1 (僅 usw1-az1 和 usw1-az3)
美國西部 (奧勒岡)	us-west-2
非洲 (開普敦)	af-south-1
亞太區域 (香港)	ap-east-1
亞太區域 (孟買)	ap-south-1
亞太區域 (東京)	ap-northeast-1 (僅 apne1-az1 、apne1-az2 和 apne1-az4)
亞太區域 (首爾)	ap-northeast-2
亞太區域 (大阪)	ap-northeast-3
亞太區域 (海德拉巴)	ap-south-2

區域名稱	區域
亞太區域 (新加坡)	ap-southeast-1
亞太區域 (雪梨)	ap-southeast-2
亞太區域 (雅加達)	ap-southeast-3
亞太區域 (墨爾本)	ap-southeast-4
亞太地區 (馬來西亞)	ap-southeast-5
加拿大 (中部)	ca-central-1
加拿大西部 (卡加利)	ca-west-1
中國 (北京)	cn-north-1 (僅 `cnn1-az1` 和 `cnn1-az2`)
中國 (寧夏)	cn-northwest-1
歐洲 (法蘭克福)	eu-central-1
歐洲 (蘇黎世)	eu-central-2
歐洲 (愛爾蘭)	eu-west-1
歐洲 (倫敦)	eu-west-2
歐洲 (巴黎)	eu-west-3
歐洲 (米蘭)	eu-south-1
歐洲 (西班牙)	eu-south-2
歐洲 (斯德哥爾摩)	eu-north-1
南美洲 (聖保羅)	sa-east-1
以色列 (特拉維夫)	il-central-1
中東 (巴林)	me-south-1

區域名稱	區域
中東 (阿拉伯聯合大公國)	me-central-1
AWS GovCloud （美國東部）	us-gov-east-1
AWS GovCloud （美國西部）	us-gov-west-1

AWS Fargate 上的 Windows 容器

下列 AWS Fargate 支援 上的 Amazon ECS Windows 容器 AWS 區域。適用時，支援的可用區域 ID 會註明。

區域名稱	區域
美國東部 (俄亥俄)	us-east-2
美國東部 (維吉尼亞北部)	us-east-1 (只有 `use1-az1`、`use1-az2`、`use1-az4`、`use1-az5` 和 `use1-az6`)
美國西部 (加利佛尼亞北部)	us-west-1 (僅 `usw1-az1` 和 `usw1-az3`)
美國西部 (奧勒岡)	us-west-2
非洲 (開普敦)	af-south-1
亞太區域 (香港)	ap-east-1
亞太區域 (孟買)	ap-south-1
亞太區域 (海德拉巴)	ap-south-2
亞太區域 (大阪)	ap-northeast-3
亞太區域 (首爾)	ap-northeast-2
亞太區域 (新加坡)	ap-southeast-1
亞太區域 (雪梨)	ap-southeast-2

區域名稱	區域
亞太區域 (墨爾本)	ap-southeast-4
亞太地區 (馬來西亞)	ap-southeast-5
亞太區域 (東京)	ap-northeast-1 (僅 apne1-az1 、apne1-az2 和 apne1-az4)
加拿大 (中部)	ca-central-1 (僅 cac1-az1 和 cac1-az2)
加拿大西部 (卡加利)	ca-west-1
中國 (北京)	cn-north-1 (僅 cnn1-az1 和 cnn1-az2)
中國 (寧夏)	cn-northwest-1
歐洲 (法蘭克福)	eu-central-1
歐洲 (蘇黎世)	eu-central-2
歐洲 (愛爾蘭)	eu-west-1
歐洲 (倫敦)	eu-west-2
歐洲 (巴黎)	eu-west-3
歐洲 (米蘭)	eu-south-1
歐洲 (西班牙)	eu-south-2
歐洲 (斯德哥爾摩)	eu-north-1
南美洲 (聖保羅)	sa-east-1
以色列 (特拉維夫)	il-central-1
中東 (阿拉伯聯合大公國)	me-central-1
中東 (巴林)	me-south-1

建構 Amazon 的解決方案 ECS

使用 Amazon 之前ECS，您需要先做出容量、聯網、帳戶設定和記錄的決策，才能正確設定 Amazon ECS 資源。

容量

容量是容器執行所在的基礎設施。以下是選項：

- Amazon EC2執行個體
- 無伺服器 (AWS Fargate)
- 內部部署虛擬機器或伺服器

您可以在建立叢集時指定基礎設施。您也可以在註冊任務定義時指定基礎設施類型。任務定義將基礎設施稱為「啟動類型」。當您執行獨立任務或部署服務時，您也可以使用 啟動類型。如需啟動類型選項的相關資訊，請參閱 Amazon ECS啟動類型。

聯網

AWS 資源是在子網路中建立的。當您使用EC2執行個體時，Amazon 會在您建立叢集時指定的子網路中ECS啟動執行個體。您的任務在執行個體子網路中執行。對於 Fargate 或內部部署虛擬機器，您可以在執行任務或建立服務時指定子網路。

視您的應用程式而定，子網路可以是私有或公有子網路，子網路可以是下列任何 AWS 資源：

- 可用區域
- 本機區域
- Wavelength 區域
- AWS 區域
- AWS Outposts

如需詳細資訊，請參閱 共用子網路、本機區域和 Wavelength 區域中的 Amazon ECS 應用程式 或 上的 Amazon Elastic Container Service AWS Outposts。

您可以使用下列其中一種方法，讓您的應用程式連線至網際網路：

- 具有網際網路閘道的公有子網路

  當您的公有應用程式需要大量頻寬或最低延遲時，請使用公有子網路。適用的案例包括影片串流和遊戲服務。

- 具有NAT閘道的私有子網路

  當您想要保護容器免於直接外部存取時，請使用私有子網路。適用的案例包括付款處理系統或儲存使用者資料和密碼的容器。

- AWS PrivateLink

  使用 在 VPCs、 AWS 服務與內部部署網路之間 AWS PrivateLink 建立私有連線，而不會將您的流量暴露到公有網際網路。

功能存取

您可以使用 Amazon ECS帳戶設定來存取下列功能：

- Container Insights

  CloudWatch Container Insights 會從容器化應用程式和微服務收集、彙總和摘要指標和日誌。這些指標包括 資源的使用率，例如 CPU、記憶體、磁碟和網路。

- `awsvpc` 中繼

  對於某些EC2執行個體類型，您可以在新啟動的容器執行個體上提供額外的網路介面 (ENIs)。

- 標記授權

  使用者必須擁有建立 資源之動作的許可，例如 `ecsCreateCluster`。如果在資源建立動作中指定標籤， 會對`ecs:TagResource`動作 AWS 執行其他授權，以驗證使用者或角色是否具有建立標籤的許可。

- Fargate FIPS-140 合規

  Fargate 支援聯邦資訊處理標準 (FIPS-140)，該標準指定了保護敏感資訊之密碼編譯模組的安全要求。這是目前美國和加拿大政府標準，適用於必須符合美國聯邦資訊安全管理法案 (FISMA) 或聯邦風險與授權管理計劃 (Fed) 的系統RAMP。

- Fargate 任務淘汰時間變更

  您可以設定 Fargate 任務淘汰以進行修補之前的等待期。

- 雙堆疊 VPC

允許任務透過 IPv4、 IPv6或兩者進行通訊。

- Amazon Resource Name (ARN) 格式

  標記授權等特定功能需要新的 Amazon Resource Name (ARN) 格式。

如需詳細資訊，請參閱使用帳戶設定存取 Amazon ECS功能。

IAM 角色

IAM 角色是您可以在帳戶中建立的IAM身分，具有特定許可。在 Amazon 中ECS，您可以建立角色，以授予許可給 Amazon ECS 資源，例如容器或服務。

有些 Amazon ECS功能需要 角色。如需詳細資訊，請參閱IAM Amazon 的 角色 ECS。

日誌

記錄和監控是維護 Amazon ECS工作負載可靠性、可用性和效能的重要層面。以下是可用的選項：

- Amazon CloudWatch 日誌 - 路由日誌至 Amazon CloudWatch
- FireLens for Amazon ECS - 將日誌路由到服務或 AWS AWS Partner Network 目的地，以進行日誌儲存和分析。 AWS Partner Network 是一個由合作夥伴組成的全球社群，利用計劃、專業知識和資源來建置、行銷和銷售客戶產品。

Amazon ECS啟動類型

任務定義啟動類型會定義任務可以執行的容量，例如 AWS Fargate。

選擇啟動類型後，Amazon ECS 會驗證您設定的任務定義參數是否與啟動類型搭配使用。

Fargate

Fargate 是一種無伺服器運算引擎， pay-as-you-go可讓您專注於建置應用程式，而無需管理伺服器。當您選擇 Fargate 時，您不需要管理EC2基礎設施。您只需要建置容器映像，並定義您要執行應用程式的叢集。Fargate 已與 AWS 服務進行原生整合，包括：

- Amazon VPC

- Auto Scaling
- Elastic Load Balancing
- IAM
- Secrets Manager

您可以使用 Fargate 進行比 更多的控制，EC2因為您選取應用程式所需的確切CPU和記憶體。Fargate 會處理容量的擴展，因此您不需要擔心流量遽增。這表示 Fargate 的操作工作較少。

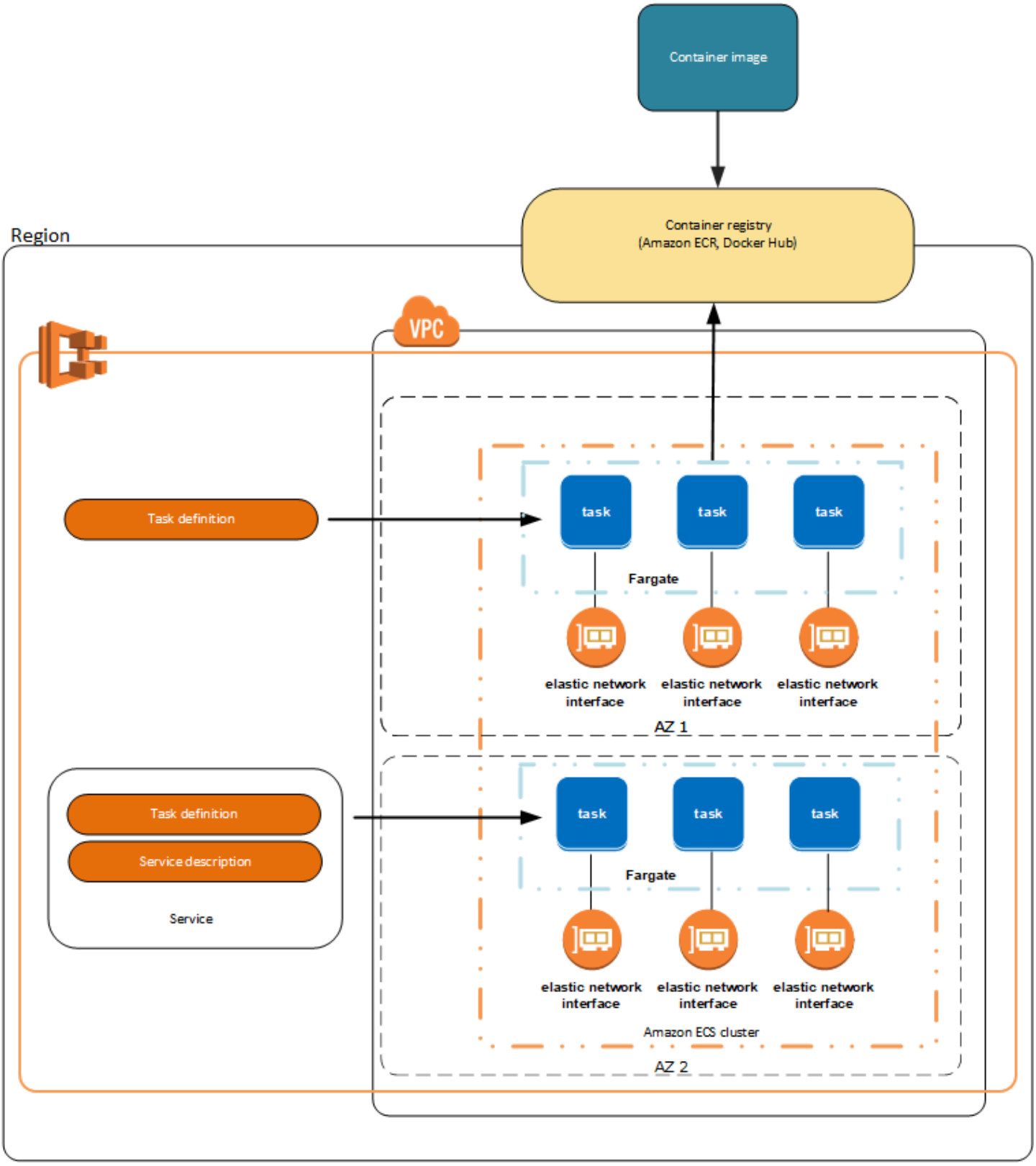
Fargate 符合合規計劃的標準，包括 PCI、140-2、Fed FIPS RAMP和 HIPAA。如需詳細資訊，請參閱AWS 依合規計劃範圍中的服務。

Fargate 適用於下列工作負載：

- 需要低廉營運開銷的大型工作負載
- 偶爾會爆量的小型工作負載
- 微小工作負載
- 批次工作負載

如需支援 Fargate 的區域的資訊，請參閱the section called “AWS Fargate 區域”。

下圖顯示一般架構。



如需 Fargate ECS上的 Amazon 詳細資訊，請參閱 AWS Fargate 適用於 Amazon ECS。

EC2

EC2 啟動類型適用於必須最佳化價格的大型工作負載。

在考慮如何使用 EC2 啟動類型建立任務定義和服務模型時，建議您考慮哪些程序必須一起執行，以及您可以如何擴展每個元件。

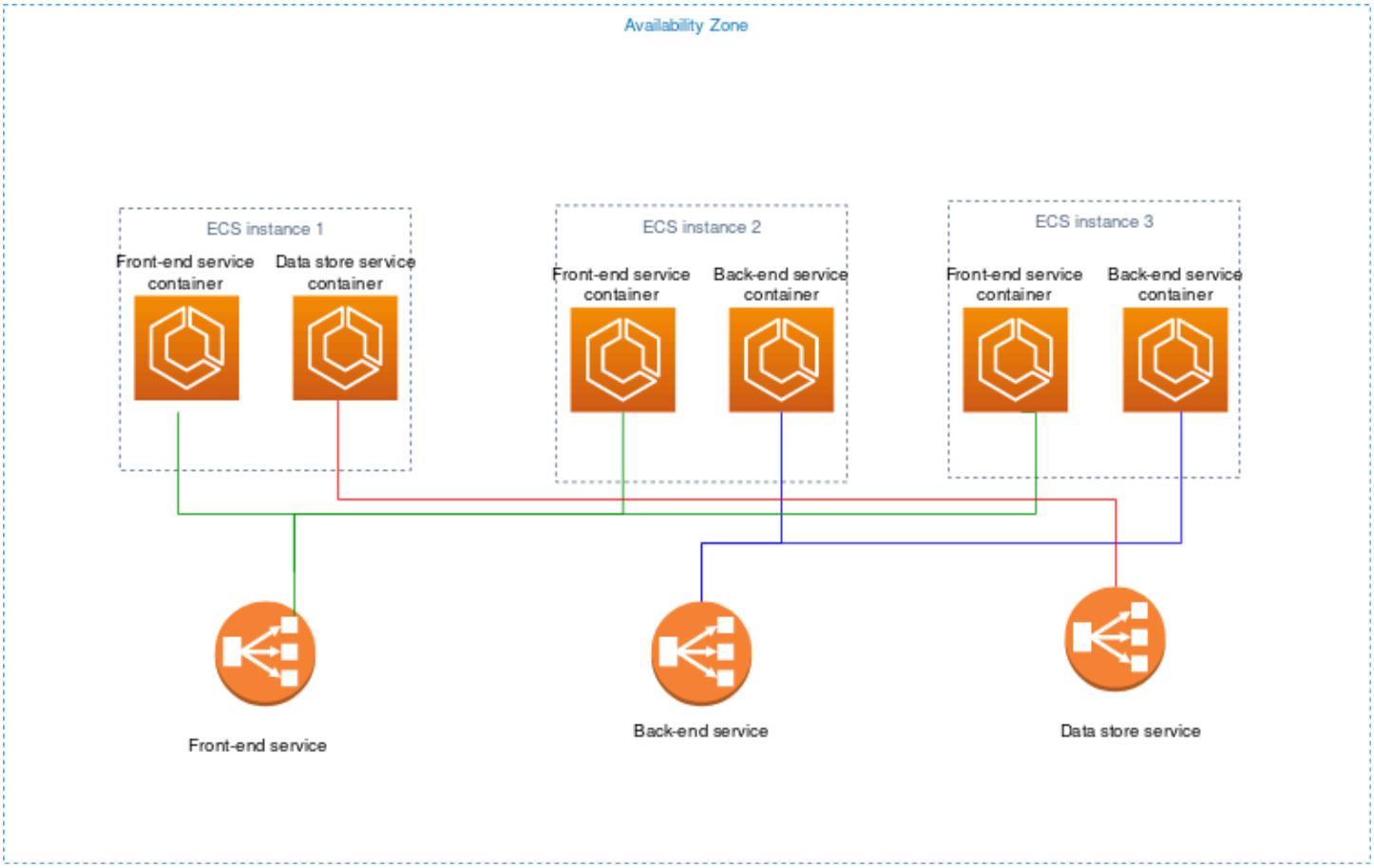
舉例來說，假設由下列元件組成的應用程式：

- 在網頁顯示資訊的前端服務
- APIs 為前端服務提供的後端服務
- 資料存放區

在此範例中，請建立將用於常見用途的容器分為同一群組的任務定義。將不同的元件分隔為多個單獨的任務定義。下列的範例叢集具有在三個前端服務容器上執行的三個容器執行個體、兩個後端服務容器，以及一個資料存放區服務容器。

您可以將任務定義中的相關容器分組，例如必須一起執行的連結容器。例如，將日誌串流容器新增到您的前端服務，並將此服務包含在相同的任務定義中。

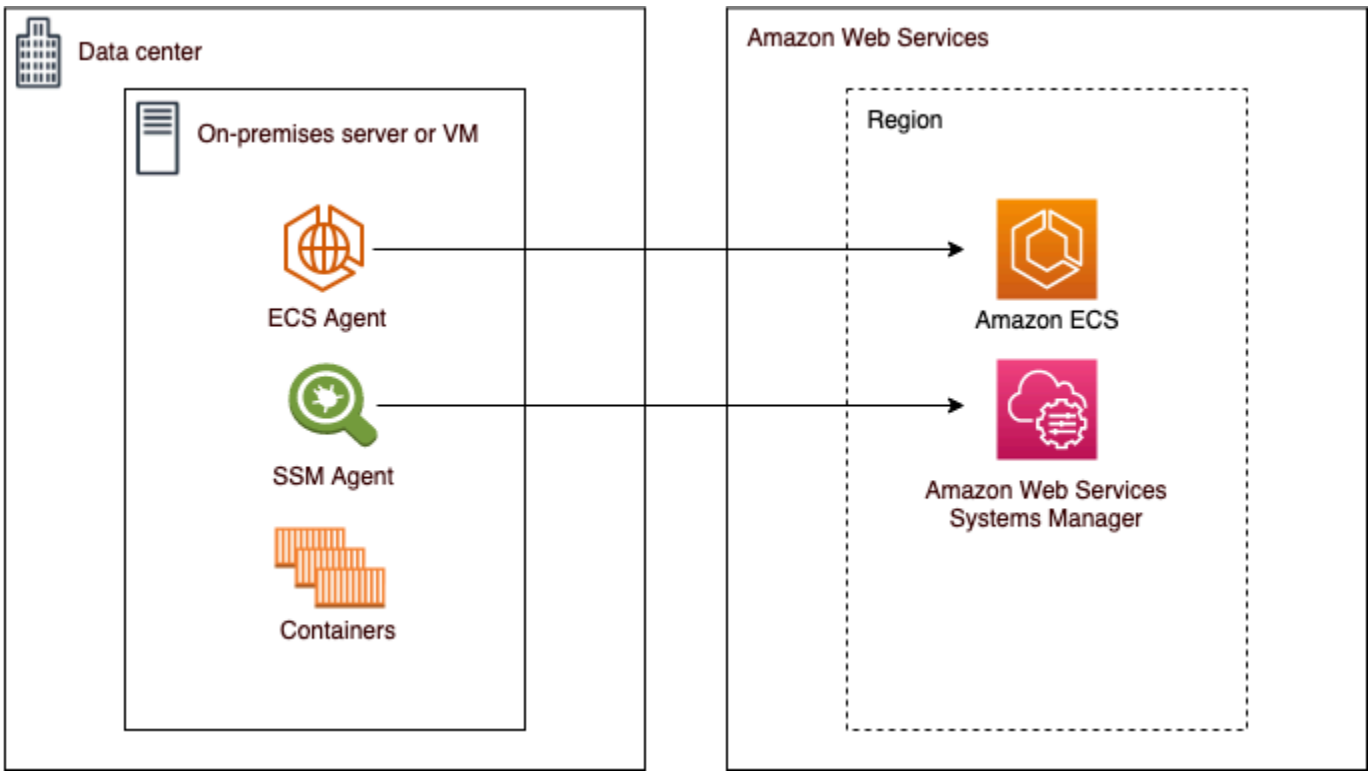
在您擁有任務定義之後，您可以從中建立服務，以維護您所需任務的可用性。如需詳細資訊，請參閱使用 主控台建立 Amazon ECS 服務。在您的服務中，您可以建立容器與 Elastic Load Balancing 負載平衡器的關聯。如需詳細資訊，請參閱使用負載平衡來分發 Amazon ECS 服務流量。當您的應用程式需求變更時，您可以更新您的服務以調整所需任務的數量。或者，您也可以更新您的服務以部署任務中的新版容器。如需詳細資訊，請參閱使用主控台更新 Amazon ECS 服務。



外部 (Amazon ECS Anywhere)

Amazon ECS Anywhere 支援將內部部署伺服器或虛擬機器 (VM) 等外部執行個體註冊到您的 Amazon ECS叢集。外部執行個體已進行優化，可執行產生傳出流量或處理資料的應用程式。如果您的應用程式需要傳入流量，缺乏 Elastic Load Balancing 支援會使這些工作負載的執行效率降低。Amazon ECS 新增了新的EXTERNAL啟動類型，可用於在外部執行個體上建立服務或執行任務。

以下提供 Amazon ECS Anywhere 的高階系統架構概觀。您的內部部署伺服器已安裝 Amazon ECS代理程式和 SSM代理程式。



如需詳細資訊，請參閱外部啟動類型的 Amazon ECS 叢集。

共用子網路、本機區域和 Wavelength 區域中的 Amazon ECS 應用程式

Amazon ECS支援使用 Local Zones、Wavelength Zones 和 AWS Outposts 的工作負載，當需要低延遲或本機資料處理時。

- 您可以使用 Local Zones 作為 的延伸 AWS 區域 ，將資源放置在靠近最終使用者的多個位置。
- 您可以使用 Wavelength Zone 建置可為 5G 裝置與最終使用者提供極低延遲的應用程式。Wavelength 會將標準 AWS 運算和儲存服務部署到電信業者的 5G 網路邊緣。
- AWS Outposts 將原生、 AWS 服務基礎設施和操作模型帶入幾乎所有資料中心、主機代管空間或內部部署設施。

> **Important**
>
> Local Zones、Wavelength Zones 或 上的 AWS Fargate 工作負載 AWS Outposts 目前不支援 Amazon ECS on。

如需 Local Zones、Wavelength Zones 和 之間的差異資訊 AWS Outposts ，請參閱 中關於何時使用 AWS Wavelength、 AWS Local Zones 或需要低延遲或本機資料處理 AWS Outposts 的應用程式，我應該如何考慮 AWS Wavelength FAQs。

共用子網路

您可以使用VPC共用來與相同 內的其他 AWS 帳戶共用子網路 AWS Organizations。

您可以VPCs針對EC2啟動類型使用共用，並考量下列因素：

- VPC 子網路擁有者必須與參與者帳戶共用子網路，該帳戶才能將其用於 Amazon ECS 資源。
- 您無法為容器執行個體使用VPC預設安全群組，因為它屬於擁有者。此外，參與者無法使用其他參與者或擁有者擁有的安全群組啟動執行個體。
- 在共用子網路中，參與者和擁有者會分別控制每個各別帳戶內的安全群組。子網路擁有者可以查看由參與者建立的安全群組，但無法執行任何動作。如果子網路擁有者想要移除或修改這些安全群組，建立安全群組的參與者必須採取動作。
- 共用VPC擁有者無法檢視、更新或刪除參與者在共用子網路中建立的叢集。除了每個帳戶具有不同存取權VPC的資源之外，還有這項功能。如需詳細資訊，請參閱《Amazon VPC使用者指南》中的擁有者和參與者的責任和許可。

您可以VPCs針對 Fargate 啟動類型使用共用 ，並考量下列因素：

- VPC 子網路擁有者必須與參與者帳戶共用子網路，該帳戶才能將其用於 Amazon ECS 資源。
- 您無法建立服務或使用 的預設安全群組執行任務，VPC因為它屬於擁有者。此外，參與者無法使用其他參與者或擁有者擁有的安全群組建立服務或執行任務。
- 在共用子網路中，參與者和擁有者會分別控制每個各別帳戶內的安全群組。子網路擁有者可以查看由參與者建立的安全群組，但無法執行任何動作。如果子網路擁有者想要移除或修改這些安全群組，建立安全群組的參與者必須採取動作。
- 共用VPC擁有者無法檢視、更新或刪除參與者在共用子網路中建立的叢集。除了每個帳戶具有不同存取權VPC的資源之外，還有這項功能。如需詳細資訊，請參閱《Amazon VPC使用者指南》中的擁有者和參與者的責任和許可。

如需VPC子網路共用的詳細資訊，請參閱《Amazon VPC使用者指南》中的VPC與其他帳戶共用您的
。

本機區域

Local Zone 是 AWS 區域 的延伸，鄰近您的使用者。Local Zones 有自己的網際網路連線，並支援 AWS Direct Connect。在 Local Zone 中建立的資源可以為本機使用者提供低延遲的通訊服務。如需詳細資訊，請參閱 AWS Local Zones。

Local Zone 由一個區域代碼加上一個代表位置的識別符來表示 (例如 `us-west-2-lax-1a`)。

若要使用 Local Zone，您必須先選擇加入區域。選擇加入後，您必須在 Local Zone 中建立 Amazon VPC和子網路。

您可以啟動 Amazon EC2執行個體、Amazon FSx 檔案伺服器和 Application Load Balancer，以用於 Amazon ECS叢集和任務。

如需詳細資訊，請參閱 AWS Local Zones 使用者指南中的什麼是 Local Zones？。 AWS

Wavelength 區域

您可使用 AWS Wavelength 建置可為行動裝置與最終使用者提供極低延遲的應用程式。Wavelength 會將標準 AWS 運算和儲存服務部署到電信業者的 5G 網路邊緣。您可將 Amazon Virtual Private Cloud 延伸至一或多個 Wavelength Zone。然後，您可以使用 Amazon EC2執行個體之類的 AWS 資源，在 AWS 服務 區域中執行需要超低延遲和 連線的應用程式。

Wavelength Zone 是部署 Wavelength 基礎設施之電信業者位置的隔離區域。Wavelength 區域會繫結至 AWS 區域。Wavelength 區域是區域的邏輯延伸，並且由區域中的控制平面管理。

Wavelength Zone 由一個區域代碼加上一個代表 Wavelength Zone 的識別符來表示 (例如 `us-east-1-wl1-bos-wlz-1`)。

若要使用 Wavelength Zone，您必須先選擇加入區域。選擇加入後，您必須在 Wavelength 區域中建立 Amazon VPC和子網路。然後，您可以在 區域中啟動 Amazon EC2執行個體，以用於 Amazon ECS叢集和任務。

如需詳細資訊，請參閱 AWS Wavelength 開發人員指南中的 AWS Wavelength入門。

Wavelength Zones 無法全部使用 AWS 區域。如需支援 Wavelength 區域的區域相關資訊，請參閱 AWS Wavelength 開發人員指南中的可用 Wavelength 區域。

上的 Amazon Elastic Container Service AWS Outposts

AWS Outposts 在內部部署設施中啟用原生 AWS 服務、基礎設施和操作模型。在 AWS Outposts 環境中，您可以使用您在 中使用的相同 AWS APIs、工具和基礎設施 AWS 雲端。

Amazon ECS on AWS Outposts 非常適合需要在靠近內部部署資料和應用程式的情況下執行的低延遲工作負載。

如需詳細資訊 AWS Outposts，請參閱 AWS Outposts 使用者指南。

考量事項

以下是ECS在 上使用 Amazon 的考量 AWS Outposts事項：

- Amazon Elastic Container Registry AWS Identity and Access Management和 Network Load Balancer 會在 AWS 區域中執行，而不是開啟 AWS Outposts。這將提高這些服務和容器之間的延遲。
- AWS Fargate 無法在 上使用 AWS Outposts。

以下是 的網路連線考量 AWS Outposts：

- 如果 AWS Outposts 與其 AWS 區域之間的網路連線中斷，您的叢集將繼續執行。不過，在連線恢復之前，您將無法建立新叢集或對現有叢集採取新動作。在執行個體失敗的情況下，執行個體將不會被自動替換。 CloudWatch Logs 代理程式將無法更新日誌和事件資料。
- 我們建議您在 AWS Outposts 及其 AWS 區域之間提供可靠、高可用性和低延遲的連線。

必要條件

以下是ECS在 上使用 Amazon 的先決條件 AWS Outposts：

- 內部部署資料中心必須已安裝和設定 Outpost。
- Outpost 與其 AWS 區域之間必須有可靠的網路連線。

上的叢集建立概觀 AWS Outposts

以下是組態的概觀：

1. 建立具有 權限的角色和政策 AWS Outposts。
2. 建立具有 權限的IAM執行個體設定檔 AWS Outposts。
3. 建立 VPC，或使用與 位於相同區域中的現有 AWS Outposts。
4. 建立子網路或使用與 相關聯的現有子網路 AWS Outposts。

這是容器執行個體執行所在的子網路。

5. 為叢集中的容器執行個體建立安全群組。
6. 建立 Amazon ECS叢集。
7. 定義 Amazon ECS容器代理程式環境變數，以在叢集中啟動執行個體。
8. 執行容器。

如需如何整合 Amazon ECS與 的詳細資訊 AWS Outposts，請參閱將 Amazon 延伸ECS到兩個 AWS Outposts 機架。

下列範例會在 上建立 Amazon ECS叢集 AWS Outposts。

1. 建立具有 權限的角色和政策 AWS Outposts。

   `role-policy.json` 檔案是政策文件，其中包含資源的效果和動作。如需檔案格式的相關資訊，請參閱 IAM API 參考PutRolePolicy中的

```
aws iam create-role --role-name ecsRole \
    --assume-role-policy-document file://ecs-policy.json
aws iam put-role-policy --role-name ecsRole --policy-name ecsRolePolicy \
    --policy-document file://role-policy.json
```

2. 建立具有 權限的IAM執行個體設定檔 AWS Outposts。

```
aws iam create-instance-profile --instance-profile-name outpost
aws iam add-role-to-instance-profile --instance-profile-name outpost \
    --role-name ecsRole
```

3. 建立 VPC。

```
aws ec2 create-vpc --cidr-block 10.0.0.0/16
```

4. 建立與 相關聯的子網路 AWS Outposts。

```
aws ec2 create-subnet \
    --cidr-block 10.0.3.0/24 \
    --vpc-id vpc-xxxxxxxx \
    --outpost-arn arn:aws:outposts:us-west-2:123456789012:outpost/op-
xxxxxxxxxxxxxxxx \
    --availability-zone-id usw2-az1
```

5. 為容器執行個體建立安全群組，為 指定適當的CIDR範圍 AWS Outposts。（此步驟與 不同 AWS Outposts。）

```
aws ec2 create-security-group --group-name MyOutpostSG
aws ec2 authorize-security-group-ingress --group-name MyOutpostSG --protocol tcp \
    --port 22 --cidr 10.0.3.0/24
aws ec2 authorize-security-group-ingress --group-name MyOutpostSG --protocol tcp \
    --port 80 --cidr 10.0.3.0/24
```

6. 建立叢集。

7. 定義 Amazon ECS容器代理程式環境變數，以在上一個步驟中建立的叢集中啟動執行個體，並定義您要新增的任何標籤，以協助識別叢集（例如，Outpost指出叢集適用於 Outpost)。

```
#! /bin/bash
cat << 'EOF' >> /etc/ecs/ecs.config
ECS_CLUSTER=MyCluster
ECS_IMAGE_PULL_BEHAVIOR=prefer-cached
ECS_CONTAINER_INSTANCE_TAGS={"environment": "Outpost"}
EOF
```

> **Note**
>
> 為了避免從 ECR 區域中的 Amazon 提取容器映像所造成的延遲，請使用映像快取。若要這樣做，每次執行任務時，請將 Amazon ECS代理程式設定為預設，透過將ECS_IMAGE_PULL_BEHAVIOR設定為 ，在執行個體本身上使用快取映像prefer-cached。

8. 建立容器執行個體，指定此執行個體應執行的 VPC AWS Outposts 和子網路，以及 上可用的執行個體類型 AWS Outposts。（此步驟與 不同 AWS Outposts。）

   userdata.txt 檔案中包含使用者資料，執行個體可以使用這些使用者資料來執行常見的自動化組態任務，甚至在執行個體啟動之後執行指令碼。如需 API呼叫檔案的相關資訊，請參閱《Amazon EC2 使用者指南》中的在啟動時在 Linux 執行個體上執行命令。

```
aws ec2 run-instances --count 1 --image-id ami-xxxxxxxx --instance-type c5.large \
    --key-name aws-outpost-key --subnet-id subnet-xxxxxxxxxxxxxxxxx \
    --iam-instance-profile Name outpost --security-group-id sg-xxxxxx \
    --associate-public-ip-address --user-data file://userdata.txt
```

> **Note**
>
> 將其他執行個體新增至叢集時，也會使用此指令。在叢集中部署的任何容器都會放置在該特定 AWS Outposts上。

最佳化 Amazon ECS容量和可用性

應用程式可用性對於提供無錯誤體驗和將應用程式延遲降至最低至關重要。可用性取決於是否具有足夠的可用資源，這些資源具有足夠的容量來滿足需求。 AWS 提供數種機制來管理可用性。對於託管在 Amazon 上的應用程式ECS，包括自動擴展和可用區域 (AZs)。Autoscaling 會根據您定義的指標來管理任務或執行個體的數量，而可用區域可讓您在隔離但地理位置接近的位置託管應用程式。

與任務大小一樣，容量和可用性存在您必須考慮的特定權衡。在理想情況下，容量將與需求完全一致。永遠只有足夠容量來為請求和程序任務提供服務，以滿足服務水準目標 (SLOs)，包括低延遲和錯誤率。容量永遠不會太高，導致成本過高；也不會太低，導致高延遲和錯誤率。

Autoscaling 是隱含的程序。首先，即時指標必須交付至 CloudWatch。然後，它們需要彙總以供分析，這可能需要幾分鐘的時間，這取決於 metric. CloudWatch comp 的精細程度，比較指標與警示閾值，以識別資源不足或過多。為了防止不穩定，請將警示設定為要求在警示關閉之前超過設定的閾值幾分鐘。佈建新任務和終止不再需要的任務也需要一些時間。

由於所述系統中存在這些潛在的延遲，因此請務必透過過度佈建來維護一些總空間。這樣做有助於因應短期的需求暴增。這也有助於您的應用程式在未達到飽和的情況下，為其他請求提供服務。作為最佳實務，您可以將擴展目標設定為 60-80% 的使用率。這有助於您的應用程式更妥善地處理大量額外需求，同時額外容量仍在佈建中。

我們建議您過度佈建的另一個原因是，讓您可以快速回應可用區域故障。我們建議從多個可用區域提供生產工作負載。這是因為如果發生可用區域故障，您在剩餘可用區域中執行的任務仍然可以滿足需求。如果您的應用程式在兩個可用區域中執行，您需要將正常任務計數加倍。如此一來，您就可以在任何潛在的故障期間提供立即容量。如果您的應用程式在三個可用區域中執行，我們建議您執行一般任務計數的 1.5 倍。也就是說，對一般服務所需的每兩個執行三個任務。

最大化擴展速度

Autoscaling 是一種被動程序，需要一些時間才能生效。不過，有一些方法可協助將擴展所需的時間降至最低。

將影像大小降至最低。較大的映像需要較長的時間才能從映像儲存庫下載並解壓縮。因此，保持較小的影像大小可減少容器啟動所需的時間量。若要減少影像大小，您可以遵循下列特定建議：

- 如果您可以建置靜態二進位檔或使用 Golang，請建置映像FROM暫存，並在產生的映像中只包含您的二進位應用程式。
- 使用上游特派廠商的最小化基礎映像，例如 Amazon Linux 或 Ubuntu。
- 請勿在最終映像中包含任何建置成品。使用多階段組建有助於解決此問題。
- 盡可能精簡RUN階段。每個RUN階段都會建立新的映像層，導致額外的往返行程來下載層。由 加入多個命令的單一RUN階段&&，其層數少於具有多個RUN階段的層數。
- 如果您想要在最終映像中包含資料，例如 ML 推論資料，請僅包含啟動和開始服務流量所需的資料。如果您根據需求從 Amazon S3 或其他儲存體擷取資料，而不影響服務，請改為將您的資料存放在這些位置。

保持影像關閉。網路延遲越高，下載映像所需的時間就越長。在工作負載所在的相同區域中，將映像託管在儲存庫中。Amazon ECR 是高效能映像儲存庫，可在 Amazon ECS提供的每個區域中使用。避免周遊網際網路或下載容器映像VPN的連結。在相同區域中託管映像可改善整體可靠性。它可降低不同區域中網路連線問題和可用性問題的風險。或者，您也可以實作 Amazon ECR跨區域複寫來協助執行此操作。

降低負載平衡器運作狀態檢查閾值。負載平衡器會在傳送流量至應用程式之前執行運作狀態檢查。目標群組的預設運作狀態檢查組態可能需要 90 秒或更久的時間。在此期間，負載平衡器會檢查運作狀態並接收請求。降低運作狀態檢查間隔和閾值計數可讓應用程式更快接受流量，並減少其他任務的負載。

考慮冷啟動效能。有些應用程式使用執行期，例如 Java 執行 Just-In-Time (JIT) 編譯。編譯程序啟動時至少會降低應用程式效能。解決方法是使用不會造成冷啟動效能懲罰的語言，將工作負載的延遲關鍵部分重寫為。

使用步驟擴展，而非目標追蹤擴展政策。任務有數個 Application Auto Scaling 選項。目標追蹤是最容易使用的模式。有了它，您只需要為指標設定目標值，例如CPU平均使用率。然後，自動縮放器會自動管理獲得該值所需的任務數量。使用步驟擴展，您可以更快對需求變化做出反應，因為您定義了擴展指標的特定閾值，以及超越閾值時要新增或刪除的任務數量。而且，更重要的是，您可以透過最大限度減少閾值警報違反的時間來對需求變化做出非常快速的反應。如需詳細資訊，請參閱 Service Auto Scaling。

如果您使用 Amazon EC2執行個體來提供叢集容量，請考慮下列建議：

使用較大的 Amazon EC2執行個體和更快的 Amazon EBS磁碟區。您可以使用較大的 Amazon EC2執行個體和更快的 Amazon EBS磁碟區來改善映像下載和準備速度。在指定的執行個體系列

中，網路和 Amazon EBS最大輸送量會隨著執行個體大小的增加而增加（例如，從 m5.xlarge到 m5.2xlarge)。此外，您也可以自訂 Amazon EBS磁碟區，以提高其輸送量和 IOPS。例如，如果您使用gp2磁碟區，請使用較大的磁碟區來提供更多基準輸送量。如果您使用gp3磁碟區，請在建立磁碟區IOPS時指定輸送量和 。

針對在 Amazon EC2執行個體上執行的任務使用橋接網路模式。在 Amazon 上使用bridge網路模式的任務EC2啟動速度比使用awsvpc網路模式的任務快。使用awsvpc網路模式時，Amazon 會在啟動任務之前，將彈性網路介面 (ENI) ECS連接至執行個體。這會導致額外的延遲。不過，使用橋接聯網有幾個權衡。這些任務無法取得自己的安全群組，而且負載平衡有一些影響。如需詳細資訊，請參閱 Elastic Load Balancing 使用者指南中的負載平衡器目標群組。

處理需求衝擊

某些應用程式會突然發生大量的需求衝擊。發生這種情況的原因有很多：新聞事件、大型銷售、媒體事件或一些其他會傳播並導致流量在非常短的時間內快速大幅增加的事件。如果未計劃，這可能會導致需求快速超過可用資源。

處理需求衝擊的最佳方法是預測並相應地進行規劃。由於自動擴展可能需要一些時間，我們建議您在需求衝擊開始之前擴展應用程式。為了獲得最佳結果，我們建議您制定商業計劃，該計畫涉及使用共用行事曆的團隊之間的緊密合作。規劃事件的團隊應事先與負責應用程式的團隊緊密合作。這可讓該團隊有足夠的時間制定明確的排程計劃。他們可以排定容量在事件之前向外擴展，並在事件之後向內擴展。如需詳細資訊，請參閱《Application Auto Scaling 使用者指南》中的排程擴展。

如果您有企業支援計劃，請務必也與您的技術客戶經理 () 合作TAM。您的 TAM可以驗證您的服務配額，並確保在事件開始之前產生任何必要的配額。如此一來，您就不會意外達到任何服務配額。它們也可以預先暖機服務，例如負載平衡器，以確保您的事件順利進行。

處理未排程的需求衝擊是更困難的問題。如果幅度夠大，未排程的震動可能會快速導致容量超出需求。它也可以超過自動擴展反應的能力。準備意外衝擊的最佳方式是過度佈建資源。您必須有足夠的資源，以隨時處理最大預期流量需求。

在預期非排定需求衝擊時維持最大容量可能成本高昂。若要降低成本影響，請尋找預測大型需求衝擊的領導指標或事件即將到來。如果指標或事件可靠地提供重大的預先通知，請在事件發生或指標超過您設定的特定閾值時，立即開始橫向擴展程序。

如果您的應用程式容易突然發生未排程的需求衝擊，請考慮為應用程式新增高效能模式，以犧牲非關鍵功能，但保留客戶的重要功能。例如，假設您的應用程式可以從產生昂貴的自訂回應切換為提供靜態回應頁面。在此案例中，您可以大幅提高輸送量，完全不需要擴展應用程式。

您可以考慮分開單體服務，以更好地處理需求衝擊。如果您的應用程式是執行成本高且擴展速度慢的單體服務，您可能可以擷取或重寫效能關鍵片段，並以個別服務執行它們。然後，這些新服務可以獨立於較不關鍵的元件進行擴展。具有將效能關鍵功能與應用程式其他部分分開擴展的彈性，可以減少增加容量和協助節省成本所需的時間。

Amazon ECS 網路最佳實務

現代應用程式通常由多個彼此通訊的分散式元件所建置。例如，行動或 Web 應用程式可能會與API端點通訊，而 API可能由透過網際網路通訊的多個微服務提供支援。

如需將應用程式連線至網際網路的最佳實務的相關資訊，請參閱 將 Amazon ECS 應用程式連接至網際網路。

如需ECS從網際網路接收 Amazon 傳入連線的最佳實務資訊，請參閱 ECS 從網際網路接收 Amazon 傳入連線的最佳實務。

如需將 Amazon ECS連線至其他服務的最佳實務的相關資訊 AWS ，請參閱 從 內部將 Amazon ECS 連線至 AWS 服務的最佳實務 VPC。

如需在 中連接服務的最佳實務的相關資訊VPC，請參閱 在 中連接 Amazon ECS服務的最佳實務 VPC。

如需跨 AWS 帳戶 和 聯網服務最佳實務的相關資訊VPCs，請參閱 跨 AWS 帳戶 和 聯網 Amazon ECS 服務的最佳實務 VPCs。

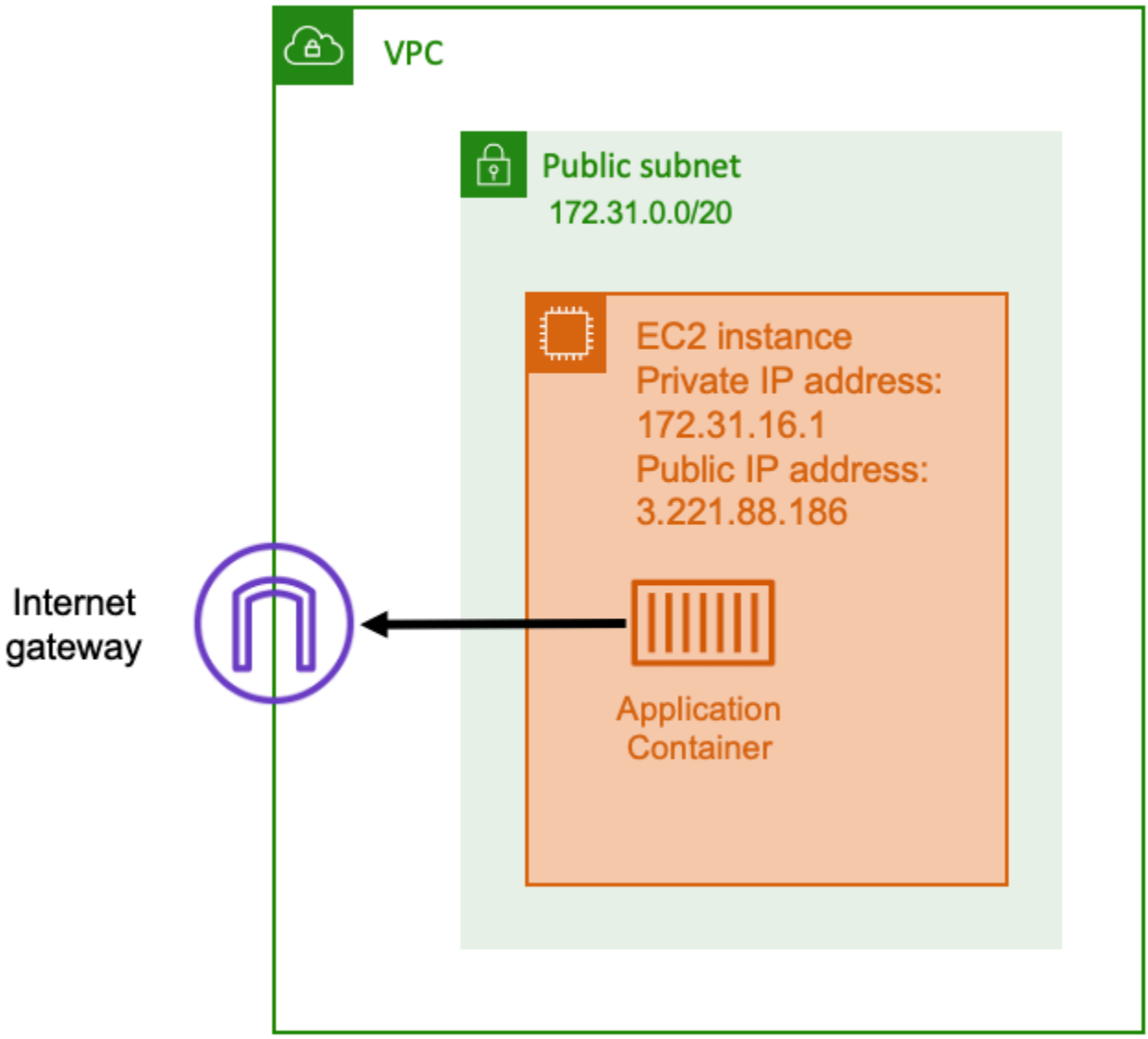
如需針對網路問題進行故障診斷的服務最佳實務的相關資訊，請參閱 AWS 適用於 Amazon ECS 網路故障診斷的 服務。

將 Amazon ECS 應用程式連接至網際網路

大多數容器化應用程式至少有一些元件需要對外存取網際網路。例如，行動應用程式的後端需要傳出推送通知的存取權。

Amazon Virtual Private Cloud 有兩種主要方法可促進 VPC和網際網路之間的通訊。

公有子網路和網際網路閘道



當您使用具有網際網路閘道路由的公有子網路時，您的容器化應用程式可以在公有子網路 內的主機VPC上執行。執行容器的主機會獲指派公有 IP 地址。此公有 IP 地址可從網際網路路由。如需詳細資訊，請參閱《Amazon VPC使用者指南》中的網際網路閘道。

此網路架構可促進執行應用程式之主機與網際網路上其他主機之間的直接通訊。通訊是雙向的。這表示您不僅可以建立與網際網路上任何其他主機的傳出連線，而且網際網路上的其他主機也可能嘗試連接到您的主機。因此，您應該密切注意您的安全群組和防火牆規則。這可確保網際網路上的其他主機無法開啟您不想開啟的任何連線。

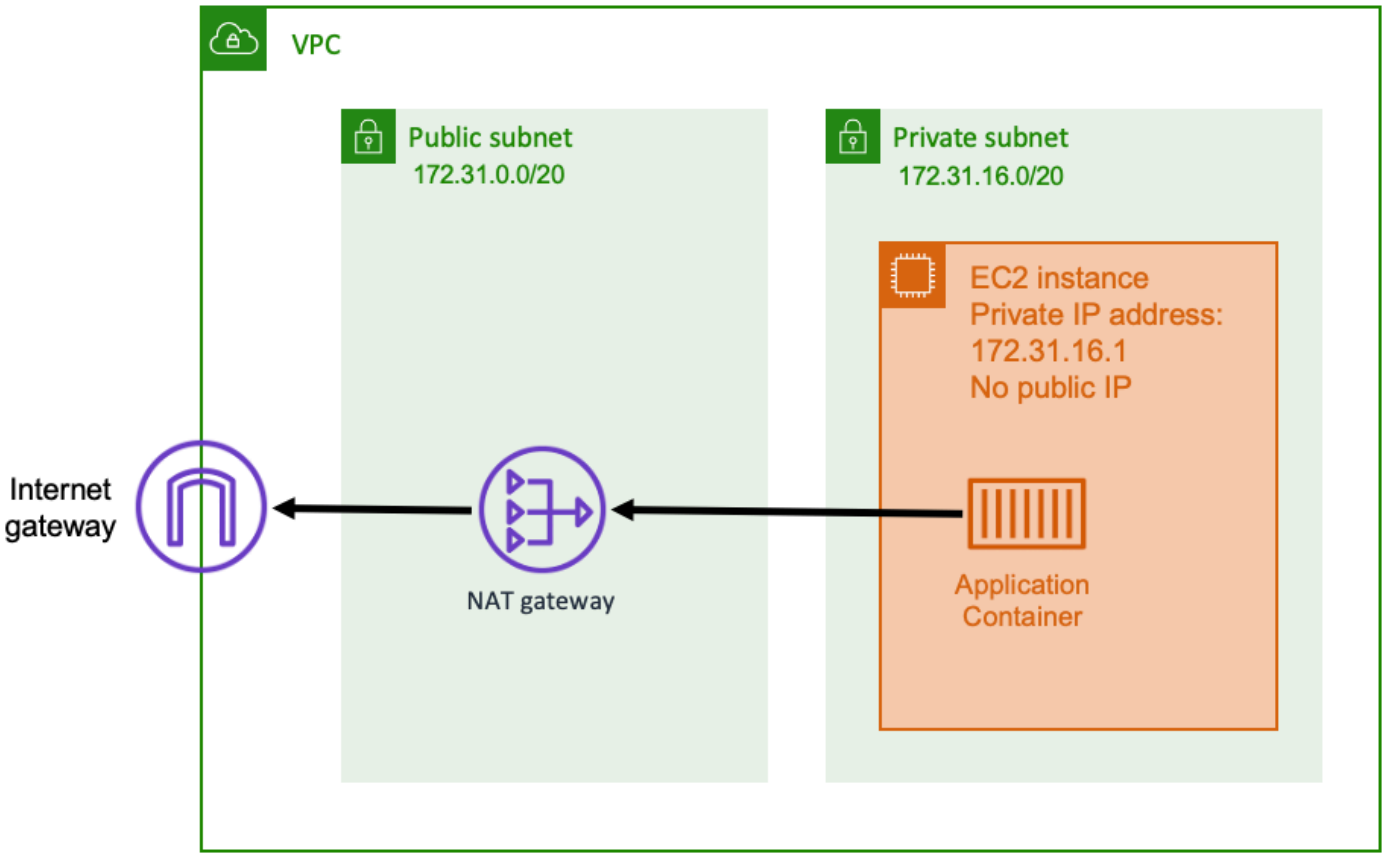
例如，如果您的應用程式在 Amazon 上執行EC2，請確定連接埠 22 未開啟以供SSH存取。否則，您的執行個體可能會收到來自網際網路上惡意機器人的持續SSH連線嘗試。這些機器人會透過公有 IP 地址進行編目。找到開放SSH連接埠後，他們會嘗試強制密碼來嘗試存取您的執行個體。因此，許多組織會限制公有子網路的使用，如果不是全部，則偏好在私有子網路內擁有大部分的資源。

使用公有子網路進行聯網適用於需要大量頻寬或最低延遲的公有應用程式。適用的使用案例包括影片串流和遊戲服務。

當您ECS在 Amazon 上使用 Amazon EC2時和使用時，都支援此聯網方法 AWS Fargate。

- Amazon EC2 — 您可以在公有子網路上啟動EC2執行個體。Amazon ECS使用這些EC2執行個體做為叢集容量，而執行個體上執行的任何容器都可以使用主機的基礎公有 IP 地址進行傳出聯網。這同時適用於 `host`和 `bridge` 網路模式。不過，`awsvpc`網路模式不會提供ENIs具有公有 IP 地址的任務。因此，他們無法直接使用網際網路閘道。
- Fargate — 當您建立 Amazon ECS服務時，請為服務的聯網組態指定公有子網路，並使用指派公有 IP 地址選項。每個 Fargate 任務都在公有子網路中聯網，並擁有自己的公有 IP 地址，可直接與網際網路通訊。

私有子網路和NAT閘道



當您使用私有子網路和NAT閘道時，您可以在私有子網路中的主機上執行容器化應用程式。因此，此主機具有可在 內路由的私有 IP 地址VPC，但無法從網際網路路由。這表示 內的其他主機VPC可以使用其私有 IP 地址連線到主機，但網際網路上的其他主機無法對主機進行任何傳入通訊。

透過私有子網路，您可以使用網路地址轉譯 (NAT) 閘道，允許私有子網路內的主機連線至網際網路。網際網路上的主機會收到傳入連線，似乎來自公有子網路內NAT閘道的公有 IP 地址。NAT 閘道負責充當網際網路與私有子網路之間的橋樑。基於安全考量，通常偏好此組態，因為這表示您的 VPC 不受網際網路上的攻擊者直接存取。如需詳細資訊，請參閱《Amazon VPC使用者指南》中的NAT閘道。

此私有聯網方法適用於您希望保護容器免於直接外部存取的情況。適用的案例包括付款處理系統或儲存使用者資料與密碼的容器。您需要支付在帳戶中建立和使用NAT閘道的費用。NAT閘道每小時用量和資料處理費率也適用。基於備援目的，您應該在每個可用區域中有一個NAT閘道。如此一來，單一可用區域可用性的損失，就不會影響對外連線。因此，如果您的工作負載很小，使用私有子網路和NAT閘道可能更具成本效益。

在 Amazon ECS上使用 Amazon EC2時和使用時，都支援此聯網方法 AWS Fargate。

- Amazon EC2 — 您可以在私有子網路上啟動EC2執行個體。在這些EC2主機上執行的容器會使用基礎主機聯網，而傳出請求會通過NAT閘道。
- Fargate — 當您建立 Amazon ECS服務時，請為服務的網路組態指定私有子網路，並且不使用指派公有 IP 地址選項。每個 Fargate 任務都託管在私有子網路中。其傳出流量會透過您與該私有子網路相關聯的任何NAT閘道路由。

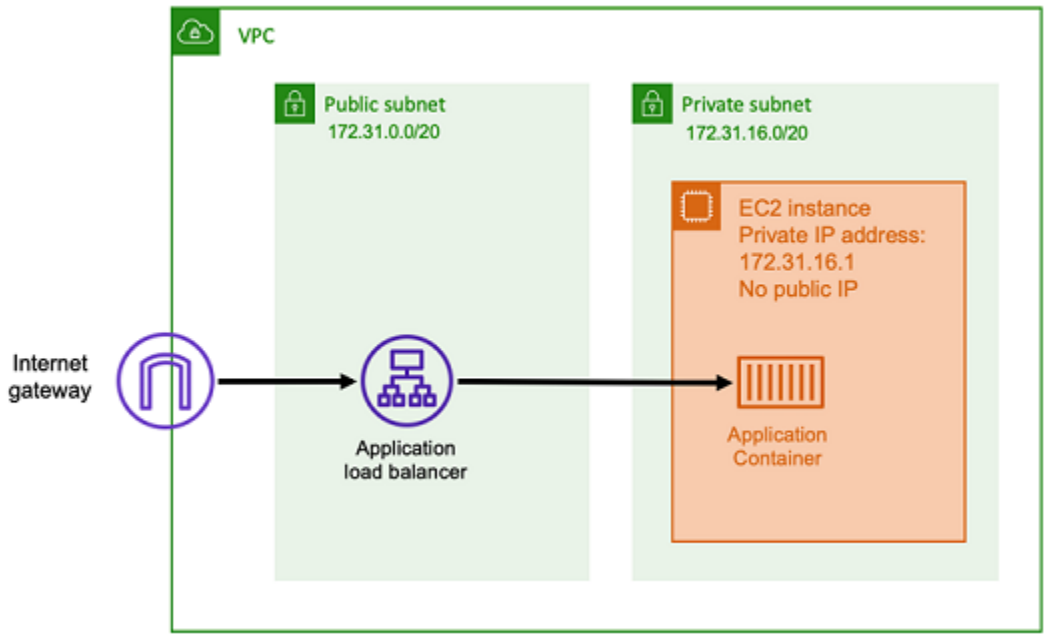
ECS 從網際網路接收 Amazon 傳入連線的最佳實務

如果您執行公有服務，則必須接受來自網際網路的傳入流量。例如，您的公有網站必須接受來自瀏覽器的傳入HTTP請求。在這種情況下，網際網路上的其他主機也必須啟動與應用程式主機的傳入連線。

此問題的一個方法是在具有公有 IP 地址的公有子網路主機上啟動您的容器。不過，我們不建議大規模應用程式使用此功能。對於這些，更好的方法是在網際網路和應用程式之間具有可擴展的輸入層。對於此方法，您可以使用本節中列出的任何 AWS 服務做為輸入。

Application Load Balancer

Application Load Balancer 函數位於應用程式層。這是開放系統互連 (OSI) 模型的第七層。這可讓 Application Load Balancer 適用於公有HTTP服務。如果您有網站或 HTTP REST API，則 Application Load Balancer 是適合此工作負載的負載平衡器。如需詳細資訊，請參閱《Application Load Balancer 使用者指南》中的什麼是 Application Load Balancer？。



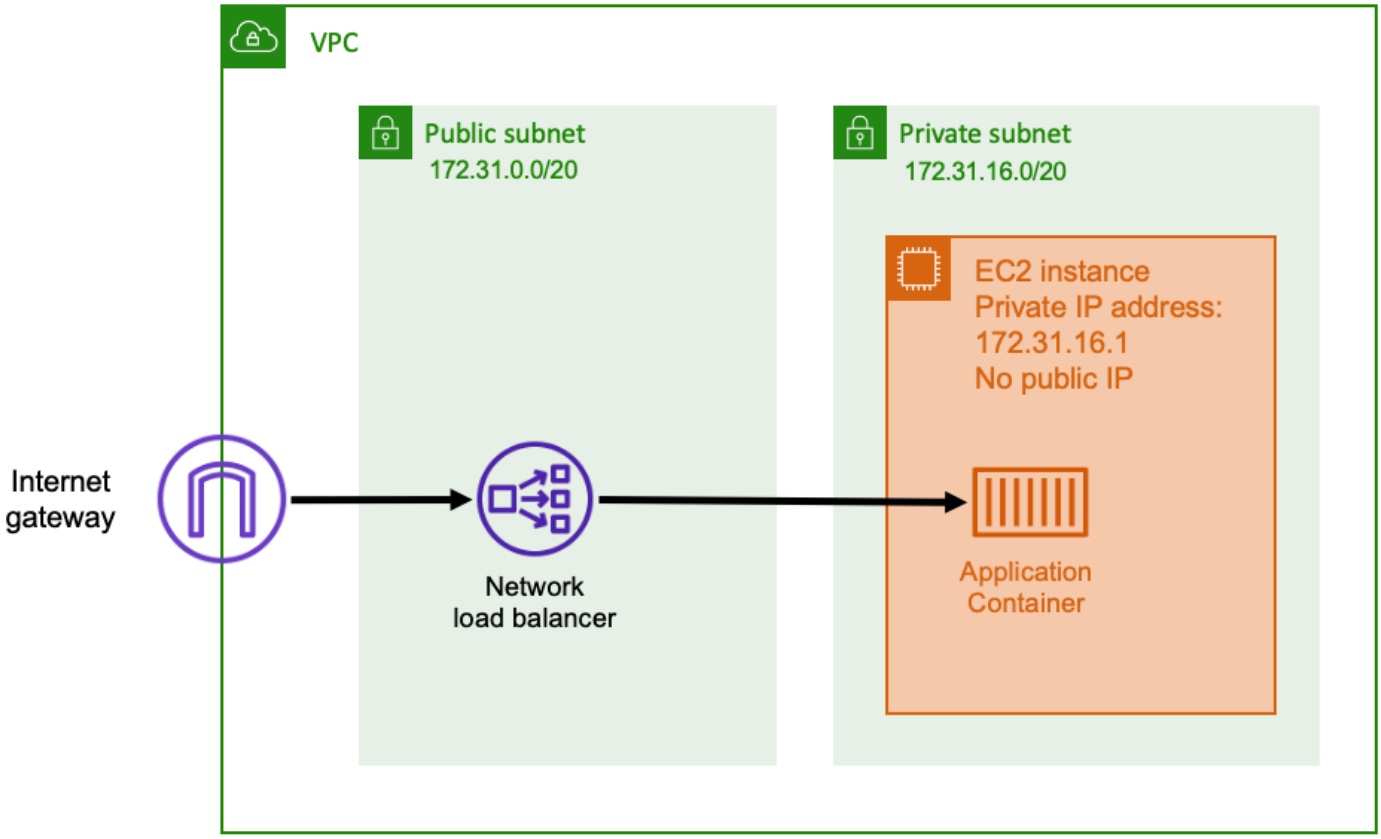
使用此架構，您可以在公有子網路中建立 Application Load Balancer，使其具有公有 IP 地址，並可從網際網路接收傳入連線。當 Application Load Balancer 收到傳入連線，或更具體地收到HTTP請求時，它會使用其私有 IP 地址開啟與應用程式的連線。然後，它會透過內部連線轉送請求。

Application Load Balancer 具有下列優點。

- SSL/TLS 終止 — Application Load Balancer 可以維持與用戶端HTTPS通訊的安全通訊和憑證。它可以選擇性地終止負載平衡器層級的SSL連線，讓您不必在自己的應用程式中處理憑證。
- 進階路由 — Application Load Balancer 可以具有多個DNS主機名稱。它還具有進階路由功能，可根據主機名稱或HTTP請求路徑等指標，將傳入的請求傳送至不同的目的地。這表示您可以使用單一 Application Load Balancer 做為許多不同內部服務的輸入，或甚至是 REST 不同路徑上的微服務 API。
- gRPC 支援和 Websocket：Application Load Balancer 不僅可以處理 HTTP。它也可以透過 HTTP/2 支援來載入平衡 gRPC 和 Websocket 型服務。
- 安全：Application Load Balancer 可協助保護您的應用程式免於惡意流量。它包含功能，例如HTTP取消同步緩解，並與 AWS Web Application Firewall (AWS WAF) 整合。 AWS WAF 可以進一步篩選可能包含攻擊模式的惡意流量，例如SQL注入或跨網站指令碼。

Network Load Balancer

Network Load Balancer 會在開放系統互連 (OSI) 模型的第四層運作。它適用於需要加密的非HTTP通訊協定或案例 end-to-end，但沒有 Application Load Balancer 的相同 HTTP特定功能。因此，Network Load Balancer 最適合不使用 的應用程式HTTP。如需詳細資訊，請參閱 Network Load Balancer 使用者指南中的什麼是 Network Load Balancer？。



使用 Network Load Balancer 做為輸入時，其運作方式與 Application Load Balancer 類似。這是因為它是在公有子網路中建立的，並且具有可在網際網路上存取的公有 IP 地址。然後，Network Load Balancer 會開啟與執行您容器之主機私有 IP 地址的連線，並將封包從公有端傳送至私有端。

Network Load Balancer 功能

由於 Network Load Balancer 在網路堆疊的較低層級運作，因此它沒有 Application Load Balancer 執行的相同功能集。不過，它確實具有下列重要功能。

- End-to-end 加密 — 由於 Network Load Balancer 會在OSI模型的第四層運作，因此不會讀取封包的內容。這使得它適用於需要 end-to-end加密的負載平衡通訊。
- TLS encryption — 除了加密之外 end-to-end，Network Load Balancer 也可以終止TLS連線。如此一來，您的後端應用程式就不必實作自己的 TLS。
- UDP 支援 — 由於 Network Load Balancer 會在OSI模型的第四層運作，因此適合 以外的非HTTP工作負載和通訊協定TCP。

關閉連線

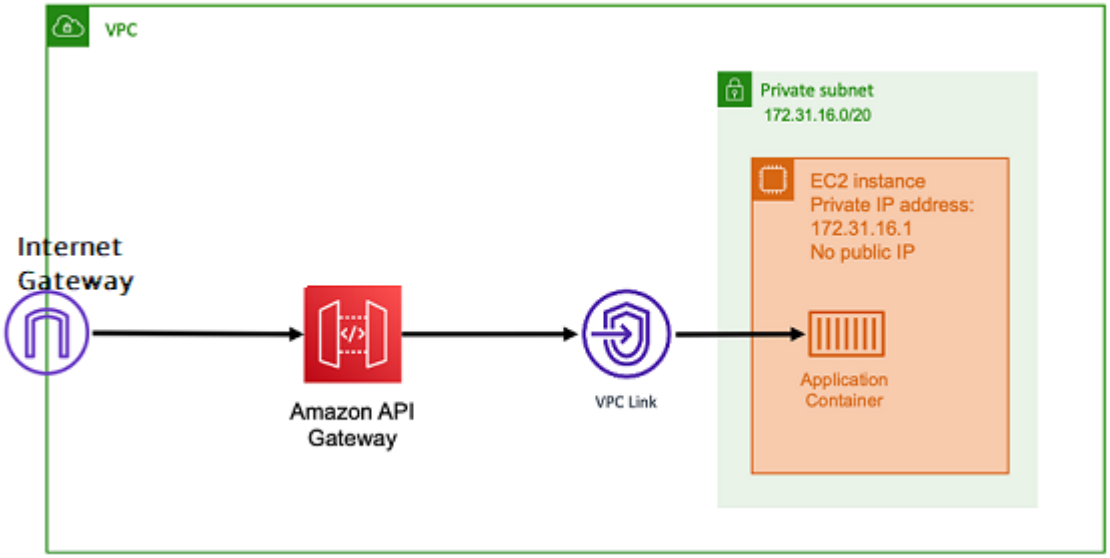
由於 Network Load Balancer 不會在OSI模型的較高層觀察應用程式通訊協定，因此無法傳送關閉訊息給這些通訊協定中的用戶端。與 Application Load Balancer 不同，這些連線需要由應用程式關閉，或者您可以設定 Network Load Balancer 在任務停止或取代時關閉第四層連線。請參閱 Network Load Balancer 文件中 Network Load Balancer 目標群組的連線終止設定。

如果用戶端未處理，讓 Network Load Balancer 關閉第四層的連線可能會導致用戶端顯示不需要的錯誤訊息。有關建議用戶端組態的詳細資訊，請參閱這裡的建置程式庫。

關閉連線的方法因應用程式而異，但其中一種方法是確保 Network Load Balancer 目標取消註冊延遲比用戶端連線逾時更長。用戶端會先逾時，並透過 Network Load Balancer 正常重新連線至下一個任務，而舊任務會慢慢耗盡其所有用戶端。如需 Network Load Balancer 目標取消註冊延遲的詳細資訊，請參閱 Network Load Balancer 文件。

Amazon API Gateway HTTP API

Amazon API Gateway 適用於請求磁碟區或低請求磁碟區突然爆增HTTP的應用程式。如需詳細資訊，請參閱API《閘道開發人員指南》中的什麼是 Amazon Gateway？。 API



Application Load Balancer 和 Network Load Balancer 的定價模型包含每小時價格，讓負載平衡器隨時可供接受傳入連線。相反地，APIGateway 會分別針對每個請求收取費用。這會影響如果沒有請求進來，就不會產生任何費用。在高流量負載下，Application Load Balancer 或 Network Load Balancer 可以比 API Gateway 更便宜的每次請求價格處理更多請求。不過，如果您的整體請求數量較少，或流量期間很低，則使用 API Gateway 的累積價格應該比支付每小時費用更符合成本效益，以維持未充分利用的負載平衡器。API Gateway 也可以快取API回應，這可能會導致較低的後端請求率。

API 閘道函數使用VPC連結，允許 AWS 受管服務VPC使用其私有 IP 地址，連接到 私有子網路內的主機。它可以透過查看由 Amazon ECS Service Discovery 管理 AWS Cloud Map 的服務探索記錄來偵測這些私有 IP 地址。

API Gateway 支援下列功能。

- API Gateway 操作類似於負載平衡器，但具有API管理特有的其他功能
- API Gateway 提供用戶端授權、用量層和請求/回應修改的其他功能。如需詳細資訊，請參閱 Amazon API Gateway 功能。
- API Gateway 可以支援邊緣、區域和私有API閘道端點。邊緣端點可透過受管 CloudFront 分佈取得。區域和私有端點都是區域本機端點。
- SSL/TLS 終止
- 將不同的HTTP路徑路由到不同的後端微服務

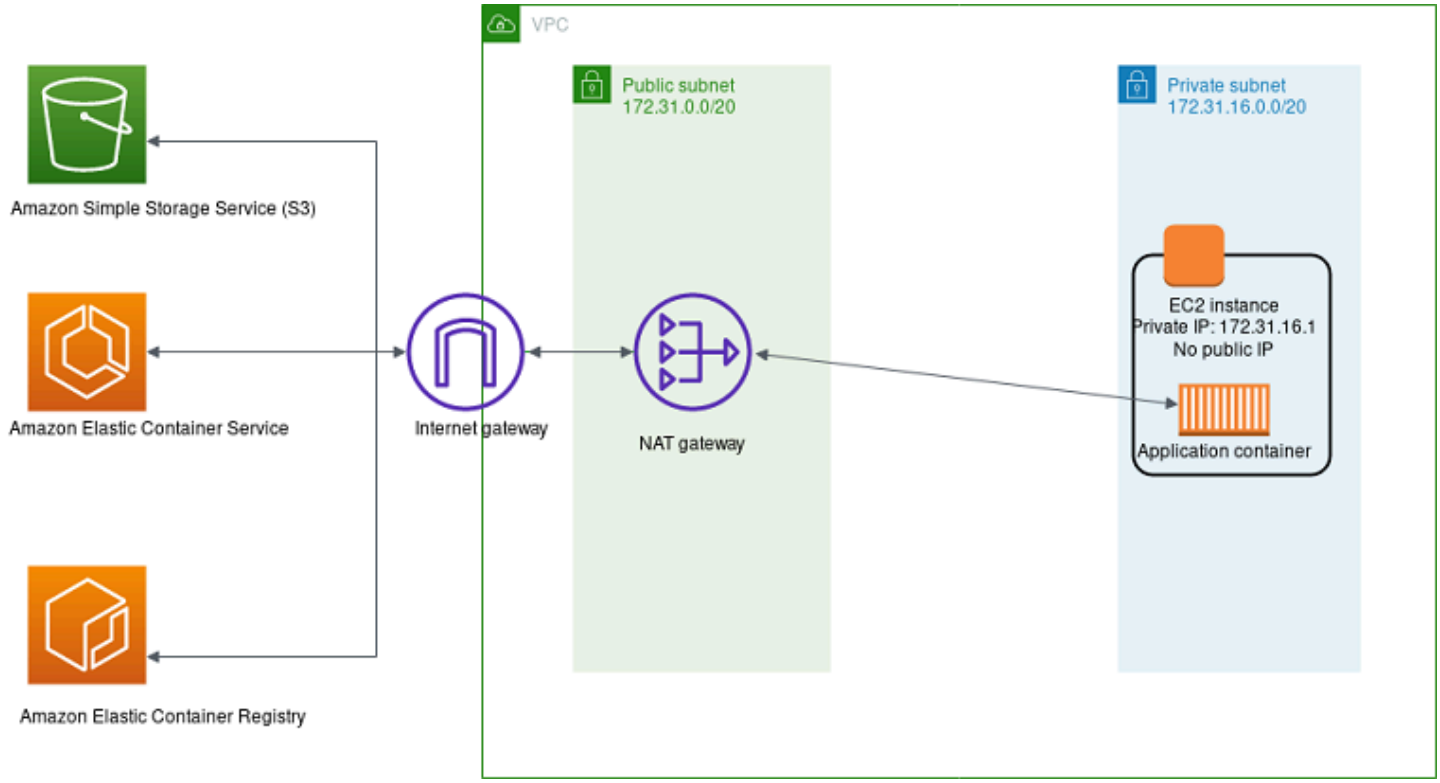
除了上述功能之外，APIGateway 也支援使用自訂 Lambda 授權方，您可以使用這些授權方來保護您的 API 不受未經授權的使用。如需詳細資訊，請參閱欄位備註：APIs使用 Amazon ECS和 Amazon API Gateway 的無伺服器容器型。

從 內部將 Amazon ECS 連線至 AWS 服務的最佳實務 VPC

若要ECS讓 Amazon 正常運作，在每個主機上執行的 Amazon ECS容器代理程式必須與 Amazon ECS 控制平面通訊。如果您要將容器映像儲存在 Amazon 中ECR，Amazon EC2 主機必須與 Amazon ECR 服務端點和儲存映像層的 Amazon S3 通訊。如果您將其他服務 AWS 用於容器化應用程式，例如保留存放在 DynamoDB 中的資料，請再次檢查這些服務是否也具有必要的聯網支援。

NAT 閘道

使用NAT閘道是確保您的 Amazon ECS任務可以存取其他服務的最簡單方式 AWS 。如需此方法的詳細資訊，請參閱 私有子網路和NAT閘道。



以下是使用此方法的缺點：

- 您無法限制NAT閘道可與哪些目的地通訊。您也無法限制後端層可以通訊的目的地，而不會中斷來自的所有傳出通訊VPC。
- NAT 閘道會針對傳遞的每個 GB 資料收取費用。如果您將NAT閘道用於下列任何操作，則會向您收取每 GB 頻寬的費用：
  - 從 Amazon S3 下載大型檔案
  - 對 DynamoDB 執行大量資料庫查詢
  - 從 Amazon 提取映像 ECR

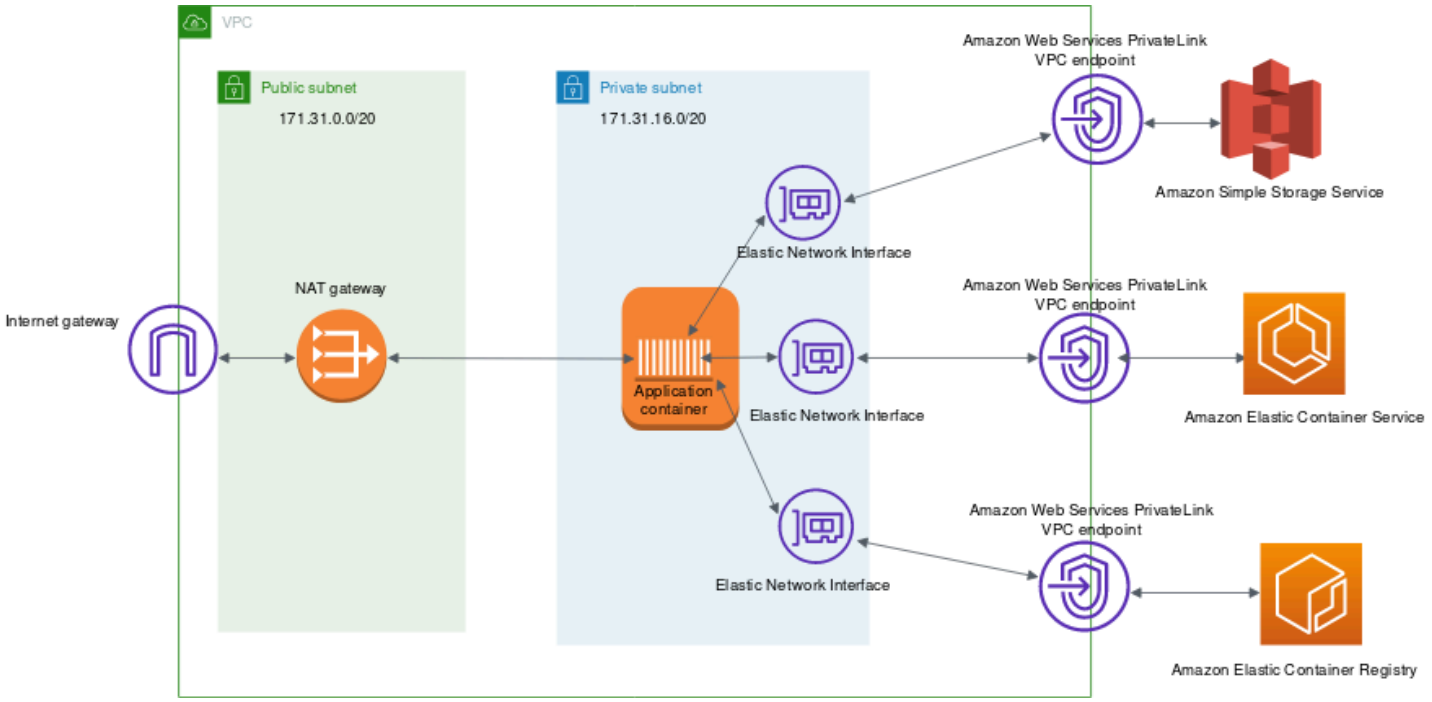
  此外，NAT閘道支援 5 Gbps 的頻寬，並自動擴展至 45 Gbps。如果您透過單一NAT閘道路由，需要極高頻寬連線的應用程式可能會遇到聯網限制。作為解決方法，您可以將工作負載分割到多個子網路，並將自己的NAT閘道提供給每個子網路。

AWS PrivateLink

AWS PrivateLink 提供 VPCs、 AWS 服務與內部部署網路之間的私有連線，而不會將您的流量暴露到公有網際網路。

VPC 端點允許您的 VPC和 支援 AWS 的服務與VPC端點服務之間的私有連線。您的 VPC與其他 服務之間的流量不會離開 Amazon 網路。VPC 端點不需要網際網路閘道、虛擬私有閘道、NAT裝置、VPN 連線或 AWS Direct Connect 連線。您 中的 Amazon EC2執行個體VPC不需要公有 IP 地址，即可與 服務中的資源通訊。

下圖顯示當您使用 VPC 端點而非網際網路閘道時， AWS 服務通訊的運作方式。 AWS PrivateLink 佈建子網路內的彈性網路介面 (ENIs)，而VPC路由規則會用來透過 ENI將任何通訊直接傳送至目的地 AWS 服務，以傳送到服務主機名稱。此流量不再需要使用NAT閘道或網際網路閘道。



以下是與 Amazon ECS服務搭配使用的一些常見VPC端點。

- Amazon S3 的閘道端點
- DynamoDB VPC端點
- Amazon ECSVPC端點
- Amazon ECRVPC端點

許多 AWS 其他服務都支援VPC端點。如果您大量使用任何 AWS 服務，您應該查詢該服務的特定文件，以及如何為該流量建立VPC端點。

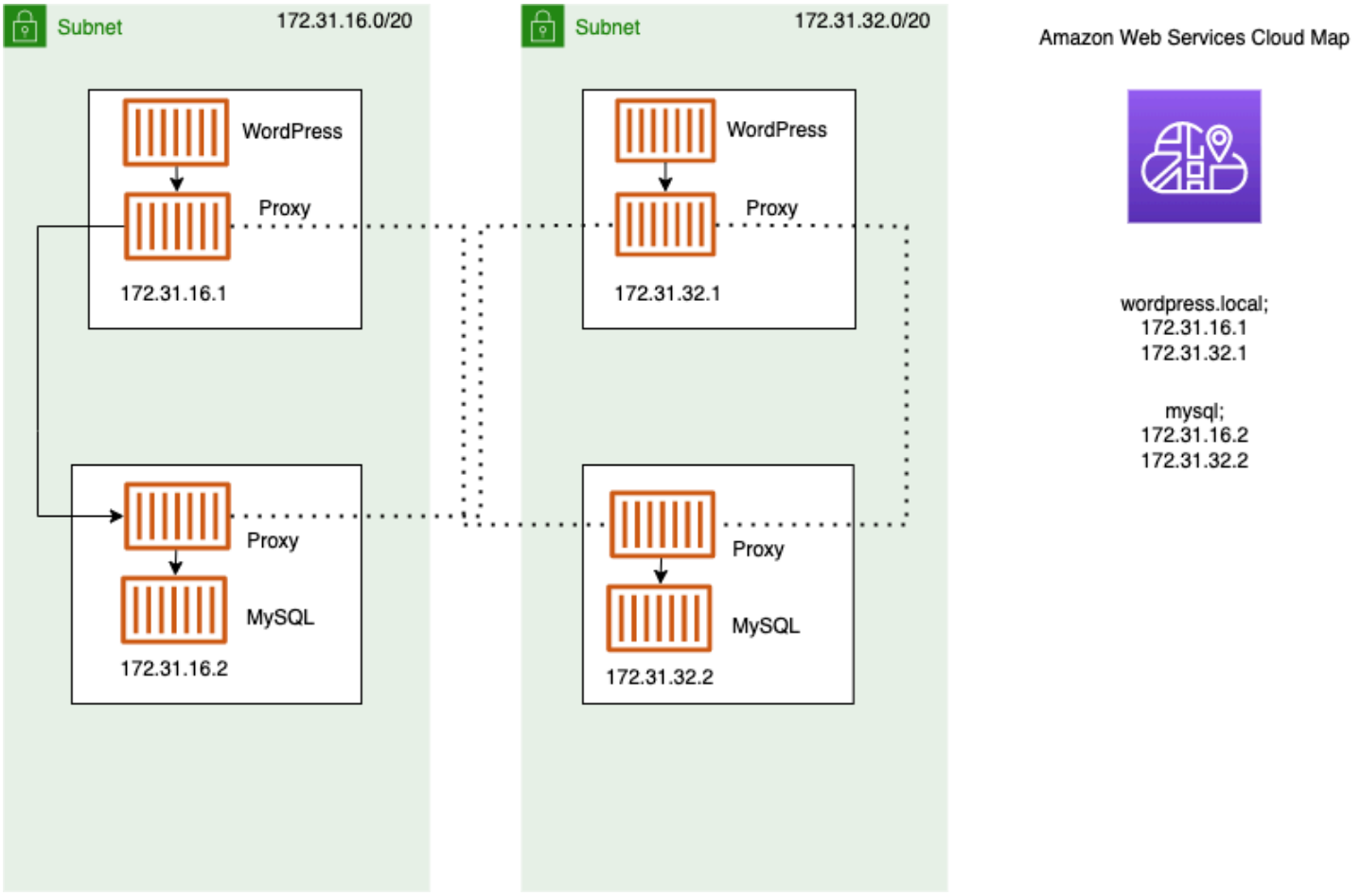
在 中連接 Amazon ECS服務的最佳實務 VPC

在 中使用 Amazon ECS任務VPC，您可以將單體應用程式分割為可在安全環境中獨立部署和擴展的個別部分。此架構稱為服務導向架構 (SOA) 或微型服務。不過，要確保 內外的所有這些部分可以互相通訊VPC，可能很困難。促進溝通的方法有幾種，所有方法都有不同的優點和缺點。

使用 Service Connect

我們建議 Service Connect，其提供 Amazon ECS組態，用於服務探索、連線能力和流量監控。使用 Service Connect，您的應用程式可以使用短名稱和標準連接埠來連接到相同叢集中的服務，其他叢集，包括相同區域中VPCs的 。如需詳細資訊，請參閱 Amazon ECS Service Connect。

當您使用 Service Connect 時，Amazon 會ECS管理服務探索的所有部分：建立可探索的名稱、在任務開始和停止時動態管理每個任務的項目、在設定為探索名稱的每個任務中執行代理程式。您的應用程式可以使用 標準功能來查詢DNS名稱並進行連線。如果您的應用程式已經這樣做，則不需要修改應用程式即可使用 Service Connect。

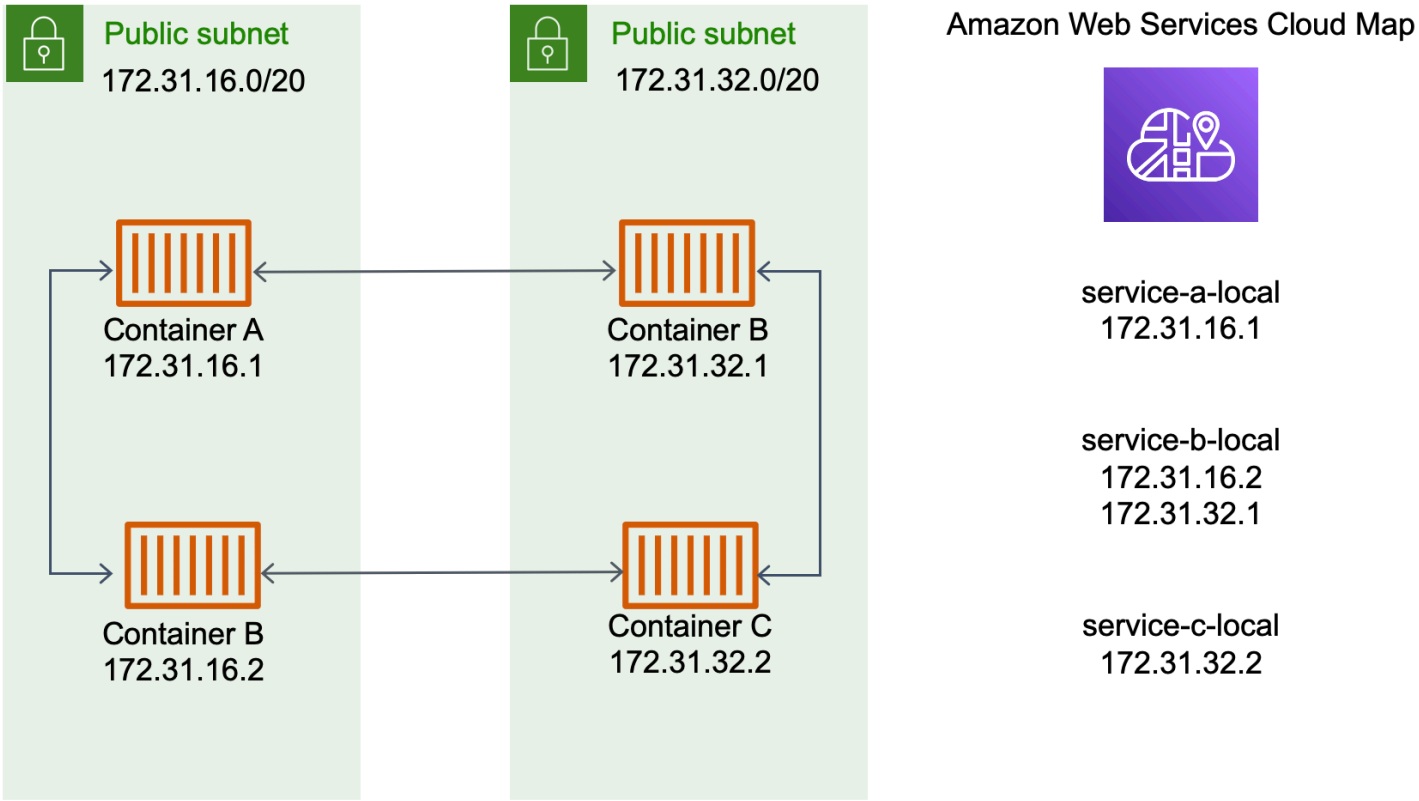


變更只會在部署期間發生

您可以在每個服務和任務定義內提供完整的組態。Amazon 會在每個服務部署中ECS管理此組態的變更，以確保部署中的所有任務都以相同的方式運作。例如，DNS當服務探索時， 的常見問題是控制遷移。如果您變更DNS名稱以指向新的替換 IP 地址，則可能需要最長TTL的時間，所有用戶端才會開始使用新的服務。透過 Service Connect，用戶端部署會取代用戶端任務來更新組態。您可以設定部署斷路器和其他部署組態，以與任何其他部署相同的方式影響 Service Connect 變更。

使用服務探索

另一種通訊方法是 service-to-service使用 服務探索進行直接通訊。在此方法中，您可以使用 AWS Cloud Map 服務探索與 Amazon 的整合ECS。使用服務探索，Amazon 會將啟動的任務清單ECS同步到 AWS Cloud Map，該清單會維護DNS主機名稱，以解析該特定服務中一或多個任務的內部 IP 地址。Amazon 中的其他服務VPC可以使用此DNS主機名稱，使用其內部 IP 地址直接將流量傳送到另一個容器。如需詳細資訊，請參閱服務探索。



在上圖中，有三種服務。 `service-a-local` 有一個容器，並與 通訊`service-b-local`，其中有兩個容器。 還`service-b-local`必須與 通訊`service-c-local`，其中有一個容器。這些服務中的每個容器都可以使用來自 的內部DNS名稱 AWS Cloud Map ，從其需要通訊的下游服務尋找容器的內部 IP 地址。

這種通訊方法 service-to-service提供低延遲。乍看之下，這也很簡單，因為容器之間沒有額外的元件。流量會直接從一個容器流向另一個容器。

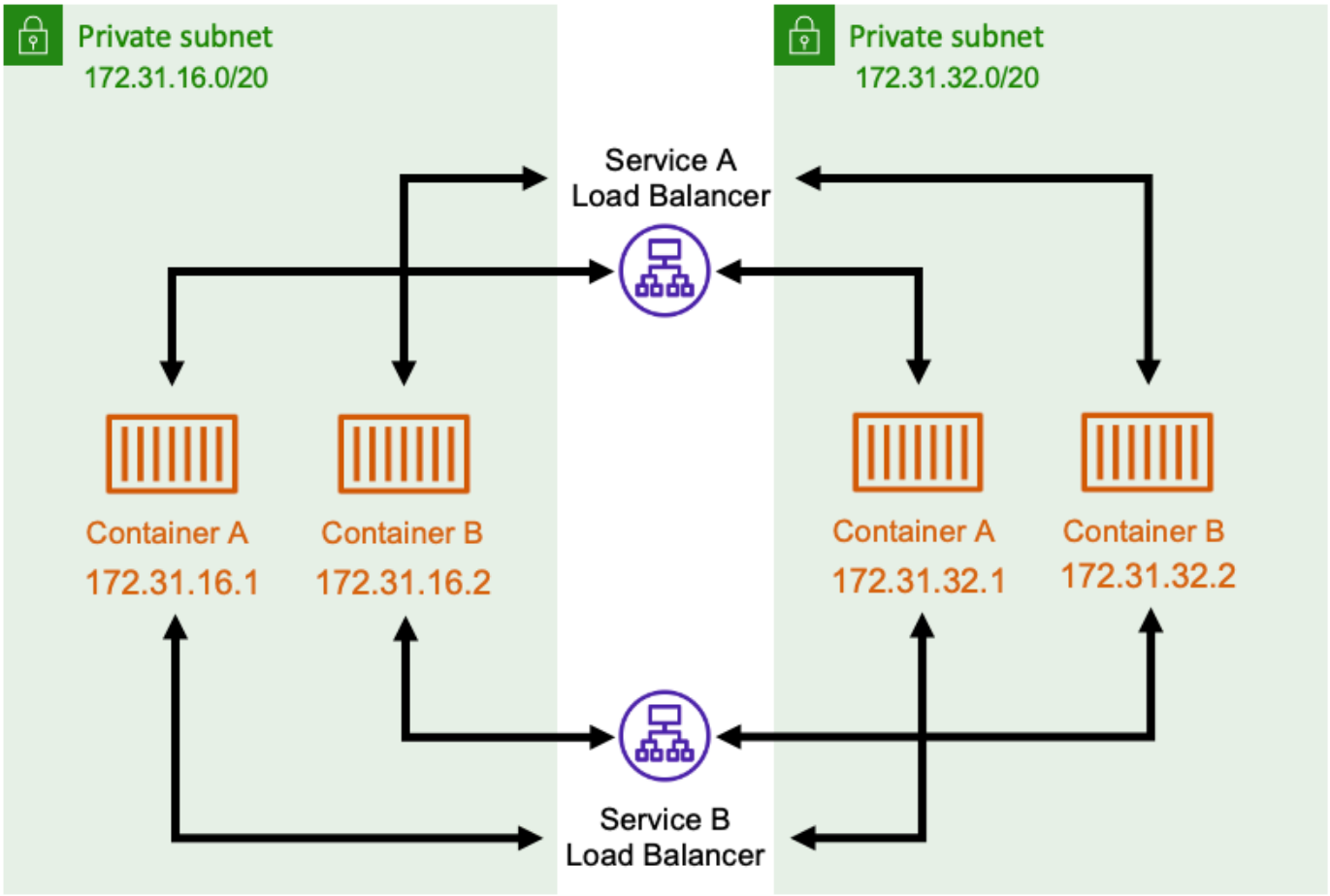
此方法適合使用awsvpc網路模式，其中每個任務都有自己的唯一 IP 地址。大多數軟體只支援使用DNSA記錄，這些記錄會直接解析為 IP 地址。使用awsvpc網路模式時，每個任務的 IP 地址都是A記錄。不過，如果您使用bridge網路模式，多個容器可能會共用相同的 IP 地址。此外，動態連接埠映射會導致容器在該單一 IP 地址上隨機指派連接埠號碼。此時，A記錄已不足以進行服務探索。您還必須使用 SRV記錄。這種類型的記錄可以追蹤 IP 地址和連接埠號碼，但要求您適當地設定應用程式。您使用的某些預先建置應用程式可能不支援SRV記錄。

awsvpc 網路模式的另一個優點是，每個服務都有唯一的安全群組。您可以設定此安全群組，只允許來自需要與該服務通訊之特定上游服務的傳入連線。

使用服務探索進行直接 service-to-service通訊的主要缺點是，您必須實作額外的邏輯，才能重試並處理連線失敗。 DNS記錄有 time-to-live(TTL) 期間，可控制快取的時間長度。記錄DNS更新和快取過期需要一些時間，以便您的應用程式可以挑選DNS記錄的最新版本。因此，您的應用程式最終可能會解析DNS記錄，以指向不再存在的另一個容器。您的應用程式需要處理重試，並具有邏輯來忽略不良後端。

使用內部負載平衡器

另一種通訊方式 service-to-service是使用內部負載平衡器。內部負載平衡器完全存在於 內部VPC，只有 內部的服務才能存取VPC。



負載平衡器透過將備援資源部署到每個子網路來維持高可用性。當來自 的容器serviceA需要與來自 的容器通訊時serviceB，它會開啟負載平衡器的連線。然後，負載平衡器會從 開啟與容器的連線service B。負載平衡器可做為管理每個服務之間所有連線的集中位置。

如果 的容器serviceB停止，負載平衡器可以從集區中移除該容器。負載平衡器也會針對其集區中的每個下游目標執行運作狀態檢查，並自動從集區移除不良目標，直到它們再次運作良好為止。這些應用程式不再需要知道有多少下游容器。他們只要開啟與負載平衡器的連線。

此方法適用於所有網路模式。負載平衡器在使用awsvpc網路模式時可以追蹤任務 IP 地址，以及在使用bridge網路模式時更進階的 IP 地址和連接埠組合。它將流量平均分配到所有 IP 地址和連接埠組合，即使多個容器實際上託管在相同的 Amazon EC2執行個體上，只是在不同連接埠上。

這種方法的一個缺點是成本。若要高度可用，負載平衡器需要在每個可用區域中擁有資源。這會增加額外的成本，因為支付負載平衡器的費用和經過負載平衡器的流量。

不過，您可以透過讓多個服務共用負載平衡器來降低額外負荷成本。這特別適合使用 Application Load Balancer REST的服務。您可以建立以路徑為基礎的路由規則，將流量路由到不同的 服務。例如，

/api/user/*可能會路由到user屬於服務一部分的容器，而 /api/order/*可能會路由到相關聯的order服務。使用此方法，您只需支付一個 Application Load Balancer，並URL為您的 設定一致的一個API。不過，您可以將流量分割為後端上的各種微服務。

跨 AWS 帳戶 和 聯網 Amazon ECS服務的最佳實務 VPCs

如果您是擁有多個團隊和分支的組織的一部分，您可能會將服務獨立部署到共用 VPCs內的個別 ，AWS 帳戶 或部署到與多個個人VPCs相關聯的 AWS 帳戶。無論您部署服務的方式為何，建議您補充聯網元件，以協助在 之間路由流量VPCs。為此，您可以使用數個 AWS 服務來補充現有的聯網元件。

- AWS Transit Gateway — 您應該先考慮此聯網服務。此服務可做為中央中樞 AWS 帳戶，用於在 Amazon VPCs、 和內部部署網路之間路由連線。如需詳細資訊，請參閱《Amazon VPC Transit Gateways 指南》中的什麼是傳輸閘道？。
- Amazon VPC和 VPN支援 — 您可以使用此服務建立 site-to-siteVPN連線，以將內部部署網路連線至您的 VPC。如需詳細資訊，請參閱AWS Site-to-Site VPN 《 使用者指南》中的什麼是 AWS Site-to-Site VPN？。
- Amazon VPC — 您可以使用 Amazon VPC對等互連，協助您VPCs在相同帳戶或跨帳戶連接多個 。如需詳細資訊，請參閱《Amazon VPC對等指南》中的什麼是對等互連？。 VPC
- 共用 VPCs — 您可以在多個 AWS 帳戶間使用 VPC和 VPC子網路。如需詳細資訊，請參閱《Amazon VPC使用者指南》中的使用共用VPCs的 。

AWS 適用於 Amazon ECS 網路故障診斷的 服務

下列服務和功能可協助您深入了解網路和服務組態。您可以使用此資訊來疑難排解聯網問題，並更好地最佳化您的服務。

CloudWatch 容器洞見

CloudWatch Container Insights 會從容器化應用程式和微服務收集、彙總和摘要指標和日誌。指標包括資源的使用率，例如 CPU、記憶體、磁碟和網路。它們可在自動儀表板中使用 CloudWatch。如需詳細資訊，請參閱《Amazon CloudWatch 使用者指南》中的在 Amazon 上設定 Container InsightsECS。

AWS X-Ray

AWS X-Ray 是一種追蹤服務，可用來收集應用程式提出之網路請求的相關資訊。您可以使用 SDK來檢測您的應用程式，並擷取服務之間，以及服務 AWS 與服務端點之間的流量時間和回應代碼。如需詳細資訊，請參閱 AWS X-Ray 開發人員指南中的什麼是 AWS X-Ray。

您也可以探索 服務如何彼此聯網的 AWS X-Ray 圖表。或者，使用它們來探索每個 service-to-service 連結執行方式的彙總統計資料。最後，您可以深入了解任何特定交易，以查看代表網路呼叫的客群與該特定交易的關聯。

您可以使用這些功能來識別是否有聯網瓶頸，或網路中的特定服務是否未如預期般執行。

VPC 流程日誌

您可以使用 Amazon VPC流程日誌來分析網路效能和偵錯連線問題。啟用VPC流程日誌後，您可以擷取 中所有連線的日誌VPC。這包括與 Elastic Load Balancing、Amazon RDS、NAT閘道和您可能正在使用的其他金鑰 AWS 服務相關聯的聯網介面連線。如需詳細資訊，請參閱《Amazon VPC使用者指南》中的VPC流程日誌。

網路調校秘訣

您可以微調一些設定，以改善您的聯網。

nofile ulimit

如果您預期應用程式具有高流量並處理許多並行連線，您應該考慮允許的檔案數量的系統配額。當許多網路通訊端開啟時，每個通訊端都必須以檔案描述符表示。如果您的檔案描述符配額太低，則會限制您的網路通訊端。這會導致連線失敗或發生錯誤。您可以更新 Amazon ECS任務定義中檔案數量的容器特定配額。如果您是在 Amazon EC2 (而非 AWS Fargate) 上執行，則您可能還需要調整基礎 Amazon EC2執行個體上的這些配額。

sysctl 網

另一個可調整設定類別是`sysctl`網路設定。您應該參考所選 Linux 發行版本的特定設定。其中許多設定會調整讀取和寫入緩衝區的大小。在某些情況下，這可以在執行具有大量容器的大型 Amazon EC2 執行個體時提供幫助。

使用帳戶設定存取 Amazon ECS功能

您可以前往 Amazon ECS帳戶設定，以選擇加入或退出特定功能。針對每個 AWS 區域，您可以在帳戶層級或針對特定的使用者或角色選擇加入或退出每個帳戶設定。

如果以下任何一項與您相關，則您可能會想要選擇加入或退出特定功能：

- 使用者或角色可以為其個別帳戶選擇加入或退出特定帳戶設定。

- 使用者或角色可以為帳戶上的所有使用者設定預設選擇加入或退出設定。
- 根使用者或具有管理員權限的使用者可以選擇加入或退出帳戶上的任何特定角色或使用者。如果根使用者的帳戶設定發生變更，對於未選取個別帳戶設定的所有使用者和角色，將會設為預設值。

> **Note**
>
> 聯合身分使用者會採用根使用者的帳戶設定，而且無法另外為他們設定明確的帳戶設定。

下列帳戶設定可供使用。您必須分別選擇加入和選擇退出每個帳戶設定。

資源名稱	進一步了解
`containerInsights`	Container Insights
`serviceLongArnFormat` `taskLongArnFormat` `containerInstanceLongArnFormat`	Amazon Resource Names (ARNs) 和 IDs
`tagResourceAuthorization`	標記授權
`fargateFIPSMode`	AWS Fargate 聯邦資訊處理標準 (FIPS-140) 合規
`fargateTaskRetirementWaitPeriod`	AWS Fargate 任務淘汰等待時間
`guardDutyActivate`	執行期監控 (Amazon GuardDuty 整合)
`dualStackIPv6`	雙堆疊 IPv6 VPC
`awsvpcTrunking`	增加 Linux 容器執行個體網路介面

Amazon Resource Names (ARNs) 和 IDs

建立 Amazon ECS 資源時，會為每個資源指派唯一的 Amazon Resource Name (ARN) 和資源識別符 (ID)。如果您使用命令列工具或 Amazon 來ECSAPI使用 Amazon ECS，IDs則某些命令需要 資源

ARNs或 。例如，如果您使用 stop-task AWS CLI 命令來停止任務，您必須在命令中指定任務ARN或ID。

Amazon ECS 正在為 Amazon Resource Names (ARNs) 引進新格式，並為 Amazon ECS服務、任務和容器執行個體IDs引進資源。每個資源類型的選擇加入狀態會決定資源使用的 Amazon Resource Name (ARN) 格式。您必須選擇加入新ARN格式，才能使用該資源類型的資源標記等功能。

您可以依區域選擇加入和選擇退出新的 Amazon Resource Name (ARN) 和資源 ID 格式。目前，預設會選擇加入建立的所有新帳戶。

您可以隨時選擇加入或選擇退出新的 Amazon Resource Name (ARN) 和資源 ID 格式。選擇加入後，您建立的任何新資源都會使用新的格式。

> **Note**
>
> 資源 ID 在建立之後即不會變更。因此，選擇加入或退出新格式不會影響您現有的資源IDs。

下列各節說明 ARN和資源 ID 格式的變更方式。如需轉換至新格式的詳細資訊，請參閱 Amazon Elastic Container Service。 FAQ

Amazon Resource Name (ARN) 格式

部分資源具有使用者易記名稱 (例如，名為 `production` 的服務)。在其他情況下，您必須使用 Amazon Resource Name (ARN) 格式指定資源。Amazon ECS任務、服務和容器執行個體的新ARN格式包含叢集名稱。如需選擇加入新ARN格式的詳細資訊，請參閱 修改 Amazon ECS帳戶設定。

下表顯示每個資源類型的當前格式和新格式。

資源類型	ARN
容器執行個體	當前：`arn:aws:ecs:` *`region`*`:`*`aws_account_id`* `:container-instance/` *`container-instance-id`* 新：`arn:aws:ecs:` *`region`*`:`*`aws_account_id`* `:container-instance/` *`cluster-name`* `/`*`container-instance-id`*
Amazon ECS服務	當前：`arn:aws:ecs:` *`region`*`:`*`aws_account_id`* `:service/` *`service-name`*

資源類型	ARN
	新：arn:aws:ecs: *region*:*aws_account_id* :service/ *cluster-name* /*service-name*
Amazon ECS任務	當前：arn:aws:ecs: *region*:*aws_account_id* :task/*task-id* 新：arn:aws:ecs: *region*:*aws_account_id* :task/*cluster-name* /*task-id*

資源 ID 長度

資源 ID 的形式採用字母和數字的唯一組合。新的資源 ID 格式包括較短IDs的 Amazon ECS任務和容器執行個體。當前資源 ID 格式長度為 36 個字元。新的 IDs 採用 32 個字元格式，不包含任何連字號。如需選擇加入新資源 ID 格式的資訊，請參閱 修改 Amazon ECS帳戶設定。

預設值為 enabled。

只有選擇加入後啟動的資源才會收到新的ARN和資源 ID 格式。所有現有的資源不會受到影響。若要讓Amazon ECS服務和任務轉換為新的ARN和資源 ID 格式，您必須重新建立服務或任務。若要將容器執行個體轉換為新的ARN和資源 ID 格式，必須耗盡容器執行個體，且必須啟動新的容器執行個體並註冊至叢集。

> **Note**
>
> Amazon ECS服務啟動的任務只有在 2018 年 11 月 16 日當天或之後建立服務，且建立服務的使用者已選擇加入新的任務格式時，才能接收新的ARN和資源 ID 格式。

ARN 和資源 ID 格式時間軸

Amazon 資源的新 Amazon Resource Name (ARN) 的加入和退出期間時間表，以及 Amazon ECS 資源的資源 ID 格式已於 2021 年 4 月 1 日結束。根據預設，所有帳戶都會選擇加入新格式。所有建立的新資源都會收到新格式，且您無法再選擇退出。

Container Insights

在 2024 年 12 月 2 日， AWS 發行了 Container Insights，並增強了 Amazon 的可觀測性ECS。此版本支援使用 Amazon EC2和 Fargate 啟動類型的 Amazon ECS叢集增強可觀測性。在 Amazon 上設定

Container Insights 並增強可觀測性後ECS，Container Insights 會自動收集詳細的基礎設施遙測，從叢集層級到環境中的容器層級，並在儀表板中顯示您的資料，讓您了解各種指標和維度。然後，您可以在 Container Insights 主控台上使用這些 out-of-the-box儀表板，以更好地了解容器的運作狀態和效能，並透過識別異常狀況來更快地緩解問題。

我們建議您使用 Container Insights 搭配增強的可觀測性，而不是 Container Insights，因為它可在您的容器環境中提供詳細的可見性，從而減少解決的平均時間。如需詳細資訊，請參閱Amazon CloudWatch 《使用者指南》中的具有增強型可觀測性指標的 Amazon ECS Container Insights。

`containerInsights` 帳戶設定的預設值為 `disabled`。

具有增強可觀測性的容器洞見

使用下列命令，以增強可觀測性開啟 Container Insights。

將`containerInsights`帳戶設定設為 `enhanced`。

```
aws ecs put-account-setting --name containerInsights --value enhanced
```

範例輸出

```
{
    "setting": {
        "name": "containerInsights",
        "value": "enhanced",
        "principalArn": "arn:aws:iam::123456789012:johndoe",
         "type": user
    }
}
```

設定此帳戶設定後，所有新叢集會自動使用 Container Insights 並增強可觀測性。使用 `update-cluster-settings`命令將具有增強型可觀測性的 Container Insights 新增至現有叢集，或將叢集從 Container Insights 升級到具有增強型可觀測性的 Container Insights。

```
aws ecs update-cluster-settings --cluster cluster-name --settings
 name=containerInsights,value=enhanced
```

您也可以使用 主控台來設定 Container Insights 與增強的可觀測性。如需詳細資訊，請參閱修改 Amazon ECS帳戶設定。

Container Insights

當您將`containerInsights`帳戶設定設定為 時`enabled`，所有新叢集預設都會啟用 Container Insights。您可以使用 修改現有的叢集`update-cluster-settings`。

若要使用 Container Insights，請將`containerInsights`帳戶設定設為 `enabled`。使用下列命令來開啟 Container Insights。

```
aws ecs put-account-setting --name containerInsights --value enabled
```

範例輸出

```
{
    "setting": {
        "name": "containerInsights",
        "value": "enabled",
        "principalArn": "arn:aws:iam::123456789012:johndoe",
         "type": user
    }
}
```

當您將`containerInsights`帳戶設定設定為 時`enabled`，所有新叢集預設都會啟用 Container Insights。使用 `update-cluster-settings`命令將 Container Insights 新增至現有叢集。

```
aws ecs update-cluster-settings --cluster cluster-name --settings
 name=containerInsights,value=enabled
```

您也可以使用 主控台來設定 Container Insights。如需詳細資訊，請參閱修改 Amazon ECS帳戶設定。

AWS Fargate 聯邦資訊處理標準 (FIPS-140) 合規

Fargate 支援聯邦資訊處理標準 (FIPS-140)，該標準指定了保護敏感資訊之密碼編譯模組的安全要求。這是目前美國和加拿大政府標準，適用於必須符合美國聯邦資訊安全管理法案 (FISMA) 或聯邦風險與授權管理計劃 (Fed) 的系統RAMP。

資源名稱為 `fargateFIPSMode`。

預設值為 `disabled`。

您必須在 Fargate 上開啟聯邦資訊處理標準 (FIPS-140) 合規。如需詳細資訊，請參閱the section called “AWS Fargate FIPS-140 合規”。

> **Important**
>
> `fargateFIPSMode` 帳戶設定只能使用 Amazon ECSAPI或 進行變更 AWS CLI。如需詳細資訊，請參閱修改 Amazon ECS帳戶設定。

執行 `put-account-setting-default` 並將 `fargateFIPSMode` 選項設定為 `enabled`。如需詳細資訊，請參閱《Amazon Elastic Container Service API參考put-account-setting-default》中的 。

- 您可以使用下列命令來開啟 FIPS-140 合規。

```
aws ecs put-account-setting-default --name fargateFIPSMode --value enabled
```

範例輸出

```
{
    "setting": {
        "name": "fargateFIPSMode",
        "value": "enabled",
        "principalArn": "arn:aws:iam::123456789012:root",
         "type": user
    }
}
```

您可以執行 `list-account-settings`來檢視目前的 FIPS-140 合規狀態。使用 `effective-settings` 選項來檢視帳戶層級設定。

```
aws ecs list-account-settings --effective-settings
```

標記授權

Amazon ECS 正在引入資源建立的標記授權。使用者必須具有建立資源之動作的標記許可，例如 `ecsCreateCluster`。當您建立資源並指定該資源的標籤時， AWS 會執行額外的授權，以確認有建立標籤的許可。因此，您必須授予使用 `ecs:TagResource` 動作的明確許可。如需詳細資訊，請參閱the section called “在建立期間標記資源”。

若要選擇加入標記授權，請執行 `put-account-setting-default`，並將 `tagResourceAuthorization` 選項設為 `enable`。如需詳細資訊，請參閱《Amazon Elastic

Container Service API參考put-account-setting-default》中的 。您可以執行 `list-account-settings` 以檢視目前的標記授權狀態。

- 您可以使用下列命令來啟用標記授權。

```
aws ecs put-account-setting-default --name tagResourceAuthorization --value on --
region region
```

範例輸出

```
{
    "setting": {
        "name": "tagResourceAuthorization",
        "value": "on",
        "principalArn": "arn:aws:iam::123456789012:root",
        "type": user
    }
}
```

啟用標記授權之後，您必須設定適當的許可，以允許使用者在建立時標記資源。如需詳細資訊，請參閱the section called “在建立期間標記資源”。

您可以執行 `list-account-settings` 以檢視目前的標記授權狀態。使用 `effective-settings` 選項來檢視帳戶層級設定。

```
aws ecs list-account-settings --effective-settings
```

標記授權時間表

您可以透過執行 `list-account-settings` 檢視 `tagResourceAuthorization` 值來確認標記授權是否處於作用中狀態。當值為 on 時，表示正在使用標記授權。如需詳細資訊，請參閱《Amazon Elastic Container Service API參考list-account-settings》中的 。

以下是與標記授權相關的重要日期。

- 2023 年 4 月 18 日 – 引入標記授權。所有新帳戶和現有帳戶均須選擇加入使用該功能。您可以選擇加入以開始使用標記授權。選擇加入後，您必須授予適當的許可。
- 2024 年 2 月 9 日至 2024 年 3 月 6 日 – 所有新帳戶和非受影響的現有帳戶預設會標記授權。您可以啟用或停用`tagResourceAuthorization`帳戶設定，以驗證您的IAM政策。

AWS 已通知受影響的帳戶。

若要停用此功能，請`put-account-setting-default`執行 ，並將 `tagResourceAuthorization`選項設定為 `off`。

- 2024 年 3 月 7 日 – 如果您已啟用標記授權，則無法再停用帳戶設定。

  我們建議您在此日期之前完成IAM政策測試。
- 2024 年 3 月 29 日 – 所有帳戶都使用標記授權。帳戶層級設定將無法再在 Amazon ECS主控台或 中使用 AWS CLI。

AWS Fargate 任務淘汰等待時間

AWS 當您在標記為淘汰的平台版本修訂上執行 Fargate 任務時， 會傳送通知。如需詳細資訊，請參閱Amazon ECS AWS Fargate 的任務淘汰和維護 。

AWS 負責修補和維護 AWS Fargate 的基礎基礎設施。當 AWS 判斷在 Fargate 上託管的 Amazon ECS任務需要安全或基礎設施更新時，需要停止任務並啟動新任務來取代它們。您可以設定任務淘汰以進行修補之前的等待期。您可以選擇立即淘汰任務、等待 7 個日曆天，或等待 14 個日曆天。

此設定位於帳戶層級。

您可以設定 Fargate 開始任務淘汰的時間。對於需要立即套用更新的工作負載，請選擇立即設定 (0)。當您需要更多控制項時，例如當任務只能在特定時段中停止時，請設定 7 日 (7) 或 14 日 (14) 選項。

我們建議您選擇較短的等待期，以便更快獲得更新的平台版本修訂版。

透過執行 `put-account-setting-default`或 `put-account-setting` 做為根使用者或管理使用者來設定等待期間。對 `name` 使用 `fargateTaskRetirementWaitPeriod` 選項，並將 `value` 選項設定為以下值之一：

- `0` - AWS 傳送通知，並立即開始淘汰受影響的任務。
- 7 - AWS 傳送通知，並等待 7 個日曆天，再開始淘汰受影響的任務。
- 14 - AWS 傳送通知，並等待 14 個日曆日，然後再開始淘汰受影響的任務。

預設值是 7 天。

如需詳細資訊，請參閱《Amazon Elastic Container Service API參考put-account-setting》中的 put-account-setting-default和 。

您可以執行下列命令，將等待期設定為 14 天。

```
aws ecs put-account-setting-default --name fargateTaskRetirementWaitPeriod --value 14
```

範例輸出

```
{
    "setting": {
        "name": "fargateTaskRetirementWaitPeriod",
        "value": "14",
        "principalArn": "arn:aws:iam::123456789012:root",
        "type: user"
    }
}
```

您可以執行 `list-account-settings` 以檢視目前的 Fargate 任務淘汰等待時間。設定 `effective-settings` 選項。

```
aws ecs list-account-settings --effective-settings
```

增加 Linux 容器執行個體網路介面

每個使用 `awsvpc` 網路模式的 Amazon ECS任務都會收到自己的彈性網路介面 (ENI)，其會連接至託管它的容器執行個體。可連接到 Amazon EC2執行個體的網路介面數量有預設限制，而主要網路介面會計為一個。例如，根據預設，`c5.large`執行個體最多可ENIs連接三個執行個體。執行個體的主要網路介面會計為一個，因此您可以將額外的兩個連接到ENIs執行個體。因為使用`awsvpc`網路模式的每個任務都需要 ENI，您通常只能在此執行個體類型上執行兩個這類任務。

Amazon ECS支援啟動容器執行個體 ENI 使用支援的 Amazon EC2執行個體類型。當您使用這些執行個體類型並開啟`awsvpcTrunking`帳戶設定時，新啟動的容器執行個體ENIs上會提供額外的 。此組態可讓您在每個容器執行個體中安排更多任務。

例如，使用 的`c5.large`執行個體`awsvpcTrunking`有增加的 ENI 限制為 12。容器執行個體將具有主要網路介面，Amazon ECS會建立「trunk」網路介面並將其連接至容器執行個體。因此，此組態可讓您在容器執行個體中啟動十項任務，而不是目前的兩項任務。

執行期監控 (Amazon GuardDuty 整合)

執行期監控是一種智慧型威脅偵測服務，透過持續監控 AWS 日誌和聯網活動，以識別惡意或未經授權的行為，來保護 Fargate 和EC2容器執行個體上執行的工作負載。

`guardDutyActivate` 參數在 Amazon 中為唯讀ECS，並指出 Amazon ECS帳戶中的安全管理員是否開啟或關閉執行期監控。 會代表您 GuardDuty 控制此帳戶設定。如需詳細資訊，請參閱使用執行期監控保護 Amazon ECS工作負載。

您可以執行 `list-account-settings` 來檢視目前的 GuardDuty 整合設定。

```
aws ecs list-account-settings
```

範例輸出

```
{
    "setting": {
        "name": "guardDutyActivate",
        "value": "on",
        "principalArn": "arn:aws:iam::123456789012:doej",
        "type": aws-managed"
    }
}
```

雙堆疊 IPv6 VPC

除了主要私有IPv6地址之外，Amazon ECS還支援為任務提供 IPv4地址。

若要讓 任務接收IPv6地址，該任務必須使用 `awsvpc` 網路模式，必須在VPC設定為雙堆疊模式的 中啟動，且必須啟用`dualStackIPv6`帳戶設定。如需其他需求的詳細資訊，請參閱 在雙堆疊模式下使用 VPC 以取得EC2啟動類型，以及 在雙堆疊模式下使用 VPC 以取得 Fargate 啟動類型。

> **Important**
>
> `dualStackIPv6` 帳戶設定只能使用 Amazon ECSAPI或 進行變更 AWS CLI。如需詳細資訊，請參閱修改 Amazon ECS帳戶設定。

如果您在IPv6啟用的子網路中使用`awsvpc`網路模式執行任務的日期是 2020 年 10 月 1 日至 2020 年 11 月 2 日，則任務在 中執行的區域中的預設`dualStackIPv6`帳戶設定為 `disabled`。如果不符合該條件，則該區域中的預設 `dualStackIPv6` 設定為 `enabled`。

預設值為 `disabled`。

使用主控台檢視 Amazon ECS帳戶設定

在主控台中檢視您的帳戶設定，以查看您可以存取哪些功能。

> **Important**
>
> 只能使用 AWS CLI來檢視或變更 `dualStackIPv6`、`fargateFIPSMode` 和 `fargateTaskRetirementWaitPeriod` 帳戶設定。

1. 在 https://console.aws.amazon.com/ecs/v2 開啟主控台。
2. 在頂部導覽列中，選取要檢視帳戶設定的區域。
3. 在導覽頁面中，選擇帳戶設定。

修改 Amazon ECS帳戶設定

修改您的帳戶設定以存取 Amazon ECS功能。

`guardDutyActivate` 參數在 Amazon 中為唯讀ECS，並指出 Amazon ECS帳戶中的安全管理員是否開啟或關閉執行期監控。 會代表您 GuardDuty 控制此帳戶設定。如需詳細資訊，請參閱使用執行期監控保護 Amazon ECS工作負載。

> **Important**
>
> 只能使用 AWS CLI來檢視或變更 `dualStackIPv6`、`fargateFIPSMode` 和 `fargateTaskRetirementWaitPeriod` 帳戶設定。

1. 在 https://console.aws.amazon.com/ecs/v2 開啟主控台。
2. 在頂部導覽列中，選取要檢視帳戶設定的區域。
3. 在導覽頁面中，選擇 Account Settings (帳戶設定)。
4. 選擇更新。
5. 若要增加或減少每個EC2執行個體在 awsvpc 網路模式中執行的任務數量，請在 AWSVPC Trunking 下選取 AWSVPC Trunking。
6. 若要使用或停用叢集的 CloudWatch Container Insights，請在 CloudWatch Container Insights 可觀測性下，選擇下列其中一個選項：

- 若要使用具有增強型可觀測性的 Container Insights，請選擇具有增強型可觀測性的 Container Insights。
- 若要使用 Container Insights，請選擇 Container Insights。
- 若要停止使用 Container Insights，請選擇關閉。

7. 若要啟用或停用標記授權，請在資源標記授權下，選取或清除資源標記授權。
8. 選擇 Save changes (儲存變更)。
9. 在確認畫面，選擇確認以儲存選項。

後續步驟

如果您使用增強型可觀測性開啟 Container Insights 或 Container Insights，您可以選擇更新現有叢集以使用 功能。如需詳細資訊，請參閱更新 Amazon ECS 叢集。

還原至預設 Amazon ECS帳戶設定

您可以使用 AWS Management Console 將 Amazon ECS帳戶設定還原為預設值。

只有當帳戶設定不再是預設設定時，才能使用還原為帳戶預設值選項。

1. 在 https://console.aws.amazon.com/ecs/v2 開啟主控台。
2. 在頂部導覽列中，選取要檢視帳戶設定的區域。
3. 在導覽頁面中，選擇 Account Settings (帳戶設定)。
4. 選擇更新。
5. 選擇還原為帳戶預設值。
6. 在確認畫面，選擇確認以儲存選項。

使用 管理 Amazon ECS帳戶設定 AWS CLI

您可以使用 Amazon ECS API、 AWS CLI 或 管理您的帳戶設定SDKs。`dualStackIPv6`、`fargateFIPSMode`和 `fargateTaskRetirementWaitPeriod`帳戶設定只能使用這些工具檢視或變更。

如需任務定義可用API動作的相關資訊，請參閱《Amazon Elastic Container Service API參考》中的帳戶設定動作。

使用以下其中一個命令來為帳戶中所有使用者或角色修改預設帳戶設定。除非使用者或角色自行明確覆寫這些設定，否則這些變更適用於整個 AWS 帳戶。

- put-account-setting-default (AWS CLI)

```
aws ecs put-account-setting-default --name serviceLongArnFormat --value enabled --
region us-east-2
```

  您也可以使用此命令來修改其他帳戶設定。若要執行此操作，請使用對應的帳戶設定來取代 name 參數。

- Write-ECSAccountSetting (AWS Tools for Windows PowerShell)

```
Write-ECSAccountSettingDefault -Name serviceLongArnFormat -Value enabled -Region us-
east-1 -Force
```

修改使用者帳戶的帳戶設定 (AWS CLI)

使用以下其中一個命令來修改使用者的帳戶設定。若您以根使用者身分使用這些命令，變更會套用到整個 AWS 帳戶，除非使用者或角色自行明確覆寫這些設定。

- put-account-setting (AWS CLI)

```
aws ecs put-account-setting --name serviceLongArnFormat --value enabled --region us-
east-1
```

  您也可以使用此命令來修改其他帳戶設定。若要執行此操作，請使用對應的帳戶設定來取代 name 參數。

- Write-ECSAccountSetting (AWS Tools for Windows PowerShell)

```
Write-ECSAccountSetting -Name serviceLongArnFormat -Value enabled -Force
```

修改特定使用者或角色的帳戶設定 (AWS CLI)

使用下列其中一個命令，並在請求中指定ARN使用者、角色或根使用者的 ，以修改特定使用者或角色的帳戶設定。

- put-account-setting (AWS CLI)

```
aws ecs put-account-setting --name serviceLongArnFormat --value enabled --principal-
arn arn:aws:iam::aws_account_id:user/principalName --region us-east-1
```

您也可以使用此命令來修改其他帳戶設定。若要執行此操作，請使用對應的帳戶設定來取代 name 參數。

- Write-ECSAccountSetting (AWS Tools for Windows PowerShell)

```
Write-ECSAccountSetting -Name serviceLongArnFormat -Value enabled -PrincipalArn
 arn:aws:iam::aws_account_id:user/principalName -Region us-east-1 -Force
```

IAM Amazon 的 角色 ECS

IAM 角色是您可以在帳戶中建立的IAM身分，具有特定許可。在 Amazon 中ECS，您可以建立角色，以授予 Amazon ECS 資源許可，例如容器或服務。

Amazon ECS所需的角色取決於任務定義啟動類型和您使用的功能。使用下表來判斷您需要哪些Amazon IAM角色ECS。

角色	定義	需要時	其他資訊
任務執行角色	此角色允許 Amazon AWS 代表您ECS使用其他服務。	您的任務託管在 AWS Fargate外部執行個體上或外部執行個體上，並且： • 從 Amazon ECR私有儲存庫提取容器映像。 • 從執行任務的帳戶的不同帳戶中的 Amazon ECR私有儲存庫提取容器映像。 • 會使用 CloudWatch 日誌驅動程式	Amazon ECS 任務執行 IAM 角色

角色	定義	需要時	其他資訊
		將容器日誌傳送至`awslogs`日誌。 您的任務託管在 AWS Fargate 或 Amazon EC2執行個體上，並且： • 使用私有登錄驗證。 • 使用執行期監控。 • 任務定義使用 Secrets Manager 秘密或 AWS Systems Manager 參數存放區參數來參考敏感資料。	
任務角色	此角色可讓您的應用程式程式碼（在容器上）使用其他 AWS 服務。	您的應用程式會存取其他服務 AWS ，例如 Amazon S3。	Amazon ECS 任務 IAM 角色
容器執行個體角色	此角色可讓您的EC2執行個體或外部執行個體向叢集註冊。	您的任務託管在 Amazon EC2執行個體或外部執行個體上。	Amazon ECS 容器執行個體 IAM 角色
Amazon ECS Anywhere 角色	此角色可讓您的外部執行個體存取 AWS APIs。	您的任務託管在外部執行個體上。	Amazon ECS Anywhere IAM 角色
Amazon ECS CodeDeploy 角色	此角色允許 CodeDeploy 更新 服務。	您可以使用 CodeDeploy 藍/綠部署類型來部署 服務。	Amazon ECS CodeDeploy IAM 角色

角色	定義	需要時	其他資訊
Amazon ECS EventBridge 角色	此角色允許 EventBridge 更新 服務。	您可以使用 EventBridge 規則和目標來排程任務。	Amazon ECS EventBridge IAM 角色
Amazon ECS基礎設施角色	此角色可讓 Amazon ECS管理叢集中的基礎設施資源。	• 您想要將 Amazon EBS磁碟區連接至 Fargate 或EC2啟動類型 Amazon ECS 任務。基礎設施角色可讓 Amazon 為您的任務ECS管理 Amazon EBS磁碟區。 • 您想要使用 Transport Layer Security (TLS) 來加密 Amazon ECS Service Connect 服務之間的流量。 • 您想要建立 VPC Lattice 目標群組。	Amazon ECS 基礎設施 IAM 角色

Amazon ECS 任務定義

任務定義是您應用程式的藍圖。其是一種 JSON 格式的文字檔案，描述了構成應用程式的參數和一個或多個容器。

以下是您可在任務定義中指定的一些參數：

- 要使用的啟動類型，決定您任務託管所在的基礎設施
- 和您任務中每個容器一起使用的 Docker 映像
- 每個任務或任務中每個容器使用多少 CPU 和記憶體
- 記憶體和 CPU 需求
- 任務執行所在的容器作業系統
- 您任務中的容器所使用的 Docker 聯網模式
- 用於任務的記錄組態
- 如果容器完成或失敗，任務是否繼續執行
- 容器啟動時執行的命令
- 任務中的容器使用的任何資料磁碟區
- 任務使用的 IAM 角色

如需有關任務定義參數的完整清單，請參閱 Amazon ECS 任務定義參數。

建立任務定義之後，可以將任務定義當做任務或服務來執行。

- 任務是在叢集內將任務定義執行個體化。您在 Amazon ECS 內為應用程式建立任務定義後，可以指定要在您叢集上執行的任務數量。
- Amazon ECS 服務在 Amazon ECS 叢集中同時執行和維護您所需的任務數量。其運作方式為，如果有任何任務因任何原因而出現故障或停止，Amazon ECS 服務排程器就會根據您的任務定義啟動另一個執行個體。這樣就可以取代該任務，從而在服務中保持所需的任務數量。

主題

- 使用主控台取消註冊 Amazon ECS 任務定義修訂
- 使用主控台刪除 Amazon ECS 任務定義修訂
- Amazon ECS 任務定義使用案例
- Amazon ECS 任務定義參數
- Amazon ECS 任務定義範本
- Amazon ECS 任務定義範例

Amazon ECS 任務定義狀態

當您建立、取消註冊或刪除任務定義時，任務定義會變更狀態。您可以在 主控台或使用 檢視任務定義狀態DescribeTaskDefinition。

以下是可能出現的任務定義狀態：

ACTIVE

任務定義在 Amazon ECS 上註冊之後就會是 ACTIVE 狀態。您可以使用狀態為 ACTIVE 的任務定義來執行任務或建立服務。

非作用中

取消註冊任務定義後，任務定義會從 ACTIVE 狀態轉變為 INACTIVE 狀態。您可以透過呼叫 DescribeTaskDefinition 來擷取處於 INACTIVE 狀態的任務定義。您不能使用狀態為 INACTIVE 的任務定義，來執行新任務或建立新服務。這不會對現有服務或任務造成影響。

DELETE_IN_PROGRESS

提交要刪除的任務定義後，這些任務定義會從 INACTIVE 狀態轉換為 DELETE_IN_PROGRESS 狀態。任務定義處於 DELETE_IN_PROGRESS 狀態後，Amazon ECS 會定期驗證目標任務定義未被任何作用中的任務或部署引用，然後永久刪除該任務定義。您不能使用狀態為 DELETE_IN_PROGRESS 的任務定義，來執行新任務或建立新服務。任務定義可以隨時提交刪除，而不會影響現有的任務和服務。

您可以在主控台中檢視狀態為 DELETE_IN_PROGRESS 的任務定義，且可以透過呼叫 DescribeTaskDefinition 來擷取這些任務定義。

當您刪除所有 INACTIVE 任務定義修訂時，任務定義名稱不會顯示在主控台中，也不會在 API 中傳回。如果任務定義修訂處於 DELETE_IN_PROGRESS 狀態，則任務定義名稱會顯示在主控台中，並在 API 中傳回。Amazon ECS 會保留任務定義名稱，下次您使用該名稱建立任務定義時，修訂版本將會遞增。

如果您使用 AWS Config 來管理您的任務定義， 會針對所有任務定義註冊 AWS Config 向您收取費用。您只需支付將最新處於 ACTIVE 狀態的任務定義取消註冊的費用。刪除任務定義無須付費。如需定價的詳細資訊，請參閱AWS Config 定價。

可以封鎖刪除的 Amazon ECS 資源

當有任何 Amazon ECS 資源依賴於任務定義修訂時，任務定義刪除請求將不會完成。以下資源可能會阻止任務定義遭刪除：

- Amazon ECS 獨立任務 - 需要任務定義，才能讓任務正常運作。
- Amazon ECS 服務任務 - 需要任務定義，才能讓任務正常運作。
- Amazon ECS 服務部署和任務集 - 為 Amazon ECS 部署或任務集啟動擴展事件時，需要任務定義。

如果您的任務定義仍處於 DELETE_IN_PROGRESS 狀態，您可以使用 主控台或 AWS CLI 來識別，然後停止封鎖任務定義刪除的資源。

移除封鎖的資源後刪除任務定義

移除封鎖任務定義刪除的資源後，以下規則適用：

- Amazon ECS 任務 - 任務停止後，任務定義刪除最多可能需要 1 小時才能完成。
- Amazon ECS 服務部署和任務集 - 在刪除部署或任務集之後，刪除任務定義最多可能需要 24 小時才能完成。

為 Amazon ECS 建構您的應用程式

您可以透過為應用程式建立任務定義來建構應用程式。任務定義包含定義應用程式相關資訊的參數，包括：

- 要使用的啟動類型，決定任務託管所在的基礎設施。

  當您使用 EC2 啟動類型時，您也可以選擇執行個體類型。對於某些執行個體類型，例如 GPU，您需要設定其他參數。如需詳細資訊，請參閱Amazon ECS 任務定義使用案例。
- 容器映像，其中會存放您的應用程式程式碼，以及應用程式程式碼執行所需的所有相依性。
- 用於任務中容器的聯網模式

  網路模式決定您的任務如何透過網路進行通訊。

對於在 EC2 執行個體上執行的任務，有多個選項，但我們建議您使用awsvpc網路模式。awsvpc 網路模式可簡化容器聯網，因為您更能控制應用程式如何與 VPCs內的彼此和其他服務通訊。

對於在 Fargate 上執行的任務，您只能使用awsvpc網路模式。

- 要用於任務的記錄組態。
- 與任務中的容器搭配使用的任何資料磁碟區。

如需有關任務定義參數的完整清單，請參閱 Amazon ECS 任務定義參數。

建立任務定義時，請遵循下列準則：

- 將每個任務定義系列僅用於一個業務目的。

  如果您將多種類型的應用程式容器分組在相同的任務定義中，則無法獨立擴展這些容器。例如，網站和 API 不太可能都需要以相同的速率進行橫向擴展。隨著流量增加，所需的 Web 容器數量將與 API 容器不同。如果這兩個容器部署在相同的任務定義中，則每個任務都會執行相同數量的 Web 容器和 API 容器。
- 將每個應用程式版本與任務定義系列中的任務定義修訂版進行比對。

  在任務定義系列中，將每個任務定義修訂視為特定容器映像設定的時間點快照集。這類似於容器是執行特定應用程式程式碼版本所需之所有內容的快照。

  請確定應用程式程式碼版本、容器映像標記和任務定義修訂版本之間存在一對一的映射。典型的發佈過程涉及 git commit，該提交會轉換為使用 git commit SHA 標記的容器映像。然後，該容器映像標籤會取得自己的 Amazon ECS 任務定義修訂版。最後，Amazon ECS 服務已更新，以指示其部署新的任務定義修訂版本。
- 針對每個任務定義系列使用不同的 IAM 角色。

  使用自己的 IAM 角色定義每個任務定義。此建議應與我們為每個業務組件提供自己的任務定義系列的建議一起完成。透過實作這兩個最佳實務，您可以限制每個服務對您 AWS 帳戶中資源的存取量。例如，您可以授予身分驗證服務存取權限以連線到您的密碼資料庫。同時，您還可以確保只有您的訂單服務才能存取信用卡付款資訊。

Amazon ECS 容器映像的最佳實務

容器映像是一組關於如何建置容器的指示。容器映像會包含您應用程式的程式碼，以及應用程式的程式碼執行所需的所有相依性。應用程式相依性包括應用程式程式碼所依賴的原始程式碼套件、解譯語言的語言執行期，以及動態連結程式碼所依賴的二進位套件。

設計和建置容器映像時，請遵循下列準則：

- 將所有應用程式相依性儲存為容器映像內的靜態檔案，使容器映像完整。

  如果您變更容器映像中的某些內容，請使用變更建立新的容器映像。
- 在容器內執行單一應用程式處理程序。

  容器生命週期是應用程式處理程序執行的長度。Amazon ECS 會取代損毀的程序，並決定從何處啟動取代程序。完整映像可讓整體部署更具彈性。
- 讓您的應用程式處理 `SIGTERM`。

  當 Amazon ECS 停止任務時，它會先傳送 SIGTERM 訊號給任務，通知應用程式需要完成並關閉。然後，Amazon ECS 會傳送訊息`SIGKILL`。當應用程式忽略 時`SIGTERM`，Amazon ECS 服務必須等待 傳送`SIGKILL`訊號以終止程序。

  您需要識別應用程式完成其工作所需的時間，並確保應用程式處理`SIGTERM`訊號。應用程式訊號處理需要停止應用程式接受新工作並完成進行中的工作，或將未完成的工作儲存到任務外的儲存體，當工作花費太久才完成。
- 設定容器化應用程式以將日誌寫入 `stdout` 和 `stderr`。

  將日誌處理與應用程式程式碼分離，可讓您靈活地調整基礎設施層級的日誌處理。其中一個範例是變更您的記錄系統。您可以調整設定，而不是修改您的服務，以及建置和部署新的容器映像。
- 使用標籤對容器映像進行版本控制。

  容器映像儲存在容器登錄中。登錄中的每個映像皆以標籤識別。有一個標籤名為 `latest`。此標籤的功能是指向最新版應用程式容器映像的指標，類似於 git 儲存庫中的 HEAD。建議您僅將 `latest` 標記用於測試目的。最佳實務是使用每個建置的唯一標籤來標記容器映像。我們建議您使用用於建置映像的 git commit 的 git SHA 來標記映像。

  您不需要為每次提交建置容器映像。不過，我們建議您在每次將特定程式碼提交發佈至生產環境時，都建置新的容器映像。我們還建議您使用與映像內部程式碼的 git commit 相對應的標籤來標記映像。如果您使用 git commit 標記映像，則可以更快找到映像正在執行的程式碼版本。

我們也建議您在 Amazon Elastic Container Registry 中開啟不可變的映像標籤。使用此設定時，您無法變更標記指向的容器映像。反之，Amazon ECR 會強制執行必須將新映像上傳至新標籤。如需詳細資訊，請參閱《Amazon ECR 使用者指南》中的映像標籤可變性。

當您建構應用程式以在 上執行時 AWS Fargate，您必須決定將多個容器部署到相同的任務定義，以及在多個任務定義中分別部署容器。如需要下列條件，我們建議您在相同的任務定義中部署多個容器：

- 容器共用相同的生命週期 (亦即，它們一起啟動和終止)。
- 容器必須在相同的基礎主機上執行 (亦即，一個容器參考 localhost 連接埠上的另一個容器)。
- 您的容器共用資源。
- 您的容器共用資料磁碟區。

如果不需要這些條件，我們建議在多個任務定義中分別部署容器。這可讓您分別擴展、佈建和取消佈建容器。

Amazon ECS 任務大小的最佳實務

您的容器和任務大小對於擴展和容量規劃都至關重要。在 Amazon ECS 中，CPU 和記憶體是用於容量的兩個資源指標。CPU 的測量單位為完整 vCPU 的 1/1024 （其中 1024 個單位等於整個 vCPU)。記憶體的測量單位為 MB。您可以在任務定義中設定資源保留和限制。

設定保留時，您要設定任務所需的資源數量下限。您的任務至少會收到請求的資源量。您的應用程式可能可以使用比您宣告的保留更多的 CPU 或記憶體。不過，這受限於您同時宣告的任何限制。使用超過預留數量稱為爆量。在 Amazon ECS 中，保證保留。例如，如果您使用 Amazon EC2 執行個體來提供容量，Amazon ECS 不會將任務放置在無法履行保留的執行個體上。

限制是您的容器或任務可以使用的 CPU 單位或記憶體數量上限。任何嘗試使用超過此限制的 CPU 都會導致限流。任何嘗試使用更多記憶體會導致您的容器停止。

選擇這些值可能具有挑戰性。這是因為最適合您應用程式的值，很大程度上取決於您應用程式的資源需求。負載測試您的應用程式是成功規劃資源需求的關鍵，並更好地了解應用程式的需求。

無狀態應用程式

對於水平擴展的無狀態應用程式，例如負載平衡器後方的應用程式，我們建議您先判斷應用程式在提供請求時耗用的記憶體量。若要這樣做，您可以使用傳統工具，例如 `ps`或 `top`，或監控解決方案，例如 CloudWatch Container Insights。

判斷 CPU 保留時，請考慮如何擴展應用程式以符合您的業務需求。您可以使用較小的 CPU 保留，例如 256 個 CPU 單位 （或 1/4 vCPU)，以精細的方式擴展，將成本降至最低。但是，它們的擴展速度可能不夠快，無法滿足大量需求激增。您可以使用較大的 CPU 保留來更快地擴展，因此可以更快地符合需求激增。不過，較大的 CPU 保留成本更高。

其他應用程式

對於不水平擴展的應用程式，例如單頓工作者或資料庫伺服器，可用容量和成本代表您最重要的考量。您應該根據負載測試指出您需要提供流量以滿足服務層級目標的負載，選擇記憶體和 CPU 數量。Amazon ECS 可確保應用程式放置在具有足夠容量的主機上。

EC2 啟動類型的 Amazon ECS 任務聯網選項

在 Amazon EC2 執行個體上託管的 Amazon ECS 任務聯網行為取決於任務定義中定義的網路模式。建議使用 `awsvpc` 網路模式，除非您有使用不同網路模式的特定需要。

以下是可用的網路模式。

網路模式	EC2 上的 Linux 容器	EC2 上的 Windows 容器	描述
`awsvpc`	是	是	任務會配置自己的彈性網路介面 (ENI) 和主要私有 IPv4 地址。這會為任務提供與 Amazon EC2 執行個體相同的聯網屬性。
`bridge`	是	否	任務會使用 Docker 的 Linux 內建虛擬網路，該虛擬網路在託管任務的每個 Amazon EC2 執行個體內執行。Linux 上的內建虛擬網路會使用 `bridge` Docker 網路驅動程式。如果未在任務定義中指定網路模式，則此為 Linux 上的預設網路模式。
`host`	是	否	任務會使用主機網路，並直接將容器連接埠映射至託管任務的 Amazon EC2 執行個體 ENI，以略過 Docker 的內建虛擬網路。動態連接埠映射無法在此網路模式下使用。任務定義中使用此模式的容器必須指定特定 `hostPort` 數字。主機上的連接埠號碼無法由多個任務使用。因此，您

網路模式	EC2 上的 Linux 容器	EC2 上的 Windows 容器	描述
			無法在單一 Amazon EC2 執行個體上執行同一個任務定義的多個任務。
none	是	否	任務沒有外部網路連線。
default	否	是	任務會使用 Docker 的 Windows 內建虛擬網路，該虛擬網路在託管任務的每個 Amazon EC2 執行個體內執行。Windows 上的內建虛擬網路會使用 nat Docker 網路驅動程式。如果未在任務定義中指定網路模式，則此為 Windows 上的預設網路模式。

如需有關 Docker 的 Linux 聯網詳細資訊，請參閱 Docker 文件中的 Networking overview (聯網概觀)。

如需 Windows 上的 Docker 聯網功能詳細資訊，請參閱 Microsoft Containers on Windows Documentation (Windows 上的容器文件) 中的 Windows container networking (Windows 容器聯網功能)。

為 Amazon ECS 任務配置網路介面

awsvpc 網路模式提供的任務聯網功能可為 Amazon ECS 任務提供與 Amazon EC2 執行個體相同的聯網屬性。使用awsvpc網路模式可簡化容器聯網，因為您更能控制應用程式如何與 VPCs中的彼此和其他服務通訊。awsvpc 網路模式也可讓您在任務內更精細地使用安全群組和網路監控工具，為您的容器提供更高的安全性。您也可以使用其他 Amazon EC2 網路功能，例如 VPC 流程日誌，來監控往返任務的流量。此外，屬於同一個任務的容器可以透過 localhost 界面進行通訊。

任務彈性網路介面 (ENI) 是 Amazon ECS 的全受管功能。Amazon ECS 會建立 ENI 並將其連接到具有指定安全群組的主機 Amazon EC2 執行個體。任務在 ENI 上傳送和接收網路流量的方式，與 Amazon EC2 執行個體處理其主要網路介面的方式相同。根據預設，會對每個任務 ENI 指派一個私有 IPv4 地址。如果已對雙堆疊模式啟用 VPC，並且您使用具有 IPv6 CIDR 區塊的子網路，則任務 ENI 也會收到 IPv6 地址。每個任務只能有一個 ENI。

這些 ENIs會顯示在您帳戶的 Amazon EC2 主控台中。您的帳戶無法分離或修改 ENIs。這是為了防止意外刪除與正在執行之任務相關聯的 ENI。您可以在 Amazon ECS 主控台中或使用 DescribeTasks API 操作來檢視任務的 ENI 連接資訊。當任務停止或服務縮小規模時，任務 ENI 即予以分離和刪除。

當您需要增加 ENI 密度時，請使用 awsvpcTrunking 帳戶設定。Amazon ECS 也會為您的容器執行個體建立並連接「trunk」網路介面。幹線網路由 Amazon ECS 全受管。當您在 Amazon ECS 叢集中終止或取消註冊您的容器執行個體時，即會刪除幹線 ENI。如需awsvpcTrunking帳戶設定的詳細資訊，請參閱 必要條件。

您可以在任務定義的 networkMode 參數awsvpc中指定 。如需詳細資訊，請參閱網路模式。

然後，當您執行任務或建立服務時，請使用包含一或多個子網路的 networkConfiguration 參數，將任務放置在 中，以及要連接到 ENI 的一或多個安全群組。如需詳細資訊，請參閱網路組態。任務放置在與這些子網相同之可用區域中的有效 Amazon EC2 執行個體上，而指定之安全群組則與針對任務所佈建的 ENI 建立關聯。

Linux 考量事項

使用 Linux 作業系統時，請考量下列事項。

- 如果您在 awsvpc 模式下使用 p5.48xlarge 執行個體，則無法在執行個體上執行超過 1 個任務。
- 使用awsvpc網路模式的任務和服務需要 Amazon ECS 服務連結角色，為 Amazon ECS 提供代表您呼叫其他 AWS 服務的許可。這個角色會在您建立叢集，或在 AWS Management Console中建立或更新服務時，自動為您建立。如需詳細資訊，請參閱使用 Amazon ECS 的服務連結角色。您也可以使用下列 AWS CLI 命令建立服務連結角色：

```
aws iam create-service-linked-role --aws-service-name ecs.amazonaws.com
```

- 您的 Amazon EC2 Linux 執行個體需要容器代理程式的版本 1.15.0 或更新版本，以執行使用 awsvpc 網路模式的任務。如果您使用的是 Amazon ECS 最佳化 AMI，您的執行個體至少需要 1.15.0-4 版的 ecs-init 套件。
- 在 VPC 上同時啟用 enableDnsHostnames 和 enableDnsSupport 選項時，Amazon ECS 會使用 Amazon 提供的 (內部) DNS 主機名稱填入任務的主機名稱。如果未啟用這些選項，任務的 DNS 主機名稱會設定為隨機的主機名稱。如需 VPC DNS 設定的詳細資訊，請參閱《Amazon VPC 使用者指南》中的搭配使用 DNS 與 VPC。
- 使用 awsvpc 網路模式的每個 Amazon ECS 任務都會收到自己的彈性網路介面 (ENI)，它連接到託管該任務的 Amazon EC2 執行個體。Amazon EC2 Linux 執行個體可連接的網路介面數量有預設配額。主要網路介面視為一個配額。例如，根據預設，c5.large 執行個體最多只有三個可與其連接的 ENI。執行個體的主要網路介面視為一個配額。您可以將額外兩個 ENI 連接到執行個體。因為每項使用 awsvpc 網路模式的任務都需要 ENI，所以通常只能對此執行個體類型執行兩個這類任務。如需每個執行個體類型預設 ENI 限制的詳細資訊，請參閱《Amazon EC2 使用者指南》中的每個執行個體類型的每個網路介面 IP 地址。

- Amazon ECS 支援使用受支援的 Amazon EC2 Linux 執行個體類型啟動加強 ENI 密度的容器執行個體。當您選擇加入 `awsvpcTrunking` 帳戶設定並使用這些執行個體類型向叢集註冊 Amazon EC2 Linux 執行個體，則這些執行個體的 ENI 配額較高。使用這些具有更高配額的執行個體代表您可以在每個 Amazon EC2 Linux 執行個體中安排更多任務。若要使用具有中繼功能的增強 ENI 密度，您的 Amazon EC2 執行個體必須使用 `1.28.1` 版或更新版本的容器代理程式。如果您使用的是 Amazon ECS 最佳化 Linux AMI，您的執行個體也至少需要 `1.28.1-2` 版的 `ecs-init` 套裝服務。如需選擇使用的 `awsvpcTrunking` 帳戶設定詳細資訊，請參閱 使用帳戶設定存取 Amazon ECS功能。如需 ENI 中繼的詳細資訊，請參閱增加 Amazon ECS Linux 容器執行個體網路介面。
- 當在 Amazon EC2 Linux 執行個體上託管使用 `awsvpc` 網路模式的任務時，您的任務 ENI 不會提供公有 IP 位址。若要存取網際網路，必須在設定為使用 NAT 閘道的私有子網路中啟動任務。如需詳細資訊，請參閱 《Amazon VPC 使用者指南》中的 NAT 閘道。入站網路存取必須出自使用私有 IP 位址的 VPC，或自 VPC 透過負載平衡器路由。從公有子網路內啟動的任務無法存取網際網路。
- Amazon ECS 只會識別連接到您的 Amazon EC2 Linux 執行個體的 ENI。如果您將 ENI 手動連接到執行個體，Amazon ECS 可能會嘗試向沒有足夠網路介面卡的執行個體新增任務。這可能導致任務逾時並進入解除佈建狀態，然後進入已停止狀態。建議您不要將 ENI 手動連接到容器執行個體。
- Amazon EC2 Linux 執行個體必須使用 `ecs.capability.task-eni` 註冊，用以考量來放置具 `awsvpc` 網路模式的任務。執行 `1.15.0-4` 版或更新版本 `ecs-init` 的容器執行個體使用此屬性註冊。
- 您的帳戶無法手動分離或修改由 Amazon EC2 Linux 執行個體建立並連接的 ENI。這是為了防止意外刪除與正在執行之任務相關聯的 ENI。若要釋出任務的 ENI，請停止該任務。
- 當執行任務或建立使用 `awsvpc` 網路模式的服務時，限制只能在 `awsVpcConfiguration` 中指定 16 個子網路和 5 個安全群組。如需詳細資訊，請參閱 Amazon Elastic Container Service API 參考中的 AwsVpcConfiguration。
- 當任務在 `awsvpc` 網路模式中啟動時，Amazon ECS 容器代理程式會先為每項任務建立額外的 `pause` 容器，然後在任務定義中啟動容器。接著，它會執行 `pauseamazon-ecs-cni-plugins` CNI 外掛程式，來設定 容器的網路命名空間。然後代理會啟動任務中的其他容器，讓它們共用 `pause` 容器的網路堆疊。這表示任務中的所有容器都可由 ENI 的 IP 地址定址，而且它們彼此之間可以透過 `localhost` 界面通訊。
- 服務的任務使用 `awsvpc` 網路模式僅支援 Application Load Balancer 和 Network Load Balancer。當您為這些服務建立任何目標群組時，必須選擇 `ip` 做為目標類型。請勿選擇 `instance`。這是因為使用 `awsvpc` 網路模式的任務與 ENI 相關聯，不與 Amazon EC2 Linux 執行個體相關聯。如需詳細資訊，請參閱使用負載平衡來分發 Amazon ECS 服務流量。
- 如果您的 VPC 已更新以變更其使用的 DHCP 選項集，則無法將這些變更套用到現有的任務。啟動套用這些變更的新任務，驗證這些任務是否正確運作，接著停止現有的任務，以便安全地變更這些網路組態。

Windows 考量

當您使用 Windows 作業系統時，請考慮下列事項：

- 使用 Amazon ECS 最佳化 Windows Server 2016 AMI 的容器執行個體無法託管使用 awsvpc 網路模式的任務。如果您的叢集包含 Amazon ECS 最佳化且支援 awsvpc 網路模式的 Windows Server 2016 AMI 和 Windows AMI，使用 awsvpc 網路模式的任務不會在 Windows 2016 Server 執行個體上啟動。但會在支援 awsvpc 網路模式的執行個體上啟動。
- 您的 Amazon EC2 Windows 執行個體需要容器代理程式的版本 1.57.1 或更新版本，以使用採用 awsvpc 網路模式的 Windows 容器 CloudWatch 指標。
- 使用awsvpc網路模式的任務和服務需要 Amazon ECS 服務連結角色，為 Amazon ECS 提供代表您呼叫其他 AWS 服務的許可。這個角色會在您建立叢集，或在 AWS Management Console中建立或更新服務時，自動為您建立。如需詳細資訊，請參閱使用 Amazon ECS 的服務連結角色。您也可以使用下列 AWS CLI 命令建立服務連結角色。

```
aws iam create-service-linked-role --aws-service-name ecs.amazonaws.com
```

- 您的 Amazon EC2 Windows 執行個體需要容器代理程式的版本 1.54.0 或更新版本，以執行使用 awsvpc 網路模式的任務。啟動執行個體時，必須設定 awsvpc 網路模式所需的選項。如需詳細資訊，請參閱the section called “引導容器執行個體”。
- 在 VPC 上同時啟用 enableDnsHostnames 和 enableDnsSupport 選項時，Amazon ECS 會使用 Amazon 提供的 (內部) DNS 主機名稱填入任務的主機名稱。如果未啟用這些選項，任務的 DNS 主機名稱會是隨機的主機名稱。如需 VPC DNS 設定的詳細資訊，請參閱《Amazon VPC 使用者指南》中的搭配使用 DNS 與 VPC。
- 每項使用 awsvpc 網路模式的 Amazon ECS 任務都會收到自己的彈性網路介面 (ENI)，連接到裝載該任務的 Amazon EC2 Windows 執行個體。Amazon EC2 Windows 執行個體可連接的網路介面數量有預設配額。主要網路介面視為一個配額。例如，根據預設，c5.large 執行個體最多只有三個可與其連接的 ENI。執行個體的主要網路介面視為一個配額。您可以將額外兩個 ENI 連接到執行個體。因為每項使用 awsvpc 網路模式的任務都需要 ENI，所以通常只能對此執行個體類型執行兩個這類任務。如需每個執行個體類型預設 ENI 限制的詳細資訊，請參閱《Amazon EC2 使用者指南》中的每個執行個體類型的每個網路介面 IP 地址。
- 當在 Amazon EC2 Windows 執行個體上託管使用 awsvpc 網路模式的任務時，您的任務 ENI 不會提供公有 IP 位址。若要存取網際網路，在設定為使用 NAT 閘道的私有子網路中啟動任務。如需詳細資訊，請參閱 《Amazon VPC 使用者指南》中的 NAT 閘道。入站網路存取必須出自使用私有 IP 位址的 VPC，或自 VPC 透過負載平衡器路由。從公有子網路內啟動的任務無法存取網際網路。

- Amazon ECS 只會識別連接到您的 Amazon EC2 Windows 執行個體的 ENI。如果您將 ENI 手動連接到執行個體，Amazon ECS 可能會嘗試向沒有足夠網路介面卡的執行個體新增任務。這可能導致任務逾時並進入解除佈建狀態，然後進入已停止狀態。建議您不要將 ENI 手動連接到容器執行個體。
- Amazon EC2 Windows 執行個體必須使用 `ecs.capability.task-eni` 註冊，用以考量來放置具 `awsvpc` 網路模式的任務。
- 您無法手動修改或分離由 Amazon EC2 Windows 執行個體建立並連接的 ENI。這是為了防止您意外刪除與正在執行之任務相關聯的 ENI。若要釋出任務的 ENI，請停止該任務。
- 您在執行任務或建立使用 `awsvpc` 網路模式的服務時，您最多只能在 `awsVpcConfiguration` 指定 16 個子網和五個安全群組。如需詳細資訊，請參閱 Amazon Elastic Container Service API 參考中的 AwsVpcConfiguration。
- 當任務在 `awsvpc` 網路模式中啟動時，Amazon ECS 容器代理程式會先為每項任務建立額外的 `pause` 容器，然後在任務定義中啟動容器。接著，它會執行 `pauseamazon-ecs-cni-plugins` CNI 外掛程式，來設定 容器的網路命名空間。然後代理會啟動任務中的其他容器，讓它們共用 `pause` 容器的網路堆疊。這表示任務中的所有容器都可由 ENI 的 IP 地址定址，而且它們彼此之間可以透過 `localhost` 界面通訊。
- 服務的任務使用 `awsvpc` 網路模式僅支援 Application Load Balancer 和 Network Load Balancer。當您為這些服務建立任何目標群組時，必須選擇 `ip` 做為目標類型，而不是選擇 `instance`。這是因為使用 `awsvpc` 網路模式的任務與 ENI 相關聯，不與 Amazon EC2 Windows 執行個體相關聯。如需詳細資訊，請參閱使用負載平衡來分發 Amazon ECS 服務流量。
- 如果您的 VPC 已更新以變更其使用的 DHCP 選項集，則無法將這些變更套用到現有的任務。啟動套用這些變更的新任務，驗證這些任務是否正確運作，接著停止現有的任務，以便安全地變更這些網路組態。
- 若您在 EC2 Windows 組態中使用 `awsvpc` 網路模式，則系統不支援下列項目：
  - 雙堆疊組態
  - IPv6
  - ENI 中繼

在雙堆疊模式下使用 VPC

在雙堆疊模式下使用 VPC 時，您的任務可以透過 IPv4、IPv6 或兩者進行通訊。IPv4 和 IPv6 地址彼此獨立。因此，您必須在 VPC 中為 IPv4 和 IPv6 分別設定路由和安全性。如需如何將 VPC 設定為雙堆疊模式的詳細資訊，請參閱《Amazon VPC 使用者指南》中的遷移至 IPv6。

如果您為 VPC 設定網際網路閘道或傳出限定網際網路閘道，則可以在雙堆疊模式下使用 VPC。藉此，指派 IPv6 位址的工作就能透過網際網路閘道或僅限出口的網際網路閘道存取網際網路。NAT 閘道是選用。如需詳細資訊，請參閱 《Amazon VPC 使用者指南》 中的網際網路閘道和輸出限定網際網路閘道。

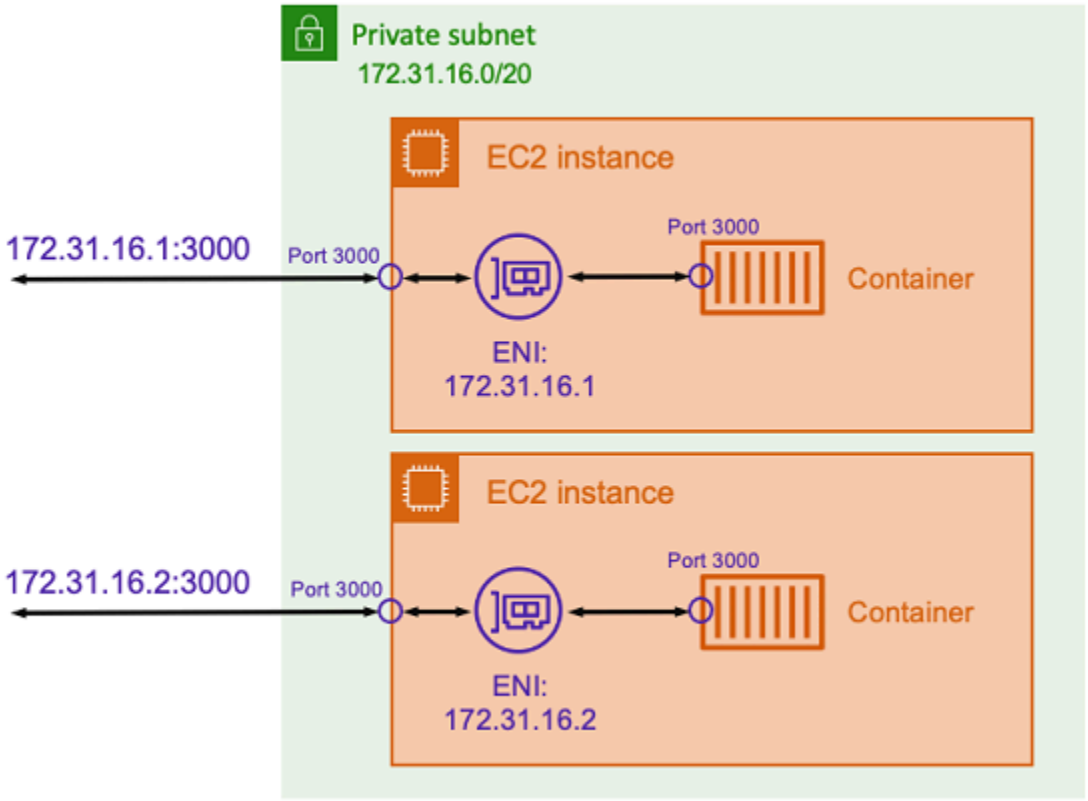
如果符合下列條件，將會為 Amazon ECS 任務指派 IPv6 地址：

- Amazon EC2 Linux 執行個體託管的任務正在使用版本 `1.45.0` 或更新版本的容器代理程式。如需如何檢查執行個體使用之代理程式版本以及視需要進行更新的相關資訊，請參閱 更新 Amazon ECS 容器代理程式。
- 此 `dualStackIPv6` 帳戶設定已啟用。如需詳細資訊，請參閱使用帳戶設定存取 Amazon ECS功能。
- 您的任務是使用 `awsvpc` 網路模式。
- 您的 VPC 和子網已針對 IPv6 進行設定。組態包含在指定子網路中建立的網路介面。如需如何將 VPC 設定為雙堆疊模式的詳細資訊，請參閱《Amazon VPC 使用者指南》中的遷移至 IPv6 和修改您子網路的公有 IPv6 定址屬性。

將 Amazon ECS 容器連接埠映射至 EC2 執行個體網路介面

僅 Amazon EC2 執行個體上託管的 Amazon ECS 任務支援 `host` 網路模式。使用 Fargate 上的 Amazon ECS 時不支援。

`host` 網路模式是 Amazon ECS 支援的最基本網路模式。使用託管模式，容器的聯網功能會直接繫結至執行容器的基礎主機。



假設您正在使用 Express 應用程式執行 Node.js 容器，該應用程式會接聽連接埠 3000 (類似於上圖中所示的連接埠)。使用 host 網路模式時，容器會使用基礎主機 Amazon EC2 執行個體的 IP 地址在連接埠 3000 上接收流量。我們不建議使用此模式。

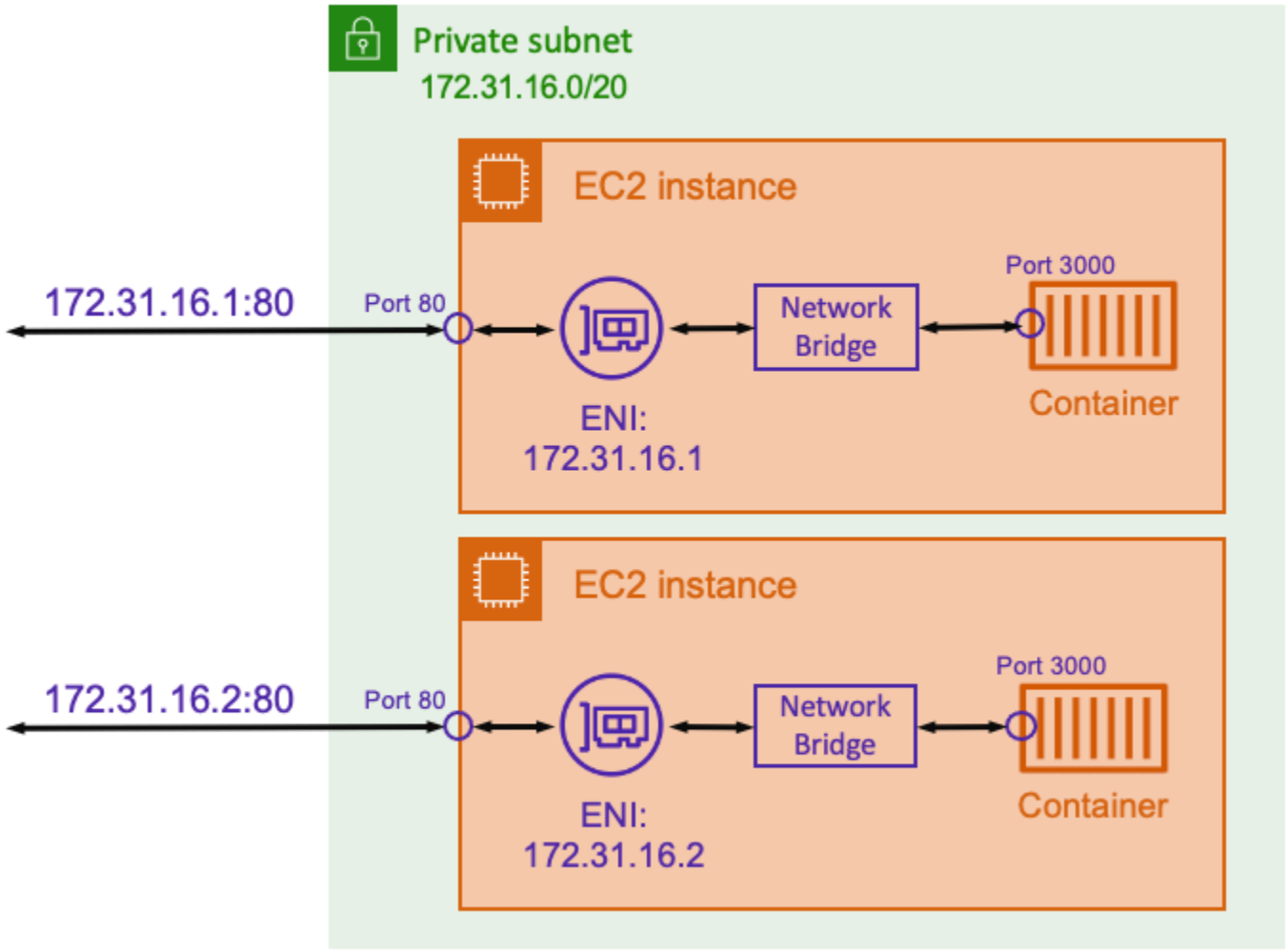
使用此網路模式存在重大的缺點。您不能在每台主機上執行多個任務的執行個體化。這是因為只有第一個任務可以繫結至 Amazon EC2 執行個體上所需的連接埠。使用 host 網路模式時，也無法重新對應容器連接埠。例如，如果應用程式需要接聽特定的連接埠號碼，則無法直接重新對應連接埠號碼。相反地，您必須透過變更應用程式組態來管理任何連接埠衝突。

使用 host 網路模式時也存在安全隱患。此模式允許容器模擬主機，並允許容器連線至主機上的私有迴路網路服務。

將 Docker 的虛擬網路用於 Amazon ECS Linux 任務

僅 Amazon EC2 執行個體上託管的 Amazon ECS 任務支援 bridge 網路模式。

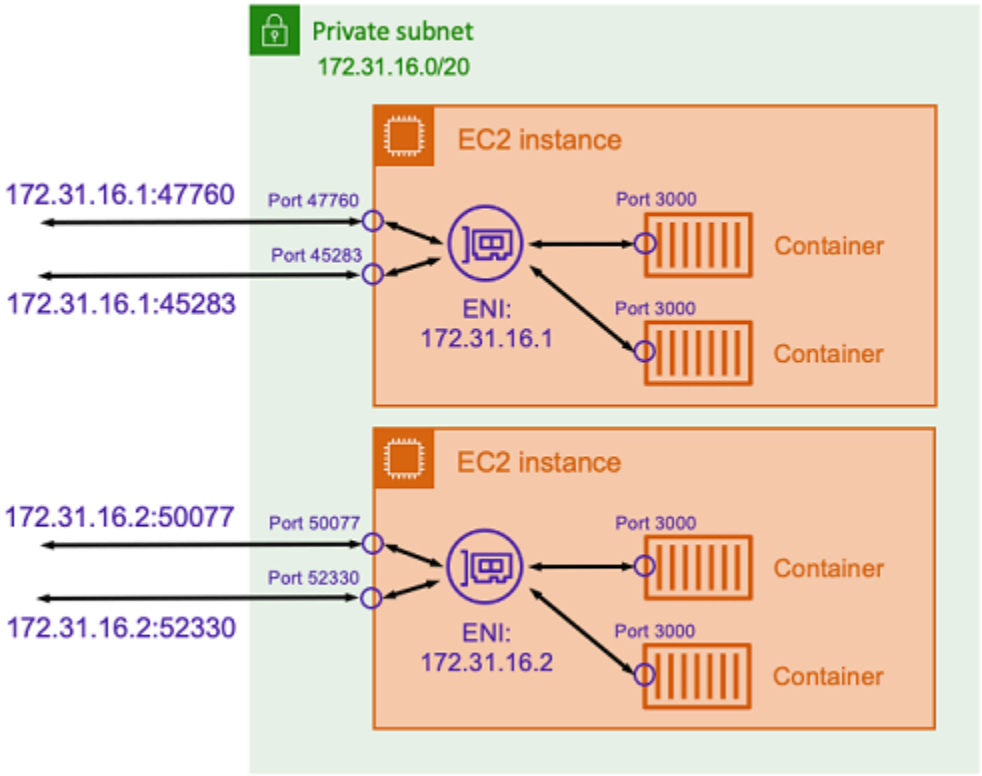
在 bridge 模式下，您正在使用虛擬網路橋接器，在主機和容器的網路之間建立一個層。如此一來，您就可以建立將主機連接埠重新對應至容器連接埠的連接埠映射。映射可以是靜態或動態模式。



透過靜態連接埠映射，您可以明確定義要對應至容器連接埠的主機連接埠。使用上述範例，主機上的 80 連接埠會對應至容器上的連接埠 3000。若要與容器化應用程式通訊，請將流量傳送至 Amazon EC2 執行個體 IP 地址的連接埠 80。從容器化應用程式的角度來看，其可以看到連接埠 3000 上的入站流量。

如果您只想變更流量連接埠，則適用於靜態連接埠映射。但是，這仍然與使用 host 網路模式具有相同的缺點。您不能在每台主機上執行多個任務的執行個體化。這是因為靜態連接埠映射僅允許單一容器對應至連接埠 80。

為了解決這個問題，可以考慮使用具有動態連接埠映射的 bridge 網路模式，如下圖所示。



透過不在連接埠映射中指定主機連接埠，您可以讓 Docker 從暫時連接埠範圍中選擇一個隨機、未使用的連接埠，並將其指派為容器的公有主機連接埠。例如，在容器上接聽連接埠 3000 的 Node.js 應用程式可能會被指派一個隨機的高編號連接埠，例如在 Amazon EC2 主機上的 47760。這樣做意味著您可以在主機上執行該容器的多個副本。此外，每個容器都可以在主機上指派自己的連接埠。容器的每個副本都會在連接埠 3000 上接收流量。不過，傳送流量至這些容器的用戶端會使用隨機指派的主機連接埠。

Amazon ECS 可協助您追蹤每個任務的隨機指派連接埠。它透過自動更新負載平衡器目標群組 AWS Cloud Map 和服務探索，以取得任務 IP 地址和連接埠的清單來執行此操作。這樣可以更輕鬆地使用透過動態連接埠的 `bridge` 模式運作的服務。

但是，使用 `bridge` 網路模式的缺點之一是難以鎖定服務對服務通訊。由於服務可能會指派給任何隨機、未使用的連接埠，因此必須在主機之間開放廣泛的連接埠範圍。然而，建立特定規則以使特定服務只能與另一個特定服務通訊並不容易。這些服務沒有可用於安全群組聯網規則的特定連接埠。

Fargate 啟動類型的 Amazon ECS 任務聯網選項

依預設，Fargate 上的每個 Amazon ECS 任務都會提供彈性網路介面 (ENI)，具有主要私有 IP 地址。使用公有子網路時，您可以選擇性地將公有 IP 位址指派給任務的 ENI。如果您的 VPC 設定為雙堆疊模式，且您使用具有 IPv6 CIDR 區塊的子網路，則任務的 ENI 也會接收 IPv6 地址。任務在指定時間

內只能有一個相關聯的 ENI。屬於同一個任務的容器也可透過 `localhost` 介面進行通訊。如需 VPCs 和子網路的詳細資訊，請參閱《Amazon VPC 使用者指南》中的 Amazon VPC 運作方式。

若要讓 Fargate 上的任務提取容器映像，任務必須有通往網際網路的路由。以下說明如何確認您的任務具有通往網際網路的路由。

- 使用公有子網路時，您可將公有 IP 地址指派給任務 ENI。
- 使用私人子網路時，子網路可以連結 NAT 閘道。
- 使用 Amazon ECR 中託管的容器映像時，您可以將 Amazon ECR 設定為使用介面 VPC 端點，並且透過任務的私有 IPv4 位址進行映像提取 。如需詳細資訊，請參閱 《Amazon Elastic Container Registry 使用者指南》中的 Amazon ECR 介面 VPC 端點 (AWS PrivateLink)。

因為每項任務都會取得自己的 ENI，所以您可以利用聯網功能，例如 VPC 流程日誌，以便您監控任務的進出流量。如需詳細資訊，請參閱「Amazon VPC 使用者指南」中的 VPC 流程日誌。

您也可以利用 AWS PrivateLink。您可以設定 VPC 介面端點，以便透過私有 IP 地址存取 Amazon ECS APIs。 AWS PrivateLink 會限制 VPC 和 Amazon ECS 之間的所有網路流量到 Amazon 網路。您不需要網際網路閘道、NAT 裝置或虛擬私有閘道。如需詳細資訊，請參閱 Amazon ECS 介面 VPC 端點 (AWS PrivateLink)。

如需如何使用 `NetworkConfiguration` 資源的範例 AWS CloudFormation，請參閱 the section called “使用獨立堆疊建立 Amazon ECS 資源”。

建立的 ENI 由 AWS Fargate全受管。此外，還有用來將許可授予 Fargate 的相關 IAM 政策。對於使用 Fargate 平台版本 `1.4.0` 或更新版本的任務，該任務會收到單一 ENI (也稱為任務 ENI)，所有網路流量會流經 VPC 中的 ENI。此流量記錄在您的 VPC 流程日誌中。對於使用 Fargate 平台版本 `1.3.0` 和舊版的任務，除了任務 ENI 以外，該任務也會收到個別 Fargate 擁有的 ENI，其用於某些網路流量，這些流量不會顯示在 VPC 流程日誌中。以下資料表描述網路流量行為，以及每個平台版本所需的 IAM 政策。

動作	使用 Linux 平台版本 **`1.3.0`** 和更早版本時的流量流程	使用 Linux 平台版本 **`1.4.0`** 時的流量流程	使用 Windows 平台版本 **`1.0.0`** 時的流量流程	IAM 許可
擷取 Amazon ECR 登入憑證	Fargate 擁有的 ENI	任務 ENI	任務 ENI	任務執行 IAM 角色

動作	使用 Linux 平台版本 **1.3.0** 和更早版本時的流量流程	使用 Linux 平台版本 **1.4.0** 時的流量流程	使用 Windows 平台版本 **1.0.0** 時的流量流程	IAM 許可
映像提取	任務 ENI	任務 ENI	任務 ENI	任務執行 IAM 角色
透過日誌驅動程式傳送日誌	任務 ENI	任務 ENI	任務 ENI	任務執行 IAM 角色
透過 FireLens for Amazon ECS 傳送日誌	任務 ENI	任務 ENI	任務 ENI	任務 IAM 角色
從 Secrets Manager 或 Systems Manager 中擷取秘密	Fargate 擁有的 ENI	任務 ENI	任務 ENI	任務執行 IAM 角色
Amazon EFS 檔案系統流量	無	任務 ENI	任務 ENI	任務 IAM 角色
應用程式流量	任務 ENI	任務 ENI	任務 ENI	任務 IAM 角色

考量事項

使用任務聯網時，請考量下列事項。

- Amazon ECS 服務連結角色需要為 Amazon ECS 提供代表您呼叫其他 AWS 服務的許可。這個角色會在您建立叢集，或在 AWS Management Console中建立或更新服務時建立。如需詳細資訊，請參閱使用 Amazon ECS 的服務連結角色。您也可以使用下列 AWS CLI 命令建立服務連結角色。

```
aws iam create-service-linked-role --aws-service-name ecs.amazonaws.com
```

- 在 VPC 上同時啟用 `enableDnsHostnames` 和 `enableDnsSupport` 選項時，Amazon ECS 會使用 Amazon 提供的 DNS 主機名稱填入任務的主機名稱。如果未啟用這些選項，任務的 DNS 主機

名稱會設定為隨機的主機名稱。如需 VPC DNS 設定的詳細資訊，請參閱《Amazon VPC 使用者指南》中的搭配使用 DNS 與 VPC。

- 您為 `awsVpcConfiguration` 最多只能指定 16 個子網和 5 個安全群組。如需詳細資訊，請參閱 Amazon Elastic Container Service API 參考 中的 AwsVpcConfiguration。
- 您的帳戶無法手動分離或修改由 Fargate 建立並連接的 ENI。這是為了防止意外刪除與正在執行之任務相關聯的 ENI。若要釋出任務的 ENI，請停止該任務。
- 如果 VPC 子網已更新以變更其使用的 DHCP 選項集，您也無法將這些變更套用到使用 VPC 的現有任務。啟動新任務，其將在測試新變更接著停止舊任務的同時，接收新的設定以順利遷移 (如果不需要轉返)。
- 在具有 IPv6 CIDR 區塊的子網中啟動的任務只會在使用 Fargate 平台 Linux `1.4.0` 版或更新版或 Windows `1.0.0` 版時接收 IPv6 位址。
- 對於使用平台 Linux `1.4.0` 版或更新版或者 Windows `1.0.0` 版的任務，任務 ENI 支援巨型訊框。網路界面皆以最大傳輸單位 (MTU) 來設定，這是適合單一框架的最大酬載大小。MTU 越大，單一框架能容納的應用程式酬載越多，可降低每個框架的額外負荷並提高效率。當任務和目的地之間的網路路徑支援巨型訊框時，支援巨型訊框會降低額外負荷。
- 使用 Fargate 啟動類型的服務與任務僅支援 Application Load Balancer 和 Network Load Balancer。不支援 Classic Load Balancer。當您建立任何目標群組時，必須選擇 `ip` 做為目標類型，而不是選擇 `instance`。如需詳細資訊，請參閱使用負載平衡來分發 Amazon ECS 服務流量。

在雙堆疊模式下使用 VPC

在雙堆疊模式中使用 VPC 時，您的任務可透過 IPv4 或 IPv6 或兩者進行通訊。IPv4 和 IPv6 地址彼此互相獨立。您必須在您的 VPC 中分別為 IPv4 和 IPv6 設定路由和安全。如需將 VPC 設定為雙堆疊模式的詳細資訊，請參閱 《Amazon VPC 使用者指南》 中的遷移至 IPv6。

如果符合下列條件，則會將 IPv6 位址指派給 Fargate 上的 Amazon ECS 任務：

- 您的 Amazon ECS `dualStackIPv6`帳戶設定已開啟 (`enabled`)，供 IAM 主體在您要啟動任務的區域中啟動任務。此設定只能使用 API 或 進行修改 AWS CLI。您可以選擇透過設定您的帳戶預設設定，為 帳戶或整個帳戶的特定 IAM 主體開啟此設定。如需詳細資訊，請參閱使用帳戶設定存取 Amazon ECS功能。
- 已對 IPv6 啟用 VPC 和子網路。如需如何將 VPC 設定為雙堆疊模式的詳細資訊，請參閱《Amazon VPC 使用者指南》中的遷移至 IPv6。
- 您的子網路已啟用自動指派 IPv6 地址。如需詳細資訊，請參閱《Amazon VPC 使用者指南》中的修改子網路的 IPv6 地址屬性。

- 任務或服務使用適用於 Linux 的 Fargate 平台版本 `1.4.0` 或更高版本。

如果您將 VPC 設定網際網路閘道或傳出限定網際網路閘道，在 Fargate 上指派 IPv6 位址的 Amazon ECS 任務就可存取網際網路。不需要 NAT 閘道。如需詳細資訊，請參閱 《Amazon VPC 使用者指南》 中的網際網路閘道和輸出限定網際網路閘道。

Amazon ECS 任務的儲存選項

Amazon ECS 根據您的需求，為您提供靈活、經濟實惠且easy-to-use的資料儲存選項。Amazon ECS 支援容器的下列資料磁碟區選項：

資料量	支援的啟動類型	支援的作業系統	儲存體持久性	使用案例
Amazon Elastic Block Store (Amazon EBS)	Fargate、Amazon EC2	Linux	連接到獨立任務時可以保留。連接到由 服務維護的任務時出現偶像。	Amazon EBS 磁碟區為資料密集型容器化工作負載提供經濟實惠、耐用、高效能的區塊儲存。常見的使用案例包括交易工作負載，例如資料庫、虛擬桌面和根磁碟區，以及輸送量密集的工作負載，例如日誌處理和 ETL 工作負載。如需詳細資訊，請參閱搭配 Amazon ECS 使用 Amazon EBS 磁碟區。
Amazon Elastic File System (Amazon EFS)	Fargate、Amazon EC2	Linux	持續	Amazon EFS 磁碟區提供簡單、可擴展且持久的

資料量	支援的啟動類型	支援的作業系統	儲存體持久性	使用案例
				共用檔案儲存，可搭配 Amazon ECS 任務使用，在您新增和移除檔案時自動成長和縮減。Amazon EFS 磁碟區支援並行，適用於水平擴展且需要儲存功能的容器化應用程式，例如低延遲、高輸送量和read-after-write一致性。常見的使用案例包括資料分析、媒體處理、內容管理和 Web 服務等工作負載。如需詳細資訊，請參閱搭配 Amazon ECS 使用 Amazon EFS 磁碟區。

資料量	支援的啟動類型	支援的作業系統	儲存體持久性	使用案例
Amazon FSx for Windows File Server	Amazon EC2	Windows	持續	FSx for Windows File Server 磁碟區提供全受管 Windows 檔案伺服器，可用來佈建需要持久性、分散式、共用和靜態檔案儲存的 Windows 任務。常見的使用案例包括 .NET 應用程式，這些應用程式可能需要本機資料夾作為持久性儲存體來儲存應用程式輸出。Amazon FSx for Windows File Server 在容器中提供本機資料夾，允許多個容器在由 SMB 共用支援的相同檔案系統上讀寫。如需詳細資訊，請參閱搭配 Amazon ECS 使用 FSx for Windows File Server 磁碟區。

資料量	支援的啟動類型	支援的作業系統	儲存體持久性	使用案例
Docker 磁碟區	Amazon EC2	Windows，Linux	持續	Docker 磁碟區是 Docker 容器執行時間的一項功能，透過從主機的檔案系統掛載目錄，允許容器保留資料。Docker 磁碟區驅動程式（也稱為外掛程式）用於整合容器磁碟區與外部儲存系統。Docker 磁碟區可由第三方驅動程式或內建`local`驅動程式管理。Docker 磁碟區的常見使用案例包括提供持久性資料磁碟區，或在同一容器執行個體上不同容器的不同位置共用磁碟區。如需詳細資訊，請參閱搭配 Amazon ECS 使用 Docker 磁碟區。

資料量	支援的啟動類型	支援的作業系統	儲存體持久性	使用案例
綁定掛載	Fargate、Amazon EC2	Windows，Linux	偶像	繫結掛載由主機上的檔案或目錄組成，例如 Amazon EC2 執行個體 AWS Fargate或掛載至容器。繫結掛載的常見使用案例包括與相同任務中的其他容器共用來源容器的磁碟區，或在一或多個容器中掛載主機磁碟區或空磁碟區。如需詳細資訊，請參閱搭配 Amazon ECS 使用繫結掛載。

搭配 Amazon ECS 使用 Amazon EBS 磁碟區

Amazon Elastic Block Store (Amazon EBS) 磁碟區為資料密集型工作負載提供高可用性、經濟實惠、耐用、高效能的區塊儲存。Amazon EBS 磁碟區可與 Amazon ECS 任務搭配使用，以用於高輸送量和交易密集型應用程式。

在獨立任務啟動期間，您可以提供組態，用於將一個 EBS 磁碟區連接至任務。在服務建立或更新期間，您可以提供組態，用於將每個任務的一個 EBS 磁碟區連接至 Amazon ECS 服務管理的每個任務。

透過在啟動時間提供磁碟區組態，而不是在任務定義中，您可以建立任務定義，而這些定義不受特定資料磁碟區類型或特定 EBS 磁碟區設定的限制。然後，您可以在不同的執行時間環境中重複使用任務定義。例如，您可以在部署期間為生產工作負載提供比生產前環境更多的輸送量。

連接至 Amazon ECS 任務的 Amazon EBS 磁碟區是由 Amazon ECS 代表您管理。您可以使用 AWS Key Management Service (AWS KMS) 金鑰來加密磁碟區，以保護您的資料。您可以設定新的空白磁碟區進行連接，也可以使用快照從現有磁碟區載入資料。

若要監控磁碟區的效能，您也可以使用 Amazon CloudWatch 指標。如需 Amazon EBS 磁碟區 Amazon ECS 指標的詳細資訊，請參閱 Amazon ECS CloudWatch 指標 和 Amazon ECS Container Insights 指標。

AWS 區域 支援 Amazon ECS 的所有商業和中國都支援將 Amazon EBS 磁碟區連接到任務。

如需 Amazon EBS 磁碟區的詳細資訊，請參閱《Amazon EBS 使用者指南》中的 Amazon EBS 磁碟區。

支援的作業系統和啟動類型

下表提供支援的作業系統和啟動類型組態。

啟動類型	Linux	Windows
Fargate	平台版本 `1.4.0`或更新版本 (Linux) 支援 Amazon EBS 磁碟區。如需詳細資訊，請參閱適用於 Amazon ECS 的 Fargate 平台版本。	不支援
EC2	使用 Amazon ECS 最佳化 Amazon Machine Image (AMIs) 在 Nitro型 Linux 執行個體上託管的任務。如需執行個體類型的詳細資訊，請參閱《Amazon EC2 使用者指南》中的執行個體類型。 ECS 最佳化 AMI `20231219`或更新版本支援 Amazon EBS 磁碟區。如需詳細資訊，請參閱擷取 Amazon ECS 最佳化 AMI 中繼資料。	使用 Amazon ECS 最佳化 Amazon Machine Image (AMIs) 在 Nitro型 Linux 執行個體上託管的任務。如需執行個體類型的詳細資訊，請參閱《Amazon EC2 使用者指南》中的執行個體類型。 ECS 最佳化 AMI `20241017`或更新版本支援 Amazon EBS 磁碟區。如需詳細資訊，請參閱擷取 Amazon ECS 最佳化 Windows AMI 中繼資料。

考量事項

使用 Amazon EBS 磁碟區時，請考慮下列事項：

- 您無法在`use1-az3`可用區域中設定 Amazon EBS 磁碟區以連接至 Fargate 啟動類型 Amazon ECS 任務。
- Fargate 上託管的任務不支援磁性 (`standard`) Amazon EBS 磁碟區類型。如需 Amazon EBS 磁碟區類型的詳細資訊，請參閱《Amazon Amazon EC2 EBS 磁碟區。
- 建立服務或獨立任務，在部署時設定磁碟區時，需要 Amazon ECS 基礎設施 IAM 角色。您可以將 AWS 受管 `AmazonECSInfrastructureRolePolicyForVolumes` IAM 政策連接至角色，也可以使用 受管政策做為建立和連接您自己的政策的指南，其許可可滿足您的特定需求。如需詳細資訊，請參閱Amazon ECS 基礎設施 IAM 角色。
- 您最多可以將一個 Amazon EBS 磁碟區連接至每個 Amazon ECS 任務，而且它必須是新的磁碟區。您無法將現有的 Amazon EBS 磁碟區連接至任務。不過，您可以使用現有磁碟區的快照，在部署時設定新的 Amazon EBS 磁碟區。
- 您只能針對使用滾動更新部署類型和複本排程策略的服務，在部署時設定 Amazon EBS 磁碟區。
- 若要讓任務中的容器寫入掛載的 Amazon EBS 磁碟區，您必須以根使用者身分執行容器。
- Amazon ECS 會自動將預留標籤`AmazonECSCreated`和 `AmazonECSManaged` 新增至連接的磁碟區。如果您從磁碟區中移除這些標籤，Amazon ECS 將無法代表您管理磁碟區。如需標記 Amazon EBS 磁碟區的詳細資訊，請參閱標記 Amazon EBS 磁碟區。如需標記 Amazon ECS 資源的詳細資訊，請參閱標記 Amazon ECS 資源。
- 不支援從包含分割區的 Amazon EBS 磁碟區的快照佈建磁碟區。
- 連接到由 服務管理之任務的磁碟區不會保留，且一律會在任務終止時刪除。
- 您無法設定 Amazon EBS 磁碟區以連接至執行中的 Amazon ECS 任務 AWS Outposts。

延遲磁碟區組態，以在 Amazon ECS 任務定義中啟動時間

若要設定 Amazon EBS 磁碟區以連接至您的任務，您必須在任務定義中指定掛載點組態，並命名磁碟區。您也必須`configuredAtLaunch`將 設定為 ，`true`因為 Amazon EBS 磁碟區無法在任務定義中設定為連接。反之，Amazon EBS 磁碟區會設定為在部署期間連接。

若要使用 AWS Command Line Interface (AWS CLI) 註冊任務定義，請將範本儲存為 JSON 檔案，然後傳遞 檔案做為`register-task-definition`命令的輸入。

若要使用 建立和註冊任務定義 AWS Management Console，請參閱 使用主控台建立 Amazon ECS 任務定義。

下列任務定義顯示任務定義中 mountPoints和 volumes 物件的語法。如需任務定義參數的詳細資訊，請參閱Amazon ECS 任務定義參數。若要使用此範例，請以您自己的資訊取代 *user input placeholders*。

Linux

```
{
    "family": "mytaskdef",
    "containerDefinitions": [
        {
            "name": "nginx",
            "image": "public.ecr.aws/nginx/nginx:latest",
            "networkMode": "awsvpc",
           "portMappings": [
                {
                    "name": "nginx-80-tcp",
                    "containerPort": 80,
                    "hostPort": 80,
                    "protocol": "tcp",
                    "appProtocol": "http"
                }
            ],
            "mountPoints": [
                {
                    "sourceVolume": "myEBSVolume",
                    "containerPath": "/mount/ebs",
                    "readOnly": true
                }
            ]
        }
    ],
    "volumes": [
        {
            "name": "myEBSVolume",
            "configuredAtLaunch": true
        }
    ],
    "requiresCompatibilities": [
        "FARGATE", "EC2"
    ],
    "cpu": "1024",
    "memory": "3072",
    "networkMode": "awsvpc"
```

```
}
```

Windows

```
{
    "family": "mytaskdef",
     "memory": "4096",
     "cpu": "2048",
    "family": "windows-simple-iis-2019-core",
    "executionRoleArn": "arn:aws:iam::012345678910:role/ecsTaskExecutionRole",
    "runtimePlatform": {"operatingSystemFamily": "WINDOWS_SERVER_2019_CORE"},
    "requiresCompatibilities": ["EC2"]
    "containerDefinitions": [
        {
             "command": ["New-Item -Path C:\\inetpub\\wwwroot\\index.html -Type file
 -Value '<html> <head> <title>Amazon ECS Sample App</title> <style>body {margin-top:
 40px; background-color: #333;} </style> </head><body> <div style=color:white;text-
align:center> <h1>Amazon ECS Sample App</h1> <h2>Congratulations!</h2> <p>Your
 application is now running on a container in Amazon ECS.</p>'; C:\\ServiceMonitor.exe
 w3svc"],
            "entryPoint": [
                "powershell",
                "-Command"
            ],
            "essential": true,
            "cpu": 2048,
            "memory": 4096,
            "image": "mcr.microsoft.com/windows/servercore/iis:windowsservercore-
ltsc2019",
            "name": "sample_windows_app",
            "portMappings": [
                {
                    "hostPort": 443,
                    "containerPort": 80,
                    "protocol": "tcp"
                }
            ],
            "mountPoints": [
                {
                    "sourceVolume": "myEBSVolume",
                    "containerPath": "drive:\ebs",
                    "readOnly": true
                }
```

```
            ]
        }
    ],
    "volumes": [
        {
            "name": "myEBSVolume",
            "configuredAtLaunch": true
        }
    ],
    "requiresCompatibilities": [
        "FARGATE", "EC2"
    ],
    "cpu": "1024",
    "memory": "3072",
    "networkMode": "awsvpc"
}
```

`mountPoints`

類型：物件陣列

必要：否

容器中資料磁碟區的掛載點。此參數會映射至 建立容器 Docker API `Volumes`中的 ，以及 Docker 執行`--volume`的選項。

Windows 容器可在 `$env:ProgramData` 所在的相同磁碟上掛載整個目錄。Windows 容器無法在不同的磁碟機上掛載目錄，而且掛載點無法跨磁碟機使用。您必須指定掛載點，將 Amazon EBS 磁碟區直接連接至 Amazon ECS 任務。

`sourceVolume`

類型：字串

必要：是 (當使用 `mountPoints` 時)

要掛載的磁碟區名稱。

`containerPath`

類型：字串

必要：是 (當使用 `mountPoints` 時)

要掛載磁碟區的容器中的路徑。

`readOnly`

類型：布林值

必要：否

如果此數值為 `true`，容器擁有磁碟區的唯讀存取權。如果此值為 `false`，則容器可寫入磁碟區。預設值為 `false`。

對於在執行 Windows 作業系統的 EC2 執行個體上執行的任務，請將 值保留為預設值 `false`。

`name`

類型：字串

必要：否

磁碟區名稱。最多允許 255 個字母 （大寫和小寫）、數字、連字號 (-) 和底線 (_)。此名稱會在容器定義`mountPoints`物件的 `sourceVolume` 參數中參考。

`configuredAtLaunch`

類型：布林值

必要：是，當您想要使用 將 EBS 磁碟區直接連接至任務時。

指定磁碟區是否可在啟動時設定。設定為 時`true`，您可以在執行獨立任務時，或建立或更新服務時設定磁碟區。設為 時`false`，您將無法在任務定義中提供另一個磁碟區組態。必須提供此參數，並將其設定為 `true`，以設定要連接到任務的 Amazon EBS 磁碟區。

加密存放在 Amazon ECS 的 Amazon EBS 磁碟區中的資料

您可以使用 AWS Key Management Service (AWS KMS) 來建立和管理加密金鑰，以保護您的資料。Amazon EBS 磁碟區會使用 進行靜態加密 AWS KMS keys。以下類型的資料會加密：

- 磁碟區上靜態存放的資料
- 磁碟輸入/輸出
- 從磁碟區建立的快照
- 從快照建立的新磁碟區

您可以預設設定 Amazon EBS 加密，以便使用您為帳戶設定的 KMS 金鑰，加密所有建立並連接至任務的新磁碟區。如需預設 Amazon EBS 加密和加密的詳細資訊，請參閱《Amazon Amazon EC2 EBS 加密。

連接至任務的 Amazon EBS 磁碟區可以使用預設 AWS 受管金鑰 的別名 `alias/aws/ebs`或對稱客戶受管金鑰進行加密。 AWS 帳戶 每個 的預設值 AWS 受管金鑰 都是唯一的 AWS 區域 ，且會自動建立。若要建立對稱客戶受管金鑰，請遵循 AWS KMS 開發人員指南中的建立對稱加密 KMS 金鑰中的步驟。

客戶受管 KMS 金鑰政策

若要使用客戶受管金鑰加密連接至任務的 EBS 磁碟區，您必須設定 KMS 金鑰政策，以確保您用於磁碟區組態的 IAM 角色具有使用金鑰的必要許可。金鑰政策必須包含 `kms:CreateGrant`和 `kms:GenerateDataKey*`許可。`kms:ReEncryptTo` 和 `kms:ReEncryptFrom`許可對於加密使用快照建立的磁碟區是必要的。如果您只想要設定和加密新的空白磁碟區以供連接，則可以排除 `kms:ReEncryptTo`和 `kms:ReEncryptFrom`許可。

下列 JSON 程式碼片段顯示您可以連接到 KMS 金鑰政策的金鑰政策陳述式。使用這些陳述式將可讓 Amazon ECS 存取 ，以使用 金鑰來加密 EBS 磁碟區。若要使用範例政策陳述式，請將 取代*user input placeholders*為您自己的資訊。一如往常，只設定您需要的許可。

```
{
      "Effect": "Allow",
      "Principal": { "AWS": "arn:aws:iam::111122223333:role/ecsInfrastructureRole" },
      "Action": "kms:DescribeKey",
      "Resource":"*"
    },
    {
      "Effect": "Allow",
      "Principal": { "AWS": "arn:aws:iam::111122223333:role/ecsInfrastructureRole" },
      "Action": [
      "kms:GenerateDataKey*",
      "kms:ReEncryptTo",
      "kms:ReEncryptFrom"
      ],
      "Resource":"*",
      "Condition": {
        "StringEquals": {
          "kms:CallerAccount": "aws_account_id",
          "kms:ViaService": "ec2.region.amazonaws.com"
        },
```

```
        "ForAnyValue:StringEquals": {
          "kms:EncryptionContextKeys": "aws:ebs:id"
        }
      }
    },
    {
      "Effect": "Allow",
      "Principal": { "AWS": "arn:aws:iam::111122223333:role/ecsInfrastructureRole" },
      "Action": "kms:CreateGrant",
      "Resource":"*",
      "Condition": {
        "StringEquals": {
          "kms:CallerAccount": "aws_account_id",
          "kms:ViaService": "ec2.region.amazonaws.com"
        },
        "ForAnyValue:StringEquals": {
          "kms:EncryptionContextKeys": "aws:ebs:id"
        },
        "Bool": {
          "kms:GrantIsForAWSResource": true
        }
      }
    }
```

如需有關金鑰政策和許可的詳細資訊，請參閱《 AWS KMS 開發人員指南》中的 AWS KMS和 AWS KMS 許可。如需針對與金鑰許可相關的 EBS 磁碟區連接問題進行故障診斷，請參閱 對 Amazon ECS 任務的 Amazon EBS 磁碟區附件進行故障診斷 。

在 Amazon ECS 部署指定 Amazon EBS 磁碟區組態

將 `configuredAtLaunch` 參數設定為 來註冊任務定義後`true`，您可以在執行獨立任務時，或在建立或更新服務時，於部署時設定 Amazon EBS 磁碟區。

若要轉換磁碟區，您可以使用 Amazon ECS APIs，或傳遞 JSON 檔案做為下列 AWS CLI 命令的輸入：

- `run-task` 執行獨立 ECS 任務。
- `start-task` 在特定容器執行個體中執行獨立 ECS 任務。此命令不適用於 Fargate 啟動類型任務。
- `create-service` 建立新的 ECS 服務。
- `update-service` 以更新現有的服務。

> **Note**
>
> 若要讓任務中的容器寫入掛載的 Amazon EBS 磁碟區，您必須以根使用者身分執行容器。

您也可以使用 設定 Amazon EBS 磁碟區 AWS Management Console。如需詳細資訊，請參閱以 Amazon ECS 任務執行應用程式、使用 主控台建立 Amazon ECS 服務及使用主控台更新 Amazon ECS 服務。

下列 JSON 程式碼片段顯示可在部署時設定的 Amazon EBS 磁碟區的所有參數。若要將這些參數用於磁碟區組態，請將 取代*user input placeholders*為您自己的資訊。如需這些參數的詳細資訊，請參閱磁碟區組態。

```
"volumeConfigurations": [
        {
            "name": "ebs-volume",
            "managedEBSVolume": {
                "encrypted": true,
                "kmsKeyId": "arn:aws:kms:us-
east-1:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
                "volumeType": "gp3",
                "sizeInGiB": 10,
                "snapshotId": "snap-12345",
                "iops": 3000,
                "throughput": 125,
                "tagSpecifications": [
                    {
                        "resourceType": "volume",
                        "tags": [
                            {
                                "key": "key1",
                                "value": "value1"
                            }
                        ],
                        "propagateTags": "NONE"
                    }
                ],
                "roleArn": "arn:aws:iam:1111222333:role/ecsInfrastructureRole",
                 "terminationPolicy": {
                    "deleteOnTermination": true//can't be configured for service-
managed tasks, always true
                },
```

```
                "filesystemType": "ext4"
            }
        }
    ]
```

> **Important**
>
> 請確定volumeName您在組態中指定的 與volumeName您在任務定義中指定的 相同。

如需檢查磁碟區連接狀態的資訊，請參閱 對 Amazon ECS 任務的 Amazon EBS 磁碟區附件進行故障診斷 。如需 EBS 磁碟區連接所需的 Amazon ECS 基礎設施 AWS Identity and Access Management (IAM) 角色相關資訊，請參閱 Amazon ECS 基礎設施 IAM 角色。

以下是顯示 Amazon EBS 磁碟區的組態的 JSON 程式碼片段範例。這些範例可以透過將程式碼片段儲存在 JSON 檔案中，並將檔案傳遞為 AWS CLI 命令的參數（使用 `--cli-input-json file://filename` 參數）來使用。以您自己的資訊取代 *user input placeholders*。

設定獨立任務的磁碟區

以下程式碼片段顯示設定 Amazon EBS 磁碟區以連接至獨立任務的語法。下列 JSON 程式碼片段顯示設定 volumeType、encrypted、 sizeInGiB和 kmsKeyId設定的語法。JSON 檔案中指定的組態用於建立 EBS 磁碟區並將其連接至獨立任務。

```
{
   "cluster": "mycluster",
   "taskDefinition": "mytaskdef",
   "volumeConfigurations": [
        {
            "name": "datadir",
            "managedEBSVolume": {
                "volumeType": "gp3",
                "sizeInGiB": 100,
                "roleArn":"arn:aws:iam:1111222333:role/ecsInfrastructureRole",
                "encrypted": true,
                "kmsKeyId":
 "arn:aws:kms:region:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
            }
        }
   ]
}
```

在建立服務時設定磁碟區

以下程式碼片段顯示設定 Amazon EBS 磁碟區以連接至服務所管理之任務的語法。磁碟區是使用 從快照取得`snapshotId`。JSON 檔案中指定的組態用於建立 EBS 磁碟區，並將其連接至服務管理的每個任務。

```
{
   "cluster": "mycluster",
   "taskDefinition": "mytaskdef",
   "serviceName": "mysvc",
   "desiredCount": 2,
   "volumeConfigurations": [
        {
            "name": "myEbsVolume",
            "managedEBSVolume": {
              "roleArn":"arn:aws:iam:1111222333:role/ecsInfrastructureRole",
              "snapshotId": "snap-12345"
            }
        }
   ]
}
```

在服務更新時設定磁碟區

下列 JSON 程式碼片段顯示更新先前未設定 Amazon EBS 磁碟區以連接至任務之服務的語法。您必須提供任務定義修訂的 ARN，並將 `configuredAtLaunch`設為 `true`。下列 JSON 程式碼片段顯示設定 `volumeType`、`throughput`、 `sizeInGiB`和 `iops`和 `filesystemType`設定的語法。此組態用於建立 EBS 磁碟區，並將其連接至服務管理的每個任務。

```
{
   "cluster": "mycluster",
   "taskDefinition": "mytaskdef",
   "serviceName": "mysvc",
   "desiredCount": 2,
  "volumeConfigurations": [
        {
            "name": "myEbsVolume",
            "managedEBSVolume": {
              "roleArn":"arn:aws:iam:1111222333:role/ecsInfrastructureRole",
               "volumeType": "gp3",
                "sizeInGiB": 100,
                 "iops": 3000,
```

```
                "throughput": 125,
                "filesystemType": "ext4"
            }
        }
    ]
}
```

設定服務不再使用 Amazon EBS 磁碟區

下列 JSON 程式碼片段顯示更新服務以不再使用 Amazon EBS 磁碟區的語法。您必須提供任務定義的 ARN，並將 configuredAtLaunch設為 false，或沒有 configuredAtLaunch 參數的任務定義。您還必須提供空volumeConfigurations物件。

```
{
   "cluster": "mycluster",
   "taskDefinition": "mytaskdef",
   "serviceName": "mysvc",
   "desiredCount": 2,
   "volumeConfigurations": []
}
```

Amazon EBS 磁碟區的終止政策

當 Amazon ECS 任務終止時，Amazon ECS 會使用 deleteOnTermination值來判斷是否應刪除與終止任務相關聯的 Amazon EBS 磁碟區。根據預設，在任務終止時，會刪除連接至任務的 EBS 磁碟區。對於獨立任務，您可以變更此設定，以在任務終止時保留磁碟區。

> **Note**
>
> 連接到由 服務管理之任務的磁碟區不會保留，且一律會在任務終止時刪除。

標記 Amazon EBS 磁碟區

您可以使用 tagSpecifications 物件來標記 Amazon EBS 磁碟區。您可以使用 物件提供自己的標籤，並設定任務定義或服務中的標籤傳播，取決於磁碟區是否連接到獨立任務或服務中的任務。可連接至磁碟區的標籤數量上限為 50。

> **Important**
>
> Amazon ECS 會自動將 AmazonECSCreated和AmazonECSManaged預留標籤連接到 Amazon EBS 磁碟區。這表示您可以控制最多 48 個額外標籤連接至磁碟區。這些額外的標籤可以是使用者定義、ECS 管理或傳播的標籤。

如果您想要將 Amazon ECS 受管標籤新增至磁碟區，則必須在 UpdateService、CreateServiceRunTask或 StartTask呼叫true中enableECSManagedTags將 設定為 。如果您開啟 Amazon ECS 受管標籤，Amazon ECS 會使用叢集和服務資訊 (aws:ecs:*clusterName* 和) 自動標記磁碟區aws:ecs:*serviceName*。如需標記 Amazon ECS 資源的詳細資訊，請參閱標記 Amazon ECS 資源。

下列 JSON 程式碼片段顯示使用使用者定義標籤，標記連接到服務中每個任務的每個 Amazon EBS 磁碟區的語法。若要使用此範例建立服務，請將 取代*user input placeholders*為您自己的資訊。

```
{
   "cluster": "mycluster",
   "taskDefinition": "mytaskdef",
   "serviceName": "mysvc",
   "desiredCount": 2,
   "enableECSManagedTags": true,
   "volumeConfigurations": [
        {
            "name": "datadir",
            "managedEBSVolume": {
                "volumeType": "gp3",
                "sizeInGiB": 100,
                 "tagSpecifications": [
                    {
                        "resourceType": "volume",
                        "tags": [
                            {
                                "key": "key1",
                                "value": "value1"
                            }
                        ],
                        "propagateTags": "NONE"
                    }
                ]
                "roleArn":"arn:aws:iam:1111222333:role/ecsInfrastructureRole",
```

```
                "encrypted": true,
                "kmsKeyId":
 "arn:aws:kms:region:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
            }
        }
    ]
}
```

> **Important**
>
> 您必須指定volume資源類型來標記 Amazon EBS 磁碟區。

Fargate 隨需任務的 Amazon EBS 磁碟區效能

Fargate 隨需任務可用的基準 Amazon EBS 磁碟區 IOPS 和輸送量取決於您請求任務的總 CPU 單位。如果您為您的 Fargate 任務請求 0.25、0.5 或 1 個虛擬 CPU 單元 (vCPU)，我們建議您設定一般用途 SSD 磁碟區 (gp2 或 gp3) 或硬碟 (HDD) 磁碟區 (st1 或 sc1)。如果您為 Fargate 任務請求多個 vCPU，則下列基準效能限制適用於連接至任務的 Amazon EBS 磁碟區。您可能暫時獲得比下列限制更高的 EBS 效能。不過，我們建議您根據這些限制來規劃工作負載。

請求vCPUs 為單位）	基準 Amazon EBS IOPS(16 KiB I/O)	基準 Amazon EBS 輸送量 (MiBps、128 KiB I/O)	基準頻寬（以 Mbps 為單位）
2	3,000	75	360
4	5,000	120	1,150
8	10,000	250	2,300
16	15,000	500	4，500

> **Note**
>
> 當您設定 Amazon EBS 磁碟區以連接至 Fargate 任務時，會在任務的暫時性儲存體和連接的磁碟區之間共用 Fargate 任務的 Amazon EBS 效能限制。

EC2 任務的 Amazon EBS 磁碟區效能

Amazon EBS 提供下列各有不同效能特性及價格的磁碟區類型，可讓您量身打造符合您應用程式需求的儲存效能和成本。如需效能的相關資訊，包括每個磁碟區的 IOPS 和每個磁碟區的輸送量，請參閱《Amazon Elastic Block Store 使用者指南》中的 Amazon EBS 磁碟區類型。

對 Amazon ECS 任務的 Amazon EBS 磁碟區附件進行故障診斷

您可能需要對 Amazon EBS 磁碟區連接至 Amazon ECS 任務進行故障診斷或驗證。

檢查磁碟區連接狀態

您可以使用 AWS Management Console 檢視 Amazon EBS 磁碟區連接至 Amazon ECS 任務的狀態。如果任務啟動且附件失敗，您也會看到可用來進行故障診斷的狀態原因。建立的磁碟區將被刪除，任務也會停止。如需狀態原因的詳細資訊，請參閱 Amazon EBS 磁碟區連接至 Amazon ECS 任務的狀態原因。

使用主控台檢視磁碟區的連接狀態和狀態原因

1. 開啟主控台，網址為 https://console.aws.amazon.com/ecs/v2。
2. 在叢集頁面上，選擇任務正在執行的叢集。叢集的詳細資訊頁面隨即出現。
3. 在叢集的詳細資訊頁面上，選擇任務索引標籤。
4. 選擇您要檢視磁碟區連接狀態的任務。您可能需要使用篩選所需的狀態，如果要檢查的任務已停止，請選擇已停止。
5. 在任務的詳細資訊頁面上，選擇磁碟區索引標籤。您將能夠在附件狀態下看到 Amazon EBS 磁碟區的附件狀態。如果磁碟區無法連接至任務，您可以在附件狀態下選擇狀態，以顯示失敗的原因。

您也可以使用 DescribeTasks API 檢視任務的磁碟區連接狀態和相關狀態原因。

服務和任務失敗

您可能會遇到不特定於 Amazon EBS 磁碟區的服務或任務失敗，而這可能會影響磁碟區連接。如需詳細資訊，請參閱

- 服務事件訊息
- 已停止的任務錯誤碼
- API 失敗原因

Amazon EBS 磁碟區連接至 Amazon ECS 任務的狀態原因

使用下列參考來修正您在設定 Amazon EBS 磁碟區以連接至 Amazon ECS 任務 AWS Management Console 時，可能遇到的狀態原因形式問題。如需在主控台中尋找這些狀態原因的詳細資訊，請參閱檢查磁碟區連接狀態。

ECS 無法擔任設定的 ECS 基礎設施角色 'arn：aws：iam：：*111122223333*：role/*ecsInfrastructureRole*'。請確認傳遞的角色與 Amazon ECS 具有適當的信任關係

此狀態原因會顯示在下列案例中。

- 您提供未連接必要信任政策的 IAM 角色。如果角色沒有必要的信任政策，Amazon ECS 無法存取您提供的 Amazon ECS 基礎設施 IAM 角色。任務可能會卡在 DEPROVISIONING 狀態。如需必要信任政策的詳細資訊，請參閱Amazon ECS 基礎設施 IAM 角色。
- 您的 IAM 使用者沒有將 Amazon ECS 基礎設施角色傳遞至 Amazon ECS 的許可。任務可能會卡在 DEPROVISIONING 狀態。若要避免此問題，您可以將 PassRole 許可連接至您的使用者。如需詳細資訊，請參閱Amazon ECS 基礎設施 IAM 角色。
- 您的 IAM 角色沒有 Amazon EBS 磁碟區連接的必要許可。任務可能會卡在 DEPROVISIONING 狀態。如需將 Amazon EBS 磁碟區連接至任務所需的特定許可的詳細資訊，請參閱 Amazon ECS 基礎設施 IAM 角色。

> **Note**
>
> 您也可能因為角色傳播延遲而看到此錯誤訊息。如果在等待幾分鐘後重試使用該角色，無法解決問題，則您可能已為該角色設定錯誤信任政策。

ECS 無法設定 EBS 磁碟區。遇到 IdempotentParameterMismatch"； "您提供的用戶端字符與已刪除的資源相關聯。請使用不同的用戶端字符。」

下列 AWS KMS 關鍵案例可能會導致IdempotentParameterMismatch訊息出現：

- 您可以指定無效的 KMS 金鑰 ARN、ID 或別名。在此案例中，任務可能看起來已成功啟動，但任務最終會失敗，因為 會以非同步方式 AWS 驗證 KMS 金鑰。如需詳細資訊，請參閱《Amazon EC2 使用者指南》中的 Amazon EBS 加密。 Amazon EC2
- 您提供的客戶受管金鑰缺少允許 Amazon ECS 基礎設施 IAM 角色使用金鑰進行加密的許可。若要避免金鑰政策許可問題，請參閱 Amazon EBS 磁碟區的資料加密中的 AWS KMS 金鑰政策範例。

您可以設定 Amazon EventBridge 將 Amazon EBS 磁碟區事件和 Amazon ECS 任務狀態變更事件傳送至目標，例如 Amazon CloudWatch 群組。然後，您可以使用這些事件來識別影響磁碟區連接的特定客戶受管金鑰相關問題。如需詳細資訊，請參閱

- 如何建立 CloudWatch 日誌群組，以做為 re：Post 上 EventBridge 規則的目標？。 AWS
- 任務狀態變更事件。
- Amazon EBS 使用者指南中的 Amazon EventBridge 事件。

設定 EBS 磁碟區連接至任務時 ECS 逾時。

下列檔案系統格式案例會產生此訊息。

- 您在組態期間指定的檔案系統格式與任務的作業系統不相容。
- 您設定要從快照建立的 Amazon EBS 磁碟區，且快照的檔案系統格式與任務的作業系統不相容。對於從快照建立的磁碟區，您必須指定建立快照時磁碟區所使用的相同檔案系統類型。

您可以利用 Amazon ECS 容器代理程式日誌，針對 Amazon EC2 啟動類型任務疑難排解此訊息。如需詳細資訊，請參閱 Amazon ECS 日誌檔案位置和 Amazon ECS 日誌收集器。

搭配 Amazon ECS 使用 Amazon EFS 磁碟區

Amazon Elastic File System (Amazon EFS) 提供簡單且可擴展的檔案儲存體，可與 Amazon ECS 任務搭配使用。利用 Amazon EFS，儲存容量即可有彈性。儲存容量會隨著您新增和移除檔案時自動擴展和縮減。您的應用程式可在需要時具備所需的儲存容量。

您可以搭配使用 Amazon EFS 檔案系統與 Amazon ECS，來匯出整個容器執行個體之機群的檔案系統資料。如此一來，無論您的任務位於哪個執行個體，都可存取相同的持久性儲存體。您的任務定義也必須參考容器執行個體上所掛載的磁碟區，才可使用檔案系統。

如需教學，請參閱使用主控台設定 Amazon ECS 的 Amazon EFS 檔案系統。

考量事項

使用 Amazon EFS 磁碟區時，請考量下列事項：

- 對於使用 EC2 啟動類型的任務，透過 Amazon ECS 最佳化 AMI 版本 `20191212` 與容器代理程式 1.35.0 版，Amazon EFS 檔案系統支援被新增為公開預覽。不過，Amazon EFS 檔案系統支援廣泛使用 Amazon ECS 最佳化 AMI `20200319` 版與容器代理程式 1.38.0 版，這包含 Amazon EFS 存取點和 IAM 授權功能。建議您使用 Amazon ECS 最佳化 AMI 版本 `20200319` 或更新版本以使用這些功能。如需詳細資訊，請參閱Amazon ECS 最佳化 Linux AMIs。

> **Note**
>
> 如果您建立自己的 AMI，則必須使用容器代理程式 1.38.0 或更新版本、`ecs-init` 1.38.0-1 版或更新版本，並在 Amazon EC2 執行個體上執行下列命令以啟用 Amazon ECS 磁碟區外掛程式。這些命令取決於您是否使用 Amazon Linux 2 或 Amazon Linux 作為基礎映像。
>
> Amazon Linux 2
>
> ```
> yum install amazon-efs-utils
> systemctl enable --now amazon-ecs-volume-plugin
> ```
>
> Amazon Linux
>
> ```
> yum install amazon-efs-utils
> sudo shutdown -r now
> ```

- 對於託管於 Fargate 的任務，平台版本 1.4.0 或更新版本 (Linux) 上支援 Amazon EFS 檔案系統。如需詳細資訊，請參閱適用於 Amazon ECS 的 Fargate 平台版本。
- 在 Fargate 上託管的任務使用 Amazon EFS 磁碟區時，Fargate 會建立負責管理 Amazon EFS 磁碟區的監督容器。主管容器使用少量任務的記憶體和 CPU。查詢任務中繼資料第 4 版端點時，可看見監督容器。此外，其在 CloudWatch Container Insights 中顯示為容器名稱 `aws-fargate-supervisor`。如需使用 Amazon EC2 啟動類型的詳細資訊，請參閱 Amazon ECS 任務中繼資料端點第 4 版。如需使用 Fargate 啟動類型的詳細資訊，請參閱 Fargate 上任務的 Amazon ECS 任務中繼資料端點第 4 版。
- 外部執行個體不支援使用 Amazon EFS 磁碟區或指定 `EFSVolumeConfiguration`。
- 建議您將代理程式組態檔案中的 `ECS_ENGINE_TASK_CLEANUP_WAIT_DURATION` 參數，設定為小於預設值 (約 1 小時) 的值。此變更有助於防止 EFS 掛載憑證過期，並允許清除未在使用中的掛載。如需詳細資訊，請參閱 Amazon ECS 容器代理程式組態。

使用 Amazon EFS 存取點

Amazon EFS 存取點是應用程式特定的 EFS 檔案系統進入點，以管理應用程式存取共用資料集。如需有關 Amazon EFS 存取點及如何控制對它們的存取的詳細資訊，請參閱《Amazon Elastic File System 使用者指南》中的使用 Amazon EFS 存取點。

存取點可以針對透過存取點提出的所有檔案系統要求，強制執行使用者身分 (包括使用者的 POSIX 群組)。存取點也可以針對檔案系統強制使用不同的根目錄。如此一來，用戶端只能存取指定目錄或其子目錄中的資料。

> **Note**
>
> 建立 EFS 存取點時，您可以在檔案系統上指定要做為根目錄的路徑。在 Amazon ECS 任務定義中使用存取點 ID 參考 EFS 檔案系統時，必須忽略根目錄或將其設定為 /，這會強制執行 EFS 存取點上設定的路徑。

您可以使用 Amazon ECS 任務 IAM 角色，以強制執行特定應用程式使用特定存取點。透過結合 IAM 政策與存取點，您可以為應用程式提供特定資料集的安全存取。如需如何使用任務 IAM 角色的詳細資訊，請參閱Amazon ECS 任務 IAM 角色。

搭配 Amazon ECS 使用 Amazon EFS 磁碟區的最佳實務

當您搭配 Amazon ECS 使用 Amazon EFS 時，請注意下列最佳實務建議。

Amazon EFS 磁碟區的安全性和存取控制

Amazon EFS 提供存取控制功能，可讓您用來確保存放在 Amazon EFS 檔案系統中的資料安全，而且只能從需要它的應用程式存取。您可以透過啟用靜態和傳輸中加密來保護資料。如需詳細資訊，請參閱 Amazon Elastic File System 使用者指南中的 Amazon EFS 的資料加密。

除了資料加密之外，您也可以使用 Amazon EFS 來限制檔案系統的存取。有三種方法可在 EFS 中實作存取控制。

- 安全群組：使用 Amazon EFS 掛載目標，您可以設定用於允許和拒絕網路流量的安全群組。您可以設定連接到 Amazon EFS 的安全群組，以允許來自連接到 Amazon ECS 執行個體的安全群組的 NFS 流量 （連接埠 2049)，或使用`awsvpc`網路模式時，允許 Amazon ECS 任務。
- IAM：您可以使用 IAM 限制對 Amazon EFS 檔案系統的存取。設定後，Amazon ECS 任務需要 IAM 角色，才能讓檔案系統存取以掛載 EFS 檔案系統。如需詳細資訊，請參閱《Amazon Elastic File System File System 使用者指南》中的使用 IAM 控制檔案系統資料存取。

  IAM 政策也可以強制執行預先定義的條件，例如要求用戶端在連線至 Amazon EFS 檔案系統時使用 TLS。如需詳細資訊，請參閱《Amazon Elastic File System 使用者指南》中的用戶端的 Amazon EFS 條件金鑰。 Amazon Elastic File System
- Amazon EFS 存取點 - Amazon EFS 存取點是 Amazon EFS 檔案系統的應用程式特定進入點。您可以使用存取點來強制執行使用者身分，包括使用者 POSIX 群組，用於透過存取點提出的所有檔案系

統請求。存取點也可以針對檔案系統強制使用不同的根目錄。這是為了讓用戶端只能存取指定目錄中的資料或其子目錄中的資料。

IAM 政策

您可以使用 IAM 政策來控制對 Amazon EFS 檔案系統的存取。

您可以使用檔案系統政策，為存取檔案系統的用戶端指定下列動作。

動作	描述
`elasticfilesystem:ClientMount`	提供檔案系統的唯讀存取權。
`elasticfilesystem:ClientWrite`	在檔案系統上提供具有寫入權限。
`elasticfilesystem:ClientRootAccess`	存取檔案系統時，提供使用根使用者的功能。

您需要在政策中指定每個動作。政策可以透過下列方式定義：

- 用戶端型 - 將政策連接至任務角色

  當您建立任務定義時，請設定 IAM 授權選項。
- 資源型 - 將政策連接至 Amazon EFS 檔案系統

  如果資源型政策不存在，預設情況下，檔案系統建立存取權會授予所有主體 (*)。

當您設定 IAM 授權選項時，我們會合併與任務角色和 Amazon EFS 資源為基礎的政策。IAM 授權選項會將任務身分（任務角色）與政策傳遞給 Amazon EFS。這可讓 Amazon EFS 資源型政策具有政策中指定之 IAM 使用者或角色的內容。如果您未設定 選項，Amazon EFS 資源層級政策會將 IAM 使用者識別為「匿名」。

請考慮在 Amazon EFS 檔案系統上實作所有三個存取控制，以獲得最佳安全性。例如，您可以設定連接到 Amazon EFS 掛載點的安全群組，以僅允許從與容器執行個體或 Amazon ECS 任務相關聯的安全群組傳入 NFS 流量。此外，您可以設定 Amazon EFS 要求 IAM 角色存取檔案系統，即使連線來自允許的安全群組。最後，您可以使用 Amazon EFS 存取點來強制執行 POSIX 使用者許可，並指定應用程式的根目錄。

下列任務定義程式碼片段說明如何使用存取點掛載 Amazon EFS 檔案系統。

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

Amazon EFS 磁碟區效能

Amazon EFS 提供兩種效能模式：一般用途和最大輸入/輸出。 一般用途適用於延遲敏感的應用程式，例如內容管理系統和 CI/CD 工具。相反地，Max I/O 檔案系統適用於資料分析、媒體處理和機器學習等工作負載。這些工作負載需要從數百或甚至數千個容器執行平行操作，並需要盡可能最高的彙總輸送量和 IOPS。如需詳細資訊，請參閱《Amazon Elastic File System 使用者指南》中的 Amazon EFS 效能模式。 Amazon Elastic File System

有些延遲敏感工作負載需要由最大 I/O 效能模式提供的較高 I/O 層級，以及一般用途效能模式提供的較低延遲。對於這類工作負載，我們建議建立多個一般用途效能模式的檔案系統。如此一來，只要工作負載和應用程式可以支援，您就可以將應用程式工作負載分散到所有這些檔案系統。

Amazon EFS 磁碟區輸送量

所有 Amazon EFS 檔案系統都有相關聯的計量輸送量，取決於使用佈建輸送量的檔案系統的佈建輸送量量，或使用爆量輸送量的檔案系統儲存在 EFS 標準或單區域儲存類別中的資料量。如需詳細資訊，請參閱《Amazon Elastic File System 使用者指南》中的了解計量輸送量。

Amazon EFS 檔案系統的預設輸送量模式為爆量模式。使用爆量模式時，檔案系統可用的輸送量會隨著檔案系統成長而向內或向外擴展。由於檔案型工作負載通常會飆升，因此需要一段時間內的高輸送量和其餘時間的較低輸送量，Amazon EFS 的設計旨在爆量，以允許一段時間內的高輸送量。此外，由於許多工作負載具有讀取密集性，因此讀取操作會比其他 NFS 操作（例如寫入）以 1：3 的比例計量。

所有 Amazon EFS 檔案系統都會為每個 TB 的 Amazon EFS Standard 或 Amazon EFS One Zone 儲存提供 50 MB/s 的一致基準效能。所有檔案系統（無論大小）都可能爆量到 100 MB/s。具有超過

1TB EFS Standard 或 EFS One Zone 儲存體的檔案系統，每個 TB 的爆量可達 100 MB/s。由於讀取操作是以 1：3 的比率計量，因此每個 MiBs/s。 TiB 當您將資料新增至檔案系統時，檔案系統可用的最大輸送量會隨著 Amazon EFS 標準儲存類別中的儲存量線性且自動擴展。如果您需要的輸送量超過儲存的資料量，您可以將佈建輸送量設定為工作負載所需的特定量。

檔案系統輸送量會跨所有連接到檔案系統的 Amazon EC2 執行個體共用。例如，可以爆量至 100 MB/s 輸送量的 1TB 檔案系統可以從單一 Amazon EC2 執行個體驅動 100 MB/s，每個磁碟機都可以驅動 10 MB/s。如需詳細資訊，請參閱《Amazon Elastic File System 使用者指南》中的 Amazon EFS 效能。Amazon Elastic File System

最佳化 Amazon EFS 磁碟區的成本

Amazon EFS 可為您簡化擴展儲存。當您新增更多資料時，Amazon EFS 檔案系統會自動成長。特別是使用 Amazon EFS 爆量輸送量模式時，Amazon EFS 的輸送量會隨著標準儲存類別中的檔案系統大小增加而擴展。若要在 EFS 檔案系統上改善輸送量，而無須為佈建的輸送量支付額外費用，您可以與多個應用程式共用 Amazon EFS 檔案系統。您可以使用 Amazon EFS 存取點，在共用的 Amazon EFS 檔案系統中實作儲存隔離。如此一來，即使應用程式仍然共用相同的檔案系統，他們也無法存取資料，除非您授權。

隨著資料的增長，Amazon EFS 可協助您自動將不常存取的檔案移至較低的儲存類別。Amazon EFS Standard-Infrequent Access (IA) 儲存類別可降低未每天存取之檔案的儲存成本。這樣做不會犧牲 Amazon EFS 提供的高可用性、高耐用性、彈性和 POSIX 檔案系統存取權。如需詳細資訊，請參閱《Amazon Elastic File System 使用者指南》中的 EFS 儲存類別。

考慮使用 Amazon EFS 生命週期政策，透過將不常存取的檔案移至 Amazon EFS IA 儲存體來自動節省成本。如需詳細資訊，請參閱《Amazon Elastic File System 使用者指南》中的 Amazon EFS 生命週期管理。

建立 Amazon EFS 檔案系統時，您可以選擇 Amazon EFS 是否跨多個可用區域（標準）複寫您的資料，或將您的資料以備援方式存放在單一可用區域。與 Amazon EFS 標準儲存類別相比，Amazon EFS 單區域儲存類別可以大幅降低儲存成本。對於不需要多可用區域彈性的工作負載，請考慮使用 Amazon EFS 單區域儲存類別。您可以將不常存取的檔案移至 Amazon EFS 單區域不常存取，進一步降低 Amazon EFS 單區域儲存的成本。如需詳細資訊，請參閱 Amazon EFS 不頻繁存取。

Amazon EFS 磁碟區資料保護

Amazon EFS 會使用標準儲存類別，跨檔案系統的多個可用區域以備援方式存放您的資料。如果您選取 Amazon EFS 單區域儲存類別，您的資料會備援地存放在單一可用區域中。此外，Amazon EFS 旨在提供指定年份內 99.999999999% (11 個 9) 的耐用性。

如同任何環境，最佳實務是備份和建置防止意外刪除的防護措施。對於 Amazon EFS 資料，該最佳實務包含正常運作且定期測試的備份 AWS Backup。除非您選擇停用此功能，否則使用 Amazon EFS 單區域儲存類別的檔案系統會設定為在建立檔案系統時，依預設自動備份檔案。如需詳細資訊，請參閱《Amazon Elastic File System 使用者指南》中的備份 EFS 檔案系統。 Amazon Elastic File System

在 Amazon ECS 任務定義中指定 Amazon EFS 檔案系統

若要為您的容器使用 Amazon EFS 檔案系統磁碟區，您必須在任務定義中指定磁碟區並掛載點組態。以下任務定義 JSON 片段說明容器 `volumes` 和 `mountPoints` 物件的語法。

```
{
    "containerDefinitions": [
        {
            "name": "container-using-efs",
            "image": "amazonlinux:2",
            "entryPoint": [
                "sh",
                "-c"
            ],
            "command": [
                "ls -la /mount/efs"
            ],
            "mountPoints": [
                {
                    "sourceVolume": "myEfsVolume",
                    "containerPath": "/mount/efs",
                    "readOnly": true
                }
            ]
        }
    ],
    "volumes": [
        {
            "name": "myEfsVolume",
            "efsVolumeConfiguration": {
                "fileSystemId": "fs-1234",
                "rootDirectory": "/path/to/my/data",
                "transitEncryption": "ENABLED",
                "transitEncryptionPort": integer,
                "authorizationConfig": {
                    "accessPointId": "fsap-1234",
                    "iam": "ENABLED"
                }
```

```
            }
        }
    ]
}
```

`efsVolumeConfiguration`

類型：物件

必要：否

只有使用 Amazon EFS 磁碟區時才會指定此參數。

`fileSystemId`

類型：字串

必要：是

要使用的 Amazon EFS 檔案系統識別碼。

`rootDirectory`

類型：字串

必要：否

在 Amazon EFS 檔案系統中的目錄，其將掛載作為主機內的根目錄。如果省略此參數，使用 Amazon EFS 磁碟區的根目錄。指定 / 的效果與忽略此參數的效果相同。

> **Important**
>
> 如果在 `authorizationConfig` 中指定 EFS 存取點，則必須忽略根目錄參數或設定為 /，這將強制執行 EFS 存取點上設定的路徑。

`transitEncryption`

類型：字串

有效值：`ENABLED` | `DISABLED`

必要：否

指定是否要對在 Amazon ECS 主機和 Amazon EFS 伺服器之間傳輸中的 Amazon EFS 資料啟用加密功能。如果使用 Amazon EFS IAM 授權，則必須啟用傳輸加密。如果省略此參數，系統會使用 DISABLED 的預設值。如需詳細資訊，請參閱《Amazon Elastic File System 使用者指南》中的加密傳輸中的資料。

`transitEncryptionPort`

類型：整數

必要：否

在 Amazon ECS 主機和 Amazon EFS 伺服器之間傳送加密資料時所使用的連接埠。如果您未指定傳輸加密連接埠，它會使用 Amazon EFS 掛載協助程式使用的連接埠選擇策略。如需詳細資訊，請參閱《Amazon Elastic File System 使用者指南》中的 EFS 掛載協助程式。

`authorizationConfig`

類型：物件

必要：否

Amazon EFS 檔案系統的授權組態詳細資訊。

`accessPointId`

類型：字串

必要：否

要使用的存取點 ID。如果指定了存取點，則必須省略 `efsVolumeConfiguration` 中的根目錄值，或設定為 `/`，這將強制執行在 EFS 存取點上設定的路徑。如果使用存取點，則必須在 `EFSVolumeConfiguration` 中啟用傳輸加密。如需詳細資訊，請參閱《Amazon Elastic File System 使用者指南》中的使用 Amazon EFS 存取點。

`iam`

類型：字串

有效值：`ENABLED` | `DISABLED`

必要：否

掛載 Amazon ECS 檔案系統時，指定是否使用任務定義中定義的 Amazon EFS 任務 IAM 角色。如果已啟用，必須在 `EFSVolumeConfiguration` 中啟用傳輸加密。如果省略此參數，系統會使用 `DISABLED` 的預設值。如需詳細資訊，請參閱任務的 IAM 角色。

使用主控台設定 Amazon ECS 的 Amazon EFS 檔案系統

了解如何搭配 Amazon ECS 使用 Amazon Elastic File System (Amazon EFS) 檔案系統。

步驟 1：建立 Amazon ECS 叢集。

使用下列步驟來建立 Amazon ECS 叢集。

建立新叢集 (Amazon ECS 主控台)

在開始之前，請指派適當的 IAM 許可。如需詳細資訊，請參閱the section called “Amazon ECS 叢集範例”。

1. 開啟主控台，網址為 https://console.aws.amazon.com/ecs/v2。
2. 從導覽列中選取要使用的「區域」。
3. 在導覽窗格中，選擇叢集。
4. 在 Clusters (叢集) 頁面上，選擇 Create cluster (建立叢集)。
5. 在叢集組態下的叢集名稱中，輸入叢集名稱的 `EFS-tutorial`。
6. (選用) 若要變更任務和服務啟動所在的 VPC 和子網路，請在 Networking (聯網) 下，執行下列任一操作：
   - 若要移除子網路，請在 Subnets (子網路) 下，對您要移除之每一個子網路選擇 X。
   - 若要變更為非 default (預設) VPC，請在 VPC 下，選擇現有的 VPC，然後在 Subnets (子網路) 下選擇各個子網路。
7. 若要將 Amazon EC2 執行個體新增至叢集，展開 Infrastructure (基礎設施)，然後選取 Amazon EC2 執行個體。接下來，設定作為容量提供者的 Auto Scaling 群組：
   - 若要建立 Auto Scaling 群組，請從 Auto Scaling group (ASG) (Auto Scaling 群組 (ASG)) 中選取 Create new group (建立新群組)，然後提供有關該群組的下列詳細資訊：
     - 在作業系統/架構中，選擇 Amazon Linux 2。
     - 對於 EC2 instance type (EC2 執行個體類型)，選擇 `t2.micro`。

       對於 SSH key pair (SSH 金鑰對)，選擇在連線到執行個體時證明您身分的金鑰對。
     - 在容量中，輸入 1。
8. 選擇 Create (建立)。

步驟 2：為 Amazon EC2 執行個體和 Amazon EFS 檔案系統建立安全群組

在此步驟中，您將為 Amazon EC2 執行個體建立安全群組，以允許連接埠 80 上的入站網路流量，並為 Amazon EFS 檔案系統建立一個允許從容器執行個體進行入站存取的安全群組。

使用下列選項為 Amazon EC2 執行個體建立安全群組：

- 安全群組名稱 - 為您的安全群組輸入唯一的名稱。
- VPC - 選擇您先前為叢集所確認的 VPC。
- 傳入規則
  - 類型 - HTTP
  - 資源來源 - 0.0.0.0/0。

使用下列選項為 Amazon EFS 檔案系統建立安全群組：

- 安全群組名稱 - 為您的安全群組輸入唯一的名稱。例如：`EFS-access-for-sg-dc025fa2`。
- VPC - 選擇您先前為叢集所確認的 VPC。
- 傳入規則
  - 類型 - NFS
  - 來源 - 使用您為執行個體建立的安全群組 ID 進行自訂。

如需有關如何建立安全群組的資訊，請參閱《Amazon EC2 使用者指南》中的為您的 Amazon EC2 執行個體建立安全群組。 Amazon EC2

步驟 3：建立 Amazon EFS 檔案系統

在此步驟中，您將建立一個 Amazon EFS 檔案系統。

建立 Amazon ECS 任務的 Amazon EFS 檔案系統。

1. 開啟 Amazon Elastic File System 主控台，網址為 https://console.aws.amazon.com/efs/。
2. 選擇 Create file system (建立檔案系統)。
3. 輸入檔案系統的名稱，然後選擇容器執行個體所託管的 VPC。根據預設，指定之 VPC 中的每個子網路，都會收到一個使用該 VPC 之預設安全群組的掛載目標。然後，選擇自訂。

> **Note**
>
> 本教學課程假設您的 Amazon EFS 檔案系統、Amazon ECS 叢集、容器執行個體和任務位於相同的 VPC 中。如需從不同 VPC 掛載檔案系統的詳細資訊，請參閱《Amazon EFS 使用者指南》中的逐步解說：從不同 VPC 掛載檔案系統。

4. 在檔案系統設定頁面上，設定選用設定，然後在效能設定下，為您的檔案系統選擇高載輸送量模式。設定完設定之後，請選取下一步。

   a. (選用) 為您的檔案系統新增標籤。例如，您可以在 Name (名稱) 鍵旁的 Value (值) 欄中輸入名稱，藉此為檔案系統指定不會重複的一名稱。

   b. (選用) 啟用生命週期管理，以避免在不常存取的儲存體上浪費成本。如需詳細資訊，請參閱《Amazon Elastic File System 使用者指南》中的 EFS 生命週期管理。

   c. (選用) 啟用加密。選取此核取方塊以啟用靜態 Amazon EFS 檔案系統的加密。

5. 在網路存取頁面的掛載目標下，將每個可用區域的現有安全群組組態替換為您為 步驟 2：為 Amazon EC2 執行個體和 Amazon EFS 檔案系統建立安全群組 中的檔案系統建立的安全群組，然後選擇下一步。
6. 您不需要為本教學課程設定檔案系統政策，因此您可以選擇下一步略過此區段。
7. 檢閱您的檔案系統選項，然後選擇建立以完成程序。
8. 從檔案系統畫面中，記錄檔案系統 ID。在下一個步驟中，您將會在 Amazon ECS 任務定義中參考這個值。

步驟 4：將內容新增至 Amazon EFS 檔案系統

在此步驟中，您會將 Amazon EFS 檔案系統掛載到 Amazon EC2 執行個體，並將內容新增到其中。這是為了在本教學課程中進行測試，以說明資料的持久性。使用此功能時，通常會將應用程式或其他方法的資料寫入 Amazon EFS 檔案系統。

建立 Amazon EC2 執行個體並掛載 Amazon EFS 檔案系統

1. 在 https://console.aws.amazon.com/ec2/ 開啟 Amazon EC2 主控台。
2. 選擇 Launch Instance (啟動執行個體)。
3. 在應用程式和作業系統映像 (Amazon Machine Image) 下，選取 Amazon Linux 2 AMI (HVM)。
4. 在執行個體類型中，保留預設執行個體類型 `t2.micro`。
5. 在金鑰對 (登入) 下，選取用於 SSH 存取執行個體的金鑰對。

6. 在網路設定下，選取為 Amazon EFS 檔案系統和 Amazon ECS 叢集指定的 VPC。選取子網路和在 步驟 2：為 Amazon EC2 執行個體和 Amazon EFS 檔案系統建立安全群組 中建立的執行個體安全群組。設定執行個體的安全群組。確認已啟用自動指派公有 IP。
7. 在設定儲存下，選擇檔案系統的編輯按鈕，然後選擇 EFS。選取您在 步驟 3：建立 Amazon EFS 檔案系統 中建立的檔案系統。您可以選擇性地變更掛載點，或保留預設值。

   > **Important**
   >
   > 您必須先選擇子網路，然後才能將檔案系統新增至執行個體。

8. 清除自動建立並連接安全群組。保持選取另一個核取方塊。選擇 Add shared file system (新增共用檔案系統)。
9. 在 Advanced Details (進階詳細資訊) 下，確保透過 Amazon EFS 檔案系統掛載步驟自動填入使用者資料指令碼。
10. 在摘要下，確定執行個體數量為 1。選擇啟動執行個體。
11. 在啟動執行個體頁面上，選擇檢視所有執行個體以查看執行個體的狀態。初始時，您的執行個體狀態為 PENDING。當狀態變更為 RUNNING 且執行個體通過所有狀態檢查之後，執行個體即可供使用。

現在，您可以連線到 Amazon EC2 執行個體，並將內容新增至 Amazon EFS 檔案系統。

連線到 Amazon EC2 執行個體，並將內容新增至 Amazon EFS 檔案系統。

1. 您建立之 Amazon EC2 執行個體的 SSH。如需詳細資訊，請參閱《Amazon EC2 使用者指南》中的使用 SSH 連線至 Linux 執行個體。
2. 從終端機視窗，執行 df -T 命令，確認 Amazon EFS 檔案系統已掛載。在下列輸出中，我們已反白顯示 Amazon EFS 檔案系統掛載。

```
$ df -T
Filesystem     Type            1K-blocks    Used        Available Use% Mounted on
devtmpfs       devtmpfs           485468       0           485468   0% /dev
tmpfs          tmpfs              503480       0           503480   0% /dev/shm
tmpfs          tmpfs              503480     424           503056   1% /run
tmpfs          tmpfs              503480       0           503480   0% /sys/fs/
cgroup
/dev/xvda1     xfs               8376300 1310952          7065348  16% /
127.0.0.1:/    nfs4     9007199254739968       0 9007199254739968   0% /mnt/efs/fs1
```

```
tmpfs           tmpfs               100700       0           100700    0% /run/
user/1000
```

3. 導覽至掛載 Amazon EFS 檔案系統所在的目錄。在上述範例中，即為 `/mnt/efs/fs1`。
4. 建立名為 `index.html` 且具有下列內容的檔案：

```
<html>
    <body>
        <h1>It Works!</h1>
        <p>You are using an Amazon EFS file system for persistent container
 storage.</p>
    </body>
</html>
```

步驟 5：建立任務定義

以下任務定義會建立名稱為 `efs-html` 的資料磁碟區。nginx 容器會將主機資料磁碟區掛載於 NGINX 根目錄 `/usr/share/nginx/html`。

使用 Amazon ECS 主控台建立新的任務定義

1. 開啟主控台，網址為 https://console.aws.amazon.com/ecs/v2。
2. 在導覽窗格中，選擇 Task Definitions (任務定義)。
3. 選擇 Create new task definitio (建立新任務定義)、Create new task definition with JSON (使用 JSON 建立新的任務定義)。
4. 在 JSON 編輯器方塊中，複製並貼上以下 JSON 文字，以 Amazon EFS 檔案系統的 ID 取代 `fileSystemId`。

```
{
    "containerDefinitions": [
        {
            "memory": 128,
            "portMappings": [
                {
                    "hostPort": 80,
                    "containerPort": 80,
                    "protocol": "tcp"
                }
            ],
            "essential": true,
```

```
            "mountPoints": [
                {
                    "containerPath": "/usr/share/nginx/html",
                    "sourceVolume": "efs-html"
                }
            ],
            "name": "nginx",
            "image": "nginx"
        }
    ],
    "volumes": [
        {
            "name": "efs-html",
            "efsVolumeConfiguration": {
                "fileSystemId": "fs-1324abcd",
                "transitEncryption": "ENABLED"
            }
        }
    ],
    "family": "efs-tutorial",
    "executionRoleArn":"arn:aws:iam::111122223333:role/ecsTaskExecutionRole"
}
```

> **Note**
>
> 您可以將下列許可新增至 Amazon ECS 任務執行 IAM 角色，以允許 Amazon ECS 代理程式在啟動時尋找 Amazon EFS 檔案系統並將其掛載至任務。
>
> - `elasticfilesystem:ClientMount`
> - `elasticfilesystem:ClientWrite`
> - `elasticfilesystem:DescribeMountTargets`
> - `elasticfilesystem:DescribeFileSystems`

5. 選擇 Create (建立)。

步驟 6：執行任務並檢視結果

現在，已建立您的 Amazon EFS 檔案系統，並且提供 NGINX 容器的 Web 內容，您可以使用建立的任務定義來執行任務。NGINX Web 伺服器會隨即提供您簡單的 HTML 頁面。如果您在 Amazon EFS 檔案系統中更新了內容，這些變更也會傳播至所有也掛載該檔案系統的容器。

任務會在您為叢集定義的子網路中執行。

使用主控台執行任務並檢視結果

1. 開啟主控台，網址為 https://console.aws.amazon.com/ecs/v2。
2. 在 Clusters (叢集) 頁面上，選取要在哪個叢集中執行獨立任務。

   決定您要從中啟動服務的資源。

啟動服務的資源	步驟
叢集	a. 在叢集頁面，選取您要在哪個叢集中建立服務。 b. 在 Tasks (任務) 標籤中，選擇 Run new task (執行新任務)。
啟動類型	a. 在 Task (任務) 頁面中，選擇任務定義。 b. 如果有多個修訂版，則請選取修訂版。 c. 依序選擇 Create (建立)、Run task (執行任務)。

3. (選用) 選擇已排程的任務在叢集基礎設施中的分佈方式。展開 Compute configuration (運算組態)，然後執行下列操作：

分發方法	步驟
啟動類型	a. 在運算選項區段中，選取啟動類型。 b. 在啟動類型中，選擇 EC2。

4. 針對 Application type (應用程式類型)，選擇 Task (任務)。
5. 在任務定義中，選擇您先前建立的 `efs-tutorial` 任務定義。

6. 在所需任務中，輸入 1。
7. 選擇 Create (建立)。
8. 在叢集頁面上，選擇基礎設施。
9. 在容器執行個體下，選擇要連線的容器執行個體。
10. 在容器執行個體頁面的聯網下，記錄執行個體的公有 IP 或公有 DNS。
11. 開啟瀏覽器並輸入公有 IP 地址。您應該看到下列訊息：

```
It works!
You are using an Amazon EFS file system for persistent container storage.
```

> **Note**
>
> 如果您未看到該訊息，請確認您容器執行個體的安全群組，允許連接埠 80 上的入站網路流量，且檔案系統的安全群組允許來自容器執行個體的入站存取。

搭配 Amazon ECS 使用 FSx for Windows File Server 磁碟區

FSx for Windows File Server 提供全受管 Windows 檔案伺服器，由 Windows 檔案系統提供支援。搭配使用 FSx for Windows File Server 與 ECS 時，您可以使用持續、分散、共用、靜態的檔案儲存來佈建 Windows 任務。如需詳細資訊，請參閱什麼是 FSx for Windows File Server？。

> **Note**
>
> 使用 Amazon ECS 最佳化 Windows Server 2016 Full AMI 的 EC2 執行個體不支援 FSx for Windows File Server ECS 任務磁碟區。
> 您無法在 Fargate 組態的 Windows 容器中使用 FSx for Windows File Server 磁碟區。反之，您可以修改容器以在啟動時掛載它們。

您可以使用 FSx for Windows File Server 來部署需要存取共用外部儲存體、高可用性區域儲存體或高輸送量儲存體的 Windows 工作負載。您可以將一或多個 FSx for Windows File Server 檔案系統磁碟區掛載到在 Amazon ECS Windows 執行個體上執行的 Amazon ECS 容器。您可以在單一 Amazon ECS 任務中的多個 Amazon ECS 容器之間共用 FSx for Windows File Server 檔案系統磁碟區。

若要讓 FSx for Windows File Server 與 ECS 搭配使用，在任務定義中包含 FSx for Windows File Server 檔案系統 ID 和相關資訊。如下列任務定義 JSON 片段範例。在建立並執行任務定義之前，您必須準備好以下事項。

- 加入有效的網域的 ECS Windows EC2 執行個體。它可由 AWS Directory Service for Microsoft Active Directory、內部部署 Active Directory 或 Amazon EC2 上的自我託管 Active Directory 託管。
- AWS Secrets Manager 秘密或 Systems Manager 參數，其中包含用來加入 Active Directory 網域和連接 FSx for Windows File Server 檔案系統的登入資料。憑證值是您在建立 Active Directory 時輸入的名稱和密碼憑證。

如需相關教學，請參閱了解如何為 Amazon ECS 設定 FSx for Windows File Server 檔案系統。

考量事項

在使用 FSx for Windows File Server 磁碟區時，請考量下列事項：

- FSx for Windows File Server 與 Amazon ECS 僅支援 Windows Amazon EC2 執行個體。不支援 Linux Amazon EC2 執行個體。
- FSx for Windows File Server 與 Amazon ECS 不支援 AWS Fargate。
- 具有 `awsvpc` 網路模式的 FSx for Windows File Server 與 Amazon ECS 需要容器代理程式的版本 `1.54.0` 或更新版本。
- Amazon ECS 任務使用的磁碟機代號的上限為 23。具有 FSx for Windows File Server 磁碟區的每個任務都會取得指派給它的磁碟機代號。
- 根據預設，任務資源清除時間為任務結束後的 3 小時。即使沒有任務正在使用它，任務建立的檔案映射會持續 3 小時。使用 Amazon ECS 環境變數 `ECS_ENGINE_TASK_CLEANUP_WAIT_DURATION` 可設定預設清理時間。如需詳細資訊，請參閱Amazon ECS 容器代理程式組態。
- 任務通常只會在與 FSx for Windows File Server 檔案系統相同的 VPC 中執行。但是，如果 Amazon ECS 叢集 VPC 與 FSx for Windows File Server 檔案系統之間透過 VPC 對等互連建立了網路連線，則可以跨 VPC 支援。
- 透過設定 VPC 安全群組，可在網路層級控制對 FSx for Windows File Server 檔案系統的存取。只有加入 Active Directory 網域且已正確設定 Active Directory 安全群組的 EC2 執行個體上託管的任務，才能存取 FSx for Windows File Server 檔案共用。如果安全群組設定錯誤，Amazon ECS 無法使用下列錯誤訊息啟動任務：`unable to mount file system fs-id`。”
- FSx for Windows File Server 已與 AWS Identity and Access Management (IAM) 整合，以控制 IAM 使用者和群組可以對特定 FSx for Windows File Server 資源採取的動作。透過用戶端授權，客戶可

以定義允許或拒絕存取特定 FSx for Windows File Server 檔案系統的 IAM 角色，選擇性地要求唯讀存取，以及選擇性地允許或禁止從用戶端對檔案系統進行根存取。如需詳細資訊，請參閱《Amazon FSx Windows 使用者指南》中的安全性。

搭配 Amazon ECS 使用 FSx for Windows File Server 的最佳實務

當您將 FSx for Windows File Server 與 Amazon ECS 搭配使用時，請注意下列最佳實務建議。

FSx for Windows File Server 的安全性和存取控制

FSx for Windows File Server 提供下列存取控制功能，可讓您用來確保存放在 FSx for Windows File Server 檔案系統中的資料是安全的，而且只能從需要它的應用程式存取。

FSx for Windows File Server 磁碟區的資料加密

FSx for Windows File Server 支援兩種形式的檔案系統加密。它們是傳輸中資料的加密和靜態加密。在支援 SMB 通訊協定 3.0 或更新版本的容器執行個體上映射的檔案共用支援傳輸中的資料加密。建立 Amazon FSx 檔案系統時，會自動啟用靜態資料加密。當您存取檔案系統時，Amazon FSx 會使用 SMB 加密自動加密傳輸中的資料，而不需要您修改應用程式。如需詳細資訊，請參閱《Amazon FSx for Windows File Server 使用者指南》中的 Amazon FSx 中的資料加密。 FSx

使用 Windows ACLs進行資料夾層級存取控制

Windows Amazon EC2 執行個體使用 Active Directory 登入資料存取 Amazon FSx 檔案共用。它使用標準 Windows 存取控制清單 (ACLs) 進行精細的檔案層級和資料夾層級存取控制。您可以建立多個登入資料，每個登入資料都會對應至特定任務，用於共用內特定資料夾。

在下列範例中，任務App01可以使用儲存在 Secrets Manager 中的登入資料來存取 資料夾。其 Amazon Resource Name (ARN) 為 1234。

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

在另一個範例中，任務App02可以使用儲存在 Secrets Manager 中的登入資料來存取 資料夾。其 ARN 是 6789。

```
"rootDirectory": "\\path\\to\\my\\data\App02",
"credentialsParameter": "arn-6789",
```

```
"domain": "corp.fullyqualified.com",
```

在 Amazon ECS 任務定義中指定 FSx for Windows File Server 檔案系統

若要將 FSx for Windows File Server 檔案系統磁碟區用於您的容器，請在任務定義中指定磁碟區並掛載點組態。以下任務定義 JSON 片段說明容器 `volumes` 和 `mountPoints` 物件的語法。

```
{
    "containerDefinitions": [
        {
            "entryPoint": [
                "powershell",
                "-Command"
            ],
            "portMappings": [],
            "command": ["New-Item -Path C:\\fsx-windows-dir\\index.html -ItemType file
 -Value '<html> <head> <title>Amazon ECS Sample App</title> <style>body {margin-top:
 40px; background-color: #333;} </style> </head><body> <div style=color:white;text-
align:center> <h1>Amazon ECS Sample App</h1> <h2>It Works!</h2> <p>You are using Amazon
 FSx for Windows File Server file system for persistent container storage.</p>' -
Force"],
            "cpu": 512,
            "memory": 256,
            "image": "mcr.microsoft.com/windows/servercore/iis:windowsservercore-
ltsc2019",
            "essential": false,
            "name": "container1",
            "mountPoints": [
                {
                    "sourceVolume": "fsx-windows-dir",
                    "containerPath": "C:\\fsx-windows-dir",
                    "readOnly": false
                }
            ]
        },
        {
            "entryPoint": [
                "powershell",
                "-Command"
            ],
            "portMappings": [
                {
                    "hostPort": 443,
```

```
                    "protocol": "tcp",
                    "containerPort": 80
                }
            ],
            "command": ["Remove-Item -Recurse C:\\inetpub\\wwwroot\\* -Force; Start-
Sleep -Seconds 120; Move-Item -Path C:\\fsx-windows-dir\\index.html -Destination C:\
\inetpub\\wwwroot\\index.html -Force; C:\\ServiceMonitor.exe w3svc"],
            "mountPoints": [
                {
                    "sourceVolume": "fsx-windows-dir",
                    "containerPath": "C:\\fsx-windows-dir",
                    "readOnly": false
                }
            ],
            "cpu": 512,
            "memory": 256,
            "image": "mcr.microsoft.com/windows/servercore/iis:windowsservercore-
ltsc2019",
            "essential": true,
            "name": "container2"
        }
    ],
    "family": "fsx-windows",
    "executionRoleArn": "arn:aws:iam::111122223333:role/ecsTaskExecutionRole",
    "volumes": [
        {
            "name": "fsx-windows-dir",
            "fsxWindowsFileServerVolumeConfiguration": {
                "fileSystemId": "fs-0eeb5730b2EXAMPLE",
                "authorizationConfig": {
                    "domain": "example.com",
                    "credentialsParameter": "arn:arn-1234"
                },
                "rootDirectory": "share"
            }
        }
    ]
}
```

FSxWindowsFileServerVolumeConfiguration

類型：物件

必要：否

當您為任務儲存使用 FSx for Windows File Server 檔案系統時，指定此參數。

`fileSystemId`

類型：字串

必要：是

要使用的 FSx for Windows File Server 檔案系統 ID。

`rootDirectory`

類型：字串

必要：是

FSx for Windows File Server 檔案系統中的目錄，其將掛載做為主機內的根目錄。

`authorizationConfig`

`credentialsParameter`

類型：字串

必要：是

授權憑證選項：

- Secrets Manager 秘密的 Amazon Resource Name (ARN)。
- Systems Manager 參數的 Amazon Resource Name (ARN)。

`domain`

類型：字串

必要：是

由 AWS Directory Service for Microsoft Active Directory(AWS Managed Microsoft AD) 目錄或自我託管 EC2 Active Directory 託管的完整網域名稱。

儲存 FSx for Windows File Server 磁碟區登入資料的方法

有兩種不同的方法可存放憑證，以便與憑證參數搭配使用。

- AWS Secrets Manager 秘密

  您可以使用其他類型的秘密類別，在 AWS Secrets Manager 主控台中建立此登入資料。您可以為每個鍵/值對、使用者名稱/管理員以及密碼/##新增一列。

- Systems Manager 參數

  您可以在 Systems Manager 參數主控台中建立此憑證，方法是在下列範例程式碼片段的表單中輸入文字。

```
{
  "username": "admin",
  "password": "password"
}
```

任務定義 `FSxWindowsFileServerVolumeConfiguration` 參數中的 `credentialsParameter` 保留秘密 ARN 或 Systems Manager 參數 ARN。如需詳細資訊，請參閱 《Secrets Manager 使用者指南》中的什麼是 AWS Secrets Manager 和 Systems Manager 使用者指南中的Systems Manager 參數存放區。

了解如何為 Amazon ECS 設定 FSx for Windows File Server 檔案系統

了解如何啟動 Amazon ECS 最佳化 Windows 執行個體，該執行個體託管 FSx for Windows File Server 檔案系統和可存取檔案系統的容器。若要這麼做，請先建立 AWS Directory Service AWS Managed Microsoft Active Directory。然後，您可以使用 Amazon EC2 執行個體和任務定義建立 FSx for Windows File Server 檔案伺服器檔案系統和叢集。您可以設定容器的任務定義，以使用 FSx for Windows File Server 檔案系統。最後，您將測試檔案系統。

每次啟動或刪除 Active Directory 或 FSx for Windows File Server 檔案系統時，需要 20 至 45 分鐘。請準備保留至少 90 分鐘以完成教學課程，或在幾個工作階段中完成教學課程。

教學課程的先決條件

- 管理使用者。請參閱 設定以使用 Amazon ECS。。
- (選用) 用於透過 RDP 存取連線至 EC2 Windows 執行個體的 PEM 金鑰對。如需有關如何建立金鑰對的資訊，請參閱《Amazon EC2 使用者指南》中的 Amazon EC2 金鑰對和 Amazon EC2 Amazon EC2執行個體。 Amazon EC2
- 具有至少一個公有、一個私有子網路和一個安全群組的 VPC。您可以使用預設 VPC。您不需要 NAT 閘道或裝置。 AWS Directory Service 不支援與 Active Directory 搭配的網路位址轉譯 (NAT)。為了

可以運作，Active Directory、FSx for Windows File Server 檔案系統、ECS 叢集和 EC2 執行個體必須位於 VPC 內。如需 VPCs和作用中目錄的詳細資訊，請參閱建立 VPC 和建立 AWS Managed Microsoft AD 的先決條件。

- IAM ecsInstanceRole 和 ecsTaskExecutionRole 許可與您的帳戶建立關聯。這些服務連結的角色可讓服務代表您進行 API 呼叫，並存取容器、機密、目錄和檔案伺服器。

步驟 1：建立 IAM 存取角色

使用 AWS Management Console建立叢集。

1. 請參閱 Amazon ECS 容器執行個體 IAM 角色 來檢查您是否擁有 ecsInstanceRole，並查看如何在沒有的情況下建立一個。
2. 建議您針對實際生產環境中的最低許可自訂角色政策。為了完成本教學課程，請確認下列 AWS 受管政策已連接至您的 ecsInstanceRole。連接政策 (如果尚未連接)。

   - AmazonEC2ContainerServiceforEC2Role
   - AmazonSSMManagedInstanceCore
   - AmazonSSMDirectoryServiceAccess

   連接 AWS 受管政策。

   a. 開啟 IAM 主控台。
   b. 在導覽窗格中，選擇 Roles (角色)。
   c. 選擇 AWS 受管角色。
   d. 選擇 Permissions, Attach policies. (許可、連接政策。)。
   e. 若要縮小可連接的政策，請使用 Filter (篩選)。
   f. 選取適用的政策，然後選擇 Attach Policy (連接政策)。
3. 請參閱 Amazon ECS 任務執行 IAM 角色 來檢查您是否擁有 ecsTaskExecutionRole，並查看如何在沒有的情況下建立一個。

   建議您針對實際生產環境中的最低許可自訂角色政策。為了完成本教學課程，請確認下列 AWS 受管政策已連接至您的 ecsTaskExecutionRole。連接政策 (如果尚未連接)。使用上一節中提供的程序來連接 AWS 受管政策。

   - SecretsManagerReadWrite
   - AmazonFSxReadOnlyAccess

- AmazonSSMReadOnlyAccess
- AmazonECSTaskExecutionRolePolicy

步驟 2：建立 Windows Active Directory (AD)

1. 請遵循 Directory Service 管理指南中建立 AWS Managed Microsoft AD AWS 中所述的步驟。使用您在本教學課程中指定的 VPC。在建立 AWS Managed Microsoft AD 的步驟 3 中，儲存使用者名稱和管理員密碼，以供後續步驟使用。此外，記下未來步驟的完整目錄 DNS 名稱。您可以在建立 Active Directory 時完成下列步驟。
2. 建立要在下列步驟中使用的 AWS Secrets Manager 秘密。如需詳細資訊，請參閱《Secrets Manager 使用者指南》中的 Secrets Manager 入門。 AWS

   a. 開啟 Secrets Manager 主控台。
   b. 按一下 Store a new secret (存放新機密)。
   c. 選取 Other type of secrets (其他機密類型)。
   d. 對於 Secret key/value (機密鍵/值)，在第一個資料列中建立具有值 **admin** 的索引鍵 **username**。按一下 + Add row (+ 新增列)。
   e. 在新資料列中，建立索引鍵 **password**。針對值，輸入您在建立 AWS 受管 AD 目錄的步驟 3 中輸入的密碼。
   f. 按一下 Next (下一步) 按鈕。
   g. 提供機密名稱和描述。按一下 Next (下一步)。
   h. 按一下 Next (下一步)。按一下 Store (存放)。
   i. 從 Secrets (機密) 頁面清單上，按一下您剛才建立的機密。
   j. 儲存新機密的 ARN，以便在後續步驟中使用。
   k. 您可以繼續進行下一個步驟，同時建立 Active Directory。

步驟 3：驗證並更新安全群組

在此步驟中，您會驗證並更新您正在使用的安全群組的規則。為此，您可以使用為 VPC 建立的預設安全群組。

驗證並更新安全群組。

您需要建立或編輯安全群組，以便連接埠可以接收和傳送資料，請參閱 FSx for Windows File Server 使用者指南中的 Amazon VPC 安全群組。您可以藉由建立安全群組傳入規則完成此操作，規則顯示在

以下傳入規則資料表的第一列。此規則允許來自網路界面 (及其關聯執行個體) 的傳入流量，而這些網路界面會指派給安全群組。您建立的所有雲端資源都位於同一 VPC 內，並連接至相同的安全群組。因此，此規則允許所需的流量往來於 FSx for Windows File Server 檔案系統、Active Directory 和 ECS 執行個體。其他傳入規則允許流量提供網站和 RDP 存取服務，以便連線至您的 ECS 執行個體。

下表顯示本教學課程所需的安全群組傳入規則。

Type	通訊協定	連接埠範圍	來源
所有流量	全部	全部	*sg-securitygroup*
HTTPS	TCP	443	0.0.0.0/0
RDP	TCP	3389	您的筆記型電腦 IP 地址

下表顯示本教學課程所需的安全群組傳出規則。

Type	通訊協定	連接埠範圍	目的地
所有流量	全部	全部	0.0.0.0/0

1. 開啟 EC2 主控台並從左側選單選取 Security Groups (安全群組)。
2. 從現在顯示的安全群組清單中，選取您要用於本教學課程的安全群組左側的核取方塊。

   隨即顯示您的安全群組詳細資訊。
3. 編輯傳入和傳出規則，方法是選取 Inbound rules (傳入規則) 或 Outbound rules (傳出規則) 索引標籤，然後選擇 Edit inbound rules (編輯傳入規則) 或 Edit outbound rules (編輯傳出規則) 按鈕。編輯規則以與上表中顯示的規則相符。在本教學稍後建立 EC2 執行個體之後，請使用 EC2 執行個體的公有 IP 地址編輯傳入規則 RDP 來源，如 Amazon EC2 使用者指南中的使用 RDP 連線至 Windows 執行個體中所述。

步驟 4：建立 FSx for Windows File Server 檔案系統

在您的安全群組經過驗證和更新，且您的 Active Directory 已建立並處於作用中狀態之後，請在與您的 Active Directory 相同的 VPC 中建立 FSx for Windows File Server 檔案系統。請依照下列步驟，建立 FSx for Windows File Server 檔案系統，以供您的 Windows 任務使用。

建立第一個檔案系統。

1. 開啟 Amazon FSx 主控台。
2. 在儀表板上，選擇 Create file system (建立檔案系統) 以啟動檔案系統建立精靈。
3. 在 Select file system type (選取檔案系統類型) 頁面中，選擇 FSx for Windows File Server，然後選擇 Next (下一步)。Create file system (建立檔案系統) 頁面隨即顯示。
4. 在 File system details (檔案系統詳細資訊) 區段中，輸入檔案系統的名稱。命名檔案系統可讓您更輕鬆地尋找和管理檔案系統。您最多可使用 256 個 Unicode 字元。允許的字元如下：英文字母、數字、空格和特殊字元加號 (+)、減號 (-)、等號 (=)、句點 (.)、下劃線 (_)、冒號 (:) 和斜線 (/)。
5. 對於 Deployment type (部署類型)，請選擇 Single-AZ (單一可用區)，部署在單一可用區域中部署的檔案系統。單一可用區 2 是最新一代的單一可用區域檔案系統，支援 SSD 和 HDD 儲存。
6. 對於 Storage type (儲存類型)，請選擇 EBS。
7. 對於 Storage capacity (儲存容量)，請輸入最小儲存容量。
8. 保留 Throughput capacity (輸送容量) 的預設設定。
9. 在網路與安全區段中，選擇您為 AWS Directory Service 目錄選擇的相同 Amazon VPC。
10. 對於 VPC Security Groups (VPC 安全群組) 下，選擇您在步驟 3：驗證並更新您的安全群組中驗證的安全群組。
11. 對於 Windows authentication (Windows 身分驗證)，選擇 AWS 受管 Microsoft Active Directory，然後從清單中選擇您的 AWS Directory Service 目錄。
12. 對於 Encryption (加密)，請保留 aws/fsx (default) (aws/fsx (預設)) 的預設 Encryption key (加密金鑰) 設定。
13. 請保留 Maintenance preferences (維護偏好設定) 的預設設定。
14. 按一下 Next (下一步) 按鈕。
15. 檢閱顯示在 Create file system (建立檔案系統) 頁面上的檔案系統組態。請注意建立檔案系統後，您可以修改哪些檔案系統設定，以供參考。選擇 Create file system (建立檔案系統)。
16. 請注意檔案系統 ID。您會在稍後的步驟中需要使用。

    您可以在建立 FSx for Windows File Server 檔案系統時，繼續建立叢集和 EC2 執行個體的後續步驟。

步驟 5：建立 Amazon ECS 叢集。

使用 Amazon ECS 主控台建立叢集

1. 開啟主控台，網址為 https://console.aws.amazon.com/ecs/v2。
2. 從導覽列中選取要使用的「區域」。
3. 在導覽窗格中，選擇叢集。
4. 在 Clusters (叢集) 頁面上，選擇 Create cluster (建立叢集)。
5. 在叢集組態下的叢集名稱中，輸入 windows-fsx-cluster。
6. 展開基礎設施、clear AWS Fargate (serverless)，然後選取 Amazon EC2 執行個體。
    - 若要建立 Auto Scaling 群組，請從 Auto Scaling group (ASG) (Auto Scaling 群組 (ASG)) 中選取 Create new group (建立新群組)，然後提供有關該群組的下列詳細資訊：
        - 在作業系統/架構中，選擇 Windows Server 2019 Core。
        - 對於 EC2 執行個體類型，選擇 t2.medium 或 t2.micro。
7. 選擇 Create (建立)。

步驟 6：建立 Amazon ECS 最佳化的 Amazon EC2 執行個體

建立 Amazon ECS Windows 容器執行個體。

建立 Amazon ECS 執行個體

1. 使用 `aws ssm get-parameters` 命令擷取託管 VPC 的區域 AMI 名稱。如需詳細資訊，請參閱擷取 Amazon ECS 最佳化 AMI 中繼資料。
2. 使用 Amazon EC2 主控台來啟動執行個體。
    a. 在 https://console.aws.amazon.com/ec2/ 開啟 Amazon EC2 主控台。
    b. 從導覽列中選取要使用的「區域」。
    c. 在 EC2 儀表板中，選擇 Launch instance (啟動執行個體)。
    d. 在 Name (名稱) 輸入唯一的名稱。
    e. 針對應用程式和作業系統映像 (Amazon Machine Image)，在搜尋欄位中，輸入您擷取到的 AMI 名稱。
    f. 對於執行個體類型，選擇 t2.medium 或 t2.micro。
    g. 在 Key pair (login) (金鑰對 (登入)) 欄位中，選擇一個金鑰對。如果您未指定金鑰對，則

h. 在網路設定下，針對 VPC 和子網路，選擇您的 VPC 和公有子網路。

i. 在 Network settings (網路設定) 下的 Security group (安全群組) 欄位中，選擇一個現有的安全群組，或建立一個新的安全群組。確定您選擇的安全群組具有在 教學課程的先決條件 中定義的輸入和輸出規則

j. 在 Network settings (網路設定) 下的 Auto-assign Public IP (自動指派公有 IP) 欄位中，選取 Enable (啟用)。

k. 展開進階詳細資訊，然後針對網域加入目錄，選取您建立的 Active Directory 的 ID。啟動 EC2 執行個體時，此選項網域會加入您的 AD。

l. 在 Advanced details (進階詳細資料) 下的 IAM instance profile (IAM 執行個體設定檔) 欄位中，選擇 ecsInstanceRole。

m. 利用下列使用者資料設定您的 Amazon ECS 容器執行個體。在 Advanced details (進階詳細資料) 下的 User data (使用者資料) 欄位中，貼入下列指令碼，以您的叢集名稱取代 *cluster_name*。

```
<powershell>
Initialize-ECSAgent -Cluster windows-fsx-cluster -EnableTaskIAMRole
</powershell>
```

n. 準備就緒後，請選取 acknowledgment (確認) 欄位，再選擇 Launch Instances (啟動執行個體)。

o. 會有確認頁面讓您知道您的執行個體正在啟動。選擇 View Instances (檢視執行個體) 關閉確認頁面並返回主控台。

3. 開啟主控台，網址為 https://console.aws.amazon.com/ecs/v2。

4. 在導覽窗格中選擇叢集，然後選擇 windows-fsx-cluster。

5. 選擇基礎設施索引標籤，並確認您的執行個體已在 windows-fsx-cluster 叢集註冊。

步驟 7：註冊 Windows 任務定義

您必須先註冊任務定義，才能在您的 Amazon ECS 叢集中執行 Windows 容器。以下任務定義範例會顯示簡單的網頁。任務會啟動兩個可存取 FSx 檔案系統的容器。第一個容器會將 HTML 檔案寫入檔案系統。第二個容器從檔案系統下載 HTML 檔案並為網頁提供服務。

1. 開啟主控台，網址為 https://console.aws.amazon.com/ecs/v2。

2. 在導覽窗格中，選擇 Task Definitions (任務定義)。

3. 選擇 Create new task definitio (建立新任務定義)、Create new task definition with JSON (使用 JSON 建立新的任務定義)。
4. 在 JSON 編輯器方塊中，取代任務執行角色的值和有關 FSx 檔案系統的詳細資訊，然後選擇儲存。

```
{
    "containerDefinitions": [
        {
            "entryPoint": [
                "powershell",
                "-Command"
            ],
            "portMappings": [],
            "command": ["New-Item -Path C:\\fsx-windows-dir\\index.html -ItemType
 file -Value '<html> <head> <title>Amazon ECS Sample App</title> <style>body
 {margin-top: 40px; background-color: #333;} </style> </head><body> <div
 style=color:white;text-align:center> <h1>Amazon ECS Sample App</h1> <h2>It
 Works!</h2> <p>You are using Amazon FSx for Windows File Server file system for
 persistent container storage.</p>' -Force"],
            "cpu": 512,
            "memory": 256,
            "image": "mcr.microsoft.com/windows/servercore/iis:windowsservercore-
ltsc2019",
            "essential": false,
            "name": "container1",
            "mountPoints": [
                {
                    "sourceVolume": "fsx-windows-dir",
                    "containerPath": "C:\\fsx-windows-dir",
                    "readOnly": false
                }
            ]
        },
        {
            "entryPoint": [
                "powershell",
                "-Command"
            ],
            "portMappings": [
                {
                    "hostPort": 443,
                    "protocol": "tcp",
                    "containerPort": 80
```

```
                }
            ],
            "command": ["Remove-Item -Recurse C:\\inetpub\\wwwroot\\* -Force;
 Start-Sleep -Seconds 120; Move-Item -Path C:\\fsx-windows-dir\\index.html -
Destination C:\\inetpub\\wwwroot\\index.html -Force; C:\\ServiceMonitor.exe
 w3svc"],
            "mountPoints": [
                {
                    "sourceVolume": "fsx-windows-dir",
                    "containerPath": "C:\\fsx-windows-dir",
                    "readOnly": false
                }
            ],
            "cpu": 512,
            "memory": 256,
            "image": "mcr.microsoft.com/windows/servercore/iis:windowsservercore-
ltsc2019",
            "essential": true,
            "name": "container2"
        }
    ],
    "family": "fsx-windows",
    "executionRoleArn": "arn:aws:iam::111122223333:role/ecsTaskExecutionRole",
    "volumes": [
        {
            "name": "fsx-windows-dir",
            "fsxWindowsFileServerVolumeConfiguration": {
                "fileSystemId": "fs-0eeb5730b2EXAMPLE",
                "authorizationConfig": {
                    "domain": "example.com",
                    "credentialsParameter": "arn:arn-1234"
                },
                "rootDirectory": "share"
            }
        }
    ]
}
```

步驟 8：執行任務並檢視結果

在執行任務之前，請確認 FSx for Windows File Server 檔案系統的狀態為 Available (可用)。可用之後，您可以使用您建立的任務定義來執行任務。透過使用檔案系統建立相互之間隨機顯示 HTML 檔案的容器，以啟動任務。隨機顯示後，Web 伺服器提供簡單的 HTML 頁面。

> **Note**
>
> 您可能無法從 VPN 內連線至網站。

使用 Amazon ECS 主控台執行任務並檢視結果。

1. 開啟主控台，網址為 https://console.aws.amazon.com/ecs/v2。
2. 在導覽窗格中選擇叢集，然後選擇 windows-fsx-cluster。
3. 選擇任務索引標籤，然後選擇執行新任務。
4. 針對 Launch Type (啟動類型)，選擇 EC2。
5. 在部署組態下，針對任務定義選擇 fsx-windows，然後選擇建立。
6. 當您的任務狀態為執行中時，請選擇任務 ID。
7. 在容器下，當容器 1 狀態為已停止時，選取容器 2 以檢視容器的詳細資訊。
8. 在 container2 的容器詳細資訊下，選取網路繫結，然後按一下與容器關聯的外部 IP 地址。您的瀏覽器將會開啟並顯示以下訊息。

```
Amazon ECS Sample App
It Works!
You are using Amazon FSx for Windows File Server file system for persistent
 container storage.
```

> **Note**
>
> 可能需要幾分鐘的時間才會顯示訊息。如果幾分鐘後您未看到此訊息，請檢查您是否未在 VPN 中執行，並確認容器執行個體的安全群組，允許連接埠 443 上的傳入網路 HTTP 流量。

步驟 9：清除

> **Note**
>
> 刪除 FSx for Windows File Server 檔案系統或 AD 需要 20 到 45 分鐘的時間。您必須等到 FSx for Windows File Server 檔案系統刪除操作完成後，才能開始 AD 刪除操作。

刪除 FSx for Windows File Server 檔案系統。

1. 開啟 Amazon FSx 主控台
2. 選取您剛建立之 FSx for Windows File Server 檔案系統左側的選項按鈕。
3. 選擇動作。
4. 選取 Delete file system (刪除檔案系統)。

刪除 AD。

1. 開啟 AWS Directory Service 主控台。
2. 選擇您剛建立之 AD 左側的選項按鈕。
3. 選擇動作。
4. 選擇 Delete directory (刪除目錄)。

刪除叢集。

1. 開啟主控台，網址為 https://console.aws.amazon.com/ecs/v2。
2. 在導覽窗格中選擇叢集，然後選擇 fsx-windows-cluster。
3. 選擇 Delete cluster (刪除叢集)。
4. 輸入短語，然後選擇刪除。

終止 EC2 執行個體。

1. 開啟 Amazon EC2 主控台。
2. 從左側選單中，選取 Instances (執行個體)。
3. 勾選您建立之 EC2 執行個體左側的核取方塊。

4. 按一下執行個體狀態和終止執行個體。

刪除密碼。

1. 開啟 Secrets Manager 主控台。
2. 選取您為此逐步解說所建立的機密。
3. 按一下 Actions (動作)。
4. 選擇 Delete secret (刪除機密)。

搭配 Amazon ECS 使用 Docker 磁碟區

使用 Docker 磁碟區時，可以使用內建的 `local` 驅動程式或第三方磁碟區驅動程式。Docker 磁碟區是由 Docker 管理，並且會在容器執行個體上的 `/var/lib/docker/volumes` 中建立一個目錄，其中包含磁碟區資料。

若要使用 Docker 磁碟區，請在您的任務定義中指定 `dockerVolumeConfiguration`。如需詳細資訊，請參閱 Docker 文件中的磁碟區。

Docker 磁碟區的一些常用案例如下：

- 提供為與容器搭配使用的持久性資料磁碟區
- 在相同的容器執行個體上不同容器的不同位置共用定義的資料磁碟區
- 定義空的非持久性資料磁碟區，並將它掛載在相同任務內的多個容器上
- 提供資料磁碟區給由第三方驅動程式所管理的任務

使用 Docker 磁碟區的考量事項

使用 Docker 磁碟區時，請考量下列事項：

- 只有在使用 EC2 啟動類型或外部執行個體時才支援 Docker 磁碟區。
- Windows 容器只支援使用 `local` 驅動程式。
- 如果使用第三方驅動程式，確保先在容器執行個體上安裝及使用它，容器代理程式才會啟動。如果在啟動代理程式之前沒有啟動第三方驅動程式，您可以使用下列命令之一重新啟動容器代理程式：
  - 對於 Amazon ECS 最佳化 Amazon Linux 2 AMI：

    ```
    sudo systemctl restart ecs
    ```

- 對於 Amazon ECS 最佳化 Amazon Linux AMI：

```
sudo stop ecs && sudo start ecs
```

在 Amazon ECS 任務定義中指定 Docker 磁碟區

您必須先在任務定義中指定磁碟區和掛載點組態，您的容器才可以使用資料磁碟區。本節說明容器的磁碟區組態。對於使用 Docker 磁碟區的任務，指定 `dockerVolumeConfiguration`。對於使用綁定掛載主機磁碟區的任務，指定 `host` 和選用的 `sourcePath`。

以下任務定義 JSON 說明容器 `volumes` 和 `mountPoints` 物件的語法。

```
{
    "containerDefinitions": [
        {
            "mountPoints": [
                {
                    "sourceVolume": "string",
                    "containerPath": "/path/to/mount_volume",
                    "readOnly": boolean
                }
            ]
        }
    ],
    "volumes": [
        {
            "name": "string",
            "dockerVolumeConfiguration": {
                "scope": "string",
                "autoprovision": boolean,
                "driver": "string",
                "driverOpts": {
                    "key": "value"
                },
                "labels": {
                    "key": "value"
                }
            }
        }
    ]
}
```

name

類型：字串

必要：否

磁碟區名稱。最多允許 255 個字母 （ 大寫和小寫 ）、數字、連字號 (-) 和底線 (_)。容器定義mountPoints物件的 sourceVolume 參數中會參考此名稱。

dockerVolumeConfiguration

類型：DockerVolumeConfiguration 物件

必要：否

此參數只有使用 Docker 磁碟區時才會指定。只有在 EC2 執行個體上執行任務時，才支援 Docker 磁碟區。Windows 容器僅支援使用local驅動程式。若要使用綁定掛載，請指定 host。

scope

類型：字串

有效值: task | shared

必要：否

決定生命週期的 Docker 磁碟區範圍。範圍受限於 task 的 Docker 磁碟區，會在任務啟動時自動佈建，以及在任務停止時銷毀。範圍為 shared 的 Docker 磁碟區會在任務停止之後保留。

autoprovision

類型：布林值

預設值：false

必要：否

若此數值為 true，Docker 磁碟區便得以建立 (若它尚不存在)。只有在 scope為 時，才會使用此欄位shared。如果 scope是 task，則必須省略此參數。

driver

類型：字串

必要：否

要使用的 Docker 磁碟區驅動程式。驅動程式值必須符合 Docker 提供的驅動程式名稱，因為此名稱用於任務置放。如果驅動程式是使用 Docker 外掛程式 CLI 安裝，請使用 從您的容器執行個體`docker plugin ls`擷取驅動程式名稱。如果驅動程式是透過使用其他方法安裝，請使用 Docker 外掛程式探索來擷取驅動程式名稱。

`driverOpts`

類型：字串

必要：否

要傳遞的 Docker 驅動程式特定選項映射。此參數在 Docker 的建立磁碟區區段`DriverOpts`中映射至 。

`labels`

類型：字串

必要：否

自訂中繼資料，新增到您的 Docker 磁碟區。

`mountPoints`

類型：物件陣列

必要：否

容器中資料磁碟區的掛載點。此參數會映射至 create-container Docker API `Volumes`中的 ，以及 Docker 執行`--volume`的選項。

Windows 容器可在 `$env:ProgramData` 所在的相同磁碟上掛載整個目錄。Windows 容器無法在不同的磁碟機上掛載目錄，而且掛載點無法跨磁碟機使用。您必須指定掛載點，才能將 Amazon EBS 磁碟區直接連接至 Amazon ECS 任務。

`sourceVolume`

類型：字串

必要：是 (當使用 `mountPoints` 時)

要掛載的磁碟區名稱。

`containerPath`

類型：字串

必要：是 (當使用 mountPoints 時)

要掛載磁碟區的容器中的路徑。

`readOnly`

類型：布林值

必要：否

如果此數值為 `true`，容器擁有磁碟區的唯讀存取權。如果此值為 `false`，則容器可寫入磁碟區。預設值為 `false`。

對於在執行 Windows 作業系統的 EC2 執行個體上執行的任務，請將 值保留為預設值 `false`。

Docker 磁碟區範例

使用 Docker 磁碟區為容器提供暫時性儲存

在此範例中，容器使用任務完成後清除的空白資料磁碟區。舉一個使用案例範例，您的容器在任務期間需要存取一些暫存檔案儲存位置。使用 Docker 磁碟區可達成此任務。

1. 在任務定義 `volumes` 區段，以 `name` 和 `DockerVolumeConfiguration` 值定義資料磁碟區。在本範例中，我們指定範圍為 `task`，所以會在任務停止後刪除磁碟區，並使用內建的 `local` 驅動程式。

```
"volumes": [
    {
        "name": "scratch",
        "dockerVolumeConfiguration" : {
            "scope": "task",
            "driver": "local",
            "labels": {
                "scratch": "space"
            }
        }
    }
]
```

2. 在 `containerDefinitions` 區段中定義容器，並使用 `mountPoints` 值來參考已定義的磁碟區名稱，使用 `containerPath` 值將磁碟區掛載於容器。

```
"containerDefinitions": [
```

```
    {
        "name": "container-1",
        "mountPoints": [
            {
              "sourceVolume": "scratch",
              "containerPath": "/var/scratch"
            }
        ]
    }
]
```

使用 Docker 磁碟區提供容器的持久性儲存

在此範例中，您要多個容器使用一個共用磁碟區，並且要在使用它的任何單一任務停止之後加以保留。正在使用內建的 `local` 驅動程式。如此一來，磁碟區仍受限於容器執行個體的生命週期。

1. 在任務定義 `volumes` 區段，以 `name` 和 `DockerVolumeConfiguration` 值定義資料磁碟區。在此範例中，指定 `shared` 範圍，以便持續保留磁碟區，請將 autoprovision 設定為 `true`。如此一來，已建立該磁碟區以供使用。然後，還可以使用內建 `local` 驅動程式。

```
"volumes": [
    {
        "name": "database",
        "dockerVolumeConfiguration" : {
            "scope": "shared",
            "autoprovision": true,
            "driver": "local",
            "labels": {
                "database": "database_name"
            }
        }
    }
]
```

2. 在 `containerDefinitions` 區段中定義容器，並使用 `mountPoints` 值來參考已定義的磁碟區名稱，使用 `containerPath` 值將磁碟區掛載於容器。

```
"containerDefinitions": [
    {
        "name": "container-1",
        "mountPoints": [
```

```
        {
          "sourceVolume": "database",
          "containerPath": "/var/database"
        }
      ]
    },
    {
      "name": "container-2",
      "mountPoints": [
        {
          "sourceVolume": "database",
          "containerPath": "/var/database"
        }
      ]
    }
  ]
```

使用 Docker 磁碟區提供容器的 NFS 持久性儲存

在此範例中，容器使用 NFS 資料磁碟區，該資料磁碟區在任務啟動時自動掛載並在任務停止時自動卸載。這可使用 Docker 內建 `local` 驅動程式加以實現。一個範例使用案例是，您可能有本機 NFS 儲存體，並且需要從 ECS Anywhere 任務進行存取。這可以使用具有 NFS 驅動程式選項的 Docker 磁碟區來達成。

1. 在任務定義 `volumes` 區段，以 `name` 和 `DockerVolumeConfiguration` 值定義資料磁碟區。在此範例中，指定 `task` 範圍，以便在任務停止後卸載磁碟區。使用 `local` 驅動程式並相應地使用 `type`、`device` 和 `o` 選項設定 `driverOpts`。使用 NFS 伺服器端點取代 `NFS_SERVER`。

```
"volumes": [
       {
           "name": "NFS",
           "dockerVolumeConfiguration" : {
               "scope": "task",
               "driver": "local",
               "driverOpts": {
                   "type": "nfs",
                   "device": "$NFS_SERVER:/mnt/nfs",
                   "o": "addr=$NFS_SERVER"
               }
           }
       }
```

```
    ]
```

2. 在 containerDefinitions 區段中定義容器，並使用 mountPoints 值來參考已定義的磁碟區名稱，使用 containerPath 值將磁碟區掛載在容器上。

```
"containerDefinitions": [
       {
           "name": "container-1",
           "mountPoints": [
               {
                 "sourceVolume": "NFS",
                 "containerPath": "/var/nfsmount"
               }
           ]
       }
   ]
```

搭配 Amazon ECS 使用繫結掛載

透過繫結掛載，主機上的檔案或目錄，例如 Amazon EC2 執行個體，會掛載到容器中。在 Fargate 和 Amazon EC2 執行個體上託管的任務支援綁定掛載。繫結掛載與使用它們的容器的生命週期相關聯。使用綁定掛載的所有容器停止之後，例如當任務停止時，資料就會被移除。對於託管在 Amazon EC2 執行個體上的任務，資料可以透過在任務定義中指定 host和選用sourcePath值來繫結至主機 Amazon EC2 執行個體的生命週期。如需詳細資訊，請參閱 Docker 文件中的繫結掛載。

以下是綁定掛載的常用案例。

- 若要提供空白的資料磁碟區以便在一個或多個容器中掛載。
- 若要在一個或多個容器中掛載主機資料磁碟區。
- 若要與相同任務中的其他容器共用來源容器的資料磁碟區。
- 若要將 Dockerfile 中的路徑及其內容公開給一個或多個容器。

使用綁定掛載時的考量

使用綁定掛載時，請考量下列事項。

- 根據預設， AWS Fargate 使用平台版本 1.4.0 或更新版本 (Linux) 1.0.0或更新版本 (Windows) 在 上託管的任務，會為繫結掛載接收至少 20 GiB 的暫時性儲存。您可以透過在任務定義中指定 ephemeralStorage 參數，將暫時性儲存的總量增加到最高 200 GiB。

- 若要在執行任務時將 Dockerfile 中的檔案公開至資料磁碟區，Amazon ECS 資料平面會尋找 VOLUME 指令。如果在 VOLUME 指令中指定的絕對路徑與在任務定義中指定的 containerPath 相同，則 VOLUME 指令路徑中的資料會複製到資料磁碟區。在下列 Dockerfile 範例中，/var/log/exported 目錄中名為 examplefile 的檔案會寫入主機，然後掛載在容器內。

```
FROM public.ecr.aws/amazonlinux/amazonlinux:latest
RUN mkdir -p /var/log/exported
RUN touch /var/log/exported/examplefile
VOLUME ["/var/log/exported"]
```

  根據預設，磁碟區許可設定為 0755 和擁有者設定為 root。您可以在 Dockerfile 中自訂這些許可。在下列範例中，將目錄的擁有者設定為 node。

```
FROM public.ecr.aws/amazonlinux/amazonlinux:latest
RUN yum install -y shadow-utils && yum clean all
RUN useradd node
RUN mkdir -p /var/log/exported && chown node:node /var/log/exported
RUN touch /var/log/exported/examplefile
USER node
VOLUME ["/var/log/exported"]
```

- 對於在 Amazon EC2 執行個體上託管的任務，當 host 和 sourcePath 值未指定時，Docker 常駐程式會為您管理綁定掛載。當沒有任何容器參考此綁定掛載時，Amazon ECS 容器代理程式任務清除服務最終會予以刪除。根據預設，這會在容器退出的 3 小時後發生。不過，您可以使用 ECS_ENGINE_TASK_CLEANUP_WAIT_DURATION 代理程式變數設定此持續時間。如需詳細資訊，請參閱Amazon ECS 容器代理程式組態。如果您需要此資料的保留時間超過容器的生命週期，請為綁定掛載指定 sourcePath 值。

在 Amazon ECS 任務定義中指定繫結掛載

對於託管在 Fargate 或 Amazon EC2 執行個體上的 Amazon ECS 任務，下列任務定義 JSON 程式碼片段會顯示任務定義的 volumes、 mountPoints和 ephemeralStorage 物件的語法。

```
{
   "family": "",
   ...
   "containerDefinitions" : [
      {
         "mountPoints" : [
            {
```

```
                "containerPath" : "/path/to/mount_volume",
                "sourceVolume" : "string"
            }
          ],
          "name" : "string"
       }
    ],
    ...
    "volumes" : [
       {
          "name" : "string"
       }
    ],
    "ephemeralStorage": {
    "sizeInGiB": integer
    }
}
```

對於託管於 Amazon EC2 執行個體上的 Amazon ECS 任務，您可以在指定任務磁碟區詳細資訊時使用選用的 host 參數和 sourcePath。指定時，它會將綁定掛載任務的生命週期，而不是容器。

```
"volumes" : [
    {
        "host" : {
            "sourcePath" : "string"
        },
        "name" : "string"
    }
]
```

下面會更詳細地描述每個任務定義參數。

name

類型：字串

必要：否

磁碟區名稱。最多允許 255 個字母（大寫和小寫）、數字、連字號 (-) 和底線 (_)。容器定義mountPoints物件的 sourceVolume 參數中會參考此名稱。

host

必要：否

host 參數用於將綁定掛載的生命週期綁定到主機 Amazon EC2 執行個體，而非任務，以及它的儲存位置。如果 host 參數是空的，則 Docker 常駐程式會為您的資料磁碟區指派主機路徑，但其相關聯的容器停止執行後，不保證會保留資料。

Windows 容器可在 $env:ProgramData 所在的相同磁碟上掛載整個目錄。

> **Note**
>
> 只有在使用託管在 Amazon EC2 執行個體上的任務時，才支援 sourcePath 參數。

sourcePath

類型：字串

必要：否

使用 host 參數時，指定 sourcePath 以宣告在主機 Amazon EC2 執行個體上提供給容器的路徑。如果此參數是空的，則 Docker 常駐程式會為您指派主機路徑。如果 host 參數包含 sourcePath 檔案位置，資料磁碟區將保留在主機 Amazon EC2 執行個體上的指定位置，直到您手動將其刪除為止。如果 sourcePath 值不存在於主機 Amazon EC2 執行個體，Docker 常駐程式將建立該值。如果位置存在，將匯出來源路徑資料夾的內容。

mountPoints

類型：物件陣列

必要：否

容器中資料磁碟區的掛載點。此參數會映射至 create-container Docker API Volumes中的 ，以及 Docker 執行--volume的選項。

Windows 容器可在 $env:ProgramData 所在的相同磁碟上掛載整個目錄。Windows 容器無法在不同的磁碟機上掛載目錄，而且掛載點無法跨磁碟機使用。您必須指定掛載點，才能將 Amazon EBS 磁碟區直接連接至 Amazon ECS 任務。

sourceVolume

類型：字串

必要：是 (當使用 mountPoints 時)

要掛載的磁碟區名稱。

`containerPath`

類型：字串

必要：是 (當使用 `mountPoints` 時)

要掛載磁碟區的容器中的路徑。

`readOnly`

類型：布林值

必要：否

如果此數值為 `true`，容器擁有磁碟區的唯讀存取權。如果此值為 `false`，則容器可寫入磁碟區。預設值為 `false`。

對於在執行 Windows 作業系統的 EC2 執行個體上執行的任務，請將 值保留為預設值 `false`。

`ephemeralStorage`

類型：物件

必要：否

為任務配置的暫時性儲存量。對於 AWS Fargate 使用平台版本 或更新版本 (Linux) `1.4.0`或更新版本 (`1.0.0`Windows) 託管的任務，此參數用於擴展可用的暫時性儲存總量，超過預設數量。

您可以使用 Copilot CLI、CloudFormation、 AWS SDK 或 CLI 來指定繫結掛載的暫時性儲存。

綁定掛載範例

下列範例涵蓋使用容器繫結掛載的常見使用案例。

若要為 Fargate 任務分配更多的暫時性儲存量

對於託管於使用平台版本 `1.4.0` 或更新版本 (Linux) 或者 `1.0.0` (Windows) 的 Fargate 上的 Amazon ECS 任務，您可以為任務中的容器分配比預設暫時性儲存量更多的空間以供使用。此範例可以整合到其他範例中，為您的 Fargate 任務分配更多的暫時性儲存。

- 在任務定義中，定義 `ephemeralStorage` 物件。`sizeInGiB` 必須是 `21` 和 `200` 之間的整數並以 GiB 為單位來表示。

```
"ephemeralStorage": {
    "sizeInGiB": integer
```

```
}
```

若要為一個或多個容器提供空白的資料磁碟區

在某些情況下，您希望在任務中為容器提供一些暫存空間。例如，您可能有兩個資料庫容器，在任務期間需要存取相同的暫存檔案儲存位置。這可以使用綁定掛載來達成。

1. 在任務定義 volumes 區段中，以名稱 database_scratch 定義綁定掛載。

```
  "volumes": [
    {
      "name": "database_scratch"
    }
  ]
```

2. 在 containerDefinitions 區段中，建立資料庫容器定義。如此一來，它們就可以掛載磁碟區。

```
"containerDefinitions": [
    {
      "name": "database1",
      "image": "my-repo/database",
      "cpu": 100,
      "memory": 100,
      "essential": true,
      "mountPoints": [
        {
          "sourceVolume": "database_scratch",
          "containerPath": "/var/scratch"
        }
      ]
    },
    {
      "name": "database2",
      "image": "my-repo/database",
      "cpu": 100,
      "memory": 100,
      "essential": true,
      "mountPoints": [
        {
          "sourceVolume": "database_scratch",
          "containerPath": "/var/scratch"
```

```
        }
      ]
    }
  ]
```

若要將 Docker 文件中的路徑及其內容公開給容器

在此範例中，您有一個 Dockerfile，它會寫入您想要掛載在容器內的資料。此範例適用於 Fargate 或 Amazon EC2 執行個體上託管的任務。

1. 建立 Dockerfile。下列範例使用公有 Amazon Linux 2 容器映像，並在我們想要在容器內部掛載的 `/var/log/exported` 目錄中建立名為 `examplefile` 的檔案。`VOLUME` 指令應該指定一個絕對路徑。

   ```
   FROM public.ecr.aws/amazonlinux/amazonlinux:latest
   RUN mkdir -p /var/log/exported
   RUN touch /var/log/exported/examplefile
   VOLUME ["/var/log/exported"]
   ```

   根據預設，磁碟區許可設定為 `0755` 和擁有者設定為 `root`。可以在 Docker 檔案中變更這些許可。在下列範例中，`/var/log/exported` 目錄的擁有者設定為 `node`。

   ```
   FROM public.ecr.aws/amazonlinux/amazonlinux:latest
   RUN yum install -y shadow-utils && yum clean all
   RUN useradd node
   RUN mkdir -p /var/log/exported && chown node:node /var/log/exported
   USER node
   RUN touch /var/log/exported/examplefile
   VOLUME ["/var/log/exported"]
   ```

2. 在任務定義 `volumes` 區段中，以名稱 `application_logs` 定義一個磁碟區。

   ```
   "volumes": [
     {
       "name": "application_logs"
     }
   ]
   ```

3. 在 `containerDefinitions` 區段中，建立應用程式容器定義。如此一來，它們就可以掛載儲存。`containerPath` 值必須符合 Dockerfile `VOLUME` 指令中指定的絕對路徑。

```
"containerDefinitions": [
  {
    "name": "application1",
    "image": "my-repo/application",
    "cpu": 100,
    "memory": 100,
    "essential": true,
    "mountPoints": [
      {
        "sourceVolume": "application_logs",
        "containerPath": "/var/log/exported"
      }
    ]
  },
  {
    "name": "application2",
    "image": "my-repo/application",
    "cpu": 100,
    "memory": 100,
    "essential": true,
    "mountPoints": [
      {
        "sourceVolume": "application_logs",
        "containerPath": "/var/log/exported"
      }
    ]
  }
]
```

若要為與主機 Amazon EC2 執行個體生命週期相關聯的容器提供空資料磁碟區

對於在 Amazon EC2 執行個體上託管的任務，您可以使用綁定掛載，並將資料與主機 Amazon EC2 執行個體的生命週期相關聯。您可以使用 `host` 參數並指定 `sourcePath` 值來達成此操作。位在 `sourcePath` 的任何檔案都會出現在值為 `containerPath` 的容器中。寫入 `containerPath` 值的任何檔案都會寫入主機 Amazon EC2 執行個體上的 `sourcePath` 值。

> **Important**
>
> Amazon ECS 不會在 Amazon EC2 執行個體之間同步您的儲存。使用持久性儲存的任務可以放置在您有可用容量之叢集中的任何 Amazon EC2 執行個體。如果您的任務在停止和重新啟動

後需要持久性儲存，請務必在任務啟動時間使用 AWS CLI start-task 命令指定相同的 Amazon EC2 執行個體。也可使用 Amazon EFS 磁碟區以供持久性儲存。如需詳細資訊，請參閱搭配 Amazon ECS 使用 Amazon EFS 磁碟區。

1. 在任務定義 volumes 區段，以 name 和 sourcePath 值定義綁定掛載。在下列範例中，主機 Amazon EC2 執行個體包含要掛載在容器內的 /ecs/webdata 的資料。

```
"volumes": [
  {
    "name": "webdata",
    "host": {
      "sourcePath": "/ecs/webdata"
    }
  }
]
```

2. 在 containerDefinitions 區段中定義容器，並使用 mountPoints 值來參考綁定掛載名稱，使用 containerPath 值在容器上掛載綁定掛載。

```
"containerDefinitions": [
  {
    "name": "web",
    "image": "nginx",
    "cpu": 99,
    "memory": 100,
    "portMappings": [
      {
        "containerPort": 80,
        "hostPort": 80
      }
    ],
    "essential": true,
    "mountPoints": [
      {
        "sourceVolume": "webdata",
        "containerPath": "/usr/share/nginx/html"
      }
    ]
  }
]
```

在不同位置的多個容器中掛載已定義的磁碟區

您可以在任務定義中定義資料磁碟區，並將該磁碟區掛載到不同容器的不同位置。例如，您的主機容器有一個位於 /data/webroot 的網站資料夾。建議您以唯讀方式將該資料磁碟區掛載到有不同文件根目錄的兩個不同 Web 伺服器上。

1. 在任務定義 volumes 區段中，以名稱 webroot 和來源路徑 /data/webroot 定義資料磁碟區。

```
"volumes": [
  {
    "name": "webroot",
    "host": {
      "sourcePath": "/data/webroot"
    }
  }
]
```

2. 在 containerDefinitions 區段中，使用 mountPoints 值為每個 Web 伺服器定義容器，這些值會建立 webroot 磁碟區與指向該容器文件根目錄之 containerPath 值的關聯性。

```
"containerDefinitions": [
  {
    "name": "web-server-1",
    "image": "my-repo/ubuntu-apache",
    "cpu": 100,
    "memory": 100,
    "portMappings": [
      {
        "containerPort": 80,
        "hostPort": 80
      }
    ],
    "essential": true,
    "mountPoints": [
      {
        "sourceVolume": "webroot",
        "containerPath": "/var/www/html",
        "readOnly": true
      }
    ]
  },
  {
    "name": "web-server-2",
```

```
      "image": "my-repo/sles11-apache",
      "cpu": 100,
      "memory": 100,
      "portMappings": [
        {
          "containerPort": 8080,
          "hostPort": 8080
        }
      ],
      "essential": true,
      "mountPoints": [
        {
          "sourceVolume": "webroot",
          "containerPath": "/srv/www/htdocs",
          "readOnly": true
        }
      ]
    }
  ]
```

使用 **volumesFrom** 掛載來自其他容器的磁碟區

針對在 Amazon EC2 執行個體上託管的任務，您可以在容器上定義一或多個磁碟區，然後在相同的任務中的不同的容器定義中使用 volumesFrom 參數，將所有來自 sourceContainer 的磁碟區掛載在其原始定義的掛載點。volumesFrom 參數適用於在任務定義中定義的磁碟區，以及使用 Dockerfile 內建在映像中的磁碟區。

1. (選用) 若要共用內建在映像的磁碟區，請使用 Dockerfile 中的 VOLUME 指令。以下範例 Dockerfile 使用 httpd 映像，然後新增磁碟區，再將之掛載到 Apache 文件根中的 dockerfile_volume。該資料夾由 httpd Web 伺服器使用。

   ```
   FROM httpd
   VOLUME ["/usr/local/apache2/htdocs/dockerfile_volume"]
   ```

   您可以使用此 Dockerfile 建立映像，並將之推送到儲存庫，例如 Docker Hub，然後在您的任務定義中使用。下列步驟中使用的範例my-repo/httpd_dockerfile_volume映像是使用上述 Dockerfile 建置。
2. 建立任務定義，定義您容器的其他磁碟區及掛載點。在這個範例 volumes 區段中，您要建立一個名為 empty 的空磁碟區，它是由 Docker 常駐程式管理。您還要定義一個稱為 host_etc 的主機磁碟區。它會匯出主機容器執行個體的 /etc 資料夾。

```
{
  "family": "test-volumes-from",
  "volumes": [
    {
      "name": "empty",
      "host": {}
    },
    {
      "name": "host_etc",
      "host": {
        "sourcePath": "/etc"
      }
    }
  ],
```

在容器定義區段中，建立一個容器，掛載之前定義的磁碟區。在此範例中，web 容器掛載 empty 和 host_etc 磁碟區。這是使用由 Dockerfile 中的磁碟區建立的映像的容器。

```
"containerDefinitions": [
    {
      "name": "web",
      "image": "my-repo/httpd_dockerfile_volume",
      "cpu": 100,
      "memory": 500,
      "portMappings": [
        {
          "containerPort": 80,
          "hostPort": 80
        }
      ],
      "mountPoints": [
        {
          "sourceVolume": "empty",
          "containerPath": "/usr/local/apache2/htdocs/empty_volume"
        },
        {
          "sourceVolume": "host_etc",
          "containerPath": "/usr/local/apache2/htdocs/host_etc"
        }
      ],
      "essential": true
```

```
    },
```

建立另一個容器，使用 volumesFrom 掛載與 web 容器相關聯的所有磁碟區。web 容器上的所有磁碟區同樣掛載在 busybox 容器上。這包括在 Dockerfile 中指定的磁碟區，該磁碟區用於建置 my-repo/httpd_dockerfile_volume 映像。

```
    {
      "name": "busybox",
      "image": "busybox",
      "volumesFrom": [
        {
          "sourceContainer": "web"
        }
      ],
      "cpu": 100,
      "memory": 500,
      "entryPoint": [
        "sh",
        "-c"
      ],
      "command": [
        "echo $(date) > /usr/local/apache2/htdocs/empty_volume/date && echo $(date)
 > /usr/local/apache2/htdocs/host_etc/date && echo $(date) > /usr/local/apache2/
htdocs/dockerfile_volume/date"
      ],
      "essential": false
    }
  ]
}
```

執行此任務時，兩個容器會掛載磁碟區，而 busybox 容器中的 command 會將日期和時間寫入檔案。這個檔案在每個磁碟區資料夾中稱為 date。然後，在 web 容器所顯示的網站上，就能看到這些資料夾。

> **Note**
>
> 由於 busybox 容器執行快速命令，然後結束，所以在容器定義中必須設為 "essential": false。否則，它結束時會停止整個任務。

管理 Amazon ECS 上的容器交換記憶體空間

透過 Amazon ECS，您可在容器層級控制 Linux Amazon EC2 執行個體上的交換記憶體空間用量。使用每個容器交換組態，任務定義內的每個容器都可以啟用或停用交換。對於已啟用交換的容器，可以限制所使用的最大交換空間量。例如，對延遲要求嚴格的容器可以停用交換。相對的，具有高暫時性記憶體需求的容器可以開啟交換，以降低容器在處於低負載時發生記憶體不足錯誤的機會。

容器的交換組態由下列容器定義參數管理：

`maxSwap`

容器可以使用的交換記憶體總量 (以 MiB 為單位)。此參數會轉譯為 docker 執行`--memory-swap`的選項，其中值是容器記憶體加`maxSwap`值的總和。

如果將 `maxSwap` 值指定為 `0`，容器不會使用交換。接受的值為 `0` 或任何正整數。如果省略 `maxSwap` 參數，容器使用其執行所在的容器執行個體的交換組態。必須設定 `maxSwap` 值，才能使用 `swappiness` 參數。

`swappiness`

您可藉此調整容器的記憶體交換行為。`swappiness` 的值若為 `0` 將導致交換不會發生 (除非有需要)。為 `100` 的 `swappiness` 值導致積極地交換頁面。接受的值為介於 `0` 與 `100` 之間的整數。如果未指定 `swappiness` 參數，則會使用預設值 `60`。如果未對 `maxSwap` 指定值，則會忽略此參數。此參數會映射至 `--memory-swappiness`選項，以執行 Docker。

在以下範例中，提供了 JSON 語法。

```
"containerDefinitions": [{
        ...
        "linuxParameters": {
            "maxSwap": integer,
            "swappiness": integer
        },
        ...
}]
```

考量事項

當您使用每個容器交換組態時，請考量下列事項。

- 必須在託管任務的 Amazon EC2 執行個體上啟用和配置交換空間，容器才能使用這些空間。根據預設，Amazon ECS 最佳化 AMI 沒有啟用交換功能。您必須在執行個體上啟用交換，才能使用此功能。如需詳細資訊，請參閱《Amazon EC2 使用者指南》中的執行個體存放區交換磁碟區，或如何配置記憶體以做為 Amazon EC2 執行個體中的交換空間？
- 只有指定 EC2 啟動類型的任務定義才支援交換空間容器定義參數。僅供 Fargate 上的 Amazon ECS 使用的任務定義不支援這些參數。
- 只有 Linux 容器才支援此功能。目前不支援 Windows 容器。
- 如果從任務定義中省略 `maxSwap` 和 `swappiness` 容器定義參數，每個容器都有一個為 `60` 的預設 `swappiness` 值。此外，總交換用量限制為容器記憶體的兩倍。
- 如果您在 Amazon Linux 2023 上使用任務，則不支援 `swappiness` 參數。

Fargate 啟動類型的 Amazon ECS 任務定義差異

若要使用 Fargate，您必須將任務定義設定為使用 Fargate 啟動類型。使用 Fargate 時有其他考量。

任務定義參數

使用 Fargate 啟動類型的任務不支援所有可用的 Amazon ECS 任務定義參數。有些參數完全不予以支援，而其他參數對 Fargate 任務會有不同的行為。

下列任務定義參數在 Fargate 任務中無效：

- `disableNetworking`
- `dnsSearchDomains`
- `dnsServers`
- `dockerSecurityOptions`
- `extraHosts`
- `gpu`
- `ipcMode`
- `links`
- `placementConstraints`
- `privileged`
- `maxSwap`
- `swappiness`

下列任務定義參數在 Fargate 任務中有效，但應注意其限制：

- `linuxParameters` - 當指定套用於容器的特定於 Linux 的選項時，對於 `capabilities`，您可以新增的唯一功能是 `CAP_SYS_PTRACE`。不支援 `devices`、`sharedMemorySize` 和 `tmpfs` 參數。如需詳細資訊，請參閱Linux 參數。
- `volumes` - Fargate 任務僅支援繫結掛載主機磁碟區，所以不支援 `dockerVolumeConfiguration` 參數。如需詳細資訊，請參閱磁碟區。
- `cpu` - 對於 AWS Fargate 上的 Windows 容器，值不可少於 1 個 vCPU。
- `networkConfiguration` - Fargate 任務一律使用`awsvpc`網路模式。

為了確保您的任務定義通過驗證可與 Fargate 搭配使用，您可以在登錄任務定義時指定下列項目：

- 在 中 AWS Management Console，針對需要相容性欄位，指定 `FARGATE`。
- 在 中 AWS CLI，指定 `--requires-compatibilities`選項。
- 在 Amazon ECS API 中，指定 `requiresCompatibilities` 標記。

作業系統和架構

當您為 AWS Fargate 設定任務和容器定義，則必須指定容器執行的作業系統。AWS Fargate 支援以下作業系統：

- Amazon Linux 2

> **Note**
>
> Linux 容器只會使用主機作業系統的核心和核心組態。例如，核心組態包括 `sysctl` 系統控制項。Linux 容器映像可以從包含任何 Linux 發行版本的檔案和程式的基礎映像製作。如果 CPU 架構相符，您可以從任何作業系統上的任何 Linux 容器映像執行容器。

- Windows Server 2019 Full
- Windows Server 2019 Core
- Windows Server 2022 Full
- Windows Server 2022 Core

當您在 AWS Fargate 上執行 Windows 容器時，必須具有 X86_64 CPU 架構。

當您在 AWS Fargate 上執行 Linux 容器時，對於基於 ARM 的應用程式，可以使用 X86_64 CPU 架構或 ARM64 架構。如需詳細資訊，請參閱the section called “64 位元 ARM 工作負載的任務定義”。

任務 CPU 和記憶體

AWS Fargate 的 Amazon ECS 任務定義需要您指定任務層級的 CPU 和記憶體。雖然您也可以指定 Fargate 任務之容器層級的 CPU 和記憶體，但這是選用的。只要在任務層級指定這些資源，就可以滿足大多數使用案例。下表顯示有效的任務層級 CPU 和記憶體組合。您可以在任務定義中將記憶體值指定為 MiB 或 GB 的字串。例如，您可以指定 MiB `3072` 或 GB `3 GB`的記憶體值。您可以將 JSON 檔案中的 CPU 值指定為 CPU 單位或虛擬 CPUsvCPUs) 中的字串。例如，您可以在 CPU 單位或 vCPUs1 `vCPU`中指定 `1024` CPU 值。

CPU 數值	記憶體數值	AWS Fargate 支援的作業系統
256 (.25 vCPU)	512 MiB、1 GB、2 GB	Linux
512 (.5 vCPU)	1 GB、2 GB、3 GB、4 GB	Linux
1024 (1 vCPU)	2 GB、3 GB、4 GB、5 GB、6 GB、7 GB、8 GB	Linux、Windows
2048 (2 vCPU)	介於 4 GB 與 16 GB 之間，以 1 GB 為單位遞增	Linux、Windows
4096 (4 vCPU)	介於 8 GB 與 30 GB 之間，以 1 GB 為單位遞增	Linux、Windows
8192 (8 vCPU) **Note** 此選項需要 Linux 平台 `1.4.0` 或更新版本。	介於 16 GB 與 60 GB 之間，以 4 GB 為單位遞增	Linux
16384 (16vCPU)	介於 32 GB 與 120 GB 之間，以 8 GB 為單位遞增	Linux

CPU 數值	記憶體數值	AWS Fargate 支援的作業系統

> **Note**
>
> 此選項需要 Linux 平台 `1.4.0` 或更新版本。

任務聯網

AWS Fargate 的 Amazon ECS 任務需要 `awsvpc` 網路模式，該網路模式會為每個任務提供彈性網絡介面。當您使用此網路模式執行任務或建立服務時，必須指定一或多個子網路來連接網路界面，以及指定一或多個安全群組來套用至網路界面。

如果使用公有子網路，請決定是否提供網路界面的公有 IP 地址。若要讓公有子網路中的 Fargate 任務提取容器映像，需要透過路由至網際網路，或可將請求路由至網際網路的 NAT 閘道，將公有 IP 地址指派給任務的彈性網路界面。若要讓私有子網路中的 Fargate 任務提取容器映像，您需要子網路中的 NAT 閘道以將請求路由到網際網路。在 Amazon ECR 中託管容器映像時，可以將 Amazon ECR 設定為使用介面 VPC 端點。在這種情況下，任務的私有 IPv4 地址會用於映像提取。如需 Amazon ECR 介面端點的詳細資訊，請參閱 Amazon Elastic Container Registry 使用者指南中的 Amazon ECR 介面 VPC 端點 (AWS PrivateLink)。

以下是 Fargate 服務的 `networkConfiguration`區段範例：

```
"networkConfiguration": {
   "awsvpcConfiguration": {
      "assignPublicIp": "ENABLED",
      "securityGroups": [ "sg-12345678" ],
      "subnets": [ "subnet-12345678" ]
   }
}
```

任務資源限制

AWS Fargate 上適用於 Linux 容器的 Amazon ECS 任務定義支援 `ulimits` 參數定義要為容器設定的資源限制。

AWS Fargate 上適用於 Windows 的 Amazon ECS 任務定義不支援 `ulimits` 參數定義要為容器設定的資源限制。

在 Fargate 上託管的 Amazon ECS 任務會使用作業系統設定的預設資源限制值，但是 nofile 資源限制參數除外。nofile 資源限制會對容器可使用的開放檔案數量設限。在 Fargate 上，預設的 nofile 軟限制為 65535，硬限制為 65535。您可以將兩個限制的值設定為 1048576。

以下是範例任務定義程式碼片段，示範如何定義已加倍的自訂 nofile 限制：

```
"ulimits": [
    {
       "name": "nofile",
       "softLimit": 2048,
       "hardLimit": 8192
    }
]
```

如需可調整之其他資源限制的詳細資訊，請參閱 資源限制。

日誌

事件記錄

Amazon ECS 記錄 EventBridge 採取的動作。您可以使用適用於 EventBridge 的 Amazon ECS 事件，接收有關您的 Amazon ECS 叢集、服務和任務目前狀態的近乎即時通知。此外，您可以自動執行動作來回應這些事件。如需詳細資訊，請參閱使用 EventBridge 自動化對 Amazon ECS 錯誤的回應。

任務生命週期日誌記錄

在 Fargate 上執行的任務會發佈時間戳記，以透過任務生命週期的狀態追蹤任務。您可以在 和 中查看任務詳細資訊中的時間戳記， AWS Management Console 方法是在 AWS CLI 和 SDKs 中描述任務。例如，您可以使用時間戳記來評估任務下載容器映像所花費的時間，並決定是否應最佳化容器映像大小，或使用 Seekable OCI 索引。如需有關容器映像實務的詳細資訊，請參閱 Amazon ECS 容器映像的最佳實務。

應用程式日誌記錄

AWS Fargate 的 Amazon ECS 任務定義支援日誌組態的 awslogs、splunk 和 awsfirelens 日誌驅動程式。

awslogs 日誌驅動程式會設定您的 Fargate 任務，將日誌資訊傳送給 Amazon CloudWatch Logs。以下顯示任務定義片段，其中設定了 awslogs 日誌驅動程式：

```
"logConfiguration": {
   "logDriver": "awslogs",
```

```
    "options": {
       "awslogs-group" : "/ecs/fargate-task-definition",
       "awslogs-region": "us-east-1",
       "awslogs-stream-prefix": "ecs"
    }
}
```

如需在任務定義中使用 awslogs 日誌驅動程式將您的容器日誌傳送到 CloudWatch Logs 的詳細資訊，請參閱將 Amazon ECS 日誌傳送至 CloudWatch 。

如需任務定義中 awsfirelens 日誌驅動程式的詳細資訊，請參閱 將 Amazon ECS 日誌傳送至 AWS 服務或 AWS Partner。

如需在任務定義中使用 splunk 日誌驅動程式的詳細資訊，請參閱 splunk 日誌驅動程式。

任務儲存體

對於 Fargate 上託管的 Amazon ECS 任務，支援下列儲存類型：

- Amazon EBS 磁碟區為資料密集型容器化工作負載提供經濟實惠、耐用、高效能的區塊儲存。如需詳細資訊，請參閱搭配 Amazon ECS 使用 Amazon EBS 磁碟區。
- 適用於持久性儲存的 Amazon EFS 磁碟區。如需詳細資訊，請參閱搭配 Amazon ECS 使用 Amazon EFS 磁碟區。
- 用於暫時性儲存的繫結掛載。如需詳細資訊，請參閱搭配 Amazon ECS 使用繫結掛載。

使用 Seekable OCI (SOCI) 延遲載入容器映像

在使用 Linux 平台版本 1.4.0 的 Fargate 上，Amazon ECS 任務可以使用 Seekable OCI (SOCI) 來協助更快地啟動任務。使用 SOCI 時，容器只需耗費幾秒鐘就可以開始提取映像，因此在背景下載影像時，可為環境設定和應用程式執行個體化提供時間。這稱為延遲載入。當 Fargate 啟動 Amazon ECS 任務時，Fargate 會自動偵測任務中映像是否存在 SOCI 索引，並啟動容器，而無需等待整個映像下載完成。

對於在沒有 SOCI 索引的情況下執行的容器，容器映像會在容器啟動之前完全下載。此行為在 Fargate 的所有其他平台版本以及 Amazon EC2 執行個體上的 Amazon ECS 最佳化 AMI 上均相同。

Seekable OCI 索引

Seekable OCI (SOCI) 是一種由 開發的開放原始碼技術 AWS ，可透過延遲載入容器映像來更快速地啟動容器。SOCI 的運作原理是在現有容器映像中建立檔案索引 (SOCI 索引)。此索引有助於更快地啟動

容器，提供在下載整個映像之前從容器映像中擷取個別檔案的功能。SOCI 索引必須以成品的形式儲存在與容器登錄中的映像相同的儲存庫中。您應該只使用來自信任來源的 SOCI 索引，因為索引是映像內容的權威來源。如需詳細資訊，請參[引入 Seekable OCI 用於延遲載入容器映像](#)。

考量事項

如果您希望 Fargate 使用 SOCI 索引在任務中延遲載入容器映像，請考慮以下幾點：

- 只有在 Linux 平台版本 `1.4.0` 上執行的任務才能使用 SOCI 索引。不支援在 Fargate 上執行 Windows 容器的任務。
- 支援在 X86_64 或 ARM64 CPU 架構上執行的任務。
- 任務定義中的容器映像均須在與映像相同的容器登錄中具有 SOCI 索引。
- 任務定義中的容器映像均須儲存在相容的映像登錄中。以下列出相容的登錄：
  - Amazon ECR 私有登錄檔。
- 僅支援使用 gzip 壓縮或未壓縮的容器映像。不支援使用 zstd 壓縮的容器映像。
- 我們建議您嘗試延遲載入大於 250 MiB 壓縮大小的容器映像。您不太可能看到載入較小映像的時間減少。
- 由於延遲載入可能會變更任務開始所需的時間，因此您可能需要變更各種逾時，例如 Elastic Load Balancing 的運作狀態檢查寬限期。
- 如果您想要防止容器映像延遲載入，請從容器登錄中刪除 SOCI 索引。如果任務中的容器映像不符合其中一項考量事項，則會透過預設方法下載該容器映像。

建立 Seekable OCI 索引

若要讓容器映像延遲載入，它需要建立 SOCI 索引（中繼資料檔案），並沿著容器映像的側邊存放在容器映像儲存庫中。若要建立和推送 SOCI 索引，您可以使用 GitHub 上的開放原始碼 [soci-snapshotter CLI 工具](#)。或者，您可以部署 CloudFormation AWS SOCI 索引建置器。這是無伺服器解決方案，當容器映像推送至 Amazon ECR 時， 會自動建立和推送 SOCI 索引。如需解決方案和安裝步驟的詳細資訊，請參閱 GitHub 上的 [CloudFormation AWS SOCI 索引建置器](#)。CloudFormation AWS SOCI 索引建置器是一種自動化 SOCI 入門的方式，而開放原始碼社交工具在索引產生方面具有更大的靈活性，並且能夠將索引產生整合到持續整合和持續交付 (CI/CD) 管道中。

> **Note**
>
> 若要為映像建立 SOCI 索引，映像必須存在於執行 `soci-snapshotter` 的電腦上的 containerd 映像存放區中。如果映像位於 Docker 映像存放區中，則無法找到該映像。

驗證任務是否使用延遲載入

若要驗證任務是否使用 SOCI 延遲載入，請從任務內部檢查任務中繼資料端點。當您查詢任務中繼資料端點版本 4 時，您從中查詢的容器的預設路徑中有一個 `Snapshotter` 欄位。此外，`/task` 路徑中的每個容器都有 `Snapshotter` 欄位。此欄位的預設值為 `overlayfs`，`soci`如果使用 SOCI，則此欄位會設為 。

執行 Windows 之 EC2 執行個體的 Amazon ECS 任務定義差異

在 EC2 Windows 執行個體上執行的任務不支援所有可用的 Amazon ECS 任務定義參數。完全不支援某些參數，而其他參數的行為則不同。

Amazon EC2 Windows 任務定義不支援下列任務定義參數：

- `containerDefinitions`
  - `disableNetworking`
  - `dnsServers`
  - `dnsSearchDomains`
  - `extraHosts`
  - `links`
  - `linuxParameters`
  - `privileged`
  - `readonlyRootFilesystem`
  - `user`
  - `ulimits`
- `volumes`
  - `dockerVolumeConfiguration`
- `cpu`

  我們建議為 Windows 容器指定容器層級的 CPU。
- `memory`

  我們建議為 Windows 容器指定容器層級的記憶體。
- `proxyConfiguration`
- `ipcMode`

- `pidMode`
- `taskRoleArn`

  EC2 Windows 執行個體功能上任務的 IAM 角色需要額外的組態，但此組態的大部分都類似於在 Linux 容器執行個體上設定任務的 IAM 角色。如需詳細資訊，請參閱 the section called “ Amazon EC2 Windows 執行個體額外組態”。

使用主控台建立 Amazon ECS 任務定義

您可以建立任務定義，以便定義您作為任務或服務執行的應用程式。

當您為外部啟動類型建立任務定義時，您需要使用 JSON 編輯器建立任務定義，並將 `requireCapabilities` 參數設定為 `EXTERNAL`。

您可以使用主控台體驗或指定 JSON 檔案來建立任務定義。

JSON 驗證

Amazon ECS 主控台 JON 編輯器會對 JSON 檔案的以下方面進行驗證：

- 檔案是有效的 JSON 檔案。
- 檔案不包含任何外部金鑰。
- 檔案包含 `familyName` 參數。
- 在 下至少有一個項目`containerDefinitions`。

AWS CloudFormation 堆疊

下列行為適用於 2023 年 1 月 12 日之前在新的 Amazon ECS 主控台中建立的任務定義。

當您建立任務定義時，Amazon ECS 主控台會自動建立名稱開頭為 的 CloudFormation 堆疊`ECS-Console-V2-TaskDefinition-`。如果您使用 AWS CLI 或 AWS SDK 取消註冊任務定義，則必須手動刪除任務定義堆疊。如需詳細資訊，請參閱AWS CloudFormation 《 使用者指南》中的刪除堆疊。

2023 年 1 月 12 日之後建立的任務定義不會自動為其建立 CloudFormation 堆疊。

程序

Amazon ECS console

1. 開啟主控台，網址為 https://console.aws.amazon.com/ecs/v2。
2. 在導覽窗格中，選擇 Task Definitions (任務定義)。
3. 在建立新任務定義功能表上，選擇建立新任務定義。
4. 在任務定義系列中，請為任務定義指定唯一名稱。
5. 在啟動類型中，選擇應用程式環境。主控台預設為 AWS Fargate（無伺服器）。Amazon ECS 使用此值執行驗證，以確保任務定義參數對基礎設施類型有效。
6. 對於 Operating system/Architecture (作業系統/架構)，選擇適用於任務的作業系統和 CPU 架構。

   若要在 64 位元 ARM 架構上執行任務，請選擇 Linux/ARM64。如需詳細資訊，請參閱the section called “執行時間平台”。

   若要在 Windows 容器上執行 AWS Fargate 任務，請選擇支援的 Windows 作業系統。如需詳細資訊，請參閱the section called “作業系統和架構”。
7. 在 Task size (任務大小) 中，選擇要為任務預留的 CPU 和記憶體大小。CPU 值指定為 vCPU，記憶體則指定為 GB。

   對於在 Fargate 上託管的任務，下表顯示了有效的 CPU 和記憶體組合。

CPU 數值	記憶體數值	AWS Fargate 支援的作業系統
256 (.25 vCPU)	512 MiB、1 GB、2 GB	Linux
512 (.5 vCPU)	1 GB、2 GB、3 GB、4 GB	Linux
1024 (1 vCPU)	2 GB、3 GB、4 GB、5 GB、6 GB、7 GB、8 GB	Linux、Windows
2048 (2 vCPU)	介於 4 GB 與 16 GB 之間，以 1 GB 為單位遞增	Linux、Windows

CPU 數值	記憶體數值	AWS Fargate 支援的作業系統
4096 (4 vCPU)	介於 8 GB 與 30 GB 之間，以 1 GB 為單位遞增	Linux、Windows
8192 (8 vCPU) **Note** 此選項需要 Linux 平台 `1.4.0` 或更新版本。	介於 16 GB 與 60 GB 之間，以 4 GB 為單位遞增	Linux
16384 (16vCPU) **Note** 此選項需要 Linux 平台 `1.4.0` 或更新版本。	介於 32 GB 與 120 GB 之間，以 8 GB 為單位遞增	Linux

對於在 Amazon EC2 上託管的任務，支援的任務 CPU 值介於 128 個 CPU 單位 (0.125 vCPU) 到 10240 個 CPU 單位 (10 vCPU) 之間。若要以 GB 指定記憶體值，請在該值後輸入 GB。例如，若要將記憶體值設定為 3GB，請輸入 3GB。

> **Note**
>
> Windows 容器會忽略任務層級的 CPU 和記憶體參數。

8. 在 Network mode (網路模式) 中選擇要使用的網路模式。預設網路模式為 awsvpc 模式。如需詳細資訊，請參閱 Amazon ECS 任務聯網。

   如果您選擇橋接，請在連接埠映射下，針對主機連接埠輸入容器執行個體上的連接埠號碼，以預留容器。

9. （選用）展開任務角色區段來設定任務的 AWS Identity and Access Management (IAM) 角色：

   a. 在 Task role (任務角色) 中，選擇要指派給任務的 IAM 角色。任務 IAM 角色為任務中的容器提供呼叫 AWS API 操作的許可。

   b. 在任務執行角色中，選擇角色。

      如需有關何時使用任務執行角色的資訊，請參閱 the section called “任務執行 IAM 角色”。如果您不需要該角色，請選擇無。

10. （選用）展開任務置放區段以新增置放限制。任務置放限制條件可讓您使用內建或自訂屬性來篩選用於置放任務的容器執行個體。

11. （選用）展開故障注入區段以啟用故障注入。故障注入可讓您測試應用程式對特定損害案例的回應。

12. 請為您任務定義中要定義的每個容器完成以下步驟。

    a. 在 Name (名稱) 中，輸入容器的名稱。

    b. 在 Image URI (映像 URI) 中，輸入要用來啟動容器的映像。Amazon ECR Public Gallery 登錄檔中的影像只能使用 Amazon ECR Public 登錄檔名稱來指定。例如，如果指定 `public.ecr.aws/ecs/amazon-ecs-agent:latest` ，則會使用託管在 Amazon ECR Public Gallery 上的 Amazon Linux 容器。對於所有其他儲存庫，請使用 `repository-url/image:tag`或 `repository-url/image@digest` 格式來指定儲存庫。

    c. 如果您的映像位於 Amazon ECR 以外的私有登錄中，請在私有登錄下，開啟私有登錄身分驗證。然後，在 Secrets Manager ARN 或名稱中，輸入密碼的 Amazon Resource Name (ARN)。

    d. 對於必要容器，如果您的任務定義已定義兩個或多個容器，您可以指定容器是否應視為必要容器。當容器標示為 Essential 時，如果該容器停止，則任務會停止。每個任務定義必須至少包含一個基本容器。

    e. 連接埠映射允許容器存取主機上的連接埠，以傳送或接收流量。在 Port mappings (連接埠映射) 下執行以下其中一項動作：

       - 若您使用 awsvpc 網路模式，請在 Container port (容器連接埠) 和 Protocol (通訊協定) 中，選擇要用於容器的連接埠映射。
       - 若您使用 bridge 網路模式，請在 Container port (容器連接埠) 和 Protocol (通訊協定) 中，選擇要用於容器的連接埠映射。

選擇 Add more port mappings (新增更多連接埠映射)，以指定其他容器連接埠映射。

f. 若要授予容器對其根檔案系統的唯讀存取權限，請在唯讀根檔案系統中選取唯讀。

g. （選用）若要定義與任務層級值不同的容器層級 CPU、GPU 和記憶體限制，請在資源配置限制下執行下列動作：

- 針對 CPU，輸入 Amazon ECS 容器代理程式保留給容器的 CPU 單位數量。
- 在 GPU 中，輸入容器執行個體的 GPU 單元數量。

  支援 GPU 的 Amazon EC2 執行個體，每個 GPU 都有 1 個 GPU 單元。如需詳細資訊，請參閱the section called “GPU 工作負載的任務定義”。
- 針對記憶體硬性限制，以 GB 為單位輸入要呈現給容器的記憶體量。如果容器嘗試超過硬限制，容器將會停止。
- Docker 20.10.0 或更新版本的協助程式會為容器保留至少 6 MB (MiB) 的記憶體，因此請不要為您的容器指定少於 6 MiB 的記憶體。

  Docker 19.03.13-ce 或更早的協助程式會為容器保留至少 4 MiB 的記憶體，因此請不要為您的容器指定少於 4 MiB 的記憶體。
- 在記憶體軟限制中，輸入為容器保留的記憶體軟限制 (GB)。

  當系統記憶體爭用時，Docker 會嘗試將容器記憶體保持在此軟性限制。如果您未指定任務層級的記憶體，您必須為記憶體硬限制和記憶體軟限制之一 (或兩者) 指定非零整數。如果同時指定兩者，記憶體硬限制必須大於記憶體軟限制。

  Windows 容器不支援此功能。

h. (選用) 展開環境變數區段，以指定要注入到容器中的環境變數。您可以使用鍵值對或大量指定託管在 Amazon S3 儲存貯體中的環境變數檔案，來個別指定環境變數。如需有關如何格式化環境變數檔案的資訊，請參閱 將個別環境變數傳遞至 Amazon ECS 容器。

  當您指定秘密儲存的環境變數時，請在金鑰中輸入秘密名稱。然後，針對 ValueFrom，輸入 Systems Manager 參數存放區秘密或 Secrets Manager 秘密的完整 ARN

i. (選用) 選擇使用日誌收集選項來指定日誌組態。每個可用的日誌驅動程序式都有要指定的日誌驅動程序選項。預設選項會將容器日誌傳送至 Amazon CloudWatch Logs。其他日誌驅動程式選項是透過使用 來設定 AWS FireLens。如需詳細資訊，請參閱將 Amazon ECS 日誌傳送至 AWS 服務或 AWS Partner。

  下方更詳細地描述了每個容器日誌目的地。

- Amazon CloudWatch – 設定任務，將容器日誌傳送至 CloudWatch Logs。系統會提供預設日誌驅動程式選項，這些選項會代表您建立 CloudWatch 日誌群組。若要指定不同的日誌群組名稱，請變更驅動程式選項值。
- 匯出日誌至 Splunk - 設定任務將容器日誌傳送至將日誌傳送至遠端服務的Splunk驅動程式。您必須輸入 Splunk Web 服務的 URL。Splunk 字符會指定為秘密選項，因為它可以被視為敏感資料。
- 匯出日誌至 Amazon Data Firehose – 設定任務將容器日誌傳送至 Firehose。會提供預設日誌驅動程式選項，將日誌傳送至 Firehose 交付串流。若要指定不同的交付串流名稱，請變更驅動程式選項值。
- 匯出日誌至 Amazon Kinesis Data Streams – 設定任務將容器日誌傳送至 Kinesis Data Streams。提供預設日誌驅動程式選項，可將日誌傳送至 Kinesis Data Streams 串流。若要指定不同的串流名稱，請變更驅動程式選項值。
- 將日誌匯出至 Amazon OpenSearch Service – 設定任務將容器日誌傳送至 OpenSearch Service 網域。務必提供日誌驅動程式選項。
- 將日誌匯出至 Amazon S3 – 設定任務將容器日誌傳送至 Amazon S3 儲存貯體。提供預設日誌驅動程式選項，但您必須指定有效的 Amazon S3 儲存貯體名稱。

j. (選用) 設定其他容器參數。

若要設定此選項	執行此作業	
重新啟動政策 這些選項會定義重新啟動政策，以在容器結束時重新啟動容器。	展開重新啟動政策，然後設定下列項目： • 若要為容器啟用重新啟動政策，請開啟啟用重新啟動政策。 • 對於忽略的結束代碼，請指定以逗號分隔的整數容器結束代碼清單。如果容器以任何指定的結束代碼結束，Amazon ECS 將不會嘗試重新啟動容器。如果未指定任何結束代碼，Amazon ECS 將不會忽略任何結束代碼。 • 針對嘗試重設期間，請指定容器必須在 執行的整數期間，之後才能在結束時嘗試重新啟動。Amazon ECS 只能嘗試在每個嘗試重設期間秒重新啟動容器一次。如果未指定任何項目，容器必須執行 300 秒，才能嘗試重新啟動。	

若要設定此選項	執行此作業	
HealthCheck 這些是判斷容器是否運作狀態良好的命令。如需詳細資訊，請參閱使用容器運作狀態檢查判斷 Amazon ECS 任務運作狀態。	展開 HealthCheck，然後設定下列項目： • 在 Command (命令) 欄位中，輸入以逗號分隔的命令清單。您可以用 CMD 作為此命令的開頭，如此能直接執行命令引數；或以 CMD-SHELL 為開頭，藉以使用容器預設的 Shell 來執行命令。如果均尚未指定，則使用 CMD。 • 在 Interval (間隔) 欄位中，輸入每次運作狀態檢查之間的秒數。有效值介於 5 到 30。 • 在 Timeout (逾時) 欄位中，輸入在判定為失敗之前，等待運作狀態檢查成功執行的時間 (以秒為單位)。有效值介於 2 到 60。 • 在 Start period (開始期間) 欄位中，輸入在運作狀態檢查命令執行之前，等待容器引導的時間 (以秒為單位)。有效值介於 0 到 300。 • 在 Retries (重試次數) 欄位中，輸入失敗時重試	

若要設定此選項	執行此作業	
	運作狀態檢查命令的次數。有效值介於 1 到 10 之間。	
啟動相依性排序 此選項定義容器啟動和關閉的相依性。容器可包含多個相依性。	展開啟動相依性順序，然後設定下列項目： a. 選擇新增容器相依性。 b. 在容器中，選擇容器。 c. 在條件中，選擇啟動相依性條件。 若要新增其他相依性，請選擇新增容器相依性。	
容器逾時 這些選項決定何時啟動和停止容器。	展開容器逾時，然後設定以下項目： • 若要設定在放棄解決容器相依性之前等待的時間，請在啟動逾時中輸入秒數。 • 若要設定容器未自行正常退出時，在容器停止之前等待的時間，請在停止逾時中輸入秒數。	

若要設定此選項	執行此作業	
容器網路設定 這些選項決定是否要在容器內使用聯網功能。	展開容器網路設定，然後設定以下項目： • 若要停用容器聯網功能，請選取關閉聯網。 • 若要設定顯示給容器的 DNS 伺服器 IP 地址，請在 DNS 伺服器中，以個別的行輸入每個伺服器的 IP 地址。 • 若要設定 DNS 網域來搜尋呈現給容器non-fully-qualified主機名稱，請在 DNS 搜尋網域中，以個別行輸入每個網域。 模式是 **^[a-zA-Z0-9-.]{0,253}[a-zA-Z0-9]$** 。 • 若要設定容器主機名稱，請在主機名稱中輸入容器主機名稱。 • 若要新增附加至容器上 `/etc/hosts` 檔案的主機名稱和 IP 地址映射，請選擇新增額外主機，然後在主機名稱和 IP 地址中輸入主機名稱和 IP 地址。	

若要設定此選項	執行此作業	
Docker 組態 這些會覆寫 中的值Docker file。	展開Docker組態，然後設定下列項目： • 在命令中，輸入容器的可執行命令。 此參數在Docker遠端 API 的建立容器區段Cmd中映射至 ，並將 COMMAND選項映射至 docker run 。此參數會覆寫 中的CMD指令Dockerfile。 • 針對進入點，輸入傳遞至容器的 Docker ENTRYPOINT。 此參數在Docker遠端 API 的建立容器區段Entrypoint 中映射至 ，並將 --entrypoint 選項映射至 docker run。此參數會覆寫中的ENTRYPOINT 指令Dockerfile。 • 在工作目錄中，輸入容器將執行所提供的任何進入點和命令指示的目錄。 此參數在Docker遠端 API 的建立容器區	

若要設定此選項	執行此作業	
	段WorkingDir 中映射至 ，而 --workdir 選項映射至 docker run 。此參數會覆寫 中的WORKDIR指令Dockerfile。	
資源限制 (Ulimits) 這些值會覆寫作業系統的預設資源配額設定。 此參數會映射到 Docker Remote API 的建立容器區段中的 Ulimits 以及 docker run 的 --ulimit 選項。	展開資源限制 (ulimits)，然後選擇新增 ulimit。針對限制名稱，選擇限制。然後，在軟限制和硬限制中輸入值。 若要新增其他 ulimits，請選擇新增 ulimit。	
Docker 標籤 此選項會將中繼資料新增至您的容器。 此參數會映射到 Docker Remote API 的建立容器區段中的 Labels 以及 docker run 的 --label 選項。	展開Docker標籤，選擇新增索引鍵值對，然後輸入索引鍵和值。 若要新增其他Docker標籤，請選擇新增索引鍵值對。	

k. (選用) 選擇 Add more containers (新增更多容器)，以新增其他容器至任務定義。

13. （選用）儲存區段用於擴展 Fargate 上託管任務的暫時性儲存量。您也可以使用本節來新增任務的資料磁碟區組態。

    - 若要將 Fargate 任務的可用暫時性儲存擴展到大於 20 gibibytes (GiB) 的預設值，請在 Amount (數量) 中輸入值，可輸入的最大數值為 200 GiB。

14. （選用）若要為任務定義新增資料磁碟區組態，請選擇新增磁碟區，然後遵循下列步驟。

a. 在 Volume name (磁碟區名稱) 中，輸入資料磁碟區的名稱。在建立容器掛載點時會使用資料磁碟區名稱。

b. 針對磁碟區組態，選取您要在建立任務定義時或在部署期間設定磁碟區。

> **Note**
>
> 建立任務定義時可設定的磁碟區包括繫結掛載、、Amazon EFS Docker和 Amazon FSx for Windows File Server。可在執行任務時或在建立或更新服務時設定磁碟區包含 Amazon EBS。

c. 針對磁碟區類型，選取與您選取的組態類型相容的磁碟區類型，然後設定磁碟區類型。

磁碟區類型	步驟	
繫結掛載	a. 選擇 Add mount point (新增掛載點)，然後設定下列項目： • 對於 Container (容器)，選擇掛載點的容器。 • 在來源磁碟區中，選擇要掛載到容器的資料磁碟區。 • 在 Container path (容器路徑) 中，輸入容器的路徑，以掛載磁碟區。 • 在唯讀中，選取容器是否具有對磁碟區的唯讀存取權。 b. 若要新增其他掛載點，請 Add mount point (新增掛載點)。	

磁碟區類型	步驟	
EFS	a. 在 File system ID (檔案系統 ID) 欄位中，選擇 Amazon EFS 檔案系統 ID。 b. (選用) 在 Root directory (根目錄) 欄位中，輸入 Amazon EFS 檔案系統中的目錄，其將掛載作為主機內的根目錄。如果省略此參數，使用 Amazon EFS 磁碟區的根目錄。 如果您計劃使用 EFS 存取點，請將此欄位留空。 c. (選用) 在 Access point (存取點) 欄位中，選擇要使用的存取點 ID。 d. (選用) 若要加密 Amazon EFS 檔案系統與 Amazon ECS 主機之間的資料，或在掛載磁碟區時使用任務執行角色，請選擇 Advanced configurations (進階組態)，然後設定下列項目： • 若要加密 Amazon EFS 檔案系統與 Amazon ECS 主機之間的資料，請選取 Transit encryption (傳輸加	

磁碟區類型	步驟	
	密)，然後在 Port (連接埠) 欄位中輸入要在 Amazon ECS 主機與 Amazon EFS 伺服器之間傳送加密資料時使用的連接埠。如果您未指定傳輸加密連接埠，它會使用 Amazon EFS 掛載協助程式使用的連接埠選擇策略。如需詳細資訊，請參閱《Amazon Elastic File System 使用者指南》中的 EFS 掛載協助程式。 • 若要在掛載 Amazon ECS 檔案系統時，使用任務定義中定義的 Amazon EFS 任務 IAM 角色，請選擇 IAM authorization (IAM 授權)。 e. 選擇 Add mount point (新增掛載點)，然後設定下列項目： • 對於 Container (容器)，選擇掛載點的容器。 • 在來源磁碟區中，選擇要掛載到容器的資料磁碟區。	

磁碟區類型	步驟	
	• 在 Container path (容器路徑) 中，輸入容器的路徑，以掛載磁碟區。 • 在唯讀中，選取容器是否具有對磁碟區的唯讀存取權。 f. 若要新增其他掛載點，請 Add mount point (新增掛載點)。	

磁碟區類型	步驟	
Docker	a. 針對驅動程式，輸入Docker磁碟區組態。Windows 容器僅支援使用本機驅動程式。若要使用綁定掛載，請指定主機。 b. 在 Scope (範圍) 欄位中，選擇磁碟區生命週期。 • 若要在任務開始與停止時讓生命週期持續，請選擇 Task (任務)。 • 若要讓磁碟區在任務停止後可持續執行，請選擇 Shared (共享)。 c. 選擇 Add mount point (新增掛載點)，然後設定下列項目： • 對於 Container (容器)，選擇掛載點的容器。 • 在來源磁碟區中，選擇要掛載到容器的資料磁碟區。 • 在 Container path (容器路徑) 中，輸入容器的路徑，以掛載磁碟區。 •	

磁碟區類型	步驟	
	在唯讀中，選取容器是否具有對磁碟區的唯讀存取權。 d. 若要新增其他掛載點，請 Add mount point (新增掛載點)。	

磁碟區類型	步驟	
FSx for Windows File Server	a. 在檔案系統 ID 中，選擇 FSx for Windows File Server 檔案系統 ID。 b. 對於根目錄，輸入目錄，在 FSx for Windows File Server 檔案系統中輸入要掛載為主機內根目錄的目錄。 c. 在憑證參數中，請選擇憑證的儲存方式。 • 若要使用 AWS Secrets Manager，請輸入 Secrets Manager 秘密的 Amazon Resource Name (ARN)。 • 若要使用 AWS Systems Manager，請輸入 Systems Manager 參數的 Amazon Resource Name (ARN)。 d. 針對網域，輸入由 AWS Directory Service for Microsoft Active Directory (AWS Managed Microsoft AD) 目錄或自我託管 EC2 Active Directory 託管的完整網域名稱。	

磁碟區類型	步驟	
	e. 選擇 Add mount point (新增掛載點)，然後設定下列項目： • 對於 Container (容器)，選擇掛載點的容器。 • 在來源磁碟區中，選擇要掛載到容器的資料磁碟區。 • 在 Container path (容器路徑) 中，輸入容器的路徑，以掛載磁碟區。 • 在唯讀中，選取容器是否具有對磁碟區的唯讀存取權。 f. 若要新增其他掛載點，請 Add mount point (新增掛載點)。	

磁碟區類型	步驟	
Amazon EBS	a. 選擇 Add mount point (新增掛載點)，然後設定下列項目： • 對於 Container (容器)，選擇掛載點的容器。 • 在來源磁碟區中，選擇要掛載到容器的資料磁碟區。 • 在 Container path (容器路徑) 中，輸入容器的路徑，以掛載磁碟區。 • 在唯讀中，選取容器是否具有對磁碟區的唯讀存取權。 b. 若要新增其他掛載點，請 Add mount point (新增掛載點)。	

15. 若要從另一個容器新增磁碟區，請選擇新增磁碟區來源，然後設定下列項目：
    - 在容器中，選擇容器。
    - 在來源中，選擇包含您要掛載的磁碟區的容器。
    - 在唯讀中，選取容器是否具有對磁碟區的唯讀存取權。
16. （選用）若要使用 AWS Distro for OpenTelemetry整合來設定應用程式追蹤和指標收集設定，請展開監控，然後選取使用指標收集來收集任務指標，並傳送至 Amazon CloudWatch 或 Amazon Managed Service for Prometheus。選取此選項時，Amazon ECS 會建立 AWS

Distro for OpenTelemetry預先設定以傳送應用程式指標的容器附屬項目。如需詳細資訊，請參閱使用應用程式指標關聯 Amazon ECS 應用程式效能。

a. 若選取 Amazon CloudWatch，您的自定應用程式指標將作為自定指標路由至 CloudWatch。如需詳細資訊，請參閱將應用程式指標匯出至 Amazon CloudWatch。

> **Important**
>
> 若將應用程式指標匯出至 Amazon CloudWatch，您的任務定義需要具有所需許可的任務 IAM 角色。如需詳細資訊，請參閱AWS Distro for OpenTelemetry 與 Amazon CloudWatch 整合所需的 IAM 許可。

b. 若您選取 Amazon Managed Service and Prometheus (Prometheus libraries instrumentation) (Amazon Managed Service for Prometheus (Prometheus 程式庫檢測))，則您的任務層級 CPU、記憶體、網路，以及儲存指標和自定應用程式指標，都將路由至 Amazon Managed Service for Prometheus。針對工作區遠端寫入端點，輸入Prometheus工作區的遠端寫入端點 URL。針對擴展目標，輸入收集器可用來抓取指標資料的主機和連接埠 AWS Distro for OpenTelemetry。如需詳細資訊，請參閱將應用程式指標匯出至 Amazon Managed Service for Prometheus。

> **Important**
>
> 若將應用程式指標匯出至 Amazon Managed Service for Prometheus，您的任務定義需要具有所需許可的任務 IAM 角色。如需詳細資訊，請參閱AWS Distro for OpenTelemetry 與 Amazon Managed Service for Prometheus 整合所需的 IAM 許可。

c. 當您選取 Amazon Managed Service for Prometheus (OpenTelemetry 檢測） 時，您的任務層級 CPU、記憶體、網路和儲存指標和自訂應用程式指標會路由至 Amazon Managed Service for Prometheus。針對工作區遠端寫入端點，輸入Prometheus工作區的遠端寫入端點 URL。如需詳細資訊，請參閱將應用程式指標匯出至 Amazon Managed Service for Prometheus。

> **Important**
>
> 若將應用程式指標匯出至 Amazon Managed Service for Prometheus，您的任務定義需要具有所需許可的任務 IAM 角色。如需詳細資訊，請參閱AWS Distro for

OpenTelemetry 與 Amazon Managed Service for Prometheus 整合所需的 IAM 許可。

17. (選用) 展開 Tags (標籤) 區段，將標籤作為鍵值對新增至任務定義中。
    - [新增標籤] 選擇新增標籤，然後執行下列操作：
      - 在索引鍵中，輸入索引鍵名稱。
      - 在值中，進入索引鍵值。
    - [移除標籤] 在標籤旁邊，選擇 移除標籤。
18. 選擇建立來註冊任務定義。

Amazon ECS console JSON editor

1. 開啟主控台，網址為 https://console.aws.amazon.com/ecs/v2。
2. 在導覽窗格中，選擇 Task Definitions (任務定義)。
3. 在建立新任務定義功能表上，選擇使用 JSON 建立新任務定義。
4. 在 JSON 編輯工具方塊中，編輯您的 JSON 檔案，

   JSON 必須通過 the section called “JSON 驗證” 中指定的驗證檢查。
5. 選擇 Create (建立)。

使用主控台更新 Amazon ECS 任務定義

任務定義修訂版是目前任務定義的複本，並以新參數值取代現有參數值。您未修改的所有參數都位於新修訂版中。

若要更新任務定義，請建立任務定義修訂版。如果任務定義是用於服務，則您必須更新該服務才能使用更新的任務定義。

在建立修訂版時，您可以修改以下容器屬性和環境屬性。

- 容器映像 URI
- 連接埠映射
- 環境變數
- 任務大小

- 容器大小
- 任務角色
- 任務執行角色
- 磁碟區和容器掛載點
- 私有登錄檔

JSON 驗證

Amazon ECS 主控台 JON 編輯器會對 JSON 檔案的以下方面進行驗證：

- 檔案為有效的 JSON 檔案
- 檔案不包含任何無關的金鑰
- 該檔案包含 `familyName` 參數
- `containerDefinitions` 下至少有一個項目

程序

Amazon ECS console

1. 開啟主控台，網址為 https://console.aws.amazon.com/ecs/v2。
2. 從導覽列中選擇包含您任務定義的區域。
3. 在導覽窗格中，選擇 Task Definitions (任務定義)。
4. 選擇任務定義。
5. 選擇任務定義修訂版，然後選擇建立新修訂版、建立新修訂版。
6. 在 Create new task definition revision (建立新任務定義修訂版) 頁面上進行變更。例如，若要變更現有的容器定義 (例如容器映像、記憶體限制或連接埠映射)，請選取容器，然後選擇進行變更。
7. 驗證資訊，然後選擇更新。
8. 如果您的任務定義是用於服務，請使用更新的任務定義來更新您的服務。如需詳細資訊，請參閱使用主控台更新 Amazon ECS 服務。

Amazon ECS console JSON editor

1. 開啟主控台，網址為 https://console.aws.amazon.com/ecs/v2。

2. 在導覽窗格中，選擇 Task Definitions (任務定義)。
3. 選擇 Create new revision (建立新修訂版)，以及 Create new revision with JSON (使用 JSON 建立新修訂版)。
4. 在 JSON 編輯工具方塊中，編輯您的 JSON 檔案，

   JSON 必須通過 the section called “JSON 驗證” 中指定的驗證檢查。
5. 選擇 Create (建立)。

使用主控台取消註冊 Amazon ECS 任務定義修訂

您可以取消註冊任務定義修訂，以便在您想要執行任務或更新服務時，它不會再顯示於您的 `ListTaskDefinition` API 呼叫或主控台中。

當您取消註冊任務定義修訂時，會立即標示為 `INACTIVE`。參考 `INACTIVE` 任務定義修訂的現有任務和服務會繼續執行，而不會中斷。參考 `INACTIVE` 任務定義修訂的現有服務仍然可以透過修改服務所需的計數進行擴展或縮減。

您不能使用 `INACTIVE` 任務定義修訂來執行新任務或建立新服務。您也無法更新現有服務以參考 `INACTIVE` 任務定義修訂 (雖然取消註冊後最多有 10 分鐘的空檔，但這些限制在此期間尚未生效)。

> **Note**
>
> 取消註冊任務系列中的所有修訂時，任務定義系列會移至 `INACTIVE` 清單。新增 `INACTIVE` 任務定義修訂，會將任務定義系列移至 `ACTIVE` 清單。
> 此時，`INACTIVE` 任務定義修訂會在您的帳戶中無限期地保持可發現狀態。不過，此行為未來可能變更。因此，你不應該依賴持續超出任何相關任務和服務生命週期的 `INACTIVE` 任務定義修訂。

AWS CloudFormation 堆疊

下列行為適用於 2023 年 1 月 12 日之前在新的 Amazon ECS 主控台中建立的任務定義。

當您建立任務定義時，Amazon ECS 主控台會自動建立名稱開頭為 的 CloudFormation 堆疊`ECS-Console-V2-TaskDefinition-`。如果您使用 AWS CLI 或 AWS SDK 取消註冊任務定義，則必須手動刪除任務定義堆疊。如需詳細資訊，請參閱AWS CloudFormation 《 使用者指南》中的刪除堆疊。

2023 年 1 月 12 日之後建立的任務定義不會自動為其建立 CloudFormation 堆疊。

程序

取消註冊新任務定義 (Amazon ECS 主控台)

1. 開啟主控台，網址為 https://console.aws.amazon.com/ecs/v2。
2. 從導覽列中選擇包含您任務定義的區域。
3. 在導覽窗格中，選擇 Task Definitions (任務定義)。
4. 在 Task Definitions (任務定義) 頁面，選擇包含您想要取消註冊之一個或多個修訂版的任務定義系列。
5. 在任務定義名稱頁面上，選取要刪除的修訂版，然後選擇動作、取消註冊。
6. 驗證 Deregister (取消註冊) 視窗中的資訊，然後選擇 Deregister (取消註冊) 以完成操作。

使用主控台刪除 Amazon ECS 任務定義修訂

當不再需要 Amazon ECS 中的特定任務定義修訂時，您可以刪除任務定義修訂。

任務定義修訂版刪除後，任務定義修訂版會立即從 `INACTIVE` 狀態轉變為 `DELETE_IN_PROGRESS` 狀態。參考 `DELETE_IN_PROGRESS` 任務定義修訂版的現有任務和服務會繼續執行，不會中斷。

您不能使用狀態為 `DELETE_IN_PROGRESS` 的任務定義修訂版來執行新任務或建立新服務。您也無法更新現有服務以參考狀態為 `DELETE_IN_PROGRESS` 的任務定義修訂版。

當您刪除所有 `INACTIVE` 任務定義修訂時，任務定義名稱不會顯示在主控台中，也不會在 API 中傳回。如果任務定義修訂處於 `DELETE_IN_PROGRESS` 狀態，則任務定義名稱會顯示在主控台中，並在 API 中傳回。Amazon ECS 會保留任務定義名稱，下次您使用該名稱建立任務定義時，修訂版本將會遞增。

可以封鎖刪除的 Amazon ECS 資源

當有任何 Amazon ECS 資源依賴於任務定義修訂時，任務定義刪除請求將不會完成。以下資源可能會阻止任務定義遭刪除：

- Amazon ECS 獨立任務 - 需要任務定義，才能讓任務正常運作。
- Amazon ECS 服務任務 - 任務定義是必要的，才能保持運作狀態。
- Amazon ECS 服務部署和任務集 - 為 Amazon ECS 部署或任務集啟動擴展事件時，需要任務定義。

如果您的任務定義仍處於 DELETE_IN_PROGRESS 狀態，您可以使用 主控台或 AWS CLI 來識別，然後停止封鎖任務定義刪除的資源。

移除封鎖的資源後刪除任務定義

移除封鎖任務定義刪除的資源後，以下規則適用：

- Amazon ECS 任務 - 任務停止後，任務定義刪除最多可能需要 1 小時才能完成。
- Amazon ECS 服務部署和任務集 - 在刪除部署或任務集之後，刪除任務定義最多可能需要 24 小時才能完成。

程序

刪除任務定義 (Amazon ECS 主控台)

您必須先取消註冊任務定義修訂版，然後才能刪除它。如需詳細資訊，請參閱the section called “使用主控台取消註冊任務定義修訂版”。

1. 開啟主控台，網址為 https://console.aws.amazon.com/ecs/v2。
2. 從導覽列中選擇包含您任務定義的區域。
3. 在導覽窗格中，選擇 Task Definitions (任務定義)。
4. 在任務定義頁面，選擇包含您想要刪除之一或多個修訂版的任務定義系列。
5. 在任務定義名稱頁面上，選取要刪除的修訂，然後選擇動作、刪除。

   如果無法使用刪除，您必須取消註冊任務定義。
6. 驗證刪除確認方塊中的資訊，然後選擇刪除以完成。

Amazon ECS 任務定義使用案例

進一步了解如何撰寫各種 AWS 服務和功能的任務定義。

根據您的工作負載，需要設定特定任務定義參數。此外，對於 EC2 啟動類型，您必須選擇專為工作負載設計的特定執行個體。

主題

- AWS Neuron 機器學習工作負載的 Amazon ECS 任務定義
- 適用於深度學習執行個體的 Amazon ECS 任務定義
- 適用於 64 位元 ARM 工作負載的 Amazon ECS 任務定義
- 將 Amazon ECS 日誌傳送至 CloudWatch
- 將 Amazon ECS 日誌傳送至 AWS 服務或 AWS Partner
- 在 Amazon ECS 中使用非AWS 容器映像
- 使用容器重新啟動政策重新啟動 Amazon ECS 任務中的個別容器
- 將敏感資料傳遞至 Amazon ECS 容器

GPU 工作負載的 Amazon ECS 任務定義

您建立具有支援 GPU 容器執行個體的叢集時，Amazon ECS 支援使用 GPU 的工作負載。使用 p2、p3、p5、g3、g4 和 g5 執行個體類型的 Amazon EC2 GPU 型容器執行個體可讓您存取 NVIDIA GPU。如需詳細資訊，請參閱《Amazon EC2 執行個體類型指南》中的 Linux 加速運算執行個體。Amazon EC2

Amazon ECS 提供的 GPU 最佳化 AMI 可與預先設定的 NVIDIA 核心驅動程式和 Docker GPU 執行時間搭配使用。如需詳細資訊，請參閱Amazon ECS 最佳化 Linux AMIs。

您可以在容器定義中指定 GPU 數量，以便在容器層級考量任務放置。Amazon ECS 會對支援 GPU 的可用容器執行個體排程並將實體 GPU 固定至適當容器以獲得最佳效能。

支援下列 Amazon EC2 GPU 型執行個體類型。如需詳細資訊，請參閱 Amazon EC2 P2 執行個體、Amazon EC2 P3 執行個體、Amazon EC2 P4d 執行個體、Amazon EC2 P5 執行個體、Amazon EC2 G3 執行個體、Amazon EC2 G4 執行個體、Amazon EC2 G5 執行個體、Amazon EC2 G6 執行個體和 Amazon EC2 G6e 執行個體。

執行個體類型	GPU	GPU 記憶體 (GiB)	vCPU	記憶體 (GiB)
p3.2xlarge	1	16	8	61
p3.8xlarge	4	64	32	244
p3.16xlarge	8	128	64	488
p3dn.24xlarge	8	256	96	768

執行個體類型	GPU	GPU 記憶體 (GiB)	vCPU	記憶體 (GiB)
p4d.24xlarge	8	320	96	1152
p5.48xlarge	8	640	192	2048
g3s.xlarge	1	8	4	30.5
g3.4xlarge	1	8	16	122
g3.8xlarge	2	16	32	244
g3.16xlarge	4	32	64	488
g4dn.xlarge	1	16	4	16
g4dn.2xlarge	1	16	8	32
g4dn.4xlarge	1	16	16	64
g4dn.8xlarge	1	16	32	128
g4dn.12xlarge	4	64	48	192
g4dn.16xlarge	1	16	64	256
g5.xlarge	1	24	4	16
g5.2xlarge	1	24	8	32
g5.4xlarge	1	24	16	64
g5.8xlarge	1	24	32	128
g5.16xlarge	1	24	64	256
g5.12xlarge	4	96	48	192
g5.24xlarge	4	96	96	384
g5.48xlarge	8	192	192	768

執行個體類型	GPU	GPU 記憶體 (GiB)	vCPU	記憶體 (GiB)
g6.xlarge	1	24	4	16
g6.2xlarge	1	24	8	32
g6.4xlarge	1	24	16	64
g6.8xlarge	1	24	32	128
g6.16.xlarge	1	24	64	256
g6.12xlarge	4	96	48	192
g6.24xlarge	4	96	96	384
g6.48xlarge	8	192	192	768
g6.metal	8	192	192	768
gr6.4xlarge	1	24	16	128
gr6e.xlarge	1	48	4	32
g6e.2xlarge	1	48	8	64
g6e.4xlarge	1	48	16	128
g6e.8xlarge	1	48	32	256
g6e16.xlarge	1	48	64	512
g6e12.xlarge	4	192	48	384
g6e24.xlarge	4	192	96	768
g6e48.xlarge	8	384	192	1536
gr6.8xlarge	1	24	32	256

您可以透過查詢 AWS Systems Manager 參數存放區 API 來擷取 Amazon ECS 最佳化 AMIs的 Amazon Machine Image (AMI) ID。若使用此參數，您無需手動查詢 Amazon ECS 最佳化 AMI ID。如需 Systems Manager 參數存放區 API 的詳細資訊，請參閱 GetParameter。您使用的使用者必須擁有 `ssm:GetParameter` IAM 許可，才能擷取 Amazon ECS 最佳化 AMI 中繼資料。

```
aws ssm get-parameters --names /aws/service/ecs/optimized-ami/amazon-linux-2/gpu/
recommended --region us-east-1
```

考量事項

> **Note**
>
> 對 g2 執行個體系列類型的支援已被棄用。
> p2 執行個體系列類型僅在早於 Amazon ECS GPU 最佳化 AMI 20230912 之前的版本上支援。如果您需要繼續使用 p2 執行個體，請參閱 您需要 P2 執行個體時要採取的動作。
> 在這兩種執行個體系列類型上就地更新 NVIDIA/CUDA 驅動程式，將導致潛在的 GPU 工作負載失敗。

建議您開始在 Amazon ECS 上使用 GPU 前考量下列事項。

- 您的叢集可以混合 GPU 和非 GPU 容器執行個體。
- 您可在外部執行個體上執行 GPU 工作負載。在您叢集中註冊外部執行個體時，請確保安裝指令碼中包含 `--enable-gpu` 標記。如需詳細資訊，請參閱將外部執行個體註冊至 Amazon ECS 叢集。
- 您必須在代理程式組態檔案中將 `ECS_ENABLE_GPU_SUPPORT` 設定為 `true`。如需詳細資訊，請參閱the section called “容器代理程式組態”。
- 執行任務或建立服務時，您可以在設定任務放置限制條件時使用執行個體類型屬性，來決定任務在哪個容器執行個體啟動。這樣便可更有效地使用您的資源。如需詳細資訊，請參閱Amazon ECS 如何在容器執行個體上放置任務。

  以下範例會在預設叢集的 `g4dn.xlarge` 容器執行個體啟動任務。

  ```
  aws ecs run-task --cluster default --task-definition ecs-gpu-task-def \
      --placement-constraints type=memberOf,expression="attribute:ecs.instance-type ==
    g4dn.xlarge" --region us-east-2
  ```

- 對於在容器定義中指定 GPU 資源需求的每個容器，Amazon ECS 將容器執行時間設定為 NVIDIA 容器執行時間。

- NVIDIA 容器執行時間需要在容器中設定一些環境變數才能正常運作。如需這些環境變數的清單，請參閱使用 Docker 的專用組態。Amazon ECS 將 NVIDIA_VISIBLE_DEVICES 環境變數值設定為 Amazon ECS 指派給容器的 GPU 裝置 ID 清單。對於其他必要的環境變數，Amazon ECS 不會對其進行設定。因此，請確保容器映像會設定它們，或在容器定義中設定它們。
- Amazon ECS GPU 最佳化 AMI 版本 20230929 和更新版本支援 p5 執行個體類型系列。
- Amazon ECS GPU 最佳化 AMI 版本 20230913 和更新版本支援 g4 執行個體類型系列。如需詳細資訊，請參閱Amazon ECS 最佳化 Linux AMIs。Amazon ECS 主控台的 Create Cluster (建立叢集) 工作流程中不支援此系列。若要使用這些執行個體類型，您必須使用 Amazon EC2 主控台 AWS CLI 或 API，並將執行個體手動註冊到您的叢集。
- p4d.24xlarge 執行個體類型僅適用於 CUDA 11 或更新版本。
- Amazon ECS GPU 最佳化 AMI 已啟用 IPv6，這會在使用 yum 時導致問題。透過設定 yum 以搭配使用 IPv4 與下列命令，可解決此問題。

```
echo "ip_resolve=4" >> /etc/yum.conf
```

- 在建置一個不使用 NVIDIA/CUDA 基礎映像的容器映像時，必須將 NVIDIA_DRIVER_CAPABILITIES 容器執行時間變數設定為下列其中一個值：
  - utility,compute
  - all

  如需如何設定變數的詳細資訊，請參閱 NVIDIA 網站上的控制 NVIDIA 容器執行時間。
- Windows 容器不支援 GPU。

啟動 Amazon ECS 的 GPU 容器執行個體

若要在 Amazon ECS 上使用 GPU 執行個體，您需要建立啟動範本、使用者資料檔案，以及啟動執行個體。

然後，您可以執行使用為 GPU 設定的任務定義的任務。

使用啟動範本

您可以建立啟動範本。

- 建立使用 Amazon ECS 最佳化 GPU AMI ID for the AMI 的啟動範本。如需有關如何建立啟動範本的資訊，請參閱《Amazon EC2 使用者指南》中的使用您定義的參數建立新的啟動範本。

使用 Amazon Machine 映像上一個步驟的 AMI ID。如需有關如何使用 Systems Manager 參數指定 AMI ID 的資訊，請參閱《Amazon EC2 使用者指南》中的在啟動範本中指定 Systems Manager 參數。

將下列項目新增至啟動範本中的使用者資料。以您的叢集名稱取代 *cluster-name*。

```
#!/bin/bash
echo ECS_CLUSTER=cluster-name >> /etc/ecs/ecs.config;
echo ECS_ENABLE_GPU_SUPPORT=true >> /etc/ecs/ecs.config
```

使用 AWS CLI

您可以使用 AWS CLI 來啟動容器執行個體。

1. 建立稱為 `userdata.toml` 的檔案。此檔案會用於執行個體使用者資料。以您的叢集名稱取代 *cluster-name*。

```
#!/bin/bash
echo ECS_CLUSTER=cluster-name >> /etc/ecs/ecs.config;
echo ECS_ENABLE_GPU_SUPPORT=true >> /etc/ecs/ecs.config
```

2. 執行下列命令以取得 GPU AMI ID。您會在以下步驟中使用此 ID。

```
aws ssm get-parameters --names /aws/service/ecs/optimized-ami/amazon-linux-2/gpu/
recommended --region us-east-1
```

3. 執行下列命令來啟動 GPU 執行個體。請記得替換以下參數：

   - 將*###*替換為執行個體將在其中啟動的私有或公有子網路的 ID。
   - 將 *gpu_ami* 取代為上一個步驟的 AMI ID。
   - 將 *t3.large* 替換為您要使用的執行個體類型。
   - 將 *region* 替換為區域代碼。

```
aws ec2 run-instances --key-name ecs-gpu-example \
   --subnet-id subnet \
   --image-id gpu_ami \
   --instance-type t3.large \
   --region region \
```

```
    --tag-specifications 'ResourceType=instance,Tags=[{Key=GPU,Value=example}]' \
    --user-data file://userdata.toml \
    --iam-instance-profile Name=ecsInstanceRole
```

4. 執行下列命令來驗證容器執行個體是否已註冊至叢集。當您執行此命令時，請記得替代下列參數：

   - 將 *cluster* 替代為叢集名稱。
   - 將 *region* 替換為您的區域代碼。

```
aws ecs list-container-instances --cluster cluster-name --region region
```

在 Amazon ECS 任務定義中指定 GPUs

若要使用容器執行個體的 GPU 和 Docker GPU 執行時間，確保您在任務定義中指定容器所需的 GPU 數量。放置支援 GPU 的容器後，Amazon ECS 容器代理程式會將所需數量的實體 GPU 固定至適當的容器。為任務中所有容器保留的 GPU 數量不可超過任務啟動所在之容器執行個體上可用的 GPU 數量。如需詳細資訊，請參閱使用主控台建立 Amazon ECS 任務定義。

> **Important**
>
> 如果未在任務定義中指定 GPU 要求，該任務會使用預設 Docker 執行時間。

下列顯示任務定義中 GPU 要求的 JSON 格式：

```
{
  "containerDefinitions": [
     {
        ...
        "resourceRequirements" : [
            {
               "type" : "GPU",
               "value" : "2"
            }
        ],
     },
...
}
```

以下範例會示範指定 GPU 要求的 Docker 容器語法。此容器會使用 2 個 GPU，執行 nvidia-smi 公用程式，然後結束。

```
{
  "containerDefinitions": [
    {
      "memory": 80,
      "essential": true,
      "name": "gpu",
      "image": "nvidia/cuda:11.0.3-base",
      "resourceRequirements": [
         {
           "type":"GPU",
           "value": "2"
         }
      ],
      "command": [
        "sh",
        "-c",
        "nvidia-smi"
      ],
      "cpu": 100
    }
  ],
  "family": "example-ecs-gpu"
}
```

共用 GPUs

當您想要多個容器共用 1 個 GPU 時，請將下列使用者資料新增至您的執行個體。如需詳細資訊，請參閱《Amazon EC2 使用者指南》中的使用使用者資料輸入啟動 EC2 執行個體時執行命令。 Amazon EC2

使用最新支援的 GPU 最佳化 AMI

您可以使用 GPU 最佳化 AMI 的 20230906 版本，並將下列項目新增至執行個體使用者資料。

以您的叢集名稱取代 cluster-name。

```
const userData = ec2.UserData.forLinux();
 userData.addCommands(
 'sudo rm /etc/sysconfig/docker',
```

```
 'echo DAEMON_MAXFILES=1048576 | sudo tee -a /etc/sysconfig/docker',
 'echo OPTIONS="--default-ulimit nofile=32768:65536 --default-runtime nvidia" | sudo
 tee -a /etc/sysconfig/docker',
 'echo DAEMON_PIDFILE_TIMEOUT=10 | sudo tee -a /etc/sysconfig/docker',
 'sudo systemctl restart docker',
);
```

您需要 P2 執行個體時要採取的動作

如果您需要使用 P2 執行個體，您可以使用下列其中一個選項來繼續使用執行個體。

您必須修改這兩個選項的執行個體使用者資料。如需詳細資訊，請參閱《Amazon EC2 使用者指南》中的使用使用者資料輸入啟動 EC2 執行個體時執行命令。 Amazon EC2

使用最新支援的 GPU 最佳化 AMI

您可以使用 GPU 最佳化 AMI 的 20230906 版本，並將下列項目新增至執行個體使用者資料。

以您的叢集名稱取代 cluster-name。

```
#!/bin/bash
echo "exclude=*nvidia* *cuda*" >> /etc/yum.conf
echo "ECS_CLUSTER=cluster-name" >> /etc/ecs/ecs.config
```

使用最新的 GPU 最佳化 AMI，並更新使用者資料

您可以將下列內容新增至執行個體使用者資料。這將解除安裝 Nvidia 535/Cuda12.2 驅動程式，然後安裝 Nvidia 470/Cuda11.4 驅動程式並修復該版本。

```
#!/bin/bash
yum remove -y cuda-toolkit* nvidia-driver-latest-dkms*
tmpfile=$(mktemp)
cat >$tmpfile <<EOF
[amzn2-nvidia]
name=Amazon Linux 2 Nvidia repository
mirrorlist=\$awsproto://\$amazonlinux.\$awsregion.\$awsdomain/\$releasever/amzn2-
nvidia/latest/\$basearch/mirror.list
priority=20
gpgcheck=1
gpgkey=https://developer.download.nvidia.com/compute/cuda/repos/rhel7/
x86_64/7fa2af80.pub
enabled=1
exclude=libglvnd-*
```

```
EOF

mv $tmpfile /etc/yum.repos.d/amzn2-nvidia-tmp.repo
yum install -y system-release-nvidia cuda-toolkit-11-4 nvidia-driver-latest-
dkms-470.182.03
yum install -y libnvidia-container-1.4.0 libnvidia-container-tools-1.4.0 nvidia-
container-runtime-hook-1.4.0 docker-runtime-nvidia-1

echo "exclude=*nvidia* *cuda*" >> /etc/yum.conf
nvidia-smi
```

建立您自己的 P2 相容 GPU 最佳化 AMI

您可以建立與 P2 執行個體相容的自訂 Amazon ECS GPU 最佳化 AMI，然後使用 AMI 啟動 P2 執行個體。

1. 執行下列命令以複製 amazon-ecs-ami repo。

   ```
   git clone https://github.com/aws/amazon-ecs-ami
   ```

2. 設定所需的 Amazon ECS 代理程式，並在 release.auto.pkrvars.hcl 或 overrides.auto.pkrvars.hcl 中取得 Amazon Linux AMI 版本。
3. 執行以下命令建立私有 P2 相容的 EC2 AMI。

   將區域替換為具有執行個體區域的區域。

   ```
   REGION=region make al2keplergpu
   ```

4. 將 AMI 與下列執行個體使用者資料搭配使用，以連線至 Amazon ECS 叢集。

   以您的叢集名稱取代 cluster-name。

   ```
   #!/bin/bash
   echo "ECS_CLUSTER=cluster-name" >> /etc/ecs/ecs.config
   ```

視訊轉碼工作負載的 Amazon ECS 任務定義

若要在 Amazon ECS 上使用影片轉碼工作負載，請註冊 Amazon EC2 VT1 執行個體。註冊這些執行個體後，您可以在 Amazon ECS 將即時和預先渲染的影片轉碼工作負載作為任務執行。Amazon EC2 VT1 執行個體使用 Xilinx U30 媒體轉碼卡來加速即時和預先渲染的影片轉碼工作負載。

> **Note**
>
> 如需如何在 Amazon ECS 以外的容器中執行影片轉碼工作負載的說明，請參閱 Xilinx 文件。

考量事項

當您開始在 Amazon ECS 上部署 VT1 之前，請考量下列事項：

- 您的叢集可包含 VT1 和非 VT1 執行個體組合。
- 您需要使用具有加速 AVC (H.264) 和 HEVC (H.265) 解碼器的 Xilinx U30 媒體轉碼卡的 Linux 應用程式。

  > **Important**
  >
  > 使用其他解碼器的應用程式在 VT1 執行個體上可能沒有提升效能。

- U30 卡上只能執行一個轉碼任務。每張卡都有兩個與其關聯的裝置。只要您的每個 VT1 執行個體都有轉碼卡，您就可以執行所需數量的轉碼任務。
- 建立服務或執行獨立任務時，您可以在設定任務置放限制條件時使用執行個體類型屬性。這可確保在您指定的容器執行個體上啟動任務。此舉有助於確保您有效地運用資源，並確保您的影片轉碼工作負載任務位於 VT1 執行個體上。如需詳細資訊，請參閱Amazon ECS 如何在容器執行個體上放置任務。

  在以下範例中，在您的 `default` 叢集的 `vt1.3xlarge` 執行個體上執行任務。

  ```
  aws ecs run-task \
       --cluster default \
       --task-definition vt1-3xlarge-xffmpeg-processor \
       --placement-constraints type=memberOf,expression="attribute:ecs.instance-type ==
   vt1.3xlarge"
  ```

- 您可以設定容器，以便在主機容器執行個體上使用可用的特定 U30 卡。您可使用 `linuxParameters` 參數並指定裝置詳細資訊來實現這一操作。如需詳細資訊，請參閱任務定義需求。

使用 VT1 AMI

您有兩種可選的選項，在 Amazon EC2 上執行 Amazon ECS 容器執行個體的 AMI。第一個選項是使用 AWS Marketplace上的 Xilinx 正式 AMI。第二個選項是從範本儲存庫建置自己的 AMI。

- Xilinx 在 上提供 AMIs AWS Marketplace。
- Amazon ECS 提供了一個範本儲存庫，您可以使用該範本儲存庫為影片轉碼工作負載建置 AMI。此 AMI 隨附 Xilinx U30 驅動程式。您可以在 GitHub 上找到包含 Packer 指令碼的儲存庫。如需 Packer 的詳細資訊，請參閱 Packer documentation (《Packer 文件》)。

任務定義需求

若要在 Amazon ECS 上執行視影片轉碼容器，您的任務定義必須包含使用加速 H.264/AVC 和 H.265/HEVC 解碼器的影片轉碼應用程式。您可以遵循 Xilinx GitHub 的步驟，建置容器映像。

任務定義必須根據執行個體類型專門設定。執行個體類型為 3xlarge、6xlarge 和 24xlarge。
您必須設定容器，以便在主機容器執行個體上使用可用的特定 Xilinx U30 裝置。您可以使用 `linuxParameters` 參數進行該動作。下表詳細說明特定於每種執行個體類型的卡和裝置 SoC。

執行個體類型	vCPU	RAM (GiB)	U30 加速器卡	可定址 XCU30 SoC 裝置	裝置路徑
vt1.3xlarge	12	24	1	2	`/dev/dri/renderD128 ,/dev/dri/renderD129`
vt1.6xlarge	24	48	2	4	`/dev/dri/renderD128 ,/dev/dri/renderD129 ,/dev/dri/renderD13`

執行個體類型	vCPU	RAM (GiB)	U30 加速器卡	可定址 XCU30 SoC 裝置	裝置路徑
					0 ,/dev/dri/renderD131

執行個體類型	vCPU	RAM (GiB)	U30 加速器卡	可定址 XCU30 SoC 裝置	裝置路徑
vt1.24xlarge	96	182	8	16	/dev/dri/renderD128 ,/dev/dri/renderD129 ,/dev/dri/renderD130 ,/dev/dri/renderD131 ,/dev/dri/renderD132 ,/dev/dri/renderD133 ,/dev/dri/renderD134 ,/dev/dri/renderD135 ,/dev/dri/renderD136 ,/dev/dri/renderD137 ,/dev/dri/renderD13

執行個體類型	vCPU	RAM (GiB)	U30 加速器卡	可定址 XCU30 SoC 裝置	裝置路徑
					8 ,/dev/dri/renderD139 ,/dev/dri/renderD140 ,/dev/dri/renderD141 ,/dev/dri/renderD142 ,/dev/dri/renderD143

> **Important**
>
> 如果任務定義列出 EC2 執行個體沒有的裝置，則任務無法執行。當任務失敗時，下列錯誤訊息會出現在 stoppedReason：`CannotStartContainerError: Error response from daemon: error gathering device information while adding custom device "/dev/dri/renderD130": no such file or directory`

在 Amazon ECS 任務定義中指定視訊轉碼

在以下範例中，提供了用於 Amazon EC2 上 Linux 容器的任務定義的語法。此任務定義適用於遵循 Xilinx 文件提供的程序建置的容器映像。如果您使用此範例，請用自己的映像替換 `image`，接著將影片檔案複製到 `/home/ec2-user` 目錄的執行個體。

vt1.3xlarge

1. 使用下列內容建立名為 vt1-3xlarge-ffmpeg-linux.json 的文字檔案。

```
{
    "family": "vt1-3xlarge-xffmpeg-processor",
    "requiresCompatibilities": ["EC2"],
    "placementConstraints": [
        {
            "type": "memberOf",
            "expression": "attribute:ecs.os-type == linux"
        },
        {
            "type": "memberOf",
            "expression": "attribute:ecs.instance-type == vt1.3xlarge"
        }
    ],
    "containerDefinitions": [
        {
            "entryPoint": [
                "/bin/bash",
                "-c"
            ],
            "command": ["/video/ecs_ffmpeg_wrapper.sh"],
            "linuxParameters": {
                "devices": [
                    {
                        "containerPath": "/dev/dri/renderD128",
                        "hostPath": "/dev/dri/renderD128",
                        "permissions": [
                            "read",
                            "write"
                        ]
                    },
                    {
                        "containerPath": "/dev/dri/renderD129",
                        "hostPath": "/dev/dri/renderD129",
                        "permissions": [
                            "read",
                            "write"
                        ]
                    }
                ]
```

```
            },
            "mountPoints": [
                {
                    "containerPath": "/video",
                    "sourceVolume": "video_file"
                }
            ],
            "cpu": 0,
            "memory": 12000,
            "image": "0123456789012.dkr.ecr.us-west-2.amazonaws.com/aws/xilinx-
xffmpeg",
            "essential": true,
            "name": "xilinix-xffmpeg"
        }
    ],
    "volumes": [
        {
            "name": "video_file",
            "host": {"sourcePath": "/home/ec2-user"}
        }
    ]
}
```

2. 註冊任務定義。

```
aws ecs register-task-definition --family vt1-3xlarge-xffmpeg-processor --cli-
input-json file://vt1-3xlarge-xffmpeg-linux.json --region us-east-1
```

vt1.6xlarge

1. 使用下列內容建立名為 vt1-6xlarge-ffmpeg-linux.json 的文字檔案。

```
{
    "family": "vt1-6xlarge-xffmpeg-processor",
    "requiresCompatibilities": ["EC2"],
    "placementConstraints": [
        {
            "type": "memberOf",
            "expression": "attribute:ecs.os-type == linux"
        },
        {
            "type": "memberOf",
```

```
            "expression": "attribute:ecs.instance-type == vt1.6xlarge"
        }
    ],
    "containerDefinitions": [
        {
            "entryPoint": [
                "/bin/bash",
                "-c"
            ],
            "command": ["/video/ecs_ffmpeg_wrapper.sh"],
            "linuxParameters": {
                "devices": [
                    {
                        "containerPath": "/dev/dri/renderD128",
                        "hostPath": "/dev/dri/renderD128",
                        "permissions": [
                            "read",
                            "write"
                        ]
                    },
                    {
                        "containerPath": "/dev/dri/renderD129",
                        "hostPath": "/dev/dri/renderD129",
                        "permissions": [
                            "read",
                            "write"
                        ]
                    },
                    {
                        "containerPath": "/dev/dri/renderD130",
                        "hostPath": "/dev/dri/renderD130",
                        "permissions": [
                            "read",
                            "write"
                        ]
                    },
                    {
                        "containerPath": "/dev/dri/renderD131",
                        "hostPath": "/dev/dri/renderD131",
                        "permissions": [
                            "read",
                            "write"
                        ]
                    }
```

```
                ]
            },
            "mountPoints": [
                {
                    "containerPath": "/video",
                    "sourceVolume": "video_file"
                }
            ],
            "cpu": 0,
            "memory": 12000,
            "image": "0123456789012.dkr.ecr.us-west-2.amazonaws.com/aws/xilinx-
xffmpeg",
            "essential": true,
            "name": "xilinix-xffmpeg"
        }
    ],
    "volumes": [
        {
            "name": "video_file",
            "host": {"sourcePath": "/home/ec2-user"}
        }
    ]
}
```

2. 註冊任務定義。

```
aws ecs register-task-definition --family vt1-6xlarge-xffmpeg-processor --cli-
input-json file://vt1-6xlarge-xffmpeg-linux.json --region us-east-1
```

vt1.24xlarge

1. 使用下列內容建立名為 vt1-24xlarge-ffmpeg-linux.json 的文字檔案。

```
{
    "family": "vt1-24xlarge-xffmpeg-processor",
    "requiresCompatibilities": ["EC2"],
    "placementConstraints": [
        {
            "type": "memberOf",
            "expression": "attribute:ecs.os-type == linux"
        },
        {
```

```
            "type": "memberOf",
            "expression": "attribute:ecs.instance-type == vt1.24xlarge"
        }
    ],
    "containerDefinitions": [
        {
            "entryPoint": [
                "/bin/bash",
                "-c"
            ],
            "command": ["/video/ecs_ffmpeg_wrapper.sh"],
            "linuxParameters": {
                "devices": [
                    {
                        "containerPath": "/dev/dri/renderD128",
                        "hostPath": "/dev/dri/renderD128",
                        "permissions": [
                            "read",
                            "write"
                        ]
                    },
                    {
                        "containerPath": "/dev/dri/renderD129",
                        "hostPath": "/dev/dri/renderD129",
                        "permissions": [
                            "read",
                            "write"
                        ]
                    },
                    {
                        "containerPath": "/dev/dri/renderD130",
                        "hostPath": "/dev/dri/renderD130",
                        "permissions": [
                            "read",
                            "write"
                        ]
                    },
                    {
                        "containerPath": "/dev/dri/renderD131",
                        "hostPath": "/dev/dri/renderD131",
                        "permissions": [
                            "read",
                            "write"
                        ]
```

```
                    },
                    {
                        "containerPath": "/dev/dri/renderD132",
                        "hostPath": "/dev/dri/renderD132",
                        "permissions": [
                            "read",
                            "write"
                        ]
                    },
                    {
                        "containerPath": "/dev/dri/renderD133",
                        "hostPath": "/dev/dri/renderD133",
                        "permissions": [
                            "read",
                            "write"
                        ]
                    },
                    {
                        "containerPath": "/dev/dri/renderD134",
                        "hostPath": "/dev/dri/renderD134",
                        "permissions": [
                            "read",
                            "write"
                        ]
                    },
                    {
                        "containerPath": "/dev/dri/renderD135",
                        "hostPath": "/dev/dri/renderD135",
                        "permissions": [
                            "read",
                            "write"
                        ]
                    },
                    {
                        "containerPath": "/dev/dri/renderD136",
                        "hostPath": "/dev/dri/renderD136",
                        "permissions": [
                            "read",
                            "write"
                        ]
                    },
                    {
                        "containerPath": "/dev/dri/renderD137",
                        "hostPath": "/dev/dri/renderD137",
```

```
                        "permissions": [
                            "read",
                            "write"
                        ]
                    },
                    {
                        "containerPath": "/dev/dri/renderD138",
                        "hostPath": "/dev/dri/renderD138",
                        "permissions": [
                            "read",
                            "write"
                        ]
                    },
                    {
                        "containerPath": "/dev/dri/renderD139",
                        "hostPath": "/dev/dri/renderD139",
                        "permissions": [
                            "read",
                            "write"
                        ]
                    },
                    {
                        "containerPath": "/dev/dri/renderD140",
                        "hostPath": "/dev/dri/renderD140",
                        "permissions": [
                            "read",
                            "write"
                        ]
                    },
                    {
                        "containerPath": "/dev/dri/renderD141",
                        "hostPath": "/dev/dri/renderD141",
                        "permissions": [
                            "read",
                            "write"
                        ]
                    },
                    {
                        "containerPath": "/dev/dri/renderD142",
                        "hostPath": "/dev/dri/renderD142",
                        "permissions": [
                            "read",
                            "write"
                        ]
```

```
                    },
                    {
                        "containerPath": "/dev/dri/renderD143",
                        "hostPath": "/dev/dri/renderD143",
                        "permissions": [
                            "read",
                            "write"
                        ]
                    }
                ]
            },
            "mountPoints": [
                {
                    "containerPath": "/video",
                    "sourceVolume": "video_file"
                }
            ],
            "cpu": 0,
            "memory": 12000,
            "image": "0123456789012.dkr.ecr.us-west-2.amazonaws.com/aws/xilinx-
xffmpeg",
            "essential": true,
            "name": "xilinix-xffmpeg"
        }
    ],
    "volumes": [
        {
            "name": "video_file",
            "host": {"sourcePath": "/home/ec2-user"}
        }
    ]
}
```

2. 註冊任務定義。

```
aws ecs register-task-definition --family vt1-24xlarge-xffmpeg-processor --cli-
input-json file://vt1-24xlarge-xffmpeg-linux.json --region us-east-1
```

AWS Neuron 機器學習工作負載的 Amazon ECS 任務定義

您可將 Amazon EC2 Trn1、Amazon EC2 Inf1 和 Amazon EC2 Inf2 執行個體註冊至叢集，用於機器學習工作負載。

Amazon EC2 Trn1 執行個體採用 AWS Trainium 晶片。這些執行個體為雲端中的機器學習提供高效能和低成本的訓練。您可以在 Trn1 執行個體上使用 AWS Neuron 打造機器學習架構，來訓練機器學習推論模型。然後，您可以在 Inf1 執行個體或 Inf2 執行個體上執行模型，以使用 AWS Inferentia 晶片的加速。

Amazon EC2 Inf1 執行個體和 Inf2 執行個體由 AWS Inferentia 晶片提供支援。這些晶片在雲端提供高效能和最低成本的推論。

機器學習模型使用 AWS Neuron 部署至容器，這是一款特殊軟體開發套件 (SDK)。軟體開發套件包含編譯器、執行時間及分析工具，可最佳化機器學習晶片的 AWS 機器學習效能。 AWS Neuron 支援熱門的機器學習架構，例如 TensorFlow、PyTorch 和 Apache MXNet。

考量事項

當您開始在 Amazon ECS 上部署 Neuron 之前，請考量下列事項：

- 您的叢集可包含 Trn1、Inf1、Inf2 和其他執行個體的組合。
- 您需要容器中的 Linux 應用程式，該容器使用支援 AWS Neuron 的機器學習架構。

> **Important**
>
> 使用其他架構的應用程式在 Trn1、Inf1 和 Inf2 執行個體上的效能可能不會有很大提升。

- 每個 AWS Trainium 或 AWS Inferentia 晶片一次僅可執行一個推論或推論訓練任務。若為 Inf1，每個晶片具有 4 個 NeuronCores。若為 Trn1 和 Inf2，每個晶片則具有 2 個 NeuronCores。每個 Trn1、Inf1 和 Inf2 執行個體上有多少個晶片就可以執行多少個任務。
- 建立服務或執行獨立任務時，您可以在設定任務置放限制條件時使用執行個體類型屬性。這可確保在您指定的容器執行個體上啟動任務。藉此可協助您最佳化整體資源使用率，並確保推論工作負載的任務都位於 Trn1、Inf1 或 Inf2 執行個體上。如需詳細資訊，請參閱Amazon ECS 如何在容器執行個體上放置任務。

  在以下範例中，在您的 `default` 叢集的 `Inf1.xlarge` 執行個體上執行任務。

```
aws ecs run-task \
     --cluster default \
     --task-definition ecs-inference-task-def \
     --placement-constraints type=memberOf,expression="attribute:ecs.instance-type ==
 Inf1.xlarge"
```

- 無法在任務定義中定義 Neuron 資源需求。反之，您可以將容器設定為使用主機容器執行個體上可用的特定 AWS Trainium 或 AWS Inferentia 晶片。您可使用 `linuxParameters` 參數並指定裝置詳細資訊來執行此動作。如需詳細資訊，請參閱任務定義需求。

使用 Amazon ECS 最佳化 Amazon Linux 2023 (Neuron) AMI

Amazon ECS 提供以 Amazon Linux 2023 為基礎的 Amazon ECS 最佳化 AMI，適用於 AWS Trainium 和 AWS Inferentia 工作負載。它隨附 Docker 的 AWS Neuron 驅動程式和執行期。此 AMI 使得在 Amazon ECS 上執行機器學習推論工作負載更輕鬆。

我們建議您在啟動 Amazon EC2 Trn1, Inf1 和 Inf2 執行個體時使用 Amazon Amazon EC2 Linux 2023 (Neuron) AMI。

您可以使用 AWS CLI 搭配下列命令來擷取目前的 Amazon ECS 最佳化 Amazon Linux 2023 (Neuron) AMI。

```
aws ssm get-parameters --names /aws/service/ecs/optimized-ami/amazon-linux-2023/neuron/recommended
```

下列區域支援 Amazon ECS 最佳化 Amazon Linux 2023 (Neuron) AMI：

- 美國東部 (維吉尼亞北部)
- 美國東部 (俄亥俄)
- 美國西部 (加利佛尼亞北部)
- 美國西部 (奧勒岡)
- 亞太區域 (孟買)
- 亞太區域 (大阪)
- 亞太區域 (首爾)
- 亞太區域 (東京)
- 亞太區域 (新加坡)
- 亞太區域 (悉尼)
- 加拿大 (中部)
- 歐洲 (法蘭克福)
- 歐洲 (愛爾蘭)
- 歐洲 (倫敦)

- 歐洲 (巴黎)
- 歐洲 (斯德哥爾摩)
- 南美洲 (聖保羅)

使用 Amazon ECS 最佳化 Amazon Linux 2 (Neuron) AMI

Amazon ECS 提供以 Amazon Linux 2 為基礎的 Amazon ECS 最佳化 AMI，適用於 AWS Trainium 和 AWS Inferentia 工作負載。它隨附 Docker 的 AWS Neuron 驅動程式和執行期。此 AMI 使得在 Amazon ECS 上執行機器學習推論工作負載更輕鬆。

建議在啟動 Amazon EC2 Trn1、Inf1 和 Inf2 執行個體時，使用 Amazon ECS 最佳化 Amazon Linux 2 (Neuron) AMI。

您可以使用 AWS CLI 搭配下列命令，擷取目前的 Amazon ECS 最佳化 Amazon Linux 2 (Neuron) AMI。

```
aws ssm get-parameters --names /aws/service/ecs/optimized-ami/amazon-linux-2/inf/
recommended
```

下列區域支援 Amazon ECS 最佳化 Amazon Linux 2 (Neuron) AMI：

- 美國東部 (維吉尼亞北部)
- 美國東部 (俄亥俄)
- 美國西部 (加利佛尼亞北部)
- 美國西部 (奧勒岡)
- 亞太區域 (孟買)
- 亞太區域 (大阪)
- 亞太區域 (首爾)
- 亞太區域 (東京)
- 亞太區域 (新加坡)
- 亞太區域 (悉尼)
- 加拿大 (中部)
- 歐洲 (法蘭克福)
- 歐洲 (愛爾蘭)
- 歐洲 (倫敦)

- 歐洲 (巴黎)
- 歐洲 (斯德哥爾摩)
- 南美洲 (聖保羅)

任務定義需求

要在 Amazon ECS 上部署 Neuron，您的任務定義必須包含預先構建的容器的容器定義，該容器為TensorFlow 提供推論模型。它由 AWS 深度學習容器提供。此容器包含 AWS Neuron 執行期和 TensorFlow Serving 應用程式。啟動時，此容器會從 Amazon S3 中擷取您的模型、使用儲存的模型啟動 Neuron TensorFlow Serving，並等待預測請求。在以下範例中，容器映像擁有 TensorFlow 1.15 和 Ubuntu 18.04。在 GitHub 上維護為 Neuron 最佳化的預先建置的 Deep Learning Containers 完整清單。如需詳細資訊，請參閱使用 AWS Neuron TensorFlow Serving。

```
763104351884.dkr.ecr.us-east-1.amazonaws.com/tensorflow-inference-neuron:1.15.4-neuron-py37-ubuntu18.04
```

或者，您可以建置自己的 Neuron 附屬容器映像。如需詳細資訊，請參閱《AWS 深度學習 AMIs 開發人員指南》中的教學課程：Neuron TensorFlow Serving。

任務定義必須根據單一執行個體類型專門設定。您必須將容器設定為使用主機容器執行個體上可用的特定 AWS Trainium 或 AWS Inferentia 裝置。您可以使用 `linuxParameters` 參數進行該動作。下表詳細說明特定於每種執行個體類型的晶片。

執行個體類型	vCPU	RAM (GiB)	AWS ML 加速器晶片	裝置路徑
trn1.2xlarge	8	32	1	`/dev/neuron0`
trn1.32xlarge	128	512	16	`/dev/neuron0` ,`/dev/neuron1` ,`/dev/neuron2` ,`/dev/neuron3` ,`/dev/neuron4` ,`/dev/neuron5` ,

執行個體類型	vCPU	RAM (GiB)	AWS ML 加速器晶片	裝置路徑
				/dev/neuron6 ,/dev/neuron7 ,/dev/neuron8 ,/dev/neuron9 ,/dev/neuron10 ,/dev/neuron11 ,/dev/neuron12 ,/dev/neuron13 ,/dev/neuron14 ,/dev/neuron15
inf1.xlarge	4	8	1	/dev/neuron0
inf1.2xlarge	8	16	1	/dev/neuron0
inf1.6xlarge	24	48	4	/dev/neuron0 ,/dev/neuron1 ,/dev/neuron2 ,/dev/neuron3

執行個體類型	vCPU	RAM (GiB)	AWS ML 加速器晶片	裝置路徑
inf1.24xlarge	96	192	16	/dev/neuron0 ,/dev/neuron1 ,/dev/neuron2 ,/dev/neuron3 ,/dev/neuron4 ,/dev/neuron5 ,/dev/neuron6 ,/dev/neuron7 ,/dev/neuron8 ,/dev/neuron9 ,/dev/neuron10 ,/dev/neuron11 ,/dev/neuron12 ,/dev/neuron13 ,/dev/neuron14 ,/dev/neuron15
inf2.xlarge	8	16	1	/dev/neuron0
inf2.8xlarge	32	64	1	/dev/neuron0

執行個體類型	vCPU	RAM (GiB)	AWS ML 加速器晶片	裝置路徑
inf2.24xlarge	96	384	6	/dev/neuron0 ,/dev/neuron1 ,/dev/neuron2 ,/dev/neuron3 ,/dev/neuron4 ,/dev/neuron5 ,
inf2.48xlarge	192	768	12	/dev/neuron0 ,/dev/neuron1 ,/dev/neuron2 ,/dev/neuron3 ,/dev/neuron4 ,/dev/neuron5 ,/dev/neuron6 ,/dev/neuron7 ,/dev/neuron8 ,/dev/neuron9 ,/dev/neuron10 ,/dev/neuron11

在 Amazon ECS 任務定義中指定 AWS Neuron 機器學習

以下是 `inf1.xlarge` 的 Linux 任務定義範例，顯示要使用的語法。

```
{
    "family": "ecs-neuron",
    "requiresCompatibilities": ["EC2"],
```

```
    "placementConstraints": [
        {
            "type": "memberOf",
            "expression": "attribute:ecs.os-type == linux"
        },
        {
            "type": "memberOf",
            "expression": "attribute:ecs.instance-type == inf1.xlarge"
        }
    ],
    "executionRoleArn": "${YOUR_EXECUTION_ROLE}",
    "containerDefinitions": [
        {
            "entryPoint": [
                "/usr/local/bin/entrypoint.sh",
                "--port=8500",
                "--rest_api_port=9000",
                "--model_name=resnet50_neuron",
                "--model_base_path=s3://amzn-s3-demo-bucket/resnet50_neuron/"
            ],
            "portMappings": [
                {
                    "hostPort": 8500,
                    "protocol": "tcp",
                    "containerPort": 8500
                },
                {
                    "hostPort": 8501,
                    "protocol": "tcp",
                    "containerPort": 8501
                },
                {
                    "hostPort": 0,
                    "protocol": "tcp",
                    "containerPort": 80
                }
            ],
            "linuxParameters": {
                "devices": [
                    {
                        "containerPath": "/dev/neuron0",
                        "hostPath": "/dev/neuron0",
                        "permissions": [
                            "read",
```

```
                            "write"
                        ]
                    }
                ],
                "capabilities": {
                    "add": [
                        "IPC_LOCK"
                    ]
                }
            },
            "cpu": 0,
            "memoryReservation": 1000,
            "image": "763104351884.dkr.ecr.us-east-1.amazonaws.com/tensorflow-
inference-neuron:1.15.4-neuron-py37-ubuntu18.04",
            "essential": true,
            "name": "resnet50"
        }
    ]
}
```

適用於深度學習執行個體的 Amazon ECS 任務定義

若要在 Amazon ECS 上使用深度學習工作負載，請將 Amazon EC2 DL1 執行個體註冊到您的叢集。Amazon EC2 DL1 執行個體由 Habana 實驗室 (Intel 公司) 的 Gaudi 加速器提供。使用 Habana SynapseAI SDK 連線到 Habana Gaudi 加速器。SDK 支援流行的機器學習架構、TensorFlow 和 PyTorch。

考量事項

當您開始在 Amazon ECS 上部署 DL1 之前，請考量下列事項：

- 您的叢集可包含 DL1 和非 DL1 執行個體組合。
- 建立服務或執行獨立任務時，您可以在設定任務置放限制條件時使用執行個體類型屬性，以確定任務於指定的容器執行個體啟動。藉此可確保您的資源得到有效利用，並確保深度學習工作負載的任務位於 DL1 執行個體上。如需詳細資訊，請參閱Amazon ECS 如何在容器執行個體上放置任務。

  以下範例在 `default` 叢集的 `dl1.24xlarge` 執行個體上執行任務。

```
aws ecs run-task \
     --cluster default \
     --task-definition ecs-dl1-task-def \
```

```
    --placement-constraints type=memberOf,expression="attribute:ecs.instance-type ==
dl1.24xlarge"
```

使用 DL1 AMI

對於在 Amazon EC2 DL1 執行個體上執行 Amazon ECS 的 AMI，您有三個選項：

- Habana AWS Marketplace AMIshttps://aws.amazon.com/marketplace/pp/prodview-h24gzbgqu75zq。
- 由 Amazon Web Services 提供的 Habana 深度學習 AMI。由於其不包含在內，您需要單獨安裝 Amazon ECS 容器代理程式。
- 使用 Packer 建置由 GitHub 儲存庫提供的自訂 AMI。如需詳細資訊，請參閱 Packer documentation (《Packer 文件》)。

在 Amazon ECS 任務定義中指定深度學習

若要在 Amazon ECS 上執行 Habana Gaudi 加速的深度學習容器，您的任務定義必須包含預先建置容器的容器定義，該容器使用 AWS 深度學習容器提供的 Habana SynapseAI 為 TensorFlow 或 PyTorch 提供深度學習模型。

以下容器映像擁有 TensorFlow 2.7.0 和 Ubuntu 20.04。在 GitHub 上維護為 Habana Gaudi 加速器最佳化的預先建置的 Deep Learning Containers 完整清單。如需詳細資訊，請參閱 Habana Training Containers (Habana 訓練容器)。

```
763104351884.dkr.ecr.us-east-1.amazonaws.com/tensorflow-training-habana:2.7.0-hpu-py38-
synapseai1.2.0-ubuntu20.04
```

以下是 Amazon EC2 上的 Linux 容器任務定義範例，顯示了要使用的語法。此範例使用包含 Habana 實驗室系統管理介面工具 (HL-SMI) 的映像，請參閱：`vault.habana.ai/gaudi-docker/1.1.0/ubuntu20.04/habanalabs/tensorflow-installer-tf-cpu-2.6.0:1.1.0-614`

```
{
    "family": "dl-test",
    "requiresCompatibilities": ["EC2"],
    "placementConstraints": [
        {
            "type": "memberOf",
```

```
            "expression": "attribute:ecs.os-type == linux"
        },
        {
            "type": "memberOf",
            "expression": "attribute:ecs.instance-type == dl1.24xlarge"
        }
    ],
    "networkMode": "host",
    "cpu": "10240",
    "memory": "1024",
    "containerDefinitions": [
        {
            "entryPoint": [
                "sh",
                "-c"
            ],
            "command": ["hl-smi"],
            "cpu": 8192,
            "environment": [
                {
                    "name": "HABANA_VISIBLE_DEVICES",
                    "value": "all"
                }
            ],
            "image": "vault.habana.ai/gaudi-docker/1.1.0/ubuntu20.04/habanalabs/
tensorflow-installer-tf-cpu-2.6.0:1.1.0-614",
            "essential": true,
            "name": "tensorflow-installer-tf-hpu"
        }
    ]
}
```

適用於 64 位元 ARM 工作負載的 Amazon ECS 任務定義

Amazon ECS 支援使用 64 位元 ARM 應用程式。您可以在採用 AWS Graviton 處理器的平台上執行應用程式。該平台適用多種工作負載。其包括各種工作負載，例如應用程式伺服器、微型服務、高效能運算、CPU 型機器學習推論、影片編碼、電子設計自動化、遊戲、開放原始碼資料庫和記憶體內快取。

考量事項

在您開始部署使用 64 位元 ARM 架構的任務定義之前，請考量下列事項：

- 應用程式可使用 Fargate 或 EC2 啟動類型。

- 應用程式僅能使用 Linux 作業系統。
- 對於 Fargate 類型，應用程式必須使用 Fargate 平台版本 `1.4.0` 或更新版本。
- 應用程式可使用 Fluent Bit 或 CloudWatch 進行監控。
- 對於 Fargate 啟動類型，以下 AWS 區域 不支援 64 位元 ARM 工作負載：
  - 美國東部 (維吉尼亞北部)、`use1-az3` 可用區域
- 對於 Amazon EC2 啟動類型，請參閱以下內容，以驗證您所在區域是否支援您要使用的執行個體類型：
  - Amazon EC2 M6g 執行個體
  - Amazon EC2 T4g 執行個體
  - Amazon EC2 C6g 執行個體
  - Amazon EC2 R6gd 執行個體
  - Amazon EC2 X2gd 執行個體

  您也可以使用 Amazon EC2 `describe-instance-type-offerings` 命令搭配篩選條件，查看您所在區域的執行個體優惠。

  ```
  aws ec2 describe-instance-type-offerings --filters Name=instance-
  type,Values=instance-type --region region
  ```

  以下範例檢查美國東部 (維吉尼亞北部) (us-east-1) 區域中的 M6 執行個體類型可用性。

  ```
  aws ec2 describe-instance-type-offerings --filters "Name=instance-type,Values=m6*" --
  region us-east-1
  ```

  如需詳細資訊，請參閱《Amazon EC2 命令列參考》中的 describe-instance-type-offerings。

在 Amazon ECS 任務定義中指定 ARM 架構

若要使用 ARM 架構，請為 `cpuArchitecture` 任務定義參數指定 `ARM64`。

在以下範例中，ARM 架構是在任務定義中指定的。其為 JSON 格式。

```
{
    "runtimePlatform": {
        "operatingSystemFamily": "LINUX",
        "cpuArchitecture": "ARM64"
```

```
    },
...
}
```

以下為顯示"hello world" 的 ARM 架構的任務定義範例。

```
{
 "family": "arm64-testapp",
 "networkMode": "awsvpc",
 "containerDefinitions": [
    {
        "name": "arm-container",
        "image": "arm64v8/busybox",
        "cpu": 100,
        "memory": 100,
        "essential": true,
        "command": [ "echo hello world" ],
        "entryPoint": [ "sh", "-c" ]
    }
 ],
 "requiresCompatibilities": [ "EC2" ],
 "cpu": "256",
 "memory": "512",
 "runtimePlatform": {
        "operatingSystemFamily": "LINUX",
        "cpuArchitecture": "ARM64"
  },
 "executionRoleArn": "arn:aws:iam::123456789012:role/ecsTaskExecutionRole"
}
```

將 Amazon ECS 日誌傳送至 CloudWatch

您可以在任務中設定容器，將日誌資訊傳送給 CloudWatch Logs。如果您為任務使用 Fargate 啟動類型，您可檢視容器的日誌。如果您使用 EC2 啟動類型，您可在單一便利位置檢視容器的不同日誌，並防止容器日誌佔用容器執行個體的磁碟空間。

> **Note**
>
> 任務中容器所記錄的資訊類型，絕大部分取決於其 ENTRYPOINT 命令。在預設情況下，擷取的日誌會顯示您在本機執行容器時，通常會在互動式終端機中看見的命令輸出，其為 STDOUT 和 STDERR I/O 串流。awslogs 日誌驅動程式只會將這些日誌從 Docker 傳遞至 CloudWatch

Logs。如需 Docker 日誌處理方式 (包括擷取不同檔案資料或串流的替代方法) 的詳細資訊，請參閱 Docker 文件中的檢視容器或服務的日誌。

若要將系統日誌從 Amazon ECS 容器執行個體傳送至 CloudWatch Logs，請參閱《Amazon CloudWatch Logs 使用者指南》中的監控日誌檔案和 CloudWatch Logs 配額。

Fargate 啟動類型

如果您對任務使用 Fargate 啟動類型，您需將必要的 `logConfiguration` 參數新增至任務定義，以開啟 `awslogs` 日誌驅動程式。如需詳細資訊，請參閱Amazon ECS 任務定義範例：將日誌路由至 CloudWatch。

針對 Fargate 上的 Windows 容器，當您的任何任務定義參數具有特殊字元時，請執行下列其中一個選項，例如 `& \ < > ^ |`：

- 新增逸出 (\)，並在整個參數字串周圍加上雙引號

  範例

  ```
  "awslogs-multiline-pattern": "\"^[|DEBUG|INFO|WARNING|ERROR\"",
  ```

- 在每個特殊字元周圍新增逸出 (^) 字元

  範例

  ```
  "awslogs-multiline-pattern": "^^[^|DEBUG^|INFO^|WARNING^|ERROR",
  ```

EC2 啟動類型

如果您對任務使用 EC2 啟動類型，並且想要開啟 `awslogs` 日誌驅動程式，則 Amazon ECS 容器執行個體需要至少 1.9.0 版的容器代理程式。如需如何檢查代理程式版本及更新至最新版本的資訊，請參閱「更新 Amazon ECS 容器代理程式」。

> **Note**
>
> 您必須使用 Amazon ECS 最佳化 AMI 或自訂 AMI，至少具有 `ecs-init` 套件`1.9.0-1`的 版本。使用自訂 AMI 時，您必須在 docker run 陳述式或環境變數檔案中使用以下環境變數啟動代理程式時，指定 Amazon EC2 執行個體上的`awslogs`記錄驅動程式可供使用。

```
ECS_AVAILABLE_LOGGING_DRIVERS=["json-file","awslogs"]
```

您的 Amazon ECS 容器執行個體也需要可用來啟動容器執行個體之 IAM 角色的 `logs:CreateLogStream` 和 `logs:PutLogEvents` 許可。在 Amazon ECS 中啟用 `awslogs` 日誌驅動程式支援前，如果您已建立 Amazon ECS 容器執行個體角色，您可能需要新增此許可。`ecsTaskExecutionRole` 在其被指派給任務時使用，且應該包含正確的許可。如需任務執行角色的相關資訊，請參閱 Amazon ECS 任務執行 IAM 角色。如果您的容器執行個體使用容器執行個體的受管 IAM 政策，則您的容器執行個體應該具有正確的許可。如需容器執行個體受管 IAM 政策的相關資訊，請參閱 Amazon ECS 容器執行個體 IAM 角色。

Amazon ECS 任務定義範例：將日誌路由至 CloudWatch

您必須先在任務定義中指定容器的 `awslogs` 日誌驅動程式，容器才能將日誌傳送至 CloudWatch。如需日誌參數的詳細資訊，請參閱 儲存與記錄

隨後的任務定義 JSON 具有為每個容器指定的 `logConfiguration` 物件。一個是用於 WordPress 容器，該容器將日誌發送到名為 `awslogs-wordpress` 的日誌群組。另一個用於 MySQL 容器，該容器將日誌發送到名為 `awslogs-mysql` 的日誌群組。兩個容器使用的日誌串流前綴皆為 `awslogs-example`。

```
{
    "containerDefinitions": [
        {
            "name": "wordpress",
            "links": [
                "mysql"
            ],
            "image": "wordpress",
            "essential": true,
            "portMappings": [
                {
                    "containerPort": 80,
                    "hostPort": 80
                }
            ],
            "logConfiguration": {
                "logDriver": "awslogs",
                "options": {
```

```
                    "awslogs-create-group": "true",
                    "awslogs-group": "awslogs-wordpress",
                    "awslogs-region": "us-west-2",
                    "awslogs-stream-prefix": "awslogs-example"
                }
            },
            "memory": 500,
            "cpu": 10
        },
        {
            "environment": [
                {
                    "name": "MYSQL_ROOT_PASSWORD",
                    "value": "password"
                }
            ],
            "name": "mysql",
            "image": "mysql",
            "cpu": 10,
            "memory": 500,
            "essential": true,
            "logConfiguration": {
                "logDriver": "awslogs",
                "options": {
                    "awslogs-create-group": "true",
                    "awslogs-group": "awslogs-mysql",
                    "awslogs-region": "us-west-2",
                    "awslogs-stream-prefix": "awslogs-example",
                    "mode": "non-blocking",
                    "max-buffer-size": "25m"
                }
            }
        }
    ],
    "family": "awslogs-example"
}
```

後續步驟

- 您可以使用 CloudWatch AWS CLI 或 API，選擇性地設定日誌群組的保留政策。如需詳細資訊，請參閱 AWS Command Line Interface 參考中的 put-retention-policy。

- 當您在容器定義日誌組態中將任務定義註冊到 awslogs 日誌驅動程式後，就可以執行任務，或使用該任務定義建立服務，以開始將日誌傳送至 CloudWatch Logs。如需詳細資訊，請參閱 以 Amazon ECS 任務執行應用程式 和 使用 主控台建立 Amazon ECS 服務。

將 Amazon ECS 日誌傳送至 AWS 服務或 AWS Partner

您可以使用 FireLens for Amazon ECS 來使用任務定義參數，將日誌路由至 AWS 服務或 AWS Partner Network (APN) 目的地，以進行日誌儲存和分析。 AWS Partner Network 是一個全球合作夥伴社群，利用計劃、專業知識和資源來建置、行銷和銷售客戶產品。如需詳細資訊，請參閱 AWS Partner。FireLens 使用 Fluentd 和 Fluent Bit。我們提供 AWS for Fluent Bit 映像，或者您也可以使用自己的 Fluentd 或 Fluent Bit 映像。

根據預設，Amazon ECS 會設定容器相依性，讓 Firelens 容器在使用該容器的任何容器之前啟動。Firelens 容器也會在使用它的所有容器停止後停止。

使用 FireLens for Amazon ECS 時，請考量下列事項：

- 建議您將 my_service_ 新增至日誌容器名稱，以便在 主控台中輕鬆區分容器名稱。
- 根據預設，Amazon ECS 會在應用程式容器與FireLens容器之間新增啟動容器順序相依性。當您在應用程式容器與FireLens容器之間指定容器順序時，預設的啟動容器順序會遭到覆寫。
- 託管於 Linux 上的 AWS Fargate 和 Linux 上的 Amazon EC2 的任務支援 Amazon ECS 的 FireLens。Windows 容器不支援 FireLens。

  如需如何為 Windows 容器設定集中式記錄的相關資訊，請參閱 Centralized logging for Windows containers on Amazon ECS using Fluent Bit (《使用 Fluent Bit 為 Amazon ECS 上的 Windows 容器設定集中式記錄》)。
- 您可以使用 AWS CloudFormation 範本來設定 FireLens Amazon ECS。如需詳細資訊，請參閱《AWS CloudFormation 使用者指南》中的 AWS::ECS::TaskDefinition FirelensConfiguration。
- FireLens 在連接埠上監聽 24224，因此為了確保 FireLens 日誌路由器無法在任務之外連線，您不能在允許任務使用的安全群組中連接埠 24224 上的傳入流量。對於使用 awsvpc 網路模式的任務，這是與任務相關聯的安全群組。對於使用 host 網路模式的任務，這是與託管任務的 Amazon EC2 執行個體相關聯的安全群組。對於使用 bridge 網路模式的任務，請不要建立使用連接埠 24224 的任何連接埠映射。
- 對於使用 bridge 網路模式的工作，具有 FireLens 配置的容器必須在任何依賴該模式的應用程式容器啟動之前啟動。若要控制容器的起始順序，請在工作定義中使用相依性條件。如需詳細資訊，請參閱容器相依性。

> **Note**
>
> 如果您在使用 FireLens 配置的容器定義中使用相依性條件參數，請確定每個容器都有 START 或 HEALTHY 條件需求。

- 根據預設，FireLens 將叢集和任務定義名稱以及叢集的 Amazon Resource Name (ARN) 作為中繼資料索引鍵新增到 stout/stderr 容器紀錄。以下是中繼資料格式的範例。

```
"ecs_cluster": "cluster-name",
"ecs_task_arn": "arn:aws:ecs:region:111122223333:task/cluster-
name/f2ad7dba413f45ddb4EXAMPLE",
"ecs_task_definition": "task-def-name:revision",
```

  如果您不想在紀錄中使用中繼資料，請在任務定義 `firelensConfiguration` 部分中將 `false` 設定為 `enable-ecs-log-metadata`。

```
"firelensConfiguration":{
   "type":"fluentbit",
   "options":{
      "enable-ecs-log-metadata":"false",
      "config-file-type":"file",
      "config-file-value":"/extra.conf"
}
```

若要使用此功能，您必須為任務建立 IAM 角色，提供使用任務所需的任何 AWS 服務所需的許可。例如，如果容器正在將日誌路由到 Firehose，任務需要呼叫 `firehose:PutRecordBatch` API 的許可。如需詳細資訊，請參閱 《IAM 使用者指南》中的新增和移除 IAM 身分許可。

在下列情況下，您的任務也可能需要 Amazon ECS 任務執行角色。如需詳細資訊，請參閱Amazon ECS 任務執行 IAM 角色。

- 如果您的任務託管在 Fargate 上，而且您要從 Amazon ECR 提取容器映像，或在日誌組態 AWS Secrets Manager 中參考來自 的敏感資料，則必須包含任務執行 IAM 角色。
- 當您使用 Amazon S3 中託管的自訂組態檔案時，您的任務執行 IAM 角色必須包含 `s3:GetObject`許可。

如需如何搭配 Amazon ECS 使用多個組態檔案的資訊，包括您在 Amazon S3 中託管的檔案或檔案，請參閱 ECS 上的 Fluent Bit 啟動程序、多組態支援。

設定高輸送量的 Amazon ECS 日誌

建立任務定義時，您可以透過在 中指定 值，來指定在記憶體中緩衝的日誌行數`log-driver-buffer-limit`。如需詳細資訊，請參閱 Docker 文件中的 Fluentd 登入驅動程式。

當輸送量高時，請使用此選項，因為 Docker 可能會耗盡緩衝記憶體並捨棄緩衝訊息，因此可以新增訊息。

使用 FireLens for Amazon ECS 搭配緩衝區限制選項時，請考量下列 ：

- Amazon EC2 啟動類型和具有平台版本的 `1.4.0` 或更新版本的 Fargate 啟動類型支援此選項。
- 只有在 `logDriver` 設定為 `awsfirelens` 時，此選項才有效。
- 預設緩衝限制為`1048576`日誌行。
- 緩衝區限制必須大於或等於 `0` 且小於`536870912`日誌行。
- 此緩衝區使用的記憶體數量上限是每個日誌行大小和緩衝區大小的乘積。例如，如果應用程式的日誌行是平均 2 KiB，緩衝區限制為 4096 最多會使用 8 MiB。除了日誌驅動程式記憶體緩衝區之外，在任務層級配置的記憶體總量應大於為所有容器配置的記憶體量。

在任務定義中指定 `awsfirelens` 日誌驅動程式時，Amazon ECS 代理程式會將下列環境變數插入容器中：

`FLUENT_HOST`

指派給 FireLens 容器的 IP 位址。

> **Note**
>
> 如果您使用 EC2 啟動類型搭配`bridge`網路模式，應用程式容器中`FLUENT_HOST`的環境變數可能會在重新啟動 FireLens 日誌路由器容器 （ 容器定義中具有`firelensConfiguration`物件的容器） 之後變得不準確。這是因為 `FLUENT_HOST` 是動態 IP 地址，重新啟動後可能會變更。地址變更後，直接從應用程式容器記錄到 `FLUENT_HOST` IP 地址可能會開始失敗。如需重新啟動個別容器的詳細資訊，請參閱使用容器重新啟動政策重新啟動 Amazon ECS 任務中的個別容器。

FLUENT_PORT

流利轉送通訊協定正在接聽的連接埠。

您可以使用 FLUENT_HOST 和 FLUENT_PORT 環境變數從程式碼直接登入日誌路由器，而不是透過 stdout。如需詳細資訊，請參閱 GitHub 上的 流利記錄器 golang 。

以下顯示指定 的語法log-driver-buffer-limit。my_service_ 將 取代為您服務的名稱：

```
{
    "containerDefinitions": [
        {
            "name": "my_service_log_router",
            "image": "public.ecr.aws/aws-observability/aws-for-fluent-bit:stable",
            "cpu": 0,
            "memoryReservation": 51,
            "portMappings": [],
            "essential": true,
            "environment": [],
            "mountPoints": [],
            "volumesFrom": [],
            "user": "0",
            "logConfiguration": {
                "logDriver": "awslogs",
                "options": {
                    "awslogs-group": "/ecs/ecs-aws-firelens-sidecar-container",
                    "mode": "non-blocking",
                    "awslogs-create-group": "true",
                    "max-buffer-size": "25m",
                    "awslogs-region": "us-east-1",
                    "awslogs-stream-prefix": "firelens"
                },
                "secretOptions": []
            },
            "systemControls": [],
            "firelensConfiguration": {
                "type": "fluentbit"
            }
        },
        {
            "essential": true,
            "image": "httpd",
            "name": "app",
```

```
            "logConfiguration": {
                "logDriver": "awsfirelens",
                "options": {
                    "Name": "firehose",
                    "region": "us-west-2",
                    "delivery_stream": "my-stream",
                    "log-driver-buffer-limit": "51200"
                }
            },
            "dependsOn": [
                {
                    "containerName": "log_router",
                    "condition": "START"
                }
            ],
            "memoryReservation": 100
        }
    ]
}
```

AWS 適用於 Amazon ECS Fluent Bit的影像儲存庫

AWS 為 CloudWatch Logs 和 Firehose 提供具有外掛程式Fluent Bit的影像。我們建議您將 Fluent Bit 用作日誌路由器，因為它的資源使用率低於 Fluentd。如需詳細資訊，請參閱 CloudWatch Logs for Fluent Bit 和 Amazon Kinesis Firehose for Fluent Bit。

AWS 適用於 Fluent Bit 的 映像可在 Amazon ECR 公有圖庫和大多數的 Amazon ECR 儲存庫上的 Amazon ECR 上使用， AWS 區域 以實現高可用性。

Amazon ECR Public Gallery

AWS 適用於Fluent Bit映像的 可在 Amazon ECR 公有圖庫上取得。這是下載 AWS 適用於Fluent Bit映像的 的建議位置，因為它是公有儲存庫，可從所有 使用 AWS 區域。如需詳細資訊，請參閱 Amazon ECR Public Gallery 上的 aws-for-fluent-bit。

Linux

Amazon ECR Public Gallery 中的 AWS for Fluent Bit映像支援具有 `ARM 64`、 或 `x86-64`架構的 Amazon Linux 作業系統。

您可以從 Amazon ECR Public Gallery 中提取 AWS 適用於Fluent Bit映像的 ，方法是使用所需的映像標籤指定儲存庫 URL。在 Amazon ECR Public Gallery 的映像標籤索引標籤中可找到可用的映像標籤。

以下顯示 Docker CLI 要使用的語法。

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

例如，您可以使用此 Docker AWS CLI 命令提取影像的最新穩定Fluent Bit。

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

> **Note**
>
> 允許未經驗證的提取，但速率限制低於已驗證的提取。若要在提取之前使用 AWS 您的帳戶進行身分驗證，請使用下列命令。
>
> ```
> aws ecr-public get-login-password --region us-east-1 | docker login --username
>  AWS --password-stdin public.ecr.aws
> ```

Windows

Amazon ECR Public Gallery 中的 AWS for Fluent Bit image 支援具有下列作業系統的AMD64架構：

- Windows Server 2022 Full
- Windows Server 2022 Core
- Windows Server 2019 Full
- Windows Server 2019 Core

AWS Fargate 上的 Windows 容器不支援 FireLens。

您可以從 Amazon ECR Public Gallery 中提取 AWS 適用於Fluent Bit映像的 ，方法是使用所需的映像標籤指定儲存庫 URL。在 Amazon ECR Public Gallery 的映像標籤索引標籤中可找到可用的映像標籤。

以下顯示 Docker CLI 要使用的語法。

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

例如，您可以使用此 Docker CLI 命令 AWS ，為Fluent Bit映像提取最新穩定的 。

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

> **Note**
>
> 允許未經驗證的提取，但速率限制低於已驗證的提取。若要在提取之前使用 AWS 您的帳戶進行身分驗證，請使用下列命令。
>
> ```
> aws ecr-public get-login-password --region us-east-1 | docker login --username
>  AWS --password-stdin public.ecr.aws
> ```

Amazon ECR

AWS 適用於 Fluent Bit 的 映像可在 Amazon ECR 上取得，以獲得高可用性。這些映像大部分都可用 AWS 區域，包括 AWS GovCloud (US)。

Linux

最新的 Fluent Bit 影像 URI AWS 穩定狀態，可以使用下列命令擷取。

```
aws ssm get-parameters \
      --names /aws/service/aws-for-fluent-bit/stable \
      --region us-east-1
```

所有版本的 AWS for Fluent Bit 映像都可以使用下列命令來列出，以查詢 Systems Manager 參數存放區參數。

```
aws ssm get-parameters-by-path \
      --path /aws/service/aws-for-fluent-bit \
      --region us-east-1
```

最新的 AWS Fluent Bit 影像穩定功能，可透過參考 Systems Manager 參數存放區名稱，在範本中 AWS CloudFormation 參考。以下是範例：

```
Parameters:
  FireLensImage:
    Description: Fluent Bit image for the FireLens Container
    Type: AWS::SSM::Parameter::Value<String>
```

```
    Default: /aws/service/aws-for-fluent-bit/stable
```

Windows

最新的 Fluent Bit 影像 URI AWS 穩定狀態，可以使用下列命令擷取。

```
aws ssm get-parameters \
      --names /aws/service/aws-for-fluent-bit/windowsservercore-stable \
      --region us-east-1
```

所有版本的 AWS for Fluent Bit 映像都可以使用下列命令來列出，以查詢 Systems Manager 參數存放區參數。

```
aws ssm get-parameters-by-path \
      --path /aws/service/aws-for-fluent-bit/windowsservercore \
      --region us-east-1
```

您可以在範本中 AWS CloudFormation 參考 Systems Manager 參數存放區名稱，以參考最新穩定的 AWS Fluent Bit 映像。以下是範例：

```
Parameters:
  FireLensImage:
    Description: Fluent Bit image for the FireLens Container
    Type: AWS::SSM::Parameter::Value<String>
    Default: /aws/service/aws-for-fluent-bit/windowsservercore-stable
```

Amazon ECS 任務定義範例：將日誌路由至 FireLens

若要搭配 FireLens 使用自訂日誌路由，您必須在工作定義中指定下列項目：

- 包含 FireLens 組態的日誌路由器容器。我們建議將容器標示為 `essential`。
- 包含指定日誌驅動程式的日誌配置的一或多個應用 `awsfirelens` 程式容器。
- 任務IAM 角色 Amazon Resource Name (ARN)，其中包含路由日誌所需的任務許可。

使用 建立新的任務定義時 AWS Management Console，有一個 FireLens 整合區段，可讓您輕鬆地新增日誌路由器容器。如需詳細資訊，請參閱使用主控台建立 Amazon ECS 任務定義。

Amazon ECS 轉換日誌組態並產生 Fluentd 或 Fluent Bit 輸出組態。輸出配置掛載在流利位和 `/fluentd/etc/fluent.conf` 流利位 `/fluent-bit/etc/fluent-bit.conf` 的日誌路由容器中。

> **Important**
>
> FireLens 會監聽連接埠 24224。因此為了確保 FireLens 日誌路由器無法在任務之外連線，您不能在任務使用的安全群組中允許連接埠 24224 上的傳入流量。對於使用 awsvpc 網路模式的任務，這是與任務相關聯的安全群組。對於使用 host 網路模式的任務，這是與託管任務的 Amazon EC2 執行個體相關聯的安全群組。對於使用 bridge 網路模式的任務，請不要建立使用連接埠 24224 的任何連接埠映射。

根據預設，Amazon ECS 會在日誌項目中新增其他欄位，以協助識別日誌的來源。

- ecs_cluster - 任務所屬叢集的名稱。
- ecs_task_arn – 容器所屬任務的完整 Amazon Resource Name (ARN)。
- ecs_task_definition - 任務正在使用的任務定義名稱和修訂版本。
- ec2_instance_id：託管容器的 Amazon EC2 執行個體 ID。此欄位僅用於使用 EC2 啟動類型的任務。

false 如果您不希望中繼資料，您可以將 enable-ecs-log-metadata 設定為 。

以下任務定義範例定義了一個日誌路由器容器，其使用 Fluent Bit，將其日誌路由至 CloudWatch Logs。它還定義了使用日誌組態將日誌路由到 Amazon Data Firehose 的應用程式容器，並將用於緩衝事件的記憶體設定為 2 MiB。

> **Note**
>
> 如需更多任務定義範例，請參閱 GitHub 上的 Amazon ECS FireLens 範例。

```
{
  "family": "firelens-example-firehose",
  "taskRoleArn": "arn:aws:iam::123456789012:role/ecs_task_iam_role",
  "containerDefinitions": [
    {
            "name": "log_router",
            "image": "public.ecr.aws/aws-observability/aws-for-fluent-bit:stable",
            "cpu": 0,
            "memoryReservation": 51,
            "portMappings": [],
```

```
            "essential": true,
            "environment": [],
            "mountPoints": [],
            "volumesFrom": [],
            "user": "0",
            "logConfiguration": {
                "logDriver": "awslogs",
                "options": {
                    "awslogs-group": "/ecs/ecs-aws-firelens-sidecar-container",
                    "mode": "non-blocking",
                    "awslogs-create-group": "true",
                    "max-buffer-size": "25m",
                    "awslogs-region": "us-east-1",
                    "awslogs-stream-prefix": "firelens"
                },
                "secretOptions": []
            },
            "systemControls": [],
            "firelensConfiguration": {
                "type": "fluentbit"
            }
        },
    {
      "essential": true,
      "image": "httpd",
      "name": "app",
      "logConfiguration": {
        "logDriver": "awsfirelens",
        "options": {
          "Name": "firehose",
          "region": "us-west-2",
          "delivery_stream": "my-stream",
          "log-driver-buffer-limit": "2097152"
        }
      },
      "memoryReservation": 100
    }
  ]
}
```

在 `logConfiguration` 物件中指定為選項的索引鍵值組，用來產生 Fluentd 或 Fluent Bit 輸出組態。以下是來自 Fluent Bit 輸出定義的程式碼範例。

```
[OUTPUT]
    Name   firehose
    Match  app-firelens*
    region us-west-2
    delivery_stream my-stream
```

> **Note**
>
> FireLens 管理 `match` 組態。您未在任務定義中指定`match`組態。

使用自訂組態檔案

您可以指定自訂組態檔案。組態檔格式是您所使用日誌路由器的原生格式。如需詳細資訊，請參閱 Fluentd Config 檔案語法和 YAML 組態。

在您的自訂組態檔案中，對於使用 `bridge` 或 `awsvpc` 網路模式的任務，不要透過 TCP 設定 Fluentd 或 Fluent Bit 向前輸入，因為 FireLens 將其新增到輸入組態中。

您的 FireLens 組態必須包含下列選項，才能指定自訂組態檔案：

`config-file-type`

自訂組態檔案的來源位置。可用選項為 `s3` 或 `file`。

> **Note**
>
> 上託管的任務 AWS Fargate 僅支援`file`組態檔案類型。

`config-file-value`

自訂組態檔案的來源。如果使用 `s3` 組態檔案類型，則組態檔案值是 Amazon S3 儲存貯體和檔案的完整 ARN。如果使用 `file` 組態檔案類型，組態檔案值就是容器映像中或掛載在容器中的磁碟區上的組態檔案的完整路徑。

> **Important**
>
> 使用自訂組態檔案時，您必須指定與 FireLens 使用的路徑不同的路徑。Amazon ECS 會為 Fluent Bit 保留 `/fluent-bit/etc/fluent-bit.conf` filepath，並為 Fluentd 保留 `/fluentd/etc/fluent.conf`。

下列範例顯示指定自訂組態時所需的語法。

> **Important**
>
> 若要指定託管於 Amazon S3 的自訂組態檔案，請確定您已建立具有適當許可的任務執行 IAM 角色。

以下顯示指定自訂組態時所需的語法。

```
{
  "containerDefinitions": [
    {
      "essential": true,
      "image": "906394416424.dkr.ecr.us-west-2.amazonaws.com/aws-for-fluent-
bit:stable",
      "name": "log_router",
      "firelensConfiguration": {
        "type": "fluentbit",
        "options": {
          "config-file-type": "s3 | file",
          "config-file-value": "arn:aws:s3:::amzn-s3-demo-bucket/fluent.conf
 | filepath"
        }
      }
    }
  ]
}
```

> **Note**
>
> 上託管的任務 AWS Fargate 僅支援`file`組態檔案類型。

在 Amazon ECS 中使用非AWS 容器映像

使用私有登錄檔將登入資料存放在 中 AWS Secrets Manager，然後在任務定義中參考登入資料。這提供了一種方法，可參考存在於 外部私有登錄檔中的容器映像 AWS ，這些登錄檔需要在您的任務定義中驗證。託管於 Fargate、Amazon EC2 執行個體以及使用 Amazon ECS Anywhere 的外部執行個體上的任務支援此功能。

> **Important**
>
> 如果您的任務定義參考存放在 Amazon ECR 中的映像，則此主題不適用。如需詳細資訊，請參閱 《Amazon Elastic Container Registry 使用者指南》中的搭配使用 Amazon ECR 映像與 Amazon ECS。

對於託管於 Amazon EC2 執行個體上的任務，此功能要求具備版本 `1.19.0` 或更新的容器代理程式。不過，我們建議您使用最新版的容器代理程式。如需如何檢查代理程式版本及更新至最新版本的資訊，請參閱「更新 Amazon ECS 容器代理程式」。

對於託管於 Fargate 上的任務，此功能需要平台版本 `1.2.0` 或更新版本。如需相關資訊，請參閱 適用於 Amazon ECS 的 Fargate 平台版本。

在您的容器定義內，使用您所建立的秘密的詳細資訊來指定 `repositoryCredentials` 物件。參考的秘密可以來自與使用任務不同的 AWS 區域 帳戶。

> **Note**
>
> 使用 Amazon ECS API AWS CLI或 AWS SDK 時，如果秘密與您啟動 AWS 區域 的任務位於相同的 中，您可以使用秘密的完整 ARN 或名稱。如果此秘密已存在於不同帳戶中，則必須指定秘密的完整 ARN。使用 時 AWS Management Console，一律必須指定秘密的完整 ARN。

以下是任務定義的程式碼片段，其會顯示所需的參數：

取代下列參數：

- 具有私有儲存庫主機名稱的 private*-repo*
- 具有映像名稱的 *private-image*
- *arn#aws#secretsmanager#region#aws_account_id#secret#secret_name* 搭配秘密 Amazon Resource Name (ARN)

```
"containerDefinitions": [
    {
        "image": "private-repo/private-image",
        "repositoryCredentials": {
            "credentialsParameter":
 "arn:aws:secretsmanager:region:aws_account_id:secret:secret_name"
        }
    }
]
```

> **Note**
>
> 啟用私有登錄檔身分驗證的另一個方法，是使用 Amazon ECS 容器代理程式環境變數來向私有登錄檔進行身分驗證。只有託管於 Amazon EC2 執行個體上的任務才支援此方法。如需詳細資訊，請參閱為私有 Docker 映像設定 Amazon ECS 容器執行個體 。

使用私有登錄檔

1. 任務定義必須具有任務執行角色。這可讓容器代理程式提取容器映像。如需詳細資訊，請參閱Amazon ECS 任務執行 IAM 角色。

   若要將存取提供給您建立的秘密，請將以下許可字作為內嵌政策，新增到任務執行角色。如需詳細資訊，請參閱新增和移除 IAM 政策。

   - `secretsmanager:GetSecretValue`
   - `kms:Decrypt` - 只有在您的金鑰使用自訂 KMS 金鑰而非預設金鑰時，才需要此項目。您的自訂金鑰的 Amazon Resource Name (ARN) 必須新增為資源。

   下列為新增許可的內嵌政策範例。

   ```
   {
       "Version": "2012-10-17",
       "Statement": [
           {
               "Effect": "Allow",
               "Action": [
                   "kms:Decrypt",
                   "secretsmanager:GetSecretValue"
               ],
   ```

```
            "Resource": [

 "arn:aws:secretsmanager:<region>:<aws_account_id>:secret:secret_name",
                "arn:aws:kms:<region>:<aws_account_id>:key/key_id"
            ]
        }
    ]
}
```

2. 使用 為您的私有登錄檔登入資料 AWS Secrets Manager 建立秘密。如需有關如何建立秘密的資訊，請參閱AWS Secrets Manager 《 使用者指南》中的建立 AWS Secrets Manager 秘密。

   使用下列格式輸入您的私有登錄檔登入資料：

```
{
  "username" : "privateRegistryUsername",
  "password" : "privateRegistryPassword"
}
```

3. 註冊任務定義。如需詳細資訊，請參閱the section called “使用主控台建立任務定義”。

使用容器重新啟動政策重新啟動 Amazon ECS 任務中的個別容器

您可以為任務定義中定義的每個必要和非必要容器啟用重新啟動政策，以更快地克服暫時性故障並維持任務可用性。當您為容器啟用重新啟動政策時，如果容器結束，Amazon ECS 可以重新啟動容器，而不需要取代任務。

根據預設，容器不會啟用重新啟動政策。當您為容器啟用重新啟動政策時，您可以指定不會重新啟動容器的結束代碼。這些可以是指示成功的結束代碼，例如不需要重新啟動0的結束代碼 。您也可以指定容器成功執行的時間，然後才能嘗試重新啟動。如需這些參數的相關資訊，請參閱 重新啟動政策。如需指定這些值的任務定義範例，請參閱 在 Amazon ECS 任務定義中指定容器重新啟動政策。

您可以使用 Amazon ECS 任務中繼資料端點或 CloudWatch Container Insights 來監控容器重新啟動的次數。如需任務中繼資料端點的詳細資訊，請參閱 Amazon ECS 任務中繼資料端點第 4 版和 Fargate 上任務的 Amazon ECS 任務中繼資料端點第 4 版。如需 Amazon ECS Container Insights 指標的詳細資訊，請參閱《Amazon Amazon CloudWatch ECS Container Insights 指標。

Fargate、Amazon EC2 執行個體和使用 Amazon ECS Anywhere 的外部執行個體上託管的任務支援容器重新啟動政策。

考量事項

為您的容器啟用重新啟動政策之前，請考慮下列事項：

- 對於託管於 Amazon EC2 執行個體上的任務，此功能要求具備版本 1.86.0 或更新的容器代理程式。不過，我們建議您使用最新版的容器代理程式。如需如何檢查代理程式版本及更新至最新版本的資訊，請參閱「更新 Amazon ECS 容器代理程式」。
- 對於託管於 Fargate 上的任務，此功能需要平台版本 1.4.0 或更新版本。如需相關資訊，請參閱 適用於 Amazon ECS 的 Fargate 平台版本。
- 如果您使用 EC2 啟動類型搭配bridge網路模式，應用程式容器中FLUENT_HOST的環境變數可能會在重新啟動 FireLens 日誌路由器容器（容器定義中具有firelensConfiguration物件的容器）之後變得不準確。這是因為 FLUENT_HOST 是動態 IP 地址，重新啟動後可能會變更。地址變更後，直接從應用程式容器記錄到 FLUENT_HOST IP 地址可能會開始失敗。如需有關 FLUENT_HOST 的詳細資訊，請參閱 設定高輸送量的 Amazon ECS 日誌。
- Amazon ECS 代理程式會處理容器重新啟動政策。如果 Amazon ECS 代理程式因某些非預期的原因失敗或不再執行，則容器將不會重新啟動。
- 政策中定義的重新啟動嘗試期間決定容器在 Amazon ECS 重新啟動容器之前必須執行的期間（以秒為單位）。

在 Amazon ECS 任務定義中指定容器重新啟動政策

若要在任務定義中指定容器的重新啟動政策，請在容器定義中指定 restartPolicy 物件。如需restartPolicy物件的詳細資訊，請參閱 重新啟動政策。

以下是使用 Fargate 啟動類型上設定 Web 伺服器之 Linux 容器的任務定義。容器定義包含 restartPolicy 物件，並將 enabled 設定為 true 以啟用容器的重新啟動政策。容器必須執行 180 秒，才能重新啟動，如果結束時，則不會重新啟動0，這表示成功。

```
{
  "containerDefinitions": [
    {
      "command": [
        "/bin/sh -c \"echo '<html> <head> <title>Amazon ECS Sample App</title>
 <style>body {margin-top: 40px; background-color: #333;} </style> </head><body>
 <div style=color:white;text-align:center> <h1>Amazon ECS Sample App</h1>
 <h2>Congratulations!</h2> <p>Your application is now running on a container in Amazon
 ECS.</p> </div></body></html>' >  /usr/local/apache2/htdocs/index.html && httpd-
foreground\""
```

```
      ],
      "entryPoint": ["sh", "-c"],
      "essential": true,
      "image": "httpd:2.4",
      "logConfiguration": {
        "logDriver": "awslogs",
        "options": {
          "awslogs-group": "/ecs/fargate-task-definition",
          "awslogs-region": "us-east-1",
          "awslogs-stream-prefix": "ecs"
        }
      },
      "name": "sample-fargate-app",
      "portMappings": [
        {
          "containerPort": 80,
          "hostPort": 80,
          "protocol": "tcp"
        }
      ],
      "restartPolicy": {
        "enabled": true,
        "ignoredExitCodes": [0],
        "restartAttemptPeriod": 180
      }
    }
  ],
  "cpu": "256",
  "executionRoleArn": "arn:aws:iam::012345678910:role/ecsTaskExecutionRole",
  "family": "fargate-task-definition",
  "memory": "512",
  "networkMode": "awsvpc",
  "runtimePlatform": {
    "operatingSystemFamily": "LINUX"
  },
  "requiresCompatibilities": ["FARGATE"]
}
```

使用容器定義中的`restartPolicy`物件註冊任務定義後，您可以執行任務或使用該任務定義建立服務。如需詳細資訊，請參閱 以 Amazon ECS 任務執行應用程式 和 使用 主控台建立 Amazon ECS 服務。

將敏感資料傳遞至 Amazon ECS 容器

您可以安全地將敏感資料 (例如資料庫憑證) 傳遞至容器。

秘密 (例如 API 金鑰和資料庫憑證) 經常被應用程式用來存取其他系統。其通常由使用者名稱和密碼、憑證或 API 金鑰組成。對這些秘密的存取應限制在使用 IAM 的特定 IAM 主體，並在執行期插入容器。

秘密可以從 AWS Secrets Manager 和 Amazon EC2 Systems Manager 參數存放區無縫注入容器。這些秘密可以在您的任務中引用為以下任何內容。

1. 被引用為使用 `secrets` 容器定義參數的環境變數。
2. 如果您的日誌平台需要驗證，則被引用為 `secretOptions`。如需詳細資訊，請參閱紀錄組態選項。
3. 如果從中提取容器的登錄檔需要驗證，則使用 `repositoryCredentials` 容器定義參數的映像會將其參照為提取的秘密。從 Amazon ECR Public Gallery 提取映像時使用此方法。如需詳細資訊，請參閱任務的私有登錄檔身分驗證。

建議您在設定秘密管理時執行下列動作。

使用 AWS Secrets Manager 或 AWS Systems Manager 參數存放區來存放秘密資料

您應該將 API 金鑰、資料庫登入資料和其他秘密資料安全地存放在 Secrets Manager 中，或做為 Systems Manager 參數存放區的加密參數。這些服務類似，因為它們都是 AWS KMS 用於加密敏感資料的受管金鑰值存放區。不過，Secrets Manager 也包含自動輪換秘密、產生隨機秘密，以及跨帳戶共用秘密的功能。如果這些重要功能，請使用 Secrets Manager，否則請使用加密的參數。

> **Important**
>
> 如果您的秘密發生變更，您必須強制執行新部署或啟動新任務，才能擷取最新的秘密值。如需詳細資訊，請參閱下列主題：
>
> - 任務 - 停止任務，然後啟動任務。如需詳細資訊，請參閱 停止 Amazon ECS 任務 和 以 Amazon ECS 任務執行應用程式。
> - 服務 - 更新服務並使用強制新部署選項。如需詳細資訊，請參閱使用主控台更新 Amazon ECS 服務。

從加密的 Amazon S3 儲存貯體擷取資料

您應該將秘密存放在加密的 Amazon S3 儲存貯體中，並使用任務角色來限制這些秘密的存取。這可防止環境變數的值在日誌中意外洩漏，並在執行 時被洩漏`docker inspect`。執行此動作時，必須寫入應用程式，才能讀取 Amazon S3 儲存貯體中的秘密。如需說明，請參閱對 Amazon S3 儲存貯體設定預設伺服器端加密行為。

使用附屬容器將秘密掛載至磁碟區

由於環境變數的資料洩漏風險較高，因此您應該執行附屬容器，從 讀取秘密 AWS Secrets Manager 並將其寫入共用磁碟區。透過使用 Amazon ECS 容器排序，此容器可以在應用程式容器之前執行和結束。執行此操作時，應用程式容器隨後會掛載寫入秘密的磁碟區。就像 Amazon S3 儲存貯體方法一樣，必須寫入您的應用程式才能從共用磁碟區讀取秘密。由於磁碟區的範圍限定在任務，因此會在任務停止後自動刪除磁碟區。有關附屬容器的範例，請參閱 aws-secret-sidecar-injector 專案。

在 Amazon EC2 上，寫入秘密的磁碟區可以使用 AWS KMS 客戶受管金鑰加密。在 上 AWS Fargate，磁碟區儲存會使用服務受管金鑰自動加密。

將個別環境變數傳遞至 Amazon ECS 容器

> **Important**
>
> 建議您將敏感資料存放在 AWS Secrets Manager 秘密或 AWS Systems Manager 參數存放區參數中。如需詳細資訊，請參閱將敏感資料傳遞至 Amazon ECS 容器。
> 任務定義中指定的環境變數可供所有使用者和角色讀取，系統允許這些使用者和角色對任務定義執行 `DescribeTaskDefinition` 動作。

您可以將環境變數透過下列方式傳遞至容器：

- 個別方式：使用 `environment` 容器定義參數。這會映射到 docker container run 的 `--env` 選項。
- 大批方式：使用 `environmentFiles` 容器定義參數列出內含環境變數的一個或多個檔案。檔案必須託管在 Amazon S3 中。這會映射到 docker run 的 `--env-file` 選項。

以下任務定義片段示範如何指定個別環境變數。

```
{
    "family": "",
    "containerDefinitions": [
```

```
        {
            "name": "",
            "image": "",
            ...
            "environment": [
                {
                    "name": "variable",
                    "value": "value"
                }
            ],
            ...
        }
    ],
    ...
}
```

將環境變數傳遞至 Amazon ECS 容器

> **Important**
>
> 建議您將敏感資料存放在 AWS Secrets Manager 秘密或 AWS Systems Manager 參數存放區參數中。如需詳細資訊，請參閱將敏感資料傳遞至 Amazon ECS 容器。
> 環境變數檔案是 Simple Storage Service (Amazon S3) 中的物件，所有 Amazon S3 安全考量事項均適用。
> 您無法在 Windows 容器和 Fargate 上的 Windows 容器上使用 `environmentFiles` 參數。

您可以建立環境變數檔案，並將其存放在 Amazon S3 中，以將環境變數傳遞至容器。

透過在檔案中指定環境變數，您可以批次導入環境變數。在容器定義中指定 `environmentFiles` 物件，其中具有內含環境變數檔案的 Amazon S3 儲存貯體清單。

Amazon ECS 不會強制執行環境變數的大小限制，但大型環境變數檔案可能會填滿磁碟空間。使用環境變數檔案的每個任務都會導致檔案複本下載到磁碟。Amazon ECS 會在任務清除過程中移除 檔案。

如需有關支援之環境變數的資訊，請參閱進階容器定義參數 - 環境。

在容器定義中指定環境變數檔案時，請考量下列事項。

- 對於 Amazon EC2 上的 Amazon ECS 任務，您的容器執行個體需要版本 `1.39.0` 或更新的容器代理程式，才能使用此功能。如需如何檢查代理程式版本及更新至最新版本的資訊，請參閱「更新 Amazon ECS 容器代理程式」。

- 對於 AWS Fargate 上的 Amazon ECS 任務，您的任務必須使用平台版本 `1.4.0` 或更新版本 (Linux) 才能使用此功能。如需詳細資訊，請參閱適用於 Amazon ECS 的 Fargate 平台版本。

  驗證作業系統平台是否支援該變數。如需詳細資訊，請參閱the section called “容器定義”及the section called “其他任務定義參數”。
- 檔案必須使用 `.env` 副檔名和 UTF-8 編碼。
- 任務執行角色是搭配 Amazon S3 的額外許可使用此功能的必要項目。這可讓容器代理程式從 Amazon S3 中提取環境變數檔案。如需詳細資訊，請參閱Amazon ECS 任務執行 IAM 角色。
- 每個任務定義限制 10 個檔案。
- 環境檔案中的每一行都必須包含 `VARIABLE=VALUE` 格式的環境變數。空格或引號會包含為 Amazon ECS 檔案值的一部分。以 # 開頭的行會被視為註解，而忽略。如需環境變數檔案語法的詳細資訊，請參閱 Docker 文件中的設定環境變數 (-e、--env、--env-file)。

  下列為適當的語法。

  ```
  #This is a comment and will be ignored
  VARIABLE=VALUE
  ENVIRONMENT=PRODUCTION
  ```
- 如果有在容器定義中使用 `environment` 參數指定的環境變數，它們的優先順序高於環境檔案中包含的變數。
- 如果指定內含相同變數的多個環境檔案，則處理順序為先進入者為優先。這表示會使用變數的第一個值，並忽略重複變數的後續值。建議您使用唯一的變數名稱。
- 如果將環境檔案指定為容器覆寫，則會使用該檔案。此外，在容器定義中指定的任何其他環境檔案都將被忽略。
- 下列規則適用於 Fargate 啟動類型：
  - 檔案的處理方式與原生 Docker env-file 類似。
  - 參考 Amazon S3 中空白環境變數的容器定義不會出現在容器中。
  - 不支援 Shell 逸出處理。
  - 容器進入點會解譯 `VARIABLE` 值。

範例

以下任務定義片段示範如何指定環境變數檔案。

```
{
```

```
    "family": "",
    "containerDefinitions": [
        {
            "name": "",
            "image": "",
            ...
            "environmentFiles": [
                {
                    "value": "arn:aws:s3:::amzn-s3-demo-
bucket/envfile_object_name.env",
                    "type": "s3"
                }
            ],
            ...
        }
    ],
    ...
}
```

在 Amazon ECS 中以程式設計方式傳遞 Secrets Manager 秘密

您可以使用 Secrets Manager 來存放敏感資料，而不是在應用程式中以純文字硬式編碼敏感資訊。

我們建議使用這種方法擷取敏感資料，因為使用這種方法後，隨後如果 Secrets Manager 秘密有更新，應用程式會自動擷取最新版本的秘密。

在 Secrets Manager 中建立秘密。建立 Secret Manager 秘密後，更新應用程式程式碼即可擷取秘密。

保護 Secrets Manager 中的敏感資料之前，請先檢閱下列考量事項。

- 僅支援存放文字資料的秘密，這些秘密是使用 CreateSecret API 的 `SecretString` 參數建立的。不支援存放二進位資料的秘密，這是使用 CreateSecret API `SecretBinary` 參數建立的秘密。
- 使用介面 VPC 端點來增強安全控制。您必須建立 Secrets Manager 的介面 VPC 端點。如需有關 VPC 端點的資訊，請參閱《AWS Secrets Manager 使用者指南》中的建立VPC 端點。
- 任務所使用的 VPC 必須使用 DNS 解析。
- 您的任務定義必須使用具有 Secrets Manager 額外許可的任務角色。如需詳細資訊，請參閱Amazon ECS 任務 IAM 角色。

建立 Secrets Manager 秘密

您可以使用 Secrets Manager 主控台為您的敏感資料建立秘密。如需有關如何建立秘密的詳細資訊，請參閱《AWS Secrets Manager 使用者指南》中的建立 AWS Secrets Manager 秘密。

更新應用程式以程式設計方式擷取 Secrets Manager 秘密

您可以直接從應用程式呼叫 Secrets Manager API 來擷取秘密。如需詳細資訊，請參閱AWS Secrets Manager 《 使用者指南》中的從 擷取秘密 AWS Secrets Manager。

若要擷取存放在 中的敏感資料 AWS Secrets Manager，請參閱 SDK Code Example Code Library 中的使用 AWS Secrets ManagerAWS SDKs程式碼範例。 AWS

在 Amazon ECS 中以程式設計方式傳遞 Systems Manager 參數存放區秘密

Systems Manager Parameter Store 提供安全的秘密儲存和管理。您可以在應用程式中將密碼、資料庫字串、EC2 執行個體 IDs 和 AMI IDs 和授權碼等資料儲存為參數值，而不是硬式編碼此資訊。您存放的值可以是純文字或加密資料。

我們建議您使用此方法來擷取敏感資料，因為如果 Systems Manager 參數存放區參數後續更新，應用程式會自動擷取最新版本。

在保護 Systems Manager Parameter Store 中的敏感資料之前，請先檢閱以下考量事項。

- 僅支援儲存文字資料的秘密。不支援儲存二進位資料的秘密。
- 使用介面 VPC 端點來增強安全控制。
- 任務所使用的 VPC 必須使用 DNS 解析。
- 對於使用 EC2 啟動類型的任務，您必須使用 Amazon ECS 代理程式組態變數`ECS_ENABLE_AWSLOGS_EXECUTIONROLE_OVERRIDE=true`來使用此功能。您可以在建立容器執行個體期間將其新增至 `/etc/ecs/ecs.config` 檔案，也可以將其新增至現有的執行個體，然後重新啟動 ECS 代理程式。如需詳細資訊，請參閱Amazon ECS 容器代理程式組態。
- 您的任務定義必須使用具有 Systems Manager 參數存放區額外許可的任務角色。如需詳細資訊，請參閱Amazon ECS 任務 IAM 角色。

建立 參數

您可以使用 Systems Manager 主控台為您的敏感資料建立 Systems Manager Parameter Store 參數。如需詳細資訊，請參閱《AWS Systems Manager 使用者指南》中的建立 Systems Manager 參數 (主控台) 或建立 Systems Manager 參數 (AWS CLI)。

更新應用程式，以程式設計方式擷取 Systems Manager Parameter Store 秘密

若要擷取儲存在 Systems Manager 參數存放區參數中的敏感資料，請參閱 SDK Code Example Code Library 中的 Systems Manager using AWS SDKs 的程式碼範例。 AWS

透過 Amazon ECS 環境變數傳遞 Secrets Manager 秘密

當您將秘密作為環境變數插入時，可以指定秘密的完整內容、秘密內的特定 JSON 金鑰或要插入的特定秘密版本。這可協助您控制向容器公開的敏感資料。如需秘密版本控制的詳細資訊，請參閱AWS Secrets Manager 《 使用者指南》中的 Secrets Manager 秘密中的內容？。

使用環境變數將 Secrets Manager 秘密注入容器時，應考慮下列事項。

- 敏感資料會在初次啟動容器時，嵌入您的容器。如果後續更新或輪換秘密，則容器不會自動收到更新的值。您必須啟動新的任務，或如果任務是服務的一部分，您可以更新服務，並使用 Force new deployment (強制新的部署) 選項強制服務來啟動新的任務。
- 對於 上的 Amazon ECS 任務 AWS Fargate，請考慮下列事項：
  - 若要將秘密的完整內容作為環境變數或在日誌組態中插入，您必須使用平台版本 `1.3.0` 或更新版本。如需相關資訊，請參閱 適用於 Amazon ECS 的 Fargate 平台版本。
  - 若要將特定 JSON 金鑰或秘密版本作為環境變數或在日誌組態中插入，您必須使用平台版本 `1.4.0` 或更新版本 (Linux)，或者 `1.0.0` (Windows)。如需相關資訊，請參閱 適用於 Amazon ECS 的 Fargate 平台版本。
- 對於 EC2 上的 Amazon ECS 任務，應考慮下列事項：
  - 若要使用秘密的特定 JSON 索引鍵或版本插入秘密，您的容器執行個體必須有 `1.37.0` 版或更新版本的容器代理程式。不過，我們建議您使用最新版的容器代理程式。如需檢查代理程式版本及更新至最新版本的資訊，請參閱「更新 Amazon ECS 容器代理程式」。

    若要插入秘密的完整內容做為環境變數，或在日誌組態中插入秘密，則容器執行個體必須有 `1.22.0` 版或更新版本的容器代理程式。
- 使用界面 VPC 端點來增強安全控制，並透過私有子網路連線至 Secrets Manager。您必須建立 Secrets Manager 的介面 VPC 端點。如需有關 VPC 端點的資訊，請參閱《AWS Secrets Manager 使用者指南》中的建立VPC 端點。如需使用 Secrets Manager 和 Amazon VPC 的詳細資訊，請參閱如何在 Amazon VPC 中連線至 Secrets Manager 服務。
- 對於設定為使用 `awslogs` 日誌記錄驅動程式的 Windows 任務，您也必須在容器執行個體上設定 `ECS_ENABLE_AWSLOGS_EXECUTIONROLE_OVERRIDE` 環境變數。使用下列語法：

```
<powershell>
```

```
[Environment]::SetEnvironmentVariable("ECS_ENABLE_AWSLOGS_EXECUTIONROLE_OVERRIDE",
 $TRUE, "Machine")
Initialize-ECSAgent -Cluster <cluster name> -EnableTaskIAMRole -LoggingDrivers
 '["json-file","awslogs"]'
</powershell>
```

- 您的任務定義必須使用任務執行角色搭配 Secrets Manager 的額外許可。如需詳細資訊，請參閱Amazon ECS 任務執行 IAM 角色。

建立 AWS Secrets Manager 秘密

您可以使用 Secrets Manager 主控台為您的敏感資料建立秘密。如需詳細資訊，請參閱AWS Secrets Manager 《 使用者指南》中的建立 AWS Secrets Manager 秘密。

將環境變數新增至容器定義

您可以在容器定義中指定下列項目：

- 包含要在容器中設定之環境變數名稱的 `secrets` 物件。
- Secrets Manager 秘密的 Amazon Resource Name (ARN)
- 包含要提供給容器之敏感資料的其他參數

下列範例示範了必須為 Secrets Manager 秘密指定的完整語法。

```
arn:aws:secretsmanager:region:aws_account_id:secret:secret-name:json-key:version-
stage:version-id
```

以下部分說明其他參數。這些參數雖然是選用，但如果您不使用，則必須包含冒號：來使用預設值。以下提供範例深入說明。

`json-key`

使用您要設為環境變數值的值，來指定金鑰/值對中的金鑰名稱。僅支援 JSON 格式的值。如果您沒有指定 JSON 金鑰，則會使用秘密的完整內容。

`version-stage`

指定您要使用之秘密版本的預備標籤。如果指定了版本預備標籤，就無法指定版本 ID。如果未指定版本階段，則預設會擷取具有 AWSCURRENT 階段標籤的秘密。

預備標籤會用來在不同版本的秘密更新或輪換時加以追蹤。每個版本的秘密都有一或多個預備標籤和 ID。如需詳細資訊，請參閱 《AWS Secrets Manager 使用者指南》中的 AWS Secrets Manager 的重要術語和概念。

`version-id`

針對您要使用的秘密版本，指定其唯一識別符。如果指定了版本 ID，就無法指定版本預備標籤。如果未指定版本 ID，則預設會擷取具有 `AWSCURRENT` 階段標籤的秘密。

版本 ID 會用來在不同版本的秘密更新或輪換時加以追蹤。每個版本的秘密都有 ID。如需詳細資訊，請參閱 《AWS Secrets Manager 使用者指南》中的 AWS Secrets Manager的重要術語和概念。

容器定義範例

下列範例示範您可以在容器定義中參考 Secrets Manager 秘密的方法。

Example 參考完整秘密

以下是任務定義的程式碼片段，顯示參考 Secrets Manager 秘密全文時的格式。

```
{
  "containerDefinitions": [{
    "secrets": [{
      "name": "environment_variable_name",
      "valueFrom": "arn:aws:secretsmanager:region:aws_account_id:secret:secret_name-
AbCdEf"
    }]
  }]
}
```

若要從容器內存取此秘密的值，您需要呼叫 `$environment_variable_name`。

Example 參考秘密中的特定金鑰

以下示範 get-secret-value 命令的範例輸出，會顯示秘密的內容，以及與其相關的版本預備標籤和版本 ID。

```
{
    "ARN": "arn:aws:secretsmanager:region:aws_account_id:secret:appauthexample-AbCdEf",
    "Name": "appauthexample",
    "VersionId": "871d9eca-18aa-46a9-8785-981ddEXAMPLE",
```

```
    "SecretString": "{\"username1\":\"password1\",\"username2\":\"password2\",
\"username3\":\"password3\"}",
    "VersionStages": [
        "AWSCURRENT"
    ],
    "CreatedDate": 1581968848.921
}
```

在 ARN 結尾指定金鑰名稱，來在容器定義中參考上一個輸出的特定金鑰。

```
{
  "containerDefinitions": [{
    "secrets": [{
      "name": "environment_variable_name",
      "valueFrom": "arn:aws:secretsmanager:region:aws_account_id:secret:appauthexample-
AbCdEf:username1::"
    }]
  }]
}
```

Example 參考特定秘密版本

以下示範 describe-secret 命令的範例輸出，會顯示秘密的未加密內容，以及所有版本秘密的中繼資料。

```
{
    "ARN": "arn:aws:secretsmanager:region:aws_account_id:secret:appauthexample-AbCdEf",
    "Name": "appauthexample",
    "Description": "Example of a secret containing application authorization data.",
    "RotationEnabled": false,
    "LastChangedDate": 1581968848.926,
    "LastAccessedDate": 1581897600.0,
    "Tags": [],
    "VersionIdsToStages": {
        "871d9eca-18aa-46a9-8785-981ddEXAMPLE": [
            "AWSCURRENT"
        ],
        "9d4cb84b-ad69-40c0-a0ab-cead3EXAMPLE": [
            "AWSPREVIOUS"
        ]
    }
}
```

在 ARN 結尾指定金鑰名稱，來在容器定義中參考上一個輸出的特定版本預備標籤。

```
{
  "containerDefinitions": [{
    "secrets": [{
      "name": "environment_variable_name",
      "valueFrom": "arn:aws:secretsmanager:region:aws_account_id:secret:appauthexample-AbCdEf::AWSPREVIOUS:"
    }]
  }]
}
```

在 ARN 結尾指定金鑰名稱，來在容器定義中參考上一個輸出的特定版本 ID。

```
{
  "containerDefinitions": [{
    "secrets": [{
      "name": "environment_variable_name",
      "valueFrom": "arn:aws:secretsmanager:region:aws_account_id:secret:appauthexample-AbCdEf:::9d4cb84b-ad69-40c0-a0ab-cead3EXAMPLE"
    }]
  }]
}
```

Example 參考秘密的特定金鑰和版本預備標籤

以下說明如何參考秘密中的特定金鑰和特定版本預備標籤。

```
{
  "containerDefinitions": [{
    "secrets": [{
      "name": "environment_variable_name",
      "valueFrom": "arn:aws:secretsmanager:region:aws_account_id:secret:appauthexample-AbCdEf:username1:AWSPREVIOUS:"
    }]
  }]
}
```

若要指定特定的金鑰和版本 ID，請使用下列語法。

```
{
```

```
  "containerDefinitions": [{
    "secrets": [{
      "name": "environment_variable_name",
      "valueFrom": "arn:aws:secretsmanager:region:aws_account_id:secret:appauthexample-
AbCdEf:username1::9d4cb84b-ad69-40c0-a0ab-cead3EXAMPLE"
    }]
  }]
}
```

如需有關如何使用環境變數中指定的秘密建立任務定義的資訊，請參閱 使用主控台建立 Amazon ECS 任務定義。

透過 Amazon ECS 環境變數傳遞 Systems Manager 參數

Amazon ECS 可讓您將敏感資料存放在 AWS Systems Manager 參數存放區參數中，然後在容器定義中參考，以將敏感資料注入容器。

使用環境變數將 Systems Manager 秘密注入容器時，請考慮下列事項。

- 敏感資料會在初次啟動容器時，嵌入您的容器。如果後續更新或輪換秘密，則容器不會自動收到更新的值。您必須啟動新的任務，或如果任務是服務的一部分，您可以更新服務，並使用 Force new deployment (強制新的部署) 選項強制服務來啟動新的任務。
- 對於 上的 Amazon ECS 任務 AWS Fargate，應考慮下列事項：
  - 若要將秘密的完整內容作為環境變數或在日誌組態中插入，您必須使用平台版本 `1.3.0` 或更新版本。如需相關資訊，請參閱 適用於 Amazon ECS 的 Fargate 平台版本。
  - 若要將特定 JSON 金鑰或秘密版本作為環境變數或在日誌組態中插入，您必須使用平台版本 `1.4.0` 或更新版本 (Linux)，或者 `1.0.0` (Windows)。如需相關資訊，請參閱 適用於 Amazon ECS 的 Fargate 平台版本。
- 對於 EC2 上的 Amazon ECS 任務，應考慮下列事項：
  - 若要使用秘密的特定 JSON 索引鍵或版本插入秘密，您的容器執行個體必須有 `1.37.0` 版或更新版本的容器代理程式。不過，我們建議您使用最新版的容器代理程式。如需檢查代理程式版本及更新至最新版本的資訊，請參閱「更新 Amazon ECS 容器代理程式」。

    若要插入秘密的完整內容做為環境變數，或在日誌組態中插入秘密，則容器執行個體必須有 `1.22.0` 版或更新版本的容器代理程式。
- 使用介面 VPC 端點來增強安全控制。您必須為 Systems Manager 建立介面 VPC 端點。如需 VPC 端點的相關資訊，請參閱AWS Systems Manager 《 使用者指南》中的使用適用於 Systems Manager 的 VPC 端點來改善 EC2 執行個體的安全性。

- 您的任務定義必須使用任務執行角色搭配 Secrets Manager 的額外許可。如需詳細資訊，請參閱Amazon ECS 任務執行 IAM 角色。
- 對於設定為使用 `awslogs` 日誌記錄驅動程式的 Windows 任務，您也必須在容器執行個體上設定 `ECS_ENABLE_AWSLOGS_EXECUTIONROLE_OVERRIDE` 環境變數。使用下列語法：

```
<powershell>
[Environment]::SetEnvironmentVariable("ECS_ENABLE_AWSLOGS_EXECUTIONROLE_OVERRIDE",
 $TRUE, "Machine")
Initialize-ECSAgent -Cluster <cluster name> -EnableTaskIAMRole -LoggingDrivers
 '["json-file","awslogs"]'
</powershell>
```

建立 Systems Manager 參數

您可以使用 Systems Manager 主控台為您的敏感資料建立 Systems Manager Parameter Store 參數。如需詳細資訊，請參閱《AWS Systems Manager 使用者指南》中的建立 Systems Manager 參數 (主控台) 或建立 Systems Manager 參數 (AWS CLI)。

將環境變數新增至容器定義

在任務定義中的容器定義中，`secrets`使用容器中設定的環境變數名稱，以及包含要呈現給容器之敏感資料的 Systems Manager 參數存放區參數的完整 ARN，來指定 。如需詳細資訊，請參閱`secrets`。

以下是任務定義的程式碼片段，顯示參考 Systems Manager 參數存放區參數的格式。如果 Systems Manager 參數存放區參數與您要啟動的任務位於相同區域中，則您可以使用參數的完整 ARN 或名稱。如果參數存在於不同區域，則請指定完整 ARN。

```
{
  "containerDefinitions": [{
    "secrets": [{
      "name": "environment_variable_name",
      "valueFrom": "arn:aws:ssm:region:aws_account_id:parameter/parameter_name"
    }]
  }]
}
```

如需有關如何使用環境變數中指定的秘密建立任務定義的資訊，請參閱 使用主控台建立 Amazon ECS 任務定義。

更新應用程式，以程式設計方式擷取 Systems Manager Parameter Store 秘密

若要擷取儲存在 Systems Manager 參數存放區參數中的敏感資料，請參閱 SDK Code Example Code Library 中的 Systems Manager using AWS SDKs 的程式碼範例。 AWS

Amazon ECS 記錄組態的傳遞秘密

您可以使用 中的 `secretOptions` 參數`logConfiguration`來傳遞用於記錄的敏感資料。

您可以在 Secrets Manager 或 Systems Manager 中存放秘密。

使用 Secrets Manager

在您的容器定義內，指定 `logConfiguration` 時，您可指定 `secretOptions`，在該參數中，需提供要在容器中設定的日誌驅動程式選項名稱，以及含有要呈現給容器之敏感資料的 Secrets Manager 秘密之完整 ARN。

以下是任務定義的程式碼片段，顯示參考 Secrets Manager 秘密時的格式。

```
{
  "containerDefinitions": [{
    "logConfiguration": [{
      "logDriver": "splunk",
      "options": {
        "splunk-url": "https://your_splunk_instance:8088"
      },
      "secretOptions": [{
        "name": "splunk-token",
        "valueFrom": "arn:aws:secretsmanager:region:aws_account_id:secret:secret_name-
AbCdEf"
      }]
    }]
  }]
}
```

將環境變數新增至容器定義

在您的容器定義內，將 `secrets` 指定為要在容器中設定的環境變數名稱，以及 Systems Manager 參數存放區參數 (含有要呈現給容器的敏感資料) 的完整 ARN。如需詳細資訊，請參閱`secrets`。

以下是任務定義的程式碼片段，顯示參考 Systems Manager 參數存放區參數的格式。如果 Systems Manager 參數存放區參數與您要啟動的任務位於相同區域中，則您可以使用參數的完整 ARN 或名稱。如果參數存在於不同區域，則請指定完整 ARN。

```
{
  "containerDefinitions": [{
    "secrets": [{
      "name": "environment_variable_name",
      "valueFrom": "arn:aws:ssm:region:aws_account_id:parameter/parameter_name"
    }]
  }]
}
```

如需有關如何使用環境變數中指定的秘密建立任務定義的資訊，請參閱 使用主控台建立 Amazon ECS 任務定義。

使用 Systems Manager

您可以將敏感資料插入日誌組態。在您的容器定義內，指定 `logConfiguration` 時，您可用要在容器中設定的日誌驅動程式選項名稱指定 `secretOptions`，以及 Systems Manager 參數存放區參數 (含有要呈現給容器的敏感資料) 的完整 ARN。

> **Important**
>
> 如果 Systems Manager 參數存放區參數與您要啟動的任務位於相同區域中，則您可以使用參數的完整 ARN 或名稱。如果參數存在於不同區域，則請指定完整 ARN。

以下是任務定義的程式碼片段，顯示參考 Systems Manager 參數存放區參數的格式。

```
{
  "containerDefinitions": [{
    "logConfiguration": [{
      "logDriver": "fluentd",
      "options": {
        "tag": "fluentd demo"
      },
      "secretOptions": [{
        "name": "fluentd-address",
        "valueFrom": "arn:aws:ssm:region:aws_account_id:parameter:/parameter_name"
      }]
    }]
  }]
}
```

在 Amazon ECS 中使用 Secrets Manager 秘密指定敏感資料

Amazon ECS 可讓您將敏感資料存放在 AWS Secrets Manager 秘密中，然後在容器定義中參考，以將敏感資料注入容器。如需詳細資訊，請參閱將敏感資料傳遞至 Amazon ECS 容器。

了解如何建立 Secrets Manager 秘密、在 Amazon ECS 任務定義中參考秘密，然後透過查詢容器內顯示秘密內容的環境變數來驗證它是否有效。

必要條件

本教學課程假設已完成下列先決條件：

- 已完成「設定以使用 Amazon ECS。」中的步驟。
- 您的使用者擁有建立 Secrets Manager 和 Amazon ECS 資源所需的 IAM 許可。

步驟 1：建立 Secrets Manager 機密

您可以使用 Secrets Manager 主控台為您的敏感資料建立秘密。在本教學課程中，我們將建立基本秘密，以供存放容器中稍後參考的使用者名稱和密碼。如需詳細資訊，請參閱AWS Secrets Manager《使用者指南》中的建立 AWS Secrets Manager 秘密。

要儲存在此秘密中的鍵/值對是教學課程結尾處容器中的環境變量值。

儲存秘密 ARN，以在後續步驟的任務執行 IAM 政策和任務定義中參考。

步驟 2：將秘密許可新增至任務執行角色

若要讓 Amazon ECS 從 Secrets Manager 秘密擷取敏感資料，您必須擁有任務執行角色的秘密許可。如需詳細資訊，請參閱Secrets Manager 或 Systems Manager 許可。

步驟 3：建立任務定義

您可以使用 Amazon ECS 主控台來建立一個參考 Secrets Manager 秘密的任務定義。

建立一個指定秘密的任務定義

使用 IAM 主控台，以所需的許可更新您的任務執行角色。

1. 開啟主控台，網址為 https://console.aws.amazon.com/ecs/v2。
2. 在導覽窗格中，選擇 Task Definitions (任務定義)。
3. 選擇 Create new task definitio (建立新任務定義)、Create new task definition with JSON (使用 JSON 建立新的任務定義)。

4. 在 JSON 編輯器方塊中，輸入下列任務定義 JSON 文字，確保您指定您在步驟 1 中建立的 Secrets Manager 秘密的完整 ARN，以及您在步驟 2 中更新的任務執行角色。選擇 Save (儲存)。

5.
```
{
    "executionRoleArn": "arn:aws:iam::aws_account_id:role/ecsTaskExecutionRole",
    "containerDefinitions": [
        {
            "entryPoint": [
                "sh",
                "-c"
            ],
            "portMappings": [
                {
                    "hostPort": 80,
                    "protocol": "tcp",
                    "containerPort": 80
                }
            ],
            "command": [
                "/bin/sh -c \"echo '<html> <head> <title>Amazon ECS Sample
 App</title> <style>body {margin-top: 40px; background-color: #333;} </style> </
head><body> <div style=color:white;text-align:center> <h1>Amazon ECS Sample App</
h1> <h2>Congratulations!</h2> <p>Your application is now running on a container in
 Amazon ECS.</p> </div></body></html>' >  /usr/local/apache2/htdocs/index.html &&
 httpd-foreground\""
            ],
            "cpu": 10,
            "secrets": [
                {
                    "valueFrom":
 "arn:aws:secretsmanager:region:aws_account_id:secret:username_value",
                    "name": "username_value"
                }
            ],
            "memory": 300,
            "image": "httpd:2.4",
            "essential": true,
            "name": "ecs-secrets-container"
        }
    ],
    "family": "ecs-secrets-tutorial"
}
```

6. 選擇 Create (建立)。

步驟 4：建立叢集

您可以使用 Amazon ECS 主控台建立一個叢集，其中包含要執行任務的容器執行個體。如果您的現有叢集有至少一個向其註冊的容器執行個體，並有可用資源可執行為此教學課程建立的一個任務定義執行個體，您可以跳到下一個步驟。

在本教學課程中，我們將使用 Amazon ECS 最佳化 Amazon Linux 2 AMI 建立具有一個 `t2.micro` 容器執行個體的叢集。

如需有關如何建立 EC2 啟動類型叢集的相關資訊，請參閱 the section called “為 Amazon EC2 啟動類型建立叢集”。

步驟 5：執行任務

您可以透過 Amazon ECS 主控台，使用您建立的任務定義來執行任務。在本教學課程中，我們將會執行使用 EC2 啟動類型的任務，並使用我們在前一個步驟中建立的叢集。

如需有關如何執行任務的資訊，請參閱 the section called “將應用程式執行為任務”。

步驟 6：驗證

您可以使用下列步驟，驗證是否已成功完成所有步驟，以及是否已在您的容器中正確建立環境變數。

驗證是否已建立環境變數

1. 尋找您容器執行個體的公有 IP 或 DNS 地址。
   a. 開啟主控台，網址為 https://console.aws.amazon.com/ecs/v2。
   b. 在導覽窗格中，選擇叢集，然後選擇您建立的叢集。
   c. 選擇基礎設施，然後選擇容器執行個體。
   d. 記錄您執行個體的公有 IP 或公有 DNS。
2. 如果您使用的是 macOS 或 Linux 電腦，請使用下列命令連線到您的執行個體，並替換為您私有金鑰的路徑及執行個體的公有地址：

   ```
   $ ssh -i /path/to/my-key-pair.pem ec2-user@ec2-198-51-100-1.compute-1.amazonaws.com
   ```

   如需使用 Windows 電腦的詳細資訊，請參閱《Amazon EC2 使用者指南》中的使用 PuTTY 連線至 Linux 執行個體。

> **Important**
>
> 如需連線到執行個體時的任何問題的詳細資訊，請參閱《Amazon EC2 使用者指南》中的連線至執行個體的故障診斷。

3. 列出在執行個體上執行的容器。請記下 `ecs-secrets-tutorial` 容器的容器 ID。

```
docker ps
```

4. 使用上一個步驟輸出中的容器 ID，連接到 `ecs-secrets-tutorial` 容器。

```
docker exec -it container_ID /bin/bash
```

5. 使用 `echo` 命令來列印環境變數的值。

```
echo $username_value
```

如果教學課程成功，您應該會看到以下輸出：

```
password_value
```

> **Note**
>
> 或者，您可以使用 `env` (或 `printenv`) 命令列出您容器中的所有環境變數。

步驟 7：清除

完成此教學課程時，建議您清除相關聯的資源，以免未使用的資源產生費用。

清除資源

1. 開啟主控台，網址為 https://console.aws.amazon.com/ecs/v2。
2. 在導覽窗格中，選擇叢集。
3. 在叢集頁面上，選擇叢集。
4. 選擇 Delete Cluster (刪除叢集)。
5. 在確認方塊中，輸入 delete ####，然後選擇刪除。

6. 在以下網址開啟 IAM 主控台：https://console.aws.amazon.com/iam/。
7. 在導覽窗格中，選擇角色。
8. 搜尋 `ecsTaskExecutionRole` 的角色清單並加以選取。
9. 選擇許可，然後選擇 ECSSecretsTutorial 旁邊的 X。選擇移除。
10. 開啟位於的 Secrets Manager 主控台https://console.aws.amazon.com/secretsmanager/。
11. 選取您所建立的 username_value 秘密，然後選擇 Actions (動作)、Delete secret (刪除秘密)。

Amazon ECS 任務定義參數

任務定義分為不同的部分：任務系列、 AWS Identity and Access Management (IAM) 任務角色、網路模式、容器定義、磁碟區、任務置放限制和啟動類型。任務定義中需要系列和容器定義。相對的，任務角色、網路模式、磁碟區、任務置放限制及啟動類型則為選用的。

您可以在 JSON 檔案中使用這些參數來設定任務定義。

以下是每個任務定義參數更詳細的描述。

系列

`family`

類型：字串

必要：是

當您註冊任務定義時，您會指定它的系列，這類似於任務定義以修訂版號碼指定的多個版本名稱。在特定系列註冊的第一個任務定義會得到修訂版 1，之後註冊的任何任務定義，則得到連續的修訂版號碼。

啟動類型

當您註冊任務定義時，您可以指定 Amazon ECS 驗證任務定義所依據的啟動類型。如果任務定義不會依據指定的相容性進行驗證，則會傳回用戶端例外狀況。如需詳細資訊，請參閱Amazon ECS啟動類型。

任務定義允許使用以下參數。

`requiresCompatibilities`

類型：字串陣列

必要：否

有效值: EC2 | FARGATE | EXTERNAL

驗證任務定義所依據的啟動類型。這會啟動檢查，以確保任務定義中使用的所有參數都能滿足啟動類型的要求。

任務角色

`taskRoleArn`

類型：字串

必要：否

當您註冊任務定義時，您可以為 IAM 角色提供任務角色，該角色可讓任務許可中的容器，代您呼叫其關聯政策中指定的 AWS API。如需詳細資訊，請參閱Amazon ECS 任務 IAM 角色。

在您啟動 Amazon ECS 最佳化 Windows Server AMI 時，Windows 上任務的 IAM 角色要求設定 - `EnableTaskIAMRole` 選項。您的容器還必須執行一些組態程式碼，以使用此功能。如需詳細資訊，請參閱 Amazon EC2 Windows 執行個體額外組態。

任務執行角色

`executionRoleArn`

類型：字串

必要：有條件

任務執行角色的 Amazon Resource Name (ARN)，授予 Amazon ECS 容器代理程式代表您進行 AWS API 呼叫的許可。

> **Note**
>
> 視任務需求而定，任務執行 IAM 角色是必要的項目。如需詳細資訊，請參閱Amazon ECS 任務執行 IAM 角色。

網路模式

`networkMode`

類型：字串

必要：否

任務中的容器所使用的 Docker 聯網模式。對於託管在 Amazon EC2 Linux 執行個體上的 Amazon ECS 任務，有效值為 `none`、`bridge`、 `awsvpc`和 `host`。如果未指定網路模式，則預設網路模式為 `bridge`。對於 Amazon EC2 Windows 執行個體上託管的 Amazon ECS 任務，有效值為 `default`、 和 `awsvpc`。如果未指定網路模式，則會使用`default`網路模式。對於在 Fargate 上託管的 Amazon ECS 任務，需要`awsvpc`網路模式。

如果網路模式設為 `none`，任務的容器即沒有外部連線，並且無法在容器定義中指定連接埠映射。

如果網路模式為 `bridge`，任務會使用 Linux 上的 Docker 內建虛擬網路，該虛擬網路會在託管任務的每個 Amazon EC2 執行個體內執行。Linux 上的內建虛擬網路會使用 `bridge` Docker 網路驅動程式。

如果網路模式為 `host`，任務會使用主機網路，並直接將容器連接埠映射至託管任務的 Amazon EC2 執行個體 ENI，以略過 Docker 的內建虛擬網路。動態連接埠映射無法在此網路模式下使用。任務定義中使用此模式的容器必須指定特定 `hostPort` 數字。主機上的連接埠號碼無法由多個任務使用。因此，您無法在單一 Amazon EC2 執行個體上執行同一個任務定義的多個任務。

> **Important**
>
> 執行使用 `host` 網路模式的任務時，為了獲得更好的安全性，不要使用根使用者 (UID 0) 執行容器。作為最佳安全實務，請一律使用非根使用者。

對於 Amazon EC2 啟動類型，如果網路模式為 `awsvpc`，則任務會配置彈性網路介面，而且您必須在建立服務或執行任務定義的任務`NetworkConfiguration`時指定 。如需詳細資訊，請參閱EC2 啟動類型的 Amazon ECS 任務聯網選項。

如果網路模式為 `default`，任務會使用 Windows 上的 Docker 內建虛擬網路，該虛擬網路會在託管任務的每個 Amazon EC2 執行個體內執行。Windows 上的內建虛擬網路會使用 `nat` Docker 網路驅動程式。

對於 Fargate 啟動類型，當網路模式為 時`awsvpc`，任務會配置彈性網路介面，而且您必須在建立服務或執行任務定義的任務`NetworkConfiguration`時指定 。如需詳細資訊，請參閱Fargate 啟

動類型的 Amazon ECS 任務聯網選項。awsvpc 網路模式為容器提供最高聯網效能，因為它們使用的是 Amazon EC2 網路堆疊。公開的容器連接埠會直接映射至連接的彈性網路介面連接埠。因此，您無法使用動態主機連接埠對應。

host 和 awsvpc 網路模式為容器提供最高聯網效能，因為它們使用的是 Amazon EC2 網路堆疊。使用 host 和 awsvpc 網路模式，公開的容器連接埠會直接映射到對應的主機連接埠 (適用於 host 網路模式) 或已連接的彈性網路介面連接埠 (適用於 awsvpc 網路模式)。因此，您無法使用動態主機連接埠對應。

如果使用 Fargate 啟動類型，則需要 awsvpc 網路模式。如果使用 EC2 啟動類型，允許的網路模式取決於基礎 EC2 執行個體的作業系統。如果是 Linux，則可以使用任何網路模式。如果是 Windows，則會使用 default 和 awsvpc 模式。

執行時間平台

operatingSystemFamily

類型：字串

必要：有條件

預設：LINUX

在 Fargate 上託管的 Amazon ECS 任務需要此參數。

在註冊任務定義時，您會指定作業系統系列。

在 Fargate 上託管的 Amazon ECS 任務的有效值為 LINUX、WINDOWS_SERVER_2019_FULL、WINDOWS_SERVER_2019_CORE、WINDOWS_SERVER_2022_ 和 WINDOWS_SERVER_2022_CORE 。

在 EC2 上託管的 Amazon ECS 任務的有效值為 LINUX、WINDOWS_SERVER_2022_CORE、WINDOWS_SERVER_2022_FULL、WINDOWS_SERVER_2019_ 以及 WINDOWS_SERVER_2019_CORE、WINDOWS_SERVER_2016_FULL、WINDOWS_SERVER_2004_CORE 和 WINDOWS_SERVER_20H2_CORE。

服務中使用的所有任務定義都必須與此參數具有相同的值。

如果任務定義為服務的一部分，此值必須與服務 platformFamily 值相符。

`cpuArchitecture`

類型：字串

必要：有條件

預設：X86_64

在 Fargate 上託管的 Amazon ECS 任務需要此參數。如果參數保留為 `null`，則會在 Fargate 上託管的任務啟動時自動指派預設值。

在註冊任務定義時，您會指定 CPU 架構。有效值為 `X86_64` 和 `ARM64`。

服務中使用的所有任務定義都必須與此參數具有相同的值。

若您具有 Fargate 啟動類型或 EC2 啟動類型的 Linux 任務，則可將值設定為 `ARM64`。如需詳細資訊，請參閱the section called “64 位元 ARM 工作負載的任務定義”。

任務大小

當您註冊任務定義時，您可以指定任務使用的 CPU 和記憶體總計。這和容器定義層級的 `cpu` 和 `memory` 值是分開的。對於在 Amazon EC2 執行個體上託管的任務，這些欄位是選用欄位。對於在 Fargate (Linux 和 Windows 皆可) 上託管的任務，這些欄位為必填欄位，而且受支援的 `cpu` 和 `memory` 具有特定值。

> **Note**
>
> Windows 容器會忽略任務層級的 CPU 和記憶體參數。我們建議為 Windows 容器指定容器層級的資源。

任務定義允許使用以下參數：

`cpu`

類型：字串

必要：有條件

> **Note**
>
> Windows 容器不支援此參數。

針對任務呈現的 CPU 單位硬性限制。您可以將 JSON 檔案中的 CPU 值指定為 CPU 單位或虛擬 CPUsvCPUs) 中的字串。例如，您可以在 CPU `1024` 單位或 vCPUs`1` vCPU中指定 CPU 值。註冊任務定義時，vCPU 值會轉換為整數，指出 CPU 單位。

對於在 EC2 或外部執行個體上執行的任務，此欄位是選用欄位。如果您的叢集沒有任何一個已註冊的容器執行個體具有可用的請求 CPU 單位，則任務會失敗。在 EC2 或外部執行個體上執行的任務支援值介於 `0.125` vCPU 和 `10` vCPU 之間。

對於在 Fargate 上 (Linux 和 Windows 容器皆可) 執行的任務，此欄位為必填欄位，而且您必須使用下列其中一個值，以決定 `memory` 參數的受支援值範圍。下表顯示有效的任務層級 CPU 和記憶體組合。

CPU 數值	記憶體數值	AWS Fargate 支援的作業系統
256 (.25 vCPU)	512 MiB、1 GB、2 GB	Linux
512 (.5 vCPU)	1 GB、2 GB、3 GB、4 GB	Linux
1024 (1 vCPU)	2 GB、3 GB、4 GB、5 GB、6 GB、7 GB、8 GB	Linux、Windows
2048 (2 vCPU)	介於 4 GB 與 16 GB 之間，以 1 GB 為單位遞增	Linux、Windows
4096 (4 vCPU)	介於 8 GB 與 30 GB 之間，以 1 GB 為單位遞增	Linux、Windows
8192 (8 vCPU)	介於 16 GB 與 60 GB 之間，以 4 GB 為單位遞增	Linux

CPU 數值	記憶體數值	AWS Fargate 支援的作業系統
Note 此選項需要 Linux 平台 `1.4.0` 或更新版本。		
16384 (16vCPU) **Note** 此選項需要 Linux 平台 `1.4.0` 或更新版本。	介於 32 GB 與 120 GB 之間，以 8 GB 為單位遞增	Linux

`memory`

類型：字串

必要：有條件

> **Note**
> Windows 容器不支援此參數。

要呈現給任務的記憶體硬性限制。您可以在任務定義中將記憶體值指定為以 MB (MiB) 或 gigabyte (GB) 為單位的字串。例如，您可以指定 MiB `3072` 或 GB `3 GB`的記憶體值。註冊任務定義時，GB 值會轉換為整數，指出 MiB。

對於在 Amazon EC2 執行個體上託管的任務，此欄位為選用欄位，可使用任意值。如果指定任務層級的記憶體值，則容器層級的記憶體值是選用的。如果您的叢集沒有任何一個已註冊的容器執行個體具有可用的請求記憶體，則任務會失敗。您可以盡可能為特定執行個體類型的任務提供最多的記憶體，以將資源使用率最大化。如需詳細資訊，請參閱保留 Amazon ECS Linux 容器執行個體記憶體。

對於在 Fargate 上 (Linux 和 Windows 皆可) 託管的任務，此欄位為必填欄位，而且您必須使用下列其中一個值，以決定 cpu 參數的受支援值的範圍：

記憶體值 （以 MiB 為單位，近似相等值以 GB 為單位）	CPU 數值	Fargate 支援的作業系統
512 (0.5 GB)、1024 (1 GB)、2048 (2 GB)	256 (.25 vCPU)	Linux
1024 (1 GB)、2048 (2 GB)、3072 (3 GB)、4096 (4 GB)	512 (.5 vCPU)	Linux
2048 (2 GB)、3072 (3 GB)、4096 (4GB)、5120 (5 GB)、6144 (6 GB)、7168 (7 GB)、8192 (8 GB)	1024 (1 vCPU)	Linux、Windows
介於 4096 (4 GB) 與 16384 (16 GB) 之間，以 1024 (1 GB) 為單位遞增	2048 (2 vCPU)	Linux、Windows
介於 8192 (8 GB) 與 30720 (30 GB) 之間，以 1024 (1 GB) 為單位遞增	4096 (4 vCPU)	Linux、Windows
介於 16 GB 與 60 GB 之間，以 4 GB 為單位遞增 **Note** 此選項需要 Linux 平台 `1.4.0` 或更新版本。	8192 (8 vCPU)	Linux
介於 32 GB 與 120 GB 之間，以 8 GB 為單位遞增	16384 (16vCPU)	Linux

記憶體值 （以 MiB 為單位，近似相等值以 GB 為單位）	CPU 數值	Fargate 支援的作業系統
Note 此選項需要 Linux 平台 1.4.0 或更新版本。		

容器定義

註冊任務定義時，您必須指定容器定義清單，以傳遞到容器執行個體上的 Docker 常駐程式。容器定義允許使用以下參數。

主題

- 標準容器定義參數
- 進階容器定義參數
- 其他容器定義參數

標準容器定義參數

下列任務定義參數為必要參數，或是會在大部分的容器定義中使用。

主題

- 名稱
- 映像
- 記憶體
- 連接埠映射
- 私有儲存庫憑證

名稱

name

類型：字串

必要：是

容器的名稱。可以包含最多可達 255 個字元 (大小寫)、數字、連字號和底線。如果您在任務定義中連結多個容器，則一個容器的 name 可以輸入另一個容器的 links。這是為了連結容器。

映像

image

類型：字串

必要：是

用來啟動容器的映像。此字串會直接傳遞至 Docker 常駐程式。根據預設，Docker Hub 登錄檔中的映像為可用。您也可以使用 *repository-url*/*image*:*tag* 或 *repository-url*/*image*@*digest* 指定其他存放庫。允許最多 255 個字元 (大小寫)、數字、連字號、底線、等號、句號、正斜線、井號。此參數會在 docker create-container 命令和 docker run 命令的 IMAGE 參數Image中映射至 。

- 當新的任務啟動時，Amazon ECS 容器代理會提取最新版本的指定映像和標籤，以供容器使用。但是，儲存庫映像後續的更新將不會散佈到已在執行中的任務。
- 當您未在任務定義中的映像路徑中指定標籤或摘要時，Amazon ECS 容器代理程式會提取指定映像的最新版本。
- 但是，儲存庫映像後續的更新將不會散佈到已在執行中的任務。
- 支援私有登錄檔中的映像。如需詳細資訊，請參閱在 Amazon ECS 中使用非AWS 容器映像。
- 可使用完整的 registry/repository:tag 或 registry/repository@digest 命名慣例來指定 Amazon ECR 儲存庫中的映像 (例如 *aws_account_id*.dkr.ecr.*region*.amazonaws.com/*my-web-app*:*latest* 或 *aws_account_id*.dkr.ecr.*region*.amazonaws.com/*my-web-app*@*sha256:94afd1f2e64d908bc90dbca0035a5b567EXAMPLE*)。
- Docker Hub 上官方儲存庫中的映像，使用的是單一名稱 (例如，ubuntu 或 mongo)。
- Docker Hub 上的其他儲存庫中的映像要求使用組織名稱 (例如，amazon/amazon-ecs-agent)。
- 其他線上儲存庫中的映像更進一步要求使用網域名稱 (例如，quay.io/assemblyline/ubuntu)。

versionConsistency

類型：字串

有效值：enabled|disabled

必要：否

指定 Amazon ECS 是否會將容器定義中提供的容器映像標籤解析為映像摘要。根據預設，此行為為 enabled。如果您將容器的值設定為 disabled，Amazon ECS 不會將容器映像標籤解析為摘要，並將使用容器定義中指定的原始映像 URI 進行部署。如需容器映像解析度的詳細資訊，請參閱容器映像解析度。

記憶體

memory

類型：整數

必要：否

提供給容器使用的記憶體數量 (MiB)。如果您的容器嘗試使用超過此處指定的記憶體，容器便會終止。為任務中所有容器預留的記憶體總量必須低於任務 memory 值 (如果已指定一個)。此參數在 docker create-container 命令Memory中映射到 ，以及 docker 執行--memory的選項。

如果使用 Fargate 啟動類型，這是選用參數。

如果使用 EC2 啟動類型，您必須指定任務層級的記憶體值或容器層級的記憶體值。如果您同時指定容器層級的 memory 和 memoryReservation 值,則 memory 值必須大於 memoryReservation 值。如果指定 memoryReservation，則會從放置容器的容器執行個體可用記憶體資源中減去該值。否則，即使用 memory 的值。

Docker 20.10.0 或更新版本的常駐程式會為容器保留最低 6 MiB 的記憶體。因此，不要為容器指定少於 6 MiB 的記憶體。

Docker 19.03.13-ce 或更舊版本的常駐程式會為容器保留最低 4 MiB 的記憶體。因此，不要為容器指定少於 4 MiB 的記憶體。

> **Note**
>
> 若您嘗試盡可能為特定執行個體類型的任務提供最多的記憶體，以將資源使用率最大化，請參閱「保留 Amazon ECS Linux 容器執行個體記憶體」。

`memoryReservation`

類型：整數

必要：否

為容器保留的記憶體軟性限制 (MiB)。當系統記憶體爭用時，Docker 會嘗試將容器記憶體保持在此軟性限制。但是，您的容器可以在需要時使用更多記憶體。容器可以使用最多達到以 `memory` 參數指定的硬性限制 (如適用)，或容器執行個體上的所有可用記憶體，以先達到者為準。此參數會映射至 docker create-container 命令`MemoryReservation`中的 ，以及 docker 執行`--memory-reservation`的選項。

如果未指定任務層級的記憶體值，您必須為容器定義中的 `memory` 或 `memoryReservation` 之一或兩者指定非零整數。若您同時指定兩者，`memory` 必須大於 `memoryReservation`。如果指定 `memoryReservation`，則會從放置容器的容器執行個體可用記憶體資源中減去該值。否則，即使用 `memory` 的值。

例如，若您的容器通常會使用 128 MiB 的記憶體，但會偶爾短期爆量到 256 MiB 的記憶體。您可以將 `memoryReservation` 設為 128 MiB，並將 `memory` 硬性限制設為 300 MiB。此組態讓容器只從容器執行個體的剩餘資源中保留 128 MiB 的記憶體。同時，其還允許容器在需要時使用更多的記憶體資源。

> **Note**
>
> Windows 容器不支援此參數。

Docker 20.10.0 或更新版本的常駐程式會為容器保留最低 6 MiB 的記憶體。因此，不要為容器指定少於 6 MiB 的記憶體。

Docker 19.03.13-ce 或更舊版本的常駐程式會為容器保留最低 4 MiB 的記憶體。因此，不要為容器指定少於 4 MiB 的記憶體。

> **Note**
>
> 若您嘗試盡可能為特定執行個體類型的任務提供最多的記憶體，以將資源使用率最大化，請參閱「保留 Amazon ECS Linux 容器執行個體記憶體」。

連接埠映射

`portMappings`

類型：物件陣列

必要：否

連接埠映射會將容器的網路連接埠公開給外部世界。這可讓用戶端存取您的應用程式。它也用於相同任務中的容器間通訊。

對於使用`awsvpc`網路模式的任務定義 (Fargate 和 EC2 啟動類型），請僅指定 `containerPort`。`hostPort` 一律會忽略 ，且容器連接埠會自動對應至主機上的隨機高編號連接埠。

Windows 上的連接埠映射會使用 `NetNAT` 閘道地址，而非 `localhost`。Windows 沒有連接埠映射回送，所以您無法從主機本身存取容器的映射連接埠。

此參數的大多數欄位（包括 `containerPort`、`hostPort`、`protocol`) 都會映射至 docker create-container 命令`PortBindings`中的 ，以及 docker 執行`--publish`的選項。若任務定義的網路模式設定為 `host`，則主機連接埠必須為未定義，或符合連接埠映射中的容器連接埠。

> **Note**
>
> 任務達到 `RUNNING` 狀態後，手動和自動主機及容器連接埠指派會顯示在下列位置：
>
> - 主控台：選取任務的容器說明的 Network Bindings (網路繫結) 區段。
> - AWS CLI：describe-tasks 命令輸出的 `networkBindings` 區段。
> - API：`DescribeTasks` 的回應。
> - 中繼資料：任務中繼資料端點。

`appProtocol`

類型：字串

必要：否

用於連接埠映射的應用程式通訊協定。此參數僅適用於 Service Connect。建議您將此參數設定為與您應用程式使用的通訊協定一致。如果您設定此參數，Amazon ECS 會將通訊協定專屬連

線處理新增至 Service Connect Proxy。如果您設定此參數，Amazon ECS 會在 Amazon ECS 主控台和 CloudWatch 中新增通訊協定專屬遙測。

如果您沒有為此參數設定值，系統會使用 TCP。不過，Amazon ECS 不會針對 TCP 新增通訊協定特定遙測。

如需詳細資訊，請參閱the section called “ Service Connect”。

有效的通訊協定值："HTTP" | "HTTP2" | "GRPC"

`containerPort`

類型：整數

必要：是 (當使用 `portMappings` 時)

容器上的連接埠號碼，該號碼繫結到使用者指定或自動指派的主機連接埠。

對於使用 Fargate 啟動類型或使用`awsvpc`網路模式的 EC2 任務，您可以使用 `containerPort` 來指定公開連接埠。

對於 Fargate 上的 Windows 容器，您不得將連接埠 3150 用於 `containerPort`。這是因為其已保留。

假設您在 EC2 啟動類型的任務中使用容器，並且指定容器連接埠而非主機連接埠。然後，您的容器會自動在暫時性連接埠範圍中接收到主機連接埠。如需詳細資訊，請參閱`hostPort`。使用此方式自動指派的連接埠映射不包含在容器執行個體 100 個保留連接埠的配限制中。

`containerPortRange`

類型：字串

必要：否

綁定到動態映射主機連接埠範圍之容器上的連接埠號碼範圍。

您只能使用 `register-task-definition` API 來設定此參數。此選項可在 `portMappings` 參數中使用。如需詳細資訊，請參閱在 參考資料AWS Command Line Interface 中的 register-task-definition。

指定 `containerPortRange` 時，以下規則適用：

- 您必須使用 `bridge` 網路模式或 `awsvpc` 網路模式。

- 此參數適用於 EC2 和 AWS Fargate 啟動類型。
- 此參數適用於 Linux 和 Windows 作業系統。
- 容器執行個體必須至少具有 1.67.0 版的容器代理程式以及至少 1.67.0-1 版的 `ecs-init` 套件。
- 每個容器最多可以指定 100 個連接埠範圍。
- 您不會指定 `hostPortRange`。`hostPortRange` 的值設定如下：
  - 對於具有 `awsvpc` 網路模式的任務中的容器，`hostPort` 會設定為與 `containerPort` 相同的值。這是靜態映射策略。
  - 對於具有 `bridge` 網路模式的任務中的容器，Amazon ECS 代理程式會從預設暫時性範圍尋找開放的主機連接埠並傳送到 Docker，以將主機連接埠綁定到容器連接埠。
- `containerPortRange` 有效值介於 1 到 65535。
- 一個連接埠只能包含在每個容器的一個連接埠映射中。
- 您無法指定重疊的連接埠範圍。
- 範圍中的第一個連接埠必須小於範圍中的最後一個連接埠。
- Docker 建議您在擁有大量連接埠時關閉 Docker 常駐程式組態檔案中的 Docker 代理。

  如需詳細資訊，請參閱 GitHub 上的問題 #11185。

  如需有關如何在 Docker 常駐程式組態檔案中關閉 Docker 代理的詳細資訊，請參閱《Amazon ECS 開發人員指南》中的 Docker 常駐程式。

您可以呼叫 `DescribeTasks` 來檢視 `hostPortRange`，此即綁定到容器連接埠的主機連接埠。

連接埠範圍未包含在傳送到 EventBridge 的 Amazon ECS 任務事件中。如需詳細資訊，請參閱the section called “使用 EventBridge 自動化對 Amazon ECS 錯誤的回應”。

`hostPortRange`

類型：字串

必要：否

與網路繫結搭配使用之主機上的連接埠號碼範圍。這由 Docker 指派，並由 Amazon ECS 代理程式交付。

`hostPort`

類型：整數

必要：否

要為您的容器保留之容器執行個體上的連接埠號碼。

如果在使用 Fargate 啟動類型的任務中使用容器，hostPort 可維持空白，或其值與 containerPort 相同。

假設您在 EC2 啟動類型的任務中使用容器。您可以為容器連接埠對應指定非保留的主機連接埠。這稱為靜態主機連接埠對應。或者，您可以在指定 containerPort 時省略 hostPort (或將其設定為 0)。您的容器會針對您的容器執行個體作業系統和 Docker 版本，自動接收暫時性連接埠範圍內的連接埠。這稱為動態主機連接埠對應。

Docker 1.6.0 版和更新版本的預設暫時性連接埠範圍列在 /proc/sys/net/ipv4/ip_local_port_range 下的執行個體上。如果此核心參數不可用，會使用預設的 49153–65535 暫時性連接埠範圍。請勿嘗試在暫時性連接埠範圍中指定主機連接埠。這是因為其已保留以便自動指派。一般而言，低於 32768 的連接埠便會位於暫時性連接埠範圍之外。

SSH 的預設保留連接埠為 22，Docker 連接埠為 2375 和 2376，Amazon ECS 容器代理程式連接埠則為 51678-51680。任何使用者先前為執行中任務指定的主機連接埠也會在執行任務時保留。在任務停止後，便會釋放主機連接埠。目前預留的連接埠會顯示在 describe-container-instances 輸出的 remainingResources 中。容器執行個體一次最多可預留 100 個連接埠，包括預設的預留連接埠。自動指派的連接埠不計入 100 個預留連接埠的限制。

name

類型：字串

必要：否，在服務中設定 Service Connect 和 VPC Lattice 的必要項目

用於連接埠映射的名稱。此參數僅適用於 Service Connect 和 VPC Lattice。此參數是您用於服務之 Service Connect 和 VPC Lattice 組態的名稱。

如需詳細資訊，請參閱使用 Service Connect 以短名稱連接 Amazon ECS 服務。

在下列範例中，會使用 Service Connect 和 VPC Lattice 的兩個必要欄位。

```
"portMappings": [
    {
        "name": string,
        "containerPort": integer
    }
```

```
]
```

`protocol`

類型：字串

必要：否

用於連接埠映射的協定。有效值為 `tcp` 和 `udp`。預設值為 `tcp`。

> **Important**
>
> Service Connect 僅支援 `tcp`。請記住，如果未設定此欄位，則會隱含 `tcp`。

> **Important**
>
> 只有在使用 1.2.0 版或更新版本的 Amazon ECS 容器代理程式 (例如 `amzn-ami-2015.03.c-amazon-ecs-optimized` AMI)，或是使用已更新到 1.3.0 版或更新版本的容器代理程式而啟動的容器執行個體上，才能使用 UDP 支援。若要將您的容器代理更新到最新版本，請參閱「更新 Amazon ECS 容器代理程式」。

若您指定主機連接埠，請使用以下語法。

```
"portMappings": [
    {
        "containerPort": integer,
        "hostPort": integer
    }
    ...
]
```

若您希望取得自動指派的主機連接埠，請使用以下語法。

```
"portMappings": [
    {
        "containerPort": integer
    }
```

```
    ...
]
```

私有儲存庫憑證

`repositoryCredentials`

類型：RepositoryCredentials 物件

必要：否

私有登錄檔身分驗證的儲存庫登入資料。

如需詳細資訊，請參閱在 Amazon ECS 中使用非AWS 容器映像。

`credentialsParameter`

類型：字串

必要：是 (當使用 `repositoryCredentials` 時)

包含私有儲存庫憑證密碼的 Amazon Resource Name (ARN)。

如需詳細資訊，請參閱在 Amazon ECS 中使用非AWS 容器映像。

> **Note**
>
> 當您使用 Amazon ECS API AWS CLI或 AWS SDKs 時，如果秘密與您啟動的任務位於相同的區域中，您可以使用完整的 ARN 或秘密的名稱。使用 時 AWS Management Console，您必須指定秘密的完整 ARN。

以下是任務定義的程式碼片段，其會顯示所需的參數：

```
"containerDefinitions": [
    {
        "image": "private-repo/private-image",
        "repositoryCredentials": {
            "credentialsParameter":
 "arn:aws:secretsmanager:region:aws_account_id:secret:secret_name"
        }
```

```
    }
]
```

進階容器定義參數

下列進階容器定義參數為 Amazon ECS 容器執行個體上用來啟動容器的 docker run 命令提供擴充功能。

主題

- 重新啟動政策
- 運作狀態檢查
- 環境
- Network settings (網路設定)
- 儲存與記錄
- 安全
- 資源限制
- Docker 標籤

重新啟動政策

`restartPolicy`

容器重新啟動政策和相關聯的組態參數。當您為容器設定重新啟動政策時，Amazon ECS 可以重新啟動容器，而不需要取代任務。如需詳細資訊，請參閱使用容器重新啟動政策重新啟動 Amazon ECS 任務中的個別容器。

`enabled`

類型：布林值

必要：是

指定是否為容器啟用重新啟動政策。

`ignoredExitCodes`

類型：Integer array

必要：否

Amazon ECS 會忽略且不嘗試重新啟動的結束代碼清單。您最多可以指定 50 個容器結束代碼。根據預設，Amazon ECS 不會忽略任何結束代碼。

`restartAttemptPeriod`

類型：整數

必要：否

容器必須執行一段時間（以秒為單位），才能嘗試重新啟動。容器每`restartAttemptPeriod`秒鐘只能重新啟動一次。如果容器無法在此期間執行並提早結束，則不會重新啟動。您可以設定最少 `restartAttemptPeriod` 60 秒，最多 `restartAttemptPeriod` 1800 秒。根據預設，容器必須執行 300 秒才能重新啟動。

運作狀態檢查

`healthCheck`

用於容器的容器運作狀態檢查命令和相關的設定參數。如需詳細資訊，請參閱使用容器運作狀態檢查判斷 Amazon ECS 任務運作狀態。

`command`

表示容器執行的命令的字串陣列，用於確定運作狀態是否良好。此字串陣列的開頭可以是 `CMD`，如此能直接執行命令引數；或是 `CMD-SHELL`，藉以使用容器預設的 shell 來執行命令。如果均尚未指定，則使用 `CMD`。

在 中註冊任務定義時 AWS Management Console，請使用逗號分隔的命令清單。這些命令會在建立任務定義後轉換為字串。以下為運作狀態檢查的範例輸入。

```
CMD-SHELL, curl -f http://localhost/ || exit 1
```

使用 JSON AWS Management Console 面板、 AWS CLI或 APIs 註冊任務定義時，請將命令清單括在括號中。以下為運作狀態檢查的範例輸入。

```
[ "CMD-SHELL", "curl -f http://localhost/ || exit 1" ]
```

結束代碼 0 (沒有 `stderr` 輸出) 表示成功，而任何非零的結束代碼則表示失敗。

`interval`

每個運作狀態檢查的時間間隔 (以秒為單位)。您可以指定 5 至 300 秒之間的值。預設 值為 30 秒。

`timeout`

判定為失敗之前，等待運作狀態檢查成功執行的時間 (以秒為單位)。您可以指定 2 至 60 秒之間的值。預設值為 5 秒。

`retries`

容器運作狀態判定為不良之前，重試失敗的運作狀態檢查的次數上限。您可以指定 1 至 10 次嘗試。預設值為重試三次。

`startPeriod`

選用的寬限期，讓容器有時間引導，不將失敗的運作狀態檢查計入重試次數上限。您可以指定 0 至 300 秒之間的值。預設情況下，`startPeriod` 是停用的。

如果運作狀態檢查在 `startPeriod` 內成功，則代表容器運作狀態良好，之後的任何故障都會計入，累積至重試次數上限。

環境

`cpu`

類型：整數

必要：否

Amazon ECS 容器代理程式保留給容器的 `cpu` 數量。在 Linux 上，此參數會映射到建立容器區段`CpuShares`中的 。

針對使用 Fargate 啟動類型的任務，此欄位為選用。為任務中所有容器預訂的 CPU 總量必須低於任務階層的 `cpu` 值。

> **Note**
>
> 您可以決定每個 Amazon EC2 執行個體類型可用的 CPU 單位數。若要執行此操作，請將 Amazon EC2 執行個體詳細資訊頁面上為執行個體類型列出的 vCPU 數乘上 1,024。

Linux 容器會按照與其配置數量相同的比例，與容器執行個體上的其他容器共用未配置的 CPU 單位。例如，若您在單一核心執行個體類型上執行單一容器任務，為該容器指定 512 個 CPU 單位。此外，該任務是在容器執行個體上運行的唯一任務。在此範例中，該容器便可在任何指定的時間內使用完整 1,024 個 CPU 單位。但是，假設您在該容器執行個體上啟動相同任務的另一個複本。每個任務都可以保證在需要的時候取得最少 512 個 CPU 單位。同樣，如果其他容器沒有使用剩餘的 CPU，每個容器都可以浮動到更高的 CPU 用量。但若兩個任務都一直維持在 100% 作用中的狀態，兩者能夠取得的單位數便會限制在 512 個 CPU 單位。

在 Linux 容器執行個體上，容器執行個體的 Docker 常駐程式會使用 CPU 值計算執行中容器的相對 CPU 共用比例。Linux 核心允許的最小有效 CPU 共享值為 2，Linux 核心允許的最大有效 CPU 共享值為 262144。不過，CPU 參數並非必要項目，您可以在容器定義中使用低於 2 和高於 262144 的 CPU 值。對於低於兩個 （包括 null) 和高於 262144 的 CPU 值，行為會根據您的 Amazon ECS 容器代理程式版本而有所不同：

- 代理程式版本 <= 1.1.0：Null 和零 CPU 值會以 0 傳遞給 Docker。然後 Docker 將此值轉換成 1,024 個 CPU 共享。CPU 值若為 1，則會以 1 傳遞給 Docker，接著 Linux 核心便會轉換成兩個 CPU 共享。
- 代理程式版本 >= 1.2.0：Null、零和值為一的 CPU 值會以兩個 CPU 共享傳遞給 Docker。
- 代理程式版本 >= 1.84.0：大於 256 vCPU 的 CPU 值會傳遞Docker至 256，相當於 262144 個 CPU 共享。

在 Windows 容器執行個體上，CPU 配額會強制為絕對配額。Windows 容器只能存取任務定義中定義的指定 CPU 數量。Null 或零的 CPU 值會以 0 形式傳遞給 Docker。然後 Windows 將此值解讀為一個 CPU 的 1%。

如需其他範例，請參閱 Amazon ECS 如何管理 CPU 和記憶體資源。

gpu

類型：ResourceRequirement 物件

必要：否

Amazon ECS 容器代理程式保留給容器的實體 GPUs 數量。為任務中所有容器保留的 GPU 數量不得超過任務啟動所在之容器執行個體上可用的 GPU 數量。如需詳細資訊，請參閱GPU 工作負載的 Amazon ECS 任務定義。

> **Note**
>
> Windows 容器或在 Fargate 上託管的容器不支援此參數。

`Elastic Inference accelerator`

類型：ResourceRequirement 物件

必要：否

對於 `InferenceAccelerator` 類型，`value` 符合任務定義中指定之 `InferenceAccelerator` 的 `deviceName`。如需詳細資訊，請參閱the section called “Elastic Inference 加速器名稱”。

> **Note**
>
> Windows 容器或在 Fargate 上託管的容器不支援此參數。

`essential`

類型：布林值

必要：否

假設容器的 `essential` 參數標記為 `true`，並且該容器因為任何理由失敗或停止。然後，會停止屬於該任務一部分的所有其他容器。若容器的 `essential` 參數標記為 `false`，則其失敗將不會影響任務中其餘的容器。若省略此參數，則容器會假設為「基本」。

所有任務都必須至少有一個基本容器。假設您有一個由多個容器組成的應用程式。然後，將用於一般用途的容器分組為元件，並把不同的元件分到多個任務定義。如需詳細資訊，請參閱為 Amazon ECS 建構您的應用程式。

```
"essential": true|false
```

`entryPoint`

> **Important**
>
> 舊版的 Amazon ECS 容器代理程式無法正確處理 `entryPoint` 參數。若您在使用 `entryPoint` 時發生任何問題，請更新您的容器代理，或改將您的命令和引數作為 `command` 陣列項目輸入。

類型：字串陣列

必要：否

傳遞至容器的進入點。

```
"entryPoint": ["string", ...]
```

`command`

類型：字串陣列

必要：否

傳遞至容器的命令。此參數會映射 create-container 命令`Cmd`中的 和要執行的 `COMMAND` 參數。如果有多個引數，確保每個引數在陣列中是分開的字串。

```
"command": ["string", ...]
```

`workingDirectory`

類型：字串

必要：否

容器內要執行命令的工作目錄。此參數會映射到 Docker Remote API 的建立容器區段中的 `WorkingDir` 以及 docker run 的 `--workdir` 選項。

```
"workingDirectory": "string"
```

`environmentFiles`

類型：物件陣列

必要：否

內含要傳遞至容器之環境變數的檔案清單。此參數會映射到 docker run 命令`--env-file`的選項。

使用 EC2 啟動類型在主機上啟用 FIPS 時，不支援具有句點 (.) 的儲存貯體名稱（例如 amzn-s3-demo-bucket1.name.example)。在儲存貯體名稱中具有句點 (.) 可防止任務啟動，因為代理程式無法從 Amazon S3 提取環境變數檔案。

這不適用於 Fargate 上的Windows容器和 Windows 容器

您最多可以指定 10 個環境檔案。檔案副檔名必須是 .env。環境檔案中的每一行都包含 VARIABLE=VALUE 格式的環境變數。以 # 開頭的行會被視為註解，而忽略。

如果在容器定義中指定了個別環境變數，它們的優先順序高於環境檔案中包含的變數。如果指定了內含相同變數的多個環境檔案，則處理順序為由上而下。建議您使用唯一的變數名稱。如需詳細資訊，請參閱將個別環境變數傳遞至 Amazon ECS 容器。

`value`

類型：字串

必要：是

包含環境變數檔案的 Amazon S3 物件的 Amazon Resource Name (ARN)。

`type`

類型：字串

必要：是

要使用的檔案類型。唯一支援的值為 s3。

`environment`

類型：物件陣列

必要：否

傳遞至容器的環境變數。此參數會在 docker create-container 命令Env中映射至 ，並將 --env選項映射至 docker run 命令。

> **Important**
> 不建議在敏感資訊 (例如登入資料) 使用純文字環境變數。

`name`

類型：字串

必要：是 (當使用 environment 時)

環境變數的名稱。

`value`

類型：字串

必要：是 (當使用 `environment` 時)

環境變數的值。

```
"environment" : [
    { "name" : "string", "value" : "string" },
    { "name" : "string", "value" : "string" }
]
```

`secrets`

類型：物件陣列

必要：否

代表公開到容器之秘密的物件。如需詳細資訊，請參閱將敏感資料傳遞至 Amazon ECS 容器。

`name`

類型：字串

必要：是

要設定為容器上環境變數的值。

`valueFrom`

類型：字串

必要：是

公開給容器的秘密。支援的值為 AWS Secrets Manager 秘密的完整 Amazon Resource Name (ARN)，或 AWS Systems Manager 參數存放區中參數的完整 ARN。

> **Note**
>
> 如果 Systems Manager 參數存放區參數或 Secrets Manager 參數與您啟動的任務位於相同的 AWS 區域 中，您可以使用完整的 ARN 或秘密的名稱。如果參數存在於不同區域，則必須指定完整 ARN。

```
"secrets": [
    {
        "name": "environment_variable_name",
        "valueFrom": "arn:aws:ssm:region:aws_account_id:parameter/parameter_name"
    }
]
```

Network settings (網路設定)

`disableNetworking`

類型：布林值

必要：否

當此參數為 true 時，聯網便會在容器中關閉。

> **Note**
>
> Windows 容器或使用 awsvpc 網路模式的任務不支援此參數。

預設值為 `false`。

```
"disableNetworking": true|false
```

`links`

類型：字串陣列

必要：否

`link` 參數可讓容器彼此通訊，而無需連接埠映射。只有在任務定義的網路模式設定為 `bridge` 時才支援此參數。`name:internalName` 建構模組與 Docker 連結中的 `name:alias` 類似。最多允許 255 個字母（大寫和小寫）、數字、連字號和底線。

> **Note**
>
> Windows 容器或使用 awsvpc 網路模式的任務不支援此參數。

> **Important**
>
> 在相同容器執行個體上配置的容器可和彼此通訊，而無需連結或主機連接埠映射。容器執行個體上的網路隔離是由安全群組和 VPC 設定所控制。

```
"links": ["name:internalName", ...]
```

`hostname`

類型：字串

必要：否

您容器要使用的主機名稱。此參數會映射至 docker create-container `Hostname`中的 ，以及 docker run `--hostname`的選項。

> **Note**
>
> 如果您使用 `awsvpc` 網路模式，則不支援 `hostname` 參數。

```
"hostname": "string"
```

`dnsServers`

類型：字串陣列

必要：否

提供給容器的 DNS 伺服器清單。

> **Note**
>
> Windows 容器或使用 `awsvpc` 網路模式的任務不支援此參數。

```
"dnsServers": ["string", ...]
```

`dnsSearchDomains`

類型：字串陣列

必要：否

模式：^[a-zA-Z0-9-.]{0,253}[a-zA-Z0-9]$

提供給容器的 DNS 搜尋網域清單。此參數在 docker create-container 命令`DnsSearch`中映射到 ，`--dns-search`選項用於 docker 執行。

> **Note**
>
> Windows 容器或使用 `awsvpc` 網路模式的任務不支援此參數。

```
"dnsSearchDomains": ["string", ...]
```

`extraHosts`

類型：物件陣列

必要：否

要附加到容器上 `/etc/hosts` 檔案的主機名稱和 IP 地址映射清單。

此參數在 docker create-container 命令`ExtraHosts`中映射到 ，以及 docker 執行`--add-host`的選項。

> **Note**
>
> Windows 容器或使用 `awsvpc` 網路模式的任務不支援此參數。

```
"extraHosts": [
      {
        "hostname": "string",
        "ipAddress": "string"
      }
      ...
```

```
    ]
```

`hostname`

類型：字串

必要：是 (當使用 `extraHosts` 時)

要在 `/etc/hosts` 項目中使用的主機名稱。

`ipAddress`

類型：字串

必要：是 (當使用 `extraHosts` 時)

要在 `/etc/hosts` 項目中使用的 IP 地址。

儲存與記錄

`readonlyRootFilesystem`

類型：布林值

必要：否

此參數為 true 時，容器會取得根檔案系統的唯讀存取權。此參數在 docker create-container 命令`ReadonlyRootfs`中映射到 ，`--read-only`選項用於 docker 執行。

> **Note**
>
> Windows 容器不支援此參數。

預設值為 `false`。

```
"readonlyRootFilesystem": true|false
```

`mountPoints`

類型：物件陣列

必要：否

容器中資料磁碟區的掛載點。此參數會映射至建立容器 Docker API Volumes中的 ，以及 Docker 執行--volume的選項。

Windows 容器可在 $env:ProgramData 所在的相同磁碟上掛載整個目錄。Windows 容器無法在不同的磁碟機上掛載目錄，而且掛載點無法跨磁碟機使用。您必須指定掛載點，將 Amazon EBS 磁碟區直接連接到 Amazon ECS 任務。

sourceVolume

類型：字串

必要：是 (當使用 mountPoints 時)

要掛載的磁碟區名稱。

containerPath

類型：字串

必要：是 (當使用 mountPoints 時)

要掛載磁碟區的容器中的路徑。

readOnly

類型：布林值

必要：否

如果此數值為 true，容器擁有磁碟區的唯讀存取權。如果此值為 false，則容器可寫入磁碟區。預設值為 false。

對於在執行 Windows 作業系統的 EC2 執行個體上執行的任務，請將 值保留為預設值 false。

volumesFrom

類型：物件陣列

必要：否

要從其他容器掛載的資料磁碟區。此參數在 docker create-container 命令VolumesFrom中映射到 ，以及 docker 執行--volumes-from的選項。

sourceContainer

類型：字串

必要：是 (當使用 volumesFrom 時)

要從其中掛載磁碟區的容器名稱。

`readOnly`

類型：布林值

必要：否

如果此數值為 `true`，容器擁有磁碟區的唯讀存取權。如果此值為 `false`，則容器可寫入磁碟區。預設值為 `false`。

```
"volumesFrom": [
                {
                  "sourceContainer": "string",
                  "readOnly": true|false
                }
              ]
```

`logConfiguration`

類型：LogConfiguration 物件

必要：否

容器的日誌組態規格。

例如，使用日誌組態的任務定義，請參閱Amazon ECS 任務定義範例。

此參數會映射至 docker create-container 命令`LogConfig`中的 ，以及 docker 執行`--log-driver`的選項。根據預設，容器會和 Docker 常駐程式使用一樣的日誌記錄驅動程式。不過，容器可以在容器定義中使用此參數指定日誌驅動程式，和 Docker 常駐程式使用不同的日誌驅動程式。若要讓容器使用不同的日誌驅動程式，必須在容器執行個體上適當設定日誌系統 (或在不同的日誌伺服器上使用遠端記錄選項)。

指定容器的日誌組態時，請考量下列事項：

- Amazon ECS 支援 Docker 常駐程式可用的日誌驅動程式子集。未來的 Amazon ECS 容器代理程式版本可能會提供更多可用的其他日誌驅動程式。
- 在您的容器執行個體上，此參數需要 1.18 版或更新版本的 Docker Remote API。
- 對於使用 EC2 啟動類型的任務，在容器執行個體上執行的 Amazon ECS 容器代理程式，必須使用 `ECS_AVAILABLE_LOGGING_DRIVERS` 環境變數註冊在該執行個體上可用的日誌驅動程式，才能讓放置在該執行個體上的容器使用這些日誌組態選項。如需詳細資訊，請參閱Amazon ECS 容器代理程式組態。

- 對於使用 Fargate 啟動類型的任務，您必須在任務之外安裝任何其他軟體。例如，Fluentd 輸出彙整工具或要傳送 Gelf 日誌執行 Logstash 的遠端主機。

```
"logConfiguration": {
      "logDriver": "awslogs","fluentd","gelf","json-
file","journald","logentries","splunk","syslog","awsfirelens",
      "options": {"string": "string"
        ...},
 "secretOptions": [{
  "name": "string",
  "valueFrom": "string"
 }]
}
```

`logDriver`

類型：字串

有效值：`"awslogs","fluentd","gelf","json-file","journald","logentries","splunk","syslog","awsfirelens"`

必要：是 (當使用 `logConfiguration` 時)

容器要使用的日誌驅動程式。根據預設，先前列出的有效值是 Amazon ECS 容器代理程式可用來通訊的日誌驅動程式。

針對使用 Fargate 啟動類型的任務，支援的日誌驅動程式為 `awslogs`、`splunk` 和 `awsfirelens`。

針對使用 EC2 啟動類型的任務，支援的日誌驅動程式為 `awslogs`、`fluentd`、`gelf`、`json-file`、`journald`、`logentries`、`syslog`、`splunk` 和 `awsfirelens`。

如需有關如何在任務定義中使用 `awslogs` 日誌驅動程式將您的容器日誌傳送到 CloudWatch Logs 的詳細資訊，請參閱將 Amazon ECS 日誌傳送至 CloudWatch 。

如需使用 `awsfirelens` 日誌驅動程式的詳細資訊，請參閱自訂日誌路由。

> **Note**
>
> 如果您有未列出的自訂驅動程式，您可以分支 GitHub 上可取得的 Amazon ECS 容器代理程式專案，並自訂以搭配此驅動程式一起使用。我們鼓勵您為想要進行的變更提交提取請求。但是，我們目前不支援執行此軟體經修改的複本。

在您的容器執行個體上，此參數需要 1.18 版或更新版本的 Docker Remote API。

`options`

類型：字串到字串映射

必要：否

要傳送到日誌驅動程式的索引鍵/值映射組態選項。

您可以指定的選項取決於日誌驅動程式。當您使用awslogs路由器將日誌路由到 Amazon CloudWatch 時，您可以指定的一些選項包括：

`awslogs-create-group`

必要：否

指定您是否希望自動建立日誌群組。若未指定此選項，則預設為 `false`。

> **Note**
>
> 在您嘗試使用 `awslogs-create-group` 之前，您的 IAM 政策必須包含 `logs:CreateLogGroup` 許可。

`awslogs-region`

必要：是

指定 AWS 區域 `awslogs`日誌驅動程式要傳送 Docker 日誌的 。您可以選擇將所有的日誌從不同區域中的叢集傳送至 CloudWatch Logs 中的單一區域。如此一來，日誌全都顯示在單一位置中。或者，您可以依照區域分隔日誌，以獲得更高的精細程度。請確定指定的日誌群組存在於您使用此選項指定的區域。

`awslogs-group`

必要：是

請務必指定 `awslogs` 日誌驅動程式傳送其日誌串流的日誌群組。

`awslogs-stream-prefix`

必要：對 EC2 啟動類型為選用，對 Fargate 啟動類型為必要。

使用 awslogs-stream-prefix 選項將日誌串流與指定字首、容器名稱以及容器所屬 Amazon ECS 任務的 ID 建立關聯。如果您使用此選項指定前綴，則日誌串流會使用下列格式。

```
prefix-name/container-name/ecs-task-id
```

如果您未使用此選項指定字首，則該日誌串流會以容器執行個體上 Docker 常駐程式指派的容器 ID 命名。因為只使用 Docker 容器 ID (僅容器執行個體提供) 很難對日誌回溯追蹤到傳送該日誌的容器，所以建議您使用此選項指定前綴。

對於 Amazon ECS 服務，您可以使用服務名稱作為前綴。藉此這可讓您對日誌串流回溯追蹤到容器所屬的服務、傳送該日誌之容器的名稱，以及容器所屬任務的 ID。

您必須指定日誌的串流字首，才可在使用 Amazon ECS 主控台時，讓日誌顯示在 Log (日誌) 窗格中。

awslogs-datetime-format

必要：否

此選項會以 Python strftime 格式定義多行開始模式。日誌訊息由符合模式的一行以及不符合模式的任何後續行所組成。符合的行是日誌訊息之間的分隔符號。

使用此格式的一個使用案例範例是用於剖析輸出，例如堆疊傾印，在其他情形下這可能會記錄在多個項目中。正確的模式可允許將它擷取在單一項目中。

如需詳細資訊，請參閱 awslogs-datetime-format。

無法同時設定 awslogs-datetime-format 和 awslogs-multiline-pattern 選項。

> **Note**
>
> 多行記錄會執行常規運算式剖析並比對所有日誌訊息。這可能會對記錄效能造成負面影響。

awslogs-multiline-pattern

必要：否

此選項定義使用常規運算式的多行開始模式。日誌訊息由符合模式的一行以及不符合模式的任何後續行所組成。符合的行是日誌訊息之間的分隔符號。

如需詳細資訊，請參閱 awslogs-multiline-pattern。

如果同時設定 awslogs-datetime-format，會忽略此選項。

無法同時設定 awslogs-datetime-format 和 awslogs-multiline-pattern 選項。

> **Note**
>
> 多行記錄會執行常規運算式剖析並比對所有日誌訊息。這可能會對記錄效能造成負面影響。

mode

必要：否

有效值：non-blocking | blocking

此選項定義日誌訊息從容器傳遞至awslogs日誌驅動程式的交付模式。當來自容器的日誌流程中斷時，您選擇的交付模式會影響應用程式的可用性。

如果您使用 blocking 模式，且 CloudWatch 的日誌流程中斷，則會封鎖從容器程式碼寫入 stderr stdout和串流的呼叫。應用程式的日誌記錄執行緒將因此封鎖。這可能會導致應用程式沒有回應，並導致容器運作狀態檢查失敗。

如果您使用 non-blocking 模式，則容器的日誌將儲存在使用 max-buffer-size 選項設定的記憶體內中間緩衝區中。這可以防止當日誌無法傳送至 CloudWatch 時應用程式發生無回應的狀況。如果您想要確保服務的可用性且可以接受一些日誌遺失，我們建議您使用此模式。如需詳細資訊，請參閱awslogs容器日誌驅動程式中的使用非封鎖模式防止日誌遺失。

max-buffer-size

必要：否

預設值：1m

使用 non-blocking 模式時，max-buffer-size 日誌選項會控制用於中繼訊息儲存的緩衝區大小。請務必根據您的應用程式指定適當的緩衝區大小。當緩衝區填滿時，無法儲存進一步的日誌。無法儲存的日誌將會遺失。

若要使用日誌路由器路由splunk日誌，您需要指定 splunk-token和 splunk-url。

當您使用awsfirelens日誌路由器將日誌路由到 AWS 服務 或 AWS Partner Network 目的地以進行日誌儲存和分析時，您可以設定 log-driver-buffer-limit選項，在傳送至日誌路由器容器之前限制記憶體中緩衝的事件數量。由於高輸送量可能會導致 Docker 內部緩衝區的記憶體不足，因此這可協助解決潛在的日誌受損問題。如需詳細資訊，請參閱the section called "為高輸送量設定日誌"。

使用 awsfirelens 路由日誌時，您可以指定的其他選項取決於目的地。當您將日誌匯出至 Amazon Data Firehose 時，可以使用 指定 AWS 區域 ，region並使用 指定日誌串流的名稱delivery_stream。

當您將日誌匯出至 Amazon Kinesis Data Streams 時，您可以使用 指定 AWS 區域 region，並使用 指定資料串流名稱stream。

當您將日誌匯出至 Amazon OpenSearch Service 時，您可以指定選項，例如 Name、Host（不含通訊協定的 OpenSearch Service 端點）Port、、Index、Type、Aws_region、Aws_authSuppress_Type_Name和 tls。

當您將日誌匯出至 Amazon S3 時，您可以使用 bucket選項指定儲存貯體。您也可以指定 region、upload_timeout、 total_file_size和 use_put_object做為選項。

在您的容器執行個體上，此參數需要 1.19 版或更新版本的 Docker Remote API。

secretOptions

類型：物件陣列

必要：否

此物件代表要傳送至日誌組態的秘密。日誌組態中使用的秘密可能包括身分驗證權杖、憑證或加密金鑰。如需詳細資訊，請參閱將敏感資料傳遞至 Amazon ECS 容器。

name

類型：字串

必要：是

要設定為容器上環境變數的值。

valueFrom

類型：字串

必要：是

公開到容器日誌組態的秘密。

```
"logConfiguration": {
 "logDriver": "splunk",
 "options": {
  "splunk-url": "https://cloud.splunk.com:8080",
  "splunk-token": "...",
  "tag": "...",
  ...
 },
 "secretOptions": [{
  "name": "splunk-token",
  "valueFrom": "/ecs/logconfig/splunkcred"
 }]
}
```

`firelensConfiguration`

類型：FirelensConfiguration 物件

必要：否

容器的 FireLens 組態。這是用來指定及設定容器日誌的日誌路由器。如需詳細資訊，請參閱將 Amazon ECS 日誌傳送至 AWS 服務或 AWS Partner。

```
{
    "firelensConfiguration": {
        "type": "fluentd",
        "options": {
            "KeyName": ""
        }
    }
}
```

`options`

類型：字串到字串映射

必要：否

設定日誌路由器時要使用的索引鍵/值映射選項。此欄位是選用欄位，可用於指定自訂組態檔案，或者將額外的中繼資料 (例如任務、任務定義、叢集和容器執行個體詳細資訊)

新增到日誌事件。如果已指定，則要使用的語法為 "options":{"enable-ecs-log-metadata":"true|false","config-file-type:"s3|file","config-file-value":"arn:aws:s3:::*amzn-s3-demo-bucket*/fluent.conf|filepath"}。如需詳細資訊，請參閱Amazon ECS 任務定義範例：將日誌路由至 FireLens。

type

類型：字串

必要：是

要使用的日誌路由器。有效值為 fluentd 或 fluentbit。

安全

如需容器安全性的詳細資訊，請參閱 Amazon ECS 任務和容器安全性最佳實務。

credentialSpecs

類型：字串陣列

必要：否

SSM 或 Amazon S3 中的 ARN 清單，用於為 Active Directory 身分驗證設定容器的憑證規格 (CredSpec) 檔案。建議您使用此參數而不是 dockerSecurityOptions. ARN 的上限為 1。

每個 ARN 有兩種格式。

credentialspecdomainless：MyARN

您使用 credentialspecdomainless:MyARN 為 Secrets Manager 中的秘密提供具其他區段的 CredSpec。您在秘密中提供網域的登入憑證。

在任何容器執行個體上執行的每項任務都可加入不同的網域。

您可以使用此格式而無需將容器執行個體加入網域。

credentialspec：MyARN

您使用 credentialspec:MyARN 為單一網域提供 CredSpec。

您必須先將容器執行個體加入網域，才能啟動此任務定義的任何任務。

在這兩種格式中，以 SSM 或 Amazon S3 中的 ARN 替換 MyARN。

credspec 必須在 Secrets Manager 中提供 ARN，以取得包含使用者名稱、密碼和要連線之網域的秘密。為了提高安全性，執行個體不會加入網域以進行無網域驗證。執行個體上的其他應用程式無法使用無網域憑證。您可以使用此參數在相同的執行個體上執行任務，即使是任務需要加入不同的網域。如需詳細資訊，請參閱在使用 Windows 容器的 gMSA 和使用 Linux 容器的 gMSA。

privileged

類型：布林值

必要：否

此參數為 true 時，容器便會取得主機容器執行個體的更高權限 (類似 root 使用者)。建議不要使用 privileged 執行容器。在大多數情況下，可以使用特定參數，而不是使用 privileged 來指定所需的確切權限。

此參數會映射至 docker create-container 命令Privileged中的 ，以及 docker 執行--privileged的選項。

> **Note**
>
> Windows 容器或使用 Fargate 啟動類型的任務不支援此參數。

預設值為 false。

```
"privileged": true|false
```

user

類型：字串

必要：否

要在容器內使用的使用者。此參數會映射到 docker create-container 命令User中的 ，以及 docker 執行--user的選項。

> **Important**
>
> 執行使用 host 網路模式的任務時，不要使用根使用者 (UID 0) 執行容器。作為最佳安全實務，請一律使用非根使用者。

您可以使用下列格式指定 `user`。如果指定 UID 或 GID，您必須使用正整數指定它。

- `user`
- `user:group`
- `uid`
- `uid:gid`
- `user:gid`
- `uid:group`

> **Note**
>
> Windows 容器不支援此參數。

```
"user": "string"
```

`dockerSecurityOptions`

類型：字串陣列

有效值：「沒有新的權限」|「AppArmor：描述檔」|「標籤：#」|「憑證規格：######」

必要：否

提供多個安全系統自訂組態的字串清單。此欄位對使用 Fargate 啟動類型之任務中的容器無效。

對於 EC2 上的 Linux 任務，此參數可用於參考 SELinux 和 AppArmor 多層級安全系統的自訂標籤。

對於 EC2 上的任何任務，此參數可在設定 Active Directory 身分驗證的容器時，用於參考憑證規格檔案。如需詳細資訊，請參閱 了解如何使用 gMSAs EC2 Windows 容器 for Amazon ECS 和 在 Amazon ECS 上使用 gMSA for EC2 Linux容器 EC2 。

此參數會映射至 docker create-container 命令`SecurityOpt`中的 ，以及 docker 執行`--security-opt`的選項。

```
"dockerSecurityOptions": ["string", ...]
```

> **Note**
>
> 在容器執行個體上執行的 Amazon ECS 容器代理程式必須先使用 ECS_SELINUX_CAPABLE=true 或 ECS_APPARMOR_CAPABLE=true 環境變數註冊，才能讓放置在該執行個體上的容器使用這些安全選項。如需詳細資訊，請參閱Amazon ECS 容器代理程式組態。

資源限制

`ulimits`

類型：物件陣列

必要：否

要為容器定義的 ulimit 值清單。此值覆寫作業系統的預設資源配額設定。此參數會映射到 docker create-container 命令Ulimits中的 ，以及 docker 執行--ulimit的選項。

在 Fargate 上託管的 Amazon ECS 任務會使用作業系統設定的預設資源限制值，但是 nofile 資源限制參數除外。nofile 資源限制會對容器可使用的開放檔案數量設限。在 Fargate 上，預設的 nofile 軟限制為 65535，硬限制為 65535。您可以將兩個限制的值設定為 1048576。如需詳細資訊，請參閱任務資源限制。

在您的容器執行個體上，此參數需要 1.18 版或更新版本的 Docker Remote API。

> **Note**
>
> Windows 容器不支援此參數。

```
"ulimits": [
      {
        "name":
 "core"|"cpu"|"data"|"fsize"|"locks"|"memlock"|"msgqueue"|"nice"|"nofile"|"nproc"|"rss"|"rtp
        "softLimit": integer,
        "hardLimit": integer
      }
      ...
    ]
```

name

類型：字串

有效值："core" | "cpu" | "data" | "fsize" | "locks" | "memlock" | "msgqueue" | "nice" | "nofile" | "nproc" | "rss" | "rtprio" | "rttime" | "sigpending" | "stack"

必要：是 (當使用 ulimits 時)

ulimit 的 type。

hardLimit

類型：整數

必要：是 (當使用 ulimits 時)

ulimit 類型的硬性限制。值可以以位元組、秒或計數形式指定，視 type的 而定ulimit。

softLimit

類型：整數

必要：是 (當使用 ulimits 時)

ulimit 類型的軟性限制。值可以以位元組、秒或計數形式指定，視 type的 而定ulimit。

Docker 標籤

dockerLabels

類型：字串到字串映射

必要：否

要新增到容器的標籤索引鍵/值映射。此參數會映射至 docker create-container 命令Labels中的 ，以及 docker 執行--label的選項。

在您的容器執行個體上，此參數需要 1.18 版或更新版本的 Docker Remote API。

```
"dockerLabels": {"string": "string"
      ...}
```

其他容器定義參數

在 Amazon ECS 主控台中使用 Configure via JSON (透過 JSON 進行設定) 選項註冊任務定義時，可以使用下列容器定義參數。如需詳細資訊，請參閱使用主控台建立 Amazon ECS 任務定義。

主題

Linux 參數

`linuxParameters`

類型：LinuxParameters 物件

必要：否

套用到容器的 Linux 特定選項，例如 KernelCapabilities。

> **Note**
>
> Windows 容器不支援此參數。

```
"linuxParameters": {
      "capabilities": {
        "add": ["string", ...],
        "drop": ["string", ...]
        }
      }
```

`capabilities`

類型：KernelCapabilities 物件

必要：否

從 Docker 提供的預設組態中新增或卸除的容器 Linux 功能。如需這些 Linux 功能的詳細資訊，請參閱 Linux 手冊頁面的功能 (7)。

add

類型：字串陣列

有效值："ALL" | "AUDIT_CONTROL" | "AUDIT_READ" | "AUDIT_WRITE" | "BLOCK_SUSPEND" | "CHOWN" | "DAC_OVERRIDE" | "DAC_READ_SEARCH" | "FOWNER" | "FSETID" | "IPC_LOCK" | "IPC_OWNER" | "KILL" | "LEASE" | "LINUX_IMMUTABLE" | "MAC_ADMIN" | "MAC_OVERRIDE" | "MKNOD" | "NET_ADMIN" | "NET_BIND_SERVICE" | "NET_BROADCAST" | "NET_RAW" | "SETFCAP" | "SETGID" | "SETPCAP" | "SETUID" | "SYS_ADMIN" | "SYS_BOOT" | "SYS_CHROOT" | "SYS_MODULE" | "SYS_NICE" | "SYS_PACCT" | "SYS_PTRACE" | "SYS_RAWIO" | "SYS_RESOURCE" | "SYS_TIME" | "SYS_TTY_CONFIG" | "SYSLOG" | "WAKE_ALARM"

必要：否

要新增至 Docker 提供之預設組態容器的 Linux 功能。此參數會映射到 docker create-container 命令CapAdd中的 ，以及 docker 執行--cap-add的選項。

> **Note**
>
> 在 Fargate 上啟動的任務僅支援新增 SYS_PTRACE 核心功能。

add

類型：字串陣列

有效值："SYS_PTRACE"

必要：否

要新增至 Docker 提供之預設組態容器的 Linux 功能。此參數會映射到 docker create-container 命令CapAdd中的 ，以及 docker 執行--cap-add的選項。

drop

類型：字串陣列

有效值："ALL" | "AUDIT_CONTROL" | "AUDIT_WRITE" | "BLOCK_SUSPEND" | "CHOWN" | "DAC_OVERRIDE" | "DAC_READ_SEARCH" | "FOWNER" | "FSETID" | "IPC_LOCK" | "IPC_OWNER" | "KILL" | "LEASE" | "LINUX_IMMUTABLE" | "MAC_ADMIN" | "MAC_OVERRIDE" | "MKNOD" | "NET_ADMIN" | "NET_BIND_SERVICE" | "NET_BROADCAST" | "NET_RAW" | "SETFCAP" | "SETGID" | "SETPCAP" | "SETUID" | "SYS_ADMIN" | "SYS_BOOT" | "SYS_CHROOT" | "SYS_MODULE" | "SYS_NICE" | "SYS_PACCT" | "SYS_PTRACE" | "SYS_RAWIO" | "SYS_RESOURCE" | "SYS_TIME" | "SYS_TTY_CONFIG" | "SYSLOG" | "WAKE_ALARM"

必要：否

要從 Docker 提供之預設組態中移除容器的 Linux 功能。此參數會映射到 docker create-container 命令`CapDrop`中的 ，以及 docker 執行`--cap-drop`的選項。

`devices`

任何要公開給容器的主機裝置。此參數會映射至 docker create-container 命令`Devices`中的 ，以及 docker 執行`--device`的選項。

> **Note**
>
> 當您使用 Fargate 啟動類型或 Windows 容器時，不支援 `devices` 參數。

類型：Device 物件的陣列

必要：否

`hostPath`

主機容器執行個體上裝置的路徑。

類型：字串

必要：是

`containerPath`

容器內的路徑，其為公開主機設備的目標路徑。

類型：字串

必要：否

`permissions`

要提供給裝置容器的明確許可。在預設情況下，容器在裝置上具有 `read`、`write` 及 `mknod` 許可。

類型：字串陣列

有效值: `read` | `write` | `mknod`

`initProcessEnabled`

在容器內執行 `init` 處理序，該處理序可轉寄訊號及獲得處理序。此參數會映射至 `--init`選項，以執行 Docker。

在您的容器執行個體上，此參數需要 1.25 版或更新版本的 Docker Remote API。

`maxSwap`

容器可以使用的交換記憶體總量 (以 MiB 為單位)。此參數會轉譯為 docker run `--memory-swap`選項，其中值為容器記憶體加`maxSwap`值的總和。

如果將 `maxSwap` 值指定為 `0`，容器不會使用交換。接受的值為 `0` 或任何正整數。如果省略 `maxSwap` 參數，容器使用其執行所在的容器執行個體的交換組態。必須設定 `maxSwap` 值，才能使用 `swappiness` 參數。

> **Note**
>
> 如果您使用的是使用 Fargate 啟動類型的任務，則不支援 `maxSwap` 參數。

`sharedMemorySize`

`/dev/shm` 磁碟區的大小值 (以 MiB 為單位)。此參數會映射至 `--shm-size`選項，以執行 Docker。

> **Note**
>
> 如果您使用的是使用 Fargate 啟動類型的任務，則不支援 `sharedMemorySize` 參數。

類型：整數

`swappiness`

您可使用此參數，調整容器的記憶體交換行為。`swappiness` 的值若為 `0` 將導致交換不會發生 (除非有需要)。`swappiness` 的值若為 `100` 將導致頻繁交換頁面。接受的值為介於 `0` 與 `100` 之間的整數。如果您未指定值，則會使用預設值 `60`。此外，如果您未對 `maxSwap` 指定值，則會忽略此參數。此參數會映射至 `--memory-swappiness`選項，以執行 Docker。

> **Note**
>
> 如果您使用的是使用 Fargate 啟動類型的任務，則不支援 `swappiness` 參數。
> 如果您在 Amazon Linux 2023 上使用任務，則不支援 `swappiness` 參數。

`tmpfs`

tmpfs 掛載的容器路徑、掛載選項和大小上限 (以 MiB 為單位)。此參數會映射至 `--tmpfs`選項，以執行 Docker。

> **Note**
>
> 如果您使用的是使用 Fargate 啟動類型的任務，則不支援 `tmpfs` 參數。

類型：Tmpfs 物件的陣列

必要：否

`containerPath`

要掛載 tmpfs 磁碟區的絕對檔案路徑。

類型：字串

必要：是

`mountOptions`

tmpfs 磁碟區掛載選項的清單。

類型：字串陣列

必要：否

有效值:"defaults" | "ro" | "rw" | "suid" | "nosuid" | "dev" | "nodev" | "exec" | "noexec" | "sync" | "async" | "dirsync" | "remount" | "mand" | "nomand" | "atime" | "noatime" | "diratime" | "nodiratime" | "bind" | "rbind" | "unbindable" | "runbindable" | "private" | "rprivate" | "shared" | "rshared" | "slave" | "rslave" | "relatime" | "norelatime" | "strictatime" | "nostrictatime" | "mode" | "uid" | "gid" | "nr_inodes" | "nr_blocks" | "mpol"

`size`

tmpfs 磁碟區大小上限 (以 MiB 為單位)。

類型：整數

必要：是

容器相依性

`dependsOn`

類型：ContainerDependency 物件陣列

必要：否

針對容器啟動和關閉而定義的相依性。容器可包含多個相依性。針對容器啟動而定義相依性時，它會保留給容器關閉。如需範例，請參閱「容器相依性」。

> **Note**
> 如果容器不符合相依性限制或在符合限制之前逾時，Amazon ECS 不會讓相依容器進入下一個狀態。

對於在 Amazon EC2 執行個體上託管的 Amazon ECS 任務，執行個體至少需要容器代理程式的 `1.26.0` 版本，才能啟用容器相依性。不過，我們建議您使用最新版的容器代理程式。如需檢查代理程式版本及更新至最新版本的資訊，請參閱「更新 Amazon ECS 容器代理程式」。如果您使用的是 Amazon ECS 最佳化 Amazon Linux AMI，您的執行個體至少需要 `1.26.0-1` 版的 `ecs-init` 套件。如果您的容器執行個體是從 `20190301` 版或更新版本啟動，它們會包含所需的容器代理程式和 `ecs-init` 版本。如需詳細資訊，請參閱Amazon ECS 最佳化 Linux AMIs。

對於在 Fargate 上託管的 Amazon ECS 任務，此參數要求任務或服務使用平台版本 `1.3.0` 或更新版本 (Linux) 或 `1.0.0` (Windows)。

```
"dependsOn": [
    {
        "containerName": "string",
        "condition": "string"
    }
]
```

`containerName`

類型：字串

必要：是

必須符合指定條件的容器名稱。

`condition`

類型：字串

必要：是

容器的相依性條件。以下是可用的條件及其行為：

- START - 此條件會模擬連結和磁碟區目前的行為。該條件可驗證相依容器先啟動後，才允許其他容器啟動。
- COMPLETE - 此條件驗證相依容器執行到完成 (結束) 後，才允許其他容器啟動。這適合用於只是執行指令碼，然後就退出的非必要容器。無法在基本容器上設定此條件。
- SUCCESS - 此條件與 COMPLETE 相同，但還要求容器必須以 `zero` 狀態結束。無法在基本容器上設定此條件。
- HEALTHY - 此條件會在驗證相依容器傳遞了其容器運作狀態檢查後，才允許其他容器啟動。這會要求相依容器在任務定義中設定運作狀態檢查。僅在任務啟動時才會確認這個條件。

容器逾時

`startTimeout`

類型：整數

必要：否

範例值：120

解析容器的相依性時在放棄之前等待的持續時間 (以秒為單位)。

例如，您在任務定義中指定兩個容器，其中的 containerA 對達到 COMPLETE、SUCCESS 或 HEALTHY 狀態的 containerB 有相依性。如果為 containerB 指定 startTimeout 值，但在該時間內沒有達到所需狀態，則 containerA 不會啟動。

> **Note**
>
> 如果容器不符合相依性限制或在符合限制之前逾時，Amazon ECS 不會讓相依容器進入下一個狀態。

對於在 Fargate 上託管的 Amazon ECS 任務，此參數要求任務或服務使用平台版本 1.3.0 或更新版本 (Linux)。最高值為 120 秒。

stopTimeout

類型：整數

必要：否

範例值：120

當容器本身未正常結束時，強制終止容器之前的等待期間 (以秒為單位)。

對於在 Fargate 上託管的 Amazon ECS 任務，此參數要求任務或服務使用平台版本 1.3.0 或更新版本 (Linux)。如果未指定參數，則會使用預設值 30 秒。Fargate 的最大值為 120 秒。

對於使用 EC2 啟動類型的任務，如果未指定 stopTimeout 參數，會使用 Amazon ECS 容器代理程式組態變數 ECS_CONTAINER_STOP_TIMEOUT 設定的值。如果未設定 stopTimeout 參數或 ECS_CONTAINER_STOP_TIMEOUT 代理程式組態變數，則對於 Linux 容器，預設值為 30 秒，對於 Windows 容器會使用預設值 30 秒。容器執行個體至少需要 1.26.0 版的容器代理程式，才能啟用容器停止逾時值。不過，我們建議您使用最新版的容器代理程式。如需如何檢查代理程式版本及更新至最新版本的資訊，請參閱「更新 Amazon ECS 容器代理程式」。如果您使用的是 Amazon ECS 最佳化 Amazon Linux AMI，您的執行個體至少需要 1.26.0-1 版的 ecs-init 套件。如果您的容器執行個體是從 20190301 版或更新版本啟動，它們會包含所需的容器代理程式和 ecs-init 版本。如需詳細資訊，請參閱Amazon ECS 最佳化 Linux AMIs。

系統控制

`systemControls`

類型：SystemControl 物件

必要：否

要在容器中設定的命名空間核心參數清單。此參數會映射至 docker create-container 命令`Sysctls`中的 ，以及 docker 執行`--sysctl`的選項。例如，您可以規劃 `net.ipv4.tcp_keepalive_time` 設定以維持壽命較長的連線。

不建議您指定單一任務中多個容器的網路相關 `systemControls` 參數，而單一任務也使用 `awsvpc` 或 `host` 網路模式。執行此操作的缺點如下：

- 對於使用包括 Fargate `awsvpc` 網路模式的任務，如果您設定任何容器的 `systemControls`，則會套用至任務中的所有容器。如果您針對單一任務中的多個容器設定不同的 `systemControls`，則最後啟動的容器會判斷哪些 `systemControls` 生效。
- 對於使用 `host` 網路模式的任務，不支援網路命名空間 `systemControls`。

如果您為任務中的容器設定要使用的 IPC 資源命名空間，以下各項條件套用到您的系統控制。如需詳細資訊，請參閱IPC 模式。

- 針對使用 `host` IPC 模式的任務，不支援 IPC 命名空間 `systemControls`。
- 針對使用 `task` IPC 模式的任務，IPC 命名空間 `systemControls` 值套用到任務內的所有容器。

> **Note**
>
> Windows 容器不支援此參數。

> **Note**
>
> 如果任務使用平台版本 `1.4.0` 或更高版本 (Linux)，只有託管於 AWS Fargate 的任務才支援此參數。Fargate 上的 Windows 容器不支援此參數。

```
"systemControls": [
```

```
    {
         "namespace":"string",
         "value":"string"
    }
]
```

namespace

類型：字串

必要：否

要設定 value的命名空間核心參數。

有效的 IPC 命名空間值："kernel.msgmax" | "kernel.msgmnb" | "kernel.msgmni" | "kernel.sem" | "kernel.shmall" | "kernel.shmmax" | "kernel.shmmni" | "kernel.shm_rmid_forced"，以及開頭為 "fs.mqueue.*" 的 Sysctls

有效的網路命名空間值：以 Sysctls開頭"net.*"。在 Fargate 上，只接受容器內存在Sysctls的命名空間。

Fargate 支援所有這些值。

value

類型：字串

必要：否

中指定的命名空間核心參數值namespace。

互動性

interactive

類型：布林值

必要：否

此參數為 true 時，您可部署需要配置 stdin 或 tty 的容器化應用程式。此參數會映射至 docker create-container 命令0penStdin中的 ，以及 docker 執行--interactive的選項。

預設值為 false。

虛擬終端機

pseudoTerminal

類型：布林值

必要：否

當此參數為 true 時，會配置 TTY。此參數會映射至 docker create-container 命令Tty中的 ，以及 docker 執行--tty的選項。

預設值為 false。

Elastic Inference 加速器名稱

任務定義的 Elastic Inference 加速器資源要求。

> **Note**
>
> Amazon Elastic Inference (EI) 不再提供給客戶。

任務定義允許使用以下參數：

deviceName

類型：字串

必要：是

彈性推論加速器裝置名稱。deviceName 也必須在容器定義中參照，請參閱 Elastic Inference accelerator。

deviceType

類型：字串

必要：是

要使用的彈性推論加速器。

任務置放限制條件

當您註冊任務定義時，您可以提供任務置放限制，自訂 Amazon ECS 放置任務的方式。

若您使用的是 Fargate 啟動類型，則不支援任務置放限制條件。根據預設，Fargate 任務會分散在可用區域中。

對於使用 EC2 啟動類型的任務，您可以使用限制，根據可用區域、執行個體類型或自訂屬性來放置任務。如需詳細資訊，請參閱定義 Amazon ECS 用於任務的容器執行個體。

容器定義允許使用以下參數：

`expression`

類型：字串

必要：否

限制所套用的叢集查詢語言運算式。如需詳細資訊，請參閱建立表達式以定義 Amazon ECS 任務的容器執行個體。

`type`

類型：字串

必要：是

限制類型。使用 `memberOf`，以將選擇限制於特定群組或有效的待選項目。

代理組態

`proxyConfiguration`

類型：ProxyConfiguration 物件

必要：否

App Mesh 代理的組態詳細資訊。

對於使用 EC2 啟動類型的任務，容器執行個體需要至少 1.26.0 版的容器代理程式和至少 1.26.0-1 版的 `ecs-init` 套裝服務，才能啟用代理組態。如果您的容器執行個體是從 Amazon ECS 最佳化 AMI `20190301` 版或更新版本啟動，則它們會包含所需的容器代理程式和 `ecs-init` 版本。如需詳細資訊，請參閱Amazon ECS 最佳化 Linux AMIs。

對於使用 Fargate 啟動類型的任務，此功能要求任務或服務必須使用平台版本 1.3.0 或更新版本。

> **Note**
>
> Windows 容器不支援此參數。

```
"proxyConfiguration": {
    "type": "APPMESH",
    "containerName": "string",
    "properties": [
        {
           "name": "string",
           "value": "string"
        }
    ]
}
```

`type`

類型：字串

有效值：APPMESH

必要：否

代理類型。唯一支援的值為 APPMESH。

`containerName`

類型：字串

必要：是

做為 App Mesh 代理的容器名稱。

`properties`

類型：KeyValuePair 物件陣列

必要：否

一組網路組態參數，用於提供指定為鍵值組的容器網路介面 (CNI) 外掛程式。

- IgnoredUID - (必要) user 參數在容器定義中定義之代理容器的使用者 ID (UID)。這是用於確保代理忽略自己的流量。如果指定了 IgnoredGID，這個欄位可以為空白。
- IgnoredGID - (必要) user 參數在容器定義中定義之代理容器的群組 ID (GID)。這是用於確保代理忽略自己的流量。如果指定了 IgnoredUID，這個欄位可以為空白。
- AppPorts - (必要) 應用程式所用連接埠的清單。這些連接埠的網路流量會轉發到 ProxyIngressPort 和 ProxyEgressPort。
- ProxyIngressPort - (必要) 指定傳至 AppPorts 的傳入流量所導向的連接埠。
- ProxyEgressPort - (必要) 指定來自 AppPorts 的傳出流量所導向的連接埠。
- EgressIgnoredPorts - (必要) 進入這些指定連接埠的傳出流量將忽略，不會重新引導至 ProxyEgressPort。它可以是空的清單。
- EgressIgnoredIPs - (必要) 進入這些指定 IP 位址的傳出流量將忽略，不會重新引導至 ProxyEgressPort。它可以是空的清單。

name

類型：字串

必要：否

鍵/值對的名稱。

value

類型：字串

必要：否

索引鍵/值對的值。

磁碟區

當您註冊任務定義時，您可以選擇指定要傳遞給容器執行個體上的Docker協助程式的磁碟區清單，然後，該清單可供相同容器執行個體上的其他容器存取。

以下為會用到的資料磁碟區類型：

- Amazon EBS 磁碟區：為資料密集型容器化工作負載提供經濟實惠、耐用、高效能的區塊儲存。您可以在執行獨立任務或建立或更新服務時，為每個 Amazon ECS 任務連接 1 個 Amazon EBS 磁碟區。在 Fargate 或 Amazon EC2 執行個體上託管的 Linux 任務支援 Amazon EBS 磁碟區。如需詳細資訊，請參閱搭配 Amazon ECS 使用 Amazon EBS 磁碟區。

- Amazon EFS 磁碟區 — 提供簡單、可擴展且持久的檔案儲存，可與 Amazon ECS 任務搭配使用。利用 Amazon EFS，儲存容量即可有彈性。儲存容量會隨著您新增和移除檔案時自動擴展和縮減。您的應用程式可以擁有他們所需的儲存體，以及他們何時需要它。在 Fargate 或 Amazon EC2 執行個體上託管的任務支援 Amazon EFS 磁碟區。如需詳細資訊，請參閱搭配 Amazon ECS 使用 Amazon EFS 磁碟區。
- FSx for Windows File Server 磁碟區 — 提供全受管 Microsoft Windows 檔案伺服器。這些檔案伺服器由 Windows 檔案系統支援。搭配使用 FSx for Windows File Server 與 Amazon ECS 時，您可以使用持續、分散、共用、靜態的檔案儲存來佈建 Windows 任務。如需詳細資訊，請參閱搭配 Amazon ECS 使用 FSx for Windows File Server 磁碟區。

  Fargate 上的 Windows 容器不支援此選項。
- Docker 磁碟區 – 在主機 Amazon EC2 執行個體`/var/lib/docker/volumes`上於 下建立的 Docker 受管磁碟區。Docker 磁碟區驅動程式 (也稱為外掛程式) 用外部儲存系統 (例如 Amazon EBS) 來整合磁碟區。可使用內建 `local` 磁碟區驅動程式或第三方磁碟區驅動程式。只有在 Amazon EC2 執行個體上執行任務時，才支援 Docker 磁碟區。Windows 容器僅支援使用`local`驅動程式。若要使用 Docker 磁碟區，請在您的任務定義中指定 `dockerVolumeConfiguration`。
- 繫結掛載 – 主機機器上掛載到容器中的檔案或目錄。在 AWS Fargate 或 Amazon EC2 執行個體上執行任務時，支援繫結掛載主機磁碟區。若要使用綁定掛載主機磁碟區，請在任務定義中指定 `host` 和選用的 `sourcePath` 值。

如需詳細資訊，請參閱Amazon ECS 任務的儲存選項。

容器定義允許使用以下參數。

`name`

類型：字串

必要：否

磁碟區名稱。最多允許 255 個字母 （ 大寫和小寫 ）、數字、連字號 (-) 和底線 (_)。此名稱會在容器定義`mountPoints`物件的 `sourceVolume` 參數中參考。

`host`

必要：否

`host` 參數用於將綁定掛載的生命週期綁定到主機 Amazon EC2 執行個體，而非任務，以及它的儲存位置。如果 `host` 參數是空的，則 Docker 常駐程式會為您的資料磁碟區指派主機路徑，但其相關聯的容器停止執行後，不保證會保留資料。

Windows 容器可在 $env:ProgramData 所在的相同磁碟上掛載整個目錄。

> **Note**
>
> 只有在使用託管在 Amazon EC2 執行個體上的任務時，才支援 sourcePath 參數。

sourcePath

類型：字串

必要：否

使用 host 參數時，指定 sourcePath 以宣告在主機 Amazon EC2 執行個體上提供給容器的路徑。如果此參數是空的，則 Docker 常駐程式會為您指派主機路徑。如果 host 參數包含 sourcePath 檔案位置，資料磁碟區將保留在主機 Amazon EC2 執行個體上的指定位置，直到您手動將其刪除為止。如果 sourcePath 值不存在於主機 Amazon EC2 執行個體，Docker 常駐程式將建立該值。如果位置存在，將匯出來源路徑資料夾的內容。

configuredAtLaunch

類型：布林值

必要：否

指定磁碟區是否可在啟動時設定。設定為 時true，您可以在執行獨立任務時，或在建立或更新服務時設定磁碟區。設為 時true，您將無法在任務定義中提供另一個磁碟區組態。此參數必須設定為 true，以設定要連接至任務的 Amazon EBS 磁碟區。configuredAtLaunch 將磁碟區組態設定為啟動階段true並延遲啟動階段，可讓您建立不受磁碟區類型或特定磁碟區設定限制的任務定義。這樣做可讓任務定義在不同執行環境中重複使用。如需詳細資訊，請參閱 Amazon EBS 磁碟區。

dockerVolumeConfiguration

類型：DockerVolumeConfiguration 物件

必要：否

此參數只有使用 Docker 磁碟區時才會指定。只有在 EC2 執行個體上執行任務時，才支援 Docker 磁碟區。Windows 容器僅支援使用local驅動程式。若要使用綁定掛載，請指定 host。

scope

類型：字串

有效值: task | shared

必要：否

決定生命週期的 Docker 磁碟區範圍。範圍受限於 task 的 Docker 磁碟區，會在任務啟動時自動佈建，以及在任務停止時銷毀。範圍為 shared 的 Docker 磁碟區會在任務停止之後保留。

autoprovision

類型：布林值

預設值：false

必要：否

若此數值為 true，Docker 磁碟區便得以建立 (若它尚不存在)。只有在 scope為 時，才會使用此欄位shared。如果 scope是 task，則必須省略此參數。

driver

類型：字串

必要：否

要使用的 Docker 磁碟區驅動程式。驅動程式值必須符合 Docker 提供的驅動程式名稱，因為此名稱用於任務置放。如果驅動程式是使用 Docker 外掛程式 CLI 安裝，請使用 從您的容器執行個體docker plugin ls擷取驅動程式名稱。如果驅動程式是透過使用其他方法安裝，請使用 Docker 外掛程式探索來擷取驅動程式名稱。

driverOpts

類型：字串

必要：否

要傳遞的 Docker 驅動程式特定選項映射。此參數會在 Docker 的建立磁碟區區段DriverOpts中映射至 。

labels

類型：字串

必要：否

自訂中繼資料，新增到您的 Docker 磁碟區。

`efsVolumeConfiguration`

類型：EFSVolumeConfiguration 物件

必要：否

只有使用 Amazon EFS 磁碟區時才會指定此參數。

`fileSystemId`

類型：字串

必要：是

要使用的 Amazon EFS 檔案系統識別碼。

`rootDirectory`

類型：字串

必要：否

在 Amazon EFS 檔案系統中的目錄，其將掛載作為主機內的根目錄。如果省略此參數，將使用 Amazon EFS 磁碟區的根目錄。指定 `/` 的效果與忽略此參數的效果相同。

> **Important**
>
> 如果在 中指定了 EFS 存取點`authorizationConfig`，則必須省略根目錄參數或將其設定為 `/`，這將強制執行 EFS 存取點上設定的路徑。

`transitEncryption`

類型：字串

有效值：`ENABLED` | `DISABLED`

必要：否

指定是否要對在 Amazon ECS 主機和 Amazon EFS 伺服器之間傳輸中的 Amazon EFS 資料啟用加密功能。如果使用 Amazon EFS IAM 授權，則必須啟用傳輸加密。如果省略此參數，系統會使用 `DISABLED` 的預設值。如需詳細資訊，請參閱《Amazon Elastic File System 使用者指南》中的加密傳輸中的資料。

`transitEncryptionPort`

類型：整數

必要：否

在 Amazon ECS 主機和 Amazon EFS 伺服器之間傳送加密資料時所使用的連接埠。如果您未指定傳輸加密連接埠，任務將使用 Amazon EFS 掛載協助程式使用的連接埠選取策略。如需詳細資訊，請參閱《Amazon Elastic File System 使用者指南》中的 EFS 掛載協助程式。

`authorizationConfig`

類型：EFSAuthorizationConfig 物件

必要：否

Amazon EFS 檔案系統的授權組態詳細資訊。

`accessPointId`

類型：字串

必要：否

要使用的存取點 ID。如果指定存取點，`efsVolumeConfiguration`則必須省略 中的根目錄值，或將 設定為 `/`，這會強制執行 EFS 存取點上設定的路徑。如果使用存取點，則必須在 `EFSVolumeConfiguration` 中啟用傳輸加密。如需詳細資訊，請參閱《Amazon Elastic File System 使用者指南》中的使用 Amazon EFS 存取點。

`iam`

類型：字串

有效值：`ENABLED` | `DISABLED`

必要：否

指定是否要在掛載 Amazon EFS 檔案系統時使用任務定義中定義的 Amazon ECS 任務 IAM 角色。如果已啟用，必須在 `EFSVolumeConfiguration` 中啟用傳輸加密。如果省略此參數，系統會使用 `DISABLED` 的預設值。如需詳細資訊，請參閱任務的 IAM 角色。

`FSxWindowsFileServerVolumeConfiguration`

類型：FSxWindowsFileServerVolumeConfiguration 物件

必要：是

當您使用 Amazon FSx for Windows File Server 檔案系統儲存任務時，會指定此參數。

`fileSystemId`

類型：字串

必要：是

要使用的 FSx for Windows File Server 檔案系統 ID。

`rootDirectory`

類型：字串

必要：是

FSx for Windows File Server 檔案系統中的目錄，其將掛載做為主機內的根目錄。

`authorizationConfig`

`credentialsParameter`

類型：字串

必要：是

授權憑證選項。

選項：

- AWS Secrets Manager 秘密的 Amazon Resource Name (ARN)。
- AWS Systems Manager 參數的 ARN。

`domain`

類型：字串

必要：是

由 AWS Directory Service for Microsoft Active Directory(AWS Managed Microsoft AD) 目錄或自我託管 EC2 Active Directory 託管的完整網域名稱。

標籤

當您註冊任務定義時，您可以選擇性地指定套用至任務定義的中繼資料標籤。標籤可協助您分類和組織您的任務定義。每個標籤皆包含索引鍵與選用值。您可以兩個都定義。如需詳細資訊，請參閱標記 Amazon ECS 資源。

> **Important**
>
> 請勿在標籤中加入個人識別資訊或其他機密或敏感資訊。許多 AWS 服務都可以存取標籤，包括帳單。標籤不適用於私有或敏感資料。

標籤物件允許使用以下參數。

`key`

類型：字串

必要：否

組成標籤的鍵值組的一部分。索引鍵是一般標籤，作用就像更特定標籤值的類別。

`value`

類型：字串

必要：否

組成標籤的鍵值組的選用部分。值就像標籤類別 (索引鍵) 內的描述項。

其他任務定義參數

在 Amazon ECS 主控台中使用 Configure via JSON (透過 JSON 進行設定) 選項註冊任務定義時，可以使用下列任務定義參數。如需詳細資訊，請參閱使用主控台建立 Amazon ECS 任務定義。

主題

- 暫時性儲存
- IPC 模式
- PID 模式
- 錯誤注入

暫時性儲存

`ephemeralStorage`

類型：EphemeralStorage 物件

必要：否

為任務配置的暫時性儲存量 (以 GB 為單位)。對於託管於 AWS Fargate的任務，此參數可用來擴充可用的暫時性儲存總量 (超過預設數量)。如需詳細資訊，請參閱the section called “綁定掛載”。

> **Note**
>
> 只有託管於 AWS Fargate 且使用平台版本 `1.4.0` 或更新版本 (Linux) 或 `1.0.0` 或更新版本 (Windows) 的任務才支援此參數。

IPC 模式

`ipcMode`

類型：字串

必要：否

要用於任務中容器的 IPC 資源命名空間。有效值為 `host`、`task` 或 `none`。如果已指定 `host`，則任務內對相同容器執行個體指定 `host` IPC 模式的所有容器，會與主機 Amazon EC2 執行個體共用相同的 IPC 資源。如果已指定 `task`，則指定任務內的所有容器會共用相同的 IPC 資源。如果已指定 `none`，則任務內的容器內的 IPC 資源為私有，並且不會與任務中或容器執行個體上的其他容器共用。如果沒有指定值，則 IPC 資源命名空間共用取決於容器執行個體上 Docker 常駐程式的設定。

如果是使用 `host` IPC 模式，這會提高將不需要的 IPC 命名空間公開的風險。

如果是使用 `systemControls` 來為任務中的容器設定命名空間核心參數，以下各項會套用到您的 IPC 資源命名空間。

- 針對使用 `host` IPC 模式的任務，不支援與 `systemControls` 相關的 IPC 命名空間。
- 針對使用 `task` IPC 模式的任務，IPC 命名空間相關的 `systemControls` 套用到任務內的所有容器。

> **Note**
>
> Windows 容器或使用 Fargate 啟動類型的任務不支援此參數。

PID 模式

`pidMode`

類型：字串

有效值: `host` | `task`

必要：否

要用於任務中容器的程序命名空間。有效值為 `host` 或 `task`。在 Linux 容器的 Fargate 上，唯一有效的值為 `task`。例如，監控附屬可能需要 `pidMode` 存取相同任務中執行之其他容器的相關資訊。

如果已指定 `host`，任務內對相容器執行個體指定 `host` PID 模式的所有容器，會與主機 Amazon EC2 執行個體共用相同的程序命名空間。

如果已指定 `task`，則指定任務內的所有容器會共用相同的程序命名空間。

如果未指定任何值，每個容器的預設值會是私有命名空間。

如果是使用 `host` PID 模式，這會提高將不需要的程序命名空間公開的風險。

> **Note**
>
> Windows 容器不支援此參數。

> **Note**
>
> 如果任務使用平台版本 `1.4.0` 或更高版本 (Linux)，只有託管於 AWS Fargate 的任務才支援此參數。Fargate 上的 Windows 容器不支援此參數。

錯誤注入

`enableFaultInjection`

類型：布林值

有效值: `true` | `false`

必要：否

如果此參數設定為 `true`，在任務的承載中，Amazon ECS 和 Fargate 接受來自任務容器的錯誤注入請求。根據預設，此參數會設定為 `false`。

Amazon ECS 任務定義範本

空白的任務定義範本如下所示。您可以使用此範本來建立任務定義，然後可以貼到主控台 JSON 輸入區域，或儲存到 檔案並搭配 AWS CLI `--cli-input-json`選項使用。如需詳細資訊，請參閱Amazon ECS 任務定義參數。

Amazon EC2 啟動類型範本

```
{
  "family": "",
  "taskRoleArn": "",
  "executionRoleArn": "",
  "networkMode": "none",
  "containerDefinitions": [
    {
      "name": "",
      "image": "",
      "repositoryCredentials": {
        "credentialsParameter": ""
      },
      "cpu": 0,
      "memory": 0,
      "memoryReservation": 0,
      "links": [""],
      "portMappings": [
        {
          "containerPort": 0,
          "hostPort": 0,
          "protocol": "tcp"
        }
      ],
      "restartPolicy": {
        "enabled": true,
        "ignoredExitCodes": [0],
        "restartAttemptPeriod": 180
      },
      "essential": true,
```

```
      "entryPoint": [""],
      "command": [""],
      "environment": [
        {
          "name": "",
          "value": ""
        }
      ],
      "environmentFiles": [
        {
          "value": "",
          "type": "s3"
        }
      ],
      "mountPoints": [
        {
          "sourceVolume": "",
          "containerPath": "",
          "readOnly": true
        }
      ],
      "volumesFrom": [
        {
          "sourceContainer": "",
          "readOnly": true
        }
      ],
      "linuxParameters": {
        "capabilities": {
          "add": [""],
          "drop": [""]
        },
        "devices": [
          {
            "hostPath": "",
            "containerPath": "",
            "permissions": ["read"]
          }
        ],
        "initProcessEnabled": true,
        "sharedMemorySize": 0,
        "tmpfs": [
          {
            "containerPath": "",
```

```
            "size": 0,
            "mountOptions": [""]
          }
        ],
        "maxSwap": 0,
        "swappiness": 0
      },
      "secrets": [
        {
          "name": "",
          "valueFrom": ""
        }
      ],
      "dependsOn": [
        {
          "containerName": "",
          "condition": "COMPLETE"
        }
      ],
      "startTimeout": 0,
      "stopTimeout": 0,
      "hostname": "",
      "user": "",
      "workingDirectory": "",
      "disableNetworking": true,
      "privileged": true,
      "readonlyRootFilesystem": true,
      "dnsServers": [""],
      "dnsSearchDomains": [""],
      "extraHosts": [
        {
          "hostname": "",
          "ipAddress": ""
        }
      ],
      "dockerSecurityOptions": [""],
      "interactive": true,
      "pseudoTerminal": true,
      "dockerLabels": {
        "KeyName": ""
      },
      "ulimits": [
        {
          "name": "nofile",
```

```
          "softLimit": 0,
          "hardLimit": 0
        }
      ],
      "logConfiguration": {
        "logDriver": "splunk",
        "options": {
          "KeyName": ""
        },
        "secretOptions": [
          {
            "name": "",
            "valueFrom": ""
          }
        ]
      },
      "healthCheck": {
        "command": [""],
        "interval": 0,
        "timeout": 0,
        "retries": 0,
        "startPeriod": 0
      },
      "systemControls": [
        {
          "namespace": "",
          "value": ""
        }
      ],
      "resourceRequirements": [
        {
          "value": "",
          "type": "InferenceAccelerator"
        }
      ],
      "firelensConfiguration": {
        "type": "fluentbit",
        "options": {
          "KeyName": ""
        }
      }
    }
  ],
  "volumes": [
```

```
    {
      "name": "",
      "host": {
        "sourcePath": ""
      },
      "configuredAtLaunch": true,
      "dockerVolumeConfiguration": {
        "scope": "shared",
        "autoprovision": true,
        "driver": "",
        "driverOpts": {
          "KeyName": ""
        },
        "labels": {
          "KeyName": ""
        }
      },
      "efsVolumeConfiguration": {
        "fileSystemId": "",
        "rootDirectory": "",
        "transitEncryption": "DISABLED",
        "transitEncryptionPort": 0,
        "authorizationConfig": {
          "accessPointId": "",
          "iam": "ENABLED"
        }
      },
      "fsxWindowsFileServerVolumeConfiguration": {
        "fileSystemId": "",
        "rootDirectory": "",
        "authorizationConfig": {
          "credentialsParameter": "",
          "domain": ""
        }
      }
    }
  ],
  "placementConstraints": [
    {
      "type": "memberOf",
      "expression": ""
    }
  ],
  "requiresCompatibilities": ["EC2"],
```

```
  "cpu": "",
  "memory": "",
  "tags": [
    {
      "key": "",
      "value": ""
    }
  ],
  "pidMode": "task",
  "ipcMode": "task",
  "proxyConfiguration": {
    "type": "APPMESH",
    "containerName": "",
    "properties": [
      {
        "name": "",
        "value": ""
      }
    ]
  },
  "inferenceAccelerators": [
    {
      "deviceName": "",
      "deviceType": ""
    }
  ],
  "ephemeralStorage": {
    "sizeInGiB": 0
  },
  "runtimePlatform": {
    "cpuArchitecture": "X86_64",
    "operatingSystemFamily": "WINDOWS_SERVER_20H2_CORE"
  }
}
```

Fargate 啟動類型範本

Important

對於 Fargate 啟動類型，您必須將 `operatingSystemFamily` 參數包含下列其中一個值：

- `LINUX`

- WINDOWS_SERVER_2019_FULL
- WINDOWS_SERVER_2019_CORE
- WINDOWS_SERVER_2022_FULL
- WINDOWS_SERVER_2022_CORE

```
{
    "family": "",
    "runtimePlatform": {"operatingSystemFamily": ""},
    "taskRoleArn": "",
    "executionRoleArn": "",
    "networkMode": "awsvpc",
    "platformFamily": "",
    "containerDefinitions": [
        {
            "name": "",
            "image": "",
            "repositoryCredentials": {"credentialsParameter": ""},
            "cpu": 0,
            "memory": 0,
            "memoryReservation": 0,
            "links": [""],
            "portMappings": [
                {
                    "containerPort": 0,
                    "hostPort": 0,
                    "protocol": "tcp"
                }
            ],
            "essential": true,
            "entryPoint": [""],
            "command": [""],
            "environment": [
                {
                    "name": "",
                    "value": ""
                }
            ],
            "environmentFiles": [
                {
```

```
                "value": "",
                "type": "s3"
            }
        ],
        "mountPoints": [
            {
                "sourceVolume": "",
                "containerPath": "",
                "readOnly": true
            }
        ],
        "volumesFrom": [
            {
                "sourceContainer": "",
                "readOnly": true
            }
        ],
        "linuxParameters": {
            "capabilities": {
                "add": [""],
                "drop": [""]
            },
            "devices": [
                {
                    "hostPath": "",
                    "containerPath": "",
                    "permissions": ["read"]
                }
            ],
            "initProcessEnabled": true,
            "sharedMemorySize": 0,
            "tmpfs": [
                {
                    "containerPath": "",
                    "size": 0,
                    "mountOptions": [""]
                }
            ],
            "maxSwap": 0,
            "swappiness": 0
        },
        "secrets": [
            {
                "name": "",
```

```
                "valueFrom": ""
            }
        ],
        "dependsOn": [
            {
                "containerName": "",
                "condition": "HEALTHY"
            }
        ],
        "startTimeout": 0,
        "stopTimeout": 0,
        "hostname": "",
        "user": "",
        "workingDirectory": "",
        "disableNetworking": true,
        "privileged": true,
        "readonlyRootFilesystem": true,
        "dnsServers": [""],
        "dnsSearchDomains": [""],
        "extraHosts": [
            {
                "hostname": "",
                "ipAddress": ""
            }
        ],
        "dockerSecurityOptions": [""],
        "interactive": true,
        "pseudoTerminal": true,
        "dockerLabels": {"KeyName": ""},
        "ulimits": [
            {
                "name": "msgqueue",
                "softLimit": 0,
                "hardLimit": 0
            }
        ],
        "logConfiguration": {
            "logDriver": "awslogs",
            "options": {"KeyName": ""},
            "secretOptions": [
                {
                    "name": "",
                    "valueFrom": ""
                }
```

```
                ]
            },
            "healthCheck": {
                "command": [""],
                "interval": 0,
                "timeout": 0,
                "retries": 0,
                "startPeriod": 0
            },
            "systemControls": [
                {
                    "namespace": "",
                    "value": ""
                }
            ],
            "resourceRequirements": [
                {
                    "value": "",
                    "type": "GPU"
                }
            ],
            "firelensConfiguration": {
                "type": "fluentd",
                "options": {"KeyName": ""}
            }
        }
    ],
    "volumes": [
        {
            "name": "",
            "host": {"sourcePath": ""},
            "configuredAtLaunch":true,
            "dockerVolumeConfiguration": {
                "scope": "task",
                "autoprovision": true,
                "driver": "",
                "driverOpts": {"KeyName": ""},
                "labels": {"KeyName": ""}
            },
            "efsVolumeConfiguration": {
                "fileSystemId": "",
                "rootDirectory": "",
                "transitEncryption": "ENABLED",
                "transitEncryptionPort": 0,
```

```
                "authorizationConfig": {
                    "accessPointId": "",
                    "iam": "ENABLED"
                }
            }
        }
    ],
    "requiresCompatibilities": ["FARGATE"],
    "cpu": "",
    "memory": "",
    "tags": [
        {
            "key": "",
            "value": ""
        }
    ],
    "ephemeralStorage": {"sizeInGiB": 0},
    "pidMode": "task",
    "ipcMode": "none",
    "proxyConfiguration": {
        "type": "APPMESH",
        "containerName": "",
        "properties": [
            {
                "name": "",
                "value": ""
            }
        ]
    },
    "inferenceAccelerators": [
        {
            "deviceName": "",
            "deviceType": ""
        }
    ]
}
```

您可以使用下列 AWS CLI 命令產生此任務定義範本。

```
aws ecs register-task-definition --generate-cli-skeleton
```

Amazon ECS 任務定義範例

您可以複製範例和程式碼片段，開始建立自己的任務定義。

您可以複製範例，然後在使用主控台中的透過 JSON 設定選項時貼上。確保自定範例，例如使用您的帳戶 ID。您可以在任務定義 JSON 中包含這些程式碼片段。如需詳細資訊，請參閱 使用主控台建立 Amazon ECS 任務定義 和 Amazon ECS 任務定義參數。

如需更多任務定義範例，請參閱 GitHub 上的 AWS 範例任務定義。

主題

Web 伺服器

以下是使用 Fargate 啟動類型上 Linux 容器並設定 Web 伺服器的範例任務定義：

```
{
   "containerDefinitions": [
      {
         "command": [
            "/bin/sh -c \"echo '<html> <head> <title>Amazon ECS Sample App</
title> <style>body {margin-top: 40px; background-color: #333;} </style> </
head><body> <div style=color:white;text-align:center> <h1>Amazon ECS Sample App</h1>
 <h2>Congratulations!</h2> <p>Your application is now running on a container in Amazon
 ECS.</p> </div></body></html>' >  /usr/local/apache2/htdocs/index.html && httpd-
foreground\""
         ],
         "entryPoint": [
            "sh",
```

```
            "-c"
         ],
         "essential": true,
         "image": "httpd:2.4",
         "logConfiguration": {
            "logDriver": "awslogs",
            "options": {
               "awslogs-group" : "/ecs/fargate-task-definition",
               "awslogs-region": "us-east-1",
               "awslogs-stream-prefix": "ecs"
            }
         },
         "name": "sample-fargate-app",
         "portMappings": [
            {
               "containerPort": 80,
               "hostPort": 80,
               "protocol": "tcp"
            }
         ]
      }
   ],
   "cpu": "256",
   "executionRoleArn": "arn:aws:iam::012345678910:role/ecsTaskExecutionRole",
   "family": "fargate-task-definition",
   "memory": "512",
   "networkMode": "awsvpc",
   "runtimePlatform": {
        "operatingSystemFamily": "LINUX"
    },
   "requiresCompatibilities": [
       "FARGATE"
    ]
}
```

以下是使用 Fargate 啟動類型上 Windows 容器並設定 Web 伺服器的範例任務定義：

```
{
    "containerDefinitions": [
        {
            "command": ["New-Item -Path C:\\inetpub\\wwwroot\\index.html -Type file
 -Value '<html> <head> <title>Amazon ECS Sample App</title> <style>body {margin-top:
 40px; background-color: #333;} </style> </head><body> <div style=color:white;text-
```

```
align:center> <h1>Amazon ECS Sample App</h1> <h2>Congratulations!</h2> <p>Your
 application is now running on a container in Amazon ECS.</p>'; C:\\ServiceMonitor.exe
 w3svc"],
            "entryPoint": [
                "powershell",
                "-Command"
            ],
            "essential": true,
            "cpu": 2048,
            "memory": 4096,
            "image": "mcr.microsoft.com/windows/servercore/iis:windowsservercore-
ltsc2019",
            "name": "sample_windows_app",
            "portMappings": [
                {
                    "hostPort": 80,
                    "containerPort": 80,
                    "protocol": "tcp"
                }
            ]
        }
    ],
    "memory": "4096",
    "cpu": "2048",
    "networkMode": "awsvpc",
    "family": "windows-simple-iis-2019-core",
    "executionRoleArn": "arn:aws:iam::012345678910:role/ecsTaskExecutionRole",
    "runtimePlatform": {"operatingSystemFamily": "WINDOWS_SERVER_2019_CORE"},
    "requiresCompatibilities": ["FARGATE"]
}
```

splunk 日誌驅動程式

以下程式碼片段示範如何在任務定義中使用 splunk 日誌驅動程式將日誌傳送到遠端服務。Splunk 字符參數被指定為秘密選項，因為它可以視為敏感資料。如需詳細資訊，請參閱將敏感資料傳遞至 Amazon ECS 容器。

```
"containerDefinitions": [{
  "logConfiguration": {
   "logDriver": "splunk",
   "options": {
    "splunk-url": "https://cloud.splunk.com:8080",
    "tag": "tag_name",
```

```
   },
   "secretOptions": [{
    "name": "splunk-token",
    "valueFrom": "arn:aws:secretsmanager:region:aws_account_id:secret:splunk-token-
KnrBkD"
}],
```

fluentd 日誌驅動程式

以下程式碼片段示範如何在任務定義中使用 fluentd 日誌驅動程式將日誌傳送到遠端服務。fluentd-address 值被指定為秘密選項，因為它可以視為敏感資料。如需詳細資訊，請參閱將敏感資料傳遞至 Amazon ECS 容器。

```
"containerDefinitions": [{
 "logConfiguration": {
  "logDriver": "fluentd",
  "options": {
   "tag": "fluentd demo"
  },
  "secretOptions": [{
   "name": "fluentd-address",
   "valueFrom": "arn:aws:secretsmanager:region:aws_account_id:secret:fluentd-address-
KnrBkD"
  }]
 },
 "entryPoint": [],
 "portMappings": [{
             "hostPort": 80,
             "protocol": "tcp",
             "containerPort": 80
             },
             {
  "hostPort": 24224,
  "protocol": "tcp",
  "containerPort": 24224
 }]
}],
```

gelf 日誌驅動程式

以下程式碼片段示範如何在任務定義中使用 gelf 日誌驅動程式，將日誌傳送到執行 Logstash (以 Gelf 日誌做為輸入) 的遠端主機。如需詳細資訊，請參閱logConfiguration。

```
"containerDefinitions": [{
 "logConfiguration": {
  "logDriver": "gelf",
  "options": {
   "gelf-address": "udp://logstash-service-address:5000",
   "tag": "gelf task demo"
  }
 },
 "entryPoint": [],
 "portMappings": [{
   "hostPort": 5000,
   "protocol": "udp",
   "containerPort": 5000
  },
  {
   "hostPort": 5000,
   "protocol": "tcp",
   "containerPort": 5000
  }
 ]
}],
```

外部執行個體上的工作負載

註冊 Amazon ECS 任務定義時，請使用 `requiresCompatibilities` 參數並指定 `EXTERNAL`，這可驗證任務定義是相容的，可在外部執行個體上執行 Amazon ECS 工作負載時使用。如果您使用主控台註冊任務定義，則必須使用 JSON 編輯器。如需詳細資訊，請參閱使用主控台建立 Amazon ECS 任務定義。

> **Important**
>
> 如果您的任務需要任務執行 IAM 角色，請確定在任務定義中指定它。

當您部署工作負載時，請在建立服務或執行獨立任務時使用 `EXTERNAL` 啟動類型。

以下是任務定義範例。

Linux

```
{
```

```
 "requiresCompatibilities": [
  "EXTERNAL"
 ],
 "containerDefinitions": [{
  "name": "nginx",
  "image": "public.ecr.aws/nginx/nginx:latest",
  "memory": 256,
  "cpu": 256,
  "essential": true,
  "portMappings": [{
   "containerPort": 80,
   "hostPort": 8080,
   "protocol": "tcp"
  }]
 }],
 "networkMode": "bridge",
 "family": "nginx"
}
```

Windows

```
{
 "requiresCompatibilities": [
  "EXTERNAL"
 ],
 "containerDefinitions": [{
  "name": "windows-container",
  "image": "mcr.microsoft.com/windows/servercore/iis:windowsservercore-ltsc2019",
  "memory": 256,
  "cpu": 512,
  "essential": true,
  "portMappings": [{
   "containerPort": 80,
   "hostPort": 8080,
   "protocol": "tcp"
  }]
 }],
 "networkMode": "bridge",
 "family": "windows-container"
}
```

Amazon ECR 映像和任務定義 IAM 角色

以下程式碼片段使用稱為 aws-nodejs-sample 的 Amazon ECR 映像，具有來自 123456789012.dkr.ecr.us-west-2.amazonaws.com 登錄檔的 v1 標籤。此任務中的容器會繼承 arn:aws:iam::123456789012:role/AmazonECSTaskS3BucketRole 角色的 IAM 許可。如需詳細資訊，請參閱Amazon ECS 任務 IAM 角色。

```
{
    "containerDefinitions": [
        {
            "name": "sample-app",
            "image": "123456789012.dkr.ecr.us-west-2.amazonaws.com/aws-nodejs-
sample:v1",
            "memory": 200,
            "cpu": 10,
            "essential": true
        }
    ],
    "family": "example_task_3",
    "taskRoleArn": "arn:aws:iam::123456789012:role/AmazonECSTaskS3BucketRole"
}
```

具有 命令的進入點

以下程式碼片段會示範使用進入點和命令引數的 Docker 容器語法。此容器 ping example.com 四次，然後結束。

```
{
    "containerDefinitions": [
        {
            "memory": 32,
            "essential": true,
            "entryPoint": ["ping"],
            "name": "alpine_ping",
            "readonlyRootFilesystem": true,
            "image": "alpine:3.4",
            "command": [
                "-c",
                "4",
                "example.com"
            ],
            "cpu": 16
```

```
        }
    ],
    "family": "example_task_2"
}
```

容器相依性

此程式碼片段示範任務定義的語法，其中有多個容器且指定容器相依性。在以下任務定義中，envoy 容器必須達到運作良好狀態 (由必要的容器運作狀態檢查參數來判斷)，app 容器才會啟動。如需詳細資訊，請參閱容器相依性。

```
{
  "family": "appmesh-gateway",
  "runtimePlatform": {
        "operatingSystemFamily": "LINUX"
  },
  "proxyConfiguration":{
      "type": "APPMESH",
      "containerName": "envoy",
      "properties": [
          {
              "name": "IgnoredUID",
              "value": "1337"
          },
          {
              "name": "ProxyIngressPort",
              "value": "15000"
          },
          {
              "name": "ProxyEgressPort",
              "value": "15001"
          },
          {
              "name": "AppPorts",
              "value": "9080"
          },
          {
              "name": "EgressIgnoredIPs",
              "value": "169.254.170.2,169.254.169.254"
          }
      ]
  },
  "containerDefinitions": [
```

```
    {
      "name": "app",
      "image": "application_image",
      "portMappings": [
        {
          "containerPort": 9080,
          "hostPort": 9080,
          "protocol": "tcp"
        }
      ],
      "essential": true,
      "dependsOn": [
        {
          "containerName": "envoy",
          "condition": "HEALTHY"
        }
      ]
    },
    {
      "name": "envoy",
      "image": "840364872350.dkr.ecr.region-code.amazonaws.com/aws-appmesh-
envoy:v1.15.1.0-prod",
      "essential": true,
      "environment": [
        {
          "name": "APPMESH_VIRTUAL_NODE_NAME",
          "value": "mesh/meshName/virtualNode/virtualNodeName"
        },
        {
          "name": "ENVOY_LOG_LEVEL",
          "value": "info"
        }
      ],
      "healthCheck": {
        "command": [
          "CMD-SHELL",
          "echo hello"
        ],
        "interval": 5,
        "timeout": 2,
        "retries": 3
      }
    }
  ],
```

```
  "executionRoleArn": "arn:aws:iam::123456789012:role/ecsTaskExecutionRole",
  "networkMode": "awsvpc"
}
```

Windows 任務定義範例

以下任務定義範例可協助您在 Amazon ECS 上開始使用 Windows 容器。

Example 適用於 Windows 的 Amazon ECS 主控台範例應用程式

以下任務定義是在 Amazon ECS 的之初次執行精靈中產生的 Amazon ECS 主控台範例應用程式，其已移植來使用 `microsoft/iis` Windows 容器映像。

```
{
  "family": "windows-simple-iis",
  "containerDefinitions": [
    {
      "name": "windows_sample_app",
      "image": "mcr.microsoft.com/windows/servercore/iis",
      "cpu": 1024,
      "entryPoint":["powershell", "-Command"],
      "command":["New-Item -Path C:\\inetpub\\wwwroot\\index.html -Type file -
Value '<html> <head> <title>Amazon ECS Sample App</title> <style>body {margin-top:
 40px; background-color: #333;} </style> </head><body> <div style=color:white;text-
align:center> <h1>Amazon ECS Sample App</h1> <h2>Congratulations!</h2> <p>Your
 application is now running on a container in Amazon ECS.</p>'; C:\\ServiceMonitor.exe
 w3svc"],
      "portMappings": [
        {
          "protocol": "tcp",
          "containerPort": 80
        }
      ],
      "memory": 1024,
      "essential": true
    }
  ],
  "networkMode": "awsvpc",
  "memory": "1024",
  "cpu": "1024"
}
```

Amazon ECS 叢集

Amazon ECS 叢集是任務或服務的邏輯分組。除了任務和服務之外，叢集還包含下列資源：

- 基礎設施容量可以結合下列項目：
  - AWS 雲端中的 Amazon EC2 執行個體
  - AWS 雲端中的無伺服器 (AWS Fargate)
  - 內部部署虛擬機器或伺服器
- 執行任務和服務的網路 (VPC 和子網路)

  當您將 Amazon EC2 執行個體用於容量時，子網路可以位於可用區域、Local Zones、Wavelength 區域或 AWS Outposts。
- 選用的命名空間

  命名空間用於與 Service Connect 進行服務對服務通訊。
- 監控選項

  CloudWatch Container Insights 需要額外付費，且為全受管服務。此服務會自動收集、彙總及總結 Amazon ECS 指標和日誌。

以下是有關 Amazon ECS 叢集的一般概念。

- 您可以建立叢集來分隔資源。
- 叢集是 AWS 區域 特定的。
- 叢集可以處於下列任何狀態。

  ACTIVE

  叢集已準備好接受任務，如果適用，您可以向叢集註冊容器執行個體。

  佈建中

  叢集具有與其相關聯的容量提供者，並且將建立容量提供者所需的資源。

  取消佈建中

  叢集具有與其相關聯的容量提供者，並且將刪除容量提供者所需的資源。

失敗

叢集具有與其相關聯的容量提供者，並且無法建立容量提供者所需的資源。

非作用中

已刪除叢集。具有 INACTIVE 狀態的叢集可能會在您的帳戶中保持可探索一段時間。此行為可能會在未來變更，因此請確定您不依賴持續存在的INACTIVE叢集。

- 叢集可以包含託管在 AWS Fargate、Amazon EC2 執行個體或外部執行個體上的任務組合。任務可以作為啟動類型或容量提供者策略在 Fargate 或 EC2 基礎設施上執行。如果您使用 EC2 做為啟動類型，Amazon ECS 不會追蹤和擴展 Amazon EC2 Auto Scaling 群組的容量。如需啟動類型的詳細資訊，請參閱「Amazon ECS啟動類型」。
- 叢集可以同時包含 Auto Scaling 群組容量提供者以及 Fargate 容量提供者。容量提供者策略只能包含 Auto Scaling 群組容量提供者或 Fargate 容量提供者。
- 您可以針對 EC2 啟動類型或 Auto Scaling 群組容量提供者使用不同的執行個體類型。執行個體一次只能註冊一個叢集。
- 您可以透過建立自訂 IAM 政策來限制對叢集的存取。如需詳細資訊，請參閱 中的Amazon ECS 叢集範例一節Amazon Elastic Container Service 身分型政策範例。
- 您可以使用 Service Auto Scaling 來擴展 Fargate 任務。如需詳細資訊，請參閱自動擴展 Amazon ECS 服務。
- 您可以設定叢集的預設 Service Connect 命名空間。設定預設的 Service Connect 命名空間之後，藉由開啟 Service Connect，在叢集中建立的任何新服務都可新增為命名空間中的用戶端服務。不需任何其他設定。如需詳細資訊，請參閱使用 Service Connect 以短名稱連接 Amazon ECS 服務。

Fargate 啟動類型的 Amazon ECS 叢集

Amazon ECS 容量提供者會管理叢集中任務的基礎設施擴展。每個叢集可以有一或多個容量提供者，以及選用的容量提供者策略。容量提供者策略會決定任務在叢集的容量提供者之間分散的方式。當您執行獨立任務或建立服務時，可以使用叢集的預設容量提供者策略，也可以使用能覆寫預設值的容量提供者策略。

當您在 上執行任務時 AWS Fargate，您不需要建立或管理容量。您只需要將下列任何預先定義的容量提供者與叢集建立關聯：

- Fargate
- Fargate Spot

使用 AWS Fargate 容量提供者的 Amazon ECS，您可以將 Fargate 和 Fargate Spot 容量與 Amazon ECS 任務搭配使用。

透過 Fargate Spot，您能以折扣價 (與 Fargate 價格相比) 執行可接受中斷的 Amazon ECS 任務。Fargate Spot 在備用運算容量上執行任務。當 AWS 需要恢復容量時，您的任務會受到兩分鐘警告。

當使用 Fargate 和 Fargate Spot 容量提供者的任務停止時，系統會將任務狀態變更事件傳送至 Amazon EventBridge。停止原因說明其原因。如需詳細資訊，請參閱Amazon ECS 任務狀態變更事件。

叢集可以同時包含 Fargate 和 Auto Scaling 群組容量提供者。不過，容量提供者策略只能包含 Fargate 或 Auto Scaling 群組容量提供者，不能同時包含兩者。如需詳細資訊，請參閱 Auto Scaling 群組容量提供者。

使用容量提供者時，請考慮下列事項：

- 您必須先將容量提供者與叢集建立關聯，才能將其與容量提供者策略建立關聯。
- 您可以為容量提供者策略指定最多 20 個容量提供者。
- 您無法將使用 Auto Scaling 群組容量提供者的服務更新為使用 Fargate 容量提供者。反之亦然。
- 在容量提供者策略中，如果沒有在主控台中對容量提供者指定 `weight` 值，則會使用預設值 1。如果使用 API 或 AWS CLI，則會0使用 的預設值。
- 在容量提供者策略中指定多個容量提供者時，至少有一個容量提供者必須具有大於零的權重值。任何權重為零的容量提供者都不會用來放置任務。如果您在策略中指定多個容量提供者權重均為零，則使用容量提供者策略的任何 `RunTask` 或 `CreateService` 動作都會失敗。
- 在容量提供者策略中，只有一個容量提供者已定義基準值。如果未指定基準值，則會使用預設值零。
- 叢集可以同時包含 Auto Scaling 群組容量提供者以及 Fargate 容量提供者。不過，容量提供者策略只能包含 Auto Scaling 群組或 Fargate 容量提供者，不能同時包含兩者。
- 叢集可以同時包含使用容量提供者和啟動類型的各種服務和獨立任務。服務可以更新為使用容量提供者策略，而非啟動類型。不過，若要執行此操作，您必須強制執行新部署。

Fargate Spot 終止通知

在需求極高期間，Fargate Spot 容量可能無法使用。這可能會導致 Fargate Spot 任務延遲。發生這種情況時，Amazon ECS 服務會重試啟動任務，直到所需的容量可用為止。Fargate 不會以隨需容量取代 Spot 容量。

當使用 Fargate Spot 容量的任務因 Spot 中斷而停止時，在任務停止之前會傳送兩分鐘的警告。警告會以任務狀態變更事件的形式傳送至 Amazon EventBridge，並作為 SIGTERM 訊號傳送給正在執行的任務。如果您使用 Fargate Spot 作為服務的一部分，在此情況中，服務排程器會收到中斷訊號，並嘗試在容量可用時啟動其他任務。只有一項任務的服務會中斷，直到容量可用為止。如需正常關機的詳細資訊，請參閱使用 ECS 正常關機。

若要確保您的容器在任務停止之前正常結束，可設定下列項目：

- 您可以在任務所使用的容器定義中指定 120 秒數的 `stopTimeout` 值 (或更少)。預設 `stopTimeout` 值為 30 秒。您可以指定更長的 `stopTimeout` 值，讓您從收到任務狀態變更事件到容器強制停止之間有更多時間。如需詳細資訊，請參閱容器逾時。
- SIGTERM 信號必須從容器內接收，以執行任何清理動作。無法處理此信號將導致任務在設定 `stopTimeout` 後收到 SIGKILL 信號，甚至可能導致資料遺失或毀損。

以下是任務狀態變更事件的程式碼片段。此程式碼片段顯示停止原因和 Fargate Spot 中斷的停止程式碼。

```
{
  "version": "0",
  "id": "9bcdac79-b31f-4d3d-9410-fbd727c29fab",
  "detail-type": "ECS Task State Change",
  "source": "aws.ecs",
  "account": "111122223333",
  "resources": [
    "arn:aws:ecs:us-east-1:111122223333:task/b99d40b3-5176-4f71-9a52-9dbd6f1cebef"
  ],
  "detail": {
    "clusterArn": "arn:aws:ecs:us-east-1:111122223333:cluster/default",
    "createdAt": "2016-12-06T16:41:05.702Z",
    "desiredStatus": "STOPPED",
    "lastStatus": "RUNNING",
    "stoppedReason": "Your Spot Task was interrupted.",
    "stopCode": "SpotInterruption",
    "taskArn": "arn:aws:ecs:us-east-1:111122223333:task/
b99d40b3-5176-4f71-9a52-9dbd6fEXAMPLE",
    ...
  }
}
```

以下是用來建立 Amazon ECS 任務狀態變更事件之 EventBridge 規則的事件模式。您可以選擇在 `detail` 欄位中指定叢集。若指定，則表示您會收到該叢集的任務狀態變更事件。如需建立 EventBridge 規則的詳細資訊，請參閱《Amazon EventBridge 使用者指南》中的 Amazon EventBridge 入門。

```
{
    "source": [
        "aws.ecs"
    ],
    "detail-type": [
        "ECS Task State Change"
    ],
    "detail": {
        "clusterArn": [
            "arn:aws:ecs:us-west-2:111122223333:cluster/default"
        ]
    }
}
```

為 Fargate 啟動類型建立 Amazon ECS 叢集

您可以建立叢集來定義任務和服務執行所在的基礎設施。

開始之前，請務必先完成 設定以使用 Amazon ECS。 中的步驟，並指派適當的 IAM 許可。如需詳細資訊，請參閱the section called “Amazon ECS 叢集範例”。Amazon ECS 主控台會透過建立 AWS CloudFormation 堆疊來建立 Amazon ECS 叢集所需的資源。

主控台會自動將 Fargate 和 Fargate Spot 容量提供者與叢集建立關聯。

除了叢集之外，主控台也會自動建立下列資源：

- 中的預設命名空間 AWS Cloud Map 與叢集的名稱相同。命名空間可讓您在叢集中建立的服務連線到命名空間中的其他服務，而不需要額外的組態。

  如需詳細資訊，請參閱互連 Amazon ECS 服務。

您可以修改下列選項：

- 變更與叢集相關聯的預設命名空間。
- 使用增強的可觀測性開啟 Container Insights，或 Container Insights。

CloudWatch Container Insights 會從您的容器化應用程式和微型服務收集、彙總及總結指標和日誌。Container Insights 還提供診斷資訊，例如容器重新啟動故障，您可以使用這些資訊快速隔離和解決這些問題。如需詳細資訊，請參閱the section called “使用具有增強可觀測性的 Container Insights 監控 Amazon ECS 容器”。

在 2024 年 12 月 2 日， AWS 發行了 Container Insights，並增強了 Amazon ECS 的可觀測性。此版本支援使用 Amazon EC2 和 Fargate 啟動類型的 Amazon ECS 叢集增強型可觀測性。在 Amazon ECS 上以增強的可觀測性設定 Container Insights 之後，Container Insights 會自動收集從叢集層級到環境中容器層級的詳細基礎設施遙測，並在儀表板中顯示您的資料，以向您顯示各種指標和維度。然後，您可以在 Container Insights 主控台上使用這些out-of-the-box儀表板，以更深入了解您的容器運作狀態和效能，並透過識別異常狀況更快速地緩解問題。

我們建議您使用具有增強可觀測性的 Container Insights，而不是 Container Insights，因為它可在您的容器環境中提供詳細的可見性，從而縮短解決的平均時間。

- 新增標籤以協助您識別叢集。

程序

建立新叢集 (Amazon ECS 主控台)

1. 開啟主控台，網址為 https://console.aws.amazon.com/ecs/v2。
2. 從導覽列中選取要使用的「區域」。
3. 在導覽窗格中，選擇叢集。
4. 在 Clusters (叢集) 頁面上，選擇 Create cluster (建立叢集)。
5. 在叢集組態下，設定下列項目：
   - 在叢集名稱下輸入唯一的名稱。

     名稱可以包含最多 255 個字母 (大小寫)、數字與連字號。
   - (選用) 若要讓 Service Connect 使用的命名空間與叢集名稱不同，請在命名空間中輸入唯一的名稱。
6. （選用）使用 Container Insights、展開監控，然後選擇下列其中一個選項：
   - 若要使用建議的 Container Insights 搭配增強型可觀測性，請選擇 Container Insights 搭配增強型可觀測性。
   - 若要使用 Container Insights，請選擇 Container Insights。

7. (選用) 為協助識別您的叢集，請展開 Tags (標籤)，然後設定標籤。

   [新增標籤] 選擇新增標籤，並執行下列動作：

   - 對於 Key (金鑰)，輸入金鑰名稱。
   - 對於 Value (值)，進入金鑰值。

   [移除標籤] 選擇標籤「金鑰」和「值」右側的移除。
8. 選擇 Create (建立)。

後續步驟

建立叢集之後，您可以為應用程式建立任務定義，然後將它們當做獨立任務或服務的一部分執行。如需詳細資訊，請參閱下列內容：

- Amazon ECS 任務定義
- 以 Amazon ECS 任務執行應用程式
- 使用 主控台建立 Amazon ECS 服務

EC2 啟動類型的 Amazon ECS 容量提供者

當您針對容量使用 Amazon EC2 執行個體時，可使用 Auto Scaling 群組來管理註冊到其叢集的 Amazon EC2 執行個體。Auto Scaling 有助於確保您有正確數量的 Amazon EC2 執行個體，可用於處理應用程式負載。

您可以使用 受管擴展功能，讓 Amazon ECS 管理 Auto Scaling 群組的縮減和縮減動作，或者您可以自行管理擴展動作。如需詳細資訊，請參閱使用叢集自動擴展自動管理 Amazon ECS 容量。

我們建議您建立新的空 Auto Scaling 群組。如果您使用現有的 Auto Scaling 群組，則任何與執行中群組相關聯以及註冊至 Amazon ECS 叢集的 Amazon EC2 執行個體在使用 Auto Scaling 群組建立容量提供者之前，可能無法正確的註冊容量提供者。在容量提供者策略中使用容量提供者時，這可能會造成問題。使用 `DescribeContainerInstances` 可以確認容器執行個體是否與容量提供者建立關聯。

> **Note**
>
> 若要建立空白 Auto Scaling 群組，請將所需的計數設定為零。建立容量提供者並將其與叢集關聯後，您就可以進行橫向擴展。

當您使用 Amazon ECS 主控台時，Amazon ECS 會代表您建立 Amazon EC2 啟動範本和 Auto Scaling 群組，做為 AWS CloudFormation 堆疊的一部分。它們字首為 `EC2ContainerService-<ClusterName>`。您可以使用 Auto Scaling 群組作為該叢集的容量提供者。

我們建議您使用受管執行個體耗盡，以允許正常終止不會中斷工作負載的 Amazon EC2 執行個體。此功能預設為開啟。如需詳細資訊，請參閱 安全地停止在 EC2 執行個體上執行的 Amazon ECS 工作負載

在主控台中使用 Auto Scaling 群組容量提供者時應考慮以下事項：

- Auto Scaling 群組的 `MaxSize` 必須大於零，才能水平擴展。
- Auto Scaling 群組不能具有執行個體權重設定。
- 如果 Auto Scaling 群組無法橫向擴展以容納執行的任務數目，則任務將無法轉換超出 `PROVISIONING` 狀態。
- 不要修改與容量提供者管理的 Auto Scaling 群組關聯的擴展政策資源。
- 如果在建立容量提供者時開啟受管擴展，則可將 Auto Scaling 群組所需的計數設定為 `0`。開啟受管擴展時，Amazon ECS 會管理 Auto Scaling 群組的縮減和橫向擴展動作。
- 您必須先將容量提供者與叢集建立關聯，才能將其與容量提供者策略建立關聯。
- 您可以為容量提供者策略指定最多 20 個容量提供者。
- 您無法將使用 Auto Scaling 群組容量提供者的服務更新為使用 Fargate 容量提供者。反之亦然。
- 在容量提供者策略中，如果沒有在主控台中對容量提供者指定 `weight` 值，則會使用預設值 1。如果使用 API 或 AWS CLI，則會`0`使用 的預設值。
- 在容量提供者策略中指定多個容量提供者時，至少有一個容量提供者必須具有大於零的權重值。任何權重為零的容量提供者都不會用來放置任務。如果您在策略中指定多個容量提供者權重均為零，則使用容量提供者策略的任何 `RunTask` 或 `CreateService` 動作都會失敗。
- 在容量提供者策略中，只有一個容量提供者已定義基準值。如果未指定基準值，則會使用預設值零。
- 叢集可以同時包含 Auto Scaling 群組容量提供者以及 Fargate 容量提供者。不過，容量提供者策略只能包含 Auto Scaling 群組或 Fargate 容量提供者，不能同時包含兩者。
- 叢集可以同時包含使用容量提供者和啟動類型的各種服務和獨立任務。服務可以更新為使用容量提供者策略，而非啟動類型。不過，若要執行此操作，您必須強制執行新部署。
- Amazon ECS 支援 Amazon EC2 Auto Scaling 暖集區。暖集區是一組準備投入使用的預先初始化 Amazon EC2 執行個體。當您的應用程式需要向外擴展時，Amazon EC2 Auto Scaling 會使用來自暖集區的預先初始化執行個體，而不是啟動冷執行個體。這可讓任何最終初始化程序在執行個體投入

服務之前執行。如需詳細資訊，請參閱為您的 Amazon ECS Auto Scaling 群組設定預先初始化的執行個體。

如需建立 Amazon EC2 Auto Scaling 啟動範本的詳細資訊，請參閱《Amazon EC2 Auto Scaling 使用者指南》中的 Auto Scaling 啟動範本。 Amazon EC2 Auto Scaling 如需有關建立 Amazon EC2 Auto Scaling 群組的詳細資訊，請參閱《Amazon EC2 Auto Scaling 使用者指南》中的 Auto Scaling 群組。

Amazon ECS 的 Amazon EC2 容器執行個體安全考量

您應該考量單一容器執行個體及其在威脅模型中的存取。例如，單一受影響的任務可能在相同執行個體上利用未受感染任務的 IAM 許可。

建議您採用下列動作以協助防止此種情況：

- 執行任務時，請勿使用管理員權限。
- 為您的任務指派具有最低權限存取的任務角色。

  容器代理程式會自動建立具有唯一憑證 ID 的字符，此字符可用於存取 Amazon ECS 資源。
- 針對任務中採用 `awsvpc` 網路模式的容器，如欲避免其存取提供給 Amazon EC2 執行個體設定檔的憑證資訊 (同時仍然允許任務角色所提供的許可)，請在代理程式組態檔案中將 `ECS_AWSVPC_BLOCK_IMDS` 代理程式組態變數設定為 true，並重新啟動代理程式。
- 使用 Amazon GuardDuty 執行期監控來偵測 AWS 環境中叢集和容器的威脅。執行期監控使用 GuardDuty 安全代理程式，為個別 Amazon ECS 工作負載新增執行期可見性，例如檔案存取、程序執行和網路連線。如需詳細資訊，請參閱《GuardDuty 使用者指南》中的 GuardDuty 執行期監控。 GuardDuty

為 Amazon EC2 啟動類型建立 Amazon ECS 叢集

您可以建立叢集來定義任務和服務執行所在的基礎設施。

開始之前，請務必先完成 設定以使用 Amazon ECS。 中的步驟，並指派適當的 IAM 許可。如需詳細資訊，請參閱the section called “Amazon ECS 叢集範例”。Amazon ECS 主控台提供簡單的方法，透過建立 AWS CloudFormation 堆疊來建立 Amazon ECS 叢集所需的資源。

為了使叢集建立程序盡可能簡單，主控台提供了許多可供選擇的預設選項，我們將在下方加以說明。主控台的大多數區段還有說明面板，以提供更多上下文。

您可以在建立叢集時註冊 Amazon EC2 執行個體，或在建立叢集之後用叢集註冊其他執行個體。

您可以修改下列預設選項：

- 變更執行個體啟動的子網路
- 變更用來控制容器執行個體流量的安全群組
- 變更與叢集相關聯的預設命名空間。

  命名空間可讓您在叢集中建立的服務連線到命名空間中的其他服務，而不需要額外的組態。預設命名空間與叢集名稱相同。如需詳細資訊，請參閱互連 Amazon ECS 服務。
- 使用增強的可觀測性開啟 Container Insights，或 Container Insights。

  CloudWatch Container Insights 會從您的容器化應用程式和微型服務收集、彙總及總結指標和日誌。Container Insights 還提供診斷資訊，例如容器重新啟動故障，您可以使用這些資訊快速隔離和解決這些問題。如需詳細資訊，請參閱the section called “使用具有增強可觀測性的 Container Insights 監控 Amazon ECS 容器”。

  在 2024 年 12 月 2 日， AWS 發行了 Container Insights，並增強了 Amazon ECS 的可觀測性。此版本支援使用 Amazon EC2 和 Fargate 啟動類型的 Amazon ECS 叢集增強型可觀測性。在 Amazon ECS 上以增強的可觀測性設定 Container Insights 之後，Container Insights 會自動收集從叢集層級到環境中容器層級的詳細基礎設施遙測，並在儀表板中顯示您的資料，以向您顯示各種指標和維度。然後，您可以在 Container Insights 主控台上使用這些out-of-the-box儀表板，以更深入了解您的容器運作狀態和效能，並透過識別異常狀況更快速地緩解問題。

  我們建議您使用具有增強可觀測性的 Container Insights，而不是 Container Insights，因為它可在您的容器環境中提供詳細的可見性，從而縮短解決的平均時間。
- 新增標籤以協助您識別叢集。

Auto Scaling 群組選項

如果使用 Amazon EC2 執行個體，則必須指定 Auto Scaling 群組來管理任務和服務執行所在的基礎設施。

當您選擇建立新的 Auto Scaling 群組時，系統會自動設定為以下行為：

- Amazon ECS 管理 Auto Scaling 群組的縮減和水平擴展動作。
- Amazon ECS 不會防止包含任務和位於 Auto Scaling 群組中的 Amazon EC2 執行個體在縮減動作期間被終止。如需詳細資訊，請參閱 AWS Auto Scaling 使用者指南中的執行個體保護。

您可以設定以下 Auto Scaling 群組屬性，以確定要為群組啟動的執行個體類型和數量：

- Amazon ECS 最佳化 AMI。
- 執行個體類型。
- 連線到執行個體時證明您身分的 SSH 金鑰對。如需有關如何建立 SSH 金鑰的資訊，請參閱《Amazon EC2 使用者指南》中的 Amazon EC2 金鑰對和 Linux 執行個體。 Amazon EC2
- 要為 Auto Scaling 群組啟動的最小執行個體數量。
- 將為 Auto Scaling 群組啟動的最大執行個體數量。

  為了水平擴展群組，最大值必須大於 0。

作為 AWS CloudFormation 堆疊的一部分，Amazon ECS 代表您建立 Amazon EC2 Auto Scaling 啟動範本和 Auto Scaling 群組。您為 AMI、執行個體類型和 SSH 金鑰對指定的值為啟動範本的一部分。範本字首會加上 `EC2ContainerService-<ClusterName>`，這使得它們容易識別。Auto Scaling 群組的字首為 `<ClusterName>-ECS-Infra-ECSAutoScalingGroup`。

為 Auto Scaling 群組啟動的執行個體使用啟動範本。

網路選項

根據預設，執行個體會啟動至「區域」的預設子網路中。系統會使用目前與子網路相關聯的安全群組，以控制傳送至容器執行個體的流量。您可以變更執行個體的子網路和安全群組。

您可以選擇現有的子網路。您可以使用現有的安全群組，也可以建立新的安全群組。當您建立新的安全群組時，您需要指定至少一個傳入規則。

傳入規則會決定哪些流量可以連接到您的容器執行個體，並包含下列屬性：

- 要允許的通訊協定
- 要允許的連接埠範圍
- 傳入流量（來源）

若要允許來自特定位址或 CIDR 區塊的傳入流量，請針對來源使用自訂選項，且具有允許的 CIDR。

若要允許來自所有目的地的傳入流量，請針對來源使用隨處。此選項會自動新增 0.0.0.0/0 IPv4 CIDR 區塊和 ::/0 IPv6 CIDR 區塊。

若要允許來自本機電腦的傳入流量，請針對來源使用來源群組。這會自動將您本機電腦目前的 IP 地址新增為允許的來源。

建立新叢集 (Amazon ECS 主控台)

在開始之前，請指派適當的 IAM 許可。如需詳細資訊，請參閱the section called “Amazon ECS 叢集範例”。

1. 開啟主控台，網址為 https://console.aws.amazon.com/ecs/v2。
2. 從導覽列中選取要使用的「區域」。
3. 在導覽窗格中，選擇叢集。
4. 在 Clusters (叢集) 頁面上，選擇 Create cluster (建立叢集)。
5. 在叢集組態下，設定下列項目：

   - 在叢集名稱下輸入唯一的名稱。

     名稱可以包含最多 255 個字母 (大小寫)、數字與連字號。

   - (選用) 若要讓 Service Connect 使用的命名空間與叢集名稱不同，請在命名空間中輸入唯一的名稱。

6. 將 Amazon EC2 執行個體新增至叢集，展開基礎設施，然後選取 Amazon EC2 執行個體。

   接下來，設定作為容量提供者的 Auto Scaling 群組：

   a. 要使用現有 Auto Scaling 群組，請從 Auto Scaling group (ASG) (Auto Scaling 群組 (ASG)) 中選取該群組。

   b. 若要建立 Auto Scaling 群組，請從 Auto Scaling group (ASG) (Auto Scaling 群組 (ASG)) 中選取 Create new group (建立新群組)，然後提供有關該群組的下列詳細資訊：

      - 針對佈建模型，選擇是否使用隨需執行個體或 Spot 執行個體。
      - 如果您選擇使用 Spot 執行個體，對於配置策略，請選擇執行個體使用的 Spot 容量集區（執行個體類型和可用區域）。

        對於大多數工作負載，您可以選擇價格容量最佳化。

        如需詳細資訊，請參閱《Amazon EC2 使用者指南》中的 Spot 執行個體的分配策略。
      - 針對容器執行個體 Amazon Machine Image (AMI)，選擇 Auto Scaling 群組執行個體的 Amazon ECS 最佳化 AMI。
      - 對於 EC2 instance type (EC2 執行個體類型)，選擇適合您工作負載的執行個體類型。

        如果 Auto Scaling 群組使用相同或類似的執行個體類型，則受管擴展效果最佳。

- 針對 EC2 執行個體角色，選擇現有的容器執行個體角色，或者您可以建立新的容器執行個體角色。

  如需詳細資訊，請參閱Amazon ECS 容器執行個體 IAM 角色。
- 對於容量，輸入 Auto Scaling 群組中要啟動的最小執行個體數和最大執行個體數。
- 對於 SSH key pair (SSH 金鑰對)，選擇在連線到執行個體時證明您身分的金鑰對。
- 若要允許較大的映像和儲存，請在根 EBS 磁碟區大小中輸入 GiB 中的值。

7. (選用) 若要變更 VPC 和子網路，請在 Amazon EC2 執行個體的聯網能力下，執行下列任一操作：

- 若要移除子網路，請在 Subnets (子網路) 下，對您要移除之每一個子網路選擇 X。
- 若要變更為非預設 VPC，請在 VPC 下，選擇現有的 VPC，然後在子網路下選擇子網路。
- 選擇安全群組。在安全群組下，選擇以下其中一個選項：
  - 若要使用現有的安全群組，請選擇使用現有安全群組，然後選擇安全群組。
  - 若要建立安全群組，請選擇建立新的安全群組。然後，針對傳入規則選擇新增規則。

    如需傳入規則的資訊，請參閱 網路選項。
- 若要自動將公有 IP 地址指派給 Amazon EC2 容器執行個體，針對自動指派公有 IP，請選擇下列其中一個選項：
  - 使用子網路設定：當執行個體啟動的子網路為公有子網路時，將公有 IP 地址指派給行個體。
  - 開啟：將公有 IP 位址指派給執行個體。

8. （選用）使用 Container Insights，展開監控，然後選擇下列其中一個選項：

- 若要使用建議的 Container Insights 搭配增強型可觀測性，請選擇 Container Insights 搭配增強型可觀測性。
- 若要使用 Container Insights，請選擇 Container Insights。

9. (選用)

   如果您搭配手動選項使用執行期監控，而且想要讓 GuardDuty 監控此叢集，請選擇新增標籤並執行下列動作：

- 針對金鑰，輸入 **`guardDutyRuntimeMonitoringManaged`**
- 針對數值，輸入 **`true`**。

10. (選用) 若要管理叢集標籤，請展開標籤，然後執行下列其中一項操作：

    [新增標籤] 選擇新增標籤，並執行下列動作：

- 對於 Key (金鑰)，輸入金鑰名稱。
- 對於 Value (值)，進入金鑰值。

[移除標籤] 選擇標籤「金鑰」和「值」右側的移除。

11. 選擇 Create (建立)。

後續步驟

建立叢集之後，您可以為應用程式建立任務定義，然後將它們當做獨立任務或服務的一部分執行。如需詳細資訊，請參閱下列內容：

- Amazon ECS 任務定義
- 以 Amazon ECS 任務執行應用程式
- 使用 主控台建立 Amazon ECS 服務

使用叢集自動擴展自動管理 Amazon ECS 容量

Amazon ECS 可管理在您的叢集註冊的 Amazon EC2 執行個體的擴展。這稱為 Amazon ECS 叢集自動擴展。當您建立 Amazon ECS Auto Scaling 群組容量提供者時，您可以開啟受管擴展。然後，您為此 Auto Scaling 群組中的執行個體使用率設定目標百分比 (`targetCapacity`)。Amazon ECS 會為您的 Auto Scaling 群組建立兩個自訂 CloudWatch 指標和目標追蹤擴展政策。然後，Amazon ECS 會根據任務使用的資源使用率來管理縮減和縮減動作。

針對每個與叢集關聯的 Auto Scaling 群組容量提供者，Amazon ECS 會建立並管理以下資源：

- 低指標值 CloudWatch 警示
- 高指標值 CloudWatch 警示
- 目標追蹤擴展政策

> **Note**
>
> Amazon ECS 會建立目標追蹤擴展政策，並將其連接到 Auto Scaling 群組。若要更新目標追蹤擴展政策，請更新容量提供者受管擴展設定，而不是直接更新擴展政策。

如果關閉受管擴展或解除容量提供者與叢集的關聯，Amazon ECS 會移除 CloudWatch 指標和目標追蹤擴展政策資源。

Amazon ECS 使用以下指標來決定要採取的動作：

`CapacityProviderReservation`

特定容量提供者使用的容器執行個體百分比。Amazon ECS 會產生此指標。

Amazon ECS 將 `CapacityProviderReservation` 值設定為 0-100 之間的數字。Amazon ECS 使用下列公式來表示 Auto Scaling 群組中剩餘容量的比率。之後，Amazon ECS 會將指標發佈到 CloudWatch。如需如何計算指標的詳細資訊，請參閱 Amazon ECS Cluster Auto Scaling 上的 Deep Dive。

```
CapacityProviderReservation = (number of instances needed) / (number of running
 instances) x 100
```

`DesiredCapacity`

Auto Scaling 群組的容量。此指標不會發佈至 CloudWatch。

之後，Amazon ECS 會在 `AWS/ECS/ManagedScaling` 命名空間中將 `CapacityProviderReservation` 指標發佈到 CloudWatch。`CapacityProviderReservation` 指標會導致發生下列其中一個動作：

`CapacityProviderReservation` 值等於 `targetCapacity`

Auto Scaling 群組不需要橫向擴展或縮減。已達到目標使用率百分比。

`CapacityProviderReservation` 值大於 `targetCapacity`

有更多任務使用比您的 `targetCapacity` 百分比高的容量百分比。`CapacityProviderReservation` 指標增加的值會導致相關聯的 CloudWatch 警示採取動作。此警示會更新 Auto Scaling 群組的 `DesiredCapacity` 值。Auto Scaling 群組會使用此值啟動 EC2 執行個體，然後向叢集註冊這些執行個體。

當 `targetCapacity` 的預設值為 100% 時，新任務會在橫向擴展期間處於 `PENDING` 狀態，因為執行個體上沒有可用容量來執行任務。新執行個體向 ECS 註冊後，這些任務將在新執行個體上開始執行。

`CapacityProviderReservation` 值小於 `targetCapacity`

使用容量百分比低於 `targetCapacity` 的百分比的任務更少，並且至少有一個可以終止的執行個體。`CapacityProviderReservation` 指標減少的值會導致相關聯的 CloudWatch 警示採取動作。此警示會更新 Auto Scaling 群組的 `DesiredCapacity` 值。Auto Scaling 群組會使用此值終止 EC2 容器執行個體，然後向叢集取消註冊這些執行個體。

Auto Scaling 群組遵循群組終止政策來確定在縮減事件期間首先終止哪些執行個體。此外，它還可以避免開啟執行個體縮減保護設定的執行個體。如果您開啟受管終止保護，叢集自動擴展可以管理哪些執行個體具有執行個體縮減保護設定。如需有關受管終止保護的詳細資訊，請參閱 控制執行個體 Amazon ECS 終止。如需 Auto Scaling 群組終止執行個體的詳細資訊，請參閱 Amazon EC2 Auto Scaling 使用者指南中的控制在縮減期間會終止的 Auto Scaling 執行個體。

使用叢集自動擴展時應考慮以下事項：

- 請勿以任何擴展政策變更或管理與容量提供者相關聯的 Auto Scaling 群組所需的容量，Amazon ECS 管理的政策除外。
- Amazon ECS 從 0 個執行個體向外擴展時，會自動啟動 2 個執行個體。
- Amazon ECS 會使用`AWSServiceRoleForECS`服務連結 IAM 角色來取得 AWS Auto Scaling 代表您呼叫 所需的許可。如需詳細資訊，請參閱使用 Amazon ECS 的服務連結角色。
- 將容量提供者與 Auto Scaling 群組搭配使用時，建立容量提供者的使用者、群組或角色需要 `autoscaling:CreateOrUpdateTags` 許可。這是因為當群組與容量提供者產生關聯時，Amazon ECS 會將標籤加入至 Auto Scaling 群組。

  Important

  確保您使用的任何工具不會將 `AmazonECSManaged` 標籤從 Auto Scaling 群組中移除。如果移除此標籤，Amazon ECS 將無法管理擴展。

- 叢集自動擴展不會修改群組的 MinimumCapacity (容量下限) 或 MaximumCapacity (容量上限)。為了讓群組進行橫向擴展，MaximumCapacity (容量上限) 必須大於零。
- 當自動擴展 (受管擴展) 開啟時，一個容量提供者一次只能連接到一個叢集。如果容量提供者已關閉受管擴展，您可以將其關聯到多個叢集。
- 當受管擴展關閉時，容量提供者不會縮減或橫向擴展。您可以使用容量提供者策略在容量提供者之間平衡您的任務。
- `binpack` 就容量而言，策略是最有效率的策略。

- 當目標容量低於 100% 時，置放策略需要使用binpack策略，而spread策略的順序不會高於binpack策略。這可防止容量提供者向外擴展，直到每個任務都有專用執行個體或達到限制為止。

開啟叢集自動擴展

您可以使用 主控台或 開啟叢集自動擴展 AWS CLI。

當您使用主控台建立 EC2 啟動類型的叢集時，Amazon ECS 會代表您建立 Auto Scaling 群組，並設定目標容量。如需詳細資訊，請參閱為 Amazon EC2 啟動類型建立 Amazon ECS 叢集。

您也可以建立 Auto Scaling 群組，然後將其指派給叢集。如需詳細資訊，請參閱更新 Amazon ECS 容量提供者。

當您使用 時 AWS CLI，在您建立叢集之後

1. 建立容量提供者前，您需要建立 Auto Scaling 群組。如需詳細資訊，請參閱《Amazon EC2 Auto Scaling 使用者指南》中的 Auto Scaling 群組。
2. 使用 `put-cluster-capacity-providers` 修改叢集容量提供者。如需詳細資訊，請參閱開啟 Amazon ECS 叢集自動擴展。

最佳化 Amazon ECS 叢集自動擴展

在 Amazon EC2 上執行 Amazon ECS 的客戶可以利用叢集自動擴展來管理 Amazon EC2 Auto Scaling 群組的擴展。透過叢集自動擴展，您可以設定 Amazon ECS 自動擴展 Auto Scaling 群組，並專注於執行任務。Amazon ECS 可確保 Auto Scaling 群組可視需要縱向擴展，而不需要進一步介入。Amazon ECS 容量提供者可用來管理叢集中的基礎設施，確保有足夠的容器執行個體可滿足您的應用程式需求。若要了解叢集自動擴展如何在 許可下運作，請參閱 Amazon ECS Cluster Auto Scaling 上的 Deep Dive。

叢集自動擴展依賴於與 Auto Scaling 群組的 CloudWatch 型整合，以調整叢集容量。因此，它具有與相關聯的固有延遲

- 發佈 CloudWatch 指標、
- 指標`CapacityProviderReservation`違反 CloudWatch 警示所需的時間 （ 高和低 ）
- 新啟動的 Amazon EC2 執行個體暖機所需的時間。您可以採取下列動作，讓叢集自動擴展更具回應性，以加快部署速度：

容量提供者步進擴展大小

Amazon ECS 容量提供者將增加/縮減容器執行個體，以滿足應用程式的需求。Amazon ECS 將啟動的執行個體數目下限預設為 1。如果需要多個執行個體來放置待定任務，這可能會為您的部署增加額外的時間。您可以透過 `minimumScalingStepSize` Amazon ECS API 增加 ，以增加 Amazon ECS 一次向內擴展或向外擴展的執行個體數量下限。太低`maximumScalingStepSize`的 可以限制一次擴展或向外擴展的容器執行個體數量，這可能會減慢部署速度。

> **Note**
>
> 此組態目前只能透過 `CreateCapacityProvider`或 `UpdateCapacityProvider` APIs使用。

執行個體暖機期

執行個體暖機期是新啟動的 Amazon EC2 執行個體可以為 Auto Scaling 群組貢獻 CloudWatch 指標的期間。在指定的暖機期到期後，執行個體會計入 Auto Scaling 群組的彙總指標，而叢集自動擴展會繼續進行其下一次的計算反覆運算，以估計所需的執行個體數量。

的預設值`instanceWarmupPeriod`為 300 秒，您可以透過 `CreateCapacityProvider`或 `UpdateCapacityProvider` APIs 將 設定為較低的值，以獲得更靈敏的擴展。建議您將值設定為大於 60 秒，以避免過度佈建。

備用容量

如果您的容量提供者沒有容器執行個體可供放置任務，則需要即時啟動 Amazon EC2 執行個體來增加（向外擴展）叢集容量，並等待它們啟動，然後才能在其上啟動容器。這可以大幅降低任務啟動率。您在此處有兩個選項。

在這種情況下，已啟動備用 Amazon EC2 容量並準備好執行任務，將提高有效的任務啟動率。您可以使用 `Target Capacity`組態來表示您想要在叢集中維護備用容量。例如，將 設定為 `Target Capacity` 80% 表示您的叢集隨時需要 20% 的備用容量。此備用容量可以允許立即啟動任何獨立任務，確保任務啟動不會受到調節。此方法的權衡是保留備用叢集容量的潛在增加成本。

您可以考慮的替代方法是將總空間新增至服務，而不是容量提供者。這表示，您可以修改目標追蹤擴展指標或服務自動擴展的步進擴展閾值，來增加服務中的複本數量，而不是減少啟動備用容量的`Target Capacity`組態。請注意，此方法只會對尖峰工作負載有所幫助，但當您第一次部署新服務並從 0 到 N 任務時，不會產生影響。如需相關擴展政策的詳細資訊，請參閱《Amazon Elastic Container Service 開發人員指南》中的目標追蹤擴展政策或步驟擴展政策。

Amazon ECS 受管擴展行為

當您有使用受管擴展的 Auto Scaling 群組容量提供者時，Amazon ECS 會預估要新增至叢集的最佳執行個體數量，並使用 值來決定要請求或發行的執行個體數量。

受管向外擴展行為

Amazon ECS 會遵循服務、獨立任務或叢集預設值中的容量提供者策略，為每個任務選取容量提供者。Amazon ECS 會針對單一容量供應商遵循這些步驟的剩餘步驟。

沒有容量提供者策略的任務會被容量提供者忽略。沒有容量提供者策略的等待中任務不會導致任何容量提供者橫向擴展。如果任務或服務設定了啟動類型，則任務或服務無法設定容量提供者策略。

以下詳細說明向外擴展行為。

- 將此容量提供者的所有佈建任務分組，以便每個群組具有相同的精確資源需求。
- 當您在 Auto Scaling 群組中使用多個執行個體類型時，Auto Scaling 群組中的執行個體會依其參數排序。這些參數包含 vCPU、記憶體、彈性網路介面 (ENI)、連接埠和 GPU。為每個參數選取最小和最大的執行個體類型。如需如何選擇執行個體類型的詳細資訊，請參閱Amazon ECS 的 Amazon EC2 容器執行個體。

  > **Important**
  >
  > 如果一組任務的資源需求大於 Auto Scaling 群組中最小執行個體類型，則該任務群組無法使用此容量提供者執行。容量提供者不會擴展 Auto Scaling 群組。任務會維持在 `PROVISIONING` 狀態。
  >
  > 為了防止任務停留在 `PROVISIONING` 狀態，建議您針對不同的最低資源需求，建立個別的 Auto Scaling 群組和容量提供者。當您執行任務或建立服務時，請僅將容量提供者新增至可以在 Auto Scaling 群組中最小執行個體類型上執行任務的容量提供者策略。對於其他參數，您可以使用置放條件限制

- 針對每個任務群組，Amazon ECS 會計算執行未放置任務所需的執行個體數目。此計算會使用 `binpack` 策略。此策略會說明任務的 vCPU、記憶體、彈性網路介面 (ENI)、連接埠和 GPU 需求。其也會說明 Amazon EC2 執行個體的資源可用性。最大執行個體類型的值會被視為所計算執行個體數目的最大值。最小執行個體類型的值會作為保護使用。如果最小的執行個體類型無法執行任務至少一個執行個體，則計算會認為該任務不相容。因此，任務會排除在橫向擴展計算之外。當所有任務都與最小執行個體類型不相容時，叢集自動擴展將會停止，而且 `CapacityProviderReservation` 值仍為 `targetCapacity` 值。

- 如果是下列情況之一，Amazon ECS 會依照 `minimumScalingStepSize` 將 `CapacityProviderReservation` 指標發佈到 CloudWatch：
  - 計算的執行個體計數上限小於最小擴展步驟大小。
  - `maximumScalingStepSize` 或計算執行個體計數上限的較低值。
- CloudWatch 警示會使用容量提供者的 `CapacityProviderReservation` 指標。當 `CapacityProviderReservation` 指標大於 `targetCapacity` 值時，警示也會增加 Auto Scaling 群組的 `DesiredCapacity`。`targetCapacity` 值是在叢集自動擴展活動期間，傳送到 CloudWatch 警示的容量提供者設定。

  預設值`targetCapacity`為 100%。
- Auto Scaling 群組會啟動其他 EC2 執行個體。為了防止過度佈建，Auto Scaling 會確保在啟動新執行個體之前，已穩定最近啟動的 EC2 執行個體容量。Auto Scaling 會檢查是否所有現有的執行個體都已超過 `instanceWarmupPeriod` (現在時間減去執行個體啟動時間)。對於位於 內的執行個體，會封鎖橫向擴展`instanceWarmupPeriod`。

  新啟動執行個體的暖機預設秒數為 300 秒。

如需詳細資訊，請參閱 Deep dive on Amazon ECS cluster auto scaling (《深入瞭解 Amazon ECS 叢集自動擴展》)。

擴增注意事項

使用擴增程序時，請注意以下事項：

- 儘管存在多個置放限制條件，我們建議您只使用 `distinctInstance` 任務置放限制條件。此舉可以防止橫向擴展程序停止，因為您使用的置放限制條件與取樣的執行個體不相容。
- 如果 Auto Scaling 群組使用相同或類似的執行個體類型，則受管擴展效果最佳。
- 當需要橫向擴展程序且目前沒有正在執行的容器執行個體時，Amazon ECS 一開始會一律橫向擴展至兩個執行個體，再執行額外橫向擴展或縮減程序。任何額外的橫向擴展都會等待執行個體暖機期間。關於縮減流程，Amazon ECS 會在橫向擴展程序後等待 15 分鐘，然後隨時開始縮減程序。
- 第二個橫向擴展步驟需要等到 `instanceWarmupPeriod` 過期，這可能會影響整體擴展限制。如果您需要減少此時間，請確定 `instanceWarmupPeriod` 足夠大，讓 EC2 執行個體啟動和啟動 Amazon ECS 代理程式 （可防止過度佈建）。
- 叢集自動擴展支援容量提供者 Auto Scaling 群組中的啟動組態、啟動範本和多種執行個體類型。您還可以使用屬性型執行個體類型選取範圍，而不使用多種執行個體類型。

- 將 Auto Scaling 群組與隨需執行個體和多個執行個體類型或 Spot 執行個體搭配使用時，較大的執行個體類型的優先級更高，並且不要指定權重。目前不支援指定權重。如需詳細資訊，請參閱 AWS Auto Scaling 使用者指南中的具有多個執行個體類型的 Auto Scaling 群組。
- 然後，如果計算出的執行個體計數上限小於擴展步驟大小下限，或為 `maximumScalingStepSize` 或計算出的執行個體計數上限中的較小者，則 Amazon ECS 會啟動 `minimumScalingStepSize`。
- 如果 Amazon ECS 服務或 `run-task`啟動任務，且容量提供者容器執行個體沒有足夠的資源來啟動任務，則 Amazon ECS 會限制每個叢集具有此狀態的任務數量，並防止任何任務超過此限制。如需詳細資訊，請參閱Service Quotas。

受管縮減行為

Amazon ECS 會監控叢集中每個容量提供者的容器執行個體。當容器執行個體未執行任務時，容器執行個體會被視為空白，Amazon ECS 會開始縮減程序。

CloudWatch 縮減警示需要 15 個資料點 (15 分鐘)，然後才能開始 Auto Scaling 群組的縮減程序。在縮減程序開始後直到 Amazon ECS 需要減少已註冊的容器執行個體數量，Auto Scaling 群組將 `DesireCapacity` 值設定為大於一個執行個體且每分鐘小於 50%。

如果 Amazon ECS 在縮減程序進行期間請求橫向擴展 (`CapacityProviderReservation` 大於 100 時)，縮減程序將停止，並在需要時從頭開始。

以下內容會詳細介紹縮減行為：

1. Amazon ECS 會計算空白容器執行個體的數量。即使常駐程式任務正在執行，則會將容器執行個體視為空白。
2. Amazon ECS 會將 `CapacityProviderReservation` 值設定為介於 0-100 之間的數字，此數字會使用下列公式來表示 Auto Scaling 群組需要相對於其實際大小的比率，並以百分比表示。之後，Amazon ECS 會將指標發佈到 CloudWatch。如需指標計算方式的詳細資訊，請參閱深入了解 Amazon ECS 叢集自動擴展

   ```
   CapacityProviderReservation = (number of instances needed) / (number of running
    instances) x 100
   ```

3. `CapacityProviderReservation` 指標會產生 CloudWatch 警示。此警示會更新 Auto Scaling 群組的 `DesiredCapacity` 值。然後，執行下列其中一個動作：

- 如果您不使用容量提供者受管終止，Auto Scaling 群組會選擇使用 Auto Scaling 群組終止政策的 EC2 執行個體，並終止執行個體，直到 EC2 執行個體數量達到 `DesiredCapacity`。之後，從叢集中取消註冊容器執行個體。
- 如果所有容器執行個體都使用受管終止保護，則 Amazon ECS 會移除空白容器執行個體的縮減保護。之後，Auto Scaling 群組能夠終止 EC2 執行個體。之後，從叢集中取消註冊容器執行個體。

控制執行個體 Amazon ECS 終止

> **Important**
>
> 您必須在 Auto Scaling 群組上開啟 Auto Scaling 執行個體縮減保護，才能使用叢集自動擴展的受管終止保護功能。

受管終止保護可讓叢集自動擴展控制哪些執行個體終止。當您使用受管終止保護時，Amazon ECS 只會終止沒有任何執行中 Amazon ECS 任務的 EC2 執行個體。系統會忽略使用 `DAEMON` 排程策略的服務執行的任務，而且即使執行個體正在執行這些任務，叢集自動擴展也可以終止執行個體。這是因為叢集中的所有執行個體都在執行這些任務。

Amazon ECS 會先開啟 Auto Scaling 群組中 EC2 執行個體的執行個體縮減保護選項。然後，Amazon ECS 會將任務放置在執行個體上。在執行個體上停止所有非常駐程式任務時，Amazon ECS 會啟動縮減程序並關閉 EC2 執行個體的縮減保護。之後，Auto Scaling 群組就能終止執行個體。

Auto Scaling 執行個體縮減保護可控制 Auto Scaling 要終止哪些 EC2 執行個體。在縮減過程中，無法終止已啟用縮減功能的執行個體。如需有關防止 Auto Scaling 執行個體縮減保護的詳細資訊，請參閱 Amazon EC2 Auto Scaling 使用者指南中的使用執行個體縮減保護。

您可以設定`targetCapacity`百分比，以便有備用容量。這有助於未來的任務更快速地啟動，因為 Auto Scaling 群組不需要啟動更多執行個體。Amazon ECS 使用目標容量值來管理服務建立的 CloudWatch 指標。Amazon ECS 會管理 CloudWatch 指標。Auto Scaling 群組會被視為穩定狀態，因此不需要擴展動作。值可以介於 0-100% 之間。例如，若要設定 Amazon ECS，使其在用於 Amazon ECS 任務時能保持 10% 的可用容量，請將該目標容量值設定為 90%。設定容量提供者的 `targetCapacity` 值時，請考慮以下事項。

- 小於 100% 的 `targetCapacity` 值，表示叢集中需要具備的可用容量數量 (Amazon EC2 執行個體)。可用容量代表沒有執行中的任務。
- 在沒有其他 `binpack` 的情況下，置放限制條件 (如可用區域) 會強制 Amazon ECS 最終為各執行個體執行一個任務，但這可能不是必要行為。

您必須在 Auto Scaling 群組上開啟 Auto Scaling 執行個體縮減保護，才能使用受管終止保護功能。如果您未開啟縮減保護，則開啟受管終止保護可能會產生不良行為。例如，您可能會讓執行個體停留在消耗狀態。如需詳細資訊，請參閱 Amazon EC2 Auto Scaling 使用者指南中的使用執行個體縮減保護。

當您搭配容量提供者使用終止保護時，請勿在與容量提供者相關聯的 Auto Scaling 群組上執行任何手動動作 (例如分離執行個體)。手動動作可能會中斷容量提供者的縮減作業。如果您從 Auto Scaling 群組中分離執行個體，您還需要從 Amazon ECS 叢集中取消註冊已分離的執行個體。

開啟 Amazon ECS 叢集自動擴展

您可以開啟叢集自動擴展，讓 Amazon ECS 管理註冊到叢集的 Amazon EC2 執行個體擴展。

如果您想要使用主控台開啟叢集自動擴展，請參閱為 Amazon ECS 建立容量提供者。

開始之前，請先建立 Auto Scaling 群組和容量提供者。如需詳細資訊，請參閱the section called “EC2 啟動類型的容量提供者”。

若要開啟叢集自動擴展，請將容量提供者與叢集建立關聯，然後開啟叢集自動擴展。

1. 使用 `put-cluster-capacity-providers` 命令，將一或多個容量提供者關聯到叢集。

   若要新增 AWS Fargate 容量提供者，請在請求中包含 `FARGATE`和 `FARGATE_SPOT` 容量提供者。如需詳細資訊，請參閱 AWS CLI 命令參考中的 `put-cluster-capacity-providers`。

   ```
   aws ecs put-cluster-capacity-providers \
     --cluster ClusterName \
     --capacity-providers CapacityProviderName FARGATE FARGATE_SPOT \
     --default-capacity-provider-strategy capacityProvider=CapacityProvider,weight=1
   ```

   若要為 EC2 啟動類型新增 Auto Scaling 群組，請在請求中包含 Auto Scaling 群組名稱。如需詳細資訊，請參閱 AWS CLI 命令參考中的 `put-cluster-capacity-providers`。

   ```
   aws ecs put-cluster-capacity-providers \
     --cluster ClusterName \
     --capacity-providers CapacityProviderName \
     --default-capacity-provider-strategy capacityProvider=CapacityProvider,weight=1
   ```

2. 使用 `describe-clusters` 命令來確認關聯是否成功。如需詳細資訊，請參閱 AWS CLI 命令參考中的 `describe-clusters`。

   ```
   aws ecs describe-clusters \
     --cluster ClusterName \
   ```

```
  --include ATTACHMENTS
```

3. 使用 update-capacity-provider 命令，啟用容量提供者的受管自動擴展。如需詳細資訊，請參閱 AWS CLI 命令參考中的 update-capacity-provider。

```
aws ecs update-capacity-provider \
  --capacity-providers CapacityProviderName \
  --auto-scaling-group-provider managedScaling=ENABLED
```

關閉 Amazon ECS 叢集自動擴展

當您需要更精細地控制已註冊至叢集的 EC2 執行個體時，您可以關閉叢集自動擴展，

若要關閉叢集的叢集自動擴展，您可以將容量提供者與叢集中開啟受管擴展的關聯取消關聯，或更新容量提供者以關閉受管擴展。

取消容量提供者的關聯

使用下列步驟解除容量提供者與叢集之間的關聯。

1. 使用 put-cluster-capacity-providers 命令解除 Auto Scaling 群組容量提供者與叢集之間的關聯。叢集可以保持與 AWS Fargate 容量提供者的關聯。如需詳細資訊，請參閱 AWS CLI 命令參考中的 put-cluster-capacity-providers。

```
aws ecs put-cluster-capacity-providers \
  --cluster ClusterName \
  --capacity-providers FARGATE FARGATE_SPOT \
  --default-capacity-provider-strategy '[]'
```

使用 put-cluster-capacity-providers 命令解除 Auto Scaling 群組容量提供者與叢集之間的關聯。如需詳細資訊，請參閱 AWS CLI 命令參考中的 put-cluster-capacity-providers。

```
aws ecs put-cluster-capacity-providers \
  --cluster ClusterName \
  --capacity-providers [] \
  --default-capacity-provider-strategy '[]'
```

2. 使用 describe-clusters 命令來確認是否成功解除關聯。如需詳細資訊，請參閱 AWS CLI 命令參考中的 describe-clusters。

```
aws ecs describe-clusters \
  --cluster ClusterName \
  --include ATTACHMENTS
```

關閉容量提供者的受管擴展

使用下列步驟關閉容量提供者的受管擴展。

- 使用 update-capacity-provider 命令，關閉容量提供者的受管自動擴展。如需詳細資訊，請參閱 AWS CLI 命令參考中的 update-capacity-provider。

```
aws ecs update-capacity-provider \
  --capacity-providers CapacityProviderName \
  --auto-scaling-group-provider managedScaling=DISABLED
```

為 Amazon ECS 建立容量提供者

叢集建立完成後，您可以為 EC2 啟動類型建立新的容量提供者 (Auto Scaling 群組）。容量提供者可協助您管理和擴展應用程式的基礎設施。

建立容量提供者前，您需要建立 Auto Scaling 群組。如需詳細資訊，請參閱《Amazon EC2 Auto Scaling 使用者指南》中的 Auto Scaling 群組。

建立叢集的容量提供者 (Amazon ECS 主控台)

1. 開啟主控台，網址為 https://console.aws.amazon.com/ecs/v2。
2. 在導覽窗格中，選擇叢集。
3. 在叢集頁面上，選擇叢集。
4. 在 Cluster:*name* (叢集：名稱) 頁面中，選擇 Infrastructure (基礎設施)，然後選擇 Create (建立)。
5. 在 Create capacity providers (建立容量提供者) 頁面中，設定下列選項。
   a. 在 Basic details (基本詳細資訊) 下的 Capacity provider name (容量提供者名稱) 中，輸入容量提供者名稱。
   b. 在 Auto Scaling group (Auto Scaling 群組) 下的 Use an existing Auto Scaling group (使用現有 Auto Scaling 群組) 欄位中，選擇 Auto Scaling 群組。
   c. (選用) 若要設定擴展政策，請在 Scaling policies (擴展政策) 下設定下列選項。

- 若要讓 Amazon ECS 管理縮減和橫向擴展動作，請選取 Turn on managed scaling (開啟受管擴展)。
- 若要防止具有執行中 Amazon ECS 任務的 EC2 執行個體遭到終止，請選取 Turn on scaling protection (開啟擴展保護)。
- 在 Set target capacity (設定目標容量) 欄位中，輸入在 Amazon ECS 受管目標追蹤擴展政策中使用的 CloudWatch 指標的目標值。

6. 選擇 Create (建立)。

更新 Amazon ECS 容量提供者

使用 Auto Scaling 群組作為容量提供者時，您可以修改群組的擴展政策。

更新叢集的容量提供者 (Amazon ECS 主控台)

1. 開啟主控台，網址為 https://console.aws.amazon.com/ecs/v2。
2. 在導覽窗格中，選擇叢集。
3. 在叢集頁面上，選擇叢集。
4. 在 Cluster : ***name*** (叢集：名稱) 頁面中，選擇 Infrastructure (基礎設施)，然後選擇 Update (更新)。
5. 在 Create capacity providers (建立容量提供者) 頁面中，設定下列選項。
   - 在 Auto Scaling 群組下的擴展政策下，設定下列選項。
     - 若要讓 Amazon ECS 管理縮減和橫向擴展動作，請選取 Turn on managed scaling (開啟受管擴展)。
     - 若要防止具有執行中 Amazon ECS 任務的 EC2 執行個體遭到終止，請選取開啟擴展保護。
     - 在 Set target capacity (設定目標容量) 欄位中，輸入在 Amazon ECS 受管目標追蹤擴展政策中使用的 CloudWatch 指標的目標值。
6. 選擇更新。

刪除 Amazon ECS 容量提供者

如果 Auto Scaling 群組容量提供者已使用完畢，則可將其刪除。刪除群組後，Auto Scaling 群組容量提供者會轉換為 `INACTIVE` 狀態。具有 `INACTIVE` 狀態的容量提供者可能會在您的帳戶中保持可探

索一段時間。不過，此行為未來可能變動，因此不建議依賴 INACTIVE 容量提供者得以保存。刪除 Auto Scaling 群組容量提供者之前，必須從所有服務的容量提供者策略中移除容量提供者。您可以使用 UpdateService API 或 Amazon ECS 主控台中的更新服務工作流程，從服務的容量提供者策略中移除容量提供者。使用強制新部署選項，以確保使用容量提供者提供的 Amazon EC2 執行個體容量的任何任務都已轉換為使用剩餘容量提供者的容量。

刪除叢集的容量提供者 (Amazon ECS 主控台)

1. 開啟主控台，網址為 https://console.aws.amazon.com/ecs/v2。
2. 在導覽窗格中，選擇叢集。
3. 在叢集頁面上，選擇叢集。
4. 在 Cluster : ***name*** (叢集：名稱) 頁面中，選擇 Infrastructure (基礎設施)、Auto Scaling 群組，然後選擇 Delete (刪除)。
5. 在確認方塊中，輸入 delete ***Auto Scaling ####***
6. 選擇 刪除 。

安全地停止在 EC2 執行個體上執行的 Amazon ECS 工作負載

受管執行個體耗盡有助於正常終止 Amazon EC2 執行個體。這可讓您的工作負載安全地停止，並重新安排至未終止的執行個體。基礎設施維護和更新會執行，而不必擔心工作負載中斷。透過使用受管執行個體耗盡，您可以簡化需要替換 Amazon EC2 執行個體的基礎設施管理工作流程，同時確保應用程式的彈性和可用性。

Amazon ECS 受管執行個體耗盡適用於 Auto Scaling 群組執行個體替換。根據執行個體重新整理和執行個體生命週期上限，客戶可以確保其容量符合最新的作業系統和安全性要求。

受管執行個體耗盡只能與 Amazon ECS 容量提供者搭配使用。您可以使用 Amazon ECS 主控台或 SDK，在建立或更新 Auto Scaling 群組容量提供者時 AWS CLI開啟受管執行個體耗盡。

Amazon ECS 受管執行個體耗盡會涵蓋下列事件。

- Auto Scaling 群組執行個體重新整理 - 使用執行個體重新整理來執行 Auto Scaling 群組中 Amazon EC2 執行個體的滾動取代，而不是分批手動執行。當您需要取代大量執行個體時，這很有用。執行個體重新整理是透過 Amazon EC2 主控台或 StartInstanceRefresh API 啟動。StartInstanceRefresh 如果您使用的是受管終止保護，請務必在呼叫時Replace為縮減保護選取 。
- 執行個體生命週期上限 - 您可以在取代 Auto Scaling 群組執行個體時定義生命週期上限。這有助於根據內部安全政策或合規來排程替換執行個體。

- Auto Scaling 群組縮減 - 根據擴展政策和排程擴展動作，Auto Scaling 群組支援執行個體的自動擴展。透過使用 Auto Scaling 群組做為 Amazon ECS 容量提供者，您可以在其中沒有任務執行時縮減 Auto Scaling 群組執行個體。
- Auto Scaling 群組運作狀態檢查 - Auto Scaling 群組支援許多運作狀態檢查，以管理終止運作狀態不佳的執行個體。
- AWS CloudFormation 堆疊更新 - 您可以在 AWS CloudFormation 堆疊中新增`UpdatePolicy`屬性，以在群組變更時執行滾動更新。
- Spot 容量重新平衡 - Auto Scaling 群組會嘗試主動取代根據 Amazon EC2 容量重新平衡通知具有較高中斷風險的 Spot 執行個體。Auto Scaling 群組會在啟動取代且運作狀態良好時終止舊執行個體。Amazon ECS 受管執行個體耗盡耗盡耗盡 Spot 執行個體的方式與耗盡非 Spot 執行個體的方式相同。
- Spot 中斷 - Spot 執行個體會在兩分鐘前通知後終止。Amazon ECS 受管執行個體耗盡會讓執行個體處於耗盡狀態以回應。

具有受管執行個體耗盡的 Amazon EC2 Auto Scaling 生命週期掛鉤

Auto Scaling 群組生命週期掛鉤可讓客戶建立由執行個體生命週期中特定事件觸發的解決方案，並在發生特定事件時執行自訂動作。Auto Scaling 群組最多允許 50 個勾點。可以存在多個終止關聯並平行執行，Auto Scaling 群組會等待所有關聯完成，然後再終止執行個體。

除了 Amazon ECS 受管勾點終止之外，您也可以設定自己的生命週期終止勾點。生命週期掛鉤具有`default action`，我們建議您`continue`將 設定為預設值，以確保其他掛鉤，例如 Amazon ECS 受管掛鉤，不會受到自訂掛鉤的任何錯誤影響。

如果您已設定 Auto Scaling 群組終止生命週期掛鉤，並啟用 Amazon ECS 受管執行個體耗盡，則會執行兩個生命週期掛鉤。不過，無法保證相對時間。生命週期掛鉤具有`default action`設定，可指定逾時過後要採取的動作。如果失敗，我們建議您使用 `continue`做為自訂掛鉤的預設結果。這可確保其他勾點，特別是 Amazon ECS 受管勾點，不會受到自訂生命週期勾點中任何錯誤的影響。的替代結果`abandon`會導致略過所有其他勾點，因此應避免。如需 Auto Scaling 群組生命週期關聯的詳細資訊，請參閱《Amazon EC2 Auto Scaling 使用者指南》中的 Amazon EC2 Auto Scaling 生命週期關聯。 Amazon EC2 Auto Scaling

任務和受管執行個體耗盡

Amazon ECS 受管執行個體耗盡會使用容器執行個體中現有的耗盡功能。容器執行個體耗盡功能會針對屬於 Amazon ECS 服務的複本任務執行替換和停止。處於 `PENDING`或 `RUNNING` 狀態的獨立任務，就像 叫用的任務`RunTask`一樣，不會受到影響。您必須等待這些項目完成或手動停止。容器執行個體

會保持 DRAINING 狀態，直到所有任務停止或超過 48 小時為止。協助程式任務是所有複本任務停止後最後停止的任務。

受管執行個體耗盡和受管終止保護

即使停用受管終止，受管執行個體耗盡仍然有效。如需受管終止保護的相關資訊，請參閱 控制執行個體 Amazon ECS 終止。

下表摘要說明受管終止和受管耗盡的不同組合的行為。

受管終止	受管耗盡	Outcome
已啟用	已啟用	Amazon ECS 可保護正在執行任務的 Amazon EC2 執行個體，避免縮減事件終止。任何正在進行終止的執行個體，例如未設定終止保護、收到 Spot 中斷，或被執行個體重新整理所強制的執行個體，都會正常耗盡。
已停用	已啟用	Amazon ECS 不會保護執行任務的

受管終止	受管耗盡	Outcome
		Amazon EC2 執行個體，使其不會縮減規模。不過，正在終止的任何執行個體都會正常耗盡。
已啟用	已停用	Amazon ECS 可保護正在執行任務的 Amazon EC2 執行個體，避免縮減事件終止。不過，如果執行個體未執行任何任務，Spot 中斷或強制執行個體重新整理仍會終止執行個體。Amazon ECS 不會對這些執行個體執行正常耗盡，並在停止後啟動替代服務任務。

受管終止	受管耗盡	Outcome
已停用	已停用	即使 Amazon EC2 執行個體正在執行 Amazon ECS 任務，也可以隨時縮減或終止。Amazon ECS 會在停止後啟動替代服務任務。

受管執行個體耗盡和 Spot 執行個體耗盡

使用 Spot 執行個體耗盡時，您可以在 Amazon ECS 代理程式`ECS_ENABLE_SPOT_INSTANCE_DRAINING`上設定環境變數，讓 Amazon ECS 將執行個體置於耗盡狀態，以回應兩分鐘的 Spot 中斷。Amazon ECS 受管執行個體耗盡有助於正常關閉因為許多原因而終止的 Amazon EC2 執行個體，而不只是 Spot 中斷。例如，您可以使用 Amazon EC2 Auto Scaling 容量重新平衡，在高中斷風險的情況下主動取代 Spot 執行個體，且受管執行個體耗盡會正常關閉要取代的 Spot 執行個體。當您使用受管執行個體耗盡時，您不需要分別啟用 Spot 執行個體耗盡，因此 `ECS_ENABLE_SPOT_INSTANCE_DRAINING` Auto Scaling 群組中的使用者資料是多餘的。如需 Spot 執行個體耗盡的詳細資訊，請參閱 Spot 執行個體。

受管執行個體耗盡如何與 EventBridge 搭配使用

Amazon ECS 受管執行個體耗盡事件會發佈至 Amazon EventBridge，Amazon ECS 會在您帳戶的預設匯流排中建立 EventBridge 受管規則，以支援受管執行個體耗盡。您可以將這些事件篩選到 Lambda、Amazon SNS 和 Amazon SQS 等 AWS 其他服務，以監控和疑難排解。

- 叫用生命週期掛鉤時，Amazon EC2 Auto Scaling 會將事件傳送至 EventBridge。
- Spot 中斷通知會發佈至 EventBridge。
- Amazon ECS 會產生錯誤訊息，您可以透過 Amazon ECS 主控台和 APIs 擷取。

- EventBridge 內建重試機制，做為暫時故障的緩解措施。

設定 Amazon ECS 容量提供者以安全地關閉執行個體

當您使用 Amazon ECS 主控台和 建立或更新 Auto Scaling 群組容量提供者時，可以開啟受管執行個體耗盡 AWS CLI。

> **Note**
>
> 當您建立容量提供者時，受管執行個體耗盡預設為開啟。

以下是使用 AWS CLI 建立已啟用受管執行個體耗盡的容量提供者，以及為叢集現有容量提供者啟用受管執行個體耗盡的範例。

建立已啟用受管執行個體耗盡的容量提供者

若要建立已啟用受管執行個體耗盡的容量提供者，請使用 `create-capacity-provider`命令。將 `managedDraining` 參數設為 `ENABLED`。

```
aws ecs create-capacity-provider \
--name capacity-provider \
--auto-scaling-group-provider '{
  "autoScalingGroupArn": "asg-arn",
  "managedScaling": {
    "status": "ENABLED",
    "targetCapacity": 100,
    "minimumScalingStepSize": 1,
    "maximumScalingStepSize": 1
  },
  "managedDraining": "ENABLED",
  "managedTerminationProtection": "ENABLED",
}'
```

回應：

```
{
    "capacityProvider": {
        "capacityProviderArn": "capacity-provider-arn",
        "name": "capacity-provider",
```

```
        "status": "ACTIVE",
        "autoScalingGroupProvider": {
            "autoScalingGroupArn": "asg-arn",
            "managedScaling": {
                "status": "ENABLED",
                "targetCapacity": 100,
                "minimumScalingStepSize": 1,
                "maximumScalingStepSize": 1
            },
            "managedTerminationProtection": "ENABLED"
            "managedDraining": "ENABLED"
        }
    }
}
```

啟用叢集現有容量提供者的受管執行個體耗盡

啟用叢集現有容量提供者的受管執行個體耗盡使用 update-capacity-provider命令。您會看到managedDraining目前顯示 DISABLED和 updateStatus UPDATE_IN_PROGRESS。

```
aws ecs update-capacity-provider \
--name cp-draining \
--auto-scaling-group-provider '{
  "managedDraining": "ENABLED"
}
```

回應：

```
{
    "capacityProvider": {
        "capacityProviderArn": "cp-draining-arn",
        "name": "cp-draining",
        "status": "ACTIVE",
        "autoScalingGroupProvider": {
            "autoScalingGroupArn": "asg-draining-arn",
            "managedScaling": {
                "status": "ENABLED",
                "targetCapacity": 100,
                "minimumScalingStepSize": 1,
                "maximumScalingStepSize": 1,
                "instanceWarmupPeriod": 300
            },
```

```
            "managedTerminationProtection": "DISABLED",
            "managedDraining": "DISABLED" // before update
        },
        "updateStatus": "UPDATE_IN_PROGRESS", // in progress and need describe again to
 find out the result
        "tags": [
        ]
    }
}
```

使用 describe-clusters命令並包含 ATTACHMENTS。受管執行個體耗盡附件status的 是 PRECREATED，而整體的 attachmentsStatus是 UPDATING。

```
aws ecs describe-clusters --clusters cluster-name --include ATTACHMENTS
```

回應：

```
{
    "clusters": [
        {
            ...

            "capacityProviders": [
                "cp-draining"
            ],
            "defaultCapacityProviderStrategy": [],
            "attachments": [
                # new precreated managed draining attachment
                {
                    "id": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
                    "type": "managed_draining",
                    "status": "PRECREATED",
                    "details": [
                        {
                            "name": "capacityProviderName",
                            "value": "cp-draining"
                        },
                        {
                            "name": "autoScalingLifecycleHookName",
                            "value": "ecs-managed-draining-termination-hook"
                        }
                    ]
```

```
                },

                ...

            ],
            "attachmentsStatus": "UPDATING"
        }
    ],
    "failures": []
}
```

更新完成時，請使用 describe-capacity-providers，您會看到 現在managedDraining是 ENABLED。

```
aws ecs describe-capacity-providers --capacity-providers cp-draining
```

回應：

```
{
    "capacityProviders": [
        {
            "capacityProviderArn": "cp-draining-arn",
            "name": "cp-draining",
            "status": "ACTIVE",
            "autoScalingGroupProvider": {
                "autoScalingGroupArn": "asg-draning-arn",
                "managedScaling": {
                    "status": "ENABLED",
                    "targetCapacity": 100,
                    "minimumScalingStepSize": 1,
                    "maximumScalingStepSize": 1,
                    "instanceWarmupPeriod": 300
                },
                "managedTerminationProtection": "DISABLED",
                "managedDraining": "ENABLED" // successfully update
            },
            "updateStatus": "UPDATE_COMPLETE",
            "tags": []
        }
    ]
}
```

Amazon ECS 受管執行個體耗盡疑難排解

您可能需要對受管執行個體耗盡的問題進行故障診斷。以下是您在使用時可能遇到的問題和解決方案範例。

使用自動擴展時，執行個體不會在使用超過執行個體生命週期上限後終止。

如果您的執行個體在使用自動擴展群組時，即使達到並超過執行個體存留期上限，也不會終止，這可能是因為它們受到保護，不會縮減。您可以關閉受管終止，並允許受管耗盡處理執行個體回收。

使用 建立 Amazon ECS 叢集自動擴展的資源 AWS Management Console

了解如何使用 建立叢集自動擴展的資源 AWS Management Console。當資源需要名稱時，我們會使用字首`ConsoleTutorial`來確保它們都有唯一的名稱，並使其易於找到。

主題

- 必要條件
- 步驟 1：建立 Amazon ECS 叢集。
- 步驟 2：註冊任務定義
- 步驟 3：執行任務
- 步驟 4：驗證
- 步驟 5：清除

必要條件

本教學課程假設已完成下列先決條件：

- 已完成「設定以使用 Amazon ECS。」中的步驟。
- 您的 AWS 使用者具有 IAM AmazonECS_FullAccess 政策範例中指定的必要許可。
- 已建立 Amazon ECS 容器執行個體 IAM 角色。如需詳細資訊，請參閱Amazon ECS 容器執行個體 IAM 角色。
- 已建立 Amazon ECS 服務連結 IAM 角色。如需詳細資訊，請參閱使用 Amazon ECS 的服務連結角色。
- 已建立 Auto Scaling 服務連結 IAM 角色。如需詳細資訊，請參閱《Amazon EC2 Auto Scaling 使用者指南》中的 Amazon EC2 Auto Scaling 的服務連結角色。

- 您已建立 VPC 和安全群組。如需詳細資訊，請參閱the section called “建立 Virtual Private Cloud”。

步驟 1：建立 Amazon ECS 叢集。

使用下列步驟來建立 Amazon ECS 叢集。

Amazon ECS 會在 AWS CloudFormation 堆疊中代表您建立 Amazon EC2 Auto Scaling 啟動範本和 Auto Scaling 群組。

1. 開啟主控台，網址為 https://console.aws.amazon.com/ecs/v2。
2. 在導覽窗格中，選擇叢集，然後選擇建立叢集。
3. 在叢集組態下的叢集名稱中，輸入 `ConsoleTutorial-cluster`。
4. 在基礎設施下，清除 AWS Fargate（無伺服器），然後選取 Amazon EC2 執行個體。接下來，設定作為容量提供者的 Auto Scaling 群組。
    - 在 Auto Scaling 群組 (ASG) 下。選取建立新 ASG，然後提供有關該群組的下列詳細資訊：
        - 在作業系統/架構中，選擇 Amazon Linux 2。
        - 在 EC2 執行個體類型中，選擇 t3.nano。
        - 對於容量，輸入 Auto Scaling 群組中要啟動的最小執行個體數和最大執行個體數。
5. (選用) 若要管理叢集標籤，請展開標籤，然後執行下列其中一項操作：

    [新增標籤] 選擇新增標籤，並執行下列動作：

    - 對於 Key (金鑰)，輸入金鑰名稱。
    - 對於 Value (值)，進入金鑰值。

    [移除標籤] 選擇標籤「金鑰」和「值」右側的移除。
6. 選擇建立。

步驟 2：註冊任務定義

您必須先註冊任務定義，才能在您的叢集上執行任務。任務定義是分在一組的容器清單。以下範例是一種簡單的任務定義，使用 Docker Hub 的 `amazonlinux` 映像，且正好處於睡眠的狀態。如需可用之任務定義參數的詳細資訊，請參閱「Amazon ECS 任務定義」。

1. 開啟主控台，網址為 https://console.aws.amazon.com/ecs/v2。

2. 在導覽窗格中，選擇 Task Definitions (任務定義)。
3. 選擇 Create new task definitio (建立新任務定義)、Create new task definition with JSON (使用 JSON 建立新的任務定義)。
4. 在 JSON 編輯器方塊中貼上下列內容。

```
{
    "family": "ConsoleTutorial-taskdef",
    "containerDefinitions": [
        {
            "name": "sleep",
            "image": "amazonlinux:2",
            "memory": 20,
            "essential": true,
            "command": [
                "sh",
                "-c",
                "sleep infinity"
            ]
        }
    ],
    "requiresCompatibilities": [
        "EC2"
    ]
}
```

5. 選擇建立。

步驟 3：執行任務

註冊帳戶的任務定義後，您就可以在叢集中執行任務。在本教學課程中，您會在 ConsoleTutorial-cluster 叢集中執行 ConsoleTutorial-taskdef 任務定義的五個執行個體。

1. 開啟主控台，網址為 https://console.aws.amazon.com/ecs/v2。
2. 在叢集頁面上，選擇 ConsoleTutorial-cluster。
3. 在任務下選擇執行新任務。
4. 在環境區段的運算選項下，選擇容量提供者策略。
5. 在部署組態下，針對應用程式類型選擇任務。
6. 從系列下拉式清單中選擇 ConsoleTutorial-taskdef。
7. 在所需的任務下，輸入 5。

8. 選擇建立。

步驟 4：驗證

在本教學課程的這個階段，您應該擁有一個正在執行五個任務的叢集，以及一個具有容量提供者的 Auto Scaling 群組。容量提供者已啟用 Amazon ECS 受管擴展。

我們可以透過檢視 CloudWatch 指標、Auto Scaling 群組設定，最後是 Amazon ECS 叢集任務計數，以確認所有項目都正常運作。

檢視叢集的 CloudWatch 指標

1. 透過 https://console.aws.amazon.com/cloudwatch/ 開啟 CloudWatch 主控台。
2. 在螢幕上方的導覽列上，選取區域。
3. 在導覽窗格的指標下方，選擇所有指標。
4. 在所有指標頁面的瀏覽索引標籤下，選擇 `AWS/ECS/ManagedScaling`。
5. 選擇 CapacityProviderName, ClusterName。
6. 選取與 `ConsoleTutorial-cluster` ClusterName 對應的核取方塊。
7. 在圖表化指標標籤下，將期間變更為 30 秒，統計值變更為最大值。

   圖表中顯示的值會顯示容量提供者的目標容量值。此值應從 `100` 開始，也就是我們設定的目標容量百分比。您應該會看到此值擴展到 `200`，這會觸發目標追蹤擴展政策的警示。此警示接著會觸發 Auto Scaling 群組水平擴展。

使用下列步驟來檢視 Auto Scaling 群組詳細資訊，以確認水平擴展動作已發生。

確認 Auto Scaling 群組已水平擴展

1. 在 https://console.aws.amazon.com/ec2/ 開啟 Amazon EC2 主控台。
2. 在螢幕上方的導覽列上，選取區域。
3. 在導覽窗格的 Auto Scaling 下，選擇 Auto Scaling Groups (Auto Scaling 群組)。
4. 選擇在此教學課程中建立的 `ConsoleTutorial-cluster` Auto Scaling 群組。檢視所需容量下的值，並檢視執行個體管理索引標籤下的執行個體，以確認您的群組是否橫向擴展至兩個執行個體。

使用下列步驟來檢視 Amazon ECS 叢集，以確認 Amazon EC2 執行個體已向叢集註冊，且您的任務已轉換為 RUNNING 狀態。

驗證 Auto Scaling 群組中的執行個體

1. 開啟主控台，網址為 https://console.aws.amazon.com/ecs/v2。
2. 在導覽窗格中，選擇叢集。
3. 在 Clusters (叢集) 頁面上，選擇 ConsoleTutorial-cluster 叢集。
4. 在任務標籤上，確認您看到五個處於 RUNNING 狀態的任務。

步驟 5：清除

完成此教學課程時，清除與其相關的資源，以免未使用的資源產生費用。不支援刪除容量提供者和任務定義，但是這些資源都沒有相關聯的成本。

清除教學課程資源

1. 開啟主控台，網址為 https://console.aws.amazon.com/ecs/v2。
2. 在導覽窗格中，選擇叢集。
3. 在叢集頁面上，選擇 ConsoleTutorial-cluster。
4. 在 ConsoleTutorial-cluster 頁面上，選擇任務索引標籤，然後依序選擇停止、全部停止。
5. 在導覽窗格中，選擇叢集。
6. 在叢集頁面上，選擇 ConsoleTutorial-cluster。
7. 在頁面的右上角，選擇刪除叢集。
8. 在確認方塊中，輸入 delete ConsoleTutorial-cluster，然後選擇刪除。
9. 使用以下步驟刪除 Auto Scaling 群組。
    a. 在 https://console.aws.amazon.com/ec2/ 開啟 Amazon EC2 主控台。
    b. 在螢幕上方的導覽列上，選取區域。
    c. 在導覽窗格的 Auto Scaling 下，選擇 Auto Scaling Groups (Auto Scaling 群組)。
    d. 選取 ConsoleTutorial-cluster Auto Scaling 群組，然後選擇動作。
    e. 在 Actions (動作) 選單中，選擇 Delete (刪除)。在確認方塊中，輸入 delete，然後選擇刪除。

Amazon ECS 的 Amazon EC2 容器執行個體

Amazon ECS 容器執行個體是執行 Amazon ECS 容器代理程式並註冊至叢集的 Amazon EC2 執行個體。當使用 EC2 啟動類型、外部啟動類型或 Auto Scaling 群組容量提供者來透過 Amazon ECS 執行任務時，您的任務會放置在作用中的容器執行個體。您必須負責容器執行個體的管理與維護。

雖然您可以建立自己的 Amazon EC2 執行個體 AMI，以符合在 Amazon ECS 上執行容器化工作負載所需的基本規格，但 Amazon ECS 最佳化 AMIs 是由 AWS 工程師在 Amazon ECS 上預先設定和測試。這是讓您快速上手並在 AWS 上執行容器最簡單的方法。

當您使用 主控台建立叢集時，Amazon ECS 會為您的執行個體建立啟動範本，並具有與所選作業系統相關聯的最新 AMI。

當您使用 AWS CloudFormation 建立叢集時，SSM 參數是 Auto Scaling 群組執行個體的 Amazon EC2 啟動範本的一部分。您可以設定 範本使用動態 Systems Manager 參數，以決定要部署的 Amazon ECS Optimized AMI。此參數可確保每次部署堆疊時，都會檢查是否有需要套用至 EC2 執行個體的可用更新。如需如何使用 Systems Manager 參數的範例，請參閱AWS CloudFormation 《 使用者指南》中的使用 Amazon ECS 最佳化 Amazon Linux 2023 AMI 建立 Amazon ECS 叢集。

- 擷取 Amazon ECS 最佳化 Linux AMI 中繼資料
- 擷取 Amazon ECS 最佳化 Bottlerocket AMI 中繼資料
- 擷取 Amazon ECS 最佳化 Windows AMI 中繼資料

您可以從與應用程式相容的執行個體類型中選擇。使用容量較大的執行個體，您可以同時啟動更多任務。使用較小的執行個體，您可以更精細地向外擴展，以節省成本。您不需要選擇適合叢集中所有應用程式的單一 Amazon EC2 執行個體類型。反之，您可以建立多個 Auto Scaling 群組，其中每個群組都有不同的執行個體類型。然後，您可以為每個群組建立 Amazon EC2 容量提供者。

使用以下指導方針來決定要使用的執行個體系列類型和執行個體類型：

- 排除不符合應用程式特定需求的執行個體類型或執行個體系列。例如，如果您的應用程式需要 GPU，您可以排除任何沒有 GPU 的執行個體類型。
- 考慮需求，包括網路輸送量和儲存。
- 考慮 CPU 和記憶體。一般而言，CPU 和記憶體必須足夠容納至少一個要執行任務的複本。

Spot 執行個體

相較於隨需執行個體，Spot 容量可大幅節省成本。Spot 容量是指超額容量，其價格明顯低於隨需或預留容量。Spot 容量適用於批次處理和機器學習工作負載，以及開發和預備環境。更一般來說，它適用於任何能容忍暫時停機的工作負載。

了解下列結果，因為 Spot 容量可能無法隨時可用。

- 在需求極高期間，Spot 容量可能無法使用。這可能會導致 Amazon EC2 Spot 執行個體啟動延遲。在這些事件中，Amazon ECS 服務會重試啟動任務，而 Amazon EC2 Auto Scaling 群組也會重試啟動執行個體，直到所需的容量可用為止。Amazon EC2 不會以隨需容量取代 Spot 容量。
- 當容量的整體需求增加時，Spot 執行個體和任務可能會終止，但只會出現兩分鐘警告。在傳送警告之後，任務應該在執行個體完全終止之前，視需要開始依序關機。這有助於將錯誤的可能性降至最低。如需正常關機的詳細資訊，請參閱使用 ECS 正常關機。

若要協助減少 Spot 容量短缺，請考慮下列建議：

- 使用多個區域和可用區域：Spot 容量會因區域和可用區域而異。您可以透過在多個區域和可用區域執行工作負載來提高 Spot 可用性。如果可能，請在您執行任務和執行個體的區域中，在所有可用區域中指定子網路。
- 使用多個 Amazon EC2 執行個體類型：當您將混合執行個體政策與 Amazon EC2 Auto Scaling 搭配使用時，會在您的 Auto Scaling 群組中啟動多個執行個體類型。這可確保在需要時能滿足 Spot 容量的請求。為了將可靠性最大化並將複雜性最小化，請在混合執行個體政策中使用 CPU 和記憶體數量大致相同的執行個體類型。這些執行個體可以來自不同世代，也可以是相同基礎執行個體類型的變體。請注意，它們可能擁有您可能不需要的其他功能。這類清單的範例可能包括 m4.large、m5.large、m5a.large、m5d.large、m5n.large、m5dn.large 和 m5ad.large。如需詳細資訊，請參閱「Amazon EC2 Auto Scaling 使用者指南」中的具備多個執行個體類型及購買選項的 Auto Scaling 群組。
- 使用容量最佳化的 Spot 配置策略：使用 Amazon EC2 Spot，您可以在容量和成本最佳化配置策略之間進行選擇。如果您在啟動新執行個體時選擇容量最佳化策略，Amazon EC2 Spot 會在所選可用區域中選取具有最高可用性的執行個體類型。這有助於降低執行個體在啟動後立即終止的可能性。

如需如何在容器執行個體上設定 Spot 終止通知的詳細資訊，請參閱：

- 設定 Amazon ECS Linux 容器執行個體接收 Spot 執行個體通知
- 設定 Amazon ECS Windows 容器執行個體以接收 Spot 執行個體通知

Amazon ECS 最佳化 Linux AMIs

Amazon ECS 提供 Amazon ECS 最佳化 AMIs，這些 AMI 已預先設定執行容器工作負載的需求和建議。建議您為 Amazon EC2 執行個體使用 Amazon ECS 最佳化 Amazon Linux 2023 AMI，除非您的應用程式需要 Amazon EC2 GPU 型執行個體、特定作業系統或在該 AMI 中尚不可用的 Docker 版本。如需有關 Amazon Linux 2 和 Amazon Linux 2023 執行個體的資訊，請參閱《Amazon Linux 2023 使用者指南》中的比較 Amazon Linux 2 和 Amazon Linux 2023。透過最新的 Amazon ECS 最佳化 AMI 啟動您的容器執行個體時，可確保您接收到最新的安全性更新和容器代理程式版本。如需執行個體啟動方式的詳細資訊，請參閱 啟動 Amazon ECS Linux 容器執行個體。

當您使用 主控台建立叢集時，Amazon ECS 會為您的執行個體建立啟動範本，並具有與所選作業系統相關聯的最新 AMI。

當您使用 AWS CloudFormation 建立叢集時，SSM 參數是 Auto Scaling 群組執行個體的 Amazon EC2 啟動範本的一部分。您可以設定 範本使用動態 Systems Manager 參數，以決定要部署的 Amazon ECS Optimized AMI。此參數可確保每次部署堆疊時，都會檢查是否有需要套用至 EC2 執行個體的可用更新。如需如何使用 Systems Manager 參數的範例，請參閱AWS CloudFormation 《 使用者指南》中的使用 Amazon ECS 最佳化 Amazon Linux 2023 AMI 建立 Amazon ECS 叢集。

如果您需要自訂 Amazon ECS 最佳化 AMI，請參閱 GitHub 上的 Amazon ECS Optimized AMI Build Recipes。

Amazon ECS 最佳化 AMI 的 Linux 變體使用 Amazon Linux 2 AMI 作為其基礎。此外，還提供 Amazon Linux 2 AMI 版本備註。如需詳細資訊，請參閱 Amazon Linux 2 版本備註。

我們建議您搭配 Linux 核心 5.10 使用 AMI，因為 Linux 核心 4.14 已於 2024 年 1 月 10 日end-of-life。

下列 Amazon ECS 最佳化 AMI 變體適用於具有 Amazon Linux 2023 作業系統的 Amazon EC2 執行個體。

作業系統	AMI	描述	儲存組態
Amazon Linux 2023	Amazon ECS 最佳化 Amazon Linux 2023 AMI	Amazon Linux 2023 是新一代的 Amazon Linux AWS。建議在大部分情況下為 Amazon ECS 工作負載啟動 Amazon EC2 執行個體。如需詳細資訊，	根據預設，Amazon ECS 最佳化 Amazon Linux 2023 AMI 附有單個 30 GiB 的根磁碟區。您可以在修改啟動階段的這個 30 GiB 根磁碟區，增加或

作業系統	AMI	描述	儲存組態
		請參閱 Amazon Linux 2023 使用者指南中的什麼是 Amazon Linux 2023。	減少您容器執行個體的可用儲存體。此儲存體用於作業系統和 Docker 映像及中繼資料。 Amazon ECS 最佳化 Amazon Linux 2023 AMI 的預設檔案系統為 `xfs`，並且 Docker 使用 `overlay2` 儲存驅動程式。如需詳細資訊，請參閱 Docker 文件中的使用 OverlayFS 儲存體驅動程式。

作業系統	AMI	描述	儲存組態
Amazon Linux 2023 (arm64)	Amazon ECS 最佳化 Amazon Linux 2023 (arm64) AMI	根據 Amazon Linux 2023，建議在啟動 Amazon EC2 執行個體時使用此 AMI，該執行個體採用 Arm 型 AWS Graviton/Graviton 2/Graviton 3/Graviton 4 處理器，適用於您的 Amazon ECS 工作負載。如需詳細資訊，請參閱《Amazon EC2 執行個體類型指南》中的 Amazon EC2 一般用途執行個體規格。Amazon EC2	根據預設，Amazon ECS 最佳化 Amazon Linux 2023 AMI 附有單個 30 GiB 的根磁碟區。您可以在修改啟動階段的這個 30 GiB 根磁碟區，增加或減少您容器執行個體的可用儲存體。此儲存體用於作業系統和 Docker 映像及中繼資料。 Amazon ECS 最佳化 Amazon Linux 2023 AMI 的預設檔案系統為 `xfs`，並且 Docker 使用 `overlay2` 儲存驅動程式。如需詳細資訊，請參閱 Docker 文件中的使用 OverlayFS 儲存體驅動程式。

作業系統	AMI	描述	儲存組態
Amazon Linux 2023 (Neuron)	Amazon ECS 最佳化 Amazon Linux 2023 AMI	根據 Amazon Linux 2023，此 AMIis適用於 Amazon EC2 Inf1, Trn1 或 Inf2 執行個體。它預先設定了 AWS Inferentia 和 AWS Trainium 驅動程式，以及適用於 Docker 的 AWS Neuron 執行期，這使得在 Amazon ECS 上執行機器學習推論工作負載變得更加容易。如需詳細資訊，請參閱AWS Neuron 機器學習工作負載的 Amazon ECS 任務定義。 Amazon ECS 最佳化 Amazon Linux 2023 (Neuron) AMI 不會隨附預先安裝的 AWS CLI 。	根據預設，Amazon ECS 最佳化 Amazon Linux 2023 AMI 附有單個 30 GiB 的根磁碟區。您可以在修改啟動階段的這個 30 GiB 根磁碟區，增加或減少您容器執行個體的可用儲存體。此儲存體用於作業系統和 Docker 映像及中繼資料。 Amazon ECS 最佳化 Amazon Linux 2023 AMI 的預設檔案系統為 `xfs`，並且 Docker 使用 `overlay2` 儲存驅動程式。如需詳細資訊，請參閱 Docker 文件中的使用 OverlayFS 儲存體驅動程式。

下列 Amazon ECS 最佳化 AMI 變體適用於具有 Amazon Linux 2 作業系統的 Amazon EC2 執行個體。

作業系統	AMI	描述	儲存組態
Amazon Linux 2	Amazon ECS 最佳化 Amazon Linux 2 核心 5.10 AMI	以 Amazon Linux 2 為基礎，在啟動 Amazon EC2 執行個體時使用此 AMI，而針對	依預設，基於 Amazon Linux 2 的 Amazon ECS 最佳化 AMI (Amazon ECS 最佳

作業系統	AMI	描述	儲存組態
		Amazon ECS 工作負載，則使用 Linux 核心 5.10 而不是核心 4.14。Amazon ECS 最佳化 Amazon Linux 2 核心 5.10 AMI 未預先安裝 AWS CLI 。	化 Amazon Linux 2 AMI、Amazon ECS 最佳化 Amazon Linux 2 (arm64) AMI 和 Amazon ECS GPU 最佳化 AMI) 隨附有一個 30-GiB 的根磁碟區。您可以在修改啟動階段的這個 30 GiB 根磁碟區，增加或減少您容器執行個體的可用儲存體。此儲存體用於作業系統和 Docker 映像及中繼資料。 Amazon ECS 最佳化 Amazon Linux 2 AMI 的預設檔案系統為 `xfs`，並且 Docker 使用 `overlay2` 儲存驅動程式。如需詳細資訊，請參閱 Docker 文件中的使用 OverlayFS 儲存體驅動程式。

作業系統	AMI	描述	儲存組態
Amazon Linux 2	Amazon ECS 最佳化 Amazon Linux 2 AMI	這適用於您的 Amazon ECS 工作負載。Amazon ECS 最佳化 Amazon Linux 2 AMI 不提供 AWS CLI 預先安裝。	依預設，基於 Amazon Linux 2 的 Amazon ECS 最佳化 AMI (Amazon ECS 最佳化 Amazon Linux 2 AMI、Amazon ECS 最佳化 Amazon Linux 2 (arm64) AMI 和 Amazon ECS GPU 最佳化 AMI) 隨附有一個 30-GiB 的根磁碟區。您可以在修改啟動階段的這個 30 GiB 根磁碟區，增加或減少您容器執行個體的可用儲存體。此儲存體用於作業系統和 Docker 映像及中繼資料。 Amazon ECS 最佳化 Amazon Linux 2 AMI 的預設檔案系統為 `xfs`，並且 Docker 使用 `overlay2` 儲存驅動程式。如需詳細資訊，請參閱 Docker 文件中的使用 OverlayFS 儲存體驅動程式。

作業系統	AMI	描述	儲存組態
Amazon Linux 2 (arm64)	Amazon ECS 最佳化 Amazon Linux 2 核心 5.10 (arm64) AMI	根據 Amazon Linux 2，此 AMI 適用於您的 Amazon EC2 執行個體，該執行個體採用 Arm 型 AWS Graviton/Graviton 2/Graviton 3/Graviton 4 處理器，而您想要針對 Amazon ECS 工作負載使用 Linux 核心 5.10 而非 Linux 核心 4.14。如需詳細資訊，請參閱《Amazon EC2 執行個體類型指南》中的 Amazon EC2 一般用途執行個體規格。 Amazon EC2 Amazon ECS 最佳化 Amazon Linux 2 (arm64) AMI 不會隨附 AWS CLI 預先安裝的 。	依預設，基於 Amazon Linux 2 的 Amazon ECS 最佳化 AMI (Amazon ECS 最佳化 Amazon Linux 2 AMI、Amazon ECS 最佳化 Amazon Linux 2 (arm64) AMI 和 Amazon ECS GPU 最佳化 AMI) 隨附有一個 30-GiB 的根磁碟區。您可以在修改啟動階段的這個 30 GiB 根磁碟區，增加或減少您容器執行個體的可用儲存體。此儲存體用於作業系統和 Docker 映像及中繼資料。 Amazon ECS 最佳化 Amazon Linux 2 AMI 的預設檔案系統為 `xfs`，並且 Docker 使用 `overlay2` 儲存驅動程式。如需詳細資訊，請參閱 Docker 文件中的使用 OverlayFS 儲存體驅動程式。

作業系統	AMI	描述	儲存組態
Amazon Linux 2 (arm64)	Amazon ECS 最佳化 Amazon Linux 2 (arm64) AMI	根據 Amazon Linux 2，此 AMI 適用於啟動 Amazon EC2 執行個體時，這些執行個體採用 Arm 型 AWS Graviton/Graviton 2/Graviton 3/Gravito n 4 處理器，適用於 Amazon ECS 工作負載。 Amazon ECS 最佳化 Amazon Linux 2 (arm64) AMI 不會隨附 AWS CLI 預先安裝的 。	依預設，基於 Amazon Linux 2 的 Amazon ECS 最佳化 AMI (Amazon ECS 最佳化 Amazon Linux 2 AMI、Amazon ECS 最佳化 Amazon Linux 2 (arm64) AMI 和 Amazon ECS GPU 最佳化 AMI) 隨附有一個 30-GiB 的根磁碟區。您可以在修改啟動階段的這個 30 GiB 根磁碟區，增加或減少您容器執行個體的可用儲存體。此儲存體用於作業系統和 Docker 映像及中繼資料。 Amazon ECS 最佳化 Amazon Linux 2 AMI 的預設檔案系統為 `xfs`，並且 Docker 使用 `overlay2` 儲存驅動程式。如需詳細資訊，請參閱 Docker 文件中的使用 OverlayFS 儲存體驅動程式。

作業系統	AMI	描述	儲存組態
Amazon Linux 2 (GPU)	Amazon ECS GPU 最佳化核心 5.10 AMI	根據 Amazon Linux 2，建議在為 Amazon ECS 工作負載使用 Linux 核心 5.10 啟動 Amazon EC2 GPU 型執行個體時使用這個 AMI。它配備有已預先設定的 NVIDIA 核心驅動程式和 Docker GPU 執行時間，可讓您在 Amazon ECS 上執行利用 GPU 的工作負載。如需詳細資訊，請參閱GPU 工作負載的 Amazon ECS 任務定義。	依預設，基於 Amazon Linux 2 的 Amazon ECS 最佳化 AMI (Amazon ECS 最佳化 Amazon Linux 2 AMI、Amazon ECS 最佳化 Amazon Linux 2 (arm64) AMI 和 Amazon ECS GPU 最佳化 AMI) 隨附有一個 30-GiB 的根磁碟區。您可以在修改啟動階段的這個 30 GiB 根磁碟區，增加或減少您容器執行個體的可用儲存體。此儲存體用於作業系統和 Docker 映像及中繼資料。 Amazon ECS 最佳化 Amazon Linux 2 AMI 的預設檔案系統為 `xfs`，並且 Docker 使用 `overlay2` 儲存驅動程式。如需詳細資訊，請參閱 Docker 文件中的使用 OverlayFS 儲存體驅動程式。

作業系統	AMI	描述	儲存組態
Amazon Linux 2 (GPU)	Amazon ECS GPU 最佳化 AMI	根據 Amazon Linux 2，建議在為 Amazon ECS 工作負載使用 Linux 核心 4.14 啟動 Amazon EC2 GPU 型執行個體時使用這個 AMI。它配備有已預先設定的 NVIDIA 核心驅動程式和 Docker GPU 執行時間，可讓您在 Amazon ECS 上執行利用 GPU 的工作負載。如需詳細資訊，請參閱GPU 工作負載的 Amazon ECS 任務定義。	依預設，基於 Amazon Linux 2 的 Amazon ECS 最佳化 AMI (Amazon ECS 最佳化 Amazon Linux 2 AMI、Amazon ECS 最佳化 Amazon Linux 2 (arm64) AMI 和 Amazon ECS GPU 最佳化 AMI) 隨附有一個 30-GiB 的根磁碟區。您可以在修改啟動階段的這個 30 GiB 根磁碟區，增加或減少您容器執行個體的可用儲存體。此儲存體用於作業系統和 Docker 映像及中繼資料。 Amazon ECS 最佳化 Amazon Linux 2 AMI 的預設檔案系統為 `xfs`，並且 Docker 使用 `overlay2` 儲存驅動程式。如需詳細資訊，請參閱 Docker 文件中的使用 OverlayFS 儲存體驅動程式。

作業系統	AMI	描述	儲存組態
Amazon Linux 2 (Neuron)	Amazon ECS 最佳化 Amazon Linux 2 (Neuron) 核心 5.10 AMI	以 Amazon Linux 2 為基礎，此 AMI 適用於 Amazon EC2 Inf1、Trn1 或 Inf2 執行個體。它預先設定 AWS 了 Inferentia 搭配 Linux 核心 5.10 和 AWS Trainium 驅動程式，以及適用於 Docker 的 AWS Neuron 執行期，這使得在 Amazon ECS 上執行機器學習推論工作負載變得更加輕鬆。如需詳細資訊，請參閱AWS Neuron 機器學習工作負載的 Amazon ECS 任務定義。Amazon ECS 最佳化 Amazon Linux 2 (Neuron) AMI 不會隨附 AWS CLI 預先安裝的 。	依預設，基於 Amazon Linux 2 的 Amazon ECS 最佳化 AMI (Amazon ECS 最佳化 Amazon Linux 2 AMI、Amazon ECS 最佳化 Amazon Linux 2 (arm64) AMI 和 Amazon ECS GPU 最佳化 AMI) 隨附有一個 30-GiB 的根磁碟區。您可以在修改啟動階段的這個 30 GiB 根磁碟區，增加或減少您容器執行個體的可用儲存體。此儲存體用於作業系統和 Docker 映像及中繼資料。 Amazon ECS 最佳化 Amazon Linux 2 AMI 的預設檔案系統為 `xfs`，並且 Docker 使用 `overlay2` 儲存驅動程式。如需詳細資訊，請參閱 Docker 文件中的使用 OverlayFS 儲存體驅動程式。

作業系統	AMI	描述	儲存組態
Amazon Linux 2 (Neuron)	Amazon ECS 最佳化 Amazon Linux 2 (Neuron) AMI	以 Amazon Linux 2 為基礎，此 AMI 適用於 Amazon EC2 Inf1、Trn1 或 Inf2 執行個體。它預先設定了 AWS Inferentia 和 AWS Trainium 驅動程式，以及適用於 Docker 的 AWS Neuron 執行期，這使得在 Amazon ECS 上執行機器學習推論工作負載變得更加容易。如需詳細資訊，請參閱AWS Neuron 機器學習工作負載的 Amazon ECS 任務定義。Amazon ECS 最佳化 Amazon Linux 2 (Neuron) AMI 不會隨附 AWS CLI 預先安裝的 。	依預設，基於 Amazon Linux 2 的 Amazon ECS 最佳化 AMI (Amazon ECS 最佳化 Amazon Linux 2 AMI、Amazon ECS 最佳化 Amazon Linux 2 (arm64) AMI 和 Amazon ECS GPU 最佳化 AMI) 隨附有一個 30-GiB 的根磁碟區。您可以在修改啟動階段的這個 30 GiB 根磁碟區，增加或減少您容器執行個體的可用儲存體。此儲存體用於作業系統和 Docker 映像及中繼資料。 Amazon ECS 最佳化 Amazon Linux 2 AMI 的預設檔案系統為 `xfs`，並且 Docker 使用 `overlay2` 儲存驅動程式。如需詳細資訊，請參閱 Docker 文件中的使用 OverlayFS 儲存體驅動程式。

Amazon ECS 在 GitHub 上提供了 Amazon ECS 最佳化 AMI 的 Linux 變體變更記錄。如需詳細資訊，請參閱變更記錄。

Amazon ECS 最佳化 AMI 的 Linux 變體使用 Amazon Linux 2 AMI 或 Amazon Linux 2023 AMI 作為其基礎。透過查詢 Systems Manager 參數存放區 API，可以擷取每個變體的 Amazon Linux 2 來源 AMI

名稱或 Amazon Linux 2023 AMI 名稱。如需詳細資訊，請參閱擷取 Amazon ECS 最佳化 Linux AMI 中繼資料。此外，還提供 Amazon Linux 2 AMI 版本備註。如需詳細資訊，請參閱 Amazon Linux 2 版本備註。此外，還提供 Amazon Linux 2023 版本備註。如需詳細資訊，請參閱 Amazon Linux 2023 版本備註。

下列頁面提供變更的其他資訊：

- 在 GitHub 上取得來源 AMI 發行說明
- Docker 文件中的 Docker 引擎版本備註
- NVIDIA 文件中的 NVIDIA 驅動程式文件
- GitHub 上的 Amazon ECS 代理程式變更日誌

  `ecs-init` 應用程式的原始程式碼，以及用來封裝代理程式的指令碼和組態，現在已成為代理程式儲存庫的一部分。對於 `ecs-init` 舊版本和封裝，請參閱 GitHub 上的 Amazon ecs-init 變更紀錄

將安全性更新套用至 Amazon ECS 最佳化 AMI

以 Amazon Linux 為基礎的 Amazon ECS 最佳化 AMIs 包含自訂的 版本cloud-init。 Cloud-init 是套件，用於在雲端運算環境中引導 Linux 映像，並在啟動執行個體時執行所需的動作。根據預設，所有在 2024 年 6 月 12 日之前發行的以 Amazon Linux 為基礎的 Amazon ECS 最佳化 AMIs，都會在執行個體啟動時套用所有「關鍵」和「重要」安全更新。

從 2024 年 6 月 12 日開始，基於 Amazon Linux 2 的 Amazon ECS 最佳化 AMIs 版本，預設行為將不再包含啟動時更新套件。反之，我們建議您更新至新的 Amazon ECS 最佳化 AMI，因為版本已推出。Amazon ECS 最佳化 AMIs 會在有可用的安全更新或基本 AMI 變更時發行。這將確保您收到最新的套件版本和安全性更新，而且套件版本在執行個體啟動時不會變動。如需擷取最新 Amazon ECS 最佳化 AMI 的詳細資訊，請參閱 擷取 Amazon ECS 最佳化 Linux AMI 中繼資料。

我們建議您將環境自動化，以便在新 AMI 可用時更新。如需可用選項的相關資訊，請參閱 Amazon ECS 可讓 EC2 容量管理更輕鬆，並搭配受管執行個體耗盡。

若要繼續在 AMI 版本上手動套用「關鍵」和「重要」安全更新，您可以在 Amazon EC2 執行個體上執行下列命令。

```
yum update --security
```

如果您想要在啟動時重新啟用安全性更新，您可以在啟動 Amazon EC2 執行個體時，將以下行新增至 Cloud-init 使用者資料`#cloud-config`區段。如需詳細資訊，請參閱《Amazon Linux 使用者指南》中的在 Amazon Linux 2 上使用 cloud-init。

```
#cloud-config
repo_upgrade: security
```

擷取 Amazon ECS 最佳化 Linux AMI 中繼資料

您可以以程式設計方式擷取 Amazon ECS 最佳化 AMI 中繼資料。中繼資料包含 AMI 名稱、Amazon ECS 容器代理程式版本，以及包含 Docker 版本的 Amazon ECS 執行期版本。

當您使用 主控台建立叢集時，Amazon ECS 會為您的執行個體建立啟動範本，並具有與所選作業系統相關聯的最新 AMI。

當您使用 AWS CloudFormation 建立叢集時，SSM 參數是 Auto Scaling 群組執行個體的 Amazon EC2 啟動範本的一部分。您可以設定 範本以使用動態 Systems Manager 參數來決定要部署的 Amazon ECS Optimized AMI。此參數可確保每次部署堆疊時，都會檢查是否有需要套用至 EC2 執行個體的可用更新。如需如何使用 Systems Manager 參數的範例，請參閱AWS CloudFormation 《 使用者指南》中的使用 Amazon ECS 最佳化 Amazon Linux 2023 AMI 建立 Amazon ECS 叢集。

透過查詢 Systems Manager 參數存放區 API，可以程式設計方式擷取 Amazon ECS 最佳化 AMI 的每個變體的 AMI ID、映像名稱、作業系統、容器代理程式版本、來源映像名稱以及執行時間版本。如需有關 Systems Manager 參數存放區 API 的詳細資訊，請參閱 GetParameters 和 GetParametersByPath。

> **Note**
>
> 您的管理使用者必須擁有以下 IAM 許可，才能擷取 Amazon ECS 最佳化 AMI 中繼資料。已將這些權限新增至 `AmazonECS_FullAccess` IAM 政策。
>
> - ssm:GetParameters
> - ssm:GetParameter
> - ssm:GetParametersByPath

Systems Manager 參數存放區參數格式。

以下是每個 Amazon ECS 最佳化 AMI 變體的參數名稱格式。

Linux Amazon ECS 最佳化 AMI

- Amazon Linux 2023 AMI 中繼資料：

```
/aws/service/ecs/optimized-ami/amazon-linux-2023/<version>
```

- Amazon Linux 2023 (arm64) AMI 中繼資料：

```
/aws/service/ecs/optimized-ami/amazon-linux-2023/arm64/<version>
```

- Amazon Linux 2023 (Neuron) AMI 中繼資料：

```
/aws/service/ecs/optimized-ami/amazon-linux-2023/neuron/<version>
```

- Amazon Linux 2 AMI 中繼資料：

```
/aws/service/ecs/optimized-ami/amazon-linux-2/<version>
```

- Amazon Linux 2 核心 5.10 AMI 中繼資料：

```
/aws/service/ecs/optimized-ami/amazon-linux-2/kernel-5.10/<version>
```

- Amazon Linux 2 (arm64) AMI 中繼資料：

```
/aws/service/ecs/optimized-ami/amazon-linux-2/arm64/<version>
```

- Amazon Linux 2 核心 5.10 (arm64) AMI 中繼資料：

```
/aws/service/ecs/optimized-ami/amazon-linux-2/kernel-5.10/arm64/<version>
```

- Amazon ECS GPU 最佳化核心 5.10 AMI 中繼資料：

```
/aws/service/ecs/optimized-ami/amazon-linux-2/kernel-5.10/gpu/<version>
```

- Amazon Linux 2 (GPU) AMI 中繼資料：

```
/aws/service/ecs/optimized-ami/amazon-linux-2/gpu/<version>
```

- Amazon ECS 最佳化 Amazon Linux 2 (Neuron) 核心 5.10 AMI 中繼資料：

```
/aws/service/ecs/optimized-ami/amazon-linux-2/kernel-5.10/inf/<version>
```

- Amazon Linux 2 (Neuron) AMI 中繼資料：

```
/aws/service/ecs/optimized-ami/amazon-linux-2/inf/<version>
```

下列參數名稱格式會使用 子參數 擷取最新建議的 Amazon ECS 最佳化 Amazon Linux 2 AMI 影像 IDimage_id。

```
/aws/service/ecs/optimized-ami/amazon-linux-2/recommended/image_id
```

以下參數名稱格式透過指定 AMI 名稱來擷取特定 Amazon ECS 最佳化 AMI 版本的中繼資料。

- Amazon ECS 最佳化 Amazon Linux 2 AMI 中繼資料：

```
/aws/service/ecs/optimized-ami/amazon-linux-2/amzn2-ami-ecs-hvm-2.0.20181112-x86_64-ebs
```

> **Note**
>
> 可擷取 Amazon ECS 最佳化 Amazon Linux 2 AMI 的所有版本。只能擷取 Amazon ECS 最佳化 AMI 版本 amzn-ami-2017.09.l-amazon-ecs-optimized (Linux) 及更新版本。

範例

下列範例顯示您可擷取每個 Amazon ECS 最佳化 AMI 變體中繼資料的方法。

擷取最新建議的 Amazon ECS 最佳化 AMI 中繼資料

您可以使用 AWS CLI 搭配下列 AWS CLI 命令來擷取最新的 Amazon ECS 最佳化 AMI。

Linux Amazon ECS 最佳化 AMI

- 對於 Amazon ECS 最佳化 Amazon Linux 2023 AMI：

```
aws ssm get-parameters --names /aws/service/ecs/optimized-ami/amazon-linux-2023/recommended --region us-east-1
```

- 對於 Amazon ECS 最佳化 Amazon Linux 2023 (arm64) AMI：

```
aws ssm get-parameters --names /aws/service/ecs/optimized-ami/amazon-linux-2023/
arm64/recommended --region us-east-1
```

- 對於 Amazon ECS 最佳化 Amazon Linux 2023 (Neuron) AMIs：

```
aws ssm get-parameters --names /aws/service/ecs/optimized-ami/amazon-linux-2023/
neuron/recommended --region us-east-1
```

- 對於 Amazon ECS 最佳化 Amazon Linux 2 核心 5.10 AMI：

```
aws ssm get-parameters --names /aws/service/ecs/optimized-ami/amazon-linux-2/
kernel-5.10/recommended --region us-east-1
```

- 對於 Amazon ECS 最佳化 Amazon Linux 2 AMI：

```
aws ssm get-parameters --names /aws/service/ecs/optimized-ami/amazon-linux-2/
recommended --region us-east-1
```

- 對於 Amazon ECS 最佳化 Amazon Linux 2 核心 5.10 (arm64) AMI：

```
aws ssm get-parameters --names /aws/service/ecs/optimized-ami/amazon-linux-2/
kernel-5.10/arm64/recommended --region us-east-1
```

- 對於 Amazon ECS 最佳化 Amazon Linux 2 (arm64) AMI：

```
aws ssm get-parameters --names /aws/service/ecs/optimized-ami/amazon-linux-2/arm64/
recommended --region us-east-1
```

- 對於 Amazon ECS GPU 最佳化核心 5.10 AMIs：

```
aws ssm get-parameters --names /aws/service/ecs/optimized-ami/amazon-linux-2/
kernel-5.10/gpu/recommended --region us-east-1
```

- 對於 Amazon ECS GPU 最佳化 AMI：

```
aws ssm get-parameters --names /aws/service/ecs/optimized-ami/amazon-linux-2/gpu/
recommended --region us-east-1
```

- 對於 Amazon ECS 最佳化 Amazon Linux 2 (Neuron) 核心 5.10 AMIs：

```
aws ssm get-parameters --names /aws/service/ecs/optimized-ami/amazon-linux-2/
kernel-5.10/inf/recommended --region us-east-1
```

- 對於 Amazon ECS 最佳化 Amazon Linux 2 (Neuron) AMI：

```
aws ssm get-parameters --names /aws/service/ecs/optimized-ami/amazon-linux-2/inf/
recommended --region us-east-1
```

擷取最新建議的 Amazon ECS 最佳化 Amazon Linux 2023 AMI 的映像 ID

您可以使用子參數 `image_id` 來擷取最新建議的 Amazon ECS 最佳化 Amazon Linux 2023 AMI ID 的映像 ID。

```
aws ssm get-parameters --names /aws/service/ecs/optimized-ami/amazon-
linux-2023/recommended/image_id --region us-east-1
```

若只要擷取 `image_id` 值，您可以查詢特定的參數值，例如：

```
aws ssm get-parameters --names /aws/service/ecs/optimized-ami/amazon-linux-2023/
recommended/image_id --region us-east-1 --query "Parameters[0].Value"
```

擷取特定 Amazon ECS 最佳化 Amazon Linux 2 AMI 版本的中繼資料

使用 AWS CLI 搭配下列 AWS CLI 命令，擷取特定 Amazon ECS 最佳化 Amazon Linux AMI 版本的中繼資料。將 AMI 名稱替換為要擷取的 Amazon ECS 最佳化 Amazon Linux AMI 的名稱。

```
aws ssm get-parameters --names /aws/service/ecs/optimized-ami/amazon-linux-2/amzn2-ami-
ecs-hvm-2.0.20200928-x86_64-ebs --region us-east-1
```

使用 Systems Manager GetParametersByPath API 擷取 Amazon ECS 最佳化 Amazon Linux 2 核心 5.10 AMI 中繼資料

使用 Systems Manager GetParametersByPath API 擷取 Amazon ECS 最佳化的 Amazon Linux 2 AMI 中繼資料，使用 AWS CLI 搭配下列命令。

```
aws ssm get-parameters-by-path --path /aws/service/ecs/optimized-ami/amazon-linux-2/
kernel-5.10/ --region us-east-1
```

擷取最新建議的 Amazon ECS 最佳化 Amazon Linux 2 核心 5.10 AMI 的影像 ID

您可以使用 子參數 來擷取最新建議的 Amazon ECS 最佳化 Amazon Linux 2 核心 5.10 AMI ID 的影像 ID。 `image_id`

```
aws ssm get-parameters --names /aws/service/ecs/optimized-ami/amazon-linux-2/
kernel-5.10/recommended/image_id --region us-east-1
```

若只要擷取 `image_id` 值，您可以查詢特定的參數值，例如：

```
aws ssm get-parameters --names /aws/service/ecs/optimized-ami/amazon-linux-2/
recommended/image_id --region us-east-1 --query "Parameters[0].Value"
```

在 AWS CloudFormation 範本中使用最新建議的 Amazon ECS 最佳化 AMI

透過參考 Systems Manager 參數存放區名稱，可以在 AWS CloudFormation 範本中參考最新建議的 Amazon ECS 最佳化 AMI。

Linux 範例

```
Parameters:kernel-5.10
  LatestECSOptimizedAMI:
    Description: AMI ID
    Type: AWS::SSM::Parameter::Value<AWS::EC2::Image::Id>
    Default: /aws/service/ecs/optimized-ami/amazon-linux-2/kernel-5.10/recommended/
image_id
```

Amazon ECS 最佳化 Linux AMI 建置指令碼

Amazon ECS 已採用開放原始碼的形式提供建置指令碼，可用於建置 Amazon ECS 最佳化 AMI 的 Linux 變體。這些建置指令碼現可於 GitHub 取得。如需詳細資訊，請參閱 GitHub 上的 amazon-ecs-ami。

如果您需要自訂 Amazon ECS 最佳化 AMI ，請參閱 GitHub 上的 Amazon ECS Optimized AMI Build Recipies。

此建置指令碼儲存庫包含 HashiCorp Packer 範本及用於產生 Amazon ECS 最佳化 AMI 每個 Linux 變數的建置指令碼。這些指令碼是 Amazon ECS 最佳化 AMI 建置的真實來源，因此您可遵循 GitHub 儲存庫來監控我們 AMI 的變更。例如，您可能希望自己的 AMI 使用的 Docker 版本，與 Amazon ECS 團隊用於官方 AMI 的版本相同。

如需詳細資訊，請參閱 GitHub 上的 Amazon ECS AMI 儲存庫，網址為 aws/amazon-ecs-ami。

建置 Amazon ECS 最佳化 Linux AMI

1. 複製 aws/amazon-ecs-ami GitHub 儲存庫。

```
git clone https://github.com/aws/amazon-ecs-ami.git
```

2. 新增環境變數，供 AWS 區域在建立 AMI 時使用。將 us-west-2 值替換為要使用的區域。

```
export REGION=us-west-2
```

3. 提供 Makefile 以用於建置 AMI。在已複製儲存庫的根目錄中，使用以下其中一項與您要建置的 Amazon ECS 最佳化 AMI 的 Linux 變數相對應的命令。

   - Amazon ECS 最佳化 Amazon Linux 2 AMI

     ```
     make al2
     ```

   - Amazon ECS 最佳化 Amazon Linux 2 (arm64) AMI

     ```
     make al2arm
     ```

   - Amazon ECS GPU 最佳化 AMI

     ```
     make al2gpu
     ```

   - Amazon ECS 最佳化 Amazon Linux 2 (Neuron) AMI

     ```
     make al2inf
     ```

   - Amazon ECS 最佳化 Amazon Linux 2023 AMI

     ```
     make al2023
     ```

   - Amazon ECS 最佳化 Amazon Linux 2023 (arm64) AMI

     ```
     make al2023arm
     ```

   - Amazon ECS 最佳化 Amazon Linux 2023 (Neuron) AMI

     ```
     make al2023neu
     ```

訂閱 Amazon ECS 最佳化 Linux AMI 更新通知

AWS 為與 Amazon Linux 2023 和 Amazon Linux 2 AMI 相關的通知提供 Amazon SNS AMIs。 ARNs 發佈新的 AL2023 或 Amazon Linux 2 AMI 時，主題會傳送更新通知。雖然這些主題並非專屬於 Amazon ECS 最佳化 Linux AMIs，但由於 Amazon ECS 最佳化 Linux AMIs 遵循相同的發行排程，因此您可以使用這些通知來指示何時更新新的 Amazon ECS 最佳化 Linux AMIs。如需訂閱 Linux AMI 通知的詳細資訊，請參閱《Amazon Linux 2023 使用者指南》中的接收新更新的通知，以及》Amazon Linux 2 使用者指南》中的 AL2 AMI 版本通知。

> **Note**
>
> 您的使用者或連接至使用者的角色必須具有 IAM `sns::subscribe` 許可，才能訂閱 Amazon SNS 主題。

Amazon ECS 最佳化 Bottlerocket AMI

Bottlerocket 是 建置的 Linux型開放原始碼作業系統，用於 AWS 在虛擬機器或裸機主機上執行容器。Amazon ECS 最佳化 Bottlerocket AMI 非常安全，只包括執行容器所需的最少數量套件。這可改善資源使用率、減少安全性受攻擊面，並有助於降低管理負荷。Bottlerocket AMI 也與 Amazon ECS 整合，有助於減少更新叢集中容器執行個體所涉及的營運開銷。

Bottlerocket 在以下方面不同於 Amazon Linux：

- Bottlerocket 不包括套件管理工具，其軟體僅可作為容器執行。Bottlerocket 的更新一律套用，而且可以在單一步驟中復原，以減少發生更新錯誤的可能性。
- 管理 Bottlerocket 主機的主要機制是使用容器排程器。與 Amazon Linux 不同的是，登入個別 Bottlerocket 執行個體的用意只是為了進階偵錯和故障排除目的，而非經常執行的操作。

如需有關 Bottlerocket 的詳細資訊，請參閱 GitHub 上的文件和版本。

核心 6.1 和核心 5.10 的 Amazon ECS 最佳化 Bottlerocket AMI 有變體。

下列變體使用核心 6.1：

- `aws-ecs-2`
- `aws-ecs-2-nvidia`

下列變體使用核心 5.10：

- `aws-ecs-1`
- `aws-ecs-1-nvidia`

  如需有關 `aws-ecs-1-nvidia` 變體的詳細資訊，請參閱宣布 NVIDIA GPU 支援 Amazon ECS 上的 Bottlerocket。

考量事項

搭配使用 Bottlerocket AMI 與 Amazon ECS 時，應考慮以下事項。

- Bottlerocket 支援具有 `x86_64` 與 `arm64` 處理器的 Amazon EC2 執行個體。Bottlerocket AMI 不建議與採用 Inferentia 晶片的 Amazon EC2 執行個體搭配使用。
- Bottlerocket 映像不包含 SSH 伺服器或 Shell。不過，您可以使用頻外管理工具取得 SSH 管理員存取權並執行啟動引導。如需詳細資訊，請參閱 GitHub 上 bottlerocket README.md 中的這些章節：
  - 探勘
  - 管理員容器
- 在預設情況下，Bottlerocket 具有已啟用的控制容器。此容器會執行 AWS Systems Manager 代理程式，可以讓您在 Amazon EC2 Bottlerocket 執行個體上執行命令或啟動 Shell 工作階段。如需詳細資訊，請參閱《AWS Systems Manager 使用者指南》中的設定工作階段管理員。
- Bottlerocket 已針對容器工作負載進行了最佳化，並著重於安全性。Bottlerocket 不包括套件管理工具，且不可變。如需有關安全性功能和指引的詳細資訊，請參閱 GitHub 上的安全性功能和安全性指引。
- Bottlerocket AMI 版本 `1.1.0` 或更新版本支援 `awsvpc` 網路模式。
- Bottlerocket AMI 版本 `1.15.0` 或更新版本支援任務定義中的 App Mesh。
- Bottlerocket AMI 版本 `1.19.0` 或更新版本支援`initProcessEnabled`任務定義參數。
- Bottlerocket AMI 也不支援下列服務和功能：
  - ECS Anywhere
  - Service Connect
  - Amazon EFS 處於加密模式
  - `awsvpc` 網路模式下的 Amazon EFS
  - Elastic Inference 加速器

擷取 Amazon ECS 最佳化 Bottlerocket AMI 中繼資料

您可以透過查詢 AWS Systems Manager 參數存放區 API 來擷取 Amazon ECS 最佳化 AMIs的 Amazon Machine Image (AMI) ID。若使用此參數，您無需手動查詢 Amazon ECS 最佳化 AMI ID。如需 Systems Manager 參數存放區 API 的詳細資訊，請參閱 GetParameter。您使用的使用者必須擁有 `ssm:GetParameter` IAM 許可，才能擷取 Amazon ECS 最佳化 AMI 中繼資料。

`aws-ecs-2` Bottlerocket AMI 變體

您可以透過 AWS 區域 和使用 AWS CLI 或 的架構來擷取最新的穩定 `aws-ecs-2` Bottlerocket AMI 變體 AWS Management Console。

- AWS CLI – 您可以使用子參數 ，透過下列 AWS CLI 命令擷取最新建議的 Amazon ECS 最佳化 Bottlerocket AMI 影像 IDimage_id。將 *region* 替換為您需要 AMI ID 的區域代碼。如需支援 AWS 區域的相關資訊，請參閱 GitHub 上的尋找 AMI。若要擷取最新版本以外的版本，請以版本號碼取代 `latest`。
  - 對於 64 位元 (x86_64) 架構：

    ```
    aws ssm get-parameter --region us-east-2 --name "/aws/service/bottlerocket/aws-ecs-2/x86_64/latest/image_id" --query Parameter.Value --output text
    ```

  - 對於 64 位元 Arm (arm64) 架構：

    ```
    aws ssm get-parameter --region us-east-2 --name "/aws/service/bottlerocket/aws-ecs-2/arm64/latest/image_id" --query Parameter.Value --output text
    ```

- AWS Management Console：您可以使用 AWS Management Console中的 URL 查詢建議的 Amazon ECS 最佳化 AMI ID。此 URL 會以參數的 ID 值開啟 Amazon EC2 Systems Manager 主控台。在下列 URL 中，將 *region* 替換為您需要 AMI ID 的區域代碼。如需 支援的資訊 AWS 區域，請參閱在 GitHub 上尋找 AMI。 GitHub
  - 對於 64 位元 (x86_64) 架構：

    ```
    https://console.aws.amazon.com/systems-manager/parameters/aws/service/bottlerocket/aws-ecs-2/x86_64/latest/image_id/description?region=region#
    ```

  - 對於 64 位元 Arm (arm64) 架構：

    ```
    https://console.aws.amazon.com/systems-manager/parameters/aws/service/bottlerocket/aws-ecs-2/arm64/latest/image_id/description?region=region#
    ```

aws-ecs-2-nvidia Bottlerocket AMI 變體

您可以使用 AWS CLI 或 ，依區域和架構擷取最新的穩定 aws-ecs-2-nvdia Bottlerocket AMI 變體 AWS Management Console。

- AWS CLI – 您可以使用 子參數 ，透過下列 AWS CLI 命令擷取最新建議的 Amazon ECS 最佳化 Bottlerocket AMI 影像 IDimage_id。將 *region* 替換為您需要 AMI ID 的區域代碼。如需支援 AWS 區域的相關資訊，請參閱 GitHub 上的尋找 AMI。若要擷取最新版本以外的版本，請以版本號碼取代 latest。
  - 對於 64 位元 (x86_64) 架構：

```
aws ssm get-parameter --region us-east-1 --name "/aws/service/bottlerocket/aws-
ecs-2-nvidia/x86_64/latest/image_id" --query Parameter.Value --output text
```

  - 對於 64 位元 Arm (arm64) 架構：

```
aws ssm get-parameter --region us-east-1 --name "/aws/service/bottlerocket/aws-
ecs-2-nvidia/arm64/latest/image_id" --query Parameter.Value --output text
```

- AWS Management Console：您可以使用 AWS Management Console中的 URL 查詢建議的 Amazon ECS 最佳化 AMI ID。此 URL 會以參數的 ID 值開啟 Amazon EC2 Systems Manager 主控台。在下列 URL 中，將 *region* 替換為您需要 AMI ID 的區域代碼。如需 支援的資訊 AWS 區域，請參閱在 GitHub 上尋找 AMI。 GitHub
  - 對於 64 位元 (x86_64) 架構：

```
https://regionconsole.aws.amazon.com/systems-manager/parameters/aws/service/
bottlerocket/aws-ecs-2-nvidia/x86_64/latest/image_id/description?region=region#
```

  - 對於 64 位元 Arm (arm64) 架構：

```
https://regionconsole.aws.amazon.com/systems-manager/parameters/aws/service/
bottlerocket/aws-ecs-2-nvidia/arm64/latest/image_id/description?region=region#
```

aws-ecs-1 Bottlerocket AMI 變體

您可以使用 AWS CLI 或 擷取最新的穩定 aws-ecs-1 Bottlerocket AMI 變體 AWS 區域 ，以及架構 AWS Management Console。

- AWS CLI – 您可以使用 子參數 ，透過下列 AWS CLI 命令擷取最新建議的 Amazon ECS 最佳化 Bottlerocket AMI 影像 IDimage_id。將 *region* 替換為您需要 AMI ID 的區域代碼。如需支援 AWS 區域的相關資訊，請參閱 GitHub 上的尋找 AMI。若要擷取最新版本以外的版本，請以版本號碼取代 latest。
  - 對於 64 位元 (x86_64) 架構：

```
aws ssm get-parameter --region us-east-1 --name "/aws/service/bottlerocket/aws-ecs-1/x86_64/latest/image_id" --query Parameter.Value --output text
```

  - 對於 64 位元 Arm (arm64) 架構：

```
aws ssm get-parameter --region us-east-1 --name "/aws/service/bottlerocket/aws-ecs-1/arm64/latest/image_id" --query Parameter.Value --output text
```

- AWS Management Console：您可以使用 AWS Management Console中的 URL 查詢建議的 Amazon ECS 最佳化 AMI ID。此 URL 會以參數的 ID 值開啟 Amazon EC2 Systems Manager 主控台。在下列 URL 中，將 *region* 替換為您需要 AMI ID 的區域代碼。如需 支援的相關資訊 AWS 區域，請參閱在 GitHub 上尋找 AMI。 GitHub
  - 對於 64 位元 (x86_64) 架構：

```
https://region.console.aws.amazon.com/systems-manager/parameters/aws/service/bottlerocket/aws-ecs-1/x86_64/latest/image_id/description
```

  - 對於 64 位元 Arm (arm64) 架構：

```
https://region.console.aws.amazon.com/systems-manager/parameters/aws/service/bottlerocket/aws-ecs-1/arm64/latest/image_id/description
```

aws-ecs-1-nvidia Bottlerocket AMI 變體

您可以使用 AWS CLI 或 ，依區域和架構擷取最新的穩定 aws-ecs-1-nvdia Bottlerocket AMI 變體 AWS Management Console。

- AWS CLI – 您可以使用子參數 ，透過下列 AWS CLI 命令擷取最新建議的 Amazon ECS 最佳化 Bottlerocket AMI 影像 IDimage_id。將 *region* 替換為您需要 AMI ID 的區域代碼。如需支援 AWS 區域的相關資訊，請參閱 GitHub 上的尋找 AMI。若要擷取最新版本以外的版本，請以版本號碼取代 latest。
  - 對於 64 位元 (x86_64) 架構：

```
aws ssm get-parameter --region us-east-1 --name "/aws/service/bottlerocket/aws-ecs-1-nvidia/x86_64/latest/image_id" --query Parameter.Value --output text
```

- 對於 64 位元 Arm (arm64) 架構：

```
aws ssm get-parameter --region us-east-1 --name "/aws/service/bottlerocket/aws-ecs-1-nvidia/arm64/latest/image_id" --query Parameter.Value --output text
```

- AWS Management Console：您可以使用 AWS Management Console中的 URL 查詢建議的 Amazon ECS 最佳化 AMI ID。此 URL 會以參數的 ID 值開啟 Amazon EC2 Systems Manager 主控台。在下列 URL 中，將 *region* 替換為您需要 AMI ID 的區域代碼。如需 支援的資訊 AWS 區域，請參閱在 GitHub 上尋找 AMI。 GitHub
  - 對於 64 位元 (x86_64) 架構：

```
https://console.aws.amazon.com/systems-manager/parameters/aws/service/bottlerocket/aws-ecs-1-nvidia/x86_64/latest/image_id/description?region=region#
```

  - 對於 64 位元 Arm (arm64) 架構：

```
https://console.aws.amazon.com/systems-manager/parameters/aws/service/bottlerocket/aws-ecs-1-nvidia/arm64/latest/image_id/description?region=region#
```

後續步驟

如需如何在 Amazon ECS 上開始使用Bottlerocket作業系統的詳細教學，請參閱部落格網站上的使用 Bottlerocket AMI 搭配 GitHub 上的 Amazon ECS 和 BottlerocketAmazon ECS AWS 。 GitHub

如需如何啟動 Bottlerocket 執行個體的詳細資訊，請參閱 啟動 Amazon ECS 的Bottlerocket執行個體

啟動 Amazon ECS 的Bottlerocket執行個體

您可以啟動Bottlerocket執行個體，以便執行容器工作負載。

您可以使用 AWS CLI 來啟動Bottlerocket執行個體。

1. 建立稱為 userdata.toml 的檔案。此檔案會用於執行個體使用者資料。以您的叢集名稱取代 *cluster-name*。

```
[settings.ecs]
```

```
cluster = "cluster-name"
```

2. 使用 the section called “擷取 Amazon ECS 最佳化 Bottlerocket AMI 中繼資料” 中包含的其中一個命令來取得 Bottlerocket AMI ID。您會在以下步驟中使用此 ID。
3. 執行下列命令以啟動 Bottlerocket 執行個體。請記得替換以下參數：
   - 將*###*替換為執行個體將在其中啟動的私有或公有子網路的 ID。
   - 將 *bottlerocket_ami* 替換為上一個步驟中取得的 AMI ID。
   - 將 *t3.large* 替換為您要使用的執行個體類型。
   - 將 *region* 替換為區域代碼。

```
aws ec2 run-instances --key-name ecs-bottlerocket-example \
   --subnet-id subnet \
   --image-id bottlerocket_ami \
   --instance-type t3.large \
   --region region \
   --tag-specifications
 'ResourceType=instance,Tags=[{Key=bottlerocket,Value=example}]' \
   --user-data file://userdata.toml \
   --iam-instance-profile Name=ecsInstanceRole
```

4. 執行下列命令來驗證容器執行個體是否已註冊至叢集。當您執行此命令時，請記得替代下列參數：
   - 將 *cluster* 替代為叢集名稱。
   - 將 *region* 替換為您的區域代碼。

```
aws ecs list-container-instances --cluster cluster-name --region region
```

如需如何在 Amazon ECS 上開始使用Bottlerocket作業系統的詳細逐步解說，請參閱在 GitHub 上使用 Bottlerocket AMI 搭配 Amazon ECS，以及 AWS 部落格網站上的 Bottlerocket和 Amazon ECS 入門。

Amazon ECS Linux 容器執行個體管理

當您將 EC2 執行個體用於 Amazon ECS 工作負載時，您必須負責維護執行個體

管理程序

- 啟動 Amazon ECS Linux 容器執行個體

- 引導 Amazon ECS Linux 容器執行個體以傳遞資料
- 設定 Amazon ECS Linux 容器執行個體接收 Spot 執行個體通知
- 啟動 Amazon ECS Linux 容器執行個體時執行指令碼
- 增加 Amazon ECS Linux 容器執行個體網路介面
- 保留 Amazon ECS Linux 容器執行個體記憶體
- 使用 從遠端管理 Amazon ECS 容器執行個體 AWS Systems Manager
- 針對 Amazon ECS Linux 容器執行個體使用 HTTP 代理
- 為您的 Amazon ECS Auto Scaling 群組設定預先初始化的執行個體
- 更新 Amazon ECS 容器代理程式

每個 Amazon ECS 容器代理程式版本都支援不同的功能集，並提供先前版本的錯誤修復。可能的話，我們建議您一律使用最新版本的 Amazon ECS 容器代理程式。若要將您的容器代理更新到最新版本，請參閱「更新 Amazon ECS 容器代理程式」。

若要查看每個代理版本中包含哪些功能和強化功能，請參閱 https://github.com/aws/amazon-ecs-agent/releases。

> **Important**
>
> 可靠指標的最低 Docker 版本是 Docker 版本 `v20.10.13` 及更新版本，該版本隨附於 Amazon ECS 最佳化 AMI `20220607` 及更新版本中。
> Amazon ECS 代理程式版本 `1.20.0` 和更新版本已不支援 `1.9.0` 之前的 Docker 版本。

啟動 Amazon ECS Linux 容器執行個體

您可以使用 Amazon EC2 主控台建立 Amazon ECS 容器執行個體。

您可以透過各種方法啟動執行個體 AWS CLI，包括 Amazon EC2 主控台和 SDK。如需啟動執行個體的其他方法的相關資訊，請參閱《Amazon EC2 使用者指南》中的啟動執行個體。

如需啟動精靈的詳細資訊，請參閱《Amazon EC2 使用者指南》中的使用新的啟動執行個體精靈啟動執行個體。

開始之前，請完成 設定以使用 Amazon ECS。 中的步驟。

您可以使用新的 Amazon EC2 精靈啟動執行個體。啟動執行個體精靈會指定啟動執行個體所需的所有啟動參數。

執行個體組態的參數

- 程序
- 名稱和標籤
- 應用程式和作業系統映像 (Amazon Machine Image)
- 執行個體類型
- 金鑰對 (登入)
- Network settings (網路設定)
- 設定儲存
- 進階詳細資訊

程序

開始之前，請完成 設定以使用 Amazon ECS。 中的步驟。

1. 在 https://console.aws.amazon.com/ec2/ 開啟 Amazon EC2 主控台。
2. 在畫面頂端的導覽列中，會顯示目前的 AWS 區域 （例如美國東部 （俄亥俄））。選取要在其中啟動執行個體的區域。
3. 從 Amazon EC2 主控台儀表板選擇 Launch Instance (啟動執行個體)。

名稱和標籤

執行個體名稱是一個標籤，其中鍵是 Name (名稱)，而值是您指定的名稱。您可以標記執行個體、磁碟區和彈性圖形。對於 Spot 執行個體，您只能標記 Spot 執行個體請求。

指定執行個體名稱和其他標籤是選用的。

- 對於 Name (名稱)，輸入執行個體的描述性名稱。如果您未指定名稱，則可以透過其 ID 來標識執行個體，該 ID 將在您啟動執行個體時自動產生。
- 若要新增其他標籤，請選擇 Add additional tags (新增其他標籤)。選取 Add tag (新增標籤)，然後輸入鍵和值，然後選取要標記的資源類型。為每個要新增的其他標籤重新選擇 Add tag (新增標籤)。

應用程式和作業系統映像 (Amazon Machine Image)

Amazon Machine Image (AMI) 包含建立執行個體所需的資訊。例如，AMI 可能包含作為 Web 伺服器所必需的軟體，例如 Apache 和您的網站。

使用搜尋列尋找由 發佈的合適 Amazon ECS 最佳化 AMI AWS。

1. 在 Search (搜尋) 列中輸入以下其中一個詞語。
   - **`ami-ecs`**
   - Amazon ECS 最佳化 AMI 的 Value (值)。

     如需最新的 Amazon ECS 最佳化 AMI 及其值，請參閱《Linux Amazon ECS 最佳化 AMI》。
2. 按 Enter。
3. 在 Choose an Amazon Machine Image (AMI) (選擇 Amazon Machine Image (AMI)) 頁面上，選取 AWS Marketplace AMIs 索引標籤。
4. 在左側的 Refine results (細化結果) 窗格中，選取 Amazon Web Services 作為 Publisher (發佈者)。
5. 在您要使用的 AMI 的列上選擇 Select (選取)。

   或者，選擇右上方的 Cancel (取消)，以返回啟動執行個體精靈，而不選擇 AMI。將會選取預設 AMI。確保 AMI 符合 Amazon ECS 最佳化 Linux AMIs中概述的要求。

執行個體類型

執行個體類型定義執行個體的硬體組態和大小。較大的執行個體類型具有較多的 CPU 和記憶體。如需詳細資訊，請參閱執行個體類型。

- 針對 Instance type (執行個體類型)，選取執行個體的執行個體類型。

  您選取的執行個體類型，會決定您任務執行所在的可用資源。

金鑰對 (登入)

針對 Key pair name (金鑰對名稱)，選擇現有的金鑰對，或選擇 Create new key pair (建立新的金鑰對) 以建立新的金鑰對。

> **Important**
>
> 如果您選擇 Proceed without key pair (Not recommended) (繼續而不使用金鑰對 (不建議)) 選項，您將無法連線到執行個體，除非您選擇已設定為允許使用者透過其他方式登入的 AMI。

Network settings (網路設定)

視需要設定網路設定。

- Networking platform (聯網平台)：選擇 Virtual Private Cloud (VPC) (虛擬私有雲端 (VPC))，然後在 Network interfaces (網路介面) 區段指定子網路。
- VPC：選取要在其中建立安全群組的現有 VPC。
- Subnet (子網)：您可以在與可用區域、Local Zone、Wavelength 區域或 Outpost 相關聯的子網中啟動執行個體。

  若要在可用區域中啟動執行個體，請選取要在當中啟動執行個體的子網。若要建立新的子網，請選擇 Create new subnet (建立新的子網)，以前往 Amazon VPC 主控台。完成後，請返回啟動執行個體精靈並選擇 Refresh (重新整理) 圖示，載入清單中的子網。

  在 Local Zone 中啟動執行個體，選取您在 Local Zone 中建立的子網。

  若要在 Outpost 中啟動執行個體，請在與 Outpost 相關聯的 VPC 中選取子網。
- Auto-assign Public IP (自動指派公有 IP)：如果您的執行個體應從網際網路存取，請確認 Auto-assign Public IP (自動指派公有 IP) 欄位設為 Enable (啟用)。如果不是，請將此欄位設為 Disable (停用)。

  > **Note**
  >
  > 容器執行個體需要存取，才可以與 Amazon ECS 服務端點通訊。可透過介面 VPC 端點或透過具備公有 IP 位址的容器執行個體來實現。
  > 如需介面 VPC 端點的詳細資訊，請參閱 Amazon ECS 介面 VPC 端點 (AWS PrivateLink)
  > 如果您沒有設定介面 VPC 端點，且容器執行個體沒有公有 IP 位址，則它們必須使用網路位址轉譯 (NAT) 來提供此存取。如需詳細資訊，請參閱《Amazon VPC 使用者指南》中的 NAT 閘道和本指南中的 針對 Amazon ECS Linux 容器執行個體使用 HTTP 代理。
- Firewall (security groups) (防火牆 (安全群組))：使用安全群組定義容器執行個體的防火牆規則。這些規則指定傳遞至容器執行個體的傳入網路流量。所有其他流量都會遭到忽略。
  - 若要選取現有的安全群組，請選擇 Select existing security group (選取現有安全群組)，然後從您在 設定以使用 Amazon ECS。 建立的安全群組選取您的安全群組。

設定儲存

您選取的 AMI 包含一或多個儲存體磁碟區，包括根磁碟區。您可以指定要連接至執行個體的其他磁碟區。

您可以使用 Simple (簡單) 檢視。

- Storage type (儲存類型)：為您的容器執行個體設定儲存。

  如果您使用的是 Amazon ECS 最佳化 Amazon Linux 2 AMI，則您的執行個體已設定單一 30 GiB 磁碟區 (在作業系統和 Docker 之間共用)。

  如果您使用的是 Amazon ECS 最佳化 AMI，則您的執行個體已設定兩個磁碟區。Root (根目錄) 磁碟區供作業系統使用，而第二個 Amazon EBS 磁碟區 (連接到 `/dev/xvdcz`) 則供 Docker 使用。

  您可以選擇性地增加或減少您執行個體的磁碟區大小，使其符合您的應用程式需求。

進階詳細資訊

針對 Advanced Details (進階詳細資訊)，展開此區段來檢視欄位，指定執行個體的其他參數。

- Purchasing option (購買選項)：選擇 Request Spot instances (請求 Spot 執行個體) 以請求 Spot 執行個體。您也需要設定與 Spot 執行個體相關的其他欄位。如需詳細資訊，請參閱「Spot 執行個體要求」。

  > **Note**
  >
  > 如果您使用的是 Spot 執行個體，而且看到 `Not available` 訊息，您可能需要選擇不同的執行個體類型。

  .

- IAM instance profile (IAM 執行個體設定檔)：選取您的容器執行個體 IAM 角色。這通常命名為 `ecsInstanceRole`。

  > **Important**
  >
  > 如果您未使用適當的 IAM 許可啟動容器執行個體，Amazon ECS 代理程式就無法連線到叢集。如需詳細資訊，請參閱Amazon ECS 容器執行個體 IAM 角色。

- 使用者資料：使用使用者資料設定 Amazon ECS 容器執行個體，例如來自 的代理程式環境變數Amazon ECS 容器代理程式組態。Amazon EC2 使用者資料指令碼僅於執行個體初次啟動時執行一次。以下是使用者資料用途的常見範例：
  - 您的容器執行個體預設會在您的預設叢集中啟動。若要在非預設的叢集中啟動，請選擇 Advanced Details (進階詳細資訊) 清單。然後，將以下指令碼貼入 User data (使用者資料) 欄位，以您的叢集名稱取代 *your_cluster_name*。

```
#!/bin/bash
echo ECS_CLUSTER=your_cluster_name >> /etc/ecs/ecs.config
```

  - 如果您在 Amazon S3 中有一個 `ecs.config` 檔案，並已為您的容器執行個體角色啟用 Amazon S3 唯讀存取，請選擇 Advanced Details (進階詳細資訊) 清單。然後，將下列指令碼貼到使用者資料欄位中，將*##_bucket_name* 取代為儲存貯體的名稱，以安裝 AWS CLI 並在啟動時寫入您的組態檔案。

> **Note**
>
> 如需此組態的詳細資訊，請參閱「在 Amazon S3 中存放 Amazon ECS 容器執行個體組態」。

```
#!/bin/bash
yum install -y aws-cli
aws s3 cp s3://your_bucket_name/ecs.config /etc/ecs/ecs.config
```

  - 使用 `ECS_CONTAINER_INSTANCE_TAGS` 組態參數為您的容器執行個體指定標籤。這會建立只與 Amazon ECS 相關聯的標籤，所以無法使用 Amazon EC2 API 列出這些標籤。

> **Important**
>
> 如果您使用 Amazon EC2 Auto Scaling 群組啟動容器執行個體，則應使用 ECS_CONTAINER_INSTANCE_TAGS 代理程式組態參數來新增標籤。這是由於標籤新增到了使用 Auto Scaling 群組啟動的 Amazon EC2 執行個體。

```
#!/bin/bash
cat <<'EOF' >> /etc/ecs/ecs.config
ECS_CLUSTER=your_cluster_name
```

```
ECS_CONTAINER_INSTANCE_TAGS={"tag_key": "tag_value"}
EOF
```

- 為您的容器執行個體指定標籤，然後使用 ECS_CONTAINER_INSTANCE_PROPAGATE_TAGS_FROM 組態參數將它們從 Amazon EC2 傳播到 Amazon ECS

  以下的使用者資料指令碼範例會傳播與容器執行個體相關聯的標籤，還會向名為 your_cluster_name 的叢集註冊容器執行個體：

```
#!/bin/bash
cat <<'EOF' >> /etc/ecs/ecs.config
ECS_CLUSTER=your_cluster_name
ECS_CONTAINER_INSTANCE_PROPAGATE_TAGS_FROM=ec2_instance
EOF
```

如需詳細資訊，請參閱引導 Amazon ECS Linux 容器執行個體以傳遞資料。

引導 Amazon ECS Linux 容器執行個體以傳遞資料

當您啟動 Amazon EC2 執行個體時，您可以將使用者資料傳遞至 EC2 執行個體。此資料可用來執行常見的自動化組態任務，甚至在執行個體啟動時，執行指令碼。對於 Amazon ECS，使用者資料的最常用案例是將組態資訊傳送到 Docker 常駐程式和 Amazon ECS 容器代理程式。

您可以將多種類型的使用者資料傳遞給 Amazon EC2，包含雲端 boothook、shell 指令碼和 cloud-init 指令。如需這些和其他格式類型的詳細資訊，請參閱 Cloud-Init 文件。

若要在使用 Amazon EC2 啟動精靈時傳遞使用者資料，請參閱 啟動 Amazon ECS Linux 容器執行個體。

您可以設定容器執行個體在容器代理程式組態或 Docker 協助程式組態中傳遞資料。

Amazon ECS 容器代理程式

Amazon ECS 最佳化 AMI 的 Linux 變體會在容器代理程式啟動時，於 /etc/ecs/ecs.config 檔案中尋找代理程式組態資料。您可以使用 Amazon EC2 使用者資料在啟動時指定此組態資料。如需可用 Amazon ECS 容器代理程式組態變數的詳細資訊，請參閱 Amazon ECS 容器代理程式組態。

若只要設定單一代理程式組態變數 (例如叢集名稱)，請使用 echo 將變數複製至組態檔案：

```
#!/bin/bash
```

```
echo "ECS_CLUSTER=MyCluster" >> /etc/ecs/ecs.config
```

如有多個變數要寫入 /etc/ecs/ecs.config，請使用以下 heredoc 格式。此格式會將開頭為 cat 和 EOF 之行間的所有項目寫入組態檔案。

```
#!/bin/bash
cat <<'EOF' >> /etc/ecs/ecs.config
ECS_CLUSTER=MyCluster
ECS_ENGINE_AUTH_TYPE=docker
ECS_ENGINE_AUTH_DATA={"https://index.docker.io/v1/":
{"username":"my_name","password":"my_password","email":"email@example.com"}}
ECS_LOGLEVEL=debug
ECS_WARM_POOLS_CHECK=true
EOF
```

若要設定自訂執行個體屬性，請設定 ECS_INSTANCE_ATTRIBUTES 環境變數。

```
#!/bin/bash
cat <<'EOF' >> ecs.config
ECS_INSTANCE_ATTRIBUTES={"envtype":"prod"}
EOF
```

Docker 常駐程式

您可以使用 Amazon EC2 使用者資料指定 Docker 常駐程式組態資訊。如需組態選項的詳細資訊，請參閱 Docker 常駐程式文件。

在下例中，自訂選項會新增到 Docker 常駐程式組態檔案，/etc/docker/daemon.json，然後在執行個體啟動時在使用者資料中指定它。

```
#!/bin/bash
cat <<EOF >/etc/docker/daemon.json
{"debug": true}
EOF
systemctl restart docker --no-block
```

在下例中，自訂選項會新增到 Docker 常駐程式組態檔案，/etc/docker/daemon.json，然後在執行個體啟動時在使用者資料中指定它。此範例顯示如何停用 Docker 常駐程式組態檔案中的 Docker 代理。

```
#!/bin/bash
```

```
cat <<EOF >/etc/docker/daemon.json
{"userland-proxy": false}
EOF
systemctl restart docker --no-block
```

設定 Amazon ECS Linux 容器執行個體接收 Spot 執行個體通知

當 Spot 價格超過請求的最高價或容量不再可用時，Amazon EC2 會終止、停止或休眠您的 Spot 執行個體。Amazon EC2 會針對終止和停止動作提前兩分鐘發出 Spot 執行個體中斷通知。其不會針對休眠動作提前兩分鐘發出通知。如果執行個體上的 Amazon ECS Spot 執行個體耗盡已開啟，Amazon ECS 會收到 Spot 執行個體中斷通知，並將執行個體置於 DRAINING 狀態。

> **Important**
>
> 當透過 Auto Scaling 容量重新平衡移除執行個體時，Amazon ECS 不會收到 Amazon EC2 發出的通知。如需詳細資訊，請參閱 Amazon EC2 Auto Scaling 容量重新平衡。

將容器執行個體設定為 DRAINING 時，Amazon ECS 會避免在容器執行個體中放置新的任務排程。PENDING 狀態下即將耗盡的容器執行個體服務任務會立即停止。如果叢集有可用的容器執行個體，則會在這些容器執行個體上啟動替代服務任務。

Spot 執行個體耗盡預設為關閉。

您可以在啟動執行個體時開啟 Spot 執行個體耗盡。將下列指令碼新增至使用者資料欄位。將 *MyCluster* 取代為要註冊容器執行個體的叢集名稱。

```
#!/bin/bash
cat <<'EOF' >> /etc/ecs/ecs.config
ECS_CLUSTER=MyCluster
ECS_ENABLE_SPOT_INSTANCE_DRAINING=true
EOF
```

如需詳細資訊，請參閱啟動 Amazon ECS Linux 容器執行個體。

針對現有的容器執行個體開啟 Spot 執行個體耗盡

1. 透過 SSH 連接到 Spot 執行個體。
2. 編輯 /etc/ecs/ecs.config 檔案並新增以下內容：

```
ECS_ENABLE_SPOT_INSTANCE_DRAINING=true
```

3. 重新啟動 ecs 服務。

   - 對於 Amazon ECS 最佳化 Amazon Linux 2 AMI：

     ```
     sudo systemctl restart ecs
     ```

4. (選用) 您可以驗證代理已在執行中，並透過查詢代理自我檢查 API 操作，查看您新的容器執行個體的一些資訊。如需詳細資訊，請參閱the section called “容器簡介”。

   ```
   curl http://localhost:51678/v1/metadata
   ```

啟動 Amazon ECS Linux 容器執行個體時執行指令碼

您可能需要在每個容器執行個體上執行特定容器，以處理操作或安全問題，例如監控、安全性、指標、服務探索或記錄。

若要執行此作業，您可以將您的容器執行個體設定為在啟動時或在某些 init 系統中 (例如 Upstart 或 systemd)，以使用者資料指令碼呼叫 docker run 命令。雖然此方法可行，但有一些缺點，因為 Amazon ECS 對容器一無所知，也無法監控 CPU、記憶體、連接埠或任何其他使用的資源。為確保 Amazon ECS 可正確說明所有任務資源，請為在您容器執行個體上執行的容器建立任務定義。然後，使用 Amazon ECS，利用 Amazon EC2 使用者資料在啟動時放置任務。

下列程序中的 Amazon EC2 使用者資料指令碼會使用 Amazon ECS 自我檢查 API 來識別容器執行個體。然後，它會使用 AWS CLI 和 start-task命令，在啟動期間自行執行指定的任務。

在容器執行個體啟動階段啟動任務

1. 修改您的 ecsInstanceRole IAM 角色，新增 StartTask API 操作的許可。如需詳細資訊，請參閱AWS Identity and Access Management 《 使用者指南》中的更新角色的許可。
2. 使用 Amazon ECS 最佳化 Amazon Linux 2 AMI 啟動一或多個容器執行個體。啟動新的容器執行個體，並在 EC2 使用者資料中使用下列範例指令碼。將 *your_cluster_name* 取代為容器執行個體要註冊的叢集，並將 *my_task_def* 取代為啟動時要在執行個體上執行的任務定義。

   如需詳細資訊，請參閱啟動 Amazon ECS Linux 容器執行個體。

> **Note**
>
> 以下的 MIME 多分段內容使用 shell 指令碼來設定組態值並安裝套件。在 ecs 服務已執行，並可使用自我檢查 API 之後，還會使用 systemd 工作啟動任務。

```
Content-Type: multipart/mixed; boundary="==BOUNDARY=="
MIME-Version: 1.0

--==BOUNDARY==
Content-Type: text/x-shellscript; charset="us-ascii"

#!/bin/bash
# Specify the cluster that the container instance should register into
cluster=your_cluster_name

# Write the cluster configuration variable to the ecs.config file
# (add any other configuration variables here also)
echo ECS_CLUSTER=$cluster >> /etc/ecs/ecs.config

START_TASK_SCRIPT_FILE="/etc/ecs/ecs-start-task.sh"
cat <<- 'EOF' > ${START_TASK_SCRIPT_FILE}
 exec 2>>/var/log/ecs/ecs-start-task.log
 set -x

 # Install prerequisite tools
 yum install -y jq aws-cli

 # Wait for the ECS service to be responsive
 until curl -s http://localhost:51678/v1/metadata
 do
  sleep 1
 done

 # Grab the container instance ARN and AWS Region from instance metadata
 instance_arn=$(curl -s http://localhost:51678/v1/metadata | jq -r '.
 | .ContainerInstanceArn' | awk -F/ '{print $NF}' )
 cluster=$(curl -s http://localhost:51678/v1/metadata | jq -r '. | .Cluster' | awk
 -F/ '{print $NF}' )
 region=$(curl -s http://localhost:51678/v1/metadata | jq -r '.
 | .ContainerInstanceArn' | awk -F: '{print $4}')
```

```
 # Specify the task definition to run at launch
 task_definition=my_task_def

 # Run the AWS CLI start-task command to start your task on this container instance
 aws ecs start-task --cluster $cluster --task-definition $task_definition --
container-instances $instance_arn --started-by $instance_arn --region $region
EOF

# Write systemd unit file
UNIT="ecs-start-task.service"
cat <<- EOF > /etc/systemd/system/${UNIT}
      [Unit]
      Description=ECS Start Task
      Requires=ecs.service
      After=ecs.service

      [Service]
      Restart=on-failure
      RestartSec=30
      ExecStart=/usr/bin/bash ${START_TASK_SCRIPT_FILE}

      [Install]
      WantedBy=default.target
EOF

# Enable our ecs.service dependent service with `--no-block` to prevent systemd
 deadlock
# See https://github.com/aws/amazon-ecs-agent/issues/1707
systemctl enable --now --no-block "${UNIT}"
--==BOUNDARY==--
```

3. 確認您的容器執行個體在正確的叢集中啟動，而且您的任務已啟動。

   a. 開啟主控台，網址為 https://console.aws.amazon.com/ecs/v2。

   b. 從導覽列選擇您叢集所在的區域。

   c. 在導覽窗格中選擇 Clusters (叢集)，並選取託管您容器執行個體的叢集。

   d. 在叢集頁面上，選擇任務，然後選擇您的任務。

      您啟動的每個容器執行個體都應該有任務在正在其中執行。

如果您沒有看到您的任務，您可以使用 SSH 登入您的容器執行個體，檢查 `/var/log/ecs/ecs-start-task.log` 檔案的偵錯資訊。

增加 Amazon ECS Linux 容器執行個體網路介面

> **Note**
> 此功能不適用於 Fargate。

使用 `awsvpc` 網路模式的每個任務都會收到自己的彈性網路介面 (ENI)，其會連接到託管它的容器執行個體。Amazon EC2 執行個體可連接的網路介面數量有預設限制，主要網路介面計算在內。例如，根據預設，`c5.large` 執行個體最多可連接三個 ENI。執行個體的主要網路界面計算在內，所以您可以再連接兩個 ENI 到執行個體。由於使用 `awsvpc` 網路模式的每個任務都需要 ENI，因此您通常只能在此執行個體類型上執行兩個此類任務。

Amazon ECS 支援使用支援的 Amazon EC2 執行個體類型來啟動ENI密度更高的容器執行個體。當您使用這些執行個體類型並開啟`awsvpcTrunking`帳戶設定時，新啟動的容器執行個體上會提供其他 ENIs。此組態可讓您在每個容器執行個體中安排更多任務。若要使用 主控台開啟功能，請參閱 修改 Amazon ECS帳戶設定。若要使用 AWS CLI 開啟功能，請參閱 使用 管理 Amazon ECS帳戶設定 AWS CLI。

例如，具有 的`c5.large`執行個體`awsvpcTrunking`有 12 個的增加ENI限制。容器執行個體會有主要網路介面，而 Amazon ECS 會建立「幹線」網路介面，並將它連接到容器執行個體。因此，此組態可讓您在容器執行個體中啟動十項任務，而不是目前的兩項任務。

幹線網路介面由 Amazon ECS 全受管，當您在叢集中終止或取消註冊您的容器執行個體時，會將其刪除。如需詳細資訊，請參閱EC2 啟動類型的 Amazon ECS 任務聯網選項。

考量事項

使用中ENI繼功能時，請考慮下列事項。

- 只有 Amazon ECS 最佳化 AMI 的 Linux 變體，或其他 Amazon Linux 變體具有 或更新版本`1.28.1`的容器代理程式，以及 版本 `1.28.1-2`或更新版本的 ecs-init 套件，支援增加ENI的限制。如果您使用 Amazon ECS 最佳化 AMI 的最新 Linux 變體，將會滿足這些要求。目前不支援 Windows 容器。

- 只有在啟用 之後啟動的新 Amazon EC2 執行個體才會awsvpcTrunking收到增加ENI的限制和幹線網路介面。無論採取何種動作，先前啟動的執行個體都不會收到這些功能。
- Amazon EC2 執行個體必須關閉資源型 IPv4 DNS 請求。若要停用此選項，請在 Amazon EC2 主控台中建立新執行個體時清除啟用資源型 IPV4 （記錄） DNS 請求選項。若要使用 停用此選項 AWS CLI，請使用下列命令。

```
aws ec2 modify-private-dns-name-options --instance-id i-xxxxxxx --no-enable-resource-name-dns-a-record --no-dry-run
```

- 不支援共用子網路中的 Amazon EC2 執行個體。就算使用也無法註冊到叢集。
- 您的任務必須使用awsvpc網路模式和 EC2 啟動類型。使用 Fargate 啟動類型的任務一律會收到專用，ENI無論啟動多少，因此不需要此功能。
- 您的任務必須在與容器執行個體相同的 Amazon VPC 中啟動。如果您的任務不在同一 VPC 中，則任務會因為屬性錯誤而失敗。
- 啟動新的容器執行個體時，執行個體將轉換為 REGISTERING 狀態，同時為執行個體佈建幹線彈性網路介面。如果註冊失敗，執行個體會轉換到 REGISTRATION_FAILED 狀態。透過說明容器執行個體以檢視說明失敗原因的 statusReason 欄位，可對失敗註冊進行故障排除。然後可以手動解除註冊或終止容器執行個體。容器執行個體成功取消註冊或終止後，Amazon ECS 會刪除中繼線 ENI。

> **Note**
>
> Amazon ECS 會發出容器執行個體狀態變更事件，您可以監控執行個體是否轉換為 REGISTRATION_FAILED 狀態。如需詳細資訊，請參閱Amazon ECS 容器執行個體狀態變更事件。

- 容器執行個體終止後，執行個體將轉換為 DEREGISTERING 狀態，同時解除佈建幹線彈性網路介面。然後，執行個體會轉換為 INACTIVE 狀態。
- 如果公有子網路中具有增加ENI限制的容器執行個體停止然後重新啟動，執行個體會失去其公有 IP 地址，而容器代理程式會失去其連線。
- 當您啟用 時awsvpcTrunking，容器執行個體會收到使用 VPC 預設安全群組的額外 ENI ，並由 Amazon ECS 管理。

必要條件

啟動具有增加ENI限制的容器執行個體之前，必須先完成下列先決條件。

- 必須先建立 Amazon ECS 的服務連結角色。Amazon ECS 服務連結角色為 Amazon ECS 提供代表您呼叫其他 AWS 服務的許可。這個角色會在您建立叢集，或在 AWS Management Console中建立或更新服務時，自動為您建立。如需詳細資訊，請參閱使用 Amazon ECS 的服務連結角色。您也可以使用下列 AWS CLI 命令建立服務連結角色。

```
aws iam create-service-linked-role --aws-service-name ecs.amazonaws.com
```

- 您的帳戶或容器執行個體 IAM 角色必須啟用 `awsvpcTrunking` 帳戶設定。建議您建立 2 個容器執行個體角色 (`ecsInstanceRole`)。然後，您可以為一個角色啟用`awsvpcTrunking`帳戶設定，並將該角色用於需要 ENI 中繼的任務。如需容器執行個體角色的相關資訊，請參閱 Amazon ECS 容器執行個體 IAM 角色。

符合先決條件後，您可以使用其中一個支援的 Amazon EC2 執行個體類型來啟動新的容器執行個體，且執行個體會有更高的ENI限制。如需支援的執行個體類型清單，請參閱 支援增加 Amazon ECS 容器網路介面的執行個體。容器執行個體必須具有容器代理程式的 `1.28.1` 版或更新版本，以及 ecs-init 套件的 `1.28.1-2` 版或更新版本。如果您使用 Amazon ECS 最佳化 AMI 的最新 Linux 變體，將會滿足這些要求。如需詳細資訊，請參閱啟動 Amazon ECS Linux 容器執行個體。

> **Important**
>
> Amazon EC2 執行個體必須關閉資源型 IPv4 DNS 請求。若要停用此選項，在使用 Amazon EC2 主控台建立新執行個體時，請務必取消選取 Enable resource-based IPV4 (A record) DNS requests (啟用資源型 IPV4 (A 記錄) DNS 請求) 選項。若要使用 停用此選項 AWS CLI，請使用下列命令。
>
> ```
> aws ec2 modify-private-dns-name-options --instance-id i-xxxxxxx --no-enable-resource-name-dns-a-record --no-dry-run
> ```

若要使用 檢視具有增加ENI限制的容器執行個體 AWS CLI

每個容器執行個體都有預設的網路界面，稱為幹線網路界面。使用下列命令，透過查詢 `ecs.awsvpc-trunk-id` 屬性來列出具有增加ENI限制的容器執行個體，這表示其具有中繼網路介面。

- list-attributes (AWS CLI)

```
aws ecs list-attributes \
      --target-type container-instance \
```

```
    --attribute-name ecs.awsvpc-trunk-id \
    --cluster cluster_name \
    --region us-east-1
```

- Get-ECSAttributeList (AWS Tools for Windows PowerShell)

```
Get-ECSAttributeList -TargetType container-instance -AttributeName ecs.awsvpc-trunk-
id -Region us-east-1
```

支援增加 Amazon ECS 容器網路介面的執行個體

下列顯示受支援的 Amazon EC2 執行個體類型，以及在啟用 awsvpcTrunking 帳戶設定的前後，可以在每個執行個體類型上啟動使用 awsvpc 網路模式的任務數量。

> **Important**
>
> 雖然同一執行個體系列支援其他執行個體類型，但不支援 a1.metal、c5.metal、c5a.8xlarge、c5ad.8xlarge、c5d.metal、m5.metal、p3dn.24xla 和 r5d.metal 執行個體類型。
> 不支援 c5n、d3、d3en、g3、g3s、g4dn、i3、i3en、inf1、m5dn、m5n、m5zn、mac1、r5b、r5n、r5 和 z1d 執行個體系列。

主題

- 一般用途
- 運算最佳化
- 記憶體最佳化
- 儲存最佳化
- 加速運算
- 高效能運算

一般用途

執行個體類型	無ENI中繼的任務限制	含ENI中繼的任務限制
a1.medium	1	10
a1.large	2	10
a1.xlarge	3	20
a1.2xlarge	3	40
a1.4xlarge	7	60
m5.large	2	10
m5.xlarge	3	20
m5.2xlarge	3	40
m5.4xlarge	7	60
m5.8xlarge	7	60
m5.12xlarge	7	60
m5.16xlarge	14	120
m5.24xlarge	14	120
m5a.large	2	10
m5a.xlarge	3	20
m5a.2xlarge	3	40
m5a.4xlarge	7	60
m5a.8xlarge	7	60
m5a.12xlarge	7	60

執行個體類型	無ENI中繼的任務限制	含ENI中繼的任務限制
m5a.16xlarge	14	120
m5a.24xlarge	14	120
m5ad.large	2	10
m5ad.xlarge	3	20
m5ad.2xlarge	3	40
m5ad.4xlarge	7	60
m5ad.8xlarge	7	60
m5ad.12xlarge	7	60
m5ad.16xlarge	14	120
m5ad.24xlarge	14	120
m5d.large	2	10
m5d.xlarge	3	20
m5d.2xlarge	3	40
m5d.4xlarge	7	60
m5d.8xlarge	7	60
m5d.12xlarge	7	60
m5d.16xlarge	14	120
m5d.24xlarge	14	120
m5d.metal	14	120
m6a.large	2	10

執行個體類型	無ENI中繼的任務限制	含ENI中繼的任務限制
m6a.xlarge	3	20
m6a.2xlarge	3	40
m6a.4xlarge	7	60
m6a.8xlarge	7	90
m6a.12xlarge	7	120
m6a.16xlarge	14	120
m6a.24xlarge	14	120
m6a.32xlarge	14	120
m6a.48xlarge	14	120
m6a.metal	14	120
m6g.medium	1	4
m6g.large	2	10
m6g.xlarge	3	20
m6g.2xlarge	3	40
m6g.4xlarge	7	60
m6g.8xlarge	7	60
m6g.12xlarge	7	60
m6g.16xlarge	14	120
m6g.metal	14	120
m6gd.medium	1	4

執行個體類型	無ENI中繼的任務限制	含ENI中繼的任務限制
m6gd.large	2	10
m6gd.xlarge	3	20
m6gd.2xlarge	3	40
m6gd.4xlarge	7	60
m6gd.8xlarge	7	60
m6gd.12xlarge	7	60
m6gd.16xlarge	14	120
m6gd.metal	14	120
m6i.large	2	10
m6i.xlarge	3	20
m6i.2xlarge	3	40
m6i.4xlarge	7	60
m6i.8xlarge	7	90
m6i.12xlarge	7	120
m6i.16xlarge	14	120
m6i.24xlarge	14	120
m6i.32xlarge	14	120
m6i.metal	14	120
m6id.large	2	10
m6id.xlarge	3	20

執行個體類型	無ENI中繼的任務限制	含ENI中繼的任務限制
m6id.2xlarge	3	40
m6id.4xlarge	7	60
m6id.8xlarge	7	90
m6id.12xlarge	7	120
m6id.16xlarge	14	120
m6id.24xlarge	14	120
m6id.32xlarge	14	120
m6id.metal	14	120
m6idn.large	2	10
m6idn.xlarge	3	20
m6idn.2xlarge	3	40
m6idn.4xlarge	7	60
m6idn.8xlarge	7	90
m6idn.12xlarge	7	120
m6idn.16xlarge	14	120
m6idn.24xlarge	14	120
m6idn.32xlarge	15	120
m6idn.metal	15	120
m6in.large	2	10
m6in.xlarge	3	20

執行個體類型	無ENI中繼的任務限制	含ENI中繼的任務限制
m6in.2xlarge	3	40
m6in.4xlarge	7	60
m6in.8xlarge	7	90
m6in.12xlarge	7	120
m6in.16xlarge	14	120
m6in.24xlarge	14	120
m6in.32xlarge	15	120
m6in.metal	15	120
m7a.medium	1	4
m7a.large	2	10
m7a.xlarge	3	20
m7a.2xlarge	3	40
m7a.4xlarge	7	60
m7a.8xlarge	7	90
m7a.12xlarge	7	120
m7a.16xlarge	14	120
m7a.24xlarge	14	120
m7a.32xlarge	14	120
m7a.48xlarge	14	120
m7a.metal-48xl	14	120

執行個體類型	無ENI中繼的任務限制	含ENI中繼的任務限制
m7g.medium	1	4
m7g.large	2	10
m7g.xlarge	3	20
m7g.2xlarge	3	40
m7g.4xlarge	7	60
m7g.8xlarge	7	60
m7g.12xlarge	7	60
m7g.16xlarge	14	120
m7g.metal	14	120
m7gd.medium	1	4
m7gd.large	2	10
m7gd.xlarge	3	20
m7gd.2xlarge	3	40
m7gd.4xlarge	7	60
m7gd.8xlarge	7	60
m7gd.12xlarge	7	60
m7gd.16xlarge	14	120
m7gd.metal	14	120
m7i.large	2	10
m7i.xlarge	3	20

執行個體類型	無ENI中繼的任務限制	含ENI中繼的任務限制
m7i.2xlarge	3	40
m7i.4xlarge	7	60
m7i.8xlarge	7	90
m7i.12xlarge	7	120
m7i.16xlarge	14	120
m7i.24xlarge	14	120
m7i.48xlarge	14	120
m7i.metal-24xl	14	120
m7i.metal-48xl	14	120
m7i-flex.large	2	4
m7i-flex.xlarge	3	10
m7i-flex.2xlarge	3	20
m7i-flex.4xlarge	7	40
m7i-flex.8xlarge	7	60
m7i-flex.12xlarge	7	120
m7i-flex.16xlarge	14	120
m8g.medium	1	4
m8g.large	2	10
m8g.xlarge	3	20
m8g.2xlarge	3	40

執行個體類型	無ENI中繼的任務限制	含ENI中繼的任務限制
m8g.4xlarge	7	60
m8g.8xlarge	7	60
m8g.12xlarge	7	60
m8g.16xlarge	14	120
m8g.24xlarge	14	120
m8g.48xlarge	14	120
m8g.metal-24xl	14	120
m8g.metal-48xl	14	120
mac2.metal	7	12
mac2-m1ultra.metal	7	12
mac2-m2.metal	7	12
mac2-m2pro.metal	7	12

運算最佳化

執行個體類型	無ENI中繼的任務限制	含ENI中繼的任務限制
c5.large	2	10
c5.xlarge	3	20
c5.2xlarge	3	40
c5.4xlarge	7	60
c5.9xlarge	7	60

執行個體類型	無ENI中繼的任務限制	含ENI中繼的任務限制
c5.12xlarge	7	60
c5.18xlarge	14	120
c5.24xlarge	14	120
c5a.large	2	10
c5a.xlarge	3	20
c5a.2xlarge	3	40
c5a.4xlarge	7	60
c5a.12xlarge	7	60
c5a.16xlarge	14	120
c5a.24xlarge	14	120
c5ad.large	2	10
c5ad.xlarge	3	20
c5ad.2xlarge	3	40
c5ad.4xlarge	7	60
c5ad.12xlarge	7	60
c5ad.16xlarge	14	120
c5ad.24xlarge	14	120
c5d.large	2	10
c5d.xlarge	3	20
c5d.2xlarge	3	40

執行個體類型	無ENI中繼的任務限制	含ENI中繼的任務限制
c5d.4xlarge	7	60
c5d.9xlarge	7	60
c5d.12xlarge	7	60
c5d.18xlarge	14	120
c5d.24xlarge	14	120
c6a.large	2	10
c6a.xlarge	3	20
c6a.2xlarge	3	40
c6a.4xlarge	7	60
c6a.8xlarge	7	90
c6a.12xlarge	7	120
c6a.16xlarge	14	120
c6a.24xlarge	14	120
c6a.32xlarge	14	120
c6a.48xlarge	14	120
c6a.metal	14	120
c6g.medium	1	4
c6g.large	2	10
c6g.xlarge	3	20
c6g.2xlarge	3	40

執行個體類型	無ENI中繼的任務限制	含ENI中繼的任務限制
c6g.4xlarge	7	60
c6g.8xlarge	7	60
c6g.12xlarge	7	60
c6g.16xlarge	14	120
c6g.metal	14	120
c6gd.medium	1	4
c6gd.large	2	10
c6gd.xlarge	3	20
c6gd.2xlarge	3	40
c6gd.4xlarge	7	60
c6gd.8xlarge	7	60
c6gd.12xlarge	7	60
c6gd.16xlarge	14	120
c6gd.metal	14	120
c6gn.medium	1	4
c6gn.large	2	10
c6gn.xlarge	3	20
c6gn.2xlarge	3	40
c6gn.4xlarge	7	60
c6gn.8xlarge	7	60

執行個體類型	無ENI中繼的任務限制	含ENI中繼的任務限制
c6gn.12xlarge	7	60
c6gn.16xlarge	14	120
c6i.large	2	10
c6i.xlarge	3	20
c6i.2xlarge	3	40
c6i.4xlarge	7	60
c6i.8xlarge	7	90
c6i.12xlarge	7	120
c6i.16xlarge	14	120
c6i.24xlarge	14	120
c6i.32xlarge	14	120
c6i.metal	14	120
c6id.large	2	10
c6id.xlarge	3	20
c6id.2xlarge	3	40
c6id.4xlarge	7	60
c6id.8xlarge	7	90
c6id.12xlarge	7	120
c6id.16xlarge	14	120
c6id.24xlarge	14	120

執行個體類型	無ENI中繼的任務限制	含ENI中繼的任務限制
c6id.32xlarge	14	120
c6id.metal	14	120
c6in.large	2	10
c6in.xlarge	3	20
c6in.2xlarge	3	40
c6in.4xlarge	7	60
c6in.8xlarge	7	90
c6in.12xlarge	7	120
c6in.16xlarge	14	120
c6in.24xlarge	14	120
c6in.32xlarge	15	120
c6in.metal	15	120
c7a.medium	1	4
c7a.large	2	10
c7a.xlarge	3	20
c7a.2xlarge	3	40
c7a.4xlarge	7	60
c7a.8xlarge	7	90
c7a.12xlarge	7	120
c7a.16xlarge	14	120

執行個體類型	無ENI中繼的任務限制	含ENI中繼的任務限制
c7a.24xlarge	14	120
c7a.32xlarge	14	120
c7a.48xlarge	14	120
c7a.metal-48xl	14	120
c7g.medium	1	4
c7g.large	2	10
c7g.xlarge	3	20
c7g.2xlarge	3	40
c7g.4xlarge	7	60
c7g.8xlarge	7	60
c7g.12xlarge	7	60
c7g.16xlarge	14	120
c7g.metal	14	120
c7gd.medium	1	4
c7gd.large	2	10
c7gd.xlarge	3	20
c7gd.2xlarge	3	40
c7gd.4xlarge	7	60
c7gd.8xlarge	7	60
c7gd.12xlarge	7	60

執行個體類型	無ENI中繼的任務限制	含ENI中繼的任務限制
c7gd.16xlarge	14	120
c7gd.metal	14	120
c7gn.medium	1	4
c7gn.large	2	10
c7gn.xlarge	3	20
c7gn.2xlarge	3	40
c7gn.4xlarge	7	60
c7gn.8xlarge	7	60
c7gn.12xlarge	7	60
c7gn.16xlarge	14	120
c7gn.metal	14	120
c7i.large	2	10
c7i.xlarge	3	20
c7i.2xlarge	3	40
c7i.4xlarge	7	60
c7i.8xlarge	7	90
c7i.12xlarge	7	120
c7i.16xlarge	14	120
c7i.24xlarge	14	120
c7i.48xlarge	14	120

執行個體類型	無ENI中繼的任務限制	含ENI中繼的任務限制
c7i.metal-24xl	14	120
c7i.metal-48xl	14	120
c7i-flex.large	2	4
c7i-flex.xlarge	3	10
c7i-flex.2xlarge	3	20
c7i-flex.4xlarge	7	40
c7i-flex.8xlarge	7	60
c7i-flex.12xlarge	7	120
c7i-flex.16xlarge	14	120
c8g.medium	1	4
c8g.large	2	10
c8g.xlarge	3	20
c8g.2xlarge	3	40
c8g.4xlarge	7	60
c8g.8xlarge	7	60
c8g.12xlarge	7	60
c8g.16xlarge	14	120
c8g.24xlarge	14	120
c8g.48xlarge	14	120
c8g.metal-24xl	14	120

執行個體類型	無ENI中繼的任務限制	含ENI中繼的任務限制
c8g.metal-48xl	14	120

記憶體最佳化

執行個體類型	無ENI中繼的任務限制	含ENI中繼的任務限制
r5.large	2	10
r5.xlarge	3	20
r5.2xlarge	3	40
r5.4xlarge	7	60
r5.12xlarge	7	60
r5.16xlarge	14	120
r5.24xlarge	14	120
r5a.large	2	10
r5a.xlarge	3	20
r5a.2xlarge	3	40
r5a.4xlarge	7	60
r5a.8xlarge	7	60
r5a.12xlarge	7	60
r5a.16xlarge	14	120
r5a.24xlarge	14	120
r5ad.large	2	10

執行個體類型	無ENI中繼的任務限制	含ENI中繼的任務限制
r5ad.xlarge	3	20
r5ad.2xlarge	3	40
r5ad.4xlarge	7	60
r5ad.8xlarge	7	60
r5ad.12xlarge	7	60
r5ad.16xlarge	14	120
r5ad.24xlarge	14	120
r5b.16xlarge	14	120
r5d.large	2	10
r5d.xlarge	3	20
r5d.2xlarge	3	40
r5d.4xlarge	7	60
r5d.8xlarge	7	60
r5d.12xlarge	7	60
r5d.16xlarge	14	120
r5d.24xlarge	14	120
r5dn.16xlarge	14	120
r6a.large	2	10
r6a.xlarge	3	20
r6a.2xlarge	3	40

執行個體類型	無ENI中繼的任務限制	含ENI中繼的任務限制
r6a.4xlarge	7	60
r6a.8xlarge	7	90
r6a.12xlarge	7	120
r6a.16xlarge	14	120
r6a.24xlarge	14	120
r6a.32xlarge	14	120
r6a.48xlarge	14	120
r6a.metal	14	120
r6g.medium	1	4
r6g.large	2	10
r6g.xlarge	3	20
r6g.2xlarge	3	40
r6g.4xlarge	7	60
r6g.8xlarge	7	60
r6g.12xlarge	7	60
r6g.16xlarge	14	120
r6g.metal	14	120
r6gd.medium	1	4
r6gd.large	2	10
r6gd.xlarge	3	20

執行個體類型	無ENI中繼的任務限制	含ENI中繼的任務限制
r6gd.2xlarge	3	40
r6gd.4xlarge	7	60
r6gd.8xlarge	7	60
r6gd.12xlarge	7	60
r6gd.16xlarge	14	120
r6gd.metal	14	120
r6i.large	2	10
r6i.xlarge	3	20
r6i.2xlarge	3	40
r6i.4xlarge	7	60
r6i.8xlarge	7	90
r6i.12xlarge	7	120
r6i.16xlarge	14	120
r6i.24xlarge	14	120
r6i.32xlarge	14	120
r6i.metal	14	120
r6idn.large	2	10
r6idn.xlarge	3	20
r6idn.2xlarge	3	40
r6idn.4xlarge	7	60

執行個體類型	無ENI中繼的任務限制	含ENI中繼的任務限制
r6idn.8xlarge	7	90
r6idn.12xlarge	7	120
r6idn.16xlarge	14	120
r6idn.24xlarge	14	120
r6idn.32xlarge	15	120
r6idn.metal	15	120
r6in.large	2	10
r6in.xlarge	3	20
r6in.2xlarge	3	40
r6in.4xlarge	7	60
r6in.8xlarge	7	90
r6in.12xlarge	7	120
r6in.16xlarge	14	120
r6in.24xlarge	14	120
r6in.32xlarge	15	120
r6in.metal	15	120
r6id.large	2	10
r6id.xlarge	3	20
r6id.2xlarge	3	40
r6id.4xlarge	7	60

執行個體類型	無ENI中繼的任務限制	含ENI中繼的任務限制
r6id.8xlarge	7	90
r6id.12xlarge	7	120
r6id.16xlarge	14	120
r6id.24xlarge	14	120
r6id.32xlarge	14	120
r6id.metal	14	120
r7a.medium	1	4
r7a.large	2	10
r7a.xlarge	3	20
r7a.2xlarge	3	40
r7a.4xlarge	7	60
r7a.8xlarge	7	90
r7a.12xlarge	7	120
r7a.16xlarge	14	120
r7a.24xlarge	14	120
r7a.32xlarge	14	120
r7a.48xlarge	14	120
r7a.metal-48xl	14	120
r7g.medium	1	4
r7g.large	2	10

執行個體類型	無ENI中繼的任務限制	含ENI中繼的任務限制
r7g.xlarge	3	20
r7g.2xlarge	3	40
r7g.4xlarge	7	60
r7g.8xlarge	7	60
r7g.12xlarge	7	60
r7g.16xlarge	14	120
r7g.metal	14	120
r7gd.medium	1	4
r7gd.large	2	10
r7gd.xlarge	3	20
r7gd.2xlarge	3	40
r7gd.4xlarge	7	60
r7gd.8xlarge	7	60
r7gd.12xlarge	7	60
r7gd.16xlarge	14	120
r7gd.metal	14	120
r7i.large	2	10
r7i.xlarge	3	20
r7i.2xlarge	3	40
r7i.4xlarge	7	60

執行個體類型	無ENI中繼的任務限制	含ENI中繼的任務限制
r7i.8xlarge	7	90
r7i.12xlarge	7	120
r7i.16xlarge	14	120
r7i.24xlarge	14	120
r7i.48xlarge	14	120
r7i.metal-24xl	14	120
r7i.metal-48xl	14	120
大	2	10
大	3	20
r7iz.2xlarge	3	40
r7iz.4xlarge	7	60
r7iz.8xlarge	7	90
r7iz.12xlarge	7	120
r7iz.16xlarge	14	120
r7iz.32xlarge	14	120
r7iz.metal-16xl	14	120
r7iz.metal-32xl	14	120
r8g.medium	1	4
r8g.large	2	10
r8g.xlarge	3	20

執行個體類型	無ENI中繼的任務限制	含ENI中繼的任務限制
r8g.2xlarge	3	40
r8g.4xlarge	7	60
r8g.8xlarge	7	60
r8g.12xlarge	7	60
r8g.16xlarge	14	120
r8g.24xlarge	14	120
r8g.48xlarge	14	120
r8g.metal-24xl	14	120
r8g.metal-48xl	14	120
u-3tb1.56xlarge	7	12
u-6tb1.56xlarge	14	12
u-18tb1.112xlarge	14	12
u-18tb1.metal	14	12
u-24tb1.112xlarge	14	12
u-24tb1.metal	14	12
u7i-6tb.112xlarge	14	120
u7i-8tb.112xlarge	14	120
u7i-12tb.224xlarge	14	120
u7in-16tb.224xlarge	15	120
u7in-24tb.224xlarge	15	120

執行個體類型	無ENI中繼的任務限制	含ENI中繼的任務限制
u7in-32tb.224xlarge	15	120
u7inh-32tb.480xlarge	15	120
x2gd.medium	1	10
x2gd.large	2	10
x2gd.xlarge	3	20
x2gd.2xlarge	3	40
x2gd.4xlarge	7	60
x2gd.8xlarge	7	60
x2gd.12xlarge	7	60
x2gd.16xlarge	14	120
x2gd.metal	14	120
x2idn.16xlarge	14	120
x2idn.24xlarge	14	120
x2idn.32xlarge	14	120
x2idn.metal	14	120
x2iedn.xlarge	3	13
x2iedn.2xlarge	3	29
x2iedn.4xlarge	7	60
x2iedn.8xlarge	7	120
x2iedn.16xlarge	14	120

執行個體類型	無ENI中繼的任務限制	含ENI中繼的任務限制
x2iedn.24xlarge	14	120
x2iedn.32xlarge	14	120
x2iedn.metal	14	120
x2iezn.2xlarge	3	64
x2iezn.4xlarge	7	120
x2iezn.6xlarge	7	120
x2iezn.8xlarge	7	120
x2iezn.12xlarge	14	120
x2iezn.metal	14	120
x8g.medium	1	4
x8g.large	2	10
x8g.xlarge	3	20
x8g.2xlarge	3	40
x8g.4xlarge	7	60
x8g.8xlarge	7	60
x8g.12xlarge	7	60
x8g.16xlarge	14	120
x8g.24xlarge	14	120
x8g.48xlarge	14	120
x8g.metal-24xl	14	120

執行個體類型	無ENI中繼的任務限制	含ENI中繼的任務限制
x8g.metal-48xl	14	120

儲存最佳化

執行個體類型	無ENI中繼的任務限制	含ENI中繼的任務限制
i4g.large	2	10
i4g.xlarge	3	20
i4g.2xlarge	3	40
i4g.4xlarge	7	60
i4g.8xlarge	7	60
i4g.16xlarge	14	120
i4i.xlarge	3	8
i4i.2xlarge	3	28
i4i.4xlarge	7	58
i4i.8xlarge	7	118
i4i.12xlarge	7	118
i4i.16xlarge	14	248
i4i.24xlarge	14	118
i4i.32xlarge	14	498
i4i.metal	14	498
i7ie.large	2	20

執行個體類型	無ENI中繼的任務限制	含ENI中繼的任務限制
i7ie.xlarge	3	29
i7ie.2xlarge	3	29
i7ie.3xlarge	3	29
i7ie.6xlarge	7	60
i7ie.12xlarge	7	60
i7ie.18xlarge	14	120
i7ie.24xlarge	14	120
i7ie.48xlarge	14	120
i8g.large	2	10
i8g.xlarge	3	20
i8g.2xlarge	3	40
i8g.4xlarge	7	60
i8g.8xlarge	7	60
i8g.12xlarge	7	60
i8g.16xlarge	14	120
i8g.24xlarge	14	120
i8g.metal-24xl	14	120
im4gn.large	2	10
im4gn.xlarge	3	20
im4gn.2xlarge	3	40

執行個體類型	無ENI中繼的任務限制	含ENI中繼的任務限制
im4gn.4xlarge	7	60
im4gn.8xlarge	7	60
im4gn.16xlarge	14	120
is4gen.medium	1	4
is4gen.large	2	10
is4gen.xlarge	3	20
is4gen.2xlarge	3	40
is4gen.4xlarge	7	60
is4gen.8xlarge	7	60

加速運算

執行個體類型	無ENI中繼的任務限制	含ENI中繼的任務限制
dl1.24xlarge	59	120
dl2q.24xlarge	14	120
f2.12xlarge	7	120
f2.48xlarge	14	120
g4ad.xlarge	1	12
g4ad.2xlarge	1	12
g4ad.4xlarge	2	12
g4ad.8xlarge	3	12

執行個體類型	無ENI中繼的任務限制	含ENI中繼的任務限制
g4ad.16xlarge	7	12
g5.xlarge	3	6
g5.2xlarge	3	19
g5.4xlarge	7	40
g5.8xlarge	7	90
g5.12xlarge	14	120
g5.16xlarge	7	120
g5.24xlarge	14	120
g5.48xlarge	6	120
g5g.xlarge	3	20
g5g.2xlarge	3	40
g5g.4xlarge	7	60
g5g.8xlarge	7	60
g5g.16xlarge	14	120
g5g.metal	14	120
g6.xlarge	3	20
g6.2xlarge	3	40
g6.4xlarge	7	60
g6.8xlarge	7	90
g6.12xlarge	7	120

執行個體類型	無ENI中繼的任務限制	含ENI中繼的任務限制
g6.16xlarge	14	120
g6.24xlarge	14	120
g6.48xlarge	14	120
g6e.xlarge	3	20
g6e.2xlarge	3	40
g6e.4xlarge	7	60
g6e.8xlarge	7	90
g6e.12xlarge	9	120
g6e.16xlarge	14	120
g6e.24xlarge	19	120
g6e.48xlarge	39	120
gr6.4xlarge	7	60
gr6.8xlarge	7	90
inf2.xlarge	3	20
inf2.8xlarge	7	90
inf2.24xlarge	14	120
inf2.48xlarge	14	120
p4d.24xlarge	59	120
p4de.24xlarge	59	120
p5.48xlarge	63	242

執行個體類型	無ENI中繼的任務限制	含ENI中繼的任務限制
p5e.48xlarge	63	242
p5en.48xlarge	63	242
trn1.2xlarge	3	19
trn1.32xlarge	39	120
trn1n.32xlarge	79	242
trn2.48xlarge	31	242
trn2u.48xlarge	31	242
vt1.3xlarge	3	40
vt1.6xlarge	7	60
vt1.24xlarge	14	120

高效能運算

執行個體類型	無ENI中繼的任務限制	含ENI中繼的任務限制
hpc6a.48xlarge	1	120
hpc6id.32xlarge	1	120
hpc7g.4xlarge	3	120
hpc7g.8xlarge	3	120
hpc7g.16xlarge	3	120

保留 Amazon ECS Linux 容器執行個體記憶體

當 Amazon ECS 容器代理程式向叢集註冊容器執行個體時，代理程式必須判斷容器執行個體可以為您的任務預留多少記憶體。由於存在平台記憶體額外負荷和由系統核心佔用的記憶體，這數字會與

Amazon EC2 執行個體公告的安裝記憶體數量不同。舉例而言，m4.large 執行個體安裝了 8 GiB 的記憶體。不過，這不一定會轉換為容器執行個體註冊時可用於任務的確切 8192 MiB 記憶體。

Amazon ECS 容器代理程式提供名為 的組態變數ECS_RESERVED_MEMORY，可用來從配置給您任務的集區中移除指定數量 MiB 的記憶體。這可為重要系統程序有效地預留記憶體。

如果您將容器執行個體上的所有記憶體佔用您的任務，則您的任務可能會與記憶體的關鍵系統程序競爭，並可能啟動系統故障。

舉例而言，若在容器代理程式檔案中指定 ECS_RESERVED_MEMORY=256，則代理程式會將記憶體總量減去 256 MiB 再註冊給該執行個體，而 256 MiB 的記憶體就無法由 ECS 任務配置。如需代理程式組態變數的詳細資訊及設定方式，請參閱 Amazon ECS 容器代理程式組態 和 引導 Amazon ECS Linux 容器執行個體以傳遞資料。

如果您為任務指定 8192 MiB，而且您的容器執行個體都沒有 8192 MiB 或更高的記憶體可滿足此要求，則任務無法放置在叢集中。如果您使用的是受管運算環境，則 AWS Batch 必須啟動較大的執行個體類型以容納請求。

也應預留一些記憶體供 Amazon ECS 容器代理程式以及容器執行個體上其他重要的系統程序使用，任務的容器才不會彼此爭奪相同的記憶體，而導致引發系統故障的可能。

Amazon ECS 容器代理程式會使用 Docker ReadMemInfo() 函數來查詢作業系統可用的記憶體總量。Linux 和 Windows 都提供命令列公用程式來判斷總記憶體。

Example - 判定 Linux 記憶體總量

free 命令會傳回作業系統辨識出的記憶體總量。

```
$ free -b
```

執行 Amazon ECS 最佳化 Amazon Linux AMI 的 m4.large 執行個體的輸出範例。

```
             total       used       free     shared    buffers     cached
Mem:    8373026816  348180480 8024846336      90112   25534464  205418496
-/+ buffers/cache:  117227520 8255799296
```

執行個體的記憶體總量有 8373026816 位元組，轉為任務可使用的記憶體則有 7985 MiB。

Example - 判定 Windows 記憶體總量

wmic 命令會傳回作業系統辨識出的記憶體總量。

```
C:\> wmic ComputerSystem get TotalPhysicalMemory
```

執行 Amazon ECS 最佳化 Windows Server AMI 之`m4.large`執行個體的範例輸出。

```
TotalPhysicalMemory
8589524992
```

執行個體的記憶體總量有 8589524992 位元組，轉為任務可使用的記憶體則有 8191 MiB。

檢視容器執行個體記憶體

您可以在 Amazon ECS 主控台 （或使用 DescribeContainerInstances API 操作） 中檢視容器執行個體向 註冊的記憶體數量。如果您嘗試為特定執行個體類型提供盡可能多的記憶體，以最大限度地提高資源使用率，您可以觀察該容器執行個體可用的記憶體，然後為您的任務指派多的記憶體。

檢視容器執行個體記憶體

1. 開啟主控台，網址為 https://console.aws.amazon.com/ecs/v2。
2. 在導覽窗格中，選擇叢集，然後選擇託管容器執行個體的叢集。
3. 選擇基礎設施，然後在容器執行個體下選擇容器執行個體。
4. 資源區段顯示容器執行個體的已註冊和可用記憶體。

   已註冊的記憶體值是容器執行個體；在第一次啟動時向 Amazon ECS 註冊，而可用的記憶體值是尚未配置給任務的值。

使用 從遠端管理 Amazon ECS 容器執行個體 AWS Systems Manager

您可以使用 AWS Systems Manager (Systems Manager) 中的執行命令功能來安全且遠端地管理 Amazon ECS 容器執行個體的組態。Run Command 提供執行常見管理任務的簡單方法，而不需在本機登入執行個體。您可以同時對多個容器執行個體執行命令，跨叢集管理組態變更。Run Command 會報告每個命令的狀態和結果。

以下是您可以使用 Run Command 執行的一些任務類型範例：

- 安裝或解除安裝套件。
- 執行安全性更新。
- 清除 Docker 映像。

- 停止或啟動服務。
- 檢視系統資源。
- 檢視日誌檔。
- 執行檔案操作。

如需執行命令的詳細資訊，請參閱 AWS Systems Manager 使用者指南中的 AWS Systems Manager 執行命令。

以下是將 Systems Manager 與 Amazon ECS 搭配使用的 p 必要條件。

1. 您必須授予容器執行個體角色 (ecsInstanceRole) 存取 Systems Manager APIs許可。您可以將 AmazonSSMManagedInstanceCore 指派給`ecsInstanceRole`角色來執行此操作。如需有關如何將政策連接至角色的資訊，請參閱AWS Identity and Access Management 《 使用者指南》中的更新角色的許可
2. 確認 SSM Agent 安裝於您的容器執行個體上。如需詳細資訊，請參閱在 Linux 的 EC2 執行個體上手動安裝和解除安裝 SSM Agent。

將 Systems Manager 受管政策連接至 `ecsInstanceRole`並確認 AWS Systems Manager 代理程式 (SSM 代理程式） 已安裝在您的容器執行個體之後，您就可以開始使用 Run Command 將命令傳送至您的容器執行個體。如需有關在執行個體上執行命令和 shell 指令碼並檢視結果輸出的詳細資訊，請參閱 AWS Systems Manager 使用者指南中的使用 Systems Manager Run Command 執行命令和執行命令演練。

常見的使用案例是使用 Run Command 更新容器執行個體軟體。您可以使用下列參數，遵循 AWS Systems Manager 使用者指南中的程序。

參數	Value
命令文件	`AWS-RunShellScript`
命令	`$ yum update -y`
目標執行個體	您的容器執行個體

針對 Amazon ECS Linux 容器執行個體使用 HTTP 代理

您可以設定您的 Amazon ECS 容器執行個體，針對 Amazon ECS 容器代理程式和 Docker 常駐程式使用 HTTP 代理。這在您的容器執行個體沒有透過 Amazon VPC 網際網路閘道、NAT 閘道或執行個體進行外部網路存取時會很有用。

若要設定您的 Amazon ECS Linux 容器執行個體以使用 HTTP 代理，請在啟動時 (使用 Amazon EC2 使用者資料) 設定相關檔案中的下列變數。您也可以手動編輯組態檔案，然後重新啟動代理程式。

`/etc/ecs/ecs.config` (Amazon Linux 2 和 AmazonLinux AMI)

`HTTP_PROXY=10.0.0.131:3128`

將此數值設為要讓 Amazon ECS 代理程式連線到網際網路的 HTTP 代理之主機名稱 (或 IP 地址) 及連接埠號碼。例如，您的容器執行個體可能沒有透過 Amazon VPC 網際網路閘道、NAT 閘道或執行個體的外部網路存取。

`NO_PROXY=169.254.169.254,169.254.170.2,/var/run/docker.sock`

將此值設為 `169.254.169.254,169.254.170.2,/var/run/docker.sock`，篩選 EC2 執行個體中繼資料、任務的 IAM 角色，以及來自代理的 Docker 常駐程式流量。

`/etc/systemd/system/ecs.service.d/http-proxy.conf` (僅限 Amazon Linux 2)

`Environment="HTTP_PROXY=10.0.0.131:3128/"`

將此數值設為要讓 `ecs-init` 連線到網際網路的 HTTP 代理之主機名稱 (或 IP 地址) 及連接埠號碼。例如，您的容器執行個體可能沒有透過 Amazon VPC 網際網路閘道、NAT 閘道或執行個體的外部網路存取。

```
Environment="NO_PROXY=169.254.169.254,169.254.170.2,/var/run/
docker.sock"
```

將此值設為 `169.254.169.254,169.254.170.2,/var/run/docker.sock`，篩選 EC2 執行個體中繼資料、任務的 IAM 角色，以及來自代理的 Docker 常駐程式流量。

`/etc/init/ecs.override` (僅限 Amazon Linux AMI)

`env HTTP_PROXY=10.0.0.131:3128`

將此數值設為要讓 `ecs-init` 連線到網際網路的 HTTP 代理之主機名稱 (或 IP 地址) 及連接埠號碼。例如，您的容器執行個體可能沒有透過 Amazon VPC 網際網路閘道、NAT 閘道或執行個體的外部網路存取。

`env NO_PROXY=169.254.169.254,169.254.170.2,/var/run/docker.sock`

將此值設為 `169.254.169.254,169.254.170.2,/var/run/docker.sock`，篩選 EC2 執行個體中繼資料、任務的 IAM 角色，以及來自代理的 Docker 常駐程式流量。

`/etc/systemd/system/docker.service.d/http-proxy.conf` (僅限 Amazon Linux 2)

`Environment="HTTP_PROXY=http://10.0.0.131:3128"`

將此數值設為要讓 Docker 常駐程式連線到網際網路的 HTTP 代理之主機名稱 (或 IP 地址) 及連接埠號碼。例如，您的容器執行個體可能沒有透過 Amazon VPC 網際網路閘道、NAT 閘道或執行個體的外部網路存取。

`Environment="NO_PROXY=169.254.169.254,169.254.170.2"`

將此數值設為 `169.254.169.254,169.254.170.2`，篩選來自代理的 EC2 執行個體中繼資料。

`/etc/sysconfig/docker` (僅限 Amazon Linux AMI 和 Amazon Linux 2)

`export HTTP_PROXY=http://10.0.0.131:3128`

將此數值設為要讓 Docker 常駐程式連線到網際網路的 HTTP 代理之主機名稱 (或 IP 地址) 及連接埠號碼。例如，您的容器執行個體可能沒有透過 Amazon VPC 網際網路閘道、NAT 閘道或執行個體的外部網路存取。

`export NO_PROXY=169.254.169.254,169.254.170.2`

將此數值設為 `169.254.169.254,169.254.170.2`，篩選來自代理的 EC2 執行個體中繼資料。

設定上述檔案中的這些環境變數只會影響 Amazon ECS 容器代理、`ecs-init` 及 Docker 常駐程式。他們不會設定任何其他服務 (例如 yum) 使用代理。

如需如何沒收代理的資訊，請參閱如何在 Amazon Linux 2 或 AL2023 中設定 Docker 和 Amazon ECS 容器代理程式的 HTTP 代理。

為您的 Amazon ECS Auto Scaling 群組設定預先初始化的執行個體

Amazon ECS 支援 Amazon EC2 Auto Scaling 暖集區。暖集區是一組準備投入使用的預先初始化 Amazon EC2 執行個體。每當應用程式需要水平擴展時，Amazon EC2 Auto Scaling 都會使用暖集區中的預初始化執行個體 (而不是啟動冷執行個體)，讓任何最終初始化程序執行，然後將執行個體投入使用。

若要瞭解有關暖集區以及如何將暖集區新增到 Auto Scaling 群組中的詳細資訊，請參閱《Amazon EC2 Auto Scaling 使用者指南》中的 Amazon EC2 Auto Scaling 的暖集區。

當您為 Amazon ECS 的 Auto Scaling 群組建立或更新暖集區時，無法設定將執行個體傳回縮減暖集區的選項 (`ReuseOnScaleIn`)。如需詳細資訊，請參閱 AWS Command Line Interface 參考中的 put-warm-pool。

若要將暖集區與 Amazon ECS 叢集配合使用，請在 Amazon EC2 Auto Scaling 群組啟動範本 User data (使用者資料) 欄位中將 `ECS_WARM_POOLS_CHECK` 代理程式組態變數設定為 `true`。

下列示範如何在 Amazon EC2 啟動範本的 User data (使用者資料) 欄位中指定代理程式組態變數。將 *`MyCluster`* 取代為您的叢集名稱。

```
#!/bin/bash
cat <<'EOF' >> /etc/ecs/ecs.config
ECS_CLUSTER=MyCluster
ECS_WARM_POOLS_CHECK=true
EOF
```

僅代理程式版本 `1.59.0` 和更新版本支援 `ECS_WARM_POOLS_CHECK` 變數。如需變數的詳細資訊，請參閱 Amazon ECS 容器代理程式組態。

更新 Amazon ECS 容器代理程式

有時，您可能需要更新 Amazon ECS 容器代理程式，才能取得錯誤修正和新功能。更新 Amazon ECS 容器代理不會中斷容器執行個體上正在執行中的任務或服務。更新代理程式的程序有所不同，這取決於您是否使用 Amazon ECS 最佳化 AMI 來啟動容器執行個體，或是否是在其他作業系統上。

> **Note**
>
> 代理更新不適用於 Windows 容器執行個體。我們建議您啟動新的容器執行個體，以更新您 Windows 叢集中的代理版本。

檢查 Amazon ECS 容器代理程式版本

您可以檢查在您的容器執行個體上執行的容器代理版本，以查看是否需要更新它。Amazon ECS 主控台中的容器執行個體檢視可提供代理程式版本。請使用下列步驟來檢查您的代理版本。

Amazon ECS console

1. 開啟主控台，網址為 https://console.aws.amazon.com/ecs/v2。
2. 從導覽列中選擇註冊外部執行個體所在的區域。
3. 在導覽窗格中選擇 Clusters (叢集)，並選取託管外部執行個體的叢集。
4. 在 Cluster : ***name*** (叢集：名稱) 頁面上，選擇 Infrastructure (基礎基礎設施) 索引標籤。
5. 在 Container instances (容器執行個體) 下，注意您容器執行個體的 Agent version (代理程式版本) 資料欄。如果容器執行個體不包含最新版的容器代理，主控台會使用訊息和標記提醒您過時的代理版本。

   若您的代理版本已過時，可以使用以下程序更新容器代理程式：

   - 若您的容器執行個體執行的是 Amazon ECS 最佳化 AMI，請參閱 在 Amazon ECS 最佳化 AMI 上更新 Amazon ECS 容器代理。
   - 若您的容器執行個體執行的不是 Amazon ECS 最佳化 AMI，請參閱 手動更新 Amazon ECS 容器代理程式 (適用於非 Amazon ECS 最佳化 AMI)。

   > **Important**
   >
   > 若要在您的 Amazon ECS 最佳化 AMI 上，更新 v1.0.0 版本之前的 Amazon ECS 代理程式版本，我們建議您終止目前的容器執行個體，並使用最近的 AMI 版本啟動新的執行個體。任何使用預覽版本的容器執行個體都應進行淘汰，並使用最近的 AMI 取代。如需詳細資訊，請參閱啟動 Amazon ECS Linux 容器執行個體。

Amazon ECS container agent introspection API

您也可以使用 Amazon ECS 容器代理程式從容器執行個體自我檢查 API 版本。如需詳細資訊，請參閱Amazon ECS 容器簡介。

若要使用自我檢查 API 檢查您的 Amazon ECS 容器代理是否執行最新版本

1. 透過 SSH 登入您的容器執行個體。
2. 查詢自我檢查 API。

```
[ec2-user ~]$ curl -s 127.0.0.1:51678/v1/metadata | python3 -mjson.tool
```

> **Note**
>
> 自我檢查 API 在 Amazon ECS 容器代理版本 v1.0.0 中新增 `Version` 資訊。若在查詢自我檢查 API 時沒有看見 `Version`，或是您的代理中甚至沒有自我檢查 API，則您執行的版本便是 v0.0.3 及更早版本。您應更新您的版本。

在 Amazon ECS 最佳化 AMI 上更新 Amazon ECS 容器代理

若您使用的是 Amazon ECS 最佳化 AMI，您有幾個選項可取得最新版本的 Amazon ECS 容器代理程式 (以下顯示的順序為建議順序)：

- 終止容器執行個體，並啟動最新版本的 Amazon ECS 最佳化 Amazon Linux 2 AMI (以手動方式或使用最新 AMI 更新您的 Auto Scaling 啟動組態)。這可提供全新的容器執行個體，以及最新已測試和驗證的 Amazon Linux、Docker、`ecs-init` 和 Amazon ECS 容器代理程式的版本。如需詳細資訊，請參閱Amazon ECS 最佳化 Linux AMIs。
- 使用 SSH 連線到執行個體，將 `ecs-init` 套裝服務 (及其相依性) 更新到最新版本。此操作可提供 Amazon Linux 儲存庫中目前經過測試及驗證的 Docker 和 `ecs-init` 版本，以及最新版 Amazon ECS 容器代理程式。如需詳細資訊，請參閱若要更新 Amazon ECS 最佳化 AMI 上的 `ecs-init` 套件。
- 使用 `UpdateContainerAgent` API 操作更新容器代理程式，無論是透過 主控台，或是使用 AWS CLI AWS SDKs 如需詳細資訊，請參閱使用 `UpdateContainerAgent` API 操作更新 Amazon ECS 容器代理程式。

> **Note**
>
> 代理更新不適用於 Windows 容器執行個體。我們建議您啟動新的容器執行個體，以更新您 Windows 叢集中的代理版本。

若要更新 Amazon ECS 最佳化 AMI 上的 **`ecs-init`** 套件

1. 透過 SSH 登入您的容器執行個體。
2. 使用以下命令更新 `ecs-init` 套裝服務。

```
sudo yum update -y ecs-init
```

> **Note**
>
> `ecs-init` 套件和 Amazon ECS 容器代理程式會立即更新。但是，直到 Docker 常駐程式重新啟動前，都不會載入較新版本的 Docker。將執行個體重新開機，或在執行個體上執行下列命令，以重新啟動：
>
> - Amazon ECS 最佳化 Amazon Linux 2 AMI：
>
>   ```
>   sudo systemctl restart docker
>   ```
>
> - Amazon ECS 最佳化 Amazon Linux AMI：
>
>   ```
>   sudo service docker restart && sudo start ecs
>   ```

使用 **`UpdateContainerAgent`** API 操作更新 Amazon ECS 容器代理程式

> **Important**
>
> 僅在 Amazon ECS 最佳化 AMI 的 Linux 變體上支援 `UpdateContainerAgent` API，但 Amazon ECS 最佳化 Amazon Linux 2 (arm64) AMI 除外。對於使用 Amazon ECS 最佳化 Amazon Linux 2 (arm64) AMI 的容器執行個體，請更新 `ecs-init` 套件以更新代理程式。針對在其他作業系統上執行的容器執行個體，請參閱「手動更新 Amazon ECS 容器代理程式 (適用於非 Amazon ECS 最佳化 AMI)」。如果您正在使用 Windows 容器執行個體，我們建議您啟動新的容器執行個體，以更新您 Windows 叢集中的代理程式版本。

當您透過主控台或使用 AWS CLI AWS SDKs 請求代理程式更新時，`UpdateContainerAgent`API 程序就會開始。Amazon ECS 會根據最新的可用代理程式版本檢查您目前的代理程式版本，以及是否可以更新。若無法取得更新 (例如若代理已在執行最近的版本)，便會傳回 `NoUpdateAvailableException`。

上圖顯示更新程序中的階段如下：

`PENDING`

有可用的代理更新，並已啟動更新程序。

`STAGING`

代理已開始下載代理更新。若代理無法下載更新，或更新的內容不正確或已損毀，則代理會傳送失敗的通知，且更新的狀態會轉換成 `FAILED` 狀態。

`STAGED`

代理下載已完成並已驗證代理內容。

`UPDATING`

`ecs-init` 服務已重新啟動，並使用最新版本的代理。若代理程式因為某些原因無法重新啟動，則更新會轉換為 `FAILED` 狀態；否則，代理程式會通知 Amazon ECS 更新已完成。

> **Note**
>
> 代理更新不適用於 Windows 容器執行個體。我們建議您啟動新的容器執行個體，以更新您 Windows 叢集中的代理版本。

若要在主控台的 Amazon ECS 最佳化 AMI 上更新 Amazon ECS 容器代理程式

1. 開啟主控台，網址為 https://console.aws.amazon.com/ecs/v2。
2. 從導覽列中選擇註冊外部執行個體所在的區域。
3. 在導覽窗格中，選擇 Clusters (叢集)，然後選取叢集。
4. 在 Cluster : ***name*** (叢集：名稱) 頁面上，選擇 Infrastructure (基礎基礎設施) 索引標籤。
5. 在容器執行個體下，選取要更新的執行個體，然後選擇動作、更新代理程式。

手動更新 Amazon ECS 容器代理程式 (適用於非 Amazon ECS 最佳化 AMI)

有時，您可能需要更新 Amazon ECS 容器代理程式，才能取得錯誤修正和新功能。更新 Amazon ECS 容器代理不會中斷容器執行個體上正在執行中的任務或服務。

> **Note**
>
> 代理更新不適用於 Windows 容器執行個體。我們建議您啟動新的容器執行個體，以更新您 Windows 叢集中的代理版本。

1. 透過 SSH 登入您的容器執行個體。
2. 檢查以查看您的代理是否是使用 ECS_DATADIR 環境變數儲存其狀態。

```
ubuntu:~$ docker inspect ecs-agent | grep ECS_DATADIR
```

輸出：

```
"ECS_DATADIR=/data",
```

> **Important**
>
> 若先前的命令並未傳回 ECS_DATADIR 環境變數，您必須停止任何在此容器執行個體上執行的任務，才能更新您的代理。較新的代理會使用 ECS_DATADIR 環境變數儲存其狀態，讓您可以在任務執行中時更新它們，而不會有任何問題。

3. 停用 Amazon ECS 容器代理程式。

```
ubuntu:~$ docker stop ecs-agent
```

4. 刪除代理容器。

```
ubuntu:~$ docker rm ecs-agent
```

5. 確認 /etc/ecs 目錄和 Amazon ECS 容器代理程式組態檔案存在於 /etc/ecs/ecs.config。

```
ubuntu:~$ sudo mkdir -p /etc/ecs && sudo touch /etc/ecs/ecs.config
```

6. 編輯 /etc/ecs/ecs.config 檔案，並確保其至少包含以下變數宣告。若您不希望您的容器執行個體使用預設叢集註冊，請將叢集名稱指定為 ECS_CLUSTER 的值。

```
ECS_DATADIR=/data
ECS_ENABLE_TASK_IAM_ROLE=true
ECS_ENABLE_TASK_IAM_ROLE_NETWORK_HOST=true
ECS_LOGFILE=/log/ecs-agent.log
ECS_AVAILABLE_LOGGING_DRIVERS=["json-file","awslogs"]
ECS_LOGLEVEL=info
ECS_CLUSTER=default
```

如需這些和其他代理執行時間選項的詳細資訊，請參閱「Amazon ECS 容器代理程式組態」。

> **Note**
>
> 您可以選擇性地將您的代理程式環境變數存放在 Amazon S3 中 (可在啟動時使用 Amazon EC2 使用者資料將其下載到您的容器執行個體)。針對敏感性資訊 (例如私有存放庫的身分驗證登入資料)，此為建議選項。如需詳細資訊，請參閱在 Amazon S3 中存放 Amazon ECS 容器執行個體組態及在 Amazon ECS 中使用非AWS 容器映像。

7. 從 Amazon Elastic Container Registry Public 取出最新的 Amazon ECS 容器代理程式映像。

```
ubuntu:~$ docker pull public.ecr.aws/ecs/amazon-ecs-agent:latest
```

輸出：

```
Pulling repository amazon/amazon-ecs-agent
a5a56a5e13dc: Download complete
511136ea3c5a: Download complete
9950b5d678a1: Download complete
c48ddcf21b63: Download complete
Status: Image is up to date for amazon/amazon-ecs-agent:latest
```

8. 在您的容器執行個體上執行最新的 Amazon ECS 容器代理程式。

> **Note**
>
> 使用 Docker 重新啟動政策或處理序管理員 (例如 upstart 或 systemd) 將容器代理程式做為服務或常駐程式處理，並確保在結束後重新啟動它。因此 Amazon ECS 最佳化 AMI 會使用 `ecs-init` RPM，您可以在 GitHub 上檢視此 RPM 的來源程式碼。

以下代理程式執行命令範例分成獨立的各行來顯示每個選項。如需這些和其他代理執行時間選項的詳細資訊，請參閱「Amazon ECS 容器代理程式組態」。

> **Important**
>
> 針對啟用 SELinux 的作業系統，需要在您的 docker run 命令中包含 `--privileged` 選項。此外，針對啟用 SELinux 的容器執行個體，我們建議您為 `/log` 和 `/data` 磁碟區掛載新增 `:Z` 選項。但是，這些磁碟區的主機掛載必須在您執行命令前存在，否則您會接收

到 `no such file or directory` 錯誤。若您在啟用 SELinux 的容器執行個體上執行 Amazon ECS 代理程式時遭遇困難，請採取以下動作：

- 在您的容器執行個體上建立主機磁碟區掛載點。

```
ubuntu:~$ sudo mkdir -p /var/log/ecs /var/lib/ecs/data
```

- 為以下 docker run 命令新增 `--privileged` 選項。
- 為以下 docker run 命令的 `/log` 和 `/data` 容器磁碟區掛載附加 `:Z` 選項 (例如，`--volume=/var/log/ecs/:/log:Z`)。

```
ubuntu:~$ sudo docker run --name ecs-agent \
--detach=true \
--restart=on-failure:10 \
--volume=/var/run:/var/run \
--volume=/var/log/ecs/:/log \
--volume=/var/lib/ecs/data:/data \
--volume=/etc/ecs:/etc/ecs \
--volume=/etc/ecs:/etc/ecs/pki \
--net=host \
--env-file=/etc/ecs/ecs.config \
amazon/amazon-ecs-agent:latest
```

> **Note**
>
> 若您接收到 `Error response from daemon: Cannot start container` 訊息，您可以使用 sudo docker rm ecs-agent 命令刪除失敗的容器，並再度嘗試執行代理程式。

Amazon ECS 最佳化 Windows AMIs

Amazon ECS 最佳化 AMI 已預先設定好執行 Amazon ECS 工作負載所需的必要元件。雖然您可以建立自己的容器執行個體 AMI，以符合在 Amazon ECS 上執行容器化工作負載所需的基本規格，但 Amazon ECS 最佳化 AMIs 是由 AWS 工程師在 Amazon ECS 上預先設定和測試。這是讓您快速上手並在 AWS 上執行容器最簡單的方法。

可以透過程式設計方式擷取每個變體的 Amazon ECS 最佳化 AMI 中繼資料，包括 AMI 名稱、Amazon ECS 容器代理程式版本，以及包含 Docker 版本的 Amazon ECS 執行時間版本。如需詳細資訊，請參閱the section called “擷取 Amazon ECS 最佳化 Windows AMI 中繼資料”。

> **Important**
>
> 8 月之後生產的所有 ECS 最佳化 AMI 變體，將從 Docker EE (Mirantis) 遷移至 Docker CE (Moby 專案)。
> 為確保客戶預設擁有最新的安全更新，Amazon ECS 至少維護最後三個 Windows Amazon ECS 最佳化 AMI。在發佈新的 Windows Amazon ECS 最佳化 AMI 之後，Amazon ECS 使舊的 Windows Amazon ECS 最佳化 AMI 變為私有。如果有您需要存取的私有 AMI，請向 Cloud Support 提出票證，從而通知我們。

Amazon ECS 最佳化 AMI 變體

下列 Amazon ECS 最佳化 AMI 的 Windows Server 變體適用於您的 Amazon EC2 執行個體。

> **Important**
>
> 8 月之後生產的所有 ECS 最佳化 AMI 變體，將從 Docker EE (Mirantis) 遷移至 Docker CE (Moby 專案)。

- Amazon ECS 最佳化 Windows Server 2022 Full AMI
- Amazon ECS 最佳化 Windows Server 2022 Core AMI
- Amazon ECS 最佳化 Windows Server 2019 Full AMI
- Amazon ECS 最佳化 Windows Server 2019 Core AMI
- Amazon ECS 最佳化 Windows Server 2016 Full AMI

> **Important**
>
> Windows Server 2016 不支援最新的 Docker 版本，例如 25.x.x。因此，Windows Server 2016 Full AMIs 將不會收到 Docker 執行時間的安全或錯誤修補程式。我們建議您移至下列其中一個 Windows 平台：
>
> - Windows Server 2022 Full

- Windows Server 2022 Core
- Windows Server 2019 Full
- Windows Server 2019 Core

在 2022 年 8 月 9 日，Amazon ECS 最佳化的 Windows Server 20H2 Core AMI 已屆支援終止日期。不再發佈此 AMI 的新版本。如需詳細資訊，請參閱 Windows Server 發行版本資訊。

Windows Server 2022、Windows Server 2019 和 Windows Server 2016 是長期維護通道 (LTSC) 的發行版本。Windows Server 20H2 是半年通道 (SAC) 的發行版本。如需詳細資訊，請參閱 Windows Server 發行版本資訊。

考量事項

以下是一些您應該知道的 Amazon EC2 Windows 容器和 Amazon ECS 有關事項。

- Windows 容器無法在 Linux 容器執行個體上執行，反之亦然。為更合理放置 Windows 和 Linux 任務，將 Windows 和 Linux 容器執行個體放置在不同的叢集中，而且只在 Windows 叢集中放置 Windows 任務。您可以設定下列置放限制：`memberOf(ecs.os-type=='windows')`，以確保 Windows 任務定義只放置在 Windows 執行個體。
- Windows 容器支援使用 EC2 和 Fargate 啟動類型的任務。
- Windows 容器和容器執行個體無法支援所有可供 Linux 容器和容器執行個體使用的任務定義參數。有些參數完全不予以支援，有些參數在 Windows 的行為表現和在 Linux 上不同。如需詳細資訊，請參閱執行 Windows 之 EC2 執行個體的 Amazon ECS 任務定義差異。
- 針對任務功能的 IAM 角色，您需要設定您的 Windows 容器執行個體，在啟動時允許該功能。您的容器必須在它們使用該功能時執行部分提供的 PowerShell 程式碼。如需詳細資訊，請參閱 Amazon EC2 Windows 執行個體額外組態。
- 任務的 IAM 角色功能會使用登入資料代理將登入資料提供給容器。此登入資料代理佔用了容器執行個體的連接埠 80，所以如果您使用任務的 IAM 角色，連接埠 80 就不提供任務使用。對於 Web 服務容器，您可以使用 Application Load Balancer 和動態連接埠映射，向您的容器提供標準的 HTTP 連接埠 80 連線。如需詳細資訊，請參閱使用負載平衡來分發 Amazon ECS 服務流量。
- Windows 伺服器 Docker 映像很大 (9 GiB)。所以您的 Windows 容器執行個體需要比 Linux 容器執行個體還多的儲存空間。
- 若要在 Windows 伺服器上執行 Windows 容器，容器的基本映像作業系統版本必須與主機的版本相符。如需詳細資訊，請參閱 Microsoft 文件網站上的 Windows 容器版本相容性。如果您的叢集執行多個 Windows 版本，您可以使用置放條件限制：`memberOf(attribute:ecs.os-family`

== WINDOWS_SERVER_<OS_Release>_<FULL or CORE>)，確保將任務置放在執行相同版本的 EC2 執行個體上。如需詳細資訊，請參閱the section called “擷取 Amazon ECS 最佳化 Windows AMI 中繼資料”。

擷取 Amazon ECS 最佳化 Windows AMI 中繼資料

透過查詢 Systems Manager 參數存放區 API，可以程式設計方式擷取 Amazon ECS 最佳化 AMI 的每個變體的 AMI ID、映像名稱、作業系統、容器代理程式版本以及執行時間版本。如需有關 Systems Manager 參數存放區 API 的詳細資訊，請參閱 GetParameters 和 GetParametersByPath。

> **Note**
>
> 您的管理使用者必須擁有以下 IAM 許可，才能擷取 Amazon ECS 最佳化 AMI 中繼資料。已將這些權限新增至 AmazonECS_FullAccess IAM 政策。
>
> - ssm:GetParameters
> - ssm:GetParameter
> - ssm:GetParametersByPath

Systems Manager 參數存放區參數格式。

> **Note**
>
> 下列 Systems Manager 參數存放區 API 參數已淘汰，不應該用來擷取最新的 Windows AMI：
>
> - `/aws/service/ecs/optimized-ami/windows_server/2016/english/full/recommended/image_id`
> - `/aws/service/ecs/optimized-ami/windows_server/2019/english/full/recommended/image_id`

以下是每個 Amazon ECS 最佳化 AMI 變體的參數名稱格式。

- Windows Server 2022 Full AMI 中繼資料：

```
/aws/service/ami-windows-latest/Windows_Server-2022-English-Full-ECS_Optimized
```

- Windows Server 2022 Core AMI 中繼資料：

```
/aws/service/ami-windows-latest/Windows_Server-2022-English-Core-ECS_Optimized
```

- Windows Server 2019 Full AMI 中繼資料：

```
/aws/service/ami-windows-latest/Windows_Server-2019-English-Full-ECS_Optimized
```

- Windows Server 2019 Core AMI 中繼資料：

```
/aws/service/ami-windows-latest/Windows_Server-2019-English-Core-ECS_Optimized
```

- Windows Server 2016 Full AMI 中繼資料：

```
/aws/service/ami-windows-latest/Windows_Server-2016-English-Full-ECS_Optimized
```

下列參數名稱格式擷取最新穩定 Windows Server 2019 Full AMI 的中繼資料。

```
aws ssm get-parameters --names /aws/service/ami-windows-latest/Windows_Server-2019-
English-Full-ECS_Optimized
```

以下是傳回參數值的 JSON 物件範例。

```
{
    "Parameters": [
        {
            "Name": "/aws/service/ami-windows-latest/Windows_Server-2019-English-Full-
ECS_Optimized",
            "Type": "String",
            "Value": "{\"image_name\":\"Windows_Server-2019-English-Full-
ECS_Optimized-2023.06.13\",\"image_id\":\"ami-0debc1fb48e4aee16\",\"ecs_runtime_version
\":\"Docker (CE) version 20.10.21\",\"ecs_agent_version\":\"1.72.0\"}",
            "Version": 58,
            "LastModifiedDate": "2023-06-22T19:37:37.841000-04:00",
            "ARN": "arn:aws:ssm:us-east-1::parameter/aws/service/ami-windows-latest/
Windows_Server-2019-English-Full-ECS_Optimized",
            "DataType": "text"
        }
    ],
    "InvalidParameters": []
}
```

輸出中的每個欄位可做為子參數查詢。透過將子參數名稱附加至所選 AMI 路徑來建構子參數的參數路徑。下列子參數可供使用：

- `schema_version`
- `image_id`
- `image_name`
- `os`
- `ecs_agent_version`
- `ecs_runtime_version`

範例

下列範例顯示您可擷取每個 Amazon ECS 最佳化 AMI 變體中繼資料的方法。

擷取最新的穩定 Amazon ECS 最佳化 AMI 的中繼資料

您可以使用 AWS CLI 搭配下列 AWS CLI 命令來擷取最新的穩定 Amazon ECS 最佳化 AMI。

- 對於 Amazon ECS 最佳化 Windows Server 2022 Full AMI：

```
aws ssm get-parameters --names /aws/service/ami-windows-latest/Windows_Server-2022-English-Full-ECS_Optimized --region us-east-1
```

- 對於 Amazon ECS 最佳化 Windows Server 2022 Core AMI：

```
aws ssm get-parameters --names /aws/service/ami-windows-latest/Windows_Server-2022-English-Core-ECS_Optimized --region us-east-1
```

- 對於 Amazon ECS 最佳化 Windows Server 2019 Full AMI：

```
aws ssm get-parameters --names /aws/service/ami-windows-latest/Windows_Server-2019-English-Full-ECS_Optimized --region us-east-1
```

- 對於 Amazon ECS 最佳化 Windows Server 2019 Core AMI：

```
aws ssm get-parameters --names /aws/service/ami-windows-latest/Windows_Server-2019-English-Core-ECS_Optimized --region us-east-1
```

- 對於 Amazon ECS 最佳化 Windows Server 2016 Full AMI：

```
aws ssm get-parameters --names /aws/service/ami-windows-latest/Windows_Server-2016-English-Full-ECS_Optimized --region us-east-1
```

在 AWS CloudFormation 範本中使用最新建議的 Amazon ECS 最佳化 AMI

透過參考 Systems Manager 參數存放區名稱，可以在 AWS CloudFormation 範本中參考最新建議的 Amazon ECS 最佳化 AMI。

```
Parameters:
  LatestECSOptimizedAMI:
    Description: AMI ID
    Type: AWS::SSM::Parameter::Value<AWS::EC2::Image::Id>
    Default: /aws/service/ami-windows-latest/Windows_Server-2019-English-Full-ECS_Optimized/image_id
```

Amazon ECS 最佳化 Windows AMI 版本

檢視 Amazon ECS 最佳化 AMIs 的目前和先前版本，以及 Amazon ECS 容器代理程式、Docker 和`ecs-init`套件的對應版本。

可透過程式設計方式擷取每個變體的 Amazon ECS 最佳化 AMI 中繼資料 (包括 AMI ID)。如需詳細資訊，請參閱the section called “擷取 Amazon ECS 最佳化 Windows AMI 中繼資料”。

以下標籤顯示 Windows Amazon ECS 最佳化 AMI 版本清單。如需在 AWS CloudFormation 範本中參考 Systems Manager 參數存放區參數的詳細資訊，請參閱 在 AWS CloudFormation 範本中使用最新建議的 Amazon ECS 最佳化 AMI。

Important

為確保客戶預設擁有最新的安全更新，Amazon ECS 至少維護最後三個 Windows Amazon ECS 最佳化 AMI。在發佈新的 Windows Amazon ECS 最佳化 AMI 之後，Amazon ECS 使舊的 Windows Amazon ECS 最佳化 AMI 變為私有。如果有您需要存取的私有 AMI，請向 Cloud Support 提出票證，從而通知我們。

Windows Server 2016 不支援最新的 Docker 版本，例如 25.x.x。因此，Windows Server 2016 Full AMIs 將不會收到 Docker 執行時間的安全或錯誤修補程式。我們建議您移至下列其中一個 Windows 平台：

- Windows Server 2022 Full
- Windows Server 2022 Core

- Windows Server 2019 Full
- Windows Server 2019 Core

Windows Server 2022 Full AMI versions

下表列出目前和舊版的 Amazon ECS 最佳化 Windows Server 2022 Full AMI 的完整清單及其對應的 Amazon ECS 容器代理程式和 Docker 版本。

Amazon ECS 最佳化 Windows Server 2022 Full AMI	Amazon ECS 容器代理程式版本	Docker 版本	Visibility
Windows_Server-2022-English-Full-ECS_Optimized-2025.1.21	1.89.2	25.0.6 (Docker CE)	公有
Windows_Server-2022-English-Full-ECS_Optimized-2024.12.11	1.89.1	25.0.6 (Docker CE)	公有
Windows_Server-2022-English-Full-ECS_Optimized-2024.11.13	1.88.0	25.0.6 (Docker CE)	公有
Windows_Server-2022-English-Full-ECS_Optimized-2024.10.17	1.87.0	25.0.6 (Docker CE)	公有
Windows_Server-2022-English-Full-ECS_Optimized-2024.09.10	1.86.3	25.0.6 (Docker CE)	私有

Amazon ECS 最佳化 Windows Server 2022 Full AMI	Amazon ECS 容器代理程式版本	Docker 版本	Visibility
Windows_Server-2022-English-Full-ECS_Optimized-2024.08.19	1.86.2	25.0.6 (Docker CE)	私有
Windows_Server-2022-English-Full-ECS_Optimized-2024.07.09	1.84.0	25.0.3 (Docker CE)	私有
Windows_Server-2022-English-Full-ECS_Optimized-2024.06.14	1.83.0	25.0.3 (Docker CE)	私有
Windows_Server-2022-English-Full-ECS_Optimized-2024.05.14	1.82.3	25.0.3 (Docker CE)	私有
Windows_Server-2022-English-Full-ECS_Optimized-2024.04.09	1.82.2	25.0.3 (Docker CE)	私有
Windows_Server-2022-English-Full-ECS_Optimized-2024.03.12	1.82.0	20.10.23 (Docker CE)	私有
Windows_Server-2022-English-Full-ECS_Optimized-2024.02.13	1.81.0	20.10.23 (Docker CE)	私有

使用下列 AWS CLI 命令來擷取目前的 Amazon ECS 最佳化 Windows Server 2022 Full AMI。

```
aws ssm get-parameters --names /aws/service/ami-windows-latest/Windows_Server-2022-English-Full-ECS_Optimized
```

Windows Server 2022 Core AMI versions

下表列出目前和舊版的 Amazon ECS 最佳化 Windows Server 2022 Core AMI 的完整清單及其對應的 Amazon ECS 容器代理程式和 Docker 版本。

Amazon ECS 最佳化 Windows Server 2022 Core AMI	Amazon ECS 容器代理程式版本	Docker 版本	Visibility
Windows_Server-2022-English-Core-ECS_Optimized-2025.1.21	1.89.2	25.0.6 (Docker CE)	公有
Windows_Server-2022-English-Core-ECS_Optimized-2024.12.11	1.89.1	25.0.6 (Docker CE)	公有
Windows_Server-2022-English-Core-ECS_Optimized-2024.11.13	1.88.0	25.0.6 (Docker CE)	公有
Windows_Server-2022-English-Core-ECS_Optimized-2024.10.17	1.87.0	25.0.6 (Docker CE)	公有
Windows_Server-2022-English-Core-ECS_Optimized-2024.09.10	1.86.3	25.0.6 (Docker CE)	私有

Amazon ECS 最佳化 Windows Server 2022 Core AMI	Amazon ECS 容器代理程式版本	Docker 版本	Visibility
Windows_Server-2022-English-Core-ECS_Optimized-2024.08.19	1.86.2	25.0.6 (Docker CE)	私有
Windows_Server-2022-English-Core-ECS_Optimized-2024.07.09	1.84.0	25.0.3 (Docker CE)	私有
Windows_Server-2022-English-Core-ECS_Optimized-2024.06.14	1.83.0	25.0.3 (Docker CE)	私有
Windows_Server-2022-English-Core-ECS_Optimized-2024.05.14	1.82.3	25.0.3 (Docker CE)	私有
Windows_Server-2022-English-Core-ECS_Optimized-2024.04.09	1.82.2	25.0.3 (Docker CE)	私有
Windows_Server-2022-English-Core-ECS_Optimized-2024.03.12	1.82.0	20.10.23 (Docker CE)	私有
Windows_Server-2022-English-Core-ECS_Optimized-2024.02.13	1.81.0	20.10.23 (Docker CE)	私有

使用下列 AWS CLI 命令來擷取目前的 Amazon ECS 最佳化 Windows Server 2022 Full AMI。

```
aws ssm get-parameters --names /aws/service/ami-windows-latest/Windows_Server-2022-English-Core-ECS_Optimized
```

Windows Server 2019 Full AMI versions

下表列出目前和舊版的 Amazon ECS 最佳化 Windows Server 2019 Full AMI 的完整清單及其對應的 Amazon ECS 容器代理程式和 Docker 版本。

Amazon ECS 最佳化 Windows Server 2019 Full AMI	Amazon ECS 容器代理程式版本	Docker 版本	Visibility
Windows_Server-2019-English-Full-ECS_Optimized-2025.1.21	1.89.2	25.0.6 (Docker CE)	公有
Windows_Server-2019-English-Full-ECS_Optimized-2024.12.11	1.89.1	25.0.6 (Docker CE)	公有
Windows_Server-2019-English-Full-ECS_Optimized-2024.11.13	1.88.0	25.0.6 (Docker CE)	公有
Windows_Server-2019-English-Full-ECS_Optimized-2024.10.17	1.87.0	25.0.6 (Docker CE)	公有
Windows_Server-2019-English-Full-ECS_Optimized-2024.09.10	1.86.3	25.0.6 (Docker CE)	私有

Amazon ECS 最佳化 Windows Server 2019 Full AMI	Amazon ECS 容器代理程式版本	Docker 版本	Visibility
Windows_Server-2019-English-Full-ECS_Optimized-2024.08.16	1.86.2	25.0.6 (Docker CE)	私有
Windows_Server-2019-English-Full-ECS_Optimized-2024.07.09	1.84.0	25.0.3 (Docker CE)	私有
Windows_Server-2019-English-Full-ECS_Optimized-2024.06.14	1.83.0	25.0.3 (Docker CE)	私有
Windows_Server-2019-English-Full-ECS_Optimized-2024.05.14	1.82.3	25.0.3 (Docker CE)	私有
Windows_Server-2019-English-Full-ECS_Optimized-2024.04.09	1.82.2	25.0.3 (Docker CE)	私有
Windows_Server-2019-English-Full-ECS_Optimized-2024.03.12	1.82.0	20.10.23 (Docker CE)	私有
Windows_Server-2019-English-Full-ECS_Optimized-2024.02.13	1.81.0	20.10.23 (Docker CE)	私有

使用下列 AWS CLI 命令來擷取目前的 Amazon ECS 最佳化 Windows Server 2019 Full AMI。

```
aws ssm get-parameters --names /aws/service/ami-windows-latest/Windows_Server-2019-English-Full-ECS_Optimized
```

Windows Server 2019 Core AMI versions

下表列出目前和舊版的 Amazon ECS 最佳化 Windows Server 2019 Core AMI 的完整清單及其對應的 Amazon ECS 容器代理程式和 Docker 版本。

Amazon ECS 最佳化 Windows Server 2019 Core AMI	Amazon ECS 容器代理程式版本	Docker 版本	Visibility
Windows_Server-2019-English-Core-ECS_Optimized-2025.1.21	1.89.2	25.0.6 (Docker CE)	公有
Windows_Server-2019-English-Core-ECS_Optimized-2024.12.11	1.89.1	25.0.6 (Docker CE)	公有
Windows_Server-2019-English-Core-ECS_Optimized-2024.11.13	1.88.0	25.0.6 (Docker CE)	公有
Windows_Server-2019-English-Core-ECS_Optimized-2024.10.17	1.87.0	25.0.6 (Docker CE)	公有
Windows_Server-2019-English-Core-ECS_Optimized-2024.09.10	1.86.3	25.0.6 (Docker CE)	公有

Amazon ECS 最佳化 Windows Server 2019 Core AMI	Amazon ECS 容器代理程式版本	Docker 版本	Visibility
Windows_Server-2019-English-Core-ECS_Optimized-2024.08.19	1.86.2	25.0.6 (Docker CE)	私有
Windows_Server-2019-English-Core-ECS_Optimized-2024.07.09	1.84.0	25.0.3 (Docker CE)	私有
Windows_Server-2019-English-Core-ECS_Optimized-2024.06.14	1.83.0	25.0.3 (Docker CE)	私有
Windows_Server-2019-English-Core-ECS_Optimized-2024.05.14	1.82.3	25.0.3 (Docker CE)	私有
Windows_Server-2019-English-Core-ECS_Optimized-2024.04.09	1.82.2	25.0.3 (Docker CE)	私有
Windows_Server-2019-English-Core-ECS_Optimized-2024.03.12	1.82.0	20.10.23 (Docker CE)	私有
Windows_Server-2019-English-Core-ECS_Optimized-2024.02.13	1.81.0	20.10.23 (Docker CE)	私有

使用下列 AWS CLI 命令來擷取目前的 Amazon ECS 最佳化 Windows Server 2019 Full AMI。

```
aws ssm get-parameters --names /aws/service/ami-windows-latest/Windows_Server-2019-English-Core-ECS_Optimized
```

Windows Server 2016 Full AMI versions

> **Important**
>
> Windows Server 2016 不支援最新的 Docker 版本，例如 25.x.x。因此，Windows Server 2016 完整 AMIs 將不會收到 Docker 執行時間的安全或錯誤修補程式。我們建議您移至下列其中一個 Windows 平台：
>
> - Windows Server 2022 Full
> - Windows Server 2022 Core
> - Windows Server 2019 Full
> - Windows Server 2019 Core

下表列出目前和舊版的 Amazon ECS 最佳化 Windows Server 2016 Full AMI 的完整清單及其對應的 Amazon ECS 容器代理程式和 Docker 版本。

Amazon ECS 最佳化 Windows Server 2016 Full AMI	Amazon ECS 容器代理程式版本	Docker 版本	Visibility
Windows_Server-2016-English-Full-ECS_Optimized-2025.1.21	1.89.2	20.10.23 (Docker CE)	公有
Windows_Server-2016-English-Full-ECS_Optimized-2024.12.11	1.89.1	20.10.23 (Docker CE)	公有
Windows_Server-2016-English-Full-ECS	1.88.0	20.10.23 (Docker CE)	公有

Amazon ECS 最佳化 Windows Server 2016 Full AMI	Amazon ECS 容器代理程式版本	Docker 版本	Visibility
_Optimized-2024.11.13			
Windows_Server-2016-English-Full-ECS_Optimized-2024.10.17	1.87.0	20.10.23 (Docker CE)	公有
Windows_Server-2016-English-Full-ECS_Optimized-2024.09.10	1.86.3	20.10.23 (Docker CE)	私有
Windows_Server-2016-English-Full-ECS_Optimized-2024.08.19	1.86.2	20.10.23 (Docker CE)	私有
Windows_Server-2016-English-Full-ECS_Optimized-2024.07.09	1.84.0	20.10.23 (Docker CE)	私有
Windows_Server-2016-English-Full-ECS_Optimized-2024.06.14	1.82.3	20.10.23 (Docker CE)	私有
Windows_Server-2016-English-Full-ECS_Optimized-2024.05.14	1.82.2	20.10.23 (Docker CE)	私有

Amazon ECS 最佳化 Windows Server 2016 Full AMI	Amazon ECS 容器代理程式版本	Docker 版本	Visibility
Windows_Server-2016-English-Full-ECS_Optimized-2024.04.09	1.82.2	20.10.23 (Docker CE)	私有
Windows_Server-2016-English-Full-ECS_Optimized-2024.03.12	1.82.0	20.10.23 (Docker CE)	私有
Windows_Server-2016-English-Full-ECS_Optimized-2024.02.13	1.81.0	20.10.23 (Docker CE)	私有

使用下列 AWS CLI Amazon ECS 最佳化 Windows Server 2016 Full AMI。

```
aws ssm get-parameters --names /aws/service/ami-windows-latest/Windows_Server-2016-English-Full-ECS_Optimized
```

建立自己的 Amazon ECS 最佳化 Windows AMI

使用 EC2 Image Builder 建置您自己的自訂 Amazon ECS 最佳化 Windows AMI。這使得在 Amazon ECS 上使用您自己的授權 Windows AMI 變得更簡單。Amazon ECS 提供受管的 Image Builder 元件，由該元件提供執行 Windows 執行個體所需的系統組態，以託管您的容器。每個 Amazon ECS 受管元件包含特定的容器代理程式和 Docker 版本。您可以自訂映像以使用最新的 Amazon ECS 受管元件，或者如果需要較舊的容器代理程式或 Docker 版本，您可以指定不同的元件。

如需使用 EC2 Image Builder 的完整逐步解說，請參閱 EC2 Image Builder 使用者指南中的 EC2 Image Builder 入門。

在使用 EC2 Image Builder 建立您自己的 Amazon ECS 最佳化 Windows AMI 時，您將建立映像配方。您的映像配方必須符合以下要求：

- 來源映像應以 Windows Server 2019 Core、Windows Server 2019 Full、Windows Server 2022 Core 或 Windows Server 2022 Full 為基礎。任何其他 Windows 作業系統不受支援，而且可能與元件不相容。
- 指定建置元件時需要 `ecs-optimized-ami-windows` 元件。為確保映像包含最新的安全性更新，建議使用 `update-windows` 元件。

  若要指定不同的元件版本，展開 Versioning options (版本控制選項) 選單，然後指定您要使用的元件版本。如需詳細資訊，請參閱列出 `ecs-optimized-ami-windows` 元件版本。

列出 **`ecs-optimized-ami-windows`** 元件版本

在建立 EC2 Image Builder 配方並指定 `ecs-optimized-ami-windows` 元件時，您可以使用預設選項，或者您也可以指定特定的元件版本。要判定可用的元件版本，以及元件所包含的 Amazon ECS 容器代理程式和 Docker 版本，您可以使用 AWS Management Console。

列出可用的 **`ecs-optimized-ami-windows`** 元件版本

1. 開啟 EC2 Image Builder 主控台，位於 https://console.aws.amazon.com/imagebuilder/。
2. 在導覽列上選取您在哪個區域建立您的映像。
3. 在導覽窗格的 Saved configurations (已儲存組態) 選單的下方，選擇 Components (元件)。
4. 在 Components (元件) 頁面的搜尋列鍵入 `ecs-optimized-ami-windows`，然後向下拉資格選單並選取 Quick start (Amazon-managed) (Quick start (Amazon 受管))。
5. 使用 Description (描述) 欄來判定元件版本，以及您的映像所需的 Amazon ECS 容器代理程序和 Docker 版本。

Amazon ECS Windows 容器執行個體管理

當您將 EC2 執行個體用於 Amazon ECS 工作負載時，您必須負責維護執行個體。

代理更新不適用於 Windows 容器執行個體。我們建議您啟動新的容器執行個體，以更新您 Windows 叢集中的代理版本。

管理程序

- 啟動 Amazon ECS Windows 容器執行個體
- 引導 Amazon ECS Windows 容器執行個體以傳遞資料
- 針對 Amazon ECS Windows 容器執行個體使用 HTTP 代理
- 設定 Amazon ECS Windows 容器執行個體以接收 Spot 執行個體通知

啟動 Amazon ECS Windows 容器執行個體

您的 Amazon ECS 容器執行個體是使用 Amazon EC2 主控台建立的。開始之前，請務必先完成 設定以使用 Amazon ECS。 中的步驟。

如需啟動精靈的詳細資訊，請參閱《Amazon EC2 使用者指南》中的使用新的啟動執行個體精靈啟動執行個體。

您可以使用新的 Amazon EC2 精靈啟動執行個體。您可以使用以下參數列表，並將參數保留不作為預設列出。下列指示會引導您完成每個參數群組。

程序

開始之前，請完成 設定以使用 Amazon ECS。 中的步驟。

1. 在 https://console.aws.amazon.com/ec2/ 開啟 Amazon EC2 主控台。
2. 在畫面頂端的導覽列中，會顯示目前的 AWS 區域 （例如美國東部 （俄亥俄））。選取要在其中啟動執行個體的區域。這項選擇非常重要，因為有些 Amazon EC2 資源可在不同區域間共享，其他則無法。
3. 從 Amazon EC2 主控台儀表板選擇 Launch Instance (啟動執行個體)。

名稱和標籤

執行個體名稱是一個標籤，其中鍵是 Name (名稱)，而值是您指定的名稱。您可以標記執行個體、磁碟區和彈性圖形。對於 Spot 執行個體，您只能標記 Spot 執行個體請求。

指定執行個體名稱和其他標籤是選用的。

- 對於 Name (名稱)，輸入執行個體的描述性名稱。如果您未指定名稱，則可以透過其 ID 來標識執行個體，該 ID 將在您啟動執行個體時自動產生。
- 若要新增其他標籤，請選擇 Add additional tags (新增其他標籤)。選取 Add tag (新增標籤)，然後輸入鍵和值，然後選取要標記的資源類型。為每個要新增的其他標籤重新選擇 Add tag (新增標籤)。

應用程式和作業系統映像 (Amazon Machine Image)

Amazon Machine Image (AMI) 包含建立執行個體所需的資訊。例如，AMI 可能包含作為 Web 伺服器所必需的軟體，例如 Apache 和您的網站。

如需最新的 Amazon ECS 最佳化 AMI 及其值，請參閱《Windows Amazon ECS 最佳化 AMI》。

使用搜尋列來尋找由 發佈的適當 Amazon ECS 最佳化 AMI AWS。

1. 根據您的要求，在 Search (搜尋) 列中輸入以下其中一個 AMI，並按下 Enter。
   - Windows_Server-2022-English-Full-ECS_Optimized
   - Windows_Server-2022-English-Core-ECS_Optimized
   - Windows_Server-2019-English-Full-ECS_Optimized
   - Windows_Server-2019-English-Core-ECS_Optimized
   - Windows_Server-2016-English-Full-ECS_Optimized
2. 在 Choose an Amazon Machine Image (AMI) (選擇 Amazon Machine Image (AMI)) 頁面上，選取 Community AMIs (社群 AMI) 索引標籤。
3. 從出現的清單中，選擇具有最新發佈日期的 Microsoft 驗證 AMI，然後按一下 Select (選取)。

執行個體類型

執行個體類型定義執行個體的硬體組態和大小。較大的執行個體類型具有較多的 CPU 和記憶體。如需詳細資訊，請參閱執行個體類型。

- 針對 Instance type (執行個體類型)，選取執行個體的執行個體類型。

  您選取的執行個體類型，會決定您任務執行所在的可用資源。

金鑰對 (登入)

針對 Key pair name (金鑰對名稱)，選擇現有的金鑰對，或選擇 Create new key pair (建立新的金鑰對) 以建立新的金鑰對。

> **Important**
>
> 如果您選擇 Proceed without key pair (Not recommended) (繼續而不使用金鑰對 (不建議)) 選項，您將無法連線到執行個體，除非您選擇已設定為允許使用者透過其他方式登入的 AMI。

Network settings (網路設定)

視需要設定網路設定。

- Networking platform (聯網平台)：選擇 Virtual Private Cloud (VPC) (虛擬私有雲端 (VPC))，然後在 Network interfaces (網路介面) 區段指定子網路。
- VPC：選取要在其中建立安全群組的現有 VPC。

- Subnet (子網)：您可以在與可用區域、Local Zone、Wavelength 區域或 Outpost 相關聯的子網中啟動執行個體。

  若要在可用區域中啟動執行個體，請選取要在當中啟動執行個體的子網。若要建立新的子網，請選擇 Create new subnet (建立新的子網)，以前往 Amazon VPC 主控台。完成後，請返回啟動執行個體精靈並選擇 Refresh (重新整理) 圖示，載入清單中的子網。

  在 Local Zone 中啟動執行個體，選取您在 Local Zone 中建立的子網。

  若要在 Outpost 中啟動執行個體，請在與 Outpost 相關聯的 VPC 中選取子網。

- Auto-assign Public IP (自動指派公有 IP)：如果您的執行個體應從網際網路存取，請確認 Auto-assign Public IP (自動指派公有 IP) 欄位設為 Enable (啟用)。如果不是，請將此欄位設為 Disable (停用)。

  > **Note**
  >
  > 容器執行個體需要存取，才可以與 Amazon ECS 服務端點通訊。可透過介面 VPC 端點或透過具備公有 IP 位址的容器執行個體來實現。
  > 如需介面 VPC 端點的詳細資訊，請參閱 Amazon ECS 介面 VPC 端點 (AWS PrivateLink)
  > 如果您沒有設定介面 VPC 端點，且容器執行個體沒有公有 IP 位址，則它們必須使用網路位址轉譯 (NAT) 來提供此存取。如需詳細資訊，請參閱《Amazon VPC 使用者指南》中的 NAT 閘道和本指南中的 針對 Amazon ECS Linux 容器執行個體使用 HTTP 代理。

- Firewall (security groups) (防火牆 (安全群組))：使用安全群組定義容器執行個體的防火牆規則。這些規則指定傳遞至容器執行個體的傳入網路流量。所有其他流量都會遭到忽略。
  - 若要選取現有的安全群組，請選擇 Select existing security group (選取現有安全群組)，然後從您在設定以使用 Amazon ECS。 建立的安全群組選取您的安全群組。

設定儲存

您選取的 AMI 包含一或多個儲存體磁碟區，包括根磁碟區。您可以指定要連接至執行個體的其他磁碟區。

您可以使用 Simple (簡單) 檢視。

- Storage type (儲存類型)：為您的容器執行個體設定儲存。

如果您使用的是 Amazon ECS 最佳化的 Amazon Linux AMI，您的執行個體會設定兩個磁碟區。Root (根目錄) 磁碟區供作業系統使用，而第二個 Amazon EBS 磁碟區 (連接到 `/dev/xvdcz`) 則供 Docker 使用。

您可以選擇性地增加或減少您執行個體的磁碟區大小，使其符合您的應用程式需求。

進階詳細資訊

針對 Advanced Details (進階詳細資訊)，展開此區段來檢視欄位，指定執行個體的其他參數。

- Purchasing option (購買選項)：選擇 Request Spot instances (請求 Spot 執行個體) 以請求 Spot 執行個體。您也需要設定與 Spot 執行個體相關的其他欄位。如需詳細資訊，請參閱「Spot 執行個體要求」。

  > **Note**
  >
  > 如果您使用的是 Spot 執行個體，而且看到 `Not available` 訊息，您可能需要選擇不同的執行個體類型。

  .

- IAM instance profile (IAM 執行個體設定檔)：選取您的容器執行個體 IAM 角色。這通常命名為 `ecsInstanceRole`。

  > **Important**
  >
  > 如果您未使用適當的 IAM 許可啟動容器執行個體，Amazon ECS 代理程式就無法連線到叢集。如需詳細資訊，請參閱Amazon ECS 容器執行個體 IAM 角色。

- (選用) User data (使用者資料)：利用使用者資料設定您的 Amazon ECS 容器執行個體，例如Amazon ECS 容器代理程式組態中的代理程式環境變數。Amazon EC2 使用者資料指令碼僅於執行個體初次啟動時執行一次。以下是使用者資料用途的常見範例：

  - 您的容器執行個體預設會在您的預設叢集中啟動。若要在非預設的叢集中啟動，請選擇 Advanced Details (進階詳細資訊) 清單。然後，將以下指令碼貼入 User data (使用者資料) 欄位，以您的叢集名稱取代 *`your_cluster_name`*。

    `EnableTaskIAMRole` 會開啟任務的任務 IAM 角色功能。

此外，當您使用 awsvpc 網路模式時，可使用以下選項。

- EnableTaskENI：此標記會開啟任務聯網，並且當您使用 awsvpc 網路模式時需要它。
- AwsvpcBlockIMDS：此選擇性標記會封鎖在 awsvpc 網路模式中執行的任務容器的 IMDS 存取。
- AwsvpcAdditionalLocalRoutes：此選擇性標記可讓您在任務命名空間中擁有其他路由。

  將 ip-address 替換為其他路由的 IP 地址，例如 172.31.42.23/32。

```
<powershell>
Import-Module ECSTools
Initialize-ECSAgent -Cluster your_cluster_name -EnableTaskIAMRole -EnableTaskENI -
AwsvpcBlockIMDS -AwsvpcAdditionalLocalRoutes
'["ip-address"]'
</powershell>
```

引導 Amazon ECS Windows 容器執行個體以傳遞資料

當您啟動 Amazon EC2 執行個體時，您可以將使用者資料傳遞至 EC2 執行個體。此資料可用來執行常見的自動化組態任務，甚至在執行個體啟動時，執行指令碼。對於 Amazon ECS，使用者資料的最常用案例是將組態資訊傳送到 Docker 常駐程式和 Amazon ECS 容器代理程式。

您可以將多種類型的使用者資料傳遞給 Amazon EC2，包含雲端 boothook、shell 指令碼和 cloud-init 指令。如需這些和其他格式類型的詳細資訊，請參閱 Cloud-Init 文件。

您可以在使用 Amazon EC2 啟動精靈時傳遞此使用者資料。如需詳細資訊，請參閱啟動 Amazon ECS Linux 容器執行個體。

預設 Windows 使用者資料

此範例使用者資料指令碼顯示您使用主控台時，Windows 容器執行個體所收到的預設使用者資料。下列指令碼會執行下列動作：

- 將叢集名稱設定為您輸入的名稱。
- 設定任務的 IAM 角色。
- 將 json-file 和 awslogs 設定為可用的記錄驅動程式。

此外，當您使用 awsvpc 網路模式時，可使用以下選項。

- `EnableTaskENI`：此標記會開啟任務聯網，並且當您使用 `awsvpc` 網路模式時需要它。
- `AwsvpcBlockIMDS`：此選擇性標記會封鎖在 `awsvpc` 網路模式中執行的任務容器的 IMDS 存取。
- `AwsvpcAdditionalLocalRoutes`：此選擇性標記可讓您擁有其他路由。

  將 `ip-address` 替換為其他路由的 IP 地址，例如 172.31.42.23/32。

您可以將此指令碼用於您自己的容器執行個體 (前提是執行個體是從 Amazon ECS 最佳化 Windows Server AMI 啟動)。

替換 `-Cluster` *`cluster-name`* 行以指定您自己的叢集名稱。

```
<powershell>
Initialize-ECSAgent -Cluster cluster-name -EnableTaskIAMRole -LoggingDrivers '["json-
file","awslogs"]' -EnableTaskENI -AwsvpcBlockIMDS -AwsvpcAdditionalLocalRoutes
'["ip-address"]'
</powershell>
```

對於設定為使用 `awslogs` 日誌記錄驅動程式的 Windows 任務，您也必須在容器執行個體上設定 `ECS_ENABLE_AWSLOGS_EXECUTIONROLE_OVERRIDE` 環境變數。請使用下列語法。

替換 `-Cluster` *`cluster-name`* 行以指定您自己的叢集名稱。

```
<powershell>
[Environment]::SetEnvironmentVariable("ECS_ENABLE_AWSLOGS_EXECUTIONROLE_OVERRIDE",
 $TRUE, "Machine")
Initialize-ECSAgent -Cluster cluster-name -EnableTaskIAMRole -LoggingDrivers '["json-
file","awslogs"]'
</powershell>
```

Windows 代理程式安裝使用者資料

此範例使用者資料指令碼會將 Amazon ECS 容器代理程式安裝在以 Windows_Server-2016-English-Full-Containers AMI 啟動的執行個體上。它已從 Amazon ECS Container Agent GitHub 儲存庫讀我檔案頁面的代理程式安裝說明調整。

> **Note**
>
> 此指令碼會基於舉例用途而共享。使用 Amazon ECS 最佳化 Windows Server AMI，更容易開始使用 Windows 容器。如需詳細資訊，請參閱為 Fargate 啟動類型建立 Amazon ECS 叢集。

您可以對您自己的容器執行個體使用此指令碼 (假設是使用 Windows_Server-2016-English-Full-Containers AMI 啟動它們)。務必取代 *windows* 一行，以指定您自己的叢集名稱 (如果您使用的叢集名稱不是 windows)。

```
<powershell>
# Set up directories the agent uses
New-Item -Type directory -Path ${env:ProgramFiles}\Amazon\ECS -Force
New-Item -Type directory -Path ${env:ProgramData}\Amazon\ECS -Force
New-Item -Type directory -Path ${env:ProgramData}\Amazon\ECS\data -Force
# Set up configuration
$ecsExeDir = "${env:ProgramFiles}\Amazon\ECS"
[Environment]::SetEnvironmentVariable("ECS_CLUSTER", "windows", "Machine")
[Environment]::SetEnvironmentVariable("ECS_LOGFILE", "${env:ProgramData}\Amazon\ECS\log
\ecs-agent.log", "Machine")
[Environment]::SetEnvironmentVariable("ECS_DATADIR", "${env:ProgramData}\Amazon\ECS
\data", "Machine")
# Download the agent
$agentVersion = "latest"
$agentZipUri = "https://s3.amazonaws.com/amazon-ecs-agent/ecs-agent-windows-
$agentVersion.zip"
$zipFile = "${env:TEMP}\ecs-agent.zip"
Invoke-RestMethod -OutFile $zipFile -Uri $agentZipUri
# Put the executables in the executable directory.
Expand-Archive -Path $zipFile -DestinationPath $ecsExeDir -Force
Set-Location ${ecsExeDir}
# Set $EnableTaskIAMRoles to $true to enable task IAM roles
# Note that enabling IAM roles will make port 80 unavailable for tasks.
[bool]$EnableTaskIAMRoles = $false
if (${EnableTaskIAMRoles}) {
  $HostSetupScript = Invoke-WebRequest https://raw.githubusercontent.com/aws/amazon-
ecs-agent/master/misc/windows-deploy/hostsetup.ps1
  Invoke-Expression $($HostSetupScript.Content)
}
# Install the agent service
New-Service -Name "AmazonECS" `
        -BinaryPathName "$ecsExeDir\amazon-ecs-agent.exe -windows-service" `
        -DisplayName "Amazon ECS" `
        -Description "Amazon ECS service runs the Amazon ECS agent" `
        -DependsOn Docker `
        -StartupType Manual
sc.exe failure AmazonECS reset=300 actions=restart/5000/restart/30000/restart/60000
sc.exe failureflag AmazonECS 1
Start-Service AmazonECS
```

```
</powershell>
```

針對 Amazon ECS Windows 容器執行個體使用 HTTP 代理

您可以設定您的 Amazon ECS 容器執行個體，針對 Amazon ECS 容器代理程式和 Docker 常駐程式使用 HTTP 代理。這在您的容器執行個體沒有透過 Amazon VPC 網際網路閘道、NAT 閘道或執行個體進行外部網路存取時會很有用。

若要設定您的 Amazon ECS Windows 容器執行個體以使用 HTTP 代理，請在啟動時 (使用 Amazon EC2 使用者資料) 設定下列變數。

```
[Environment]::SetEnvironmentVariable("HTTP_PROXY",
"http://proxy.mydomain:port", "Machine")
```

將 HTTP_PROXY 設為要讓 Amazon ECS 代理連線到網際網路的 HTTP 代理之主機名稱 (或 IP 地址) 及連接埠號碼。例如，您的容器執行個體可能沒有透過 Amazon VPC 網際網路閘道、NAT 閘道或執行個體的外部網路存取。

```
[Environment]::SetEnvironmentVariable("NO_PROXY",
"169.254.169.254,169.254.170.2,\\.\pipe\docker_engine", "Machine")
```

將 NO_PROXY 設為 169.254.169.254,169.254.170.2,\\.\pipe\docker_engine，篩選 EC2 執行個體中繼資料、任務的 IAM 角色，以及來自代理的 Docker 常駐程式流量。

Example Windows HTTP 代理使用者資料指令碼

以下範例使用者資料 PowerShell 指令碼會設定 Amazon ECS 容器代理程式和 Docker 常駐程式，以使用您指定的 HTTP 代理。您也可以指定容器執行個體自行註冊的叢集。

若要在您啟動容器執行個體時使用此指令碼，請遵循「the section called “啟動容器執行個體”」中的步驟進行。只要複製以下 PowerShell 指令碼，並在 User data (使用者資料) 欄位中貼上即可 (請確認您使用自己的代理和叢集資訊取代紅色的範例數值)。

> **Note**
>
> -EnableTaskIAMRole 選項是啟用任務的 IAM 角色的必要項目。如需詳細資訊，請參閱 Amazon EC2 Windows 執行個體額外組態。

```
<powershell>
```

```
Import-Module ECSTools

$proxy = "http://proxy.mydomain:port"
[Environment]::SetEnvironmentVariable("HTTP_PROXY", $proxy, "Machine")
[Environment]::SetEnvironmentVariable("NO_PROXY", "169.254.169.254,169.254.170.2,\\.
\pipe\docker_engine", "Machine")

Restart-Service Docker
Initialize-ECSAgent -Cluster MyCluster -EnableTaskIAMRole
</powershell>
```

設定 Amazon ECS Windows 容器執行個體以接收 Spot 執行個體通知

當 Spot 價格超過請求的最高價或容量不再可用時，Amazon EC2 會終止、停止或休眠您的 Spot 執行個體。Amazon EC2 會提供 Spot 執行個體中斷通知，在執行個體中斷前會向執行個體發出兩分鐘的警告。如果在執行個體上啟用 Amazon ECS Spot 執行個體耗盡，則 ECS 會收到 Spot 執行個體中斷通知，並將執行個體置於 DRAINING 狀態。

> **Important**
>
> Amazon ECS 會監控具有 `terminate` 和 `stop` 執行個體動作的 Spot 執行個體中斷通知。如果您在請求 Spot 執行個體或 Spot 機群時指定了 `hibernate` 執行個體中斷行為，則這些執行個體不支援 Amazon ECS Spot 執行個體耗盡。

將容器執行個體設定為 DRAINING 時，Amazon ECS 會避免在容器執行個體中放置新的任務排程。PENDING 狀態下即將耗盡的容器執行個體服務任務會立即停止。如果叢集有可用的容器執行個體，則會在這些容器執行個體上啟動替代服務任務。

您可以在啟動執行個體時開啟 Spot 執行個體耗盡。您必須先設定 ECS_ENABLE_SPOT_INSTANCE_DRAINING 參數，然後才能啟動容器代理程式。使用您叢集的名稱取代 *my-cluster*。

```
[Environment]::SetEnvironmentVariable("ECS_ENABLE_SPOT_INSTANCE_DRAINING", "true",
 "Machine")

# Initialize the agent
Initialize-ECSAgent -Cluster my-cluster
```

如需詳細資訊，請參閱the section called “啟動容器執行個體”。

外部啟動類型的 Amazon ECS 叢集

Amazon ECS Anywhere 支援將外部執行個體 (例如內部部署伺服器或虛擬機器 (VM)) 註冊到 Amazon ECS 叢集。外部執行個體已進行優化，可執行產生傳出流量或處理資料的應用程式。如果您的應用程式需要傳入流量，缺乏 Elastic Load Balancing 支援會使這些工作負載的執行效率降低。Amazon ECS 新增了一個新的 `EXTERNAL` 啟動類型，可用來在外部執行個體上建立服務或執行任務。

支援的作業系統和系統架構

以下是支援的作業系統和系統架構清單。

- Amazon Linux 2
- Amazon Linux 2023
- CentOS 7
- CentOS Stream 9
- RHEL 7、RHEL 8：Docker 或 RHEL 的開放套件儲存庫都不支援在 RHEL 上原生安裝 Docker。您必須先確定已安裝 Docker，然後再執行本文件中所述的安裝指令碼。
- Fedora 32、Fedora 33、Fedora 40
- openSUSE Tumbleweed
- Ubuntu 18、Ubuntu 20、Ubuntu 22、Ubuntu 24
- Debian 10

> **Important**
>
> Debian 9 長期支援 (LTS 支援) 於 2022 年 6 月 30 日結束，並不再受 Amazon ECS Anywhere 支援。

- Debian 11
- Debian 12
- SUSE Enterprise Server 15
- 支援 `x86_64` 和 `ARM64` CPU 架構。
- 支援以下 Windows 作業系統版本：
  - Windows Server 2022
  - Windows Server 2019

- Windows Server 2016
- Windows Server 20H2

考量事項

在您開始使用外部執行個體之前，請注意下列考量事項。

- 您可以一次向一個叢集註冊外部執行個體。如需有關如何向其他叢集註冊外部執行個體的說明，請參閱 取消註冊 Amazon ECS 外部執行個體。
- 您的外部執行個體需要 IAM 角色，允許其與 AWS APIs通訊。如需詳細資訊，請參閱Amazon ECS Anywhere IAM 角色。
- 您的外部執行個體不應該在本機定義預先設定的執行個體憑證鏈結，因為這會干擾註冊指令碼。
- 若要將容器日誌傳送至 CloudWatch Logs，請確定您已在任務定義中建立並指定任務執行 IAM 角色。
- 當外部執行個體註冊到叢集時，`ecs.capability.external` 屬性會與執行個體相關聯。此屬性會將執行個體視為外部執行個體。您可以將自訂屬性新增至外部執行個體，以作為任務置放條件限制。如需詳細資訊，請參閱自訂屬性。
- 您可以將資源標籤新增至外部執行個體。如需詳細資訊，請參閱外部容器執行個體。
- 外部執行個體支援 ECS Exec。如需詳細資訊，請參閱使用 ECS Exec 監控 Amazon ECS 容器。
- 以下是與外部執行個體聯網時特有的其他考量。如需詳細資訊，請參閱聯網 。
  - 不支援服務負載平衡。
  - 不支援服務探索。
  - 在外部執行個體上執行的任務必須使用 `bridge`、`host` 或none 網路模式。不支援 `awsvpc` 網路模式。
  - 每個 AWS 區域都有 Amazon ECS 服務網域。必須允許這些服務網域將流量傳送到您的外部執行個體。
  - 安裝在外部執行個體上的 SSM Agent 會維護 IAM 憑證，會使用硬體指紋每 30 分鐘輪換這些憑證。如果您的外部執行個體失去與 的連線 AWS，SSM Agent 會在重新建立連線後自動重新整理登入資料。如需詳細資訊，請參閱 AWS Systems Manager 使用者指南中的使用硬體指紋驗證內部部署伺服器和虛擬機器。
- 不支援 `UpdateContainerAgent` API。如需有關如何更新外部執行個體上的 SSM Agent 或 Amazon ECS 代理程式的說明，請參閱 更新外部執行個體上的 AWS Systems Manager 代理程式和 Amazon ECS 容器代理程式。

- 不支援 Amazon ECS 容量提供者。若要在外部執行個體上建立服務或執行獨立任務，請使用 EXTERNAL 啟動類型。
- 不支援 SELinux。
- 不支援使用 Amazon EFS 磁碟區或指定 EFSVolumeConfiguration。
- 不支援與 App Mesh 整合。
- 如果您使用主控台建立外部執行個體任務定義，則必須使用主控台 JSON 編輯器建立任務定義。
- 在 Windows 上執行 ECS Anywhere 時，必須在內部部署基礎設施上使用自己的 Windows 授權。
- 當您使用非 Amazon ECS 最佳化 AMI 時，請在外部容器執行個體上執行下列命令，以設定規則將 IAM 角色用於任務。如需詳細資訊，請參閱外部執行個體額外組態。

```
$ sysctl -w net.ipv4.conf.all.route_localnet=1
$ iptables -t nat -A PREROUTING -p tcp -d 169.254.170.2 --dport 80 -j DNAT --to-
destination 127.0.0.1:51679
$ iptables -t nat -A OUTPUT -d 169.254.170.2 -p tcp -m tcp --dport 80 -j REDIRECT --
to-ports 51679
```

聯網

Amazon ECS 外部執行個體已進行優化，可執行產生傳出流量或處理資料的應用程式。如果您的應用程式需要傳入流量 (例如 Web 服務)，則缺少 Elastic Load Balancing 支援會使執行這些工作負載效率降低，因為不支援將這些工作負載置於負載平衡器之後。

以下是與外部執行個體聯網時特有的其他考量。

- 不支援服務負載平衡。
- 不支援服務探索。
- 在外部執行個體上執行的 Linux 任務必須使用 bridge、host 或none 網路模式。不支援 awsvpc 網路模式。

  如需每個網路模式的詳細資訊，請參閱 EC2 啟動類型的 Amazon ECS 任務聯網選項。
- 在外部執行個體上執行的 Windows 任務必須使用 default 網路模式。
- 每個區域中有 Amazon ECS 服務網域，且必須允許將流量傳送到您的外部執行個體。
- 安裝在外部執行個體上的 SSM Agent 會維護 IAM 憑證，會使用硬體指紋每 30 分鐘輪換這些憑證。如果您的外部執行個體失去與 的連線 AWS，SSM Agent 會在重新建立連線後自動重新整理登入資料。如需詳細資訊，請參閱 AWS Systems Manager 使用者指南中的使用硬體指紋驗證內部部署伺服器和虛擬機器。

下列網域用於 Amazon ECS 服務與外部執行個體上安裝的 Amazon ECS 代理程式之間的通訊。請確定允許流量，而且 DNS 解析可以運作。對於每個端點，*##*表示 Amazon ECS 支援的 AWS 區域的區域識別符，例如美國東部 (俄亥俄) 區域的 `us-east-2`。應允許您使用的所有區域的端點。對於 `ecs-a` 和 `ecs-t` 端點，應該包含星號 (例如 `ecs-a-*`)。

- `ecs-a-*.`*`region`*`.amazonaws.com` - 管理任務時會使用此端點。
- `ecs-t-*.`*`region`*`.amazonaws.com` - 此端點可用來管理任務和容器指標。
- `ecs.`*`region`*`.amazonaws.com` - 這是適用於 Amazon ECS 的服務端點。
- `ssm.`*`region`*`.amazonaws.com` — 這是 的服務端點 AWS Systems Manager。
- `ec2messages.`*`region`*`.amazonaws.com` — 這是服務端點， AWS Systems Manager 用於在雲端中，在 Systems Manager 代理程式與 Systems Manager 服務之間進行通訊。
- `ssmmessages.`*`region`*`.amazonaws.com` — 必須使用這個服務端點建立和刪除連往雲端工作階段管理員服務的工作階段管道。
- 如果您的任務需要與任何其他 AWS 服務通訊，請確定這些服務端點是允許的。應用程式範例包括使用 Amazon ECR 來提取容器映像或將 CloudWatch 用於 CloudWatch Logs。如需詳細資訊，請參閱 AWS 一般參考中的服務端點。

Amazon FSx for Windows File Server 搭配 ECS Anywhere

若要 Amazon FSx for Windows File Server 搭配 Amazon ECS 外部執行個體使用 ，您必須在內部部署資料中心與 之間建立連線 AWS 雲端。如需關於將您的網路連線至 VPC 連線選項的資訊，請參閱 Amazon Virtual Private Cloud 連線能力選項。

gMSA 搭配 ECS Anywhere

支援下列 ECS Anywhere 使用案例。

- Active Directory 位於 AWS 雲端 - 對於此組態，您可以使用 連線在內部部署網路與 AWS 雲端 之間建立 AWS Direct Connect 連線。如需有關如何建立連線的資訊，請參閱 Amazon Virtual Private Cloud 連線選項。請在 AWS 雲端建立一個 Active Directory。如需有關如何開始使用 的資訊 AWS Directory Service，請參閱 AWS Directory Service 管理指南中的設定 AWS Directory Service。然後，您可以使用 AWS Direct Connect 連線將外部執行個體加入網域。如需搭配 Amazon ECS 使用 gMSA 的詳細資訊，請參閱the section called “了解如何針對 EC2 Windows 容器使用 gMSAs ”。
- Active Directory 位於內部部署資料中心。- 對於此組態，您將外部執行個體加入到內部部署 Active Directory。然後，您可以在執行 Amazon ECS 任務時使用本機可用的憑證。

為外部啟動類型建立 Amazon ECS 叢集

您可以建立叢集來定義任務和服務執行所在的基礎設施。

開始之前，請務必先完成 設定以使用 Amazon ECS。 中的步驟，並指派適當的 IAM 許可。如需詳細資訊，請參閱the section called “Amazon ECS 叢集範例”。Amazon ECS 主控台提供簡單的方法，透過建立 AWS CloudFormation 堆疊來建立 Amazon ECS 叢集所需的資源。

為了使叢集建立程序盡可能簡單，主控台提供了許多可供選擇的預設選項，我們將在下方加以說明。主控台的大多數區段還有說明面板，以提供更多上下文。

- 在 中建立與叢集 AWS Cloud Map 相同名稱的預設命名空間。命名空間可讓您在叢集中建立的服務連線到命名空間中的其他服務，而不需要額外的組態。

  如需詳細資訊，請參閱互連 Amazon ECS 服務。

您可以修改下列選項：

- 變更與叢集相關聯的預設命名空間。

  命名空間可讓您在叢集中建立的服務連線到命名空間中的其他服務，而不需要額外的組態。預設命名空間與叢集名稱相同。如需詳細資訊，請參閱互連 Amazon ECS 服務。
- 設定外部執行個體的叢集
- 使用增強的可觀測性開啟 Container Insights，或 Container Insights。

  CloudWatch Container Insights 會從您的容器化應用程式和微型服務收集、彙總及總結指標和日誌。Container Insights 還提供診斷資訊，例如容器重新啟動故障，您可以使用這些資訊快速隔離和解決這些問題。如需詳細資訊，請參閱the section called “使用具有增強可觀測性的 Container Insights 監控 Amazon ECS 容器”。

  在 2024 年 12 月 2 日， AWS 發行了 Container Insights 並增強了 Amazon ECS 的可觀測性。此版本支援使用 Amazon EC2 和 Fargate 啟動類型的 Amazon ECS 叢集增強可觀測性。在 Amazon ECS 上以增強的可觀測性設定 Container Insights 之後，Container Insights 會自動收集從叢集層級到環境中容器層級的詳細基礎設施遙測，並在儀表板中顯示您的資料，以向您顯示各種指標和維度。然後，您可以在 Container Insights 主控台上使用這些out-of-the-box儀表板，以更了解容器的運作狀態和效能，並透過識別異常狀況來更快速地解決問題。

  我們建議您使用具有增強可觀測性的 Container Insights，而不是 Container Insights，因為它可在您的容器環境中提供詳細的可見性，從而縮短解決的平均時間。

- 新增標籤以協助您識別叢集。

建立新叢集 (Amazon ECS 主控台)

1. 開啟主控台，網址為 https://console.aws.amazon.com/ecs/v2。
2. 從導覽列中選取要使用的「區域」。
3. 在導覽窗格中，選擇叢集。
4. 在 Clusters (叢集) 頁面上，選擇 Create cluster (建立叢集)。
5. 在叢集組態下，設定下列項目：
   - 在叢集名稱下輸入唯一的名稱。

     名稱可以包含最多 255 個字母 (大小寫)、數字與連字號。
   - (選用) 若要讓 Service Connect 使用的命名空間與叢集名稱不同，請在命名空間中輸入唯一的名稱。
6. （選用）使用 Container Insights，展開監控，然後選擇下列其中一個選項：
   - 若要使用建議的 Container Insights 搭配增強型可觀測性，請選擇 Container Insights 搭配增強型可觀測性。
   - 若要使用 Container Insights，請選擇 Container Insights。
7. (選用) 為協助識別您的叢集，請展開 Tags (標籤)，然後設定標籤。

   [新增標籤] 選擇新增標籤，並執行下列動作：
   - 對於 Key (金鑰)，輸入金鑰名稱。
   - 在值中，進入索引鍵值。
8. 選擇 Create (建立)。

後續步驟

您必須向叢集註冊執行個體。如需詳細資訊，請參閱將外部執行個體註冊至 Amazon ECS 叢集。

為外部啟動類型建立任務定義。如需詳細資訊，請參閱 使用主控台建立 Amazon ECS 任務定義

以獨立任務或 服務的一部分來執行您的應用程式。如需詳細資訊，請參閱下列內容：

- 以 Amazon ECS 任務執行應用程式

- 使用 主控台建立 Amazon ECS 服務

將外部執行個體註冊至 Amazon ECS 叢集

對於您向 Amazon ECS 叢集註冊的每個外部執行個體，它必須安裝 SSM Agent、Amazon ECS 容器代理程式和 Docker。若要將外部執行個體註冊至 Amazon ECS 叢集，必須先將其註冊為 AWS Systems Manager 受管執行個體。在 Amazon ECS 主控台上按幾下，可建立安裝指令碼。安裝指令碼包含 Systems Manager 啟用金鑰，以及安裝每個必要的代理程式和 Docker 的命令。必須在內部部署伺服器或虛擬機器上執行安裝指令碼，才能完成安裝和註冊步驟。

> **Note**
>
> 在向叢集註冊 Linux 外部執行個體之前，請先在外部執行個體上建立 `/etc/ecs/ecs.config` 檔案，並新增您想要的任何容器代理程式組態參數。將外部執行個體註冊到叢集之後，您無法執行此操作。如需詳細資訊，請參閱Amazon ECS 容器代理程式組態。

AWS Management Console

1. 開啟主控台，網址為 https://console.aws.amazon.com/ecs/v2。
2. 從導覽列中選取要使用的「區域」。
3. 在導覽窗格中，選擇叢集。
4. 在 Clusters (叢集) 頁面上，選擇要向其註冊外部執行個體的叢集。
5. 在 Cluster : ***name*** (叢集：名稱) 頁面上，選擇 Infrastructure (基礎基礎設施) 索引標籤。
6. 在 Register external instances (註冊外部執行個體) 頁面上，完成下列步驟。
   a. 對於 Activation key duration (in days) (啟用金鑰持續時間 (以天為單位))，輸入啟用金鑰保持作用中的天數。在您輸入的天數到期之後，當註冊外部執行個體時，金鑰就不再有效。
   b. 對於 Number of instances (執行個體數目)，輸入要使用啟用金鑰向叢集註冊的外部執行個體數。
   c. 對於 Instance role (執行個體角色)，選擇要與外部執行個體建立關聯的 IAM 角色。如果尚未建立角色，請選擇 Create new role (建立新角色)，讓 Amazon ECS 代表您建立角色。如需外部執行個體需要哪些 IAM 許可的詳細資訊，請參閱 Amazon ECS Anywhere IAM 角色。
   d. 複製註冊命令。應該在您要將其註冊到叢集的每個外部執行個體上執行此命令。

> **Important**
>
> 指令碼的 bash 部分必須以 root 身分執行。如果命令不是以 root 身分執行，則會傳回錯誤。

e. 選擇關閉。

AWS CLI for Linux operating systems

1. 建立 Systems Manager 啟用對。這是用於 Systems Manager 受管執行個體啟用。輸出包含 `ActivationId` 和 `ActivationCode`。在後續步驟中會使用它們。請確定指定您已建立的 ECS Anywhere IAM 角色。如需詳細資訊，請參閱Amazon ECS Anywhere IAM 角色。

```
aws ssm create-activation --iam-role ecsAnywhereRole | tee ssm-activation.json
```

2. 在您的內部部署伺服器或虛擬機器 (VM) 上，下載安裝指令碼。

```
curl --proto "https" -o "/tmp/ecs-anywhere-install.sh" "https://amazon-ecs-
agent.s3.amazonaws.com/ecs-anywhere-install-latest.sh"
```

3. (選用) 在您的內部部署伺服器或虛擬機器 (VM) 上，使用指令碼簽章檔案透過以下步驟來驗證安裝指令碼。

   a. 下載並安裝 GnuPG。如需 GNUpg 的詳細資訊，請參閱 GnuPG 網站。若您的系統為 Linux，請使用套件軟體管理工具在 Linux 上安裝 gpg。

   b. 擷取Amazon ECS PGP 公有金鑰。

```
gpg --keyserver hkp://keys.gnupg.net:80 --recv BCE9D9A42D51784F
```

   c. 下載安裝指令碼簽章。簽章是 ASCII 分離的 PGP 簽章，存放於副檔名為 `.asc` 的檔案中。

```
curl --proto "https" -o "/tmp/ecs-anywhere-install.sh.asc" "https://amazon-
ecs-agent.s3.amazonaws.com/ecs-anywhere-install-latest.sh.asc"
```

   d. 使用金鑰驗證安裝指令碼檔案。

```
gpg --verify /tmp/ecs-anywhere-install.sh.asc /tmp/ecs-anywhere-install.sh
```

預期的輸出如下：

```
gpg: Signature made Tue 25 May 2021 07:16:29 PM UTC
gpg:                using RSA key 50DECCC4710E61AF
gpg: Good signature from "Amazon ECS <ecs-security@amazon.com>" [unknown]
gpg: WARNING: This key is not certified with a trusted signature!
gpg:          There is no indication that the signature belongs to the
 owner.
Primary key fingerprint: F34C 3DDA E729 26B0 79BE  AEC6 BCE9 D9A4 2D51 784F
     Subkey fingerprint: D64B B6F9 0CF3 77E9 B5FB  346F 50DE CCC4 710E 61AF
```

4. 在您的內部部署伺服器或虛擬機器 (VM) 上，執行安裝指令碼。在第一步中指定叢集名稱、區域和 Systems Manager 啟用 ID 和啟用代碼。

```
sudo bash /tmp/ecs-anywhere-install.sh \
    --region $REGION \
    --cluster $CLUSTER_NAME \
    --activation-id $ACTIVATION_ID \
    --activation-code $ACTIVATION_CODE
```

對於內部部署伺服器，或已為 GPU 工作負載安裝 NVIDIA 驅動程式的虛擬機器 (VM)，您必須將 `--enable-gpu` 旗標新增至安裝指令碼。指定此旗標時，安裝指令碼會驗證 NVIDIA 驅動程式是否正在執行，然後新增所需的組態變數來執行您的 Amazon ECS 任務。如需關於執行 GPU 工作負載和在任務定義中指定 GPU 要求的詳細資訊，請參閱 在 Amazon ECS 任務定義中指定 GPUs。

```
sudo bash /tmp/ecs-anywhere-install.sh \
    --region $REGION \
    --cluster $CLUSTER_NAME \
    --activation-id $ACTIVATION_ID \
    --activation-code $ACTIVATION_CODE \
    --enable-gpu
```

進行以下步驟向其他叢集註冊現有的外部執行個體

若要向其他叢集註冊現有的外部執行個體

1. 停用 Amazon ECS 容器代理程式。

```
sudo systemctl stop ecs.service
```

2. 編輯 /etc/ecs/ecs.config 檔案，在 ECS_CLUSTER 行中，確保叢集名稱與要註冊外部執行個體的叢集名稱相符。
3. 移除現有的 Amazon ECS 代理程式資料。

```
sudo rm /var/lib/ecs/data/agent.db
```

4. 啟動 Amazon ECS 容器代理程式。

```
sudo systemctl start ecs.service
```

AWS CLI for Windows operating systems

1. 建立 Systems Manager 啟用對。這是用於 Systems Manager 受管執行個體啟用。輸出包含 ActivationId 和 ActivationCode。在後續步驟中會使用它們。請確定指定您已建立的 ECS Anywhere IAM 角色。如需詳細資訊，請參閱Amazon ECS Anywhere IAM 角色。

```
aws ssm create-activation --iam-role ecsAnywhereRole | tee ssm-activation.json
```

2. 在您的內部部署伺服器或虛擬機器 (VM) 上，下載安裝指令碼。

```
Invoke-RestMethod -URI "https://amazon-ecs-agent.s3.amazonaws.com/ecs-anywhere-install.ps1" -OutFile “ecs-anywhere-install.ps1”
```

3. (選用) Powershell 指令碼由 Amazon 簽署，因此 Windows 會自動對該指令碼執行相同的憑證驗證。您不需要執行任何手動驗證。

   若要手動驗證憑證，請用右鍵按一下該檔案，導覽到屬性，然後使用 Digital Signatures (數位簽署) 索引標籤取得更多詳細資料。

   僅在主機在憑證存放區中具有憑證時，此選項才可用。

   驗證應會傳回類似以下內容的資訊：

```
# Verification (PowerShell)
Get-AuthenticodeSignature -FilePath .\ecs-anywhere-install.ps1

SignerCertificate                         Status      Path
```

```
-----------------                              ------      ----
EXAMPLECERTIFICATE                             Valid       ecs-anywhere-install.ps1

...

Subject               : CN="Amazon Web Services, Inc.",...

----
```

4. 在您的內部部署伺服器或虛擬機器 (VM) 上，執行安裝指令碼。在第一步中指定叢集名稱、區域和 Systems Manager 啟用 ID 和啟用代碼。

```
.\ecs-anywhere-install.ps1 -Region $Region -Cluster $Cluster -
ActivationID $ActivationID -ActivationCode $ActivationCode
```

5. 驗證 Amazon ECS 容器代理程式是否正在執行。

```
Get-Service AmazonECS

Status   Name               DisplayName
------   ----               -----------
Running  AmazonECS          Amazon ECS
```

進行以下步驟向其他叢集註冊現有的外部執行個體

若要向其他叢集註冊現有的外部執行個體

1. 停用 Amazon ECS 容器代理程式。

```
Stop-Service AmazonECS
```

2. 修改 ECS_CLUSTER 參數，以便叢集名稱與要註冊外部執行個體的叢集名稱相符。

```
[Environment]::SetEnvironmentVariable("ECS_CLUSTER", $ECSCluster,
 [System.EnvironmentVariableTarget]::Machine)
```

3. 移除現有的 Amazon ECS 代理程式資料。

```
Remove-Item -Recurse -Force $env:ProgramData\Amazon\ECS\data\*
```

4. 啟動 Amazon ECS 容器代理程式。

```
Start-Service AmazonECS
```

AWS CLI 可用來在執行安裝指令碼之前建立 Systems Manager 啟用，以完成外部執行個體註冊程序。

取消註冊 Amazon ECS 外部執行個體

我們建議您在執行個體完成 AWS Systems Manager 之後，從 Amazon ECS 和 取消註冊執行個體。取消註冊後，外部執行個體即不再接受新的任務。

如果在您取消註冊容器執行個體時，還有執行中的任務，則這些任務會繼續執行，直到透過一些其他方式停用為止。不過，Amazon ECS 不再監控或負責這些任務。如果外部執行個體中的這些任務是 Amazon ECS 服務的一部分，則服務排程器可能會在不同的執行個體上啟動該任務的另一個副本。

取消註冊執行個體後，請清除執行個體上剩餘的 AWS 資源。然後，您可以將它註冊到新的叢集。

程序

AWS Management Console

1. 開啟主控台，網址為 https://console.aws.amazon.com/ecs/v2。
2. 從導覽列中選擇註冊外部執行個體所在的區域。
3. 在導覽窗格中選擇 Clusters (叢集)，並選取託管外部執行個體的叢集。
4. 在 Cluster : ***name*** (叢集：名稱) 頁面上，選擇 Infrastructure (基礎基礎設施) 索引標籤。
5. 在 Container instances (容器執行個體) 中，選擇要取消註冊的外部執行個體 ID。系統會將您重新導向至容器執行個體詳細資訊頁面。
6. 在 Container Instance : ***id*** (容器執行個體：ID) 頁面中選擇 Deregister (取消註冊)。
7. 檢閱取消註冊訊息。選取 Deregister from AWS Systems Manager (從 AWS Systems Manager中取消註冊)，也將作為 Systems Manager 受管執行個體來取消註冊外部執行個體。選擇 Deregister (取消註冊)。

> **Note**
>
> 在 Systems Manager 主控台中，可作為 Systems Manager 受管執行個體來取消註冊外部執行個體。如需說明，請參閱AWS Systems Manager 《 使用者指南》中的在混合和多雲端環境中取消註冊受管節點。

8. 取消註冊執行個體後，請清除現場部署伺服器或 VM 上的 AWS 資源。

作業系統	步驟	
Linux	a. 停用執行個體上的 Amazon ECS 容器代理程式和 SSM Agent 服務。 `sudo systemctl stop ecs amazon-ssm-agent` b. 移除 Amazon ECS 和 Systems Manager 套件。 對於 CentOS 7、CentOS 8 和 RHEL 7 `sudo yum remove -y amazon-ecs-init amazon-ssm-agent` 對於 SUSE Enterprise Server 15 `sudo zypper remove -y amazon-ecs-init amazon-ssm-agent` 對於 Debian 和 Ubuntu `sudo apt remove -y amazon-ecs-init amazon-ssm-agent` c. 移除剩餘的目錄。 `sudo rm -rf /var/lib/ecs /etc/ecs / var/lib/amazon/ss`	

作業系統	步驟
	`m /var/log/ecs / var/log/amazon/ssm`
Windows	a. 停用執行個體上的 Amazon ECS 容器代理程式和 SSM Agent 服務。 `Stop-Service AmazonECS` `Stop-Service AmazonSSMAgent` b. 移除 Amazon ECS 套件。 `.\ecs-anywhere-install.ps1 -Uninstall`

AWS CLI

1. 您需要執行個體 ID 和容器執行個體 ARN，才能取消註冊容器執行個體。如果您沒有這些值，請執行下列命令

   執行下列命令以取得執行個體 ID。

   您可以使用執行個體 ID (`instanceID`) 取得容器執行個體 ARN (`containerInstanceARN`)。

   ```
   instanceId=$(aws ssm describe-instance-information --region "{{ region }}" |
    jq ".InstanceInformationList[] |select(.IPAddress==\"{{ IPv4 Address }}\")
    | .InstanceId" | tr -d'"'
   ```

   執行下列命令。

您可以在命令中使用 containerInstanceArn 做為參數來取消註冊執行個體 (deregister-container-instance)。

```
instances=$(aws ecs list-container-instances --cluster "{{ cluster }}" --region
 "{{ region }}" | jq -c '.containerInstanceArns')
containerInstanceArn=$(aws ecs describe-container-instances --cluster
 "{{ cluster }}" --region "{{ region }}" --container-instances $instances
 | jq ".containerInstances[] | select(.ec2InstanceId==\"{{ instanceId }}\")
 | .containerInstanceArn" | tr -d '"')
```

2. 執行下列命令以消耗執行個體。

```
aws ecs update-container-instances-state --cluster "{{ cluster }}" --region
 "{{ region }}" --container-instances "{{ containerInstanceArn }}" --status
 DRAINING
```

3. 容器執行個體完成消耗後，執行下列命令以取消註冊執行個體。

```
aws ecs deregister-container-instance --cluster "{{ cluster }}" --region
 "{{ region }}" --container-instance "{{ containerInstanceArn }}"
```

4. 使用下列命令從 SSM 移除容器執行個體。

```
aws ssm deregister-managed-instance --region "{{ region }}" --instance-id
 "{{ instanceId }}"
```

5. 取消註冊執行個體後，請清除現場部署伺服器或 VM 上的 AWS 資源。

作業系統	步驟	
Linux	a. 停用執行個體上的 Amazon ECS 容器代理程式和 SSM Agent 服務。 `sudo systemctl stop ecs amazon-ssm-agent` b. 移除 Amazon ECS 和 Systems Manager 套件。	

作業系統	步驟	
	`` sudo (yum/apt/ zypper) remove amazon-ecs-init amazon-ssm-agent ``` c. 移除剩餘的目錄。 ``` sudo rm -rf /var/ lib/ecs /etc/ecs / var/lib/amazon/ss m /var/log/ecs / var/log/amazon/ssm ```	
Windows	a. 停用執行個體上的 Amazon ECS 容器代理程式和 SSM Agent 服務。 ``` Stop-Service AmazonECS ``` ``` Stop-Service AmazonSSMAgent ``` b. 移除 Amazon ECS 套件。 ``` .\ecs-anywhere-ins tall.ps1 -Uninstal l ```	

更新外部執行個體上的 AWS Systems Manager 代理程式和 Amazon ECS 容器代理程式

您的內部部署伺服器或 VM 在執行 Amazon ECS 工作負載時，必須同時執行 AWS Systems Manager Agent (SSM Agent) 和 Amazon ECS 容器代理程式。 新增或更新任何功能時， 會 AWS 發行這些代理程式的新版本。如果您的外部執行個體使用舊版代理程式，您可以使用下列程序來更新它們。

在外部執行個體上更新 SSM Agent

AWS Systems Manager 建議您自動執行更新執行個體上 SSM Agent 的程序。它們提供多種方法來自動化更新。如需詳細資訊，請參閱 AWS Systems Manager 使用者指南中的自動化 SSM Agent 更新。

在外部執行個體上更新 Amazon ECS 代理程式

在外部執行個體上，Amazon ECS 容器代理程式會透過升級 `ecs-init` 套件進行更新。更新 Amazon ECS 代理程式不會中斷執行中的任務或服務。Amazon ECS 在每個區域的 Amazon S3 儲存貯體中提供 `ecs-init` 套件和簽章檔案。從 `ecs-init` 版本 `1.52.1-1` 開始，Amazon ECS 提供單獨的 `ecs-init` 套件以供使用，這取決於外部執行個體所使用的作業系統和系統架構。

依據外部執行個體使用的作業系統和系統架構，使用下表來確定您應下載的 `ecs-init` 套件。

> **Note**
>
> 透過使用下列命令來確定外部執行個體所使用的作業系統和系統架構。
>
> ```
> cat /etc/os-release
> uname -m
> ```

作業系統 (架構)	ecs-init 套件
CentOS 7 (x86_64) CentOS 8 (x86_64) CentOS 串流 9 (x86_64) SUSE Enterprise Server 15 (x86_64)	`amazon-ecs-init-latest.x86_64.rpm`

作業系統 (架構)	ecs-init 套件
RHEL 7 (x86_64) RHEL 8 (x86_64)	
CentOS 7 (aarch64) CentOS 8 (aarch64) CentOS 串流 9 (aarch64) RHEL 7 (aarch64)	`amazon-ecs-init-latest.aarch64.rpm`
Debian 9 (x86_64) Debian 10 (x86_64) Debian 11 (x86_64) Debian 12 (x86_64) Ubuntu 18 (x86_64) Ubuntu 20 (x86_64) Ubuntu 22 (x86_64) Ubuntu 24 (x86_64)	`amazon-ecs-init-latest.amd64.deb`

作業系統 (架構)	ecs-init 套件
Debian 9 (aarch64) Debian 10 (aarch64) Debian 11 (aarch64) Debian 12 (aarch64) Ubuntu 18 (aarch64) Ubuntu 20 (aarch64) Ubuntu 22 (aarch64) Ubuntu 24 (aarch64)	amazon-ecs-init-latest.arm64.deb

請依照以下步驟更新 Amazon ECS 代理程式。

若要更新 Amazon ECS 代理程式

1. 確認您正在執行的 Amazon ECS 代理程式版本。

```
curl -s 127.0.0.1:51678/v1/metadata | python3 -mjson.tool
```

2. 下載適用於您的作業系統和系統架構的 ecs-init 套件。Amazon ECS 在每個區域的 Amazon S3 儲存貯體中提供 ecs-init 套件檔案。請務必將命令中的 *<region>* 識別符替換為與您的地理位置最接近的區域名稱 (例如，us-west-2)。

   amazon-ecs-init-latest.x86_64.rpm

```
curl -o amazon-ecs-init.rpm https://s3.<region>.amazonaws.com/amazon-ecs-agent-<region>/amazon-ecs-init-latest.x86_64.rpm
```

   amazon-ecs-init-latest.aarch64.rpm

```
curl -o amazon-ecs-init.rpm https://s3.<region>.amazonaws.com/amazon-ecs-agent-<region>/amazon-ecs-init-latest.aarch64.rpm
```

amazon-ecs-init-latest.amd64.deb

```
curl -o amazon-ecs-init.deb https://s3.<region>.amazonaws.com/amazon-ecs-agent-<region>/amazon-ecs-init-latest.amd64.deb
```

amazon-ecs-init-latest.arm64.deb

```
curl -o amazon-ecs-init.deb https://s3.<region>.amazonaws.com/amazon-ecs-agent-<region>/amazon-ecs-init-latest.arm64.deb
```

3. (選用) 使用 PGP 簽章確認 `ecs-init` 套件檔案的有效性。

   a. 下載並安裝 GnuPG。如需 GNUpg 的詳細資訊，請參閱 GnuPG 網站。若您的系統為 Linux，請使用套件軟體管理工具在 Linux 上安裝 gpg。

   b. 擷取Amazon ECS PGP 公有金鑰。

   ```
   gpg --keyserver hkp://keys.gnupg.net:80 --recv BCE9D9A42D51784F
   ```

   c. 下載 `ecs-init` 套件簽章。簽章是 ASCII 分離的 PGP 簽章，存放於副檔名為 `.asc` 的檔案中。Amazon ECS 在每個區域的 Amazon S3 儲存貯體中提供簽章檔案。請務必將命令中的 *<region>* 識別符替換為與您的地理位置最接近的區域名稱 (例如，`us-west-2`)。

   amazon-ecs-init-latest.x86_64.rpm

   ```
   curl -o amazon-ecs-init.rpm.asc https://s3.<region>.amazonaws.com/amazon-ecs-agent-<region>/amazon-ecs-init-latest.x86_64.rpm.asc
   ```

   amazon-ecs-init-latest.aarch64.rpm

   ```
   curl -o amazon-ecs-init.rpm.asc https://s3.<region>.amazonaws.com/amazon-ecs-agent-<region>/amazon-ecs-init-latest.aarch64.rpm.asc
   ```

   amazon-ecs-init-latest.amd64.deb

   ```
   curl -o amazon-ecs-init.deb.asc https://s3.<region>.amazonaws.com/amazon-ecs-agent-<region>/amazon-ecs-init-latest.amd64.deb.asc
   ```

   amazon-ecs-init-latest.arm64.deb

```
curl -o amazon-ecs-init.deb.asc https://s3.<region>.amazonaws.com/amazon-ecs-agent-<region>/amazon-ecs-init-latest.arm64.deb.asc
```

d. 使用金鑰驗證 ecs-init 套件檔案。

對於 **rpm** 套件

```
gpg --verify amazon-ecs-init.rpm.asc ./amazon-ecs-init.rpm
```

對於 **deb** 套件

```
gpg --verify amazon-ecs-init.deb.asc ./amazon-ecs-init.deb
```

預期的輸出如下：

```
gpg: Signature made Fri 14 May 2021 09:31:36 PM UTC
gpg:                using RSA key 50DECCC4710E61AF
gpg: Good signature from "Amazon ECS <ecs-security@amazon.com>" [unknown]
gpg: WARNING: This key is not certified with a trusted signature!
gpg:          There is no indication that the signature belongs to the owner.
Primary key fingerprint: F34C 3DDA E729 26B0 79BE  AEC6 BCE9 D9A4 2D51 784F
     Subkey fingerprint: D64B B6F9 0CF3 77E9 B5FB  346F 50DE CCC4 710E 61AF
```

4. 安裝 ecs-init 套裝服務。

對於 CentOS 7、CentOS 8 以及 RHEL 7 中的 **rpm** 套件

```
sudo yum install -y ./amazon-ecs-init.rpm
```

對於 SUSE Enterprise Server 15 中的 **rpm** 套件

```
sudo zypper install -y --allow-unsigned-rpm ./amazon-ecs-init.rpm
```

對於 **deb** 套件

```
sudo dpkg -i ./amazon-ecs-init.deb
```

5. 重新啟動 ecs 服務。

```
sudo systemctl restart ecs
```

6. 確認 Amazon ECS 代理程式版本已更新。

```
curl -s 127.0.0.1:51678/v1/metadata | python3 -mjson.tool
```

更新 Amazon ECS 叢集

您可以修改下列叢集屬性：

- 設定預設容量提供者

  每個叢集可以有一或多個容量提供者，以及選用的容量提供者策略。容量提供者策略會決定任務在叢集的容量提供者之間分散的方式。當您執行獨立任務或建立服務時，可以使用叢集的預設容量提供者策略，也可以使用能覆寫預設值的容量提供者策略。
- 開啟 Container Insights。

  CloudWatch Container Insights 會從您的容器化應用程式和微型服務收集、彙總及總結指標和日誌。Container Insights 還提供診斷資訊，例如容器重新啟動故障，您可以使用這些資訊快速隔離和解決這些問題。如需詳細資訊，請參閱the section called “使用具有增強可觀測性的 Container Insights 監控 Amazon ECS 容器”。
- 新增標籤以協助您識別叢集。

程序

1. 開啟主控台，網址為 https://console.aws.amazon.com/ecs/v2。
2. 在導覽窗格中，選擇叢集。
3. 在叢集頁面上，選擇叢集。
4. 在叢集：**##**頁面上，選擇更新叢集。
5. 若要設定預設容量提供者，請在預設容量提供者策略下選擇新增更多。

   a. 對於容量提供者，選擇容量提供者。

   b. (選用) 在基礎中，輸入在容量提供者上執行的任務的最小數目。

      您只能為一個容量提供者設定基礎值。

c. (選用) 在加權中，輸入使用指定容量提供者的任務在已啟動任務的總數中所佔的相對百分比。

d. (選用) 針對任何其他容量提供者重複上述步驟。

6. 若要開啟或關閉 Container Insights，請展開監控，然後開啟使用 Container Insights。

7. 為協助識別您的叢集，請展開標籤，然後設定標籤。

   [新增標籤] 選擇新增標籤，並執行下列動作：

   - 對於 Key (金鑰)，輸入金鑰名稱。
   - 對於 Value (值)，進入金鑰值。

   [移除標籤] 選擇標籤「金鑰」和「值」右側的移除。

8. 選擇更新。

刪除 Amazon ECS 叢集

如果叢集已使用完畢，即可將其刪除。刪除叢集後，其會轉換到 INACTIVE 狀態。具有 INACTIVE 狀態的叢集可能會在您的帳戶中保持可探索一段時間。不過，此行為未來可能變動，因此不建議依賴 INACTIVE 叢集得以保存。

刪除叢集之前，您必須執行下列操作：

- 刪除叢集中的所有服務。如需詳細資訊，請參閱the section called “刪除服務”。
- 停止所有當前正在執行的任務。如需詳細資訊，請參閱the section called “停止任務”。
- 取消註冊叢集中的所有已註冊的容器執行個體。如需詳細資訊，請參閱the section called “取消註冊容器執行個體”。
- 刪除 命名空間。如需詳細資訊，請參閱 AWS Cloud Map 開發人員指南中的刪除命名空間。

程序

1. 開啟主控台，網址為 https://console.aws.amazon.com/ecs/v2。
2. 從導覽列中選取要使用的「區域」。
3. 在導覽窗格中，選擇叢集。
4. 在 Clusters (叢集) 頁面上，選取要刪除的叢集。
5. 在頁面右上角，請選擇 Delete Cluster (刪除叢集)。

當您沒有刪除與叢集關聯的所有資源時，將顯示一則訊息。

6. 在確認方塊中，輸入 delete ***cluster name***。

取消註冊 Amazon ECS 容器執行個體

> **Important**
>
> 本主題僅適用於在 Amazon EC2 中建立的容器執行個體。如需取消註冊外部執行個體的詳細資訊，請參閱 取消註冊 Amazon ECS 外部執行個體。

完成 Amazon EC2 支援的容器執行個體後，應該可以從您的叢集中取消註冊它。取消註冊後，容器執行個體即不再接受新的任務。

如果在您取消註冊容器執行個體時，還有執行中的任務，這些任務會繼續執行，直到您終止執行個體或任務因某些緣由而停止為止。不過，這些任務是孤立的，這意味著 Amazon ECS 不再監控或負責它們。如果容器執行個體中的孤立任務是 Amazon ECS 服務的一部分，該服務排程器有可能會在不同的容器執行個體上啟動該任務的另一個副本。會取消註冊已註冊有 Application Load Balancer 目標群組的孤立服務任務中的所有容器。他們會根據負載平衡器或目標群組的設定，開始進行連接耗盡。如果孤立任務正在使用 `awsvpc` 網路模式，則會刪除其彈性網路介面。

如果在取消註冊後，您打算將容器執行個體另作他用，您應該先停止該容器執行個體上執行的所有任務，再取消註冊。這可防止任何遺棄的任務耗費資源。

取消註冊容器執行個體時，請注意下列考量事項。

- 因為每個容器執行個體都有唯一的狀態資訊，所以不應從一個叢集取消註冊，又在另一個叢集中重新註冊。若要重新定位容器執行個體資源，我們建議您從一個叢集終止容器執行個體，再於新叢集中啟動新的容器執行個體。如需詳細資訊，請參閱《Amazon EC2 使用者指南》和》中的終止執行個體啟動 Amazon ECS Linux 容器執行個體。
- 如果容器執行個體是由 Auto Scaling 群組或 AWS CloudFormation 堆疊管理，請透過更新 Auto Scaling 群組或 AWS CloudFormation 堆疊來終止執行個體。否則，Auto Scaling 群組或 AWS CloudFormation 會在您終止執行個體後建立新執行個體。
- 如果您終止的是具有連線的 Amazon ECS 容器代理的執行中容器執行個體，代理程式會自動從您的叢集取消註冊該執行個體。如果是已停止的容器執行個體或具有中斷連線代理的執行個體，則不會在終止時自動取消註冊。

- 取消註冊容器執行個體會從叢集中移除執行個體，但不會終止 Amazon EC2 執行個體。如果您已完成使用執行個體，請務必終止它以停止計費。如需詳細資訊，請參閱《Amazon EC2 使用者指南》中的終止您的執行個體。

程序

1. 開啟主控台，網址為 https://console.aws.amazon.com/ecs/v2。
2. 從導覽列中選擇註冊外部執行個體所在的區域。
3. 在導覽窗格中選擇 Clusters (叢集)，並選取託管執行個體的叢集。
4. 在 Cluster : ***name*** (叢集：名稱) 頁面上，選擇 Infrastructure (基礎基礎設施) 索引標籤。
5. 在 Container instances (容器執行個體) 中，選取要取消註冊的執行個體 ID。系統會將您重新導向至容器執行個體詳細資訊頁面。
6. 在 Container Instance : ***id*** (容器執行個體：ID) 頁面中選擇 Deregister (取消註冊)。
7. 在確認畫面上，選擇取消註冊。
8. 如果您完成容器執行個體，請終止基礎的 Amazon EC2 執行個體。如需詳細資訊，請參閱《Amazon EC2 使用者指南》中的終止您的執行個體。

耗盡 Amazon ECS 容器執行個體

有時候，您可能需要從叢集中移除容器執行個體，例如執行系統更新或縮減叢集容量。Amazon ECS 可讓您將容器執行個體轉換為 DRAINING 狀態。這稱為容器執行個體耗盡。將容器執行個體設定為 DRAINING 時，Amazon ECS 會避免在容器執行個體中放置新的任務排程。

服務的耗盡行為

會立即停止屬於 PENDING 狀態之服務的任何任務。如果叢集中有可用的容器執行個體容量，服務排程器將會啟動取代任務。如果容器執行個體容量不足，則會傳送服務事件訊息，指出該問題。

在容器執行個體上處於 RUNNING 狀態的服務的任務會轉換為 STOPPED 狀態。服務排程器會嘗試根據服務的部署類型和部署組態參數 (`minimumHealthyPercent` 和 `maximumPercent`) 來取代任務。如需詳細資訊，請參閱 Amazon ECS 服務 和 Amazon ECS 服務定義參數。

- 如果 `minimumHealthyPercent` 低於 100%，則排程器在任務取代期間可以暫時忽略 `desiredCount`。例如，`desiredCount` 為四項任務，下限 50% 允許排程器先停止兩項現有的任務，再開始兩項新的任務。如果下限為 100%，則直到替代任務視為正常運作前，服務排程器都無法移除現有的任務。如果未使用負載平衡器的服務任務為 RUNNING 狀態，則視為運作良好。如果使用

負載平衡器的服務任務為 RUNNING 狀態，且負載平衡器回報託管所在的容器執行個體運作良好，任務即視為運作良好。

> **Important**
>
> 如果您使用 Spot 執行個體且 minimumHealthyPercent 大於或等於 100%，則在 Spot 執行個體終止之前，服務將沒有足夠的時間取代任務。

- maximumPercent 參數代表任務取代期間的執行任務數量上限，這允許您定義替代批次大小。例如，如果 desiredCount 為四項任務，可先啟動四項新任務再停止四項要耗盡任務的上限為 200% (前提是有執行此作業所需的可用叢集資源)。如果上限為 100%，則在要耗盡的任務停止之前，皆無法啟動替代任務。

> **Important**
>
> 如果 minimumHealthyPercent 和 maximumPercent 兩者皆為 100%，則服務無法刪除現有任務，也無法啟動替換任務。這可以防止容器執行個體成功耗盡，並避免進行新部署。

獨立任務的耗盡行為

處於 PENDING 或 RUNNING 狀態的任何獨立任務不會受到影響；您必須等待其自行停止或手動停止。容器執行個體將維持 DRAINING 狀態。

當執行個體上執行的所有任務轉換為 STOPPED 狀態時，容器執行個體完成耗盡。容器執行個體會保持在 DRAINING 狀態，直到再次啟動或刪除為止。您可以確認容器執行個體上的任務狀態，方法是搭配使用 ListTasks 操作與 containerInstance 參數以取得執行個體上的任務清單，然後搭配使用 DescribeTasks 操作與每個任務的 Amazon Resource Name (ARN) 或 ID 以驗證任務狀態。

當您準備好讓容器執行個體重新開始託管任務時，您可以將容器執行個體的狀態從 DRAINING 變更為 ACTIVE。然後，Amazon ECS 服務排程器將再次考慮容器執行個體以進行任務放置。

程序

透過運用新的 AWS Management Console，可使用下列步驟將容器執行個體設定為耗盡。

您也可以使用 UpdateContainerInstancesState API 動作或 update-container-instances-state 命令將容器執行個體的狀態變更成 DRAINING。

AWS Management Console

1. 開啟主控台，網址為 https://console.aws.amazon.com/ecs/v2。
2. 在導覽窗格中，選擇叢集。
3. 在 Clusters (叢集) 頁面上，選擇託管您執行個體的叢集。
4. 在 Cluster : ***name*** (叢集：名稱) 頁面上，選擇 Infrastructure (基礎基礎設施) 索引標籤。然後，在 Container instances (容器執行個體)下，選取您要耗盡之每個容器執行個體的核取方塊。
5. 依序選擇動作、耗盡。

Amazon ECS 容器代理程式

Amazon ECS 代理程式是一個程序，會在向叢集註冊的每個容器執行個體上執行。它有助於您的容器執行個體與 Amazon ECS 之間的通訊。

> **Note**
>
> 在 Linux 容器執行個體上，代理程式容器掛載最上層目錄，例如 `/lib`、 `/lib64`和 `/proc`。這對於 Amazon EBS 磁碟區、`awsvpc`網路模式、Amazon ECS Service Connect 和 FireLens Amazon ECS 等 ECS 功能而言是必要的。

每個 Amazon ECS 容器代理程式版本都支援不同的功能集，並提供先前版本的錯誤修復。可能的話，我們建議您一律使用最新版本的 Amazon ECS 容器代理程式。若要將您的容器代理更新到最新版本，請參閱「更新 Amazon ECS 容器代理程式」。

若要查看每個代理版本中包含哪些功能和強化功能，請參閱 https://github.com/aws/amazon-ecs-agent/releases。

> **Important**
>
> 可靠指標的最低 Docker 版本是 Docker 版本 `v20.10.13` 及更新版本，該版本隨附於 Amazon ECS 最佳化 AMI `20220607` 及更新版本中。
> Amazon ECS 代理程式版本 `1.20.0` 和更新版本已不支援 `1.9.0` 之前的 Docker 版本。

生命週期

當 Amazon ECS 容器代理程式在您的叢集中註冊 Amazon EC2 執行個體時，Amazon EC2 執行個體會將其狀態報告為 ACTIVE，其代理程式連線狀態為 TRUE。這個容器執行個體可以接受執行任務請求。

如果您停止 (非終止) 容器執行個體，狀態會維持在 ACTIVE，但代理程式連線狀態會在幾分鐘內轉換成 FALSE。容器執行個體上執行的所有任務都會停止。如果您再次啟動容器執行個體，容器代理會與 Amazon ECS 服務重新連線，而您即可再次在執行個體上執行任務。

> **Important**
>
> 如果您停止和啟動容器執行個體，或重新啟動該執行個體，一些較舊版本的 Amazon ECS 容器代理程式會再次註冊執行個體，而未取消註冊原始容器執行個體 ID。在這種情況下，Amazon ECS 所列出的容器執行個體會比您實際在叢集中擁有的還多。(如果同一個 Amazon EC2 執行個體 ID 有重複的容器執行個體 ID，您可以安全地將列為 ACTIVE 但代理程式連線狀態為 FALSE 的重複項目取消註冊。) 最新版本的 Amazon ECS 容器代理程式已修正此問題。如需有關更新至目前版本的詳細資訊，請參閱「更新 Amazon ECS 容器代理程式」。

如果您將容器執行個體的狀態變更為 DRAINING，新的任務就不會放置在容器執行個體中。如果可能，容器執行個體上執行的所有服務任務都會移除，以便您可以執行系統更新。如需詳細資訊，請參閱耗盡 Amazon ECS 容器執行個體。

如果您取消註冊或終止容器執行個體，則容器執行個體狀態會立即變更為 INACTIVE，且在您列出容器執行個體時將不再回報該容器執行個體。不過，終止後的一小時內您仍然可以描述容器執行個體。一小時之後，即無法再進行執行個體描述。

> **Important**
>
> 您可以手動耗盡執行個體，或者建置 Auto Scaling 群組 lifecycle hook 以將執行個體狀態設定為 DRAINING。如需有關 Auto Scaling lifecycle hook 的詳細資訊，請參閱 Amazon EC2 Auto Scaling lifecycle hook。

Amazon ECS 最佳化 AMI

Amazon ECS 最佳化 AMI 的 Linux 變體使用 Amazon Linux 2 AMI 作為其基礎。透過查詢 Systems Manager 參數存放區 API，可以擷取每個變體的 Amazon Linux 2 來源 AMI 名稱。如需詳細資訊，請參閱擷取 Amazon ECS 最佳化 Linux AMI 中繼資料。透過最新的 Amazon ECS 最佳化 Amazon Linux 2 AMI 啟動我們的容器執行個體時，您會收到最新的容器代理程式版本。若要使用最新的 Amazon ECS 最佳化 Amazon Linux 2 AMI 來啟動容器執行個體，請參閱啟動 Amazon ECS Linux 容器執行個體。

其他資訊

下列頁面提供變更的其他資訊：

- GitHub 上的 Amazon ECS 代理程式變更記錄
- `ecs-init` 應用程式的原始程式碼，以及用來封裝代理程式的指令碼和組態，現在已成為代理程式儲存庫的一部分。對於 `ecs-init` 舊版本和封裝，請參閱 GitHub 上的 Amazon ecs-init 變更紀錄
- Amazon Linux 2 版本備註
- Docker 文件中的 Docker 引擎版本備註
- NVIDIA 文件中的 NVIDIA 驅動程式文件

Amazon ECS 容器代理程式組態

適用於：EC2 執行個體

Amazon ECS 容器代理程式支援許多組態選項，大部分都是透過環境變數設定。

如果是透過 Linux 版的 Amazon ECS 最佳化 AMI 來啟動容器執行個體，您可以在 `/etc/ecs/ecs.config` 檔案中設定這些環境變數，然後重新啟動代理程式。您也可以在啟動時使用 Amazon EC2 使用者資料，將這些組態變數寫入容器執行個體。如需詳細資訊，請參閱引導 Amazon ECS Linux 容器執行個體以傳遞資料。

如果是透過 Windows 版的 Amazon ECS 最佳化 AMI 來啟動容器執行個體，您可以使用 PowerShell SetEnvironmentVariable 命令設定這些環境變數，然後重新啟動代理程式。如需詳細資訊，請參閱《Amazon EC2 使用者指南》和》中的使用使用者資料輸入啟動 EC2 執行個體時執行命令the section called “引導容器執行個體”。 Amazon EC2

如果您是手動啟用 Amazon ECS 容器代理程式 (適用於非 Amazon ECS 最佳化 AMI)，則可以在用於啟用代理程式的 docker run 命令中使用這些環境變數。請搭配使用這些變數和語法 `--`

env=*VARIABLE_NAME*=*VARIABLE_VALUE*。如需私有儲存庫的身分驗證登入資料這類敏感資訊，您應該將代理環境變數存放至一個檔案中，並使用 `--env-file` *path_to_env_file* 選項一次傳遞。您可使用下列命令來新增這些變數。

```
sudo systemctl stop ecs
sudo vi /etc/ecs/ecs.config
# And add the environment variables with VARIABLE_NAME=VARIABLE_VALUE format.
sudo systemctl start ecs
```

可用參數

如需有關可用 Amazon ECS 容器代理程式組態參數的資訊，請參閱 GitHub 上的 Amazon ECS 容器代理程式。

在 Amazon S3 中存放 Amazon ECS 容器執行個體組態

Amazon ECS 容器代理程式組態是以環境變數控制。當容器代理程式啟動並隨之設定代理程式時，Linux 版的 Amazon ECS 最佳化 AMI 會在 `/etc/ecs/ecs.config` 中尋找這些變數。非敏感環境變數，例如 `ECS_CLUSTER`，可以在啟動時透過 Amazon EC2 使用者資料傳遞至容器執行個體，並寫入此檔案，而不會產生任何結果。不過，其他敏感資訊，例如您的 AWS 登入資料或`ECS_ENGINE_AUTH_DATA`變數，都不應該傳遞給使用者資料中的執行個體，或以允許它們出現在`.bash_history`檔案中`/etc/ecs/ecs.config`的方式寫入 。

在 Amazon S3 中的私有儲存貯體內存放組態資訊，並對容器執行個體 IAM 角色授予唯讀存取權，這是一種允許容器執行個體在啟動時進行設定的安全且便利方法。您可以將 `ecs.config` 文件的複本儲存在私有儲存貯體中。然後，您可以使用 Amazon EC2 使用者資料來安裝 ， AWS CLI 並在執行個體啟動`/etc/ecs/ecs.config`時將您的組態資訊複製到 。

若要在 Amazon S3 中存放 **`ecs.config`** 檔案

1. 您必須授予容器執行個體角色 (ecsInstanceRole) 許可，才能擁有 Amazon S3 的唯讀存取權。您可以將 AmazonS3ReadOnlyAccess 指派給`ecsInstanceRole`角色來執行此操作。如需有關如何將政策連接至角色的資訊，請參閱AWS Identity and Access Management 《 使用者指南》中的更新角色的許可
2. 使用下列格式建立包含有效 Amazon ECS 代理程式組態變數的 `ecs.config` 檔案。本範例將設定私有登錄檔身分驗證。如需詳細資訊，請參閱在 Amazon ECS 中使用非AWS 容器映像。

   ```
   ECS_ENGINE_AUTH_TYPE=dockercfg
   ```

```
ECS_ENGINE_AUTH_DATA={"https://index.docker.io/v1/":
{"auth":"zq212MzEXAMPLE7o6T25Dk0i","email":"email@example.com"}}
```

> **Note**
>
> 如需可用 Amazon ECS 代理程式組態變數的完整清單，請參閱 GitHub 上的 Amazon ECS 容器代理程式。

3. 若要存放您的組態檔案，請在 Amazon S3 中建立私有儲存貯體。如需詳細資訊，請參閱《Amazon Simple Storage Service 使用者指南》中的建立儲存貯體。
4. 將 `ecs.config` 檔案上傳至 S3 儲存貯體。如需詳細資訊，請參閱 Amazon Simple Storage Service 使用者指南中的上傳物件。

若要在啟動時從 Amazon S3 載入 **`ecs.config`** 檔案

1. 完成本節稍早的程序，以允許 Amazon S3 以唯讀方式存取您的容器執行個體，並將 `ecs.config` 檔案存放在私有 S3 儲存貯體。
2. 啟動新的容器執行個體，並在 EC2 使用者資料中使用下列範例指令碼。指令碼會安裝 ， AWS CLI 並將您的組態檔案複製到 `/etc/ecs/ecs.config`。如需詳細資訊，請參閱啟動 Amazon ECS Linux 容器執行個體。

```
#!/bin/bash
yum install -y aws-cli
aws s3 cp s3://your_bucket_name/ecs.config /etc/ecs/ecs.config
```

安裝 Amazon ECS 容器代理程式

如果您想要向 Amazon ECS 叢集註冊 Amazon EC2 執行個體，且該執行個體未使用基於 Amazon ECS 最佳化 AMI 的 AMI，您可以使用下列程序手動安裝 Amazon ECS 容器代理程式。若要這樣做，您可以從其中一個區域 Amazon S3 儲存貯體或 Amazon Elastic Container Registry Public 下載代理程式。如果您從其中一個區域 Amazon S3 儲存貯體下載 ，您可以選擇使用 PGP 簽章驗證容器代理程式檔案的有效性。

> **Note**
>
> Amazon ECS 和 Docker 服務的 `systemd` 單位都有一個指令，要在啟動這兩個服務前等待 `cloud-init` 完成。在您的 Amazon EC2 使用者資料完成執行前，`cloud-init` 程序不會

被視為完成。因此，透過 Amazon EC2 使用者資料啟動 Amazon ECS 或 Docker 可能會造成死鎖。若要使用 Amazon EC2 使用者資料來啟動容器代理程式，您可以使用 `systemctl enable --now --no-block ecs.service`。

在非 Amazon Linux EC2 執行個體上安裝 Amazon ECS 容器代理程式

若要在 Amazon EC2 執行個體上安裝 Amazon ECS 容器代理程式，您可以從其中一個區域 Amazon S3 儲存貯體下載代理程式並進行安裝。

> **Note**
>
> 使用非 Amazon Linux AMI 時，您的 Amazon EC2 執行個體需要 `cgroupfs` 支援 `cgroup` 驅動程式，以便 Amazon ECS 代理程式支援任務層級資源限制。如需詳細資訊，請參閱 GitHub 上的 Amazon ECS 代理程式。

每個系統架構最新的 Amazon ECS 容器代理程式檔案 (依區域) 列出如下，以供參考。

區域	區域名稱	Amazon ECS init deb 檔案	Amazon ECS init rpm 檔案
us-east-2	美國東部 (俄亥俄)	Amazon ECS init amd64 (amd64) Amazon ECS init arm64 (arm64)	Amazon ECS init x86_64 (x86_64) Amazon ECS aarch64 (aarch64)
us-east-1	美國東部 (維吉尼亞北部)	Amazon ECS init amd64 (amd64) Amazon ECS init arm64 (arm64)	Amazon ECS init x86_64 (x86_64) Amazon ECS aarch64 (aarch64)
us-west-1	美國西部 (加利佛尼亞北部)	Amazon ECS init amd64 (amd64) Amazon ECS init arm64 (arm64)	Amazon ECS init x86_64 (x86_64) Amazon ECS aarch64 (aarch64)

區域	區域名稱	Amazon ECS init deb 檔案	Amazon ECS init rpm 檔案
us-west-2	美國西部 (奧勒岡)	Amazon ECS init amd64 (amd64) Amazon ECS init arm64 (arm64)	Amazon ECS init x86_64 (x86_64) Amazon ECS aarch64 (aarch64)
ap-east-1	亞太區域 (香港)	Amazon ECS init amd64 (amd64) Amazon ECS init arm64 (arm64)	Amazon ECS init x86_64 (x86_64) Amazon ECS aarch64 (aarch64)
ap-northeast-1	亞太區域 (東京)	Amazon ECS init amd64 (amd64) Amazon ECS init arm64 (arm64)	Amazon ECS init x86_64 (x86_64) Amazon ECS aarch64 (aarch64)
ap-northeast-2	亞太區域 (首爾)	Amazon ECS init amd64 (amd64) Amazon ECS init arm64 (arm64)	Amazon ECS init x86_64 (x86_64) Amazon ECS aarch64 (aarch64)
ap-south-1	亞太區域 (孟買)	Amazon ECS init amd64 (amd64) Amazon ECS init arm64 (arm64)	Amazon ECS init x86_64 (x86_64) Amazon ECS aarch64 (aarch64)
ap-southeast-1	亞太區域 (新加坡)	Amazon ECS init amd64 (amd64) Amazon ECS init arm64 (arm64)	Amazon ECS init x86_64 (x86_64) Amazon ECS aarch64 (aarch64)

區域	區域名稱	Amazon ECS init deb 檔案	Amazon ECS init rpm 檔案
ap-southeast-2	亞太區域 (悉尼)	Amazon ECS init amd64 (amd64) Amazon ECS init arm64 (arm64)	Amazon ECS init x86_64 (x86_64) Amazon ECS aarch64 (aarch64)
ca-central-1	加拿大 (中部)	Amazon ECS init amd64 (amd64) Amazon ECS init arm64 (arm64)	Amazon ECS init x86_64 (x86_64) Amazon ECS aarch64 (aarch64)
eu-central-1	歐洲 (法蘭克福)	Amazon ECS init amd64 (amd64) Amazon ECS init arm64 (arm64)	Amazon ECS init x86_64 (x86_64) Amazon ECS aarch64 (aarch64)
eu-west-1	歐洲 (愛爾蘭)	Amazon ECS init amd64 (amd64) Amazon ECS init arm64 (arm64)	Amazon ECS init x86_64 (x86_64) Amazon ECS aarch64 (aarch64)
eu-west-2	歐洲 (倫敦)	Amazon ECS init amd64 (amd64) Amazon ECS init arm64 (arm64)	Amazon ECS init x86_64 (x86_64) Amazon ECS aarch64 (aarch64)
eu-west-3	Europe (Paris)	Amazon ECS init amd64 (amd64) Amazon ECS init arm64 (arm64)	Amazon ECS init x86_64 (x86_64) Amazon ECS aarch64 (aarch64)

區域	區域名稱	Amazon ECS init deb 檔案	Amazon ECS init rpm 檔案
sa-east-1	南美洲 (聖保羅)	Amazon ECS init amd64 (amd64) Amazon ECS init arm64 (arm64)	Amazon ECS init x86_64 Amazon ECS aarch64 (aarch64)
us-gov-east-1	AWS GovCloud （美國東部）	Amazon ECS init amd64 (amd64) Amazon ECS init arm64 (arm64)	Amazon ECS init x86_64 (x86_64) Amazon ECS aarch64 (aarch64)
us-gov-west-1	AWS GovCloud （美國西部）	Amazon ECS init amd64 (amd64) Amazon ECS init arm64 (arm64)	Amazon ECS init x86_64 (x86_64) Amazon ECS aarch64 (aarch64)

使用非 Amazon Linux AMI 在 Amazon EC2 執行個體上安裝 Amazon ECS 容器代理程式

1. 使用允許存取 Amazon ECS 的 IAM 角色來啟動 Amazon EC2 執行個體。如需詳細資訊，請參閱Amazon ECS 容器執行個體 IAM 角色。
2. 連線到您的執行個體。
3. 在您的執行個體上安裝最新版本的 Docker。
4. 檢查您的 Docker 版本，驗證您的系統符合最低版本需求。

> **Note**
>
> 可靠指標的最低 Docker 版本是 Docker 版本 `v20.10.13` 及更新版本，該版本隨附於 Amazon ECS 最佳化 AMI `20220607` 及更新版本中。
> Amazon ECS 代理程式版本 `1.20.0` 和更新版本已不支援 `1.9.0` 之前的 Docker 版本。

```
docker --version
```

5. 下載適用於您作業系統和系統架構的 Amazon ECS 代理程式檔案並加以安裝。

   對於 deb 架構：

   ```
   ubuntu:~$ curl -O https://s3.us-west-2.amazonaws.com/amazon-ecs-agent-us-west-2/amazon-ecs-init-latest.amd64.deb
   ubuntu:~$ sudo dpkg -i amazon-ecs-init-latest.amd64.deb
   ```

   對於 rpm 架構：

   ```
   fedora:~$ curl -O https://s3.us-west-2.amazonaws.com/amazon-ecs-agent-us-west-2/amazon-ecs-init-latest.x86_64.rpm
   fedora:~$ sudo yum localinstall -y amazon-ecs-init-latest.x86_64.rpm
   ```

6. 編輯 /lib/systemd/system/ecs.service 檔案，並在[Unit]區段結尾新增以下行。

   ```
   After=cloud-final.service
   ```

7. (選用) 若要將執行個體註冊到 default 叢集之外的叢集，請編輯 /etc/ecs/ecs.config 檔案並新增以下內容。以下範例會指定 MyCluster 叢集。

   ```
   ECS_CLUSTER=MyCluster
   ```

   如需這些和其他代理執行時間選項的詳細資訊，請參閱「Amazon ECS 容器代理程式組態」。

   > **Note**
   >
   > 您可以選擇性地將您的代理程式環境變數存放在 Amazon S3 中 (可在啟動時使用 Amazon EC2 使用者資料將其下載到您的容器執行個體)。針對敏感性資訊 (例如私有存放庫的身分驗證登入資料)，此為建議選項。如需詳細資訊，請參閱 在 Amazon S3 中存放 Amazon ECS 容器執行個體組態 和 在 Amazon ECS 中使用非AWS 容器映像。

8. 啟動 ecs 服務。

   ```
   ubuntu:~$ sudo systemctl start ecs
   ```

使用主機網路模式執行 Amazon ECS 代理程式

當執行 Amazon ECS 容器代理程式時，ecs-init 將使用 host 網路模式建立容器代理程式容器。對於容器代理程式容器，這是唯一支援的網路模式。

對於容器代理程式啟動的容器，這允許您封鎖對 Amazon EC2 執行個體中繼資料服務端點 (http://169.254.169.254) 的存取。這可確保容器不能從容器執行個體設定檔存取 IAM 角色憑證，並強制任務只使用 IAM 任務角色憑證。如需詳細資訊，請參閱Amazon ECS 任務 IAM 角色。

這也可讓容器代理程式不會在 docker0 橋接上爭奪連線和網路流量。

Amazon ECS 容器代理程式日誌組態參數

Amazon ECS 容器代理程式會在您的容器執行個體儲存紀錄。

針對 1.36.0 版和更新版本的容器代理程式，根據預設，紀錄會位在 Linux 執行個體的 /var/log/ecs/ecs-agent.log，以及 Windows 執行個體的 C:\ProgramData\Amazon\ECS\log\ecs-agent.log。

針對 1.35.0 版和更舊版本的容器代理程式，根據預設，紀錄會位在 Linux 執行個體的 /var/log/ecs/ecs-agent.log.*timestamp*，以及 Windows 執行個體的 C:\ProgramData\Amazon\ECS\log\ecs-agent.log.*timestamp*。

根據預設，代理程式紀錄會每小時輪換一次，最多儲存 24 個日誌。

下列是容器代理程式組態變數，可用來變更預設的代理程式紀錄行為。如需詳細資訊，請參閱Amazon ECS 容器代理程式組態。

ECS_LOGFILE

範例值：/ecs-agent.log

Linux 的預設值：Null

Windows 的預設值：Null

代理日誌應寫入其中的位置。如果您透過 執行代理程式ecs-init，這是使用 Amazon ECS 最佳化 AMI 時的預設方法，容器內路徑為 /log，並將該路徑ecs-init掛載到主機/var/log/ecs/上的 。

ECS_LOGLEVEL

範例值：crit、error、warn、info、debug

Linux 的預設值：info

Windows 的預設值：info

要記錄的細節層次。

ECS_LOGLEVEL_ON_INSTANCE

範例值：none、crit、error、warn、info、debug

Linux 上的預設值：如果 ECS_LOG_DRIVER 明確設定為非空白值，則為 none；否則會與 ECS_LOGLEVEL 的值相同

Windows 上的預設值：如果 ECS_LOG_DRIVER 明確設定為非空白值，則為 none；否則會與 ECS_LOGLEVEL 的值相同

可用於覆寫 ECS_LOGLEVEL，並設定應該記錄在執行個體上日誌檔案中的詳細資訊層級，與記錄驅動程式中記錄的層級分開。如果明確設定記錄驅動程式，則執行個體日誌預設為關閉。您可以使用此變數重新開啟。

ECS_LOG_DRIVER

範例值：awslogs、fluentd、gelf、json-file、journald、logentries、syslog、splunk

Linux 的預設值：json-file

Windows 的預設值：不適用

決定代理程式容器所使用的記錄驅動程式。

ECS_LOG_ROLLOVER_TYPE

範例值：size、hourly

Linux 的預設值：hourly

Windows 的預設值：hourly

決定容器代理程式日誌檔案是每小時輪換還是根據大小來輪換。根據預設，代理日誌檔每小時輪換一次。

ECS_LOG_OUTPUT_FORMAT

範例值：logfmt、json

Linux 的預設值：logfmt

Windows 的預設值：logfmt

決定日誌輸出格式。使用 json 格式時，日誌中的每一行都是結構化 JSON 映射。

ECS_LOG_MAX_FILE_SIZE_MB

範例值：10

Linux 的預設值：10

Windows 的預設值：10

當ECS_LOG_ROLLOVER_TYPE變數設定為 時size，此變數會決定日誌檔案的大小上限 （以 MB 為單位），然後再輪換日誌檔案。如果累積類型設為 hourly，則此變數將遭忽略。

ECS_LOG_MAX_ROLL_COUNT

範例值：24

Linux 的預設值：24

Windows 的預設值：24

決定要保留的輪換日誌檔數目。達到此限制後，就會刪除較舊的日誌檔案。

針對 1.36.0 和更新版本的容器代理程式，可參考以下使用 logfmt 格式時的範例日誌檔。

```
level=info time=2019-12-12T23:43:29Z msg="Loading configuration" module=agent.go
level=info time=2019-12-12T23:43:29Z msg="Image excluded from cleanup: amazon/amazon-
ecs-agent:latest" module=parse.go
level=info time=2019-12-12T23:43:29Z msg="Image excluded from cleanup: amazon/amazon-
ecs-pause:0.1.0" module=parse.go
level=info time=2019-12-12T23:43:29Z msg="Amazon ECS agent Version: 1.36.0, Commit:
 ca640387" module=agent.go
level=info time=2019-12-12T23:43:29Z msg="Creating root ecs cgroup: /ecs"
 module=init_linux.go
level=info time=2019-12-12T23:43:29Z msg="Creating cgroup /ecs"
 module=cgroup_controller_linux.go
level=info time=2019-12-12T23:43:29Z msg="Loading state!" module=statemanager.go
level=info time=2019-12-12T23:43:29Z msg="Event stream ContainerChange start
 listening..." module=eventstream.go
level=info time=2019-12-12T23:43:29Z msg="Restored cluster 'auto-robc'" module=agent.go
```

```
level=info time=2019-12-12T23:43:29Z msg="Restored from checkpoint file. I
 am running as 'arn:aws:ecs:us-west-2:0123456789:container-instance/auto-
robc/3330a8a91d15464ea30662d5840164cd' in cluster 'auto-robc'" module=agent.go
```

以下是使用 JSON 格式時的範例日誌檔。

```
{"time": "2019-11-07T22:52:02Z", "level": "info", "msg": "Starting Amazon Elastic
 Container Service Agent", "module": "engine.go"}
```

針對 1.35.0 和更舊版本的容器代理程式，以下是日誌檔的範例。

```
2016-08-15T15:54:41Z [INFO] Starting Agent: Amazon ECS Agent - v1.12.0 (895f3c1)
2016-08-15T15:54:41Z [INFO] Loading configuration
2016-08-15T15:54:41Z [WARN] Invalid value for task cleanup duration, will be overridden
 to 3h0m0s, parsed value 0, minimum threshold 1m0s
2016-08-15T15:54:41Z [INFO] Checkpointing is enabled. Attempting to load state
2016-08-15T15:54:41Z [INFO] Loading state! module="statemanager"
2016-08-15T15:54:41Z [INFO] Detected Docker versions [1.17 1.18 1.19 1.20 1.21 1.22]
2016-08-15T15:54:41Z [INFO] Registering Instance with ECS
2016-08-15T15:54:41Z [INFO] Registered! module="api client"
```

為私有 Docker 映像設定 Amazon ECS 容器執行個體

Amazon ECS 容器代理程式可利用基本身分驗證，使用私有登錄檔進行身分驗證。當您啟用私有登錄檔身分驗證時，您可以使用您任務定義中的私有 Docker 映像。只有使用 EC2 啟動類型的任務才支援此功能。

另一種啟用私有登錄身分驗證的方法，是使用 AWS Secrets Manager 安全地存放私有登錄檔登入資料，然後在容器定義中參考它們。這可讓您的任務使用來自私有儲存庫的映像。此方法支援使用 EC2 或 Fargate 啟動類型的任務。如需詳細資訊，請參閱在 Amazon ECS 中使用非AWS 容器映像。

Amazon ECS 容器代理程式會在啟動時尋找兩個環境變數：

- `ECS_ENGINE_AUTH_TYPE` 可指定要傳送之身分驗證資料的類型。
- `ECS_ENGINE_AUTH_DATA` 包含實際身分驗證的登入資料。

Linux 版 Amazon ECS 最佳化 AMI 會在容器執行個體啟動及每一次服務啟動時 (使用 sudo start ecs 命令)，針對這些變數掃描 `/etc/ecs/ecs.config` 檔案。不是 Amazon ECS 最佳化的 AMI 應將這些環境變數存放在一個檔案中，並使用 `--env-file` *`path_to_env_file`* 選項將它們傳遞到啟動容器代理程式的 docker run 命令。

> **Important**
>
> 我們不建議您在執行個體啟動時使用 Amazon EC2 使用者資料插入這些身分驗證環境變數，或使用 --env 選項將它們傳遞至 docker run 命令。這些方法不適合用於敏感性資料，例如身分驗證登入資料。如需將安全地將身分驗證登入資料新增至容器執行個體的詳細資訊，請參閱 在 Amazon S3 中存放 Amazon ECS 容器執行個體組態。

身分驗證格式

私有登錄檔身分驗證有兩種可用的格式，dockercfg 和 docker。

dockercfg 身分驗證格式

dockercfg 格式使用存放在執行 docker login 命令時建立之組態檔中的身分驗證資訊。您可以透過在您的本機系統上執行 docker login，輸入您的登錄使用者名稱、密碼及電子郵件地址來建立此檔案。您也可以登入容器執行個體，並在該處執行命令。根據您的 Docker 版本，這個檔案會做為 ~/.dockercfg 或 ~/.docker/config.json 儲存。

```
cat ~/.docker/config.json
```

輸出：

```
{
  "auths": {
    "https://index.docker.io/v1/": {
      "auth": "zq212MzEXAMPLE7o6T25Dk0i"
    }
  }
}
```

> **Important**
>
> 較新版本的 Docker 會如上所示，使用外部 auths 物件建立組態檔。Amazon ECS 代理程式只支援以下格式的 dockercfg 身分驗證資料，而不包含 auths 物件。如果您已安裝 jq 公用程式，即可使用以下命令擷取此資料：cat ~/.docker/config.json | jq .auths

```
cat ~/.docker/config.json | jq .auths
```

輸出：

```
{
  "https://index.docker.io/v1/": {
    "auth": "zq212MzEXAMPLE7o6T25Dk0i",
    "email": "email@example.com"
  }
}
```

在上述範例中，以下環境變數應新增到 Amazon ECS 容器代理程式在執行時間載入的環境變數檔案 (適用於 Amazon ECS 最佳化 AMI 的 `/etc/ecs/ecs.config`)。如果您不使用 Amazon ECS 最佳化 AMI，且您使用 docker run 手動啟動代理程式，請在啟動代理程式時使用 `--env-file` *path_to_env_file* 選項指定環境變數檔案。

```
ECS_ENGINE_AUTH_TYPE=dockercfg
ECS_ENGINE_AUTH_DATA={"https://index.docker.io/v1/":
{"auth":"zq212MzEXAMPLE7o6T25Dk0i","email":"email@example.com"}}
```

您可以使用以下語法設定多個私有登錄檔：

```
ECS_ENGINE_AUTH_TYPE=dockercfg
ECS_ENGINE_AUTH_DATA={"repo.example-01.com":
{"auth":"zq212MzEXAMPLE7o6T25Dk0i","email":"email@example-01.com"},"repo.example-02.com":
{"auth":"fQ172MzEXAMPLEoF7225DU0j","email":"email@example-02.com"}}
```

docker 身分驗證格式

docker 格式會使用代理程式應進行身分驗證的登錄伺服器 JSON 表示法。它也包含了該登錄所需要的身分驗證參數 (例如該帳戶的使用者名稱、密碼和電子郵件地址)。針對 Docker Hub 帳戶，JSON 表示法如以下內容所示：

```
{
  "https://index.docker.io/v1/": {
    "username": "my_name",
    "password": "my_password",
    "email": "email@example.com"
  }
}
```

在此範例中，以下環境變數應新增到 Amazon ECS 容器代理程式在執行時間載入的環境變數檔案 (適用於 Amazon ECS 最佳化 AMI 的 `/etc/ecs/ecs.config`)。如果您不使用 Amazon ECS

最佳化 AMI，且您使用 docker run 手動啟動代理程式，請在啟動代理程式時使用 `--env-file` *`path_to_env_file`* 選項指定環境變數檔案。

```
ECS_ENGINE_AUTH_TYPE=docker
ECS_ENGINE_AUTH_DATA={"https://index.docker.io/v1/":
{"username":"my_name","password":"my_password","email":"email@example.com"}}
```

您可以使用以下語法設定多個私有登錄檔：

```
ECS_ENGINE_AUTH_TYPE=docker
ECS_ENGINE_AUTH_DATA={"repo.example-01.com":
{"username":"my_name","password":"my_password","email":"email@example-01.com"},"repo.example-02
{"username":"another_name","password":"another_password","email":"email@example-02.com"}}
```

程序

使用以下程序，開啟容器執行個體的私有登錄檔。

若要在 Amazon ECS 最佳化 AMI 中啟用私有登錄檔

1. 使用 SSH 登入您的容器執行個體。
2. 開啟 `/etc/ecs/ecs.config` 檔案，為您的登錄檔和帳戶新增 `ECS_ENGINE_AUTH_TYPE` 和 `ECS_ENGINE_AUTH_DATA` 值：

   ```
   sudo vi /etc/ecs/ecs.config
   ```

   此範例會驗證 Docker Hub 使用者帳戶：

   ```
   ECS_ENGINE_AUTH_TYPE=docker
   ECS_ENGINE_AUTH_DATA={"https://index.docker.io/v1/":
   {"username":"my_name","password":"my_password","email":"email@example.com"}}
   ```

3. 檢查您的代理程式是否是使用 `ECS_DATADIR` 環境變數儲存其狀態：

   ```
   docker inspect ecs-agent | grep ECS_DATADIR
   ```

   輸出：

   ```
   "ECS_DATADIR=/data",
   ```

> **Important**
>
> 若先前的命令並未傳回 ECS_DATADIR 環境變數，您必須停止任何在此容器執行個體上執行的任務，才能停止您的代理。較新的代理會使用 ECS_DATADIR 環境變數儲存其狀態，讓您可以在任務執行中時停止和啟動它們，而不會有任何問題。如需詳細資訊，請參閱更新 Amazon ECS 容器代理程式。

4. 停止 ecs 服務：

```
sudo stop ecs
```

輸出：

```
ecs stop/waiting
```

5. 重新啟動 ecs 服務。

   - 對於 Amazon ECS 最佳化 Amazon Linux 2 AMI：

     ```
     sudo systemctl restart ecs
     ```

   - 對於 Amazon ECS 最佳化 Amazon Linux AMI：

     ```
     sudo stop ecs && sudo start ecs
     ```

6. (選用) 您可以驗證代理已在執行中，並透過查詢代理自我檢查 API 操作，查看您新的容器執行個體的一些資訊。如需詳細資訊，請參閱the section called “容器簡介”。

```
curl http://localhost:51678/v1/metadata
```

自動 Amazon ECS 任務和映像清除

每一次任務放置在容器執行個體上時，Amazon ECS 容器代理便會檢查任務中參考的映像是否是存放庫中指定標籤的最近版本。如果沒有，預設行為可讓代理程式從個別的儲存庫提取映像。若您經常更新任務和服務中的映像，您的容器執行個體儲存體可能會很快的被不再使用且可能永遠不會再使用的 Docker 映像填滿。例如，您可以使用連續整合和連續部署 (CI/CD) 管道。

> **Note**
>
> 您可以使用 ECS_IMAGE_PULL_BEHAVIOR 參數自訂 Amazon ECS 代理程式映像提取行為。如需詳細資訊，請參閱Amazon ECS 容器代理程式組態。

同樣的，屬於已停止任務的容器也會使用日誌資訊、資料磁碟區和其他成品取用容器執行個體儲存體。這些成品有助於偵錯未預期停止的容器，但其中大多數的儲存體都可在一段時間之後安全的釋放。

根據預設，Amazon ECS 容器代理會自動清除停止的任務和您容器執行個體上不再由任何任務使用的 Docker 映像。

> **Note**
>
> 自動化映像清除功能需要至少 1.13.0 版本的 Amazon ECS 容器代理程式。若要將您的代理更新到最新版本，請參閱「更新 Amazon ECS 容器代理程式」。

以下代理組態變數可用來調整您的自動化任務和映像清除體驗。如需如何在您的容器執行個體上設定這些變數的詳細資訊，請參閱「Amazon ECS 容器代理程式組態」。

`ECS_ENGINE_TASK_CLEANUP_WAIT_DURATION`

此變數可指定移除屬於已停止任務的任何容器前，應等待的時間。只要有容器仍在參考該映像，映像清除處理序便無法刪除映像。當映像不再由任何容器 (停止的容器或正在執行的容器) 參考時，映像便可進行清除。根據預設，此參數會設為 3 小時，但您可以將此期間減少至 1 秒鐘 (若您的應用程式需要的話)。如果您將值設定為小於 1 秒，則會忽略此參數。

`ECS_DISABLE_IMAGE_CLEANUP`

若您將此變數設為 `true`，則系統會關閉您的容器執行個體上的自動化映像清除，並且不會自動移除任何映像。

`ECS_IMAGE_CLEANUP_INTERVAL`

此變數指定自動化映像清除處理序檢查要刪除之映像的頻率。預設值是每 30 分鐘，但您最多可以將此期間減少至 10 分鐘，以更頻繁的移除映像。

`ECS_IMAGE_MINIMUM_CLEANUP_AGE`

此變數可指定提取映像後，到可清除該映像之間最小的時間長度。這可防止清除才剛提取的映像。預設值為 1 小時。

ECS_NUM_IMAGES_DELETE_PER_CYCLE

此變數會指定單一清除週期中可移除多少映像。預設值為 5，最小值為 1。

當 Amazon ECS 容器代理程式正在執行中，且自動化映像清除未關閉時，代理程式會檢查未被執行中或已停止的容器參考的 Docker 映像檔，檢查頻率由 ECS_IMAGE_CLEANUP_INTERVAL 變數決定。若有找到未使用的映像，且其時間比 ECS_IMAGE_MINIMUM_CLEANUP_AGE 變數指定的最小清除時間還舊，則代理最多會移除 ECS_NUM_IMAGES_DELETE_PER_CYCLE 變數所指定數量的映像。最近參考時間距離現在最久的映像會先遭到刪除。在移除映像後，代理會等待直到下一個間隔，並重複此程序。

在 Amazon ECS 上排程您的容器

Amazon Elastic Container Service (Amazon ECS) 是一種共用狀態、樂觀的並行系統，可為容器化工作負載提供彈性的排程功能。Amazon ECS 排程器會利用與 Amazon ECS API 相同的叢集狀態資訊，進行適當的置放決策。

Amazon ECS 提供為長時間執行的任務和應用程式服務排程器。它還提供為批次任務或單一執行任務執行獨立任務或排程任務的能力。您可以指定最符合您的需求的任何置放策略及執行任務的限制條件。例如，您可以指定任務跨多個可用區域或在單一可用區域內執行。另外，您還可以選擇性地將任務與您自己的自訂或第三方排程器整合。

選項	使用情況	其他資訊
服務	服務排程器適用於長時間執行的無狀態服務和應用程式。服務排程器也會選擇性地確保任務是針對 Elastic Load Balancing 負載平衡器所註冊。您可以更新由服務排程器維護的服務。這其中可能包括部署新的任務定義或變更正在執行的所需任務數。根據預設，服務排程器會將任務分散至多個可用區域。不過，您可以使用任務置放策略和限制條件，來自訂任務置放決策。	Amazon ECS 服務
獨立任務	獨立任務適合執行工作然後停止的批次任務等程序。例如，您可以在工作進入佇列時讓處理序呼叫 RunTask。任務會從佇列中提取工作、執行工作，然後結束。使用 RunTask，您可以允許預設任務置放策略在您的叢集上隨機分配任務。這可能最大限度降低單一執行個	Amazon ECS 獨立任務

選項	使用情況	其他資訊
	體取得不成比例之任務數的機率。	
排程任務	當您在叢集中以設定的間隔執行任務時，排程任務很適合，您可以使用 EventBridge Scheduler 來建立排程。您可以為備份操作或日誌掃描執行任務。您建立的 EventBridge 排程器排程可以在指定時間執行叢集中的一個或多個任務。您的排程事件可以設定為特定的時間間隔 (每隔 *N* 分鐘、小時或天執行)。此外，對於更複雜的排程，您可以使用 `cron` 運算式。	使用 Amazon EventBridge 排程器來排程 Amazon ECS 任務

運算選項

使用 Amazon ECS，您可以指定任務或服務執行所在的基礎設施。您可以使用容量提供者策略或啟動類型。

對於 Fargate，容量提供者是 Fargate 和 Fargate Spot。對於 EC2，容量提供者是具有已註冊容器執行個體的 Auto Scaling 群組。

容量提供者策略會將您的任務分散到與叢集相關聯的容量提供者。

只有已經與叢集關聯並具有 `ACTIVE` 或 `UPDATING` 狀態的容量提供者，才能在容量提供者策略中使用。您可以在建立叢集時將容量提供者關聯到叢集。

在容量提供者策略中，選用的基準值會指明指定容量提供者上至少會執行多少任務數量。容量提供者策略中只有一個容量提供者可以定義基礎。

權重值會決定使用指定容量提供者其已啟動任務總數的相對百分比。請考量下列範例。您的策略包含兩個容量提供者，且兩者的權重值都是 1。達到基準百分比時，任務會平均分配給兩個容量提供者。以此類推，假如您為 capacityProviderA 指定了權重值 1，為 capacityProviderB 指定了權重值 4。那麼，每有一個任務使用 capacityProviderA 執行，就會有四個任務使用 capacityProviderB 執行。

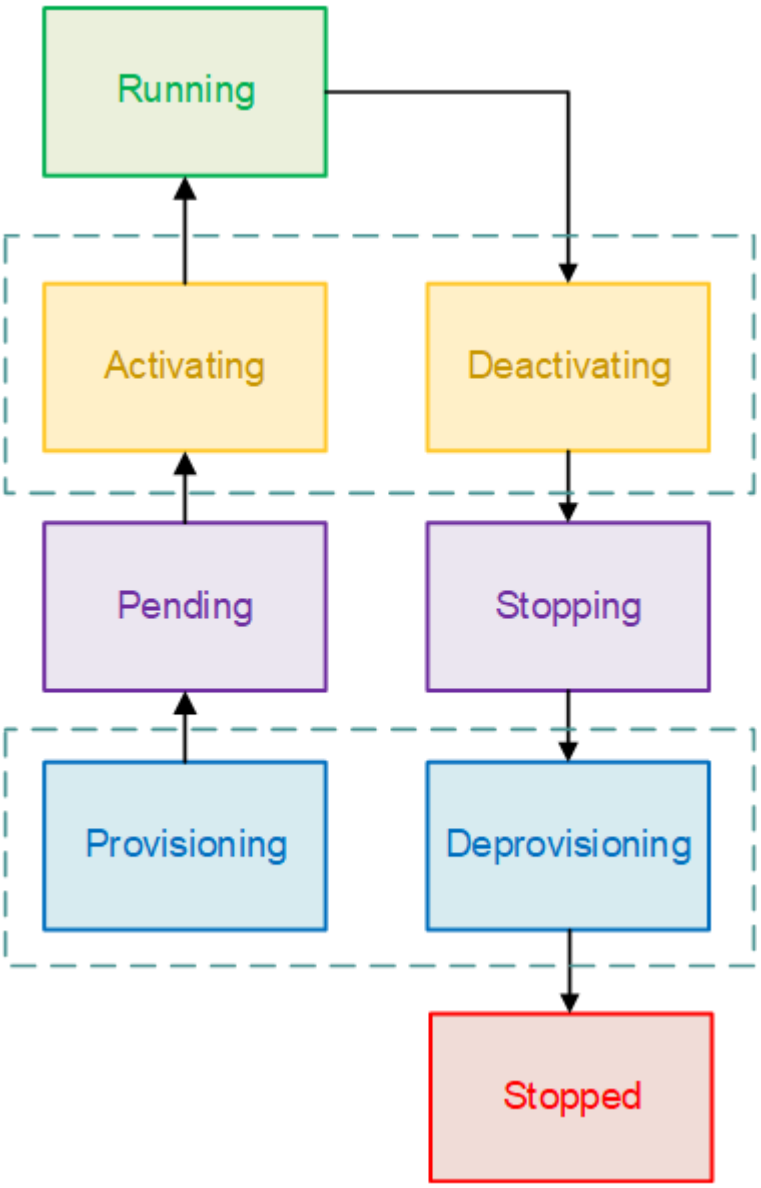
啟動類型會直接在 Fargate 或您已手動註冊到叢集的 Amazon EC2 執行個體上啟動任務。

Amazon ECS 任務生命週期

無論是以手動方式或是做為服務的一部分，當任務啟動時，它可以在自動完成或手動停止之前通過數個狀態。有些任務是做為批次任務執行的，因此會很自然地從 PENDING 前進到 RUNNING，最後前進到 STOPPED。其他可做為服務一部分的任務會不限期的持續執行，或者可根據需要擴展或縮小。

當請求任務狀態變更時 (例如停止任務或更新需要擴展或縮小的服務計數時)，Amazon ECS 容器代理程式會以任務的最後已知狀態 (`lastStatus`)，以及任務的所需狀態 (`desiredStatus`) 來追蹤這些變更。最後已知狀態和所需狀態，都可在主控台或透過使用 API 或 AWS CLI說明任務來查看。

下面的流程圖顯示任務生命週期流程。



生命週期狀態

以下是每個任務生命週期狀態的說明。

佈建中

Amazon ECS 必須執行額外的步驟，然後再啟動任務。例如，對於使用 awsvpc 網路模式的任務，需要佈建彈性網路介面。

待定

這是一個轉換狀態，Amazon ECS 在容器代理程式上等待採取進一步的動作。任務會一直處於待處理狀態，直到任務有可用的資源為止。

啟動中

這是一種轉換狀態，其中 Amazon ECS 必須在啟動任務後但在任務可轉移為 RUNNING 狀態前執行額外的步驟。這是 Amazon ECS 提取容器映像、建立容器、設定任務聯網、註冊負載平衡器目標群組，以及設定服務探索的狀態。

RUNNING (執行中)

任務成功執行中。

停用中

這是一種轉換狀態，其中 Amazon ECS 必須在任務停止之前執行其他步驟。例如，對於屬於設定為使用 Elastic Load Balancings 目標群組之服務一部分的任務，目標群組會在此狀態期間取消註冊。

停止中

這是一個轉換狀態，Amazon ECS 在容器代理程式上等待採取進一步的動作。

對於 Linux 容器，容器代理程式會傳送SIGTERM訊號，通知應用程式需要完成並關閉，然後在等待任務定義中設定的StopTimeout持續時間SIGKILL後傳送 。

取消佈建中

Amazon ECS 必須在任務停止後但在任務轉移為 STOPPED 狀態之前執行額外的步驟。例如，對於使用 awsvpc 網路模式的任務，需要分離並刪除彈性網路介面。

已停止

已成功停止任務。

如果您的任務因為錯誤而停止，請參閱 檢視 Amazon ECS 已停止的任務錯誤。

DELETED

這是任務停止時的過渡狀態。此狀態不會在主控台中顯示，但是會顯示在 describe-tasks。

Amazon ECS 如何在容器執行個體上放置任務

您可以使用任務置放來設定 Amazon ECS，將任務置放在符合特定條件的容器執行個體上，例如可用區域或執行個體類型。

以下是任務置放元件：

- 任務置放策略 - 用於選取容器執行個體以進行任務置放或終止任務的演算法。例如，Amazon ECS 可以隨機選取容器執行個體，也可以選取容器執行個體，讓任務平均分散到一組執行個體。
- 任務群組 - 一組相關的任務，例如資料庫任務。
- 任務置放限制 - 這些是必須滿足的規則，才能將任務放置在容器執行個體上。如果未符合限制條件，則任務不會放置並保持 PENDING 狀態。例如，您可以使用限制條件來僅將任務放置在特定執行個體類型上。

Amazon ECS 具有不同的啟動類型的演算法。

EC2 啟動類型

對於使用 EC2 啟動類型的任務，Amazon ECS 必須根據任務定義中指定的要求，例如 CPU 和記憶體，來決定將任務放置到何處。同樣地，當您縮減任務計數時，Amazon ECS 必須判斷要終止的任務。您可以套用任務置放策略和限制條件，來自訂 Amazon ECS 如何放置和終止任務。

預設任務置放策略取決於您是手動執行任務（獨立任務）還是在服務內執行任務。如果任務是作為 Amazon ECS 服務的一部分來執行，任務置放策略會是使用 `attribute:ecs.availability-zone` 來 `spread`。對於不在 服務中的任務，沒有預設的任務置放限制。如需詳細資訊，請參閱在 Amazon ECS 上排程您的容器。

> **Note**
>
> 任務置放策略是一種最佳作法。即使最佳置放選項無法使用，Amazon ECS 仍然會嘗試放置任務。不過，任務置放限制條件具有約束性，且可能妨礙任務置放。

您可以同時使用任務置放策略和限制條件。例如，您可以使用任務放置策略和任務放置限制，根據每個可用區域內的記憶體，跨可用區域和分箱封裝任務來分配工作，但僅適用於 G2 執行個體。

Amazon ECS 放置任務時，會使用下列程序來選取容器執行個體：

1. 識別滿足任務定義中 CPU、GPU、記憶體和連接埠需求的容器執行個體。
2. 識別滿足任務置放限制條件的容器執行個體。
3. 識別滿足任務置放策略的容器執行個體。
4. 選取用於任務置放的容器執行個體。

Fargate 啟動類型

任務放置策略和限制條件不支援使用 Fargate 啟動類型的任務。Fargate 將盡力將任務分散至可存取的可用區域。如果容量提供者同時包含 Fargate 和 Fargate Spot，則每個容量提供者的分散行為均各自獨立。

使用策略來定義 Amazon ECS 任務置放

對於使用 EC2 啟動類型的任務，Amazon ECS 必須根據任務定義中指定的要求，例如 CPU 和記憶體，來決定將任務放置到何處。同樣地，當您縮減任務計數時，Amazon ECS 必須判斷要終止的任務。您可以套用任務置放策略和限制條件，來自訂 Amazon ECS 如何放置和終止任務。

預設任務置放策略取決於您是手動執行任務（獨立任務）還是在服務內執行任務。如果任務是作為 Amazon ECS 服務的一部分來執行，任務置放策略會是使用 `attribute:ecs.availability-zone` 來 `spread`。對於不在 服務中的任務，沒有預設的任務置放限制。如需詳細資訊，請參閱在 Amazon ECS 上排程您的容器。

> **Note**
>
> 任務置放策略是一種最佳作法。即使最佳置放選項無法使用，Amazon ECS 仍然會嘗試放置任務。不過，任務置放限制條件具有約束性，且可能妨礙任務置放。

您可以同時使用任務置放策略和限制條件。例如，您可以使用任務放置策略和任務放置限制，根據每個可用區域內的記憶體，跨可用區域和分箱封裝任務來分配工作，但僅適用於 G2 執行個體。

Amazon ECS 放置任務時，會使用下列程序來選取容器執行個體：

1. 識別滿足任務定義中 CPU、GPU、記憶體和連接埠需求的容器執行個體。
2. 識別滿足任務置放限制條件的容器執行個體。
3. 識別滿足任務置放策略的容器執行個體。
4. 選取用於任務置放的容器執行個體。

您可以使用 `placementStrategy` 參數在服務定義或任務定義中指定任務置放策略。

```
"placementStrategy": [
    {
        "field": "The field to apply the placement strategy against",
        "type": "The placement strategy to use"
```

```
    }
]
```

您可以在執行任務 (RunTask)、建立新服務 (CreateService) 或更新現有服務 (UpdateService) 時指定策略。

下表說明可用的類型和欄位。

type	有效欄位值	
`binpack` 任務會放置於容器執行個體上，以便保留最少量的未使用 CPU 或記憶體。此策略會將使用中的容器執行個體數量減至最少。 當使用此策略並採取縮減動作時，Amazon ECS 會終止任務。此策略根據任務終止後保留在容器執行個體上的資源量來執行此操作。在任務終止後保留最多可用資源的容器執行個體，會終止該任務。	• cpu • memory	
`random` 隨機放置任務。	未使用	
`spread` 根據指定的值平均放置任務。服務任務會根據該服務的任務進行散佈。獨立的任務會以同一任務群組的任務為基礎分配。如需有關任務群組的詳細資訊，請參閱 群組相關的 Amazon ECS 任務 。	• `instanceId` （或 `host`，具有相同效果） • 套用至容器執行個體的任何平台或自訂屬性，例如 `attribute:ecs.availability-zone`	

type	有效欄位值	
當使用 spread 策略並執行縮減動作時，Amazon ECS 會選擇個在可用區域之間保持平衡的任務進行終止。在一個可用區域內，任務會被隨機選取。		

任務放置策略也可以為現有服務更新。如需詳細資訊，請參閱Amazon ECS 如何在容器執行個體上放置任務。

您可以透過依照您希望執行的順序建立策略陣列來建立使用多個策略的任務放置策略。例如，如果您想要跨可用區域分散任務，然後在每個可用區域內根據記憶體對任務進行分箱封裝，請指定可用區域策略，然後指定記憶體策略。如需策略範例，請參閱 Amazon ECS 任務置放策略範例。

Amazon ECS 任務置放策略範例

您可以使用下列動作指定任務置放策略：CreateService、UpdateService 和 RunTask。

範例

- 將任務平均分佈至可用區域
- 將任務平均分佈至所有執行個體
- 根據記憶體對任務進行分箱封裝
- 隨機放置任務。
- 將任務平均分佈至各個可用區域，然後將任務平均分佈至每個可用區域內的執行個體
- 將任務平均分佈至各個可用區域，然後在每個可用區域內根據記憶體對任務進行分箱封裝
- 將任務平均分佈至執行個體，然後根據記憶體對任務進行分箱封裝

將任務平均分佈至可用區域

下列策略會將任務平均分散到各個可用區域。

```
"placementStrategy": [
    {
        "field": "attribute:ecs.availability-zone",
        "type": "spread"
    }
```

```
]
```

將任務平均分佈至所有執行個體

下列策略會將任務平均分散到所有執行個體。

```
"placementStrategy": [
    {
        "field": "instanceId",
        "type": "spread"
    }
]
```

根據記憶體對任務進行分箱封裝

下列策略會根據記憶體對任務進行分箱封裝。

```
"placementStrategy": [
    {
        "field": "memory",
        "type": "binpack"
    }
]
```

隨機放置任務。

下列策略會隨機放置任務。

```
"placementStrategy": [
    {
        "type": "random"
    }
]
```

將任務平均分佈至各個可用區域，然後將任務平均分佈至每個可用區域內的執行個體

下列策略會將任務平均分散到各個可用區域，然後將任務平均分散到每個可用區域內的執行個體。

```
"placementStrategy": [
    {
        "field": "attribute:ecs.availability-zone",
        "type": "spread"
```

```
    },
    {
        "field": "instanceId",
        "type": "spread"
    }
]
```

將任務平均分佈至各個可用區域，然後在每個可用區域內根據記憶體對任務進行分箱封裝

下列策略會將任務平均分散到各個可用區域，然後根據每個可用區域內的記憶體對任務進行分箱封裝。

```
"placementStrategy": [
    {
        "field": "attribute:ecs.availability-zone",
        "type": "spread"
    },
    {
        "field": "memory",
        "type": "binpack"
    }
]
```

將任務平均分佈至執行個體，然後根據記憶體對任務進行分箱封裝

下列策略會將任務平均分佈至所有執行個體，然後在每個執行個體內根據記憶體對任務進行分箱封裝。

```
"placementStrategy": [
    {
        "field": "instanceId",
        "type": "spread"
    },
    {
        "field": "memory",
        "type": "binpack"
    }
]
```

群組相關的 Amazon ECS 任務

您可以識別一組相關任務，並將其放在任務群組中。在使用 spread 任務置放策略時，會將所有具有相同任務群組名稱的任務視為一組。例如，假設您在一個叢集執行不同的應用程式，例如資料庫和 Web 伺服器。為了確保您在各個可用區域中之資料庫的平衡，請將它們新增至名為 databases 的任

務群組，然後使用 spread 任務置放策略。如需詳細資訊，請參閱使用策略來定義 Amazon ECS 任務置放。

任務群組也可用作任務放置限制條件。當您在 memberOf 限制條件中指定任務群組時，任務只會傳送到在指定任務群組中執行的容器執行個體。如需範例，請參閱「Amazon ECS 任務置放限制範例」。

根據預設，若未指定自訂任務群組名稱，獨立任務會使用任務定義系列名稱 (如 family:my-task-definition) 做為任務群組名稱。若任務做為服務的一部分啟動，它將使用服務名稱做為任務群組名稱，而且該名稱無法被變更。

任務群組適用下列需求。

- 任務群組名稱必須在 255 個字元以下。
- 每個任務只能在一個群組中。
- 啟動任務之後，就無法修改其任務群組。

定義 Amazon ECS 用於任務的容器執行個體

任務置放限制是有關容器執行個體的規則，Amazon ECS 會使用此規則來判斷任務是否允許在執行個體上執行。至少有一個容器執行個體必須符合限制條件。如果沒有符合條件限制的執行個體，則任務將會保持 PENDING 狀態。當您建立新服務或更新現有服務時，您可以為服務的任務指定任務放置限制條件。

您可以使用 placementConstraint 參數在服務定義、任務定義或任務中指定任務置放限制。

```
"placementConstraints": [
    {
        "expression": "The expression that defines the task placement constraints",
        "type": "The placement constraint type to use"
    }
]
```

下表說明如何使用 參數。

Constraint type (限制條件類型)	可以在 時指定
distinctInstance 將每個作用中任務放在不同的容器執行個體上。	• 執行任務 RunTask • 建立新的服務 CreateService，

Constraint type (限制條件類型)	可以在 時指定	
Amazon ECS 會查看任務置放所需的任務狀態。例如，如果現有任務的所需狀態為 STOPPED，（但最後一個狀態不是），則即使distinctInstance 置放限制條件，新的傳入任務仍可放置在同一個執行個體上。因此，您可能會在同一個執行個體RUNNING上看到最後狀態為 的 2 個任務。 **Important** 我們建議希望為其任務進行強隔離的客戶使用 Fargate。Fargate 會在硬體虛擬化環境中執行每個任務。這可確保這些容器化工作負載不會與其他任務共用網路介面、Fargate 暫時性儲存、CPU 或記憶體。如需詳細資訊，請參閱 的安全概觀 AWS Fargate。		
memberOf 在滿足運算式的容器執行個體上放置任務。	• 執行任務 RunTask • 建立新的服務 CreateService， • 建立新的任務定義 RegisterTaskDefinition • 建立新的任務定義修訂 RegisterTaskDefinition • 更新服務 UpdateService	

當您使用memberOf限制類型時，您可以使用叢集查詢語言來建立表達式，該語言定義 Amazon ECS 可以放置任務的容器執行個體。表達式可讓您依屬性分組容器執行個體。表達式會進入 的 `expression` 參數`placementConstraint`。

Amazon ECS 容器執行個體屬性

您可以將自訂中繼資料新增至容器執行個體，稱為屬性。每個屬性都有名稱和選用字串值。您可以使用 Amazon ECS 所提供的內建屬性，或定義自訂屬性。

以下章節包含範例內建屬性、選擇性屬性和自訂屬性。

內建屬性

Amazon ECS 會將下列屬性自動套用至您的容器執行個體。

`ecs.ami-id`

用來啟動執行個體的 AMI ID。此屬性的範例值為 `ami-1234abcd`。

`ecs.availability-zone`

執行個體的可用區域。此屬性的範例值為 `us-east-1a`。

`ecs.instance-type`

執行個體的執行個體類型。此屬性的範例值為 `g2.2xlarge`。

`ecs.os-type`

執行個體的作業系統。此屬性的可能值為 `linux` 和 `windows`。

`ecs.os-family`

執行個體的作業系統版本。

若為 Linux 執行個體，有效值為 `LINUX`。若為 Windows 執行個體，ECS 會以 `WINDOWS_SERVER_<OS_Release>_<FULL or CORE>` 格式設定值。有效值為 `WINDOWS_SERVER_2022_FULL`、`WINDOWS_SERVER_2022_CORE`、`WINDOWS_SERVER_20H2_CORE`、`W` 與 `WINDOWS_SERVER_2016_FULL`。

這對於 Windows 容器很重要， Windows containers on AWS Fargate 因為每個 Windows 容器的作業系統版本必須符合主機的作業系統版本。如果容器映像的 Windows 版本與主機不同，則容器不會啟動。如需詳細資訊，請參閱 Microsoft 文件網站上的 Windows 容器版本相容性。

如果您的叢集執行多個 Windows 版本，您可以使用置放條件限制：`memberOf(attribute:ecs.os-family == WINDOWS_SERVER_<OS_Release>_<FULL`

or CORE>)，確保將任務置放在執行相同版本的 EC2 執行個體上。如需詳細資訊，請參閱the section called “擷取 Amazon ECS 最佳化 Windows AMI 中繼資料”。

`ecs.cpu-architecture`

執行個體的 CPU 架構。此屬性的範例值為 `x86_64` 和 `arm64`。

`ecs.vpc-id`

執行個體啟動所在位置的 VPC。此屬性的範例值為 `vpc-1234abcd`。

`ecs.subnet-id`

執行個體使用的子網路。此屬性的範例值為 `subnet-1234abcd`。

選擇性屬性

Amazon ECS 可能會將以下屬性新增到您的容器執行個體。

`ecs.awsvpc-trunk-id`

如果此屬性存在，執行個體則具有一個幹線網路界面。如需詳細資訊，請參閱增加 Amazon ECS Linux 容器執行個體網路介面。

`ecs.outpost-arn`

如果此屬性存在，它會包含 Outpost 的 Amazon Resource Name (ARN)。如需詳細資訊，請參閱the section called “上的 Amazon Elastic Container Service AWS Outposts”。

`ecs.capability.external`

若存在此屬性，執行個體將被識別為外部執行個體。如需詳細資訊，請參閱外部啟動類型的 Amazon ECS 叢集。

自訂屬性

您可以將自訂屬性套用至容器執行個體。例如，您可以定義名稱為 "stack" 且值為 "prod" 的屬性。

在指定自訂屬性時，必須考慮下列事項。

- `name` 必須包含 1 至 128 個字元，而名稱可能包含字母 (大小寫)、數字、連字號、底線、正斜線、反斜線或句號。
- `value` 必須包含 1 至 128 個字元，而且可能包含字母 (大小寫)、數字、連字號、底線、句號、@ 符號、正斜線、反斜線、冒號或空格。值不得包含任何前置或結尾空格。

建立表達式以定義 Amazon ECS 任務的容器執行個體

叢集查詢是可讓您將物件分組的運算式。例如，您可以依屬性 (例如可用區域、執行個體類型或自訂中繼資料) 將容器執行個體分組。如需詳細資訊，請參閱Amazon ECS 容器執行個體屬性。

在您定義一組容器執行個體之後，即可自訂 Amazon ECS 以根據群組在容器執行個體上放置任務。如需詳細資訊，請參閱 以 Amazon ECS 任務執行應用程式 和 使用 主控台建立 Amazon ECS 服務。您也可以在列出容器執行個體時套用群組篩選條件。

運算式語法

運算式的語法如下：

```
subject operator [argument]
```

主旨

要評估的屬性或欄位。

`agentConnected`

依 Amazon ECS 容器代理程式連線狀態來選取容器執行個體。您可以使用此篩選條件來搜尋具有代理程式已中斷連線的執行個體容器。

有效運算子：equals (==)、not_equals (!=)、in、not_in (!in)、matches (=~)、not_matches (!~)

`agentVersion`

依 Amazon ECS 容器代理程式版本來選取容器執行個體。您可以使用此篩選條件，尋找正在執行過期版本 Amazon ECS 容器代理程式的執行個體。

有效運算子：equals (==)、not_equals (!=)、greater_than (>)、greater_than_equal (>=)、less_than (<)、less_than_equal (<=)

`attribute:`*`attribute-name`*

依屬性選取容器執行個體。如需詳細資訊，請參閱Amazon ECS 容器執行個體屬性。

`ec2InstanceId`

依 Amazon EC2 執行個體 ID 選取容器執行個體。

有效運算子：equals (==)、not_equals (!=)、in、not_in (!in)、matches (=~)、not_matches (!~)

`registeredAt`

依容器執行個體註冊日期來選取容器執行個體。您可以使用此篩選條件，尋找新註冊執行個體或非常舊的執行個體。

有效運算子：equals (==)、not_equals (!=)、greater_than (>)、greater_than_equal (>=)、less_than (<)、less_than_equal (<=)

有效日期格式：2018-06-18T22:28:28+00:00, 2018-06-18T22:28:28Z, 2018-06-18T22:28:28, 2018-06-18

`runningTasksCount`

依執行中的任務數量來選取容器執行個體。您可以使用此篩選條件來尋找空白或接近空白 (在其上執行的任務很少) 的執行個體。

有效運算子：equals (==)、not_equals (!=)、greater_than (>)、greater_than_equal (>=)、less_than (<)、less_than_equal (<=)

`task:group`

依任務群組選取容器執行個體。如需詳細資訊，請參閱群組相關的 Amazon ECS 任務 。

運算子

比較運算子。下列是支援的運算子。

運算子	描述
==、equals	字串相等
!=、not_equals	字串不相等
>、greater_than	大於
>=、greater_than_equal	大於或等於
<、less_than	小於
<=、less_than_equal	小於或等於
exists	主旨存在

運算子	描述
!exists、not_exists	主旨不存在
in	引數清單中的值
!in、not_in	不在引數清單中的值
=~、matches	模式相符
!~、not_matches	模式不相符

> **Note**
>
> 單一運算式不能包含括號。不過，可以使用括號指定複合運算式中的優先順序。

引數

對於許多運算子而言，引數是一種常值。

`in` 和 `not_in` 運算子需要有參數清單做為引數。您可以指定引數清單，如下所示：

```
[argument1, argument2, ..., argumentN]
```

matches 和 not_matches 運算子需要有符合 Java 一般運算式語法的引數。如需詳細資訊，請參閱 java.util.regex.Pattern。

複合運算式

您可以使用下列布林值運算子來合併運算式：

- &&, 和
- ||, 或
- !, 非

您可以使用括號來指定優先順序：

```
(expression1 or expression2) and expression3
```

範例運算式

下列是範例運算式。

範例：字串相等

下列運算式會選取具有指定之執行個體類型的執行個體。

```
attribute:ecs.instance-type == t2.small
```

範例：引數清單

下列運算式會選取 us-east-1a 或 us-east-1b 可用區域中的執行個體。

```
attribute:ecs.availability-zone in [us-east-1a, us-east-1b]
```

範例：複合運算式

下列運算式會選取不在 us-east-1d 可用區域中的 G2 執行個體。

```
attribute:ecs.instance-type =~ g2.* and attribute:ecs.availability-zone != us-east-1d
```

範例：任務親和性

下列運算式會選取在 `service:production` 群組中託管任務的執行個體。

```
task:group == service:production
```

範例：任務反親和性

下列運算式會選取未在資料庫群組中託管任務的執行個體。

```
not(task:group == database)
```

範例：執行中任務計數

下列運算式會選取僅執行一項任務的執行個體。

```
runningTasksCount == 1
```

範例：Amazon ECS 容器代理程式版本

下列運算式會選取正在執行低於 1.14.5 版本容器代理程式的執行個體。

```
agentVersion < 1.14.5
```

範例：執行個體註冊時間

下列運算式會選取在 2018 年 2 月 13 日之前註冊的執行個體。

```
registeredAt < 2018-02-13
```

範例：Amazon EC2 執行個體 ID

下列運算式會選取具有下列 Amazon EC2 執行個體 ID 的執行個體。

```
ec2InstanceId in ['i-abcd1234', 'i-wxyx7890']
```

Amazon ECS 任務置放限制範例

下列是任務置放條件限制的範例。

此範例使用 memberOf 限制條件將任務放置在 t2 執行個體上。可以使用下列動作來指定：CreateService、UpdateService、RegisterTaskDefinition 和 RunTask。

```
"placementConstraints": [
    {
        "expression": "attribute:ecs.instance-type =~ t2.*",
        "type": "memberOf"
    }
]
```

範例使用 memberOf 限制條件以便將複本任務與常駐程式服務 daemon-service 任務群組中的任務放置到執行個體上，還要顧及其他被指定的任何任務置放策略。此限制可確保常駐程式服務任務在執行複本服務任務之前放置在 EC2 執行個體上。

以常駐程式服務的名稱取代 daemon-service。

```
"placementConstraints": [
    {
        "expression": "task:group == service:daemon-service",
        "type": "memberOf"
    }
```

```
]
```

範例使用 memberOf 限制條件以便將任務與 databases 任務群組中的其他任務放置到執行個體上，還要顧及其他被指定的任何任務置放策略。如需有關任務群組的詳細資訊，請參閱 群組相關的 Amazon ECS 任務 。可以使用下列動作來指定：CreateService、UpdateService、RegisterTaskDefinition 和 RunTask。

```
"placementConstraints": [
    {
        "expression": "task:group == databases",
        "type": "memberOf"
    }
]
```

distinctInstance 條件限制會在不同執行個體的群組中放置每個任務。可以使用下列動作來指定：CreateService、UpdateService 和 RunTask

Amazon ECS 會查看任務置放所需的任務狀態。例如，如果現有任務的所需狀態為 STOPPED，（但最後一個狀態不是），則即使distinctInstance置放限制條件，新的傳入任務仍可放置在同一個執行個體上。因此，您可能會在同一個執行個體RUNNING上看到最後狀態為 的 2 個任務。

```
"placementConstraints": [
    {
        "type": "distinctInstance"
    }
]
```

Amazon ECS 獨立任務

當您的應用程式執行一些工作，然後停止，例如批次程序時，您可以將應用程式做為任務執行。如果您想要執行任務一次，您可以使用 主控台 AWS CLI、 APIs或 SDKs。

如果您需要以速率為基礎、以 Cron 為基礎或一次性排程執行應用程式，您可以使用 EventBridge Scheduler 建立排程。

任務工作流程

當您啟動 Amazon ECS 任務（獨立任務或由 Amazon ECS 服務執行）時，系統會建立任務並最初移動至 PROVISIONING 狀態。當任務處於 PROVISIONING 狀態時，任務和容器都不存在，因為 Amazon ECS 需要尋找運算容量來放置任務。

Amazon ECS 會根據您的啟動類型或容量提供者組態，為您的任務選取適當的運算容量。您可以搭配 Fargate 和 Amazon EC2 啟動類型使用容量提供者和容量提供者策略。使用 Fargate，您不需要考慮佈建、設定和擴展叢集容量。Fargate 會為您的任務處理所有基礎設施管理。對於 EC2 啟動類型，您可以透過將 Amazon EC2 執行個體註冊到叢集來管理叢集容量，也可以使用叢集自動擴展來簡化運算容量管理。叢集自動擴展負責動態擴展叢集容量，以便您可以專注於執行任務。Amazon ECS 會根據您在任務定義中指定的要求，例如 CPU 和記憶體，以及您的置放限制條件和策略，來決定將任務置放到何處。如需詳細資訊，請參閱 Amazon ECS 如何在容器執行個體上放置任務。

如果您使用已啟用受管擴展的容量提供者，則由於運算容量不足而無法啟動的任務會移至 PROVISIONING 狀態，而不是立即失敗。找到放置任務的容量後，Amazon ECS 會為處於 awsvpc 模式的任務佈建必要的附件（例如，彈性網路界面 (ENIs))。它使用 Amazon ECS 容器代理程式來提取容器映像，然後啟動容器。佈建完成且相關容器啟動後，Amazon ECS 會將任務移至 RUNNING 狀態。如需任務狀態的相關資訊，請參閱 Amazon ECS 任務生命週期。

最佳化 Amazon ECS 任務啟動時間

為了加快任務啟動速度，請考慮下列建議。

- 快取容器映像和 binpack 執行個體

  如果您使用 EC2 啟動類型，則可以設定 Amazon ECS 容器代理程式提取行為至 ECS_IMAGE_PULL_BEHAVIOR：prefer-cached。如果沒有快取的映像，則會從遠端提取映像。否則，將使用執行個體上的快取映像，容器的自動映像清除已關閉，以確保快取映像不會移除。這可減少後續啟動的影像提取時間。當您的容器執行個體具有高任務密度時，快取的效果會更大，您可以使用binpack置放策略進行設定。快取容器映像對於通常具有大型（數十 GBs容器映像大小的視窗型工作負載特別有用。使用binpack置放策略時，您也可以考慮使用彈性網路介面 (ENI) 中繼，在每個容器執行個體上放置具有awsvpc網路模式的更多任務。ENI 中繼會增加您可以在 awsvpc 模式下執行的任務數量。例如，可能支援僅同時執行 2 個任務的 c5.large 執行個體，可以使用 ENI 中繼執行最多 10 個任務。

- 選擇最佳網路模式

  雖然網路awsvpc模式很理想，但此網路模式本質上可以增加任務啟動延遲，因為對於awsvpc模式中的每個任務，Amazon ECS 工作流程需要透過叫用 Amazon EC2 APIs 來佈建和連接 ENI，這些 API 會為您的任務啟動增加幾秒鐘的額外負荷。相反地，使用awsvpc網路模式的主要優點是每個任務都有一個安全群組來允許或拒絕流量。這表示您有更大的彈性，可以更精細地控制任務和服務之間的通訊。如果部署速度是您的優先順序，您可以考慮使用 bridge 模式來加速任務啟動。如需詳細資訊，請參閱the section called “AWSVPC 網路模式”。

- 追蹤任務啟動生命週期以尋找最佳化機會

通常很難知道應用程式啟動所需的時間。在應用程式啟動期間啟動容器映像、執行啟動指令碼和其他組態，可能需要相當驚人的時間。您可以使用任務中繼資料端點來張貼指標，在您的應用程式準備好提供流量時，追蹤從 `ContainerStartTime`到 的應用程式啟動時間。使用此資料，您可以了解應用程式對總啟動時間的貢獻，並尋找可以減少不必要的應用程式特定額外負荷並最佳化容器映像的區域。如需詳細資訊，請參閱最佳化 Amazon ECS容量和可用性。

- 選擇最佳執行個體類型 （適用於 EC2 啟動類型）

  選擇正確的執行個體類型是根據您在任務上設定的資源保留 （例如 CPU、記憶體）。因此，調整執行個體的大小時，您可以計算單一執行個體上可以放置多少任務。妥善放置任務的簡單範例，是在 m5.large 執行個體中託管 4 個需要 0.5 vCPU 和 2GB 記憶體保留的任務 （支援 2 個 vCPU 和 8 GB 記憶體）。此任務定義的保留充分利用執行個體的資源。

以 Amazon ECS 任務執行應用程式

您可以使用 建立一次性程序的任務 AWS Management Console。

建立獨立任務 (AWS Management Console)

1. 開啟主控台，網址為 https://console.aws.amazon.com/ecs/v2。
2. Amazon ECS 主控台可讓您從叢集詳細資訊頁面或任務定義修訂清單建立獨立任務。根據您選擇的資源頁面，使用下列步驟來建立獨立任務。

啟動服務的資源	步驟
叢集詳細資訊頁面...	a. 在叢集頁面，選取您要在哪個叢集中建立服務。 b. 在 Tasks (任務) 標籤中，選擇 Run new task (執行新任務)。
任務定義修訂頁面...	a. 在任務定義頁面上，選擇任務定義系列以顯示該系列的修訂。 b. 選取您要使用的修訂。

啟動服務的資源	步驟	
	c. 從部署功能表中，選擇執行任務。	

3. （選用）運算組態（進階）區段可讓您選擇任務的分佈方式。您可以使用容量提供者策略或啟動類型。若要使用容量提供者策略，您必須在叢集層級設定容量提供者。如果您尚未將叢集設定為使用容量提供者，請改用啟動類型。

分發方法	步驟	
容量提供者策略	a. 在 Compute options (運算選項) 區段中，選取 Capacity provider strategy (容量提供者策略)。 b. 選擇策略： • 若要使用叢集的預設容量提供者策略，選擇使用叢集預設。 • 若您的叢集沒有預設容量提供者策略，或要使用自訂策略，選擇 Use custom (使用自訂)、Add capacity provider strategy (新增容量提供者策略) 並透過指定 Base (基礎)、Capacity provider (容量提供者) 和 Weight (權重) 來定義您的自訂容量提供者策略。 **Note** 若要在策略中使用容量提供者，容量提供	

分發方法	步驟
	者必須與叢集相關聯。
啟動類型	a. 在運算選項區段中，選取啟動類型。 b. 針對 啟動類型，選擇啟動類型。 c. (選用) 當指定 Fargate 啟動類型時，針對平台版本，指定要使用的平台版本。如果未指定平台版本，將使用 LATEST 平台版本。

4. 針對 Application type (應用程式類型)，選擇 Task (任務)。
5. 在任務定義中，選擇任務定義系列和修訂。

> **Important**
>
> 主控台會驗證選擇，以確保所選任務定義系列和修訂與定義的運算組態相容。

6. 針對 Desired tasks (所需任務)，輸入要啟動的任務數目。
7. 如果任務定義使用 awsvpc 網路模式，請展開聯網。使用下列步驟以指定自訂組態。
   a. 針對 VPC，選擇要使用的 VPC。
   b. 針對子網路，在 VPC 中選擇一個或多個子網路，而任務排程器在放置任務時會考慮該 VPC。

   > **Important**
   >
   > awsvpc 網路模式只支援私有子網路。任務不會接收公有 IP 地址。因此必須使用 NAT 閘道進行對外網際網路存取，且入站網際網路流量會透過負載平衡器進行路由。

   c. 在安全群組中，您可以選擇現有的安全群組，或建立新的安全群組。若要使用現有的安全群組，選擇該安全群組並移至下一個步驟。若要建立新的安全群組，請選擇 建立新安全群組。您必須指定安全群組名稱、描述，然後為該安全群組新增一條或更多傳入規則。

d. 針對 Public IP (公有 IP)，選擇是否為任務的彈性網路界面 (ENI) 自動指派公有 IP 地址。

AWS Fargate 在公有子網路中執行時，任務可以指派公有 IP 地址，以便他們有網際網路的路由。EC2 任務無法使用此欄位指派公有 IP。如需詳細資訊，請參閱 Fargate 啟動類型的 Amazon ECS 任務聯網選項，以及為 Amazon ECS 任務配置網路介面。

8. 如果您的任務使用與部署時組態相容的資料磁碟區，您可以透過擴展磁碟區來設定磁碟區。

磁碟區名稱和磁碟區類型是在建立任務定義修訂時設定，且無法在執行獨立任務時變更。若要更新磁碟區名稱和類型，您必須建立新的任務定義修訂，並使用新的修訂執行任務。

設定此磁碟區類型	執行此作業
Amazon EBS	a. 針對 EBS 磁碟區類型，選擇您要連接至任務的 EBS 磁碟區類型。 b. 針對大小 (GiB)，以 GB (GiB) 為單位輸入磁碟區大小的有效值。您可以指定至少 1 GiB 和最大 16，384 GiB 磁碟區大小。除非您提供快照 ID，否則需要此值。 c. 針對 IOPS，輸入磁碟區應提供的輸入/輸出操作 (IOPS) 數目上限。此值僅適用於 `io1`、`io2`、 和 gp3磁碟區類型。 d. 針對輸送量 (MiB/s)，以每秒 MB 為單位 (MiBps 或 MiB/s)，輸入磁碟區應提供的輸送量。此值只能針對gp3磁碟區類型設定。 e. 對於快照 ID，選擇現有的 Amazon EBS 磁碟區快照，或者如果您想要從快照建立磁碟區，請輸入快

設定此磁碟區類型	執行此作業	
	照的 ARN。您也可以透過不選擇或輸入快照 ID 來建立新的空白磁碟區。 f. 對於終止政策，如果您想要在任務終止後保留為連接至任務而設定的磁碟區，請取消選取核取方塊。根據預設，在任務終止時，會刪除連接至任務的 EBS 磁碟區。 g. 針對檔案系統類型，選擇用於磁碟區上資料儲存和擷取的檔案系統類型。您可以選擇作業系統預設值或特定檔案系統類型。Linux 的預設值為 XFS。對於從快照建立的磁碟區，您必須指定建立快照時磁碟區所使用的相同檔案系統類型。如果檔案系統類型不相符，任務將無法啟動。 h. 針對基礎設施角色，選擇具有必要許可的 IAM 角色，以允許 Amazon ECS 管理任務的 Amazon EBS 磁碟區。您可以將 `AmazonECSInfrastructureRolePolicyForVolumes` 受管政策連接至 角色，也可以使用政策做為建立和連接您自己的政策的指南，該政策具有符合您特	

設定此磁碟區類型	執行此作業	
	定需求的許可。如需必要許可的詳細資訊，請參閱Amazon ECS 基礎設施 IAM 角色。 i. 對於加密，如果您想要依預設設定使用 Amazon EBS 加密，請選擇預設。如果您的帳戶已預設設定加密，則磁碟區將使用設定中指定的 AWS Key Management Service (AWS KMS) 金鑰加密。如果您選擇預設，且 Amazon EBS 預設加密未開啟，則磁碟區將取消加密。 如果您選擇自訂，您可以指定 AWS KMS key 磁碟區加密的選擇。 如果您選擇無，除非您已預設設定加密，或者您從加密的快照建立磁碟區，否則磁碟區將會取消加密。 j. 如果您已選擇 Custom for Encryption，則必須指定要使用 AWS KMS key 的 。針對 KMS 金鑰，選擇 AWS KMS key 或輸入金鑰 ARN。如果您選擇使用對稱客戶受管金鑰來加密磁碟區，請確定您已在 AWS KMS key 政策中定義	

設定此磁碟區類型	執行此作業	
	正確的許可。如需詳細資訊，請參閱 Amazon EBS 磁碟區的資料加密。 k. （選用）在標籤下，您可以透過從任務定義傳播標籤或提供您自己的標籤，將標籤新增至 Amazon EBS 磁碟區。 如果您想要從任務定義傳播標籤，請選擇傳播標籤的任務定義。如果您選擇不傳播，或者如果您不選擇值，則不會傳播標籤。 如果您想要提供自己的標籤，請選擇新增標籤，然後針對您新增的每個標籤提供金鑰和值。 如需標記 Amazon EBS 磁碟區的詳細資訊，請參閱標記 Amazon EBS 磁碟區。	

9. (選用) 若要使用預設以外的任務置放策略，請展開任務置放，然後從下列選項中選擇。

   如需詳細資訊，請參閱Amazon ECS 如何在容器執行個體上放置任務。

   - AZ 平衡分散 – 將任務分散到可用區域和可用區域中的容器執行個體。
   - AZ Balanced BinPack – 將任務分散到可用區域和具有最少可用記憶體的容器執行個體。
   - BinPack – 根據最低可用 CPU 或記憶體數量來分發任務。
   - 每個主機一個任務 – 在每個容器執行個體上最多放置一個來自 服務的任務。
   - 自訂 – 定義您自己的任務置放策略。

如果選擇自訂，則請定義置放任務的演算法，以及在任務置放期間考慮的規則。

- 在策略下，針對類型和欄位，選擇演算法以及要用於演算法的實體。

  您最多可新增 5 項策略。
- 在 Constraint (限制條件)，針對 Type (類型) 和 Expression (運算式)，選擇要用於限制條件的規則與屬性。

  例如，若要設定限制條件，以將任務置放在 T2 執行個體上，針對運算式，請輸入 attribute:ecs.instance-type =~ t2.*。

  您最多可新增 10 個限制條件。

10. (選用) 若要覆寫任務 IAM 角色或任務定義中定義的任務執行角色，請展開 Task overrides (任務覆寫)，然後完成下列步驟：

    a. 在任務角色中，為此任務選擇 IAM 角色。如需詳細資訊，請參閱Amazon ECS 任務 IAM 角色。

       僅顯示具有 `ecs-tasks.amazonaws.com` 信任關係的角色。如需為您的任務建立 IAM 角色的說明，請參閱 建立任務 IAM 角色。

    b. 在任務執行角色中，選擇任務執行角色。如需詳細資訊，請參閱Amazon ECS 任務執行 IAM 角色。

11. (選用) 若要覆寫容器命令和環境變數，請展開 Container Overrides (容器覆寫)，然後展開容器。

    - 若要將命令傳送至任務定義命令以外的容器，針對 Command override (命令複寫)，請輸入 Docker 命令。
    - 若要新增環境變數，選擇 Add Environment Variable (新增環境變數)。針對 Key (索引鍵)，輸入您的環境變數的名稱。針對 Value (值)，為您的環境值輸入一個字串值 (未使用雙引號 (" ") 括住)。

      AWS 以雙引號 (" ") 括住字串，並以下列格式將字串傳遞至容器：

      ```
      MY_ENV_VAR="This variable contains a string."
      ```

12. (選用) 為協助識別您的任務，請展開 Tags (標籤) 區段，然後設定標籤。

    若要讓 Amazon ECS 使用叢集名稱和任務定義標籤，自動標記所有新啟動的任務，請選取開啟 Amazon ECS 受管標籤，然後選取任務定義。

新增或移除標籤。

- [新增標籤] 選擇新增標籤，然後執行下列操作：
  - 在索引鍵中，輸入索引鍵名稱。
  - 在值中，進入索引鍵值。
- [移除標籤] 在標籤旁邊，選擇移除標籤。

13. 選擇 Create (建立)。

使用 Amazon EventBridge 排程器來排程 Amazon ECS 任務

EventBridge 排程器是無伺服器排程器，可讓您從單一受管的中央服務建立、執行及管理任務。其提供與事件匯流排和規則無關的一次性和週期性排程功能。EventBridge 排程器具有高度可自訂性，並透過 EventBridge 排程規則改善可擴展性，提供更廣泛的目標 API 操作和 AWS 服務。EventBridge 排程器提供下列排程，您可以在 EventBridge 排程器主控台中為您的任務設定這些排程：

- 速率型
- Cron 型

  您可以在任何時區中設定 Cron 型排程。
- 一次性排程

  您可以在任何時區設定一次性排程。

您可以使用 Amazon EventBridge 排程器來排程 Amazon ECS。

雖然您可以在 Amazon ECS 主控台中建立排程任務，但 EventBridge 排程器主控台目前提供更多功能。

在排程任務之前，請完成以下步驟：

1. 使用 VPC 主控台取得執行任務所在的子網路 ID 以及子網路的安全群組 ID。如需詳細資訊，請參閱《Amazon VPC 使用者指南》中的子網路，以及AWS 使用安全群組控制資源的流量。
2. 設定 EventBridge 排程器執行角色。如需詳細資訊，請參閱《Amazon EventBridge 排程器使用者指南》中的設定執行角色。

使用主控台建立新排程

1. 前往 https://console.aws.amazon.com/scheduler/home 開啟 Amazon EventBridge 排程器。
2. 在排程頁面上，選擇建立排程。
3. 在指定排程詳細資訊頁面的排程名稱和描述區段中，執行以下動作：

   a. 在排程名稱中，輸入排程的名稱，例如：**MyTestSchedule**。

   b. (選用) 在描述中，輸入對排程的描述，例如：**TestSchedule**。

   c. 針對排程群組，選擇排程群組。如果您沒有群組，請選擇預設值。若要建立排程群組，請選擇建立自己的排程。

      您可以使用排程群組，為不同群組的排程加上標籤。

4. 選擇排程選項。

頻率	執行此作業...
一次性排程 一次性排程只會在您指定的日期與時間調用目標一次。	針對日期和時間執行以下動作： • 依 YYYY/MM/DD 格式輸入有效日期。 • 依 hh:mm 格式輸入時間戳記 (24 小時)。 • 針對時區選擇時區。
週期性排程 週期性排程會依您指定的頻率，使用 cron 或 Rate 運算式調用目標。	a. 在排程模式中，執行下列其中一項動作： • 若要使用 Cron 運算式定義排程，請選擇 Cron 排程，然後輸入 Cron 運算式。 • 若要使用 Rate 表達式定義排程，請選擇 Rate 排程，然後輸入 Rate 表達式。

頻率	執行此作業...	
	如需 Cron 和 Rate 運算式的詳細資訊，請參閱《Amazon EventBridge 排程器使用者指南》中的 EventBridge 排程器上的排程類型。 b. 對於彈性時段，選擇關閉可關閉此選項，或者也能選擇其中一個預先定義的時間範圍。例如，如果您選擇 15 分鐘並設定週期性排程，每小時調用目標一次，則排程會在每小時一開始的 15 分鐘內執行。	

5. (選用) 如果您在上一個步驟中選擇週期性排程，請在時間範圍區段執行以下動作：
   a. 針對時區選擇時區。
   b. 對於開始日期和時間，依 YYYY/MM/DD 格式輸入有效日期，接著依 24 小時的 hh:mm 格式指定時間戳記。
   c. 對於結束日期和時間，依 YYYY/MM/DD 格式輸入有效日期，接著依 24 小時的 hh:mm 格式指定時間戳記。
6. 選擇 Next (下一步)。
7. 在選取目標頁面上，執行下列動作：
   a. 選擇所有 API，然後在搜尋方塊中輸入 ECS。
   b. 選取 Amazon ECS。
   c. 在搜尋方塊中，輸入 RunTask，然後選擇 RunTask。
   d. 在 ECS 叢集中，選擇叢集。
   e. 在 ECS 任務中，選擇要用於任務的任務定義。
   f. 若要使用啟動類型，請展開運算選項，然後選取啟動類型。然後，選擇啟動類型。

      當指定 Fargate 啟動類型時，在平台版本中，輸入要使用的平台版本。若沒有指定平台，則會使用 LATEST 平台版本。

g. 在子網路中，輸入要在其中執行任務的子網路 ID。

h. 在安全群組中，輸入子網路的安全群組 ID。

i. (選用) 若要使用預設以外的任務置放策略，請展開置放限制條件，然後輸入限制條件。

如需詳細資訊，請參閱Amazon ECS 如何在容器執行個體上放置任務。

j. (選用) 為協助識別您的任務，請在標籤下設定標籤。

若要讓 Amazon ECS 使用任務定義標籤，自動標記所有新啟動的任務，請選取啟用 Amazon ECS 受管標籤。

8. 選擇 Next (下一步)。

9. 在設定頁面執行以下動作：

a. 若要開啟排程，請在排程狀態底下切換到啟用排程。

b. 若要設定排程的重試政策，請在重試政策和無效字母佇列 (DLQ) 底下執行以下動作：

- 切換到重試。
- 在事件的最大保留時間中，輸入 EventBridge 排程器保留未處理事件的最大時數和分鐘數。
- 時間最長可設為 24 小時。
- 針對重試次數上限，輸入目標傳回錯誤時，EventBridge 排程器重新嘗試執行排程的次數上限。

  最大值為重試 185 次。

設定好重試政策後，如果排程無法調用其目標，EventBridge 排程器會重新執行排程。一旦設定此功能，您就必須設定排程的最長保留時間和重試次數。

c. 選擇 EventBridge 排程器儲存未交付事件的位置。

無效字母佇列 (DLQ) 選項	執行此作業...
不儲存	選擇無。
將事件存放在您建立排程的相同 AWS 帳戶 位置	a. 選擇在 中選取 Amazon SQS 佇列 AWS 帳戶 做為 DLQ。

無效字母佇列 (DLQ) 選項	執行此作業...	
	b. 選擇 Amazon SQS 佇列的 Amazon Resource Name (ARN)。	
將事件存放在與您要建立排程 AWS 帳戶 的位置不同的位置	a. 選擇在另一個 中指定 Amazon SQS 佇列 AWS 帳戶 做為 DLQ。 b. 輸入 Amazon SQS 佇列的 Amazon Resource Name (ARN)。	

d. 若要使用由客戶管理的金鑰加密您的目標輸入，請在加密底下選擇自訂加密設定 (進階)。

   如果選擇此選項，請輸入現有的 KMS 金鑰 ARN，或選擇建立 AWS KMS key，以導覽至 AWS KMS 控制台。如需 EventBridge 排程器如何加密靜態資料的詳細資訊，請參閱《Amazon EventBridge 排程器使用者指南》中的靜態加密。

e. 對於許可，請選擇使用現有角色，然後選取角色。

   若要讓 EventBridge 排程器為您建立新的執行角色，請選擇為此排程建立新角色。接著輸入角色名稱。如果您選擇此選項，EventBridge 排程器會將範本化目標所需的必要許可與角色連接。

10. 選擇 Next (下一步)。
11. 在檢閱和建立排程頁面上，檢閱排程的詳細資訊。在每個區段中選擇編輯，即可返回該步驟並編輯其詳細資訊。
12. 選擇建立排程。

    您可以在排程頁面檢視新建立和現有的排程。在狀態欄底下，確認您的新排程狀態為已啟用。

後續步驟

您可以使用 EventBridge 排程器主控台或 AWS CLI 來管理排程。如需詳細資訊，請參閱《Amazon EventBridge 排程器使用者指南》中的管理排程。

停止 Amazon ECS 任務

如果您不再需要讓獨立任務繼續執行，您可以停止任務。Amazon ECS 主控台可讓您輕鬆停止一或多個任務。

如果您想要停止服務，請參閱 使用主控台刪除 Amazon ECS 服務。

停止獨立任務 (AWS Management Console)

1. 開啟主控台，網址為 https://console.aws.amazon.com/ecs/v2。
2. 在導覽窗格中，選擇叢集。
3. 在叢集頁面上，選擇要導覽至叢集詳細資訊頁面的叢集。
4. 在叢集詳細資訊頁面上，選擇任務索引標籤。
5. 您可以使用篩選條件啟動類型清單，依啟動類型來篩選任務。

要停止的任務	步驟
一或多個	a. 選取任務，然後選擇停止、停止選取。 b. 在停止任務確認頁面上，選擇停止
全部	**Important** 如果您選擇使用主控台停止所有任務，Amazon ECS 會停止所有獨立任務和屬於服務一部分的任務。因此，建議您在使用此選項時小心。 a. 選擇停止 > 全部停止。

要停止的任務	步驟	
	b. 在停止任務確認頁面上，輸入停止所有任務，然後選擇停止。	

Amazon ECS 服務

您可以利用 Amazon ECS 服務在 Amazon ECS 叢集中同時執行並維持指定數目的任務定義執行個體。如果您有任務產生故障或停止，Amazon ECS 服務排程器就會根據您的任務定義啟動另一個執行個體來取而代之。這有助於維持服務中所需的任務數量。

您也可以在負載平衡器後方選擇性地執行您的服務。負載平衡器可將流量分散到與服務關聯的任務。

我們建議將服務排程器用於長時間執行的無狀態服務及應用程式。服務排程器可確保系統遵循您指定的排程策略，並在任務失敗時重新排程任務。例如，若基本的基礎設施發生故障，服務排程器會重新排程任務。您可以利用任務置放策略和限制條件來自訂排程器如何放置和終止任務。若服務中的任務停止，排程器會啟動新任務進行取代。此流程會根據您服務使用的排程策略持續進行，直至服務執行的任務達到您所需的數量為止。服務的排程策略也叫做 service type (服務類型)。

在容器運作狀態檢查或負載平衡器目標群組運作狀態檢查失敗後，服務排程器也會取代判定為狀況不佳的任務。此取代取決於 `maximumPercent` 和 `desiredCount` 服務定義參數。如果任務標記為運作狀態不佳，服務排程器會先啟動替代任務。然後，會發生以下情況。

- 如果取代任務的運作狀態為 `HEALTHY`，服務排程器會停止運作狀態不佳的任務
- 如果替代任務的運作狀態為 `UNHEALTHY`，排程器會停止運作狀態不佳的替代任務或現有運作狀態不佳的任務，以使任務總計數等於 `desiredCount`。

如果 `maximumPercent` 參數限制排程器先啟動替代任務，排程器會隨機停止運作狀態不佳的任務，以釋放容量，然後啟動替代任務。開始和停止程序會繼續進行，直到所有運作狀態不佳的任務都會取代為運作狀態良好的 一旦已取代了所有運作狀態不佳的任務，而且只執行運作狀態良好的任務，如果任務總數超過 `desiredCount`，則運作良好的任務會隨機停止，直到任務總計數等於 `desiredCount` 為止。如需有關 `maximumPercent` 和 `desiredCount` 的詳細資訊，請參閱服務定義參數。

服務排程器包含的邏輯，可在任務重複啟動失敗的情況下，針對任務重新啟動的頻率進行調節。如果任務在未進入 `RUNNING` 的狀態下停止，服務排程器會減少啟動嘗試並傳送出服務事件訊息。此行為可以

讓您在問題解決前，避免失敗的任務占用不必要的資源。在服務更新之後，服務排程器就會恢復正常的排程行為。如需詳細資訊，請參閱 Amazon ECS 服務調節邏輯 和 檢視 Amazon ECS 服務事件訊息。

現已推出兩個服務排程器策略概念：

- REPLICA - 複本排程策略會置放並在整個叢集中維持所需的任務數量。根據預設，服務排程器會將任務分散至各個可用區域。您可以使用任務置放策略和限制條件，來自訂任務置放決策。如需詳細資訊，請參閱複本策略。
- DAEMON - 常駐程式排程策略會在每個符合您於叢集中所指定所有任務置放限制條件的作用中容器執行個體上，準確地部署一個任務。使用這項策略時，不需指定所需的任務數量、任務置放策略或使用 Service Auto Scaling 政策。如需詳細資訊，請參閱協助程式策略。

> **Note**
>
> Fargate 任務不支援 DAEMON 排程策略。

協助程式策略

常駐程式排程策略會在每個活動容器執行個體上準確部署一個任務，其可滿足叢集中所有指定的任務置放限制條件。服務排程器會評估執行任務的任務置放限制，並停止不符合置放限制的任務。當您使用此策略時，您不需要指定所需的任務數量、任務置放策略，或使用 Service Auto Scaling 政策。

Amazon ECS 會為常駐程式任務保留容器執行個體運算資源，包括 CPU、記憶體和網路介面。當您在具有其他複本服務的叢集上啟動常駐程式服務時，Amazon ECS 會將常駐程式任務設為最優先。這表示協助程式任務是在執行個體上啟動的第一個任務，以及在停止所有複本任務之後停止的最後一個任務。此策略可確保資源不會被擱置的複本任務使用，且可供常駐程式任務使用。

常駐程式服務排程器不會將任何任務置放於具有 DRAINING 狀態的執行個體上。如果容器執行個體轉換為 DRAINING 狀態，常駐程式會停止運作。新的容器執行個體加入叢集時，服務排程器也會加以監控，並將常駐程式任務新增到這些執行個體。

當您指定部署組態時， maximumPercent 參數的值必須是 100（以百分比表示），如果未設定，則此值是預設值。minimumHealthyPercent 參數的預設值為 0（以百分比表示）。

當您變更常駐程式服務的置放限制條件時，您必須重新啟動服務。Amazon ECS 會為常駐程式任務動態更新合格執行個體上保留的資源。對於現有執行個體，排程器會嘗試將任務放置在執行個體上。

當任務定義中的任務大小或容器資源保留改變時，會啟動一個新的部署。更新服務或設定任務定義的不同修訂時，也會開始新的部署。Amazon ECS 會為常駐程式挑選更新的 CPU 和記憶體保留，然後為常駐程式任務封鎖容量。

若上述任一情境的資源不足，將發生下列情況：

- 任務置放失敗。
- 會產生 CloudWatch 事件。
- Amazon ECS 透過等待資源轉為可用以繼續嘗試與排程執行個體上的任務。
- Amazon ECS 會釋放任何不再符合置放限制標準的預留執行個體，並且停止對應的常駐程式任務。

常駐程式排程策略可被用於下列情況：

- 執行應用程式容器
- 執行支援容器以記錄、監控和追蹤任務

使用 Fargate 啟動類型或 CODE_DEPLOY 或 EXTERNAL 部署控制器類型的任務，不支援常駐程式排程策略。

當服務排程器停止執行中的任務時，會嘗試維護叢集中之可用區域間的平衡。排程器使用以下邏輯：

- 如已定義置放策略，請使用該策略選取要終止的任務。例如，如果服務已定義可用區域分佈策略，則會選取能讓剩餘任務完美分佈的任務。
- 如未定義任何置放策略，請使用下列邏輯維護叢集中之可用區域的平衡：
  - 排序有效的容器執行個體。優先考慮在此服務的個別可用區域中，擁有最多執行中任務數量的執行個體。例如，如果區域 A 有一項執行中的服務任務，而區域 B 和 C 各有兩項執行中的服務任務，則最適合終止的即為區域 B 或 C 中的容器執行個體。
  - 根據先前的步驟，停止最佳可用區域中容器執行個體上的任務。偏好此服務執行中任務數量最多的容器執行個體。

複本策略

複本排程策略會在叢集中放置並維持所需的任務數量。

對於在 Fargate 上執行任務的服務，當服務排程器啟動新任務或停止執行任務時，服務排程器會盡力嘗試在可用區域之間維持平衡。您無須指定任務置放策略或限制條件。

當您在 EC2 執行個體上建立執行任務的服務時，您可以選擇性地指定任務置放策略和限制條件，以自訂任務置放決策。如果未指定任務置放策略或限制，預設情況下，服務排程器會將任務分散到可用區域。服務排程器使用以下邏輯：

- 決定您叢集中的哪些容器執行個體可支援服務的任務定義 (例如，必要的 CPU、記憶體、連接埠和容器執行個體屬性)。
- 決定哪些容器執行個體可滿足針對服務所定義的任何置放限制。
- 當您有取決於常駐程式服務的複本服務時 (例如，必須先執行常駐程式日誌路由器任務，才能使用記錄)，請建立任務置放限制，以確保常駐程式服務任務在複本服務任務之前置放在 EC2 執行個體上。如需詳細資訊，請參閱Amazon ECS 任務置放限制範例。
- 如已定義置放策略，請使用該策略從剩餘的待選項目中選取執行個體。
- 如未定義任何置放策略，請使用下列邏輯平衡您叢集中可用區域的任務：
  - 排序有效的容器執行個體。優先考慮在此服務的個別可用區域中，執行中任務數量最少的執行個體。例如，如果區域 A 有一項執行中的服務任務，而區域 B 和 C 都沒有，則最適合置放的即為區域 B 或 C 中的有效容器執行個體。
  - 根據先前的步驟，將新的服務任務置放在最佳可用區域的有效容器執行個體。偏好此服務執行中任務數量最少的容器執行個體。

我們建議您在使用 `REPLICA`策略時，使用服務修復功能，因為它有助於確保服務的高可用性。

在可用區域之間平衡 Amazon ECS 服務

為了協助您的應用程式實現高可用性，建議您將多任務服務設定為跨多個可用區域執行。對於將第一個置放策略指定為可用區域分散的服務， AWS 會盡最大努力將服務任務平均分散到可用區域。不過，有時在一個可用區域中執行的任務數量可能與其他可用區域中的任務不同，例如在可用區域中斷之後。若要解決此任務不平衡，您可以啟用可用區域重新平衡功能。透過可用區域重新平衡，Amazon ECS 會持續監控每個 服務在可用區域之間的任務分佈。當 Amazon ECS 偵測到不均勻的任務分佈時，會自動採取行動來重新平衡跨可用區域的工作負載。這包括在可用區域中啟動具有最少任務的新任務，以及在過載可用區域中終止任務。此重新分佈可確保單一可用區域不會成為故障點，有助於維持容器化應用程式的整體可用性。自動化重新平衡程序不需要手動介入，可加快事件發生後的復原時間。

以下是可用區域重新平衡程序的概觀：

1. Amazon ECS 會在服務達到穩定狀態後開始監控服務，並查看每個可用區域中執行的任務數量。
2. Amazon ECS 偵測到每個可用區域中執行的任務數量不平衡時，會執行下列操作：
   - 傳送服務事件，指出可用區域重新平衡正在啟動。

- 在可用區域中以最少數目的執行中任務啟動任務
- 以最大數量的執行中任務來停止可用區域中的任務。
- 排程器會等待新啟動的任務變成 HEALTHY，RUNNING然後才停止過度擴展可用區域中的任務。
- 傳送具有可用區域重新平衡結果的服務事件。

可用區域重新平衡支援 Fargate 和 EC2 啟動類型。對於 Fargate，Amazon ECS 會自動在可用可用區域重新分配任務，以維持平衡。對於 EC2 啟動類型，Amazon ECS 會盡最大努力重新平衡現有容器執行個體的任務，以符合您定義的置放策略和限制條件。不過，ECS 無法在未充分利用的可用區域中佈建新的執行個體，做為重新平衡程序的一部分，將重新平衡限制在現有的容器執行個體。

可用區域重新平衡適用於下列組態：

- 使用 Replica策略的服務
- 指定可用區域的服務會分散為第一個任務置放策略，或不指定置放策略。

您無法將可用區域重新平衡與符合下列任一條件的服務搭配使用：

- 使用 Daemon策略
- 使用EXTERNAL啟動類型 (ECS Anywhere)
- 使用 100% 的值 maximumPercent
- 使用 Classic Load Balancer
- 使用 attribute:ecs.availability-zone做為任務置放限制條件

使用可用區域重新平衡的置放策略和置放限制

置放策略會決定 Amazon ECS 如何選取容器執行個體和可用區域來終止任務置放。任務置放限制是判斷任務是否允許在特定容器執行個體上執行的規則。對於 EC2 啟動類型，您可以使用置放策略和置放限制，搭配可用區域重新平衡。不過，若要讓可用區域重新平衡正常運作，可用區域分散置放策略必須是第一個指定的策略。可用區域重新平衡與各種置放策略組合相容。例如，您可以建立策略，先將任務平均分散到可用區域，然後 bin 會根據每個可用區域內的記憶體來封裝任務。在此情況下，可用區域重新平衡會運作，因為會先指定可用區域分散策略。請務必注意，如果置放策略陣列中的第一個策略不是可用區域分散元件，則可用區域重新平衡將無法運作。此要求可確保任務分佈的主要重點是維持可用區域的平衡，這對高可用性至關重要。如需任務置放策略和限制條件的詳細資訊，請參閱Amazon ECS 如何在容器執行個體上放置任務。

下列範例策略會將任務平均分散到可用區域，然後 bin 會根據每個可用區域內的記憶體來封裝任務。可用區域重新平衡與 服務相容，因為 spread策略是第一個。

```
"placementStrategy": [
    {
        "field": "attribute:ecs.availability-zone",
        "type": "spread"
    },
    {
        "field": "memory",
        "type": "binpack"
    }
]
```

開啟可用區域重新平衡

您需要為新的和現有的服務啟用可用區域重新平衡。

您可以使用主控台、APIs或 來啟用和停用可用區域重新平衡 AWS CLI。

服務類型	API	主控台	CLI
現有	UpdateService	使用主控台更新 Amazon ECS 服務	update-service
新增	CreateService	使用 主控台建立 Amazon ECS 服務	create-service

追蹤 Amazon ECS 可用區域重新平衡

您可以在 主控台中或呼叫 來驗證服務是否已啟用可用區域重新平衡describe-services。下列範例可用來查看 CLI 的狀態。

回應將為 ENABLED或 DISABLED。

```
aws ecs describe-services \
    --services service-name \
    --cluster cluster-name \
    --query services[0].availabilityZoneRebalancing
```

服務事件

Amazon ECS 會傳送服務動作事件，以協助您了解可用區域重新平衡生命週期。

事件	案例	Type	進一步了解
SERVICE_REBALANCING_STARTED	Amazon ECS 會啟動可用區域重新平衡操作	INFO	service *(service-name)* 與#### 1# #####、#### 2 中的####和#### 3 中的####不平衡。AZ 重新平衡進行中。
SERVICE_REBALANCING_COMPLETED	可用區域重新平衡操作完成	INFO	service (*service-name)* 在#### 1 中與####任務、#### 2 中的####任務，以及####3 ######任務之間取得 AZ 平衡。
TASKS_STARTED	Amazon ECS 會在可用區域重新平衡操作中成功啟動任務	INFO	*service-name* 已在#####開始將 *number-tasks* 任務重新平衡至 AZ：*task-ids*。
TASKS_STOPPED	Amazon ECS 在可用區域重新平衡操作中成功停止任務	INFO	*service-name* 已停止在#####執行任務的 *number-tasks* ，因為 AZ 重新平衡：*task-id*。
SERVICE_TASK_PLACEMENT_FAILURE	Amazon ECS 無法在可用區域重新平衡操作中啟動任務	ERROR	如需 EC2 啟動類型，請參閱 service *(service-name)* 無法將任務放置在#### #，因為沒有容器執行

事件	案例	Type	進一步了解
			個體符合其所有需求。 如需 Fargate 啟動類型，請參閱 service (*service-name*) 無法在#####放置任務。
TASKSET_SCALE_IN_FAILURE_BY_TASK_PROTECTION	由於任務保護正在使用中，因此會封鎖可用區域重新平衡操作。	INFO	service (*service-name*) 無法重新平衡 AZ，因為 *task-set-name* 因#無法縮減規模。
SERVICE_REBALANCING_STOPPED	可用區域重新平衡操作已停止。Amazon ECS 會傳送提供詳細資訊的其他事件。	INFO	service (*service-name*) 已停止 AZ 重新平衡。

任務狀態變更事件

Amazon ECS 會針對在重新平衡程序中啟動的每個任務傳送任務狀態變更事件 (START)。

Amazon ECS 會針對在重新平衡程序中停止的每個任務傳送任務狀態變更事件 (STOPPED) 事件。原因設定為 `Availability Zone rebalancing initiated by (deployment ecs-svc/`*`deployment-id`*`)`。

如需事件的詳細資訊，請參閱 Amazon ECS 任務狀態變更事件。

使用 主控台建立 Amazon ECS 服務

建立 服務，在叢集中同時執行和維護指定數量的任務定義執行個體。如果您有任務產生故障或停止，Amazon ECS 服務排程器就會根據您的任務定義啟動另一個執行個體來取而代之。這有助於維持服務中所需的任務數量。

建立服務之前，請先決定下列組態參數：

- 分發任務有兩種運算選項。
  - capacity provider strategy (容量供應商策略)，使 Amazon ECS 將您的任務分發至一個或多個容量供應商。
  - 啟動類型會導致 Amazon ECS 直接在 Fargate 或已註冊到叢集的 EC2 執行個體上啟動我們的任務。
- 使用 `awsvpc` 網路模式的任務定義或被設定使用負載平衡器的服務必須有聯網組態。根據預設，主控台會在預設的 Amazon VPC 中選擇預設的 Amazon VPC，以及全部子網路和預設安全群組。
- 置放策略，預設任務置放策略會將任務平均分散到可用區域。

  建議您使用可用區域重新平衡，以協助確保服務的高可用性。如需詳細資訊，請參閱在可用區域之間平衡 Amazon ECS 服務。
- 當您使用 Launch Type (啟動類型) 進行服務部署時，服務預設會在叢集 VPC 的子網路中啟動。
- 針對 capacity provider strategy (容量提供者策略)，主控台預設會選取運算選項。以下說明主控台用來選擇預設值的順序：
  - 若您的叢集定義了預設容量提供者策略，則會選取該叢集。
  - 如果您的叢集未定義預設容量提供者策略，但您已將 Fargate 容量提供者新增至叢集，則會選取使用容量提供者的自訂`FARGATE`容量提供者策略。
  - 如果您的叢集未定義預設容量提供者策略，但您有一或多個 Auto Scaling 群組容量提供者新增至叢集，則會選取使用自訂 （進階） 選項，而且您需要手動定義策略。
  - 若您的叢集未定義預設容量提供者策略，也沒有為叢集新增容量提供者，則會選擇 Fargate 啟動類型。
- 預設部署失敗偵測預設選項是使用 Amazon ECS 部署斷路器選項搭配回復失敗選項。

  如需詳細資訊，請參閱 Amazon ECS 部署斷路器如何偵測故障。
- 如果您想要使用藍/綠部署選項，請確定 CodeDeploy 如何移動應用程式。以下是可用的選項：
  - CodeDeployDefault.ECSAllAtOnce：將所有流量一次轉移至更新的 Amazon ECS 容器
  - CodeDeployDefault.ECSLinear10PercentEvery1Minutes：每分鐘轉移 10% 的流量，直到轉移所有流量為止。
  - CodeDeployDefault.ECSLinear10PercentEvery3Minutes：每 3 分鐘轉移 10% 的流量，直到轉移所有流量為止。
  - CodeDeployDefault.ECSCanary10Percent5Minutes：在第一個增量中轉移 10% 的流量。剩餘的 90% 會在五分鐘之後部署。
  - CodeDeployDefault.ECSCanary10Percent15Minutes：在第一個增量中轉移 10% 的流量。剩餘的 90% 會在 15 分鐘之後部署。

- 決定您是否希望 Amazon ECS 自動增加或減少服務中所需的任務數量。如需詳細資訊，請參閱 自動擴展 Amazon ECS 服務。
- 如果您需要應用程式連線到在 Amazon ECS 中執行的其他應用程式，請確定適合您架構的選項。如需詳細資訊，請參閱互連 Amazon ECS 服務。
- 當您建立使用 Amazon ECS 斷路器的服務時，Amazon ECS 會建立服務部署和服務修訂。這些資源可讓您檢視服務歷史記錄的詳細資訊。如需詳細資訊，請參閱使用 Amazon ECS 服務部署檢視服務歷史記錄。

  如需有關如何使用 建立服務的資訊 AWS CLI，請參閱 AWS Command Line Interface 參考create-service中的 。

  如需有關如何使用 建立服務的資訊 AWS CloudFormation，請參閱AWS CloudFormation 《 使用者指南AWS::ECS::Service》中的 。

使用預設選項建立服務

您可以使用主控台來快速建立並部署服務。服務具有下列組態：

- 在與叢集關聯的 VPC 和子網路中部署
- 部署一項任務
- 使用滾動部署
- 搭配預設容量供應商使用容量供應商策略
- 使用部署斷路器偵測故障，並將選項設定為在失敗時自動回復部署

若要使用預設參數部署服務，請遵循下列步驟。

建立服務 (Amazon ECS 主控台)

1. 開啟主控台，網址為 https://console.aws.amazon.com/ecs/v2。
2. 在導覽頁面中，選擇叢集)。
3. 在叢集頁面上，選擇要在其中建立服務的叢集。
4. 在 Services (服務) 索引標籤上，選擇 Create (建立)。
5. 在 Deployment configuration (部署組態)，指定應用程式的部署方式。
   a. 針對 Application type (應用程式類型)，選擇 Service (服務)。
   b. 針對 Task definition (任務定義)，選擇要使用的任務定義系列和修訂。

c. 針對 Service name (服務名稱)，輸入服務的名稱。

d. 針對 Desired tasks (所需任務)，輸入要在服務中啟動並維護的任務數。

6. (選用) 為協助識別您的服務和任務，請展開 Tags (標籤) 區段，然後設定標籤。

   若要讓 Amazon ECS 使用叢集名稱和任務定義標籤，自動標記所有新啟動的任務，請選取 Turn on Amazon ECS managed tags (開啟 Amazon ECS 受管標籤)，然後選取 Task definitions (任務定義)。

   若要讓 Amazon ECS 使用叢集名稱和服務標籤，自動標記所有新啟動的任務，請選取 Turn on Amazon ECS managed tags (開啟 Amazon ECS 受管標籤)，然後選取 Service (服務)。

   新增或移除標籤。

   - [新增標籤] 選擇新增標籤，然後執行下列操作：
     - 在索引鍵中，輸入索引鍵名稱。
     - 在值中，進入索引鍵值。
   - [移除標籤] 在標籤旁邊，選擇 移除標籤。

使用定義的參數建立服務

若要使用定義的參數建立 服務，請遵循下列步驟。

建立服務 (Amazon ECS 主控台)

1. 開啟主控台，網址為 https://console.aws.amazon.com/ecs/v2。
2. 決定您要從中啟動服務的資源。

啟動服務的資源	步驟
叢集	a. 在叢集頁面，選取您要在哪個叢集中建立服務。 b. 在 Services (服務) 索引標籤上，選擇 Create (建立)。
啟動類型	a. 在任務定義頁面上，選取任務定義旁的選項按鈕。

啟動服務的資源	步驟	
	b. 在部署功能表上，選擇建立服務。	

3. (選用) 選擇任務在叢集基礎設施中的分佈方式。展開 Compute configuration (運算組態)，然後選擇您的選項。

分發方法	步驟	
容量提供者策略	a. 在 Compute options (運算選項) 下，選擇 Capacity provider strategy (容量提供者策略)。 b. 選擇策略： • 若要使用叢集的預設容量提供者策略，選擇使用叢集預設。 • 若您的叢集沒有預設容量提供者策略，或要使用自訂策略，選擇 Use custom (使用自訂)、Add capacity provider strategy (新增容量提供者策略)，然後透過指定 Base (基礎)、Capacity provider (容量提供者) 和 Weight (權重) 來定義您的自訂容量提供者策略。 **Note** 若要在策略中使用容量提供者，容量提供	

分發方法	步驟
	者必須與叢集相關聯。
啟動類型	a. 在運算選項區段中，選取啟動類型。 b. 針對 啟動類型，選擇啟動類型。 c. (選用) 當指定 Fargate 啟動類型時，針對平台版本，指定要使用的平台版本。如果未指定平台版本，將使用 LATEST 平台版本。

4. 若要指定 服務的部署方式，請前往部署組態區段，然後選擇您的選項。

   a. 在應用程式類型中，沿用服務選項。

   b. 針對 Task definition (任務定義) 和 Revision (修訂)，選擇要使用的任務定義系列和修訂。

   c. 針對 Service name (服務名稱)，輸入服務的名稱。

   d. 針對 Service type (服務類型)，選擇服務排程策略。

      - 若要讓排程器在每個活動容器執行個體上準確部署一個任務，且滿足所有任務置放限制條件，請選擇 Daemon (常駐程式)。
      - 若要讓排程器在叢集中置放並維持所需的任務數量，請選擇 Replica (複寫)。

   e. 如果您選擇 Replica (複寫)，針對 Desired tasks (所需任務)，輸入要在服務中啟動並維護的任務數。

   f. 如果您選擇複本，若要讓 Amazon ECS 監控任務在可用區域之間的分佈，並在發生不平衡時重新分佈，請在可用區域服務重新平衡下，選取可用區域服務重新平衡。

   g. 決定服務的部署類型。展開部署選項，然後指定下列參數。

部署類型	步驟
滾動更新	a. 針對 Min running tasks (執行中任務下限)，輸

部署類型	步驟	
	入部署期間必須維持在 RUNNING 狀態的服務任務數量下限，它是所需任務數量的百分比 (無條件進位到最接近的整數)。如需詳細資訊，請參閱部署組態。 b. 針對 Max running tasks (執行中任務上限)，輸入部署期間允許的處於 RUNNING 或 PENDING 狀態的服務任務數目上限，它是所需任務數量的百分比 (無條件捨去到最接近的整數)。	
藍/綠部署	a. 針對 Deployment configuration (部署組態)，選擇 CodeDeploy 在部署期間將生產流量路由到您替代任務集的方式。 b. 針對 CodeDeploy 的服務角色，選擇服務用來向授權的 提出 API 請求的 IAM 角色 AWS 服務。	

h. 若要設定 Amazon ECS 如何偵測並處理部署失敗，請展開 Deployment failure detection (部署失敗偵測)，然後選擇您的選項。

i. 若要在任務無法啟動時停止部署，請選取 Use the Amazon ECS deployment circuit breaker (使用 Amazon ECS 部署斷路器)。

若要讓軟體在部署斷路器將部署設定為失敗狀態時，自動將部署復原至最後完成的部署狀態，請選取在失敗時復原。

ii. 若要根據應用程式指標停止部署，請選取使用 CloudWatch 警示 (s)。然後，從 CloudWatch 警示名稱中選擇警示。若要建立新的警示，請前往 CloudWatch 主控台。

若要讓軟體在 CloudWatch 警示將部署設定為失敗狀態時，自動將部署復原至最後完成的部署狀態，請選取在失敗時復原。

5. (選用) 若要使用 Service Connect，請選取 Turn on Service Connect (開啟 Service Connect)，然後指定下列項目：

a. 在 Service Connect configuration (Service Connect 組態) 下，指定用戶端模式。

- 如果您的服務執行的網路用戶端應用程式只需要連線到命名空間中的其他 服務，請選擇僅限用戶端。
- 如果服務執行的是網路或 Web 服務應用程式，且需要為此服務提供端點，並連線至命名空間中的其他服務，請選擇 Client and server (用戶端和伺服器)。

b. 若要使用非預設叢集命名空間的命名空間，請在 Namespace (命名空間) 欄位中選擇服務命名空間。

c. (選用) 選擇使用日誌收集選項來指定日誌組態。每個可用的日誌驅動程序式都有要指定的日誌驅動程序選項。預設選項會將容器日誌傳送至 CloudWatch 日誌。其他日誌驅動程式選項會使用 AWS FireLens 來設定。如需詳細資訊，請參閱將 Amazon ECS 日誌傳送至 AWS 服務或 AWS Partner。

下方更詳細地描述了每個容器日誌目的地。

- Amazon CloudWatch – 設定任務，將容器日誌傳送至 CloudWatch Logs。提供預設日誌驅動程式選項，可代表您建立 CloudWatch 日誌群組。若要指定不同的日誌群組名稱，請變更驅動程式選項值。
- Amazon Data Firehose – 設定任務以將容器日誌傳送至 Firehose。提供預設日誌驅動程式選項，可將日誌傳送至 Firehose 交付串流。若要指定不同的交付串流名稱，請變更驅動程式選項值。
- Amazon Kinesis Data Streams – 設定任務將容器日誌傳送至 Kinesis Data Streams。提供預設日誌驅動程式選項，可將日誌傳送至 Kinesis Data Streams 串流。若要指定不同的串流名稱，請變更驅動程式選項值。
- Amazon OpenSearch Service – 設定任務，將容器日誌傳送至 OpenSearch Service 網域。務必提供日誌驅動程式選項。
- Amazon S3 – 設定任務將容器日誌傳送至 Amazon S3 儲存貯體。提供預設日誌驅動程式選項，但您必須指定有效的 Amazon S3 儲存貯體名稱。

6. （選用）若要使用服務探索，請選取使用服務探索，然後指定下列項目。

   a. 若要使用新的命名空間，請選擇在設定命名空間下建立新的命名空間，然後提供命名空間名稱和描述。若要使用現有的命名空間，請選擇選取現有的命名空間，然後選擇您要使用的命名空間。

   b. 提供 Service Discovery 服務資訊，例如服務的名稱和描述。

   c. 若要讓 Amazon ECS 執行定期容器層級運作狀態檢查，請選取啟用 Amazon ECS 任務運作狀態傳播。

   d. 針對 DNS record type (DNS 紀錄類型)，選擇您要為服務建立的 DNS 紀錄類型。Amazon ECS 服務探索僅支援 A 和 SRV 記錄，取決於任務定義指定的網路模式。如需有關此類記錄類型的詳細資訊，請參閱 Amazon Route 53 開發人員指南中的支援的 DNS 紀錄類型。

      - 如果服務任務指定的任務定義使用的是 `bridge` 或 `host` 網路模式，僅支援類型 SRV 記錄。選擇容器名稱和連接埠組合以與該紀錄建立關聯。
      - 如果服務任務指定的任務定義使用的是 `awsvpc` 網路模式，選取 A 或 SRV 紀錄類型。如果您選擇 A，請跳到下一個步驟。如果選取 SRV，指定可找到該服務的連接埠，或是容器名稱和連接埠組合，以與該記錄建立關聯。

      針對 TTL，輸入 DNS 解析器和網頁瀏覽器快取記錄集的時間 (以秒為單位)。

7. (選用) 若要設定服務的負載平衡器，請展開 Load balancing (負載平衡) 區段。

   選擇負載平衡器。

使用此負載平衡器	執行此作業	
Application Load Balancer	a. 針對 Load balancer type (負載平衡器類型)，選擇 Application Load Balancer。 b. 選擇 Create a new load balancer (建立新的負載平衡器) 以建立新的 Application Load Balancer，或者 Use an existing load balancer (使用現有負載平衡器) 以選擇現	

使用此負載平衡器	執行此作業	
	有的 Application Load Balancer。 c. 針對 Load balancer name (負載平衡器名稱)，輸入不重複的名稱。 d. 針對 Choose container to load balance (選擇進行負載平衡的容器)，選擇託管服務的容器。 e. 針對 Listener (接聽程式)，為 Application Load Balancer 輸入連接埠和協定，以偵聽連接請求。根據預設，負載平衡器將被設定為使用連接埠 80 和 HTTP。 f. 針對 Target group name (目標群組名稱)，為目標群組輸入名稱和協定，Application Load Balancer 會將請求路由至該群組。根據預設，目標群組會將請求路由至您的任務定義中定義的第一個容器。 g. 針對取消註冊延遲，輸入負載平衡器將目標狀態變更為 的秒數UNUSED。預設為 300 秒。 h. 針對 Health check path (運作狀態檢查路徑)，輸入容器中的現有路徑，Application Load Balancer 會在該容器中定期傳送請求，以驗證 Application Load	

使用此負載平衡器	執行此作業	
	Balancer 和容器之間的連接狀態。預設使用根目錄 (/)。 i. 針對 Health check grace period (運作狀態檢查寬限期)，輸入服務排程器應忽略狀態不良的 Elastic Load Balancing 目標運作狀態檢查的時長 (秒)。	

使用此負載平衡器	執行此作業	
Network Load Balancer	a. 針對 Load balancer type (負載平衡器類型)，選取 Network Load Balancer。 b. 針對 Load Balancer (負載平衡器)，選擇現有的 Network Load Balancer。 c. 針對 Choose container to load balance (選擇進行負載平衡的容器)，選擇託管服務的容器。 d. 針對 Target group name (目標群組名稱)，為目標群組輸入名稱和協定，Network Load Balancer 會將請求路由至該群組。根據預設，目標群組會將請求路由至您的任務定義中定義的第一個容器。 e. 針對取消註冊延遲，輸入負載平衡器將目標狀態變更為 的秒數UNUSED。預設為 300 秒。 f. 針對 Health check path (運作狀態檢查路徑)，輸入容器中的現有路徑，Network Load Balancer 會在該容器中定期傳送請求，以驗證 Application Load Balancer 和容器之間的連接狀態。預設使用根目錄 (/)。 g. 針對 Health check grace period (運作狀態檢查寬限	

使用此負載平衡器	執行此作業	
	期)，輸入服務排程器應忽略狀態不良的 Elastic Load Balancing 目標運作狀態檢查的時長 (秒)。	

8. （選用）若要使用 VPC Lattice，請選取開啟 VPC Lattice，然後指定下列項目：

   a. 針對基礎設施角色，選擇基礎設施角色。

      如果您尚未建立角色，請選擇建立基礎設施角色。

   b. 在目標群組下，選擇目標群組。您需要選擇至少一個目標群組，最多可以有五個目標群組。選擇新增目標群組以新增其他目標群組。為您選擇的每個目標群組選擇連接埠名稱、通訊協定和連接埠。

      若要刪除目標群組，請選擇移除。

      > **Note**
      >
      > - 如果您想要新增現有的目標群組，則需要使用 AWS CLI。如需如何使用 新增目標群組的指示 AWS CLI，請參閱《 AWS Command Line Interface 參考》中的註冊目標。
      > - 雖然 VPC Lattice 服務可以有多個目標群組，但每個目標群組只能新增至一個服務。

   c. 若要完成 VPC Lattice 組態，請在接聽程式預設動作中包含您的新目標群組，或在 VPC Lattice 主控台中包含現有 VPC Lattice 服務的規則。如需詳細資訊，請參閱 VPC Lattice 服務的接聽程式規則。

9. （選用）若要設定服務 Auto Scaling，請展開服務自動擴展，然後指定下列參數。

   a. 若要使用服務自動擴展，請選取 Service auto scaling (服務自動擴展)。

   b. 針對任務數量下限，輸入服務自動擴展要使用的任務數量下限。所需的計數不會低於此計數。

   c. 針對任務數量上限，輸入服務自動擴展要使用的任務數量上限。所需的計數不會高於此計數。

   d. 選擇政策類型。在擴展政策類型下，選擇下列其中一個選項。

使用此政策類型	執行此作業	
目標追蹤	a. 針對 Scaling policy type (擴展政策類型)，選擇 Target tracking (目標追蹤)。 b. 針對 Policy name (政策名稱)，輸入政策的名稱。 c. 針對 ECS service metric (ECS 服務指標)，選擇下列其中一項指標。 • ECSServiceAverageCPUUtilization – 服務的平均 CPU 使用率。 • ECSServiceAverageMemoryUtilization – 服務的平均記憶體使用率。 • ALBRequestCountPerTarget – Application Load Balancer 目標群組中每個目標完成的請求數量。 d. 針對 Target value (目標值)，輸入服務為選取的指標保持的值。 e. 針對橫向擴展冷卻時間，輸入橫向擴展活動（新增任務）之後，必須通過的時間量，然後才能開始另一個橫向擴展活動。 f. 針對縮減冷卻時間，輸入縮減活動（移除任務）之後，必須通過的時間	

使用此政策類型	執行此作業	
	量，然後才能開始另一個縮減活動。 g. 若要防止政策執行縮減活動，請選取 Turn off scale-in (關閉縮減)。 h. • （選用） 如果您希望擴展政策向外擴展以增加流量，但不需要在流量減少時向內擴展，請選取關閉向內擴展。	

使用此政策類型	執行此作業	
步驟擴展	a. 針對 Scaling policy type (擴展政策類型)，選擇 Step scaling (步驟擴展)。 b. 在 Policy Name (政策名稱) 輸入政策的名稱。 c. 針對 Alarm name (警示名稱)，輸入警示的唯一名稱。 d. 針對 Amazon ECS service metric (Amazon ECS 服務指標)，選擇用於警示的指標。 e. 針對 Statistic (統計資料)，選擇警示統計資料。 f. 針對 Period (期間)，選擇警示的期間。 g. 針對 Alarm condition (警示條件)，選擇如何比較選取的指標與定義的閾值。 h. 針對 Threshold to compare metrics (比較閾值與指標) 和 Evaluation period to initiate alarm (啟動警示的評估期)，輸入用於警示的閾值以及評估閾值的時間長度。 i. 在 Scaling actions (擴展動作) 下，執行下列動作：	

使用此政策類型	執行此作業	
	• 針對動作，選取是否要新增、移除或設定服務的特定所需計數。 • 如果您選擇新增或移除任務，請在值中輸入啟動擴展動作時要新增或移除的任務數量（或現有任務的百分比）。如果您選擇設定所需的計數，請輸入任務數量。對於 Type (類型)，選取 Value (值) 是整數或是現有所需計數的百分比值。 • 針對 Lower bound (下限) 和 Upper bound (上限)，輸入步驟擴展調整的下界限和上界限。根據預設，新增政策的下限為警示閾值，而上限為無限大正數 (+) 值。根據預設，移除政策的上限為警示閾值，而下限為無限小負數 (-) 值。 • (選用) 新增其他擴展選項。選擇新增擴展動作，然後重複擴展動作步驟。 • 針對冷卻時間，以秒為單位輸入等待先前擴展活動生效的時間量。對於新增政策，這是擴展政策封鎖縮減活動，並	

使用此政策類型	執行此作業	
	限制一次可以縮減任務數量之後的時間。對於移除政策，這是在另一個縮減活動開始之前必須通過的縮減活動之後的時間。	

10. (選用) 若要使用預設以外的任務置放策略，請展開任務置放，然後從下列選項中選擇。

    如需詳細資訊，請參閱Amazon ECS 如何在容器執行個體上放置任務。

    - AZ 平衡分散 – 將任務分散到可用區域和可用區域中的容器執行個體。
    - AZ 平衡 BinPack – 將任務分散到可用區域和具有最少可用記憶體的容器執行個體。
    - BinPack – 根據最低可用 CPU 或記憶體數量來分發任務。
    - 每個主機一個任務 – 在每個容器執行個體上最多放置一個來自 服務的任務。
    - 自訂 – 定義您自己的任務置放策略。

    如果選擇自訂，則請定義置放任務的演算法，以及在任務置放期間考慮的規則。

    - 在策略下，針對類型和欄位，選擇演算法以及要用於演算法的實體。

      您最多可新增 5 項策略。
    - 在 Constraint (限制條件)，針對 Type (類型) 和 Expression (運算式)，選擇要用於限制條件的規則與屬性。

      例如，若要設定限制條件，以將任務置放在 T2 執行個體上，針對運算式，請輸入 attribute:ecs.instance-type =~ t2.*。

      您最多可新增 10 個限制條件。
11. 如果任務定義使用 awsvpc 網路模式，請展開 Networking (聯網)。使用下列步驟以指定自訂組態。

    a. 針對 VPC，選擇要使用的 VPC。

    b. 針對子網路，在 VPC 中選擇一個或多個子網路，而任務排程器在放置任務時會考慮該 VPC。

> **Important**
>
> awsvpc 網路模式只支援私有子網路。任務不會接收公有 IP 地址。因此必須使用 NAT 閘道進行對外網際網路存取，且入站網際網路流量會透過負載平衡器進行路由。

c. 針對 Security groups (安全群組)，您可以選取現有的安全群組，或建立新的安全群組。若要使用現有的安全群組，選擇該安全群組並移至下一個步驟。若要建立新的安全群組，請選擇建立新安全群組。您必須指定安全群組名稱、描述，然後為該安全群組新增一條或更多傳入規則。

d. 針對 Public IP (公有 IP)，選擇是否為任務的彈性網路界面 (ENI) 自動指派公有 IP 地址。

AWS Fargate 在公有子網路中執行時，任務可以指派公有 IP 地址，以便他們有網際網路的路由。EC2 任務無法使用此欄位指派公有 IP。如需詳細資訊，請參閱 Fargate 啟動類型的 Amazon ECS 任務聯網選項，以及為 Amazon ECS 任務配置網路介面。

12. 如果您的任務使用與部署時組態相容的資料磁碟區，您可以透過擴展磁碟區來設定磁碟區。

磁碟區名稱和磁碟區類型是在建立任務定義修訂時設定，在建立服務時無法變更。若要更新磁碟區名稱和類型，您必須建立新的任務定義修訂，並使用新的修訂來建立服務。

設定此磁碟區類型	執行此作業
Amazon EBS	a. 針對 EBS 磁碟區類型，選擇您要連接至任務的 EBS 磁碟區類型。 b. 針對大小 (GiB)，輸入以 gibibyte (GiB) 為單位的磁碟區大小有效值。您可以指定至少 1 GiB 和最大 16，384 GiB 磁碟區大小。除非您提供快照 ID，否則需要此值。 c. 針對 IOPS，輸入磁碟區應提供的輸入/輸出操作 (IOPS) 數目上限。此值僅適用於 io1io2、 和 gp3磁碟區類型。

設定此磁碟區類型	執行此作業	
	d. 針對輸送量 (MiB/s)，以每秒 MB 為單位 (MiBps 或 MiB/s)，輸入磁碟區應提供的輸送量。此值只能針對gp3磁碟區類型設定。 e. 對於快照 ID，如果您想要從快照建立磁碟區，請選擇現有的 Amazon EBS 磁碟區快照，或輸入快照的 ARN。您也可以不選擇或輸入快照 ID 來建立新的空白磁碟區。 f. 針對檔案系統類型，選擇用於磁碟區上資料儲存和擷取的檔案系統類型。您可以選擇作業系統預設值或特定檔案系統類型。Linux 的預設值為 XFS。對於從快照建立的磁碟區，您必須指定建立快照時磁碟區所使用的相同檔案系統類型。如果檔案系統類型不相符，任務將無法啟動。 g. 針對基礎設施角色，選擇具有必要許可的 IAM 角色，以允許 Amazon ECS 管理任務的 Amazon EBS 磁碟區。您可以將 `AmazonECSInfrastructureRolePolicyForVolumes` 受管政策連接至 角色，也可以使用政策做為建立	

設定此磁碟區類型	執行此作業	
	和連接您自己的政策的指南，其許可可滿足您的特定需求。如需有關必要許可的詳細資訊，請參閱 Amazon ECS 基礎設施 IAM 角色。 h. 對於加密，如果您想要依預設設定使用 Amazon EBS 加密，請選擇預設。如果您的帳戶已預設設定加密，則磁碟區將使用設定中指定的 AWS Key Management Service (AWS KMS) 金鑰加密。如果您選擇預設，且 Amazon EBS 預設加密未開啟，則磁碟區將取消加密。 如果您選擇自訂，您可以指定磁碟區加密 AWS KMS key 的 選項。 如果您選擇無，除非您已預設設定加密，或者您從加密的快照建立磁碟區，否則磁碟區將會取消加密。 i. 如果您已選擇自訂加密，則必須指定要使用 AWS KMS key 的 。針對 KMS 金鑰，選擇 AWS KMS key 或輸入金鑰 ARN。如果您選擇使用對稱客戶受管金鑰來加密磁碟區，請確	

設定此磁碟區類型	執行此作業	
	定您已在 AWS KMS key 政策中定義正確的許可。如需詳細資訊，請參閱 Amazon EBS 磁碟區的資料加密。 j. （選用）在標籤下，您可以透過從任務定義或服務傳播標籤，或提供您自己的標籤，將標籤新增至 Amazon EBS 磁碟區。 如果您想要從任務定義傳播標籤，請選擇傳播標籤的任務定義。如果您想要從服務傳播標籤，請選擇 Service for Propagate 標籤來源。如果您選擇不傳播，或者如果您不選擇值，則不會傳播標籤。 如果您想要提供自己的標籤，請選擇新增標籤，然後針對您新增的每個標籤提供金鑰和值。 如需標記 Amazon EBS 磁碟區的詳細資訊，請參閱標記 Amazon EBS 磁碟區。	

13. (選用) 為協助識別您的服務和任務，請展開 Tags (標籤) 區段，然後設定標籤。

    若要讓 Amazon ECS 使用叢集名稱和任務定義標籤，自動標記所有新啟動的任務，請選取 Turn on Amazon ECS managed tags (開啟 Amazon ECS 受管標籤)，然後針對 Propagate tags from (傳播標籤來源)，選取 Task definitions (任務定義)。

若要讓 Amazon ECS 使用叢集名稱和服務標籤，自動標記所有新啟動的任務，請選取 Turn on Amazon ECS managed tags (開啟 Amazon ECS 受管標籤)，然後針對 Propagate tags from (傳播標籤來源)，選取 Service (服務)。

新增或移除標籤。

- [新增標籤] 選擇新增標籤，然後執行下列操作：
  - 在索引鍵中，輸入索引鍵名稱。
  - 在值中，進入索引鍵值。
- [移除標籤] 在標籤旁邊，選擇移除標籤。

14. 選擇 Create (建立)。

後續步驟

追蹤您的部署，並檢視 Amazon ECS 斷路器服務的服務歷史記錄。如需詳細資訊，請參閱使用 Amazon ECS 服務部署檢視服務歷史記錄。

使用主控台更新 Amazon ECS 服務

您可以更新任務定義、所需的任務計數、容量提供者策略、平台版本和部署組態；或其中的任何組合。目前的服務組態已預先填入。

如需有關如何更新藍/綠部署組態的相關資訊，請參閱 使用主控台更新 Amazon ECS 藍/綠部署。

使用主控台時應考慮以下項目：

如果您想要暫時停止服務，請將所需任務設為 0。然後，當您準備好啟動服務時，請使用原始所需任務計數來更新服務。

使用主控台時應考慮以下項目：

- 您必須使用 AWS Command Line Interface 來更新使用下列任何參數的服務：
  - 藍/綠部署
  - 服務探索 – 您只能檢視您的服務探索組態。
  - 使用自訂指標追蹤政策
  - 更新服務 – 您無法更新awsvpc網路組態和運作狀態檢查寬限期。

如需有關如何使用 更新服務的資訊 AWS CLI，請參閱 AWS Command Line Interface 參考update-service中的 。

- 如果您要在任務定義中變更容器使用的連接埠，您可能需要更新容器執行個體的安全群組，才能使用更新的連接埠。
- Amazon ECS 不會自動更新與 Elastic Load Balancing 負載平衡器或 Amazon ECS 容器執行個體相關聯的安全群組。
- 如果您的服務使用負載平衡器，於建立時為您的服務定義的負載平衡器組態無法使用主控台變更。您可以改為使用 AWS CLI 或 SDK 來修改負載平衡器組態。如需有關如何修改組態的資訊，請參閱《Amazon Elastic Container Service API 參考UpdateService》中的 。
- 如果您為服務更新任務定義，則在負載平衡器組態中指定的容器名稱和容器連接埠必須保留在任務定義中。

您可以更新現有的服務，以變更某些服務組態參數，例如服務維護的任務數或任務使用的任務定義。另外，如果您使用的是 Fargate 任務類型，您可以變更任務使用的平台版本。使用 Linux 平台版本的服務無法更新為使用 Windows 平台版本，反之亦然。如果您的應用程式需要更多容量，您可以擴展您的服務。如果您有未使用的容量要縮減，您可以減少服務所需的任務數並釋出資源。

若要將更新容器映像用於任務，您可以使用該映像建立新任務定義修訂版，然後使用主控台中的 force new deployment (強制執行新部署) 選項來將其部署至服務。

服務排程器 (在服務的部署組態中) 使用運作狀態百分比下限和百分比上限參數，決定部署策略。

如果服務使用滾動更新 (ECS) 部署類型，運作狀態百分比下限代表部署期間，服務中必須維持在 RUNNING 狀態的任務數量下限，以所需任務數量的百分比表示 (無條件進位到最接近的整數)。如果服務包含的任務使用 EC2 啟動類型，即使有任何容器執行個體處於 DRAINING 狀態，此參數也適用。使用此參數進行部署，無須使用額外的叢集容量。例如，如果您的服務所需任務數量為四項，且運作狀態百分比下限為 50%，則排程器可先停止兩項現有的任務以釋出叢集容量，再啟動兩項新的任務。如果未使用負載平衡器的服務任務為 RUNNING 狀態，則視為運作良好。如果服務的任務使用負載平衡器，這些任務只要是處於 RUNNING 狀態，且負載平衡器也回報為運作良好，即視為運作良好。運作狀態百分比下限的預設值為 100%。

如果服務使用滾動更新 (ECS) 部署類型，則百分比上限參數代表在部署期間，服務中允許處於 PENDING、RUNNING 或 STOPPING 狀態的任務數量上限，以所需任務數量的百分比表示 (無條件捨去小數，只保留整數)。如果服務包含的任務使用 EC2 啟動類型，即使有任何容器執行個體處於 DRAINING 狀態，此參數也適用。使用此參數來定義部署批次大小。例如，如果您的服務有所需的四

個任務數目及百分比上限值 200%，則排程器可先啟動四項新任務，再停止四項舊任務。前提是有執行此操作所需的叢集資源。百分比上限的預設值為 200%。

當服務排程器於更新期間取代任務時，服務會先從負載平衡器 (如果使用) 移除任務，並等待連線耗盡。然後，對任務中執行的容器發出相當於 docker stop 的命令。這會造成 SIGTERM 信號和 30 秒逾時，並在之後傳送 SIGKILL，強制停止容器。如果容器在接收到 SIGTERM 信號後於 30 秒內從容處理完畢並結束，就不會傳送任何 SIGKILL 信號。服務排程器依據您定義的運作狀態百分比下限和百分比上限設定，啟動和停止任務。

在容器運作狀態檢查或負載平衡器目標群組運作狀態檢查失敗後，服務排程器也會取代判定為狀況不佳的任務。此取代取決於 maximumPercent 和 desiredCount 服務定義參數。如果任務標記為運作狀態不佳，服務排程器會先啟動替代任務。然後，會發生以下情況。

- 如果取代任務的運作狀態為 HEALTHY，服務排程器會停止運作狀態不佳的任務
- 如果替代任務的運作狀態為 UNHEALTHY，排程器會停止運作狀態不佳的替代任務或現有運作狀態不佳的任務，以使任務總計數等於 desiredCount。

如果 maximumPercent 參數限制排程器先啟動替代任務，排程器會隨機停止運作狀態不佳的任務，以釋放容量，然後啟動替代任務。開始和停止程序會繼續進行，直到所有運作狀態不佳的任務都會取代為運作狀態良好的 一旦已取代了所有運作狀態不佳的任務，而且只執行運作狀態良好的任務，如果任務總數超過 desiredCount，則運作良好的任務會隨機停止，直到任務總計數等於 desiredCount 為止。如需有關 maximumPercent 和 desiredCount 的詳細資訊，請參閱服務定義參數。

> **Important**
>
> 如果您要在任務定義中變更容器使用的連接埠，您可能需要更新容器執行個體的安全群組，才能使用更新的連接埠。
>
> 如果您為服務更新任務定義，則在服務建立時指定的容器名稱和容器連接埠必須保留在任務定義中。
>
> Amazon ECS 不會自動更新與 Elastic Load Balancing 負載平衡器或 Amazon ECS 容器執行個體相關聯的安全群組。

當您更新使用 Amazon ECS 斷路器的服務時，Amazon ECS 會建立服務部署和服務修訂。這些資源可讓您檢視服務歷史記錄的詳細資訊。如需詳細資訊，請參閱使用 Amazon ECS 服務部署檢視服務歷史記錄。

程序

1. 開啟主控台，網址為 https://console.aws.amazon.com/ecs/v2。
2. 在叢集頁面上，選擇叢集。
3. 在叢集詳細資訊頁面的服務區段中，選取服務旁的核取方塊，然後選擇更新。
4. 若要讓服務啟動新部署，請選取 Force new deployment (強制執行新部署)。
5. 在任務定義中，選擇任務定義系列和修訂。

> **Important**
>
> 主控台會驗證選取的任務定義系列和修訂是否與定義的運算組態相容。如果您收到警告，請確認任務定義相容性和您選擇的運算組態。

6. 如果您選擇 Replica (複寫)，針對 Desired tasks (所需任務)，輸入要在服務中啟動並維護的任務數。
7. 如果您選擇複本，若要讓 Amazon ECS 監控任務在可用區域之間的分佈，並在發生不平衡時重新分佈，請在可用區域服務重新平衡下，選取可用區域服務重新平衡。
8. 針對 Min running tasks (執行中任務下限)，輸入部署期間必須維持在 `RUNNING` 狀態的服務任務數量下限，它是所需任務數量的百分比 (無條件進位到最接近的整數)。如需詳細資訊，請參閱部署組態。
9. 針對 Max running tasks (執行中任務上限)，輸入部署期間允許的處於 `RUNNING` 或 `PENDING` 狀態的服務任務數目上限，它是所需任務數量的百分比 (無條件捨去到最接近的整數)。
10. 若要設定 Amazon ECS 如何偵測並處理部署失敗，請展開 Deployment failure detection (部署失敗偵測)，然後選擇您的選項。

    a. 若要在任務無法啟動時停止部署，請選取 Use the Amazon ECS deployment circuit breaker (使用 Amazon ECS 部署斷路器)。

       若要讓軟體在部署斷路器將部署設定為失敗狀態時，自動將部署復原至最後完成的部署狀態，請選取在失敗時復原。

    b. 若要根據應用程式指標停止部署，請選取使用 CloudWatch 警示 (s)。然後，從 CloudWatch 警示名稱中選擇警示。若要建立新的警示，請前往 CloudWatch 主控台。

       若要讓軟體在 CloudWatch 警示將部署設定為失敗狀態時，自動將部署復原至最後完成的部署狀態，請選取在失敗時復原。

11. 若要變更運算選項，請展開運算組態，然後執行下列動作：

a. 對於 上的服務 AWS Fargate，對於平台版本，請選擇新版本。

b. 對於使用容量提供者策略的服務，對於容量提供者策略，請執行下列動作：

- 若要新增其他容量提供者，請選擇 Add more (新增更多)。然後，針對 Capacity provider (容量提供者)，選擇容量提供者。
- 若要移除容量提供者，請選擇容量提供者右側的 Remove (移除)。

使用 Auto Scaling 群組容量提供者的服務無法更新為使用 Fargate 容量提供者。使用 Fargate 容量提供者的服務無法更新為使用 Auto Scaling 群組容量提供者。

12. （選用）若要設定服務 Auto Scaling，請展開服務自動擴展，然後指定下列參數。

a. 若要使用服務自動擴展，請選取 Service auto scaling (服務自動擴展)。

b. 針對任務數量下限，輸入服務自動擴展要使用的任務數量下限。所需的計數不會低於此計數。

c. 針對任務數量上限，輸入服務自動擴展要使用的任務數量上限。所需的計數不會高於此計數。

d. 選擇政策類型。在擴展政策類型下，選擇下列其中一個選項。

使用此政策類型	執行此作業	
目標追蹤	a. 針對 Scaling policy type (擴展政策類型)，選擇 Target tracking (目標追蹤)。 b. 針對 Policy name (政策名稱)，輸入政策的名稱。 c. 針對 ECS service metric (ECS 服務指標)，選擇下列其中一項指標。 • ECSServiceAverageCPUUtilization – 服務的平均 CPU 使用率。 • ECSServiceAverageMemoryUtilization – 服務的平均記憶體使用率。	

使用此政策類型	執行此作業	
	• ALBRequestCountPerTarget – Application Load Balancer 目標群組中每個目標完成的請求數量。 d. 針對 Target value (目標值)，輸入服務為選取的指標保持的值。 e. 對於橫向擴展冷卻時間，輸入橫向擴展活動（新增任務）之後，必須通過的時間量，然後才能開始另一個橫向擴展活動。 f. 對於縮減冷卻時間，輸入在縮減活動（移除任務）之後必須通過的時間量，然後才能開始另一個縮減活動。 g. 若要防止政策執行縮減活動，請選取 Turn off scale-in (關閉縮減)。 h. •（選用）如果您希望擴展政策向外擴展以增加流量，但不需要在流量減少時向內擴展，請選取關閉向內擴展。	

使用此政策類型	執行此作業	
步驟擴展	a. 針對 Scaling policy type (擴展政策類型)，選擇 Step scaling (步驟擴展)。 b. 在 Policy Name (政策名稱) 輸入政策的名稱。 c. 針對 Alarm name (警示名稱)，輸入警示的唯一名稱。 d. 針對 Amazon ECS service metric (Amazon ECS 服務指標)，選擇用於警示的指標。 e. 針對 Statistic (統計資料)，選擇警示統計資料。 f. 針對 Period (期間)，選擇警示的期間。 g. 針對 Alarm condition (警示條件)，選擇如何比較選取的指標與定義的閾值。 h. 針對 Threshold to compare metrics (比較閾值與指標) 和 Evaluation period to initiate alarm (啟動警示的評估期)，輸入用於警示的閾值以及評估閾值的時間長度。 i. 在 Scaling actions (擴展動作) 下，執行下列動作：	

使用此政策類型	執行此作業	
	• 針對動作，選取是否要新增、移除或設定服務的特定所需計數。 • 如果您選擇新增或移除任務，請在值中輸入啟動擴展動作時要新增或移除的任務數量（或現有任務的百分比）。如果您選擇設定所需的計數，請輸入任務數量。對於 Type (類型)，選取 Value (值) 是整數或是現有所需計數的百分比值。 • 針對 Lower bound (下限) 和 Upper bound (上限)，輸入步驟擴展調整的下界限和上界限。根據預設，新增政策的下限為警示閾值，而上限為無限大正數 (+) 值。根據預設，移除政策的上限為警示閾值，而下限為無限小負數 (-) 值。 • (選用) 新增其他擴展選項。選擇新增擴展動作，然後重複擴展動作步驟。 • 針對冷卻時間，以秒為單位輸入等待先前擴展活動生效的時間量。對於新增政策，這是擴展政策封鎖縮減活動，並	

使用此政策類型	執行此作業	
	限制一次可以縮減任務數量之後的時間。對於移除政策，這是在另一個縮減活動開始之前必須通過的縮減活動之後的時間。	

13. (選用) 若要使用 Service Connect，請選取 Turn on Service Connect (開啟 Service Connect)，然後指定下列項目：

    a. 在 Service Connect configuration (Service Connect 組態) 下，指定用戶端模式。

    - 如果服務執行的網路用戶端應用程式只需要連線到命名空間中的其他服務，請選擇 Client side only (僅用戶端)。
    - 如果服務執行的是網路或 Web 服務應用程式，且需要為此服務提供端點，並連線至命名空間中的其他服務，請選擇 Client and server (用戶端和伺服器)。

    b. 若要使用非預設叢集命名空間的命名空間，請在 Namespace (命名空間) 欄位中選擇服務命名空間。

14. 如果您的任務使用與部署時組態相容的資料磁碟區，您可以透過擴展磁碟區來設定磁碟區。

    磁碟區名稱和磁碟區類型是在您建立任務定義修訂時設定，且無法在更新服務時變更。若要更新磁碟區名稱和類型，您必須建立新的任務定義修訂，並使用新的修訂來更新服務。

設定此磁碟區類型	執行此作業	
Amazon EBS	a. 針對 EBS 磁碟區類型，選擇您要連接至任務的 EBS 磁碟區類型。 b. 針對大小 (GiB)，以 GB (GiB) 為單位輸入磁碟區大小的有效值。您可以指定至少 1 GiB 和最大 16，384 GiB 磁碟區大小。除非您提供快照 ID，否則需要此值。	

設定此磁碟區類型	執行此作業	
	c. 針對 IOPS，輸入磁碟區應提供的輸入/輸出操作 (IOPS) 數目上限。此值僅適用於 `io1`、`io2`、 和 gp3磁碟區類型。 d. 針對輸送量 (MiB/s)，以每秒 MB 為單位 (MiBps 或 MiB/s)，輸入磁碟區應提供的輸送量。此值只能針對gp3磁碟區類型設定。 e. 對於快照 ID，選擇現有的 Amazon EBS 磁碟區快照，或者如果您想要從快照建立磁碟區，請輸入快照的 ARN。您也可以不選擇或輸入快照 ID 來建立新的空白磁碟區。 f. 對於檔案系統類型，選擇用於磁碟區上資料儲存和擷取的檔案系統類型。您可以選擇作業系統預設值或特定檔案系統類型。Linux 的預設值為 XFS。對於從快照建立的磁碟區，您必須指定建立快照時磁碟區所使用的相同檔案系統類型。如果檔案系統類型不相符，任務將無法啟動。 g. 針對基礎設施角色，選擇具有必要許可的 IAM 角色，以允許 Amazon ECS 管理任務的 Amazon EBS 磁碟區。您可以將	

設定此磁碟區類型	執行此作業	
	AmazonECSInfrastructureRolePolicyForVolumes 受管政策連接至 角色，也可以使用政策做為建立和連接您自己的政策的指南，其許可可滿足您的特定需求。如需有關必要許可的詳細資訊，請參閱 Amazon ECS 基礎設施 IAM 角色。 h. 對於加密，如果您想要依預設設定使用 Amazon EBS 加密，請選擇預設。如果您的帳戶已預設設定加密，則磁碟區將使用設定中指定的 AWS Key Management Service (AWS KMS) 金鑰加密。如果您選擇預設，且 Amazon EBS 預設加密未開啟，則磁碟區將取消加密。 如果您選擇自訂，您可以指定 AWS KMS key 的磁碟區加密選擇。 如果您選擇無，除非您已預設設定加密，或者您從加密快照建立磁碟區，否則磁碟區將取消加密。 i. 如果您已選擇自訂加密，則必須指定要使用 AWS KMS key 的 。針對 KMS	

設定此磁碟區類型	執行此作業	
	金鑰，選擇 AWS KMS key 或輸入金鑰 ARN。如果您選擇使用對稱客戶受管金鑰來加密磁碟區，請確定您已在 AWS KMS key 政策中定義正確的許可。如需詳細資訊，請參閱 Amazon EBS 磁碟區的資料加密。 j. （選用）在標籤下，您可以透過從任務定義或服務傳播標籤，或提供您自己的標籤，將標籤新增至 Amazon EBS 磁碟區。 如果您想要從任務定義傳播標籤，請選擇傳播標籤的任務定義。如果您想要從 服務傳播標籤，請選擇 Service for Propagate 標籤來源。如果您選擇不傳播，或者如果您不選擇值，則不會傳播標籤。 如果您想要提供自己的標籤，請選擇新增標籤，然後針對您新增的每個標籤提供金鑰和值。 如需標記 Amazon EBS 磁碟區的詳細資訊，請參閱標記 Amazon EBS 磁碟區。	

15. (選用) 為協助識別您的服務，請展開 Tags (標籤) 區段，然後設定標籤。

- 【新增標籤】 選擇新增標籤，然後執行下列動作：
  - 在索引鍵中，輸入索引鍵名稱。
  - 在值中，進入索引鍵值。
- [移除標籤] 在標籤旁邊，選擇 移除標籤。

16. 選擇更新。

後續步驟

追蹤您的部署，並檢視 Amazon ECS 斷路器服務的服務歷史記錄。如需詳細資訊，請參閱使用 Amazon ECS 服務部署檢視服務歷史記錄。

使用主控台更新 Amazon ECS 藍/綠部署

您可以使用 Amazon ECS 主控台更新藍/綠部署組態 會預先填入目前的藍/綠部署組態。您可以更新下列藍/綠部署選項：

- 部署群組名稱 - CodeDeploy 部署設定
- 應用程式名稱 - CodeDeploy 部署群組
- 部署組態 - CodeDeploy 如何在部署期間將生產流量路由到您替代任務集的方式
- 在負載平衡器上的測試接聽程式 - CodeDeploy 在部署期間使用測試接聽程式，將您的測試流量路由到替代任務集

您必須先設定新選項，才能更新組態。

若要更新藍/綠部署組態 (Amazon ECS 主控台)

1. 開啟主控台，網址為 https://console.aws.amazon.com/ecs/v2。
2. 在 Clusters (叢集) 頁面上，選取您的叢集。
3. 在 Cluster overview (叢集概觀) 頁面中，選取服務，然後選擇 Update (更新)。
4. 展開Deployment (部署) 選項 - 由 CodeDeploy 提供支援，然後選擇要更新的選項：
   - 若要修改 CodeDeploy 部署群組，針對 Application name (應用程式名稱)，選擇部署群組。
   - 若要修改 CodeDeploy 部署設定，針對 Deployment group name (部署群組名稱)，選擇群組。
   - 若要修改 CodeDeploy 在部署期間將生產流量路由到替代任務集的方式，針對 Deployment configuration (部署組態)，請選擇選項。

5. 選取要在服務部署的新修訂版中運行的部署生命週期事件勾點和相關聯的 Lambda 函數。可用的生命週期勾點如下：

   - BeforeInstall：在建立替代任務集之前，使用此部署生命週期事件勾點來叫用 Lambda 函數。此生命週期事件上的 Lambda 函數結果不會起始回復。
   - AfterInstall：在建立替代任務集之後，使用此部署生命週期事件勾點來叫用 Lambda 函數。此生命週期事件上的 Lambda 函數結果可以起始回復。
   - BeforeAllowTraffic：在生產流量重新路由傳送到替代任務集之前，使用此部署生命週期事件勾點來叫用 Lambda 函數。此生命週期事件上的 Lambda 函數結果可以起始回復。
   - AfterAllowTraffic：在生產流量重新路由傳送到替代任務集之後，使用此部署生命週期事件勾點來叫用 Lambda 函數。此生命週期事件上的 Lambda 函數結果可以起始回復。

6. 若要修改測試接聽程式，請展開 Load balancing (負載平衡)，然後針對 Test listener for CodeDeploy deployment (CodeDeploy 部署的測試接聽程式)，選擇測試接聽程式。
7. 選擇更新。

使用主控台刪除 Amazon ECS 服務

以下是您刪除服務的一些原因：

- 不再需要應用程式
- 您正在將服務遷移至新的環境
- 應用程式未主動使用
- 應用程式使用的 資源超過所需，而您正嘗試最佳化成本

刪除之前，服務會自動將規模縮減至零。與該服務關聯的負載平衡器或服務探索資源，不會受到服務刪除的影響。若要刪除您的 Elastic Load Balancing 資源，請視您的負載平衡器類型，參閱下列任一主題：刪除 Application Load Balancer 或刪除 Network Load Balancer。

當您刪除服務時，Amazon ECS 會刪除服務的所有服務部署和服務修訂。

當您刪除服務時，如果仍有執行中的任務需要清除，服務狀態會從 移至 `ACTIVE` `DRAINING`，而且服務不會再出現在主控台或 `ListServices` API 操作中。在所有任務轉換為 `STOPPING` 或 `STOPPED` 狀態後，服務狀態會從 `DRAINING` 變為 `INACTIVE`。`DRAINING` 或 `INACTIVE` 狀態的服務仍可使用 `DescribeServices` API 操作檢視。

> **Important**
>
> 如果您嘗試建立與 ACTIVE或 DRAINING 狀態的現有服務名稱相同的新服務，您將會收到錯誤。

程序

1. 開啟主控台，網址為 https://console.aws.amazon.com/ecs/v2。
2. 在 Clusters (叢集) 頁面，為服務選取叢集。
3. 在 Clusters (叢集) 頁面上，選擇叢集。
4. 在 Cluster : ***name*** (叢集：名稱) 頁面上，選擇 Services (服務) 索引標籤。
5. 選取服務，然後選擇 Delete (刪除)。
6. 若要刪除服務，即使該服務並未縮減規模至零任務，請選取 Force delete service (強制刪除服務)。
7. 在確認提示中，輸入 Delete，然後選擇 Delete。

透過取代任務來部署 Amazon ECS 服務

當您建立使用滾動更新 (ECS) 部署類型的服務時，Amazon ECS 服務排程器會將目前正在執行的任務取代為新任務。Amazon ECS 在滾動更新期間從服務新增或移除的任務數量是由服務部署組態控制。

Amazon ECS 使用以下參數來判斷任務數量：

- `minimumHealthyPercent` 代表在部署期間或容器執行個體耗盡時，應為服務執行的任務數量下限 (佔服務所需任務數量的百分比)。此值會向上捨入。例如，如果最小運作狀態良好的百分比為 50，且所需的任務計數為 4，則排程器可以先停止兩項現有的任務，再開始兩項新的任務。同樣地，如果最小運作狀態良好的百分比為 75%，而所需的任務計數為 2，則排程器無法停止任何任務，因為產生的值也是 2。

  如果任務變得運作狀態不佳，Amazon ECS 服務排程器會先啟動替換任務，並維護`minimumHealthyPercent`任務，直到替換任務正常運作為止。隨著替代任務啟動並變得運作狀態良好，運作狀態不佳的任務將逐漸停止。
- `maximumPercent` 代表在部署期間或容器執行個體耗盡時，應為服務執行的任務數目上限 (佔服務所需任務數量的百分比)。此值會向下捨去。例如，如果最大百分比為 200，且所需的任務計數

為四，則排程器可以先停止四項現有的任務，再開始四項新的任務。同樣地，如果最大百分比為 125，且所需的任務計數為 3，則排程器無法停止任何任務，因為產生的值也是 3。

> **Important**
>
> 設定最小運作狀態良好的百分比或最大百分比時，您應確定排程器在部署初始化時，可以停止或啟動至少一項任務。如果您的服務部署因無效的部署組態而停滯，將傳送服務事件簡訊。如需詳細資訊，請參閱服務 (*service-name*) 在部署期間因服務部署組態而無法停止或啟動任務。更新 minimumHealthyPercent 或 maximumPercent 值，然後再試一次。。

滾動部署有 2 種方法，可讓您快速識別服務部署何時失敗：

- the section called “ 部署斷路器如何偵測故障”
- the section called “CloudWatch 警示如何偵測部署失敗”

可以單獨使用或一起使用這些方法。使用這兩種方法時，一旦符合任一失敗方法的失敗條件，部署就會設定為失敗。

請遵循下列準則來協助判斷要使用的方法：

- 斷路器 – 當您想要在任務無法啟動而停止部署時，請使用此方法。
- CloudWatch 警示 – 當您想要根據應用程式指標停止部署時，請使用此方法。

這兩種方法都支援轉返至先前的服務修訂。

容器映像解析度

根據預設，Amazon ECS 會將任務定義中指定的容器映像標籤解析為容器映像摘要。如果您建立執行和維護單一任務的服務，該任務會用來建立任務中容器的影像摘要。如果您建立執行和維護多個任務的服務，則服務排程器在部署期間啟動的第一個任務會用來建立任務中容器的影像摘要。

如果建立容器映像摘要的三次或三次以上嘗試失敗，則部署會繼續進行，而不需要映像摘要解析。如果啟用部署斷路器，則部署會額外失敗並復原。

建立容器映像摘要之後，Amazon ECS 會使用摘要來啟動任何其他所需的任務，以及用於任何未來的服務更新。這會導致服務中的所有任務一律執行相同的容器映像，進而產生軟體的版本一致性。

您可以使用容器定義中的 versionConsistency 參數，為任務中的每個容器設定此行為。如需詳細資訊，請參閱versionConsistency。

> **Note**
>
> - 低於 的 Amazon ECS Agent 版本1.31.0不支援映像摘要解析度。代理程式版本僅1.31.01.69.0支援推送至 Amazon ECR 儲存庫的影像的映像摘要解析度。客服人員版本1.70.0或更高版本支援所有映像的影像摘要解析度。
> - 映像摘要解析度的最低 Fargate Linux 平台版本為 1.3.0。映像摘要解析度的最低 Fargate Windows 平台版本為 1.0.0。
> - Amazon ECS 不會擷取由 Amazon ECS 管理的附屬容器摘要，例如 Amazon GuardDuty 安全代理程式或 Service Connect 代理。
> - 若要減少具有多個任務之 服務中與容器映像解析度相關的潛在延遲，請在 EC2 容器執行個體上執行 Amazon ECS 代理程式版本 1.83.0或更高版本。為了避免潛在的延遲，請在任務定義中指定容器映像摘要。
> - 如果您建立任務計數為零的服務，Amazon ECS 將無法建立容器摘要，直到您觸發另一個部署任務計數大於零的服務。
> - 若要建立更新的映像摘要，您可以強制新的部署。更新的摘要將用於啟動新任務，不會影響已執行的任務。如需強制新部署的詳細資訊，請參閱《Amazon ECS API 參考》中的 forceNewDeployment。

Amazon ECS 部署斷路器如何偵測故障

部署斷路器是可確定任務是否達到穩定狀態的滾動式更新機制。部署斷路器邏輯有一個選項，可將失敗的部署自動復原至處於 COMPLETED 狀態的部署。

當開始服務部署變更狀態時，Amazon ECS 會傳送服務部署狀態變更事件至 EventBridge。這提供一種程式設計方式來監控您的服務部署的狀態。如需詳細資訊，請參閱Amazon ECS 服務部署狀態變更事件。建議您使用 SERVICE_DEPLOYMENT_FAILED 的 eventName 建立和監控 EventBridge 規則，以便您可以採取手動動作來啟動部署。如需詳細資訊，請參閱《Amazon EventBridge 使用者指南》中的 EventBridge 入門。 EventBridge

當部署斷路器判斷部署失敗時，會尋找處於 COMPLETED 狀態的最新部署。這是部署斷路器用來做為復原部署的部署。復原開始時，部署會從 COMPLETED 變更為 IN_PROGRESS。這表示在達到 COMPLETED 狀態之前，部署不符合另一個復原的資格。當部署斷路器找不到處於 COMPLETED 狀態的部署時，斷路器不會啟動新任務，且會阻礙部署。

當您建立服務時，排程器會追蹤兩個階段中無法啟動的任務。

- 階段 1 - 排程器會監控任務，以查看它們是否轉換為 RUNNING 狀態。
  - 成功 - 部署有機會轉換至 COMPLETED 狀態，因為有多個任務已轉換至 RUNNING 狀態。會略過故障條件，且斷路器會移至階段 2。
  - 失敗 - 有連續任務未轉換為 RUNNING 狀態，且部署可能會轉換為 FAILED 狀態。
- 階段 2 - 當至少有一個任務處於 RUNNING 狀態時，部署會進入此階段。斷路器會檢查正在評估的目前部署中任務的運作狀態檢查。驗證的運作狀態檢查為 Elastic Load Balancing、 AWS Cloud Map 服務運作狀態檢查和容器運作狀態檢查。
  - 成功 - 至少有一個執行中狀態的任務已通過運作狀態檢查。
  - 失敗 - 由於運作狀態檢查失敗而取代的任務已達到失敗閾值。

當您對服務使用部署斷路器方法時，請考量下列事項。EventBridge 產生規則。

- `DescribeServices` 回應可提供部署狀態洞察，`rolloutState` 和 `rolloutStateReason`。當啟動新的部署時，推展狀態從 `IN_PROGRESS` 狀態開始。當服務到達穩定狀態時，推展狀態轉移到 `COMPLETED`。若服務無法達到穩定狀態，而斷路器已開啟，則部署將轉換為 `FAILED` 狀態。處於 `FAILED` 狀態的部署不會啟動任何新任務。
- 除 Amazon ECS 為已開始和已完成的部署傳送服務部署狀態變更事件以外，Amazon ECS 還會在開啟的斷路器部署失敗時傳送事件。這些事件提供有關部署失敗原因的詳細資訊，或者部署是否因為回復而開始。如需詳細資訊，請參閱Amazon ECS 服務部署狀態變更事件。
- 如果新的部署因為先前的部署失敗並發生復原而開始，服務部署狀態變更事件的 `reason` 欄位將指示部署因為復原而開始。
- 部署斷路器僅支援使用滾動更新 (ECS) 部署控制器的 Amazon ECS 服務。
- 當您搭配 CloudWatch 選項使用部署斷路器 AWS CLI 時，必須使用 Amazon ECS 主控台或 。如需詳細資訊，請參閱《AWS Command Line Interface 參考》中的 the section called “使用定義的參數建立服務” 和 create-service。

下列`create-service` AWS CLI 範例顯示當部署斷路器與復原選項搭配使用時，如何建立 Linux 服務。

```
aws ecs create-service \
     --service-name MyService \
     --deployment-controller type=ECS \
     --desired-count 3 \
```

```
    --deployment-configuration "deploymentCircuitBreaker={enable=true,rollback=true}"
\
    --task-definition sample-fargate:1 \
    --launch-type FARGATE \
    --platform-family LINUX \
    --platform-version 1.4.0 \
    --network-configuration
"awsvpcConfiguration={subnets=[subnet-12344321],securityGroups=[sg-12344321],assignPublicIp=EN
```

範例：

部署 1 處於 COMPLETED 狀態。

部署 2 無法啟動，因此斷路器會復原至部署 1。部署 1 會轉換至 IN_PROGRESS 狀態。

部署 3 會啟動，且 COMPLETED 狀態中沒有部署，因此部署 3 無法復原或啟動任務。

Failure threshold

部署斷路器會計算閾值，然後使用該值判斷何時將部署移動到 FAILED 狀態。

部署斷路器的最小閾值為 3，最大閾值為 200。 會使用下列公式中的值來判斷部署失敗。

```
Minimum threshold <= 0.5 * desired task count => maximum threshold
```

當計算結果大於最小值 3，但小於最大值 200 時，失敗閾值會設為計算閾值 （四捨五入）。

> **Note**
>
> 您不能更改任何一個閾值。

部署狀態檢查有兩個階段。

1. 部署斷路器監視部署過程中的任務，並檢查處於 RUNNING 狀態的任務。當目前部署中的任務處於 RUNNING 狀態時，排程器會忽略故障條件，並繼續進行到下一個階段。當任務無法到達 RUNNING 狀態時，部署斷路器會在故障計數上增加一。當故障計數等於閾值時，部署將標記為 FAILED。
2. 當RUNNING狀態中有一或多個任務時，就會進入此階段。部署斷路器對目前部署中的任務，針對以下資源執行運作狀態檢查：
   - Elastic Load Balancing 負載平衡器

- AWS Cloud Map 服務
- Amazon ECS 容器運作狀態檢查

當任務的運作狀態檢查失敗時，部署斷路器會在故障計數上增加一。當故障計數等於閾值時，部署將標記為 FAILED。

下表提供一些範例。

所需的任務計數	計算	Threshold
1	`3 <= 0.5 * 1 => 200`	3 （計算值小於最小值）
25	`3 <= 0.5 * 25 => 200`	13 (值會無條件進位)
400	`3 <= 0.5 * 400 => 200`	200
800	`3 <= 0.5 * 800 => 200`	200 (計算值大於最大值)

例如，當閾值為 3 時，斷路器會從失敗計數設定為 0 開始。當任務無法達到 RUNNING 狀態時，部署斷路器會將失敗計數增加一個。當失敗計數等於 3 時，部署會標記為 FAILED。

如需有關使用復原選項的其他範例，請參閱 Announcing Amazon ECS deployment circuit breaker (推出 Amazon ECS 部署斷路器)。

CloudWatch 警示如何偵測 Amazon ECS 部署失敗

您可以設定 Amazon ECS，以便在其偵測到指定的 CloudWatch 警示已進入 ALARM 狀態時，將部署設定為失敗。

您可以選擇性地設定組態，將失敗的部署復原至上次完成的部署。

下列`create-service` AWS CLI 範例顯示當部署警示與復原選項搭配使用時，如何建立 Linux 服務。

```
aws ecs create-service \
     --service-name MyService \
```

```
    --deployment-controller type=ECS \
    --desired-count 3 \
    --deployment-configuration
"alarms={alarmNames=[alarm1Name,alarm2Name],enable=true,rollback=true}" \
    --task-definition sample-fargate:1 \
    --launch-type FARGATE \
    --platform-family LINUX \
    --platform-version 1.4.0 \
    --network-configuration
"awsvpcConfiguration={subnets=[subnet-12344321],securityGroups=[sg-12344321],assignPublicIp=EN
```

當您對服務使用 Amazon CloudWatch 警示方式時，請考量下列事項。

- 封裝時間是新服務版本橫向擴展且舊服務版本縮減之後的一段時間，在此期間，Amazon ECS 會繼續監控與部署相關聯的警示。Amazon ECS 會根據與部署相關聯的警示組態來計算此時段。您無法設定此值。
- `deploymentConfiguration` 請求參數現在包含 `alarms` 資料類型。您可以指定警示名稱、是否使用該方法，以及是否在警示指出部署失敗時初始化復原。如需詳細資訊，請參閱《Amazon Elastic Container Service API 參考》中的 CreateService。
- `DescribeServices` 回應可提供部署狀態洞察，`rolloutState` 和 `rolloutStateReason`。當啟動新的部署時，推展狀態從 `IN_PROGRESS` 狀態開始。當服務達到穩定狀態且封裝時間完成時，推展狀態轉換為 `COMPLETED`。若服務無法達到穩定狀態，而警示已進入 `ALARM` 狀態，部署將轉換為 `FAILED` 狀態。處於 `FAILED` 狀態的部署不會啟動任何新任務。
- 除 Amazon ECS 為已開始和已完成的部署傳送服務部署狀態變更事件以外，Amazon ECS 還會在使用警示的部署失敗時傳送事件。這些事件提供有關部署失敗原因的詳細資訊，或者部署是否因為回復而開始。如需詳細資訊，請參閱Amazon ECS 服務部署狀態變更事件。
- 如果新的部署因為先前的部署失敗並已開啟復原而開始，服務部署狀態變更事件的 `reason` 欄位將指示部署因為復原而開始。
- 如果您使用部署斷路器和 Amazon CloudWatch 警示來偵測失敗，則任何一種方法都可以在符合任一方法的條件後立即初始化部署失敗。當您對初始化部署失敗的方法使用復原選項時，就會發生復原。
- Amazon CloudWatch 警示僅支援使用滾動更新 (ECS) 部署控制器的 Amazon ECS 服務。
- 您可以使用 Amazon ECS 主控台或 來設定此選項 AWS CLI。如需詳細資訊，請參閱《AWS Command Line Interface 參考》中的 the section called “使用定義的參數建立服務” 和 create-service。
- 您可能會注意到，部署狀態會維持 `IN_PROGRESS` 較長的時間。其原因是 Amazon ECS 在刪除作用中的部署之前不會變更狀態，並且直到封裝時間過後才會發生這種情況。視警示組態而定，部署需要的時間可能比您不使用警示多幾分鐘 (即使新的主要任務集已縱向擴展，而舊的部署也縮減規模)。如

果您使用 CloudFormation 逾時，請考慮增加逾時。如需詳細資訊，請參閱《AWS CloudFormation 使用者指南》中的在範本中建立等待條件。

- Amazon ECS 呼叫 `DescribeAlarms` 以輪詢警示。對 `DescribeAlarms` 的呼叫會計入與您帳戶相關聯的 CloudWatch 服務配額。如果您有 AWS 呼叫 的其他服務`DescribeAlarms`，Amazon ECS 可能會輪詢警示。例如，如果另一個服務發出足夠的`DescribeAlarms`呼叫來達到配額，則該服務會受到調節，Amazon ECS 也會受到調節，無法輪詢警示。如果在限流期間產生警示，Amazon ECS 可能會錯過警示，並且可能不會發生復原。針對部署沒有其他影響。如需 CloudWatch 服務配額的詳細資訊，請參閱《CloudWatch 使用者指南》中的 CloudWatch 服務配額。
- 如果警示在部署開始時處於 `ALARM` 狀態，Amazon ECS 將不會在部署期間監控警示 (Amazon ECS 會忽略警示組態)。此行為可解決您想要啟動新部署以修正初始部署失敗的情況。

建議的警示

我們建議您使用下列警示指標：

- 如果您使用 Application Load Balancer，請使用 `HTTPCode_ELB_5XX_Count` 和 `HTTPCode_ELB_4XX_Count` Application Load Balancer 指標。這些指標會檢查 HTTP 峰值。如需有關 Application Load Balancer 指標的詳細資訊，請參閱《Application Load Balancer 使用者指南》中的 Application Load Balancer 的 CloudWatch 指標。
- 如果您有現有的應用程式，請使用 `CPUUtilization` 和 `MemoryUtilization` 指標。這些指標會檢查叢集或服務使用的 CPU 和記憶體的百分比。如需詳細資訊，請參閱the section called “考量事項”。
- 如果您在任務中使用 Amazon Simple Queue Service 佇列，請使用 `ApproximateNumberOfMessagesNotVisible` Amazon SQS 指標。此指標會檢查佇列中延遲且無法立即讀取的訊息數量。如需有關 Amazon SQS 指標的詳細資訊，請參閱《Amazon Simple Queue Service 開發人員指南》中的適用於 Amazon SQS 的 CloudWatch 指標。

Amazon ECS 服務參數的最佳實務

為了確保沒有應用程式停機時間，部署程序如下所示：

1. 啟動新的應用程式容器，同時讓現有容器保持執行。
2. 檢查新容器是否正常運作。
3. 停止舊容器。

根據您的部署組態和叢集中可用、未預留的空間量，可能需要多回合才能完成以新任務取代所有舊任務。

您可以使用兩種服務組態選項來修改號碼：

- minimumHealthyPercent：100%（預設）

  部署期間，您的服務必須保持 RUNNING 狀態的任務數量下限。這是desiredCount四捨五入到最接近整數的百分比。此參數可讓您部署 ，而無需使用額外的叢集容量。

- maximumPercent：200%（預設）

  部署期間，在 RUNNING或 PENDING 狀態中允許的 服務任務數量上限。這是四desiredCount捨五入到最接近整數的百分比。

範例：預設組態選項

請考慮下列具有六個任務的服務，部署在總共有八個任務空間的叢集中。預設的服務組態選項不允許部署低於六個所需任務的 100%。

部署程序如下所示：

1. 目標是取代六個任務。
2. 排程器會啟動兩個新任務，因為預設設定需要有六個執行中的任務。

   現在有六個現有任務和兩個新任務。
3. 排程器會停止兩個現有的任務。

   現在有四個現有任務和兩個新的任務。
4. 排程器會啟動兩個額外的新任務。

   現在有四個現有任務和四個新任務。
5. 排程器會關閉兩個現有的任務。

   現在有兩個現有任務和四個新的任務。
6. 排程器會啟動兩個額外的新任務。

   現在有兩個現有任務和六個新任務
7. 排程器會關閉最後兩個現有的任務。

現在有六個新任務。

在上述範例中，如果您使用選項的預設值，則會有 2.5 分鐘的等待時間，等待每個新任務開始。此外，負載平衡器可能需要等待 5 分鐘，舊任務才能停止。

範例：修改 **minimumHealthyPercent**

您可以將 minimumHealthyPercent值設定為 50%，以加速部署。

請考慮下列具有六個任務的服務，部署在總共有八個任務空間的叢集中。部署程序如下所示：

1. 目標是取代六個任務。
2. 排程器會停止三個現有的任務。

   仍有三個現有任務正在執行，符合 minimumHealthyPercent值。
3. 排程器會啟動五個新任務。

   有三個現有任務任務和五個新任務。
4. 排程器會停止剩餘的三個現有任務。

   有五個新任務
5. 排程器會啟動最終的新任務。

   有六個新任務。

範例：修改叢集可用空間

您也可以新增額外的可用空間，以便執行其他任務。

請考慮下列具有六個任務的服務，部署在總共有十個任務空間的叢集中。部署程序如下所示：

1. 目標是取代現有的任務。
2. 排程器會停止三個現有的任務，

   有三個現有任務。
3. 排程器會啟動六個新任務。

   現有任務和六個新任務
4. 排程器會停止三個現有的任務。

有六個新任務。

建議

當您的任務閒置一段時間且沒有高使用率時，請針對服務組態選項使用下列值。

- `minimumHealthyPercent`：50%
- `maximumPercent`：200%

使用 Amazon ECS 服務部署檢視服務歷史記錄

服務部署提供部署的完整檢視。服務部署提供有關服務的下列資訊：

- 目前部署的工作負載組態（來源服務修訂版）
- 要部署的工作負載組態（目標服務修訂版）
- 部署狀態
- 偵測到迴路中斷的失敗任務數量
- 處於警示中的 CloudWatch 警示
- 服務部署開始和完成的時間
- 如果發生回復的詳細資訊

如需服務部署屬性的相關資訊，請參閱 Amazon ECS 服務部署中包含的屬性。

服務部署是唯讀的，每個都有唯一的 ID。

有三個服務部署階段：

階段	定義	關聯狀態
待定	已建立服務部署，但尚未啟動	待定
持續性	服務部署正在進行中	• IN_PROGRESS • STOP_REQUESTED • ROLLBACK_IN_PROGRESS

階段	定義	關聯狀態
已完成	服務部署已完成（成功或失敗）	• SUCCESSFUL • 已停止 • ROLLBACK_SUCCESSFUL • ROLLBACK_FAILED

您可以使用服務部署來了解服務的生命週期，並判斷是否有任何您需要採取的動作。例如，如果發生復原，您可能需要調查服務部署並查看服務事件。

您可以使用主控台、API 和 ，檢視在 2024 年 10 月 25 日或之後建立之部署的最新 90 天歷史記錄 AWS CLI。

服務部署生命週期

發生下列任何動作時，Amazon ECS 會自動建立新的服務部署：

- 使用者建立 服務。
- 使用者更新服務並使用強制新部署選項。
- 使用者更新一或多個需要部署的服務屬性。

當部署正在進行時，Amazon ECS 會更新下列服務部署屬性，以反映服務部署的進度：

- 狀態
- 執行中的任務數量

  服務修訂中指出的執行中任務數量，可能不等於實際執行中任務的數量。此數字代表部署完成時執行的任務數量。例如，如果您啟動的任務與服務部署無關，則這些服務不會包含在服務修訂版的執行中任務計數中。
- 斷路器故障偵測：
  - 無法啟動的任務數量
- CloudWatch 警示失敗偵測
  - 作用中的警示
- 回復資訊：
  - 開始時間

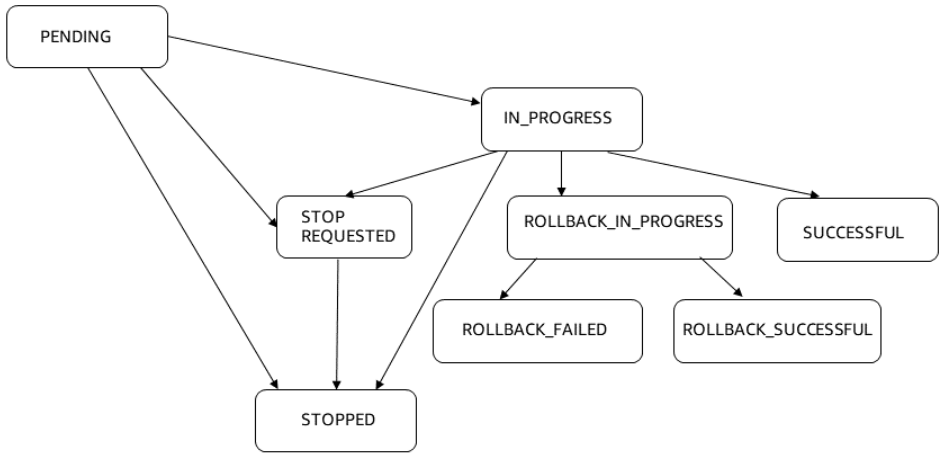
  - 復原的原因
  - 用於復原之服務修訂版的 ARN
- 狀態原因

當您刪除服務時，Amazon ECS 會刪除服務部署。

服務部署狀態

服務部署開始為 PENDING 狀態。

下圖顯示狀態後可能發生的服務部署PENDING狀態：IN_PROGRESS、SUCCESSFUL、STOP_REQUESTED、ROLLBACK_IN_PROGRESSS、ROLLBACK_FAILEROLLBACK_SUCCESSFUL和 STOPPED。



下列資訊提供有關服務部署狀態的詳細資訊：

- PENDING - 服務部署已建立，但尚未啟動。

  狀態可以移至 IN_PROGRESS、 STOP_REQUESTED或 STOPPED。
- IN_PROGRESS - 服務部署正在進行中。

狀態可以移至 SUCCESSFUL、ROLLBACK_IN_PROGRESS、 STOP_REQUESTED和 STOPPED。

- STOP_REQUESTED - 發生下列任何情況STOP_REQUESTED時，服務部署狀態會移至 ：
  - 使用者啟動新的服務部署。
  - 復原選項不適用於故障偵測機制 （斷路器或警示型），且服務不會達到 SUCCESSFUL 狀態。

  狀態會移至 STOPPED。
- SUCCESSFUL - 當服務部署成功完成SUCCESSFUL時，服務部署狀態會移至 。
- ROLLBACK_IN_PROGRESS - 當復原選項用於故障偵測機制 （斷路器或警示型） 且服務失敗ROLLBACK_IN_PROGRESS時，服務部署狀態會移至 。

  狀態會移至 ROLLBACK_SUCCESSFUL、 或 ROLLBACK_FAILED。

Amazon ECS 服務部署中包含的屬性

下列屬性包含在服務部署中。

屬性	描述	
服務部署 ARN	服務部署的 ARN。	
服務 ARN	此服務部署的服務 ARN。	
叢集 ARN	託管服務的叢集 ARN。	
服務部署建立時間	建立服務部署的時間。	
服務部署開始時間	服務部署開始的時間。	
服務部署完成時間	服務部署完成的時間。	
服務部署停止時間	服務部署停止的時間。	
服務部署更新時間	上次更新服務部署的時間。	
來源服務修訂	目前正在執行的服務修訂。	

屬性	描述
	如需包含屬性的詳細資訊，請參閱 Amazon ECS 服務修訂版中包含的屬性。
部署組態	部署參數包括斷路器組態、決定 的警示。 • 部署期間或容器執行個體耗盡時，應為服務執行的任務數量下限，以服務所需任務數量的百分比表示。 • 部署期間或容器執行個體耗盡時，應為服務執行的任務數量上限，以服務所需任務數量的百分比表示。 • 斷路器組態。 • 用於故障偵測的警示。
目標服務修訂	要部署的服務修訂版。 部署成功完成後，目標服務修訂即為執行中的服務修訂。
服務部署狀態	服務部署狀態。 有效值為 PENDING、SUCCESSFUL、STOPPED、STOP_REQUESTED、STOP_IN_PROGRESS、IN_PROGRESS、ROLLBACK_IN_PROGRESS、ROLLBACK_SUCCESSFUL 和 ROLLBACK_FAILED。

屬性	描述	
服務部署狀態資訊	服務部署為何處於目前狀態的相關資訊。例如，斷路器偵測到故障。	
回復資訊	服務部署在部署失敗時使用的復原選項。	
服務部署斷路器選項	判斷服務部署失敗的斷路器。	
服務部署的 CloudWatch 警示	決定服務部署何時失敗的 CloudWatch 警示。	

檢視 Amazon ECS 服務部署所需的許可

當您遵循授予最低權限的最佳實務時，您需要新增其他許可，才能在 主控台中檢視服務部署。

您需要存取下列動作：

- ListServiceDeployments
- DescribeServiceDeployments
- DescribeServiceRevisions

您需要存取下列資源：

- 服務
- 服務部署
- 服務修訂

下列範例政策包含必要的許可，並將動作限制為指定的服務。

將 `account`、 `cluster-name`和 取代`service-name`為您的值。

```
{
  "Statement": [
      {
          "Effect": "Allow",
```

```
            "Action": [
              "ecs:ListServiceDeployments",
              "ecs:DescribeServiceDeployments",
              "ecs:DescribeServiceRevisions"
            ],
            "Resource": [
             "arn:aws:ecs:us-east-1:123456789012:service/cluster-name/service-name",
             "arn:aws:ecs:us-east-1:123456789012:service-deployment/cluster-name/
service-name/*",
             "arn:aws:ecs:us-east-1:123456789012:service-revision/cluster-name/service-
name/*"
            ]
        }
    ]
}
```

檢視 Amazon ECS 服務部署

您可以查看在 2024 年 10 月 25 日或之後建立之部署的最新 90 天歷史記錄。服務部署可以處於下列任何狀態：

- 進行中
- 待定
- 已完成

您可以使用此資訊來判斷您是否需要更新服務的部署方式，還是服務修訂版。如需包含屬性的詳細資訊，請參閱 Amazon ECS 服務部署中包含的屬性。

開始之前，請設定檢視服務部署所需的許可。如需詳細資訊，請參閱檢視 Amazon ECS 服務部署所需的許可。

Amazon ECS Console

1. 開啟主控台，網址為 https://console.aws.amazon.com/ecs/v2。
2. 在叢集頁面上，選擇叢集。
3. 在叢集詳細資訊頁面上的服務區段中，選擇服務。

   服務詳細資訊頁面隨即顯示。
4. 在服務詳細資訊頁面上，選擇部署。
5. 選擇要檢視的服務部署。

檢視此部署類型的服務部署	執行此作業
持續部署	在持續部署下，選擇部署 ID。
上次部署	在上次部署下，選擇部署 ID。
已完成的部署	在服務部署下，選擇部署 ID。

服務部署詳細資訊頁面隨即出現。

6. （選用）比較服務修訂以檢視差異。

   在服務修訂下，選擇比較修訂，然後選擇 2 個修訂進行比較。

   服務修訂會side-by-side顯示，並反白顯示差異。

AWS CLI

1. 執行 `list-service-deployments`以擷取服務部署 ARN。

   將變數取代為您的值。

   ```
   aws ecs list-service-deployments --cluster cluster-name --service service-name
   ```

   請注意您要檢視之部署的 serviceDeploymentArn。

   ```
   {
       "serviceDeployments": [
           {
               "serviceDeploymentArn": "arn:aws:ecs:us-west-2:123456789012:service-
   deployment/example/sd-example/NCWGC2ZR-taawPAYrIaU5",
               "serviceArn": "arn:aws:ecs:us-west-2:123456789012:service/example/
   sd-example",
               "clusterArn": "arn:aws:ecs:us-west-2:123456789012:cluster/example",
               "targetServiceRevisionArn": "arn:aws:ecs:us-
   west-2:123456789012:service-revision/example/sd-example/4980306466373577095",
   ```

```
            "status": "SUCCESSFUL"
        }
    ]
}
```

2. 執行 describe-service-deployments。使用從 傳回serviceDeploymentArn的 list-service-deployments。

   將變數取代為您的值。

```
aws ecs describe-service-deployments --service-deployment-arns
 arn:aws:ecs:region:123456789012:service-deployment/cluster-name/service-
name/NCWGC2ZR-taawPAYrIaU5
```

後續步驟

您可以在 部署中檢視服務修訂的詳細資訊。如需詳細資訊，請參閱 檢視 Amazon ECS 服務修訂詳細資訊

Amazon ECS 服務修訂

服務修訂包含 Amazon ECS 正在嘗試部署的工作負載組態記錄。每當您建立或部署服務時，Amazon ECS 會自動建立和擷取您嘗試在服務修訂中部署的組態。

服務修訂為唯讀，並具有唯一的識別符。如需包含屬性的詳細資訊，請參閱 Amazon ECS 服務修訂版中包含的屬性。

服務修訂提供下列優點：

- 在服務部署期間，您可以比較目前部署的服務修訂版本 （ 來源 ） 與正在部署的服務修訂版本 （ 目標 ）。
- 當您使用服務部署的復原選項時，Amazon ECS 會自動將服務部署復原至上次成功部署的服務修訂。
- 服務修訂包含一個資源中的工作負載組態記錄。

服務修訂生命週期

當您建立服務，或更新啟動部署的服務屬性時，Amazon ECS 會自動建立新的服務修訂。

Amazon ECS 不會為復原操作建立新的服務修訂。Amazon ECS 會使用復原的上次成功服務修訂。

服務修訂是不可變的。

當您刪除服務時，Amazon ECS 會刪除服務修訂版。

您可以使用 主控台、API 和 CLI，檢視 2024 年 10 月 25 日當天或之後建立的服務修訂。

Amazon ECS 服務修訂版中包含的屬性

下列屬性包含在服務修訂中。

資源	描述	
服務 ARN	用於識別用戶端的 ARN。	
叢集 ARN	託管服務的叢集 ARN。	
任務定義 ARN	用於服務任務之任務定義的 ARN。	
服務登錄檔	用於服務探索之服務登錄檔的詳細資訊。 • 容器名稱 容器名稱值用於您的服務探索服務。 • 容器連接埠 連接埠值用於您的服務探索服務。此值也會在任務定義中指定。 • 連接埠 如果您的服務探索服務指定了 SRV 記錄，則使用該連接埠值。 • 登錄檔 ARN 服務登錄檔的 Amazon Resource Name (ARN)。	

資源	描述	
容量提供者	容量提供者策略詳細資訊。 • 容量提供者名稱 用於服務的容量提供者名稱。 • 容量提供者權重 應使用指定容量提供者啟動的任務總數的相對百分比。 • 容量提供者基礎 至少要在指定容量提供者上執行的任務數量。	
容器映像	容器映像的詳細資訊。 • 容器名稱 • 映像摘要 • 容器映像	
聯網	服務的網路組態。 • 是否有公有 IP 地址 • 任務在 中執行的子網路 • 控制服務流量的安全群組	
啟動類型	用於服務的啟動類型。	

資源	描述	
Fargate 特定屬性	當啟動類型為 Fargate 時，這是 Fargate 版本的相關資訊。 • 平台版本和平台系列 • 暫時性儲存 • 為部署配置的暫時性儲存量。 • 用於儲存加密的 KMS 金鑰。	

資源	描述	
部署時設定的 Amazon EBS 磁碟區	任務定義中指定的磁碟區的組態，為啟動時設定的磁碟區。 • Encrypted 表示磁碟區是否加密。 • 檔案系統類型。 磁碟區的 Linux 檔案系統類型。 • IOPS 每秒 I/O 操作的次數 (IOPS)。 • KMS 金鑰 • IAM 角色 要與此磁碟區建立關聯的 IAM 角色 ARN。 • 大小 磁碟區的大小 (GiB)。 • 快照 ID 磁碟區快照 ID。 • 標籤規格 磁碟區的標籤組態。 • 輸送量 為磁碟區佈建的輸送量。 • 磁碟區類型 磁碟區類型。	

資源	描述	
Service Connect	Service Connect 組態。 • 指出 Service Connect 是否正在使用 • 命名空間 • 互連服務清單 • Service Connect 記錄組態	
服務負載平衡器	路由服務流量的負載平衡器。 • 名稱 • 容器名稱 • 容器連接埠 • 目標群組 ARN	
執行期監控	指示執行期監控是否開啟。	
建立日期	建立服務修訂的日期。	
VPC Lattice	服務修訂版的 VPC Lattice 組態。	

檢視 Amazon ECS 服務修訂詳細資訊

您可以檢視 2024 年 10 月 25 日或之後建立的下列服務修訂類型的相關資訊：

- 來源 - 目前部署的工作負載組態
- 目標 - 正在部署的工作負載組態

Amazon ECS Console

1. 開啟主控台，網址為 https://console.aws.amazon.com/ecs/v2。
2. 在叢集頁面上，選擇叢集。
3. 在叢集詳細資訊頁面上的服務區段中，選擇服務。

服務詳細資訊頁面隨即顯示。

4. 在服務詳細資訊頁面上，選擇部署。
5. 選擇要檢視的服務修訂版。

檢視此部署類型的服務修訂	執行此作業
持續部署	在持續部署下，執行下列動作： • 若要檢視來源服務修訂，請在服務修訂下，選擇來源修訂類型的服務修訂 ID。 • 若要檢視目標服務修訂，請在服務修訂下，選擇目標修訂類型的服務修訂 ID。
上次部署	在上次部署下，選擇目標服務修訂。
已完成的部署	在服務部署下，執行下列動作： • 在目標服務修訂版下，選擇 ID。

AWS CLI

1. 執行 `describe-service-deployments`以擷取服務修訂版 ARN。

   將變數取代為您的值。

```
aws ecs describe-service-deployments --service-deployment-arns
 arn:aws:ecs:region:account-id:service/cluster-name/service-name/NCWGC2ZR-
taawPAYrIaU5
```

請注意arn適用於 的 sourceServiceRevisions或 targetServiceRevisions。


```
{
    "serviceDeployments": [
        {
            "serviceDeploymentArn": "arn:aws:ecs:us-west-2:123456789012:service-
deployment/example/sd-example/NCWGC2ZR-taawPAYrIaU5",
            "serviceArn": "arn:aws:ecs:us-west-2:123456789012:service/example/
sd-example",
            "clusterArn": "arn:aws:ecs:us-west-2:123456789012:cluster/example",
            "updatedAt": "2024-09-10T16:49:35.572000+00:00",
            "sourceServiceRevision": {
                "arn": "arn:aws:ecs:us-west-2:123456789012:service-revision/
example/sd-example/4980306466373578954",
                "requestedTaskCount": 0,
                "runningTaskCount": 0,
                "pendingTaskCount": 0
            },
            "targetServiceRevision": {
                "arn": "arn:aws:ecs:us-west-2:123456789012:service-revision/
example/sd-example/4980306466373577095",
                "requestedTaskCount": 0,
                "runningTaskCount": 0,
                "pendingTaskCount": 0
            },
            "status": "IN_PROGRESS",
            "deploymentConfiguration": {
                "deploymentCircuitBreaker": {
                    "enable": false,
                    "rollback": false
                },
                "maximumPercent": 200,
                "minimumHealthyPercent": 100
            }
        }
    ],
    "failures": []
}
```


2. 執行 describe-service-revisions。使用從 傳回arn的 describe-service-deployments。

將變數取代為您的值。

```
aws ecs describe-service-revisions --service-revision-arns
 arn:aws:ecs:region:123456789012:service-revision/cluster-name/service-
name/4980306466373577095
```

在部署之前驗證 Amazon ECS 服務的狀態

藍/綠部署類型使用由 CodeDeploy 控制的藍/綠部署模式。使用此部署類型在傳送生產流量到服務之前，先驗證服務的新部署。如需詳細資訊，請參閱AWS CodeDeploy 《 使用者指南》中的什麼是 CodeDeploy。在部署之前驗證 Amazon ECS 服務的狀態

在藍/綠部署期間，流量有三種轉移方式：

- Canary — 流量會以兩個增量轉移。您可以從預先定義的 canary 選項中選擇，這會指定流量轉移至您於第一次增量時更新之任務集的百分比，以及在剩餘流量於第二次增量前轉移的間隔 (以分鐘計)。
- 線性 — 流量以相等增量移動，每個增量之間的分鐘數相等。您可從預先指定的線性選項中指定每次增量的流量轉移百分比例，以及在每個增量之間的分鐘數。
- All-at-once — 所有流量都會一次從原始任務集轉移到更新的任務集。

以下是當在服務使用藍/綠部署類型時，Amazon ECS 使用的 CodeDeploy 的元件：

CodeDeploy 應用程式

CodeDeploy 資源的集合。這包含一或多個部署群組。

CodeDeploy 部署群組

部署設定。這包含下列各項：

- Amazon ECS 叢集和服務
- 負載平衡器目標群組和接聽器資訊
- 部署復原策略
- 流量重新路由設定
- 原始修訂終止設定
- 部署組態
- 可以設定為停止部署的 CloudWatch 警示組態

- 通知的 SNS 或 CloudWatch Events 設定

如需詳細資訊，請參閱 AWS CodeDeploy 使用者指南中的使用部署群組。

CodeDeploy 部署組態

指定 CodeDeploy 在部署期間將生產流量路由到您替代任務集的方式。以下預先定義的線性和金絲雀部署組態可供使用。您也可以建立自訂定義的線性和金絲雀部署。如需詳細資訊，請參閱 AWS CodeDeploy 使用者指南中的使用部署組態。

- CodeDeployDefault.ECSAllAtOnce：將所有流量一次轉移至更新的 Amazon ECS 容器
- CodeDeployDefault.ECSLinear10PercentEvery1Minutes：每分鐘轉移 10% 的流量，直到轉移所有流量為止。
- CodeDeployDefault.ECSLinear10PercentEvery3Minutes：每 3 分鐘轉移 10% 的流量，直到轉移所有流量為止。
- CodeDeployDefault.ECSCanary10Percent5Minutes：在第一個增量中轉移 10% 的流量。剩餘的 90% 會在五分鐘之後部署。
- CodeDeployDefault.ECSCanary10Percent15Minutes：在第一個增量中轉移 10% 的流量。剩餘的 90% 會在 15 分鐘之後部署。

修訂

修訂是 CodeDeploy 應用程式規格檔案 (AppSpec 檔案)。在 AppSpec 檔案中，您會指定任務定義的完整 ARN 和建立新部署時要路由流量到其中的替代任務集的容器和連接埠。容器名稱必須是在您的任務定義中參考的其中一個容器名稱。如果已在服務定義中更新網路組態或平台版本，您還必須在 AppSpec 檔案中指定這些詳細資訊。您也可以指定要在部署生命週期事件期間執行的 Lambda 函數。Lambda 函數可讓您在部署期間執行測試和傳回指標。如需詳細資訊，請參閱 AWS CodeDeploy 使用者指南中的 AppSpec 檔案參考。

考量事項

使用藍/綠部署類型時，請考慮下列各項：

- 一開始建立使用藍/綠部署類型的 Amazon ECS 服務時，會建立 Amazon ECS 任務集。
- 您必須設定服務，以使用 Application Load Balancer 或 Network Load Balancer。以下是負載平衡器需求：
  - 您必須將生產接聽器新增到負載平衡器 (用於路由傳送生產流量)。
  - 可以將選用的測試接聽器新增到負載平衡器，後者用於路由測試流量。如果您指定測試接聽程式，CodeDeploy 會在部署期間將您的測試流量路由到替代任務集。

  - 生產和測試接聽器必須屬於相同的負載平衡器。
  - 您必須為負載平衡器定義目標群組。目標群組會將流量透過生產接聽器路由到服務中的原始任務集。
  - 當使用 Network Load Balancer 時，僅支援 `CodeDeployDefault.ECSAllAtOnce` 部署組態。
- 對於被設定為使用服務自動擴展和藍/綠部署類型的服務，自動擴展不會在部署期間被封鎖，但部署可能在某些情況下失敗。以下詳細說明此行為。
  - 若服務在擴展而部署開始，綠任務集被建立，CodeDeploy 將最長等待一個小時並且不會轉移任何流量，直至綠任務集達到穩定狀態。
  - 如果服務正在進行藍/綠部署而發生擴展事件，流量將持續轉移 5 分鐘。若服務未在 5 分鐘內達到穩定狀態，CodeDeploy 將停止部署並將其標記為失敗。
  - 如果服務正在進行藍/綠部署且發生擴展事件，所需任務計數可能會設定為意外值。這是由於自動擴展將正在執行的任務計數視為當前容量，而其是所需任務計數計算中所使用的適當任務數的兩倍。
- 使用 Fargate 啟動類型或 `CODE_DEPLOY` 部署控制器類型的任務，不支援 `DAEMON` 排程策略。
- 當您一開始建立 CodeDeploy 應用程式和部署群組時，必須指定下列項目：
  - 您必須為負載平衡器定義兩個目標群組。一個目標群組必須是建立 Amazon ECS 服務時為負載平衡器定義的初始目標群組。第二個目標群組的唯一要求是，它無法與服務所使用的負載平衡器以外的其他負載平衡器相關聯。
- 當您為 Amazon ECS 服務建立 CodeDeploy 部署時，CodeDeploy 會在部署中建立替代任務集 (或綠任務集)。如果您將測試接聽程式新增至負載平衡器，CodeDeploy 會將您的測試流量路由到替代任務集。此時您可以執行任何驗證測試。然後，CodeDeploy 會根據部署群組的流量重新路由設定，將生產流量從原始任務集重新路由到替代任務集。

所需的 IAM 許可

Amazon ECS 和 CodeDeploy APIs 的組合可實現藍/綠部署。使用者必須擁有這些服務的適當許可，才能在 或 AWS Management Console 或 AWS CLI SDKs 中使用 Amazon ECS 藍/綠部署。

除了建立和更新服務的標準 IAM 許可之外，Amazon ECS 也需要下列許可。已將這些權限新增至 `AmazonECS_FullAccess` IAM 政策。如需詳細資訊，請參閱AmazonECS_FullAccess。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
```

```
            "Action": [
                "codedeploy:CreateApplication",
                "codedeploy:CreateDeployment",
                "codedeploy:CreateDeploymentGroup",
                "codedeploy:GetApplication",
                "codedeploy:GetDeployment",
                "codedeploy:GetDeploymentGroup",
                "codedeploy:ListApplications",
                "codedeploy:ListDeploymentGroups",
                "codedeploy:ListDeployments",
                "codedeploy:StopDeployment",
                "codedeploy:GetDeploymentTarget",
                "codedeploy:ListDeploymentTargets",
                "codedeploy:GetDeploymentConfig",
                "codedeploy:GetApplicationRevision",
                "codedeploy:RegisterApplicationRevision",
                "codedeploy:BatchGetApplicationRevisions",
                "codedeploy:BatchGetDeploymentGroups",
                "codedeploy:BatchGetDeployments",
                "codedeploy:BatchGetApplications",
                "codedeploy:ListApplicationRevisions",
                "codedeploy:ListDeploymentConfigs",
                "codedeploy:ContinueDeployment",
                "sns:ListTopics",
                "cloudwatch:DescribeAlarms",
                "lambda:ListFunctions"
            ],
            "Resource": ["*"]
        }
    ]
}
```

> **Note**
>
> 除了執行任務和服務所需的標準 Amazon ECS 許可之外，使用者還需要 `iam:PassRole` 許可，才能使用任務的 IAM 角色。

CodeDeploy 需要許可來呼叫 Amazon ECS API、修改 Elastic Load Balancing、叫用 Lambda 函數和說明 CloudWatch 警示，以及代表您修改服務所需計數的許可。建立使用藍/綠部署類型的 Amazon ECS 服務之前，您必須建立一個 IAM 角色 (`ecsCodeDeployRole`)。如需詳細資訊，請參閱Amazon ECS CodeDeploy IAM 角色。

建立 Amazon ECS 服務範例 和 更新 Amazon ECS 服務範例 IAM 政策範例顯示使用者在 AWS Management Console上使用 Amazon ECS 藍/綠部署時所需的許可。

使用藍/綠部署來部署 Amazon ECS 服務

了解如何建立 Amazon ECS 服務，其中包含搭配 使用藍/綠部署類型的 Fargate 任務 AWS CLI。

> **Note**
>
> 已為 AWS CloudFormation新增對於執行藍/綠部署的支援。如需詳細資訊，請參閱 AWS CloudFormation 使用者指南中的使用 AWS CloudFormation透過 CodeDeploy 執行 Amazon ECS 藍/綠部署。

必要條件

本教學課程假設已完成下列先決條件：

- AWS CLI 已安裝並設定最新版本的 。如需安裝或升級 的詳細資訊 AWS CLI，請參閱安裝或更新至最新版本的 AWS CLI。
- 已完成「設定以使用 Amazon ECS。」中的步驟。
- 您的 AWS 使用者具有 IAM AmazonECS_FullAccess 政策範例中指定的必要許可。
- 您已建立 VPC 和安全群組。如需詳細資訊，請參閱the section called “建立 Virtual Private Cloud”。
- 已建立 Amazon ECS CodeDeploy IAM 角色。如需詳細資訊，請參閱Amazon ECS CodeDeploy IAM 角色。

步驟 1：建立 Application Load Balancer

使用藍/綠部署類型的 Amazon ECS 服務需要使用 Application Load Balancer 或 Network Load Balancer。本教學課程會使用 Application Load Balancer。

建立 Application Load Balancer

1. 使用 create-load-balancer 命令來建立 Application Load Balancer。指定並非來自與安全群組相同可用區域的兩個子網路。

```
aws elbv2 create-load-balancer \
     --name bluegreen-alb \
```

```
    --subnets subnet-abcd1234 subnet-abcd5678 \
    --security-groups sg-abcd1234 \
    --region us-east-1
```

其輸出將包含負載平衡器的 Amazon Resource Name (ARN)，格式如下：

```
arn:aws:elasticloadbalancing:region:aws_account_id:loadbalancer/app/bluegreen-alb/
e5ba62739c16e642
```

2. 使用 create-target-group 命令建立目標群組。此目標群組會將流量路由到服務中的原始任務集。

```
aws elbv2 create-target-group \
    --name bluegreentarget1 \
    --protocol HTTP \
    --port 80 \
    --target-type ip \
    --vpc-id vpc-abcd1234 \
    --region us-east-1
```

其輸出將包含目標群組的 ARN，格式如下：

```
arn:aws:elasticloadbalancing:region:aws_account_id:targetgroup/
bluegreentarget1/209a844cd01825a4
```

3. 使用 create-listener 命令建立具有預設規則以將請求轉送至目標群組的負載平衡接聽程式。

```
aws elbv2 create-listener \
    --load-balancer-arn
 arn:aws:elasticloadbalancing:region:aws_account_id:loadbalancer/app/bluegreen-alb/
e5ba62739c16e642 \
    --protocol HTTP \
    --port 80 \
    --default-actions
 Type=forward,TargetGroupArn=arn:aws:elasticloadbalancing:region:aws_account_id:targetgroup
bluegreentarget1/209a844cd01825a4 \
    --region us-east-1
```

其輸出將包含接聽程式的 ARN，格式如下：

```
arn:aws:elasticloadbalancing:region:aws_account_id:listener/app/bluegreen-alb/
e5ba62739c16e642/665750bec1b03bd4
```

步驟 2：建立 Amazon ECS 叢集。

使用 create-cluster 命令來建立要使用之名為 tutorial-bluegreen-cluster 的叢集。

```
aws ecs create-cluster \
     --cluster-name tutorial-bluegreen-cluster \
     --region us-east-1
```

其輸出將包含叢集的 ARN，格式如下：

```
arn:aws:ecs:region:aws_account_id:cluster/tutorial-bluegreen-cluster
```

步驟 3：註冊任務定義

使用 register-task-definition 命令來註冊與 Fargate 相容的任務定義。這需要使用 awsvpc 網路模式。以下是用於此教學課程的任務定義範例。

首先，建立名為 fargate-task.json 且具有下列內容的檔案。確保您使用的是任務執行角色的 ARN。如需詳細資訊，請參閱Amazon ECS 任務執行 IAM 角色。

```
{
    "family": "sample-fargate",
    "networkMode": "awsvpc",
    "containerDefinitions": [
        {
            "name": "sample-app",
            "image": "public.ecr.aws/docker/library/httpd:latest",
            "portMappings": [
                {
                    "containerPort": 80,
                    "hostPort": 80,
                    "protocol": "tcp"
                }
            ],
            "essential": true,
            "entryPoint": [
                "sh",
  "-c"
            ],
            "command": [
                "/bin/sh -c \"echo '<html> <head> <title>Amazon ECS Sample
 App</title> <style>body {margin-top: 40px; background-color: #333;} </style> </
head><body> <div style=color:white;text-align:center> <h1>Amazon ECS Sample App</h1>
```

```
 <h2>Congratulations!</h2> <p>Your application is now running on a container in Amazon
 ECS.</p> </div></body></html>' >  /usr/local/apache2/htdocs/index.html && httpd-
foreground\""
            ]
        }
    ],
    "requiresCompatibilities": [
        "FARGATE"
    ],
    "cpu": "256",
    "memory": "512"
}
```

接下來，使用您建立的 fargate-task.json 檔案註冊任務定義。

```
aws ecs register-task-definition \
     --cli-input-json file://fargate-task.json \
     --region us-east-1
```

步驟 4：建立 Amazon ECS 服務

使用 create-service 命令來建立服務。

首先，建立名為 service-bluegreen.json 且具有下列內容的檔案。

```
{
    "cluster": "tutorial-bluegreen-cluster",
    "serviceName": "service-bluegreen",
    "taskDefinition": "tutorial-task-def",
    "loadBalancers": [
        {
            "targetGroupArn":
 "arn:aws:elasticloadbalancing:region:aws_account_id:targetgroup/
bluegreentarget1/209a844cd01825a4",
            "containerName": "sample-app",
            "containerPort": 80
        }
    ],
    "launchType": "FARGATE",
    "schedulingStrategy": "REPLICA",
    "deploymentController": {
        "type": "CODE_DEPLOY"
    },
```

```
    "platformVersion": "LATEST",
    "networkConfiguration": {
       "awsvpcConfiguration": {
          "assignPublicIp": "ENABLED",
          "securityGroups": [ "sg-abcd1234" ],
          "subnets": [ "subnet-abcd1234", "subnet-abcd5678" ]
       }
    },
    "desiredCount": 1
}
```

接下來，使用您建立的 service-bluegreen.json 檔案建立服務。

```
aws ecs create-service \
     --cli-input-json file://service-bluegreen.json \
     --region us-east-1
```

其輸出將包含服務的 ARN，格式如下：

```
arn:aws:ecs:region:aws_account_id:service/service-bluegreen
```

使用下列命令取得負載平衡器的 DNS 名稱。

```
 aws elbv2 describe-load-balancers --name bluegreen-alb  --query
 'LoadBalancers[*].DNSName'
```

在 Web 瀏覽器中輸入 DNS 名稱，您應該會看到一個網頁，其中顯示具有藍色背景的範例應用程式。

步驟 5：建立 AWS CodeDeploy 資源

請依照下列步驟建立 CodeDeploy 應用程式、CodeDeploy 部署群組的 Application Load Balancer 目標群組，以及 CodeDeploy 部署群組。

建立 CodeDeploy 資源

1. 使用 create-application 命令來建立 CodeDeploy 應用程式。指定 ECS 運算平台。

   ```
   aws deploy create-application \
        --application-name tutorial-bluegreen-app \
        --compute-platform ECS \
        --region us-east-1
   ```

其輸出將包含應用程式 ID，格式如下：

```
{
    "applicationId": "b8e9c1ef-3048-424e-9174-885d7dc9dc11"
}
```

2. 使用 create-target-group 命令建立第二個 Application Load Balancer 目標群組，該群組會在您建立 CodeDeploy 部署群組時用到。

```
aws elbv2 create-target-group \
     --name bluegreentarget2 \
     --protocol HTTP \
     --port 80 \
     --target-type ip \
     --vpc-id "vpc-0b6dd82c67d8012a1" \
     --region us-east-1
```

其輸出將包含目標群組的 ARN，格式如下：

```
arn:aws:elasticloadbalancing:region:aws_account_id:targetgroup/
bluegreentarget2/708d384187a3cfdc
```

3. 使用 create-deployment-group 命令來建立 CodeDeploy 部署群組。

首先，建立名為 `tutorial-deployment-group.json` 且具有下列內容的檔案。此範例會使用您建立的資源。對於 `serviceRoleArn`，請指定 Amazon ECS CodeDeploy IAM 角色的 ARN。如需詳細資訊，請參閱Amazon ECS CodeDeploy IAM 角色。

```
{
   "applicationName": "tutorial-bluegreen-app",
   "autoRollbackConfiguration": {
      "enabled": true,
      "events": [ "DEPLOYMENT_FAILURE" ]
   },
   "blueGreenDeploymentConfiguration": {
      "deploymentReadyOption": {
         "actionOnTimeout": "CONTINUE_DEPLOYMENT",
         "waitTimeInMinutes": 0
      },
      "terminateBlueInstancesOnDeploymentSuccess": {
         "action": "TERMINATE",
```

```
          "terminationWaitTimeInMinutes": 5
       }
    },
    "deploymentGroupName": "tutorial-bluegreen-dg",
    "deploymentStyle": {
       "deploymentOption": "WITH_TRAFFIC_CONTROL",
       "deploymentType": "BLUE_GREEN"
    },
    "loadBalancerInfo": {
       "targetGroupPairInfoList": [
         {
           "targetGroups": [
              {
                  "name": "bluegreentarget1"
              },
              {
                  "name": "bluegreentarget2"
              }
           ],
           "prodTrafficRoute": {
               "listenerArns": [
                   "arn:aws:elasticloadbalancing:region:aws_account_id:listener/
app/bluegreen-alb/e5ba62739c16e642/665750bec1b03bd4"
               ]
           }
         }
       ]
    },
    "serviceRoleArn": "arn:aws:iam::aws_account_id:role/ecsCodeDeployRole",
    "ecsServices": [
        {
            "serviceName": "service-bluegreen",
            "clusterName": "tutorial-bluegreen-cluster"
        }
    ]
}
```

接著建立 CodeDeploy 部署群組。

```
aws deploy create-deployment-group \
     --cli-input-json file://tutorial-deployment-group.json \
     --region us-east-1
```

其輸出將包含部署群組 ID，格式如下：

```
{
    "deploymentGroupId": "6fd9bdc6-dc51-4af5-ba5a-0a4a72431c88"
}
```

步驟 6：建立和監控 CodeDeploy 部署

在建立 CodeDeploy 部署之前，請依如下所示更新 `fargate-task.json` 中的任務定義 `command`，以將範例應用程式背景顏色變更為綠色。

```
{
    ...
        "containerDefinitions": [
            {
                ...
                "command": [
                    "/bin/sh -c \"echo '<html> <head> <title>Amazon ECS Sample
 App</title> <style>body {margin-top: 40px; background-color: #097969;} </style> </
head><body> <div style=color:white;text-align:center> <h1>Amazon ECS Sample App</h1>
 <h2>Congratulations!</h2> <p>Your application is now running on a container in Amazon
 ECS.</p> </div></body></html>' >  /usr/local/apache2/htdocs/index.html && httpd-
foreground\""
                ]
            }
        ],
       ...
}
```

使用以下命令註冊更新的任務定義。

```
aws ecs register-task-definition \
     --cli-input-json file://fargate-task.json \
     --region us-east-1
```

現在，請使用以下步驟來建立和上傳應用程式規格檔案 (AppSpec 檔案) 和 CodeDeploy 部署。

建立和監控 CodeDeploy 部署

1. 使用以下步驟來建立和上傳 AppSpec 檔案。

a. 建立名為 appspec.yaml 並具有 CodeDeploy 部署群組內容的檔案。此範例使用更新的任務定義。

```
version: 0.0
Resources:
  - TargetService:
      Type: AWS::ECS::Service
      Properties:
        TaskDefinition: "arn:aws:ecs:region:aws_account_id:task-
definition/tutorial-task-def:2"
        LoadBalancerInfo:
          ContainerName: "sample-app"
          ContainerPort: 80
        PlatformVersion: "LATEST"
```

b. 使用 s3 mb 命令為 AppSpec 檔案建立 Amazon S3 儲存貯體。

```
aws s3 mb s3://tutorial-bluegreen-bucket
```

c. 使用 s3 cp 命令來將 AppSpec 檔案上傳至 Amazon S3 儲存貯體。

```
aws s3 cp ./appspec.yaml s3://tutorial-bluegreen-bucket/appspec.yaml
```

2. 使用下列步驟來建立 CodeDeploy 部署。

a. 建立名為 create-deployment.json 並具有 CodeDeploy 部署內容的檔案。此範例會使用您先前在教學課程中建立的資源。

```
{
    "applicationName": "tutorial-bluegreen-app",
    "deploymentGroupName": "tutorial-bluegreen-dg",
    "revision": {
        "revisionType": "S3",
        "s3Location": {
            "bucket": "tutorial-bluegreen-bucket",
            "key": "appspec.yaml",
            "bundleType": "YAML"
        }
    }
}
```

b. 使用 create-deployment 命令來建立部署。

```
aws deploy create-deployment \
     --cli-input-json file://create-deployment.json \
     --region us-east-1
```

其輸出將包含部署 ID，格式如下：

```
{
    "deploymentId": "d-RPCR1U3TW"
}
```

3. 使用 get-deployment-target 命令，並從先前的輸出中指定 deploymentId 來取得部署的詳細資訊。

```
aws deploy get-deployment-target \
--deployment-id "d-IMJU3A8TW" \
--target-id tutorial-bluegreen-cluster:service-bluegreen \
--region us-east-1
```

最初，部署狀態為 InProgress。流量會導向至原始任務集，該任務集的 taskSetLabel 為 BLUE，狀態為 PRIMARY，且 trafficWeight 為 100.0。取代任務集的 taskSetLabel 為 GREEN，狀態為 ACTIVE，且 trafficWeight 為 0.0。您輸入 DNS 名稱的 Web 瀏覽器仍會以藍色背景顯示範例應用程式。

```
{
"deploymentTarget": {
"deploymentTargetType": "ECSTarget",
"ecsTarget": {
    "deploymentId": "d-RPCR1U3TW",
    "targetId": "tutorial-bluegreen-cluster:service-bluegreen",
    "targetArn": "arn:aws:ecs:region:aws_account_id:service/service-bluegreen",
    "lastUpdatedAt": "2023-08-10T12:07:24.797000-05:00",
    "lifecycleEvents": [
        {
            "lifecycleEventName": "BeforeInstall",
            "startTime": "2023-08-10T12:06:22.493000-05:00",
            "endTime": "2023-08-10T12:06:22.790000-05:00",
            "status": "Succeeded"
        },
        {
            "lifecycleEventName": "Install",
```

```
            "startTime": "2023-08-10T12:06:22.936000-05:00",
            "status": "InProgress"
        },
        {
            "lifecycleEventName": "AfterInstall",
            "status": "Pending"
        },
        {
            "lifecycleEventName": "BeforeAllowTraffic",
            "status": "Pending"
        },
        {
            "lifecycleEventName": "AllowTraffic",
            "status": "Pending"
        },
        {
            "lifecycleEventName": "AfterAllowTraffic",
            "status": "Pending"
        }
    ],
    "status": "InProgress",
    "taskSetsInfo": [
        {
            "identifer": "ecs-svc/9223370493423413672",
            "desiredCount": 1,
            "pendingCount": 0,
            "runningCount": 1,
            "status": "ACTIVE",
            "trafficWeight": 0.0,
            "targetGroup": {
                "name": "bluegreentarget2"
            },
            "taskSetLabel": "Green"
        },
        {
            "identifer": "ecs-svc/9223370493425779968",
            "desiredCount": 1,
            "pendingCount": 0,
            "runningCount": 1,
            "status": "PRIMARY",
            "trafficWeight": 100.0,
            "targetGroup": {
                "name": "bluegreentarget1"
            },
```

```
            "taskSetLabel": "Blue"
        }
    ]
}
}
}
```

繼續使用命令擷取部署詳細資訊，直到部署狀態為 Succeeded，如下列輸出所示。流量現在會重新導向至取代任務集，該任務集現在的狀態為 PRIMARY 且 trafficWeight 為 100.0。重新整理您輸入負載平衡器 DNS 名稱的 Web 瀏覽器，現在您應該會看到背景為綠色的範例應用程式。

```
{
"deploymentTarget": {
"deploymentTargetType": "ECSTarget",
"ecsTarget": {
    "deploymentId": "d-RPCR1U3TW",
    "targetId": "tutorial-bluegreen-cluster:service-bluegreen",
    "targetArn": "arn:aws:ecs:region:aws_account_id:service/service-bluegreen",
    "lastUpdatedAt": "2023-08-10T12:07:24.797000-05:00",
    "lifecycleEvents": [
        {
            "lifecycleEventName": "BeforeInstall",
            "startTime": "2023-08-10T12:06:22.493000-05:00",
            "endTime": "2023-08-10T12:06:22.790000-05:00",
            "status": "Succeeded"
        },
        {
            "lifecycleEventName": "Install",
            "startTime": "2023-08-10T12:06:22.936000-05:00",
            "endTime": "2023-08-10T12:08:25.939000-05:00",
            "status": "Succeeded"
        },
        {
            "lifecycleEventName": "AfterInstall",
            "startTime": "2023-08-10T12:08:26.089000-05:00",
            "endTime":  "2023-08-10T12:08:26.403000-05:00",
            "status": "Succeeded"
        },
        {
            "lifecycleEventName": "BeforeAllowTraffic",
            "startTime": "2023-08-10T12:08:26.926000-05:00",
            "endTime":  "2023-08-10T12:08:27.256000-05:00",
            "status": "Succeeded"
```

```
        },
        {
            "lifecycleEventName": "AllowTraffic",
            "startTime": "2023-08-10T12:08:27.416000-05:00",
            "endTime": "2023-08-10T12:08:28.195000-05:00",
            "status": "Succeeded"
        },
        {
            "lifecycleEventName": "AfterAllowTraffic",
            "startTime": "2023-08-10T12:08:28.715000-05:00",
            "endTime":  "2023-08-10T12:08:28.994000-05:00",
            "status": "Succeeded"
        }
    ],
    "status": "Succeeded",
    "taskSetsInfo": [
        {
            "identifer": "ecs-svc/9223370493425779968",
            "desiredCount": 1,
            "pendingCount": 0,
            "runningCount": 1,
            "status": "ACTIVE",
            "trafficWeight": 0.0,
            "targetGroup": {
                "name": "bluegreentarget1"
            },
            "taskSetLabel": "Blue"
        },
        {
            "identifer": "ecs-svc/9223370493423413672",
            "desiredCount": 1,
            "pendingCount": 0,
            "runningCount": 1,
            "status": "PRIMARY",
            "trafficWeight": 100.0,
            "targetGroup": {
                "name": "bluegreentarget2"
            },
            "taskSetLabel": "Green"
        }
    ]
}
}
```

```
}
```

步驟 7：清除

完成此教學課程時，清除與其相關的資源，以免未使用的資源產生費用。

清除教學課程資源

1. 使用 delete-deployment-group 命令來刪除 CodeDeploy 部署群組。

```
aws deploy delete-deployment-group \
     --application-name tutorial-bluegreen-app \
     --deployment-group-name tutorial-bluegreen-dg \
     --region us-east-1
```

2. 使用 delete-application 命令來刪除 CodeDeploy 應用程式。

```
aws deploy delete-application \
     --application-name tutorial-bluegreen-app \
     --region us-east-1
```

3. 使用 delete-service 命令來刪除 Amazon ECS 服務。使用 `--force` 旗標可讓您即使在服務未縮減為零個任務時仍能將其刪除。

```
aws ecs delete-service \
     --service arn:aws:ecs:region:aws_account_id:service/service-bluegreen \
     --force \
     --region us-east-1
```

4. 使用 delete-cluster 命令來刪除 Amazon ECS 叢集。

```
aws ecs delete-cluster \
     --cluster tutorial-bluegreen-cluster \
     --region us-east-1
```

5. 使用 s3 rm 命令來將 AppSpec 檔案從 Amazon S3 儲存貯體中刪除。

```
aws s3 rm s3://tutorial-bluegreen-bucket/appspec.yaml
```

6. 使用 s3 rb 命令來刪除 Amazon S3 儲存貯體。

```
aws s3 rb s3://tutorial-bluegreen-bucket
```

7. 使用 delete-load-balancer 命令刪除 Application Load Balancer。

```
aws elbv2 delete-load-balancer \
     --load-balancer-arn
 arn:aws:elasticloadbalancing:region:aws_account_id:loadbalancer/app/bluegreen-alb/
e5ba62739c16e642 \
     --region us-east-1
```

8. 使用 delete-target-group 命令，刪除兩個 Application Load Balancer 目標群組。

```
aws elbv2 delete-target-group \
     --target-group-arn
 arn:aws:elasticloadbalancing:region:aws_account_id:targetgroup/
bluegreentarget1/209a844cd01825a4 \
     --region us-east-1
```

```
aws elbv2 delete-target-group \
     --target-group-arn
 arn:aws:elasticloadbalancing:region:aws_account_id:targetgroup/
bluegreentarget2/708d384187a3cfdc \
     --region us-east-1
```

使用第三方控制器部署 Amazon ECS 服務

外部部署類型可讓您使用任何第三方部署控制器，以完全控制 Amazon ECS 服務的部署程序。服務的詳細資訊是由服務管理 API 動作 (`CreateService`、`UpdateService` 和 `DeleteService`) 或任務設定管理 API 動作 (`CreateTaskSet`、`UpdateTaskSet`、`UpdateServicePrimaryTaskSet` 和 `DeleteTaskSet`) 所管理。每個 API 動作會管理服務定義參數的子集。

`UpdateService` API 動作更新服務的所需計數和運作狀態檢查寬限期參數。如果需要更新啟動類型、平台版本、負載平衡器詳細資訊、網路組態或任務定義，您必須建立新的任務集。

`UpdateTaskSet` API 動作只更新任務集的擴展參數。

`UpdateServicePrimaryTaskSet` API 動作修改服務中哪個任務集是主要任務集。當您呼叫 `DescribeServices` API 動作時，它會傳回主要任務集指定的所有欄位。如果更新服務的主要任務

集，當定義新的主要任務時，存在於新主要任務集的任何任務集參數值 (不同於服務中的舊主要任務集) 將會更新為新的值。如果沒有為服務定義主要任務集，則描述服務時，任務集欄位是 null。

外部部署考量

使用外部部署類型時，請考慮下列各項：

- 支援的負載平衡器類型可為 Application Load Balancer 或 Network Load Balancer。
- Fargate 啟動類型或 EXTERNAL 部署控制器類型不支援 DAEMON 排程策略。

外部部署工作流程

以下是在 Amazon ECS 上管理外部部署的基本工作流程。

使用外部部署控制器管理 Amazon ECS 服務

1. 建立 Amazon ECS 服務。唯一的必要參數是服務名稱。使用外部部署控制器建立服務時，您可以指定以下參數。在服務內建立任務集時，指定所有其他服務參數。

   `serviceName`

   類型：字串

   必要：是

   您的服務名稱。可以包含最多可達 255 個字元 (大小寫)、數字、連字號和底線。叢集中不得有相同的服務名稱，但一個區域內或多個區域間的多個叢集中可以有類似的服務名稱。

   `desiredCount`

   指定的任務集任務定義的執行個體化數量，將放在服務內保持執行。

   `deploymentConfiguration`

   選用的部署參數，可控制在部署期間執行多少任務以及停止和啟動任務的順序。

   `tags`

   類型： 物件陣列

   必要：否

您套用到服務以協助您分類和組織的中繼資料。每個標籤皆包含由您定義的一個金鑰與一個選用值。刪除服務時，也會一併刪除標籤。服務最多可套用 50 個標籤。如需詳細資訊，請參閱標記 Amazon ECS 資源。

key

類型：字串

長度限制：長度下限為 1。長度上限為 128。

必要：否

組成標籤的鍵值組的一部分。索引鍵是一般標籤，作用就像更特定標籤值的類別。

value

類型：字串

長度限制：長度下限為 0。長度上限為 256。

必要：否

組成標籤的鍵值組的選用部分。值就像標籤類別 (索引鍵) 內的描述項。

enableECSManagedTags

指定是否對服務內的使用 Amazon ECS 受管標籤。如需詳細資訊，請參閱使用標籤計費。

propagateTags

類型：字串

有效值：TASK_DEFINITION | SERVICE

必要：否

指定是否將標籤從任務定義或服務複製到服務中的任務。如果沒有指定值，則不會複製標籤。標籤只能在建立服務期間複製到服務內的任務。若要在建立服務或任務後對任務新增標籤，請使用 TagResource API 動作。

schedulingStrategy

使用的排程策略。使用外部部署控制器的服務僅支援 REPLICA 排程策略。

`placementConstraints`

您服務的任務要使用的置放限制物件陣列。每項任務您最多可以指定 10 項限制 (此限制包含任務定義中的限制以及執行時間指定的限制)。若您使用的是 Fargate 啟動類型，則不支援任務置放限制條件。

`placementStrategy`

您服務的任務要使用的置放策略物件。每項服務您最多可以指定四項策略規則。

以下是使用外部部署控制器建立服務的服務定義範例。

```
{
    "cluster": "",
    "serviceName": "",
    "desiredCount": 0,
    "role": "",
    "deploymentConfiguration": {
        "maximumPercent": 0,
        "minimumHealthyPercent": 0
    },
    "placementConstraints": [
        {
            "type": "distinctInstance",
            "expression": ""
        }
    ],
    "placementStrategy": [
        {
            "type": "binpack",
            "field": ""
        }
    ],
    "schedulingStrategy": "REPLICA",
    "deploymentController": {
        "type": "EXTERNAL"
    },
    "tags": [
        {
            "key": "",
            "value": ""
        }
    ],
```

```
    "enableECSManagedTags": true,
    "propagateTags": "TASK_DEFINITION"
}
```

2. 建立初始任務集。任務集包含以下關於服務的詳細資訊：

   `taskDefinition`

   供任務集的任務使用的任務定義。

   `launchType`

   類型：字串

   有效值：EC2 | FARGATE | EXTERNAL

   必要：否

   您服務執行所在的啟動類型。如果未指定啟動類型，預設會使用預設的 `capacityProviderStrategy`。如需詳細資訊，請參閱Amazon ECS啟動類型。

   若有指定 `launchType`，則必須省略 `capacityProviderStrategy` 參數。

   `platformVersion`

   類型：字串

   必要：否

   您服務中任務正在其上執行的平台版本。平台版本僅會為使用 Fargate 啟動類型的任務指定。如果尚未指定，預設會使用最新版本 (LATEST)。

   AWS Fargate 平台版本用於參考 Fargate 任務基礎設施的特定執行期環境。在您執行任務或建立服務時指定 LATEST 平台版本，即可取得任務所能使用的最新平台版本。當您擴展服務規模時，這些任務會收到於服務的目前部署上所指定的平台版本。如需詳細資訊，請參閱適用於 Amazon ECS 的 Fargate 平台版本。

   > **Note**
   >
   > 並未針對使用 EC2 啟動類型的任務指定平台版本。

`loadBalancers`

用來代表您的服務所使用之負載平衡器的負載平衡器物件。當使用外部部署控制器時，僅支援 Application Load Balancer 和 Network Load Balancer。如果您使用 Application Load Balancer，每個任務集僅允許一個 Application Load Balancer 目標群組。

以下程式碼片段顯示要使用的 `loadBalancer` 物件範例。

```
"loadBalancers": [
        {
            "targetGroupArn": "",
            "containerName": "",
            "containerPort": 0
        }
]
```

> **Note**
>
> 指定 `loadBalancer` 物件時，您必須指定 `targetGroupArn` 並省略 `loadBalancerName` 參數。

`networkConfiguration`

服務的網路組態。使用 `awsvpc` 網路模式的任務定義需要此參數，才能接收其自己的彈性網路介面，其他網路模式不支援此參數。如需 Fargate 啟動類型聯網的詳細資訊，請參閱 Fargate 啟動類型的 Amazon ECS 任務聯網選項。

`serviceRegistries`

要指派給此服務的服務探索登錄檔詳細資訊。如需詳細資訊，請參閱使用服務探索將 Amazon ECS 服務與 DNS 名稱連線。

`scale`

需要放在任務集來保持執行的任務數的浮點百分比。值是以服務的 `desiredCount` 的百分比總計指定。接受的值為 0 到 100 之間的數字。

以下是為外部部署控制器建立任務集的 JSON 範例。

```
{
    "service": "",
    "cluster": "",
    "externalId": "",
    "taskDefinition": "",
    "networkConfiguration": {
        "awsvpcConfiguration": {
            "subnets": [
                ""
            ],
            "securityGroups": [
                ""
            ],
            "assignPublicIp": "DISABLED"
        }
    },
    "loadBalancers": [
        {
            "targetGroupArn": "",
            "containerName": "",
            "containerPort": 0
        }
    ],
    "serviceRegistries": [
        {
            "registryArn": "",
            "port": 0,
            "containerName": "",
            "containerPort": 0
        }
    ],
    "launchType": "EC2",
    "capacityProviderStrategy": [
        {
            "capacityProvider": "",
            "weight": 0,
            "base": 0
        }
    ],
    "platformVersion": "",
    "scale": {
        "value": null,
        "unit": "PERCENT"
```

```
    },
    "clientToken": ""
}
```

3. 需要服務變更時，請根據您更新哪些參數，使用 `UpdateService`、`UpdateTaskSet` 或 `CreateTaskSet` API 動作。如果您建立任務集，請對服務中的每個任務集使用 `scale` 參數，以決定在服務中要保持執行多少個任務。例如，如果您有一個服務包含 `tasksetA` 並建立 `tasksetB`，您可能想要先測試 `tasksetB` 的有效性，然後才將生產流量轉移給它。您可以將這兩個任務集的 `scale` 都設為 `100`，而當您準備好將所有生產流量轉移至 `tasksetB` 時，您可以將 `tasksetA` 的 `scale` 更新為 `0` 以縮減它。

使用負載平衡來分發 Amazon ECS 服務流量

您的服務可以選擇性地設定為使用 Elastic Load Balancing，將流量平均分散到服務中的任務。

> **Note**
> 使用任務集時，集中的所有任務必須全部設定為使用 Elastic Load Balancing 或不使用 Elastic Load Balancing。

在 上託管的 Amazon ECS 服務 AWS Fargate 支援 Application Load Balancer、Network Load Balancer 和 Gateway Load Balancer。使用下表了解要使用的負載平衡器類型。

Load Balancer類型	在這些情況下使用	
Application Load Balancer	路由 HTTP/HTTPS （或第 7 層）流量。 Application Load Balancer 提供數種功能，非常適合與 Amazon ECS 服務搭配使用： • 每項服務都可以透過指定多個目標群組，為來自多個負載平衡器的流量提供服務，以及公開多個負載平衡的連接埠。	

Load Balancer類型	在這些情況下使用	
	• 它們受在 Fargate 和 EC2 執行個體上託管的任務的支援。 • Application Load Balancer 允許容器使用動態主機連接埠映射 (允許每個容器執行個體之相同服務中的多個任務)。 • Application Load Balancer 支援路徑類型路由和優先順序規則 (因此多個服務可以在單一 Application Load Balancer 上使用相同的接聽程式連接埠)。	
Network Load Balancer	路由 TCP 或 UDP （或第 4 層） 流量。	
Gateway Load Balancer	路由 TCP 或 UDP （或第 4 層） 流量。 使用虛擬設備，例如防火牆、入侵偵測和預防系統，以及深度封包檢查系統。	

我們建議您將 Application Load Balancer 用於 Amazon ECS 服務，以便您可以利用這些最新功能，除非您的服務需要僅 Network Load Balancer 或 Gateway Load Balancer 提供的功能。如需有關 Elastic Load Balancing 和負載平衡器類型區別的詳細資訊，請參閱 Elastic Load Balancing 使用者指南。

使用負載平衡器時，您只需按實際用量付費。如需詳細資訊，請參閱 Elastic Load Balancing 定價。

最佳化 Amazon ECS 的負載平衡器運作狀態檢查參數

負載平衡器只會將請求路由至負載平衡器可用區域中運作狀態良好的目標。每個目標都會註冊到目標群組。負載平衡器會使用目標群組運作狀態檢查設定來檢查每個目標的運作狀態。註冊目標之後，它必須通過一個運作狀態檢查，才能視為運作狀態良好。Amazon ECS 會監控負載平衡器。負載平衡器會定

期將運作狀態檢查傳送至 Amazon ECS 容器。Amazon ECS 代理程式會監控並等待負載平衡器報告容器運作狀態。它會先這麼做，再將容器視為運作狀態良好。

兩個 Elastic Load Balancing 運作狀態檢查參數會影響部署速度：

- 運作狀態檢查間隔：決定個別容器運作狀態檢查之間的大約時間量，以秒為單位。根據預設，負載平衡器會每 30 秒檢查一次。

  此參數命名為：

  - `HealthCheckIntervalSeconds` 在 Elastic Load Balancing API 中
  - Amazon EC2 主控台上的間隔
- 運作狀態良好的閾值計數：決定在考慮運作狀態不佳的容器之前，連續進行運作狀態檢查所需的成功次數。根據預設，負載平衡器需要五個傳遞運作狀態檢查，才能報告目標容器運作狀態良好。

  此參數命名為：

  - `HealthyThresholdCount` 在 Elastic Load Balancing API 中
  - Amazon EC2 主控台上的運作狀態良好的閾值

使用預設設定，判斷容器運作狀態的總時間是兩分鐘 30 秒 (`30 seconds * 5 = 150 seconds`)。

如果您的服務在 10 秒內啟動並穩定，您可以加快運作狀態檢查程序。若要加速程序，請減少運作狀態檢查的數量和檢查之間的間隔。

- `HealthCheckIntervalSeconds` (Elastic Load Balancing API 名稱） 或間隔 (Amazon EC2 主控台名稱）：5
- `HealthyThresholdCount` (Elastic Load Balancing API 名稱） 或 Healthy threshold (Amazon EC2 主控台名稱）：2

使用此設定時，運作狀態檢查程序需要 10 秒，相較於預設值為 2 分鐘 30 秒。

如需 Elastic Load Balancing 運作狀態檢查參數的詳細資訊，請參閱Elastic Load Balancing 使用者指南》中的目標群組的運作狀態檢查。

最佳化 Amazon ECS 的負載平衡器連線耗盡參數

若要允許最佳化，用戶端會維持與容器服務的即時連線。這可讓該用戶端的後續請求重複使用現有的連線。當您想要停止對容器的流量時，請通知負載平衡器。負載平衡器會定期檢查用戶端是否關閉保持連

線。Amazon ECS 代理程式會監控負載平衡器，並等待負載平衡器報告保持運作連線已關閉（目標處於 UNUSED 狀態）。

負載平衡器將目標移至 UNUSED 狀態的等待時間量是取消註冊延遲。您可以設定下列負載平衡器參數來加速部署。

- `deregistration_delay.timeout_seconds`：300（預設）

當您的服務回應時間低於 1 秒時，請將 參數設定為下列值，讓負載平衡器只等待 5 秒，然後再中斷用戶端與後端服務之間的連線：

- `deregistration_delay.timeout_seconds`：5

> **Note**
>
> 當您的服務具有長期請求時，請勿將值設定為 5 秒，例如慢速檔案上傳或串流連線。

SIGTERM 回應能力

Amazon ECS 會先傳送 SIGTERM 訊號至任務，以通知應用程式需要完成並關閉。然後，Amazon ECS 會傳送 SIGKILL 訊息。當應用程式忽略 SIGTERM 時，Amazon ECS 服務必須等待傳送 SIGKILL 訊號以終止程序。

Amazon ECS 等待傳送 SIGKILL 訊息的時間量取決於下列 Amazon ECS 代理程式選項：

- `ECS_CONTAINER_STOP_TIMEOUT`：30（預設）

  如需容器代理程式參數的詳細資訊，請參閱 GitHub 上的 Amazon ECS Container Agent。

若要加快等待期，請將 Amazon ECS 代理程式參數設定為下列值：

- `ECS_CONTAINER_STOP_TIMEOUT`: 2

  如果您的應用程式需要超過 1 秒，請將值乘以 2，並使用該數字做為值。

在此情況下，Amazon ECS 會等待 2 秒讓容器關閉，然後 Amazon ECS 會在應用程式未停止時傳送 SIGKILL 訊息。

您也可以修改應用程式程式碼，以截留 SIGTERM 訊號並對其做出反應。以下是 JavaScript 中的範例：

```
process.on('SIGTERM', function() {
  server.close();
})
```

此程式碼會導致 HTTP 伺服器停止接聽任何新請求、完成回應任何處理中的請求，然後 Node.js 程序會終止，因為事件迴圈沒有什麼可做的。有鑑於此，如果程序只需要 500 毫秒就能完成其傳輸中請求，則其會提早終止，而不必等待停止逾時並收到 SIGKILL。

使用適用於 Amazon ECS 的 Application Load Balancer

Application Load Balancer 在應用程式層 (HTTP/HTTPS) 進行路由決策、支援以路徑為基礎的路由，並可將請求路由到您叢集中每個容器執行個體的一或多個連接埠。Application Load Balancer 支援動態主機連接埠映射。例如，如果您的任務的容器定義指定 NGINX 容器連接埠使用連接埠 80，主機連接埠使用連接埠 0，則會從容器執行個體的暫時性連接埠範圍 (例如最新的 Amazon ECS 最佳化 AMI 為 32768 至 61000) 動態選擇主機連接埠。任務啟動時，NGINX 容器會向 Application Load Balancer 註冊為執行個體 ID 和連接埠組合，而流量會分佈到與該容器對應的執行個體 ID 和連接埠。這個動態映射可讓您在同一個容器執行個體中的單一服務擁有多項任務。如需詳細資訊，請參閱 Application Load Balancer 使用者指南。

如需設定參數以加速部署的最佳實務相關資訊，請參閱：

- 最佳化 Amazon ECS 的負載平衡器運作狀態檢查參數
- 最佳化 Amazon ECS 的負載平衡器連線耗盡參數

搭配 Amazon ECS 使用 Application Load Balancer 時，請考慮下列事項：

- Amazon ECS 需要服務連結 IAM 角色，該角色會在任務建立和停止時提供向您的負載平衡器註冊與取消註冊目標所需的許可。如需詳細資訊，請參閱使用 Amazon ECS 的服務連結角色。
- 目標群組必須將 IP 地址類型設定為 IPv4。
- 對於任務使用 `awsvpc` 網路模式的服務，當您為服務建立目標群組時，必須選擇 `ip` 做為目標類型，而非 `instance`。這是由於採用 `awsvpc` 網路模式的任務是與彈性網路介面相關聯，而非與 Amazon EC2 執行個體相關聯。
- 如果您的服務需要存取多個負載平衡連接埠，例如 HTTP/HTTPS 服務的連接埠 80 和連接埠 443，您可以設定兩個接聽程式。一個接聽程式負責將請求轉送至服務的 HTTPS，另一個接聽程式負責將

HTTP 請求重新導向至適當的 HTTPS 連接埠。如需詳細資訊，請參閱 Application Load Balancer 使用者指南中的為您的 Application Load Balancer 建立接聽程式。

- 負載平衡器子網路組態必須包含您容器執行個體所在的所有可用區域。
- 建立服務之後，負載平衡器組態無法從 AWS Management Console變更。您可以使用 AWS Copilot AWS CloudFormation AWS CLI 或 SDK 來修改ECS滾動部署控制器的負載平衡器組態，而不是 AWS CodeDeploy 藍/綠或外部。當您新增、更新或移除負載平衡器組態時，Amazon ECS 會使用更新後的 Elastic Load Balancing 組態啟動新的部署。這會導致任務向負載平衡器註冊和取消註冊。我們建議您在更新 Elastic Load Balancing 組態之前，先在測試環境中驗證。如需有關如何修改組態的資訊，請參閱 Amazon Elastic Container Service API 參考 中的 UpdateService。
- 如果服務任務未通過負載平衡器運作狀態檢查條件，則任務會停止並重新啟動。此程序會持續到您的服務達到所需的執行任務數量為止。
- 如果您遇上啟用負載平衡器功能之服務的問題，請參閱對 Amazon ECS 中的服務負載平衡器進行故障診斷。
- 您的任務和負載平衡器必須位於相同的 VPC 中。
- 為每個服務使用唯一的目標群組。

  針對多個服務使用相同的目標群組，可能會導致服務部署期間發生問題。

如需有關如何建立 Application Load Balancer 的資訊，請參閱在 Application Load Balancer 中建立 Application Load Balancer

使用適用於 Amazon ECS 的網路Load Balancer

Network Load Balancer 在傳輸層進行路由決策 (TCP/SSL)。每秒可以處理數百萬個請求。在負載平衡器收到連線後，會使用流程雜湊路由演算法從目標群組選取預設規則的目標。負載平衡器會嘗試開啟接聽程式設定中指定之連接埠上所選目標的 TCP 連線。負載平衡器會轉送此請求，且不會修改標頭。Network Load Balancer 支援動態主機連接埠映射。例如，如果您的任務的容器定義指定 NGINX 容器連接埠使用連接埠 80，主機連接埠使用連接埠 0，則會從容器執行個體的暫時性連接埠範圍 (例如最新的 Amazon ECS 最佳化 AMI 為 32768 至 61000) 動態選擇主機連接埠。啟動此任務時，NGINX 容器會以執行個體 ID 和連接埠組合在 Network Load Balancer 註冊，而流量將分散到與該容器對應的執行個體 ID 和連接埠。這個動態映射可讓您在同一個容器執行個體中的單一服務擁有多項任務。如需詳細資訊，請參閱 Network Load Balancer 使用者指南。

如需設定參數以加速部署的最佳實務相關資訊，請參閱：

- 最佳化 Amazon ECS 的負載平衡器運作狀態檢查參數

- 最佳化 Amazon ECS 的負載平衡器連線耗盡參數

搭配 Amazon ECS 使用 Network Load Balancer 時，請考慮下列事項：

- Amazon ECS 需要服務連結 IAM 角色，該角色會在任務建立和停止時提供向您的負載平衡器註冊與取消註冊目標所需的許可。如需詳細資訊，請參閱使用 Amazon ECS 的服務連結角色。
- 您無法將超過五個目標群組連接至服務。
- 對於任務使用 `awsvpc` 網路模式的服務，當您為服務建立目標群組時，必須選擇 `ip` 做為目標類型，而非 `instance`。這是由於採用 `awsvpc` 網路模式的任務是與彈性網絡介面相關聯，而非與 Amazon EC2 執行個體相關聯。
- 負載平衡器子網路組態必須包含您容器執行個體所在的所有可用區域。
- 建立服務之後，負載平衡器組態無法從 AWS Management Console變更。您可以使用 AWS Copilot AWS CloudFormation AWS CLI 或 SDK 來修改ECS滾動部署控制器的負載平衡器組態，而不是 AWS CodeDeploy 藍/綠或外部。當您新增、更新或移除負載平衡器組態時，Amazon ECS 會使用更新後的 Elastic Load Balancing 組態啟動新的部署。這會導致任務向負載平衡器註冊和取消註冊。我們建議您在更新 Elastic Load Balancing 組態之前，先在測試環境中驗證。如需有關如何修改組態的資訊，請參閱 Amazon Elastic Container Service API 參考 中的 UpdateService。
- 如果服務任務未通過負載平衡器運作狀態檢查條件，則任務會停止並重新啟動。此程序會持續到您的服務達到所需的執行任務數量為止。
- 當您使用將 IP 地址設定為目標且用戶端 IP 保留關閉的 Gateway Load Balancer 時，請求會被視為來自 Gateway Load Balancer 私有 IP 地址。這表示一旦您允許目標安全群組中的傳入請求和運作狀態檢查，Gateway Load Balancer 後方的服務就會有效地向全世界開放。
- 對於 Fargate 任務，您必須使用平台版本 `1.4.0`(Linux) 或 `1.0.0`(Windows)。
- 如果您遇上啟用負載平衡器功能之服務的問題，請參閱對 Amazon ECS 中的服務負載平衡器進行故障診斷。
- 您的任務和負載平衡器必須位於相同的 VPC 中。
- Network Load Balancer 用戶端 IP 地址保留與 Fargate 目標相容。
- 為每個服務使用唯一的目標群組。

  針對多個服務使用相同的目標群組，可能會導致服務部署期間發生問題。

如需如何建立 Network Load Balancer 的資訊，請參閱在 Network Load Balancer 中建立 Network Load Balancer

> **Important**
>
> 若服務的任務定義是採用 Fargate 啟動類型所需的 `awsvpc` 網路模式，則您必須選擇 `ip` 做為目標類型，而不是選擇 `instance`。因為採用 `awsvpc` 網路模式的任務是與彈性網路界面相關聯，而非與 Amazon EC2 執行個體相關聯。
> 如果執行個體類型如下：
> C1、CC1、CC2、CG1、CG2、CR1、G1、G2、HI1、HS1、M1、M2、M3 和 T1，就不能以執行個體 ID 來登錄執行個體。您可以使用 IP 地址來登錄這些執行個體。

使用適用於 Amazon ECS 的 Gateway Load Balancer

Gateway Load Balancer 在開放系統互相連線 (OSI) 模型的第三層，網路層運作。它會接聽所有連接埠的全部 IP 封包，並轉送流量至接聽程式規則中指定的目標群組。它使用 5 元組 (針對 TCP/UDP 流) 或 3 元組 (針對非 TCP/UDP 流) 維護特定目標應用裝置的流量黏性。例如，如果您的任務的容器定義指定 NGINX 容器連接埠使用連接埠 80，主機連接埠使用連接埠 0，則會從容器執行個體的暫時性連接埠範圍 (例如最新的 Amazon ECS 最佳化 AMI 為 32768 至 61000) 動態選擇主機連接埠。任務啟動時，NGINX 容器會向 Gateway Load Balancer 註冊為執行個體 ID 和連接埠組合，而流量會分佈到與該容器對應的執行個體 ID 和連接埠。這個動態映射可讓您在同一個容器執行個體中的單一服務擁有多項任務。如需詳細資訊，請參閱 Gateway Load Balancer 中的什麼是 Gateway Load Balancer。

如需設定參數以加速部署的最佳實務相關資訊，請參閱：

- 最佳化 Amazon ECS 的負載平衡器運作狀態檢查參數
- 最佳化 Amazon ECS 的負載平衡器連線耗盡參數

搭配 Amazon ECS 使用 Gateway Load Balancer 時，請考慮下列事項：

- Amazon ECS 需要服務連結 IAM 角色，該角色會在任務建立和停止時提供向您的負載平衡器註冊與取消註冊目標所需的許可。如需詳細資訊，請參閱使用 Amazon ECS 的服務連結角色。
- 對於任務使用 `awsvpc` 網路模式的服務，當您為服務建立目標群組時，必須選擇 `ip` 做為目標類型，而非 `instance`。這是由於採用 `awsvpc` 網路模式的任務是與彈性網絡介面相關聯，而非與 Amazon EC2 執行個體相關聯。
- 負載平衡器子網路組態必須包含您容器執行個體所在的所有可用區域。
- 建立服務之後，負載平衡器組態無法從 AWS Management Console變更。您可以使用 AWS Copilot AWS CloudFormation AWS CLI 或 SDK 來修改ECS滾動部署控制器的負載平衡器組態，而不是 AWS CodeDeploy 藍/綠或外部。當您新增、更新或移除負載平衡器組態時，Amazon ECS 會使用更

新後的 Elastic Load Balancing 組態啟動新的部署。這會導致任務向負載平衡器註冊和取消註冊。我們建議您在更新 Elastic Load Balancing 組態之前，先在測試環境中驗證。如需有關如何修改組態的資訊，請參閱 Amazon Elastic Container Service API 參考 中的 UpdateService。

- 如果服務任務未通過負載平衡器運作狀態檢查條件，則任務會停止並重新啟動。此程序會持續到您的服務達到所需的執行任務數量為止。
- 當您使用將 IP 地址設定為目標的 Gateway Load Balancer 時，請求會被視為來自 Gateway Load Balancer 私有 IP 地址。這表示一旦您允許目標安全群組中的傳入請求和運作狀態檢查，Gateway Load Balancer 後方的服務就會有效地向全世界開放。
- 對於 Fargate 任務，您必須使用平台版本 `1.4.0`(Linux) 或 `1.0.0`(Windows)。
- 如果您遇上啟用負載平衡器功能之服務的問題，請參閱對 Amazon ECS 中的服務負載平衡器進行故障診斷。
- 您的任務和負載平衡器必須位於相同的 VPC 中。
- 為每個服務使用唯一的目標群組。

  針對多個服務使用相同的目標群組，可能會導致服務部署期間發生問題。

如需有關如何建立 Gateway Load Balancer 的資訊，請參閱 Gateway Load Balancer 中的 Gateway Load Balancer 入門

> **Important**
>
> 若服務的任務定義是採用 Fargate 啟動類型所需的 `awsvpc` 網路模式，則您必須選擇 `ip` 做為目標類型，而不是選擇 `instance`。因為採用 `awsvpc` 網路模式的任務是與彈性網路界面相關聯，而非與 Amazon EC2 執行個體相關聯。
>
> 如果執行個體類型如下：
>
> C1、CC1、CC2、CG1、CG2、CR1、G1、G2、HI1、HS1、M1、M2、M3 和 T1，就不能以執行個體 ID 來登錄執行個體。您可以使用 IP 地址來登錄這些執行個體。

使用 Amazon ECS 服務註冊多個目標群組

當您在服務定義中指定多個目標群組時，您的 Amazon ECS 服務可以為來自多個負載平衡器的流量提供服務，以及公開多個負載平衡的連接埠。

若要建立指定多個目標群組的服務，您必須使用 Amazon ECS API AWS CLI、 SDK 或 AWS CloudFormation 範本建立服務。建立服務之後，您可以使用 AWS Management Console檢視該服務以及向其註冊的目標群組。您必須使用 `UpdateService` 來修改現有服務的負載平衡器組態。

您可以使用以下格式在服務定義中指定多個目標群組。如需服務定義的完整語法，請參閱服務定義範本。

```
"loadBalancers":[
   {

 "targetGroupArn":"arn:aws:elasticloadbalancing:region:123456789012:targetgroup/
target_group_name_1/1234567890123456",
      "containerName":"container_name",
      "containerPort":container_port
   },
   {

 "targetGroupArn":"arn:aws:elasticloadbalancing:region:123456789012:targetgroup/
target_group_name_2/6543210987654321",
      "containerName":"container_name",
      "containerPort":container_port
   }
]
```

考量事項

當您在服務定義中指定多個目標群組時，應該考慮下列事項。

- 對於使用 Application Load Balancer 或 Network Load Balancer 的服務，您無法連接超過五個目標群組到一個服務。
- 只有在下列情況下支援在服務定義中指定多個目標群組：
  - 服務必須使用 Application Load Balancer 或 Network Load Balancer。
  - 服務必須使用滾動更新 (ECS) 部署控制器類型。
- 若服務同時包含使用 Fargate 和 EC2 啟動類型的任務，則該服務支援指定多個目標群組。
- 當建立指定多個目標群組的服務時，必須建立 Amazon ECS 服務連結角色。透過省略 API 請求中的 `role` 參數，或 AWS CloudFormation中的 `Role` 屬性來建立角色。如需詳細資訊，請參閱使用 Amazon ECS 的服務連結角色。

範例服務定義

以下是在服務定義中指定多個目標群組的一些使用案例範例。如需服務定義的完整語法，請參閱服務定義範本。

為內部和外部流量使用單獨的負載平衡器

在以下的使用案例中，服務會對相同的容器和連接埠使用兩個不同的負載平衡器，一個用於內部流量，另一個用於面向網際網路的流量。

```
"loadBalancers":[
   //Internal ELB
   {

 "targetGroupArn":"arn:aws:elasticloadbalancing:region:123456789012:targetgroup/
target_group_name_1/1234567890123456",
      "containerName":"nginx",
      "containerPort":8080
   },
   //Internet-facing ELB
   {

 "targetGroupArn":"arn:aws:elasticloadbalancing:region:123456789012:targetgroup/
target_group_name_2/6543210987654321",
      "containerName":"nginx",
      "containerPort":8080
   }
]
```

從相同容器公開多個連接埠

在以下的使用案例中，服務會使用一個負載平衡器，但公開相同容器中的多個連接埠。例如，Jenkins 容器可公開適用於 Jenkins Web 界面的連接埠 8080，以及公開適用於 API 的連接埠 50000。

```
"loadBalancers":[
   {

 "targetGroupArn":"arn:aws:elasticloadbalancing:region:123456789012:targetgroup/
target_group_name_1/1234567890123456",
      "containerName":"jenkins",
      "containerPort":8080
   },
   {

 "targetGroupArn":"arn:aws:elasticloadbalancing:region:123456789012:targetgroup/
target_group_name_2/6543210987654321",
      "containerName":"jenkins",
      "containerPort":50000
```

```
    }
]
```

公開來自多個容器的連接埠

在以下的使用案例，服務會使用一個負載平衡器和兩個目標群組，以公開不同容器中的連接埠。

```
"loadBalancers":[
   {

 "targetGroupArn":"arn:aws:elasticloadbalancing:region:123456789012:targetgroup/
target_group_name_1/1234567890123456",
      "containerName":"webserver",
      "containerPort":80
   },
   {

 "targetGroupArn":"arn:aws:elasticloadbalancing:region:123456789012:targetgroup/
target_group_name_2/6543210987654321",
      "containerName":"database",
      "containerPort":3306
   }
]
```

自動擴展 Amazon ECS 服務

自動擴展是自動增加或減少 Amazon ECS 服務中所需任務數量的功能。Amazon ECS 利用 Application Auto Scaling 服務來提供此功能。如需詳細資訊，請參閱 Application Auto Scaling 使用者指南。

Amazon ECS 會發佈具有您的服務平均 CPU 和記憶體用量的 CloudWatch 指標。如需詳細資訊，請參閱Amazon ECS 服務使用率指標。您可以使用這些指標和其他 CloudWatch 指標來向外擴展服務 (新增更多任務) 來處理尖峰時段的高需求量，以及將服務向內縮減 (執行較少任務) 以減少低使用率期間的成本。

Amazon ECS Service Auto Scaling 支援下列自動擴展類型：

- 使用目標指標來擴展 Amazon ECS 服務–根據特定指標的目標值，增加或減少您服務執行的任務數目。這與您運用電熱器維持家中溫度的方式很類似。您只要選取溫度，電熱器會自行執行其餘操作。
- 根據 CloudWatch 警示使用預先定義的增量來擴展 Amazon ECS 服務–根據一組依警示違規大小而變動的擴展調整 (稱為步驟調整)，增加或減少您服務執行的任務數目。

- 使用排程動作來擴展 Amazon ECS 服務- 根據日期和時間增加或減少服務執行的任務數量。
- 使用歷史模式，透過預測擴展來擴展 Amazon ECS 服務- 根據歷史負載資料分析增加或減少服務執行的任務數量，以偵測流量中的每日或每週模式。

考量事項

使用擴展策略時，考慮下列事項：

- Amazon ECS 每隔 1 分鐘會將指標傳送至 CloudWatch。在叢集和服務將指標傳送至 CloudWatch 前，指標皆無法使用，而且您也無法建立不存在之指標的 CloudWatch 警示。
- 擴展政策支援冷卻時間。等待先前擴展活動生效的秒數。
  - 對於向外擴展事件，其目的是連續的規模擴展 (但並非過度)。在 Service Auto Scaling 使用擴展政策成功擴展之後，就會開始計算冷卻時間。除非初始化較大的橫向擴展或冷卻時間結束，否則擴展政策不會再次增加所需的容量。在向外擴展冷卻期間有效時，由起始冷卻的向外擴展活動所增加的容量，將計入下次向外擴展活動所需的容量。
  - 對於向內縮減事件，其用意在於保守縮減以保護應用程式的可用性，所以冷卻時間過後才會開放縮減活動。不過，若在向內縮減冷卻時間另有警示起始了橫向擴展活動，Service Auto Scaling 則會立即橫向擴展目標。在這種情況下，向內擴展冷卻期間隨即停止，且不會完成。
- 服務排程器隨時會使用所需的計數，但只要您具有服務的有效擴展政策和警示，Service Auto Scaling 就可以變更您手動設定的所需計數。
- 如果服務的所需計數低於其最小容量值，且警示啟動向外擴展活動，則 Service Auto Scaling 會根據與警示相關聯的擴展政策，將所需的計數擴展到最小容量值，然後視需要繼續向外擴展。不過，向內擴展活動不會調整所需的計數，因為它已低於最小容量值。
- 如果服務的所需計數設定高於其最大容量值，且警示啟動活動擴展，則 Service Auto Scaling 會將所需計數擴展到最大容量值，然後根據與警示相關聯的擴展政策，視需要繼續擴展。不過，向外擴展活動不會調整所需的計數，因為它已高於最大容量值。
- 在擴展活動期間，服務中實際執行的任務計數是 Service Auto Scaling 用作起始點的值，而不是所需的計數。這應為處理容量。這可防止無法滿足的過度且失控的擴展，例如，有可能因容器執行個體資源不足而無法放置其他任務。如果容器執行個體容量稍後可用，則擱置的擴展活動可能會成功，接著在冷卻時間之後進行後續的擴展活動。
- 若希望您的任務計數在沒有待完成工作時縮減到零，將最小容量設定為 0。使用目標追蹤擴展政策，當實際容量為 0 而指標顯示有工作負載需求時，Service Auto Scaling 會在擴展前等待一個待傳送的資料點。在此情況下，它會向外擴展可能的數量下限做為起始點，然後以實際執行的任務計數為基礎繼續擴展。

- Application Auto Scaling 會在 Amazon ECS 部署正在進行時關閉縮減程序。不過，除非在部署期間暫停，否則向外擴展程序會繼續發生。如需詳細資訊，請參閱服務自動擴展和部署。
- 您有多個適用於 Amazon ECS 任務的 Application Auto Scaling 選項。目標追蹤是最容易使用的模式。這樣一來，您僅需要為指標設定目標值，例如 CPU 平均使用率。然後，自動縮放器會自動管理獲得該值所需的任務數量。使用步驟擴展，您可以更快對需求變化做出反應，因為您定義了擴展指標的特定閾值，以及超越閾值時要新增或刪除的任務數量。而且，更重要的是，您可以透過最大限度減少閾值警報違反的時間來對需求變化做出非常快速的反應。

最佳化 Amazon ECS 服務自動擴展

Amazon ECS 服務是受管任務集合。每個服務都有相關聯的任務定義、所需的任務計數，以及選用的置放策略。Amazon ECS 服務自動擴展是透過 Application Auto Scaling 服務實作。Application Auto Scaling 使用 CloudWatch 指標作為擴展指標的來源。它也會使用 CloudWatch 警示來設定何時縮減服務的閾值。您可以透過設定指標目標、稱為目標追蹤擴展或指定閾值，稱為步驟擴展，來提供擴展的閾值。設定 Application Auto Scaling 之後，它會持續計算服務的適當所需任務計數。它也會通知 Amazon ECS 所需的任務計數何時應變更，方法是向外擴展或向內擴展。

若要有效使用服務自動擴展，您必須選擇適當的擴展指標。

如果預測需求大於目前容量，應用程式應橫向擴展。相反地，應用程式可以縮減規模，以便在資源超過需求時節省成本。

識別指標

若要有效擴展，請務必識別表示使用率或飽和度的指標。此指標必須顯示下列屬性，才能有助於擴展。

- 指標必須與需求相關聯。當資源保持穩定，但需求變更時，指標值也必須變更。當需求增加或減少時，指標應該增加或減少。
- 指標值必須按比例擴展至容量。當需求保持恆定時，新增更多資源必須在指標值中產生比例變更。因此，將任務數量加倍應該會導致指標減少 50%。

識別使用率指標的最佳方法是在預備生產環境中進行負載測試，例如預備環境。商業和開放原始碼負載測試解決方案已廣泛推出。這些解決方案通常可以產生合成負載或模擬實際使用者流量。

若要開始負載測試程序，請為應用程式的使用率指標建置儀表板。這些指標包括 CPU 使用率、記憶體使用率、I/O 操作、I/O 佇列深度和網路輸送量。您可以使用 Container Insights 等服務收集這些指標。如需詳細資訊，請參閱使用具有增強可觀測性的 Container Insights 監控 Amazon ECS 容器。在此過程中，請確定您收集並繪製應用程式回應時間或工作完成率的指標。

從小請求或任務插入率開始。保持此速率穩定幾分鐘，讓您的應用程式暖機。然後，緩慢提高速率並保持穩定幾分鐘。重複此週期，每次增加速率，直到應用程式的回應或完成時間太慢而無法滿足您的服務層級目標 (SLOs)。

進行負載測試時，請檢查每個使用率指標。隨負載增加的指標是做為最佳使用率指標的首要候選者。

接著，識別達到飽和的資源。同時， 也會檢查使用率指標，以查看哪個指標先在高層級上水平化，或達到峰值，然後先當機您的應用程式。例如，如果 CPU 使用率在您新增負載時從 0% 增加到 70-80%，則在新增更多負載之後， 會保持在該層級，則表示 CPU 飽和是安全的。根據 CPU 架構，它可能永遠不會達到 100%。例如，假設記憶體使用率隨著您新增負載而增加，然後您的應用程式在達到任務或 Amazon EC2 執行個體記憶體限制時突然當機。在這種情況下，記憶體可能已完全耗盡。您的應用程式可能會使用多個資源。因此，請選擇代表先耗盡資源的指標。

最後，在任務或 Amazon EC2 執行個體數量加倍之後，請嘗試再次載入測試。假設索引鍵指標以之前一半的速率增加或減少。如果是這種情況，則指標與容量成比例。這是自動擴展的良好使用率指標。

現在請考量這個假設案例。假設您載入測試應用程式，並發現 CPU 使用率最終達到每秒 100 個請求的 80%。新增更多負載時，不會再增加 CPU 使用率。不過，它確實會讓您的應用程式回應速度變慢。然後，您再次執行負載測試，將任務數量加倍，但將速率保持在其先前的峰值。如果您發現平均 CPU 使用率下降到約 40%，則平均 CPU 使用率是擴展指標的良好候選項目。另一方面，如果 CPU 使用率在增加任務數量後仍維持在 80%，則平均 CPU 使用率不是良好的擴展指標。在這種情況下，需要更多研究來尋找合適的指標。

常見的應用程式模型和擴展屬性

所有類型的軟體都會在 上執行 AWS。許多工作負載是自家生產的，而其他工作負載則是以熱門的開放原始碼軟體為基礎。無論它們的來源為何，我們觀察到一些常見的服務設計模式。如何有效擴展在很大程度上取決於模式。

高效率的 CPU 繫結伺服器

有效率的 CPU 繫結伺服器幾乎不使用 CPU 和網路輸送量以外的資源。每個請求都可以由應用程式單獨處理。請求不依賴其他 服務，例如資料庫。應用程式可以處理數十萬個並行請求，並且可以有效地利用多個 CPUs來執行此操作。每個請求都由記憶體負荷低的專用執行緒進行服務，或者有非同步事件迴圈在服務請求的每個 CPU 上執行。應用程式的每個複本都同樣能夠處理請求。在 CPU 之前可能耗盡的唯一資源是網路頻寬。在 CPU 繫結服務中，記憶體使用率，即使是尖峰輸送量，也是可用資源的一小部分。

這種類型的應用程式適用於以 CPU 為基礎的自動擴展。應用程式在擴展方面享有最大的彈性。它可以透過提供較大的 Amazon EC2 執行個體或 Fargate vCPUs來垂直擴展。而且，它也可以透過新增更多

複本來水平擴展。新增更多複本，或將執行個體大小加倍，會將相對於容量的平均 CPU 使用率減少一半。

如果您為此應用程式使用 Amazon EC2 容量，請考慮將其放在運算最佳化的執行個體上，例如 c5或 c6g 系列。

高效率的記憶體繫結伺服器

高效率的記憶體繫結伺服器會在每次請求配置大量記憶體。在並行上限，但不一定是輸送量時，記憶體會在 CPU 資源耗盡之前耗盡。與請求相關聯的記憶體會在請求結束時釋出。只要有可用的記憶體，就可以接受其他請求。

這種類型的應用程式適用於記憶體型自動擴展。應用程式在擴展方面享有最大的彈性。它可以透過提供較大的 Amazon EC2 或 Fargate 記憶體資源，以垂直方式擴展。而且，它也可以透過新增更多複本來水平擴展。新增更多複本，或將執行個體大小加倍，可以將相對於容量的平均記憶體使用率減少一半。

如果您為此應用程式使用 Amazon EC2 容量，請考慮將其放在記憶體最佳化的執行個體上，例如 r5或 r6g 系列。

有些記憶體繫結應用程式不會在請求結束時釋放與請求相關聯的記憶體，因此減少並行不會導致使用的記憶體減少。因此，我們不建議您使用記憶體型擴展。

工作者型伺服器

工作者型伺服器會逐一處理每個個別工作者執行緒的請求。工作者執行緒可以是輕量型執行緒，例如 POSIX 執行緒。它們也可以是更重的執行緒，例如 UNIX 程序。無論它們是哪個執行緒，應用程式都可以支援的最大並行數。通常，並行限制會與可用的記憶體資源成比例設定。如果達到並行限制，其他請求會放入待處理項目佇列。如果待處理項目佇列溢位，會立即拒絕其他傳入的請求。符合此模式的常見應用程式包括 Apache Web 伺服器和 Gunicorn。

請求並行通常是擴展此應用程式的最佳指標。由於每個複本都有並行限制，因此請務必在達到平均限制之前向外擴展。

取得請求並行指標的最佳方法是讓您的應用程式向 CloudWatch 報告。您應用程式的每個複本都可以以高頻率將並行請求數發佈為自訂指標。建議將頻率設定為每分鐘至少一次。收集多個報告之後，您可以使用平均並行做為擴展指標。此指標的計算方式是將總並行數除以複本數量。例如，如果並行總數為 1000 且複本數量為 10，則平均並行數為 100。

如果您的應用程式位於 Application Load Balancer 之後，您也可以使用負載平衡器的 `ActiveConnectionCount`指標做為擴展指標的因素。`ActiveConnectionCount` 指標必須除以複本數量，才能取得平均值。平均值必須用於擴展，而不是原始計數值。

為了讓此設計發揮最佳效能，回應延遲的標準差應該在低請求率下很小。我們建議在低需求期間，大多數請求會在短時間內得到回應，而且沒有比平均回應時間更長的許多請求。平均回應時間應接近第 95 個百分位數的回應時間。否則，佇列溢位可能會因此發生。這會導致錯誤。我們建議您在必要時提供額外的複本，以降低溢位的風險。

等待伺服器

等待伺服器會為每個請求進行一些處理，但高度依賴一或多個下游服務來運作。容器應用程式通常會大量使用下游服務，例如資料庫和其他 API 服務。這些服務可能需要一些時間才能回應，特別是在高容量或高並行情況下。這是因為這些應用程式傾向於使用很少的 CPU 資源，並在可用的記憶體方面利用其最大並行。

等待服務適合記憶體繫結伺服器模式或工作者型伺服器模式，視應用程式的設計方式而定。如果應用程式的並行僅受限於記憶體，則平均記憶體使用率應用作擴展指標。如果應用程式的並行是基於工作者限制，則應使用平均並行做為擴展指標。

以 Java 為基礎的伺服器

如果您的 Java 型伺服器與 CPU 繫結，並依 CPU 資源比例擴展，則可能適合有效率的 CPU 繫結伺服器模式。如果是這種情況，平均 CPU 使用率可能適合做為擴展指標。不過，許多 Java 應用程式都不受 CPU 限制，因此難以擴展。

為了獲得最佳效能，建議您盡可能將記憶體配置到 Java Virtual Machine (JVM) 堆積。JVM 的最近版本，包括 Java 8 更新 191 或更新版本，會自動將堆積大小設定為盡可能大，以符合容器內。這表示在 Java 中，記憶體使用率很少與應用程式使用率成比例。隨著請求率和並行增加，記憶體使用率保持不變。因此，我們不建議根據記憶體使用率來擴展 Java 型伺服器。反之，我們通常會建議擴展 CPU 使用率。

在某些情況下，Java 型伺服器在耗盡 CPU 之前會遇到堆積耗盡。如果您的應用程式容易在高並行時堆積耗盡，則平均連線是最佳的擴展指標。如果您的應用程式容易以高輸送量堆積耗盡，則平均請求率是最佳的擴展指標。

使用其他垃圾收集執行時間的伺服器

許多伺服器應用程式是以執行垃圾收集的執行時間為基礎，例如 .NET 和 Ruby。這些伺服器應用程式可能符合上述其中一種模式。不過，與 Java 一樣，我們不建議根據記憶體擴展這些應用程式，因為他們觀察到的平均記憶體使用率通常與輸送量或並行無關。

對於這些應用程式，如果應用程式受 CPU 限制，我們建議您擴展 CPU 使用率。否則，我們建議您根據您的負載測試結果，擴展平均輸送量或平均並行。

任務處理器

許多工作負載涉及非同步任務處理。其中包括未即時接收請求的應用程式，而是訂閱工作佇列以接收任務。對於這些類型的應用程式，適當的擴展指標幾乎一律是佇列深度。佇列增長表示待定的工作超出處理容量，而空佇列表示容量比工作多。

AWS 訊息服務，例如 Amazon SQS 和 Amazon Kinesis Data Streams，提供可用於擴展的 CloudWatch 指標。對於 Amazon SQS， `ApproximateNumberOfMessagesVisible`是最佳指標。對於 Kinesis Data Streams，請考慮使用 Kinesis Client Library (KCL) 發佈的 `MillisBehindLatest` 指標。使用此指標進行擴展之前，應先在所有消費者之間進行平均。

服務自動擴展和部署

Application Auto Scaling 會在 Amazon ECS 部署正在進行時關閉縮減程序。不過，除非在部署期間暫停，否則向外擴展程序會繼續發生。如果您想要在部署正在進行時暫停向外擴展程序，請執行下列步驟。

1. 呼叫 describe-scalable-targets 命令，指定在 Application Auto Scaling 中與可擴展目標關聯的服務資源 ID (範例：`service/default/sample-webapp`)。記錄輸出。您將在呼叫下一個命令時用到它。
2. 呼叫 register-scalable-target 命令，指定資源 ID、命名空間和可擴展維度。同時為 `DynamicScalingInSuspended` 和 `DynamicScalingOutSuspended` 指定 `true`。
3. 部署完成後，您可以呼叫 register-scalable-target 命令以繼續擴展。

如需詳細資訊，請參閱暫停和繼續擴展 Application Auto Scaling。

使用目標指標來擴展 Amazon ECS 服務

使用目標追蹤擴展政策，您可以選取指標及設定目標值。Amazon ECS Service Auto Scaling 會建立和管理可控制擴展政策的 CloudWatch 警示，並根據指標和目標值來計算擴展調整。擴展政策會視需要新增或移除服務任務，以維持等於或接近指定目標值的指標。除了維持接近目標值的指標之外，目標追蹤擴展政策也會因波動負載模式調整為指標中的波動，以及將服務中執行之數個任務中的快速波動降到最低。

目標追蹤政策無需手動定義 CloudWatch 警示和擴展調整內容，Amazon ECS 會根據您設定的目標自動處理此問題。

使用目標追蹤政策時，請考慮下列事項：

- 目標追蹤擴展政策假設在指定的指標超過目標值時，應執行向外擴展。您無法使用目標追蹤擴展政策在指定的指標低於目標值時執行向外擴展。
- 所指定指標的資料不足時，目標追蹤擴展政策不會執行擴展。政策不會執行向內擴展，因為向內擴展不會將資料不足解釋為低使用率。
- 您可能會看到目標值與實際指標資料點之間有些差距。原因是 Service Auto Scaling 在決定新增或移除多少容量時，一律以四捨五入來保守處理。這樣可防止新增不足的容量，或移除過多的容量。
- 為了確保應用程式可用性，服務可以按比例快速地向外擴展到指標，但需漸漸地逐步向內擴展。
- Application Auto Scaling 會在 Amazon ECS 部署正在進行時關閉縮減程序。不過，除非在部署期間暫停，否則向外擴展程序會繼續發生。如需詳細資訊，請參閱服務自動擴展和部署。
- 您可以擁有 Amazon ECS 服務的多個目標追蹤擴展政策，但前提是每個政策都使用不同的指標。Service Auto Scaling 的用意一律以可用性為優先，因此其行為視目標追蹤政策是準備水平擴展或縮減而有所不同。如果任何目標追蹤政策已準備好橫向擴展，其就會將服務橫向擴展，但只有在所有目標追蹤政策 (已開啟縮減部分) 都已準備好要縮減時才會縮減。
- 請勿編輯或刪除 CloudWatch 警示，Service Auto Scaling 管理它們用於目標追蹤擴展政策。當您刪除擴展政策時，Service Auto Scaling 會自動刪除警示。
- 藍/綠部署類型不支援目標追蹤擴展政策的 `ALBRequestCountPerTarget` 指標。

如需有關目標追蹤擴展政策的詳細資訊，請參閱《Application Auto Scaling 使用者指南》中的目標追蹤擴展政策。

建立 Amazon ECS 服務自動擴展的目標追蹤擴展政策

建立目標追蹤擴展政策，讓 Amazon ECS 自動增加或減少服務中所需的任務計數。目標追蹤的運作與目標指標值不同。

主控台

1. 除了建立和更新服務的標準 IAM 許可之外，還需要額外的許可。如需詳細資訊，請參閱Amazon ECS 服務自動擴展所需的 IAM 許可。
2. 決定要用於政策的指標。下列指標可供使用：
   - ECSServiceAverageCPUUtilization – 服務應使用的平均 CPU 使用率。
   - ECSServiceAverageMemoryUtilization – 服務應使用的平均記憶體使用率。
   - ALBRequestCountPerTarget – 任務理想情況下應接收的每分鐘平均請求數。
3. 開啟主控台，網址為 https://console.aws.amazon.com/ecs/v2。

4. 在叢集頁面上，選擇叢集。
5. 在叢集詳細資訊頁面上的服務區段中，然後選擇服務。

   服務詳細資訊頁面隨即出現。
6. 選擇設定任務數量。
7. 在 Amazon ECS 服務任務計數下，選擇使用自動擴展。

   任務計數區段隨即出現。

   a. 對於任務數量下限，輸入服務自動擴展要使用的任務數量下限。所需的計數不會低於此計數。

   b. 針對最大值，輸入服務自動擴展要使用的任務數量上限。所需的計數不會高於此計數。

   c. 選擇 Save (儲存)。

      政策頁面隨即出現。
8. 選擇建立擴展政策。

   隨即顯示建立政策頁面。
9. 針對 Scaling policy type (擴展政策類型)，選擇 Target tracking (目標追蹤)。
10. 針對 Policy name (政策名稱)，輸入政策的名稱。
11. 對於指標類型，請從選項清單中選擇您的指標。
12. 針對目標使用率，輸入 Amazon ECS 應維護之任務百分比的目標值。服務自動擴展會擴展您的容量，直到平均使用率達到目標使用率，或達到您指定的任務數量上限。
13. 在其他設定下，執行下列動作

    a. 針對縮減冷卻時間，輸入縮減活動完成後，另一個縮減活動開始之前的秒數。

    b. 針對橫向擴展冷卻時間，輸入等待先前的橫向擴展活動生效的時間，以秒為單位。

    c. 若要僅建立橫向擴展政策，請選取停用橫向擴展。
14. 選擇建立擴展政策。

AWS CLI

1. 使用 register-scalable-target 命令，將 Amazon ECS 服務註冊為可擴展性目標。
2. 使用 put-scaling-policy 命令，來建立擴展政策。

根據 CloudWatch 警示使用預先定義的增量來擴展 Amazon ECS 服務

如果使用步進擴展政策，您可以建立及管理觸發擴展程序的 CloudWatch 警示。違反警示時，Amazon ECS 會啟動與該警示相關聯的擴展政策。步驟擴展政策會使用一組調整來擴展任務，稱為步驟調整。調整幅度會依流量達到警示閾值的程度而有所不同。

- 如果違規超過第一個閾值，Amazon ECS 會套用第一個步驟調整。
- 如果違規超過第二個閾值，Amazon ECS 會套用第二個步驟調整，以此類推。

我們強烈建議您使用目標追蹤擴展政策來擴展指標，例如平均 CPU 使用率或每個目標的平均請求計數。當容量增加時減少的指標，以及當容量減少時增加的指標，可用於使用目標追蹤按比例擴展或增加任務數量。這有助於確保 Amazon ECS 緊密遵循您應用程式的需求曲線。

建立 Amazon ECS 服務自動擴展的步驟擴展政策

建立步驟擴展政策，讓 Amazon ECS 自動增加或減少服務中所需的任務數量。步驟擴展會根據一組規模調整執行，稱為步驟調整，這些調整會根據警示違規的大小而有所不同。

主控台

1. 除了建立和更新服務的標準 IAM 許可之外，還需要額外的許可。如需詳細資訊，請參閱Amazon ECS 服務自動擴展所需的 IAM 許可。
2. 決定要用於政策的指標。下列指標可供使用：
   - ECSServiceAverageCPUUtilization – 服務應使用的平均 CPU 使用率。
   - ECSServiceAverageMemoryUtilization – 服務應使用的平均記憶體使用率。
   - ALBRequestCountPerTarget – 任務理想情況下應接收的每分鐘平均請求數。
3. 建立指標的 CloudWatch 警示。如需詳細資訊，請參閱 Amazon CloudWatch 使用者指南中的建立以靜態閾值為基礎的 CloudWatch 警示。
4. 開啟主控台，網址為 https://console.aws.amazon.com/ecs/v2。
5. 在叢集頁面上，選擇叢集。
6. 在叢集詳細資訊頁面上的服務區段中，然後選擇服務。

   服務詳細資訊頁面隨即出現。
7. 選擇設定任務數量。
8. 在 Amazon ECS 服務任務計數下，選擇使用自動擴展。

   任務計數區段隨即出現。

a. 針對任務數量下限，輸入服務自動擴展要使用的任務數量下限。所需的計數不會低於此計數。
b. 針對最大值，輸入服務自動擴展要使用的任務數量上限。所需的計數不會高於此計數。
c. 選擇 Save (儲存)。

政策頁面隨即出現。

9. 選擇建立擴展政策。

隨即顯示建立政策頁面。

10. 針對擴展政策類型，選擇步驟擴展。
11. 設定向外擴展屬性。在新增任務的步驟下，執行下列動作：

a. 針對 Policy name (政策名稱)，輸入政策的名稱。
b. 針對 CloudWatch 警示名稱，選擇 CloudWatch 警示。
c. 針對指標彙總類型，選擇如何比較選取的指標與定義的閾值。
d. 對於調整類型，選擇調整是基於任務數量的變更，還是任務百分比的變更。
e. 對於要採取的動作，輸入要採取的動作的值。

選擇新增步驟以新增其他動作。

12. 設定縮減屬性。在移除任務的步驟下，執行下列動作：

a. 針對 Policy name (政策名稱)，輸入政策的名稱。
b. 針對 CloudWatch 警示名稱，選擇 CloudWatch 警示。
c. 針對指標彙總類型，選擇如何比較選取的指標與定義的閾值。
d. 對於調整類型，選擇調整是基於任務數量的變更，還是任務百分比的變更。
e. 針對要採取的動作，輸入要採取的動作值。

選擇新增步驟以新增其他動作。

13. 針對冷卻時間，以秒為單位輸入等待先前擴展活動生效的時間量。對於新增政策，這是擴展政策封鎖縮減活動，並限制一次可以縮減任務數量之後的時間。對於移除政策，這是在另一個縮減活動開始之前必須通過的縮減活動之後的時間。
14. 選擇建立擴展政策。

AWS CLI

1. 使用 register-scalable-target 命令，將 Amazon ECS 服務註冊為可擴展性目標。

2. 使用 put-scaling-policy 命令，來建立擴展政策。

使用排程動作來擴展 Amazon ECS 服務

透過排程擴展，您可以建立排程動作來增加或減少特定時間的任務數量，藉此根據可預測的負載變更來設定應用程式的自動擴展。如此一來，您便能主動調整應用程式規模，以符合負載變動預測。

藉由這些排定的擴展動作，您可將費用和效能調整到最佳狀態。您的應用程式有足夠數量的任務來處理週中流量峰值，但不會在其他時間過度佈建任務數量。

您可以搭配利用排程擴展和擴展政策，以主動和被動的方式處理規模擴展作業，同時享有這兩種方法的好處。排程擴展動作執行後，擴展政策可以繼續決定是否進一步擴展任務數量。這可協助您確保您有足夠的任務數量來處理應用程式的負載。當您的應用程式擴展以符合需求時，目前的容量必須落在排程動作所設定的任務數量下限和上限內。

您可以使用 設定排程擴展 AWS CLI。如需排程擴展的詳細資訊，請參閱《Application Auto Scaling 使用者指南》中的排程擴展。 Auto Scaling

建立 Amazon ECS 服務自動擴展的排程動作

建立排程動作，讓 Amazon ECS 根據日期和時間增加或減少服務執行的任務數量。

主控台

1. 開啟主控台，網址為 https://console.aws.amazon.com/ecs/v2。
2. 在叢集頁面上，選擇叢集。
3. 在叢集詳細資訊頁面上的服務區段中，選擇服務。

   服務詳細資訊頁面隨即出現。
4. 選擇服務自動擴展。

   服務自動擴展頁面隨即出現。
5. 如果您尚未設定服務自動擴展，請選擇設定任務數量。

   Amazon ECS 服務任務計數區段隨即出現。

   在 Amazon ECS 服務任務計數下，選擇使用服務自動擴展來調整服務所需的任務計數。

   任務計數區段隨即出現。

   a. 對於任務數量下限，輸入服務自動擴展要使用的任務數量下限。所需的計數不會低於此計數。

b. 針對最大值，輸入服務自動擴展要使用的任務數量上限。所需的計數不會高於此計數。

c. 選擇選擇儲存。

政策頁面隨即出現。

6. 選擇排程動作，然後選擇建立。

隨即顯示建立排程動作頁面。

7. 針對動作名稱，輸入唯一的名稱。

8. 對於 Time zone (時區)，選擇時區。

所有列出的時區都來自 IANA 時區資料庫。如需詳細資訊，請參閱 tz 資料庫時區清單。

9. 針對開始時間，輸入動作開始的日期和時間。

如果選擇週期性排程，開始時間會定義週期性序列中第一個排程作業的執行時間。

10. 針對 Recurrence (週期)，選擇其中一個可用選項。

- 若要根據週期性排程進行擴展，請選擇 Amazon ECS 執行排程動作的頻率。
  - 如果您選擇以 Rate 開頭的選項，則會為您建立 Cron 表達式。
  - 如果選擇 Cron，請輸入指定何時執行動作的 Cron 表達式。
- 若要僅擴展一次，請選擇一次。

11. 在任務調整下，執行下列動作：

- 針對最小值，輸入服務應執行的任務數量下限。
- 針對最大值，輸入服務應執行的任務數量上限。

12. 選擇建立排程動作。

CLI

使用 AWS CLI 設定服務的排程擴展政策，如下所示。將每個*########*替換為自己的資訊。

範例：僅擴展一次

使用下列 put-scheduled-action 命令搭配 `--start-time "YYYY-MM-DDThh:mm:ssZ"`和 以及 `--MinCapacity`和 `--MaxCapacity`選項之一或兩者。

```
aws application-autoscaling put-scheduled-action --service-namespace ecs \
  --resource-id service/my-cluster/my-service \
```

```
  --scheduled-action-name my-one-time-schedule \
  --start-time 2021-01-30T12:00:00 \
  --scalable-target-action MinCapacity=3,MaxCapacity=10
```

範例：依週期性排程排程擴展

使用下列 put-scheduled-action 命令。將#####取代為您的值。

```
aws application-autoscaling put-scheduled-action --service-namespace ecs \
  --resource-id service/my-cluster/my-service \
  --scheduled-action-name my-recurring-action \
  --schedule "rate(5 hours)" \
  --start-time 2021-01-30T12:00:00 \
  --end-time 2021-01-31T22:00:00 \
  --scalable-target-action MinCapacity=3,MaxCapacity=10
```

指定的循環排程會根據 UTC 時區執行。若要指定不同的時區，請包含 `--time-zone`選項和 IANA 時區的名稱，如下列範例所示。

```
--time-zone "America/New_York"
```

如需詳細資訊，請參閱 tz 資料庫時區清單。

使用歷史模式，透過預測擴展來擴展 Amazon ECS 服務

預測擴展會查看過去從流量流程中載入的資料，以分析每日或每週模式。然後，它會使用此分析來預測未來需求，並視需要主動增加服務中的任務。

在下列情況中，預測性自動擴展最有用。

- 週期性流量 - 在正常上班時間增加資源使用量，並在晚上和週末減少資源使用量。
- 定期on-and-off工作負載模式 - 範例包括批次處理、測試或定期資料分析。
- 初始化時間較長的應用程式 - 這可能會影響造成明顯延遲的橫向擴展事件期間的應用程式效能。

如果您的應用程式需要很長的時間初始化，且流量以一般模式增加，您應該考慮使用預測擴展。它透過主動增加預測負載的任務數量，而不是使用動態擴展政策，例如單獨追蹤目標或步進擴展，來協助您更快地擴展。透過協助您避免過度佈建任務數量的可能性，預測擴展也可能為您節省成本。

例如，考量應用程式在營業時間內具有高使用率而在夜間具有低使用率。在每個工作日開始時，預測擴展可以在流量第一次湧入之前擴展任務。在從較低的使用率期間到較高的使用率期間時，這可協助您的

應用程式維持高可用性和效能。您不必等待動態擴展來對不斷變化的流量做出反應。您也不必花時間檢閱應用程式的載入模式，並嘗試使用排程擴展來排程正確數量的任務。

目錄

了解預測擴展如何在 Amazon ECS 中運作

在這裡，您可以了解使用預測擴展的考量、運作方式，以及限制。

使用預測擴展的考量

- 您想要確保預測擴展適用於您的工作負載。您可以在僅限預測模式下設定擴展政策，並查看 主控台的建議，藉此檢查這一點。您應該先評估預測和建議，再開始使用預測擴展。
- 在預測擴展可以開始預測之前，它需要至少 24 小時的歷史資料。可用的歷史資料越多，預測效果就越有效，兩週是最理想的。您還需要等待 24 小時，預測擴展才能在刪除 Amazon ECS 服務並建立新的預測時產生新的預測。加速此速度的其中一個方法是使用自訂指標，在新舊 Amazon ECS 服務之間彙總指標。
- 選擇準確代表應用程式完全負載的負載指標，是應用程式擴展最重要的層面。
- 具有預測擴展的動態擴展可協助您密切遵循應用程式的需求，因此您可以在霉期間進行擴展，並在意外流量增加期間進行擴展。當多個擴展政策處於作用中狀態時，每個政策會獨立決定所需的任務數量，且所需的任務數量會設為這些任務的最大值。
- 您可以使用預測擴展搭配動態擴展政策，例如目標追蹤或步進擴展，讓您的應用程式根據即時和歷史模式進行擴展。預測擴展本身不會縮減您的任務。
- 如果您在呼叫 `register-scalable-target` API 時使用自訂角色，您可能會收到錯誤，指出預測擴展政策只能在啟用 SLR 的情況下使用。在這種情況下，您應該`register-scalable-target`再次呼叫 ，但不要使用 role-arn。註冊可擴展目標並呼叫 `put-scaling-policy` API 時使用 SLR。

預測擴展的運作方式

您可以透過建立預測擴展政策來使用預測擴展，指定要監控和分析的 CloudWatch 指標。預測擴展必須有至少 24 小時的資料，才能開始預測未來值。

建立政策之後，預測擴展會開始分析過去 14 天內的指標資料，以識別模式。此分析用於產生接下來 48 小時的需求每小時預測。最新的 CloudWatch 資料會每六小時用來更新預測。隨著新資料傳入，預測擴展會持續提高未來預測的準確性。

當您第一次啟用預測擴展時，它會以僅限預測模式執行。它會在此模式下產生預測，但不會根據這些預測來擴展 Amazon ECS 服務。這表示您可以評估預測的準確性和適用性。您可以使用 `GetPredictiveScalingForecast` API 操作或 檢視預測資料 AWS Management Console。

當您決定開始使用預測擴展時，請將擴展政策切換為預測和擴展模式。處於此模式時，會發生下列情況。

您的 Amazon ECS 服務會在每小時開始時根據該小時的預測進行擴展。您可以選擇在 `PutScalingPolicy` API 操作中使用 `SchedulingBufferTime` 屬性以提早開始。這使得新任務在預測需求之前啟動，並讓他們有時間開機並準備好處理流量。

任務上限

當您註冊 Amazon ECS 服務以進行擴展時，您可以定義每個服務可啟動的任務數量上限。根據預設，當設定擴展政策時，它們無法增加超過其上限的任務數量。

或者，您可以允許在預測接近或超過 Amazon ECS 服務的任務數量上限時，自動增加服務的任務數量上限。

> **Warning**
>
> 允許自動增加任務數量上限時，請小心。如果未監控和管理增加的任務數量上限，這可能會導致啟動的任務數量超過預期。增加的任務數量上限會變成 Amazon ECS 服務的新正常任務數量上限，直到您手動更新為止。任務數量上限不會自動減少回原始上限。

支援的 區域

- 美國東部 (維吉尼亞北部)
- 美國東部 (俄亥俄)
- 美國西部 (加利佛尼亞北部)
- 美國西部 (奧勒岡)
- 非洲 (開普敦)
- Asia Pacific (Hong Kong)
- 亞太區域 (雅加達)

- 亞太區域 (孟買)
- 亞太區域 (大阪)
- 亞太區域 (首爾)
- 亞太區域 (新加坡)
- 亞太區域 (雪梨)
- 亞太區域 (東京)
- 加拿大 (中部)
- 中國 (北京)
- 中國 (寧夏)
- 歐洲 (法蘭克福)
- 歐洲 (愛爾蘭)
- 歐洲 (倫敦)
- 歐洲 (米蘭)
- 歐洲 (巴黎)
- 歐洲 (斯德哥爾摩)
- 中東 (巴林)
- 南美洲 (聖保羅)
- AWS GovCloud （美國東部）
- AWS GovCloud （美國西部）

建立 Amazon ECS 服務自動擴展的預測擴展政策

建立預測擴展政策，讓 Amazon ECS 根據歷史資料增加或減少服務執行的任務數量。

> **Note**
>
> 新服務需要提供至少 24 小時的資料，才能產生預測。

主控台

1. 除了建立和更新服務的標準 IAM 許可之外，還需要額外的許可。如需詳細資訊，請參閱Amazon ECS 服務自動擴展所需的 IAM 許可。
2. 決定要用於政策的指標。下列指標可供使用：

- ECSServiceAverageCPUUtilization – 服務應使用的平均 CPU 使用率。
- ECSServiceAverageMemoryUtilization – 服務應使用的平均記憶體使用率。
- ALBRequestCountPerTarget – 任務理想情況下應接收的每分鐘平均請求數。

或者，您可以使用自訂指標。您需要定義下列值：

- Load - 準確代表應用程式完全載入的指標，是應用程式最重要的擴展層面。

- 擴展指標 - 最適合您應用程式使用率的最佳預測器。

3. 開啟主控台，網址為 https://console.aws.amazon.com/ecs/v2。
4. 在叢集頁面上，選擇叢集。
5. 在叢集詳細資訊頁面上的服務區段中，選擇服務。

   服務詳細資訊頁面隨即出現。
6. 選擇服務自動擴展，然後選擇設定任務數量。
7. 在 Amazon ECS 服務任務計數下，選擇使用自動擴展。

   任務計數區段隨即出現。

   a. 針對任務數量下限，輸入服務自動擴展要使用的任務數量下限。所需的計數不會低於此計數。
   b. 針對最大值，輸入服務自動擴展要使用的任務數量上限。所需的計數不會高於此計數。
   c. 選擇 Save (儲存)。

      政策頁面隨即出現。
8. 選擇建立擴展政策。

   隨即顯示建立政策頁面。
9. 針對擴展政策類型，選擇預測擴展。
10. 針對 Policy name (政策名稱)，輸入政策的名稱。
11. 對於指標對，從選項清單中選擇您的指標。

    如果選擇了 Application Load Balancer request count per target (每個目標的 Application Load Balancer 請求計數)，則在 Target group (目標群組) 中選擇目標群組。只有在您已連接服務的 Application Load Balancer 目標群組時，才支援每個目標的 Application Load Balancer 請求計數。 Application Load Balancer

如果您選擇自訂指標對，請從載入指標和擴展指標的清單中選擇個別指標。

12. 針對目標使用率，輸入 Amazon ECS 應維護之任務百分比的目標值。服務自動擴展會擴展您的容量，直到平均使用率達到目標使用率，或達到您指定的任務數量上限。
13. 選擇建立擴展政策。

AWS CLI

使用 AWS CLI 設定 Amazon ECS 服務的預測擴展政策，如下所示。將每個*########*替換為自己的資訊。

如需有關您可以指定之 CloudWatch 指標的詳細資訊，請參閱《Amazon EC2 Auto Scaling API 參考》中的 PredictiveScalingMetricSpecification。

範例 1：具有預先定義記憶體的預測擴展政策。

以下是具有預先定義記憶體組態的範例政策。

```
cat policy.json
{
    "MetricSpecifications": [
        {
            "TargetValue": 40,
            "PredefinedMetricPairSpecification": {
                "PredefinedMetricType": "ECSServiceMemoryUtilization"
            }
        }
    ],
    "SchedulingBufferTime": 3600,
    "MaxCapacityBreachBehavior": "HonorMaxCapacity",
    "Mode": "ForecastOnly"
}
```

下列範例說明使用指定的組態檔案執行 put-scaling-policy 命令來建立政策。

```
aws application-autoscaling put-scaling-policy \
--service-namespace ecs \
--region us-east-1 \
--policy-name predictive-scaling-policy-example \
--resource-id service/MyCluster/test \
--policy-type PredictiveScaling \
```

```
--scalable-dimension ecs:service:DesiredCount \
--predictive-scaling-policy-configuration file://policy.json
```

如果成功，此命令會傳回政策的 ARN。

```
{
    "PolicyARN": "arn:aws:autoscaling:us-
east-1:012345678912:scalingPolicy:d1d72dfe-5fd3-464f-83cf-824f16cb88b7:resource/ecs/
service/MyCluster/test:policyName/predictive-scaling-policy-example",
    "Alarms": []
}
```

範例 2：具有預先定義 CPU 的預測擴展政策。

以下是具有預先定義 CPU 組態的範例政策。

```
cat policy.json
{
    "MetricSpecifications": [
        {
            "TargetValue": 0.00000004,
            "PredefinedMetricPairSpecification": {
                "PredefinedMetricType": "ECSServiceCPUUtilization"
            }
        }
    ],
    "SchedulingBufferTime": 3600,
    "MaxCapacityBreachBehavior": "HonorMaxCapacity",
    "Mode": "ForecastOnly"
}
```

下列範例說明使用指定的組態檔案執行 put-scaling-policy 命令來建立政策。

```
aws aas put-scaling-policy \
--service-namespace ecs \
--region us-east-1 \
--policy-name predictive-scaling-policy-example \
--resource-id service/MyCluster/test \
--policy-type PredictiveScaling \
--scalable-dimension ecs:service:DesiredCount \
--predictive-scaling-policy-configuration file://policy.json
```

如果成功，此命令會傳回政策的 ARN。

```
{
    "PolicyARN": "arn:aws:autoscaling:us-
east-1:012345678912:scalingPolicy:d1d72dfe-5fd3-464f-83cf-824f16cb88b7:resource/ecs/
service/MyCluster/test:policyName/predictive-scaling-policy-example",
    "Alarms": []
}
```

評估 Amazon ECS 的預測擴展政策

在您使用預測擴展政策來擴展服務之前，請在 Amazon ECS 主控台中檢閱政策的建議和其他資料。這很重要，因為您不希望預測擴展政策在您知道其預測準確之前擴展實際容量。

如果服務是新的，請等待 24 小時建立第一個預測。

AWS 建立預測時，會使用歷史資料。如果您的服務尚未擁有最近的歷史資料，預測擴展可能會暫時以目前可用的歷史彙總建立的彙總來回填預測。預測會在政策建立日期前的兩週內回填。

檢視您的預測擴展建議

為了有效分析，服務自動擴展應該至少有兩個預測擴展政策要比較。(不過，您仍然可以檢閱單一政策的問題清單。) 建立多個政策時，您可以根據使用不同指標的政策，評估使用一個指標的政策。您也可以評估不同目標值和指標組合的影響。建立預測擴展政策後，Amazon ECS 會立即開始評估哪些政策可以更好地擴展您的群組。

在 Amazon ECS 主控台中檢視您的建議

1. 開啟主控台，網址為 https://console.aws.amazon.com/ecs/v2。
2. 在叢集頁面上，選擇叢集。
3. 在叢集詳細資訊頁面上的服務區段中，選擇服務。

   服務詳細資訊頁面隨即出現。
4. 選擇服務自動擴展。
5. 選擇預測擴展政策，然後選擇動作、預測擴展、檢視建議。

   您可以檢視政策的詳細資訊以及我們的建議。該建議會告訴您使用預測擴展政策的結果是否優於不使用它。

   如果您不確定預測擴展政策是否適合您的群組，請檢閱可用性影響和成本影響欄，以選擇正確的政策。每一欄的資訊都會說明政策的影響。

- 可用性影響：描述與不使用政策相比，政策是否透過佈建足夠的任務來處理工作負載，以避免對可用性造成負面影響。
- 成本影響：描述與不使用政策相比，該政策是否會透過不過度佈建任務來避免對您的成本造成負面影響。透過過度佈建過多，您的服務未充分利用或閒置，只會增加成本影響。

如果您有多個政策，則以較低成本提供最多可用性優勢的政策名稱旁會顯示最佳預測標籤。可用性影響會獲得更多加權。

6. （選用）若要選取建議結果的所需時段，請從評估時段下拉式清單中選擇您偏好的值：2 天、1 週或 2 週。根據預設，評估期間是最近兩週。較長的評估期間會為建議結果提供更多資料點。不過，如果負載模式發生變更 (例如在一段異常需求期間後)，新增更多資料點可能無法改善結果。在這種情況下，您可以查看最新資料以獲得更有針對性的建議。

> **Note**
>
> 只會針對處於僅預測模式的政策產生建議。當政策在整個評估期間都處於僅預測模式時，建議功能的效果會更好。如果您在預測和擴展模式中啟動政策，並於稍後將其切換至僅預測模式，則該政策的問題清單可能會有偏差。這是因為該政策已經為實際容量做出了貢獻。

檢閱預測擴展監控圖表

在 主控台中，您可以檢閱前幾天、前幾週或前幾個月的預測，以視覺化政策隨著時間的推移效能。您也可以在決定是否讓政策擴展您的實際任務數量時，使用此資訊來評估預測的準確性。

在 Amazon ECS 主控台中檢閱預測擴展監控圖表

1. 開啟主控台，網址為 https://console.aws.amazon.com/ecs/v2。
2. 在叢集頁面上，選擇叢集。
3. 在叢集詳細資訊頁面上的服務區段中，選擇服務。

   服務詳細資訊頁面隨即出現。
4. 選擇服務自動擴展。
5. 選擇預測擴展政策，然後選擇動作、預測擴展、檢視圖形。

6. 在監控區段中，您可以根據實際值檢視政策在過去和未來的負載和容量預測。負載圖表會顯示所選負載指標的負載預測與實際值。容量圖表顯示政策預測的任務數量。它還包括實際啟動的任務數量。垂直線會將歷史值與未來預測隔開。建立政策後，這些圖表很快就可以使用。
7. (選用) 若要變更圖表中顯示的歷史資料量，請從頁面頂端的評估期間下拉式清單中選擇您偏好的值。評估期間不會以任何方式轉換此頁面上的資料。它只會變更顯示的歷史資料量。

比較負載圖表中的資料

每條水平線代表每間隔一小時報告的一組不同資料點：

1. 實際觀察到的負載會使用所選負載指標的 SUM 統計資料來顯示過去的每小時總負載。
2. 政策預測的負載會顯示每小時的負載預測。此預測是基於前兩週的實際負載觀察結果。

比較容量圖表中的資料

每條水平線代表每間隔一小時報告的一組不同資料點：

1. 實際觀察到的任務數量會顯示您過去的 Amazon ECS 服務實際容量，這取決於您其他擴展政策和所選期間內有效的最小群組大小。
2. 政策預測的容量會顯示政策處於預測和擴展模式時，可預期在每小時開始時獲得的基準容量。
3. 推斷的必要任務數量會顯示您服務中將擴展指標維持在您所選目標值的理想任務數量。
4. 最低任務數量顯示您服務中最低任務數量。
5. 最大容量顯示服務中的任務數量上限。

為了計算推斷的所需容量，我們首先假設每個任務在指定的目標值上都平均使用。實際上，任務數量不會平均使用。不過，假設使用率在任務之間均勻分散，我們可以估算所需的容量量。然後，任務數量的需求計算為與您用於預測擴展政策的擴展指標成反比。換言之，隨著任務數量的增加，擴展指標會以相同的速率減少。例如，如果任務數量加倍，擴展指標必須減少一半。

推斷的所需容量公式：

```
sum of (actualServiceUnits*scalingMetricValue)/(targetUtilization)
```

例如，我們使用特定一小時的 `actualServiceUnits` (10) 和 `scalingMetricValue` (30)。然後，我們會使用您在預測擴展政策中指定的 `targetUtilization` (60)，並計算同一小時內推斷的所需容量。這會傳回值 5。這表示 5 是維持容量與擴展指標目標值正好成反比所需的推斷容量。

> **Note**
>
> 您可以使用各種控制桿來調整和改善應用程式的成本節省效益和可用性。
>
> - 您可以針對基準容量使用預測擴展，並使用動態擴展來處理額外的容量。動態擴展會與預測擴展分開運作，可根據目前的使用率進行縮減和擴增。首先，Amazon ECS 會計算每個非排程擴展政策的建議任務數量。然後，它會根據提供最多任務數量的政策進行擴展。
> - 若要允許在負載減少時發生縮減，您的服務應一律至少有一個啟用縮減部分的動態擴展政策。
> - 您可以確保您的最小和最大容量沒有太大限制，以提高擴展效能。建議任務數量不在最小和最大容量範圍內的政策，將無法向內和向外擴展。

使用 CloudWatch 監控 Amazon ECS 的預測擴展指標

您可以使用 Amazon CloudWatch 來監控資料以進行預測擴展。預測擴展政策會收集用於預測未來負載的資料。收集的資料會定期自動儲存在 CloudWatch 中，並可用於視覺化政策隨時間執行的效能。您也可以建立 CloudWatch 警示，以便在效能指標超出您定義的限制時通知您。

視覺化歷史預測資料

您可以在 CloudWatch 中檢視預測擴展政策的負載預測資料，而且在單一圖形中將預測與其他 CloudWatch 指標的預測視覺化時非常有用。您也可以透過檢視更廣泛的時間範圍來查看一段時間內的趨勢。您可以存取長達 15 個月的歷史指標，以更加了解政策的執行狀況。

使用 CloudWatch 主控台檢視歷史預測資料

1. 透過 https://console.aws.amazon.com/cloudwatch/ 開啟 CloudWatch 主控台。
2. 在導覽窗格中，選擇 Metrics (指標)，然後選擇 All metrics (所有指標)。
3. 選擇 Application Auto Scaling 指標命名空間。
4. 選擇預測擴展負載預測。
5. 在搜尋欄位中，輸入預測擴展政策的名稱或 Amazon ECS 服務群組的名稱，然後按 Enter 篩選結果。
6. 若要將指標圖形化，請勾選指標旁的核取方塊。若要變更圖形的名稱，請選擇鉛筆圖示。若要變更時間範圍，請選取一個預先定義的值，或選擇 custom (自訂)。如需詳細資訊，請參閱《Amazon CloudWatch 使用者指南》中的建立指標圖表。

7. 若要變更統計數字，請選擇 Graphed metrics (圖表化指標) 索引標籤。選擇欄位標題或個別的值，然後選擇不同的統計資料。雖然您可以為每個指標選擇任何統計資料，但並非所有統計資料都對 PredictiveScalingLoadForecast 指標有用。例如，Average (平均值)、Minimum (最小值) 和 Maximum (最大值) 統計資料有用，但是 Sum (總和) 統計資料無用。
8. 若要將其他指標新增到圖表，請在 Browse (瀏覽) 下，選擇 All (所有)，找到特定指標，然後選取旁邊的核取方塊。您最多可新增 10 個指標。
9. (選用) 若要將圖表新增至 CloudWatch 儀表板，請選擇 Actions (動作)、Add to dashboard (新增至儀表板)。

使用指標數學建立準確度指標

利用指標數學，可查詢多個 CloudWatch 指標，並使用數學表達式根據這些指標來建立新的時間序列。您可以在 CloudWatch 主控台視覺化產生的時間序列，並將其新增至儀表板。如需有關指標數學的詳細資訊，請參閱《Amazon CloudWatch 使用者指南》中的使用指標數學。

您可以使用指標數學，以不同方式繪製服務自動擴展產生用於預測擴展的資料圖表。這可協助您監控一段時間內的政策績效，並協助您瞭解是否可以改善指標組合。

例如，您可以使用指標數學表達式來監控平均絕對誤差百分比 (MAPE)。MAPE 指標有助於監控預測值與指定預測期間觀察到的實際值之間的差異。MAPE 值的變化可指示政策的績效是否隨著應用程式性質的變化在一段時間內而降低。MAPE 的增加表示預測值與實際值之間的差距更大。

範例：指標數學表達式

如果要開始使用此類圖表，您可以建立類似於以下範例中所示的指標數學表達式。

`MetricDataQueries` 有一個指標資料查詢結構陣列,而不是單一的指標。`MetricDataQueries` 中的每個項目都會取得指標或執行數學表達式。第一項，`e1`，是數學表達式。指定的表達式將 `ReturnData` 參數設定為 `true`，最終產生單一時間序列。對於所有其他指標，`ReturnData` 值為 `false`。

在此範例中，指定的表達式會使用實際值和預測值作為輸入值，並傳回新指標 (MAPE)。`m1` 是包含實際負載值的 CloudWatch 指標 (假設 CPU 使用率是針對名為 `my-predictive-scaling-policy` 的政策最初指定的負載指標)。`m2` 是包含預測負載值的 CloudWatch 指標。MAPE 指標的數學語法如下：

Average of (abs ((Actual - Forecast)/(Actual)))

視覺化您的準確度指標並設定警示

若要視覺化準確度指標資料，請選取 CloudWatch 主控台中的 Metrics (指標) 索引標籤。您可以從那裡繪製資料圖表。如需詳細資訊，請參閱《Amazon CloudWatch 使用者指南》中的將數學表達式新增到 CloudWatch 圖表。

您也可以在 Metrics (指標) 區段中，對您監控的指標設定警示。在 Graphed metrics (圖表化指標) 索引標籤上，選取 Actions (動作) 資料欄下的 Create alarm (建立警示) 圖示。Create alarm (建立警示) 圖示表示為一個小鐘。如需詳細資訊和通知選項，請參閱《Amazon CloudWatch 使用者指南》中的根據指標數學表達式建立 CloudWatch 警示，以及通知使用者警示變更。 Amazon CloudWatch

或者，您可以使用 GetMetricData 和 PutMetricAlarm 利用指標數學來執行計算，並根據輸出建立警示。

使用排程動作覆寫 Amazon ECS 的預測值

有時候，您可能會有未來應用程式需求的其他資訊，但預測計算無法考量該資訊。例如，預測計算可能會低估即將到來的行銷活動所需的任務。您可以使用排程動作，在未來時段暫時覆寫預測。排程動作可以週期性執行，或在一次性需求波動的特定日期與時間執行。

例如，您可以建立排程動作，其任務數量高於預測。在執行時間，Amazon ECS 會更新服務中任務的最小數量。由於預測擴展會針對任務數量進行最佳化，因此會遵守任務數量下限高於預測值的排程動作。這可防止任務數量低於預期。若要停止覆寫預測，請使用第二個排程動作，將最低數量的任務傳回其原始設定。

下列程序概述在未來時段覆寫預測的步驟。

主題

- 步驟 1：(選用) 分析時間序列資料
- 步驟 2：建立兩個排程動作

> **Important**
>
> 本主題假設您嘗試覆寫預測，以擴展到比預測更高的容量。如果您需要暫時減少任務數量，而不受到預測擴展政策的干擾，請改用預測模式。在僅限預測模式下，預測擴展將繼續產生預測，但不會自動增加任務數量。然後，您可以監控資源使用率，並視需要手動減少任務數量。

步驟 1：(選用) 分析時間序列資料

從分析預測時間序列資料開始。這是選用步驟，但是如果您想要了解預測的詳細資訊，這會很有幫助。

1. 擷取預測

   建立預測之後，您可以查詢預測中的特定時段。查詢的目標是取得特定時段的時間序列資料的完整檢視。

   查詢最多可包含未來兩天的預測資料。如果已經使用預測擴展一段時間，您也可以存取過去的預測資料。不過，開始和結束時間之間的最大持續時間是 30 天。

   若要使用 get-predictive-scaling-forecast AWS CLI 命令取得預測，請在 命令中提供下列參數：

   - 在 `resource-id` 參數中輸入叢集名稱的名稱。
   - 在 `--policy-name` 參數中輸入政策名稱。
   - 在 `--start-time` 參數中輸入開始時間，以便僅將指定時間之時或之後的預測資料傳回。
   - 在 `--end-time` 參數中輸入結束時間，以便僅將指定時間之前的預測資料傳回。

```
aws application-autoscaling get-predictive-scaling-forecast \
    --service-namespace ecs \
    --resource-id service/MyCluster/test \
    --policy-name cpu40-predictive-scaling-policy \
    --scalable-dimension ecs:service:DesiredCount \
    --start-time "2021-05-19T17:00:00Z" \
    --end-time "2021-05-19T23:00:00Z"
```

   如果成功，此命令會傳回類似下方範例的資料。

```
{
    "LoadForecast": [
        {
            "Timestamps": [
                "2021-05-19T17:00:00+00:00",
                "2021-05-19T18:00:00+00:00",
                "2021-05-19T19:00:00+00:00",
                "2021-05-19T20:00:00+00:00",
                "2021-05-19T21:00:00+00:00",
                "2021-05-19T22:00:00+00:00",
                "2021-05-19T23:00:00+00:00"
```

```
            ],
            "Values": [
                153.0655799339254,
                128.8288551285919,
                107.1179447150675,
                197.3601844551528,
                626.4039934516954,
                596.9441277518481,
                677.9675713779869
            ],
            "MetricSpecification": {
                "TargetValue": 40.0,
                "PredefinedMetricPairSpecification": {
                    "PredefinedMetricType": "ASGCPUUtilization"
                }
            }
        }
    ],
    "CapacityForecast": {
        "Timestamps": [
            "2021-05-19T17:00:00+00:00",
            "2021-05-19T18:00:00+00:00",
            "2021-05-19T19:00:00+00:00",
            "2021-05-19T20:00:00+00:00",
            "2021-05-19T21:00:00+00:00",
            "2021-05-19T22:00:00+00:00",
            "2021-05-19T23:00:00+00:00"
        ],
        "Values": [
            2.0,
            2.0,
            2.0,
            2.0,
            4.0,
            4.0,
            4.0
        ]
    },
    "UpdateTime": "2021-05-19T01:52:50.118000+00:00"
}
```

回應包含兩種預測：LoadForecast 和 CapacityForecast。LoadForecast 顯示每小時負載預測。CapacityForecast 顯示每小時處理預測負載時所需的容量預測值，同時維持 TargetValue 為 40.0 (40% 的 CPU 平均使用率)。

2. 確定目標時段

   確定應發生一次性需求波動時的一個小時或數個小時。請記住，預測中顯示的日期和時間為 UTC 格式。

步驟 2：建立兩個排程動作

接下來，在應用程式具有高於預測的負載時，為特定時段建立兩個排程動作。舉例來說，如果行銷活動會在特定時段為網站帶來流量，您可以排程一次性動作，在它開始時更新最小容量。然後，排程另一個動作，以便在事件結束時將最小容量恢復至原始設定。

1. 開啟主控台，網址為 https://console.aws.amazon.com/ecs/v2。
2. 在叢集頁面上，選擇叢集。
3. 在叢集詳細資訊頁面上的服務區段中，然後選擇服務。

   服務詳細資訊頁面隨即出現。

4. 選擇 Service Auto Scaling。

   政策頁面隨即出現。

5. 選擇排程動作，然後選擇建立。

   隨即顯示建立排程動作頁面。

6. 針對動作名稱，輸入唯一的名稱。
7. 對於 Time zone (時區)，選擇時區。

   所有列出的時區都來自 IANA 時區資料庫。如需詳細資訊，請參閱 tz 資料庫時區清單。

8. 針對開始時間，輸入動作開始的日期和時間。
9. 針對 Recurrence (重複)，選擇 Once (一次)。
10. 在任務調整下，針對最小值，輸入小於或等於任務數量上限的值。
11. 選擇建立排程動作。

    政策頁面隨即出現。

12. 設定第二個排程動作，以在事件結束時將任務的最小數量傳回原始設定。只有當您為最小值設定的值低於預測值時，預測擴展才能擴展任務數量。

為一次性事件建立兩個排程動作 (AWS CLI)

若要使用 AWS CLI 建立排程動作，請使用 put-scheduled-update-group-action 命令。

例如，我們定義一個排程，在 5 月 19 日下午 5 點維持三個執行個體的最小容量，持續 8 小時。下列命令顯示如何實作此案例。

第一個 put-scheduled-update-group-action 命令會指示 Amazon EC2 Auto Scaling 在 2021 年 5 月 19 日下午 5 點 (UTC) 更新 Auto Scaling 群組的最小容量。

```
aws autoscaling put-scheduled-update-group-action --scheduled-action-name my-event-
start \
  --auto-scaling-group-name my-asg --start-time "2021-05-19T17:00:00Z" --minimum-
capacity 3
```

第二個命令指示 Amazon EC2 Auto Scaling 在 2021 年 5 月 20 日上午 1 點 (UTC)，將群組的最小容量設定為 1。

```
aws autoscaling put-scheduled-update-group-action --scheduled-action-name my-event-end
 \
  --auto-scaling-group-name my-asg --start-time "2021-05-20T01:00:00Z" --minimum-
capacity 1
```

再將這些排程動作新增到 Auto Scaling 群組後，Amazon EC2 Auto Scaling 會執行下列動作：

- 在 2021 年 5 月 19 日下午 5 點 (UTC)，第一個排程動作會執行。如果群組目前擁有少於 3 個執行個體，群組則會擴增至 3 個執行個體。在此時間和接下來的八個小時內，如果預測容量高於實際容量，或者如果動態擴展政策生效，則 Amazon EC2 Auto Scaling 可以繼續擴增。
- 在 2021 年 5 月 20 日上午 1 點 (UTC)，第二個排程動作會執行。這會在事件結束時將最小容量恢復至原始設定。

根據週期性排程擴展

若要每週覆寫相同時段的預測，請建立兩個排程動作，並使用 Cron 表達式提供時間與日期邏輯。

Cron 表達式格式由 5 個以空格分隔的欄位組成：[分鐘] [小時] [一個月的第幾日] [一年的第幾個月] [一週的第幾日]。欄位可以包含任何允許的數值，包括特殊字元。

例如，以下 Cron 表達式會在每周二上午 6:30 執行動作。使用星號作為萬用字元，以比對欄位的所有數值。

```
30 6 * * 2
```

另請參閱

如需如何管理排程動作的詳細資訊，請參閱使用排程動作來擴展 Amazon ECS 服務。

使用 Amazon ECS 自訂指標的進階預測擴展政策

您可以在預測擴展政策中使用預先定義或自訂的指標。當預先定義的指標，例如 CPU、記憶體等）不足以充分描述您的應用程式負載時，自訂指標很有用。

使用自訂指標建立預測擴展政策時，您可以指定 提供的其他 CloudWatch 指標 AWS。或者，您可以指定自己定義的指標和發佈指標。您也可以使用指標數學，將現有的指標彙總並轉換為 AWS 不會自動追蹤的新時間序列。例如，透過計算稱為彙總的新總和或平均值來合併資料中的值。產生的資料稱為彙總。

下一節包含如何建構政策的 JSON 結構的最佳實務和範例。

必要條件

若要在預測擴展政策中新增自訂指標，您必須擁有 `cloudwatch:GetMetricData` 許可。

若要指定自己的指標，而不是 AWS 提供的指標，您必須先將指標發佈至 CloudWatch。如需詳細資訊，請參閱《Amazon CloudWatch 使用者指南》中的發佈自訂指標。

如果您發佈自己的指標，應確保以至少五分鐘的頻率發佈資料點。資料點會根據所需的期間長度從 CloudWatch 擷取。例如，負載指標規範使用每小時指標來衡量應用程式的負載。CloudWatch 使用您發佈的指標資料，透過將時間戳記落於同一小時週期內的所有資料點彙總，為任何一小時週期提供單一資料值。

最佳實務

以下最佳實務可協助您更有效地使用自訂指標：

- 負載指標規格最有用的指標是代表 Auto Scaling 群組整體負載的指標。
- 擴展指標規格要擴展的最有用指標，是每個任務指標的平均輸送量或使用率。

- 目標使用率必須與擴展指標的類型相符。對於使用 CPU 使用率的政策組態，例如，這是目標百分比。
- 如果不遵循這些建議，則時間序列的預測未來值可能不正確。要驗證資料是否正確，您可以在 主控台中檢視預測值。或者，在您建立預測擴展政策之後，請檢查呼叫 GetPredictiveScalingForecast API 時傳回的`LoadForecast`物件。
- 我們強烈建議您在僅限預測模式下設定預測擴展，以便在預測擴展開始主動擴展之前評估預測。

限制

- 您可以在一個指標規範中查詢最多 10 個指標的資料點。
- 在此限制之下，一個表達式計為一個指標。

使用自訂指標對預測擴展政策進行故障診斷

如果使用自訂指標時出現問題，建議您執行以下操作：

- 如果您在使用搜尋表達式時遇到藍/綠部署中的問題，請確定您建立的搜尋表達式正在尋找部分相符，而不是完全相符。您也應該檢查查詢是否僅尋找在特定應用程式中執行的 Auto Scaling 群組。如需搜尋表達式語法的詳細資訊，請參閱《Amazon CloudWatch 使用者指南》中的 CloudWatch 搜尋表達式語法。
- put-scaling-policy 命令會在您建立擴展政策時驗證表達式。不過，此命令可能無法識別偵測到錯誤的確切原因。若要修正這些問題，請對您收到之 get-metric-data 命令請求回應中的錯誤進行疑難排解。您也可以從 CloudWatch 主控台對表達式進行疑難排解。
- 如果 `MetricDataQueries` 自己指定了 SEARCH() 函數 (在無需 SUM() 等數學函數的狀況下)，則您必須為 `ReturnData` 指定 `false`。這是因為搜尋表達式可能會傳回多個時間序列，並且基於表達式的指標規範只能傳回一個時間序列。
- 搜尋表達式中涉及的所有指標均應具有相同的解析度。

使用 Amazon ECS 建構 JSON 以預測擴展自訂指標

下一節包含如何設定預測擴展以查詢來自 CloudWatch 的資料的範例。設定此選項有兩種不同的方法，而您選擇的方法會影響您用來建構預測擴展政策的 JSON 的格式。使用指標數學時，JSON 的格式會根據所執行的指標數學而進一步變化。

1. 若要建立直接從您發佈至 CloudWatch 的其他 CloudWatch AWS 指標取得資料的政策，請參閱 使用 搭配自訂負載和擴展指標的預測擴展政策範例 AWS CLI。

使用 搭配自訂負載和擴展指標的預測擴展政策範例 AWS CLI

若要使用 建立具有自訂載入和擴展指標的預測擴展政策 AWS CLI，請將 的引數存放在名為 `--predictive-scaling-configuration`的 JSON 檔案中`config.json`。

您可以藉由將以下範例中的可替換值換成您的指標和目標使用率的值，開始新增自訂指標。

```
{
  "MetricSpecifications": [
    {
      "TargetValue": 50,
      "CustomizedScalingMetricSpecification": {
        "MetricDataQueries": [
          {
            "Id": "scaling_metric",
            "MetricStat": {
              "Metric": {
                "MetricName": "MyUtilizationMetric",
                "Namespace": "MyNameSpace",
                "Dimensions": [
                  {
                    "Name": "MyOptionalMetricDimensionName",
                    "Value": "MyOptionalMetricDimensionValue"
                  }
                ]
              },
              "Stat": "Average"
            }
          }
        ]
      },
      "CustomizedLoadMetricSpecification": {
        "MetricDataQueries": [
          {
            "Id": "load_metric",
            "MetricStat": {
              "Metric": {
                "MetricName": "MyLoadMetric",
                "Namespace": "MyNameSpace",
                "Dimensions": [
                  {
                    "Name": "MyOptionalMetricDimensionName",
                    "Value": "MyOptionalMetricDimensionValue"
                  }
```

```
                ]
              },
              "Stat": "Sum"
            }
          }
        ]
      }
    }
  ]
}
```

如需詳細資訊，請參閱《Amazon EC2 Auto Scaling API 參考》中的 MetricDataQuery。

> **Note**
>
> 以下是一些其他資源，可協助您尋找 CloudWatch 指標的指標名稱、命名空間、維度和統計資料：
>
> - 如需 AWS 服務可用指標的相關資訊，請參閱《Amazon AWS CloudWatch 使用者指南》中的發佈 CloudWatch 指標的 服務。 Amazon CloudWatch
> - 若要使用 取得 CloudWatch 指標的確切指標名稱、命名空間和維度 （ 如適用 ） AWS CLI，請參閱 清單指標。

若要建立此政策，執行使用 JSON 檔案作為輸入 put-scaling-policy 命令，如以下範例所示。

```
aws application-autoscaling put-scaling-policy --policy-name my-predictive-scaling-policy \
  --auto-scaling-group-name my-asg --policy-type PredictiveScaling \
  --predictive-scaling-configuration file://config.json
```

如果成功，此命令會傳回政策的 Amazon Resource Name (ARN)。

```
{
  "PolicyARN": "arn:aws:autoscaling:region:account-id:scalingPolicy:2f4f5048-d8a8-4d14-b13a-d1905620f345:autoScalingGroupName/my-asg:policyName/my-predictive-scaling-policy",
  "Alarms": []
}
```

使用指標數學表達式

下節提供在您的政策中使用指標數學與預測擴展政策的相關資訊。

了解指標數學

如果您只想彙總現有指標資料，則 CloudWatch 指標數學可以節省將另一個指標發佈至 CloudWatch 的工作量和成本。您可以使用 AWS 提供的任何指標，也可以使用您在應用程式中定義的指標。

如需詳細資訊，請參閱《Amazon CloudWatch 使用者指南》中的使用指標數學。

如果選擇在預測擴展政策中使用指標數學表達式，則請考慮以下幾點：

- 指標數學運算會使用指標名稱、命名空間和指標維度鍵/值對之唯一組合的資料點。
- 您可以使用任何算術運算子 (+-*/^)、統計函數 (如 AVG 或 SUM) 或 CloudWatch 支援的其他函數。
- 您可以在數學表達式的公式中同時使用其他數學表達式的指標和結果。
- 您的指標數學表達式可以由不同的彙總組成。但是，最終彙總結果的最佳實務是將 `Average` 用於擴展指標，`Sum` 用於負載指標。
- 在指標規範中使用的任何表達式都必須最終傳回單一的時間序列。

若要使用指標數學，請執行以下操作：

- 選擇一或多個 CloudWatch 指標。然後，建立表達式。如需詳細資訊，請參閱《Amazon CloudWatch 使用者指南》中的使用指標數學。
- 透過使用 CloudWatch 主控台或 CloudWatch GetMetricData API 驗證指標數學表達式是否有效。

使用指標數學組合指標的預測擴展政策範例 (AWS CLI)

有時，您可能需要首先以某種方式處理其資料，而不是直接指定指標。例如，您可能有一個從 Amazon SQS 佇列中提取工作的應用程式，並且您可能想要使用佇列中的項目數量作為預測擴展的條件。佇列中的訊息數量並不僅僅定義所需的執行個體數量。因此，需要更多的工作來建立可用於計算每個執行個體待處理項目的指標。

以下是此案例的預測擴展政策範例。它指定了基於 Amazon SQS `ApproximateNumberOfMessagesVisible` 指標的擴展和負載指標，即可從佇列中擷取的訊息數量。它還使用 Amazon EC2 Auto Scaling `GroupInServiceInstances` 指標和數學表達式來計算每個執行個體的待處理項目，以擴展指標。

```
aws application-autoscaling put-scaling-policy --policy-name my-sqs-custom-metrics-
policy \
  --policy-type PredictiveScaling \
  --predictive-scaling-configuration file://config.json
  --service-namespace ecs \
  --resource-id service/MyCluster/test \
  "MetricSpecifications": [
    {
      "TargetValue": 100,
      "CustomizedScalingMetricSpecification": {
        "MetricDataQueries": [
          {
            "Label": "Get the queue size (the number of messages waiting to be
 processed)",
            "Id": "queue_size",
            "MetricStat": {
              "Metric": {
                "MetricName": "ApproximateNumberOfMessagesVisible",
                "Namespace": "AWS/SQS",
                "Dimensions": [
                  {
                    "Name": "QueueName",
                    "Value": "my-queue"
                  }
                ]
              },
              "Stat": "Sum"
            },
            "ReturnData": false
          },
          {
            "Label": "Get the group size (the number of running instances)",
            "Id": "running_capacity",
            "MetricStat": {
              "Metric": {
                "MetricName": "GroupInServiceInstances",
                "Namespace": "AWS/AutoScaling",
                "Dimensions": [
                  {
                    "Name": "AutoScalingGroupName",
                    "Value": "my-asg"
                  }
                ]
```

```
              },
              "Stat": "Sum"
            },
            "ReturnData": false
          },
          {
            "Label": "Calculate the backlog per instance",
            "Id": "scaling_metric",
            "Expression": "queue_size / running_capacity",
            "ReturnData": true
          }
        ]
      },
      "CustomizedLoadMetricSpecification": {
        "MetricDataQueries": [
          {
            "Id": "load_metric",
            "MetricStat": {
              "Metric": {
                "MetricName": "ApproximateNumberOfMessagesVisible",
                "Namespace": "AWS/SQS",
                "Dimensions": [
                  {
                    "Name": "QueueName",
                    "Value": "my-queue"
                  }
                ],
              },
              "Stat": "Sum"
            },
            "ReturnData": true
          }
        ]
      }
    }
  ]
}
```

該範例會傳回政策的 ARN。

```
{
  "PolicyARN": "arn:aws:autoscaling:region:account-id:scalingPolicy:2f4f5048-d8a8-4d14-
b13a-d1905620f345:autoScalingGroupName/my-asg:policyName/my-sqs-custom-metrics-policy",
```

```
  "Alarms": []
}
```

互連 Amazon ECS 服務

在 Amazon ECS 任務中執行的應用程式通常需要接收來自網際網路的連線，或通常需要連線到在 Amazon ECS 服務中執行的其他應用程式。如果需要來自網際網路的外部連線，建議您使用 Elastic Load Balancing。如需有關整合負載平衡的詳細資訊，請參閱the section called “使用負載平衡來分配服務流量”。

如果您需要應用程式連線到在 Amazon ECS 服務中執行的其他應用程式，Amazon ECS 提供以下無需負載平衡器的方法：

- Amazon ECS Service Connect

  我們建議 Service Connect，其提供 Amazon ECS 組態，用於服務探索、連線能力和流量監控。使用 Service Connect，您的應用程式可以使用短名稱和標準連接埠來連接到相同叢集中的 Amazon ECS 服務，其他叢集，包括相同叢集中的跨 VPCs AWS 區域。

  當您使用 Service Connect 時，Amazon ECS 會管理服務探索的所有部分：建立可探索的名稱、在任務開始和停止時動態管理每個任務的項目、在設定為探索名稱的每個任務中執行代理程式。您的應用程式可以使用 DNS 名稱的標準功能並進行連線，來查詢名稱。如果您的應用程式已經這樣做，則不需要修改應用程式來使用 Service Connect。

  您可以在每個服務和任務定義內提供完整的組態。Amazon ECS 會管理每個服務部署中此組態的變更，以確保部署中的所有任務都以相同的方式運作。例如，DNS 做為服務探索的常見問題是控制遷移。如果您變更 DNS 名稱以指向新的替換 IP 地址，則可能需要最長的 TTL 時間，所有用戶端才會開始使用新的服務。透過 Service Connect，用戶端部署會取代用戶端任務來更新組態。您可以設定部署斷路器和其他部署組態，以與任何其他部署相同的方式影響 Service Connect 變更。

  如需詳細資訊，請參閱使用 Service Connect 以短名稱連接 Amazon ECS 服務。

- Amazon ECS 服務探索

  service-to-service通訊的另一種方法是使用服務探索進行直接通訊。在此方法中，您可以使用 AWS Cloud Map 服務探索與 Amazon ECS 的整合。使用服務探索，Amazon ECS 會將啟動任務的清單同步至 AWS Cloud Map，這會維護 DNS 主機名稱，以解析該特定服務中一或多個任務的內部 IP 地址。Amazon VPC 中的其他服務可以使用此 DNS 主機名稱，使用其內部 IP 地址直接將流量傳送到另一個容器。

這種service-to-service通訊的方法提供低延遲。容器之間沒有額外的元件。流量會直接從一個容器流向另一個容器。

此方法適合使用awsvpc網路模式，其中每個任務都有自己的唯一 IP 地址。大多數軟體僅支援使用 DNS A記錄，而 DNS 記錄會直接解析為 IP 地址。使用awsvpc網路模式時，每個任務的 IP 地址都是A記錄。不過，如果您使用bridge網路模式，多個容器可能會共用相同的 IP 地址。此外，動態連接埠映射會導致容器在該單一 IP 地址上隨機指派連接埠號碼。此時，A記錄已不足以進行服務探索。您也必須使用 SRV記錄。這種類型的記錄可以同時追蹤 IP 地址和連接埠號碼，但您需要適當地設定應用程式。您使用的某些預先建置應用程式可能不支援SRV記錄。

awsvpc 網路模式的另一個優點是，每個服務都有唯一的安全群組。您可以設定此安全群組，僅允許來自需要與該服務通訊之特定上游服務的傳入連線。

使用服務探索進行直接service-to-service服務通訊的主要缺點是，您必須實作額外的邏輯，以重試並處理連線失敗。DNS 記錄具有time-to-live(TTL) 期間，可控制快取的時間長度。更新 DNS 記錄和快取過期需要一些時間，以便您的應用程式可以取得 DNS 記錄的最新版本。因此，您的應用程式最終可能會解析 DNS 記錄，以指向不存在的另一個容器。您的應用程式需要處理重試，並具有邏輯來忽略不良後端。

如需詳細資訊，請參閱 使用服務探索將 Amazon ECS 服務與 DNS 名稱連線

- Amazon VPC Lattice

  Amazon VPC Lattice 是一種受管應用程式聯網服務，Amazon ECS 客戶可以用來觀察、保護和監控跨 AWS 運算服務、VPCs 和帳戶建置的應用程式，而無需修改其程式碼。

  VPC Lattice 使用目標群組，這是運算資源的集合。這些目標會執行您的應用程式或服務，可以是 Amazon EC2 執行個體、IP 地址、Lambda 函數和 Application Load Balancer。透過將 Amazon ECS 服務與 VPC Lattice 目標群組建立關聯，客戶現在可以在 VPC Lattice 中啟用 Amazon ECS 任務做為 IP 目標。Amazon ECS 會在已註冊服務的任務啟動時，自動將任務註冊至 VPC Lattice 目標群組。

  如需詳細資訊，請參閱使用 Amazon VPC Lattice 連線、觀察和保護您的 Amazon ECS 服務。

網路模式相容性資料表

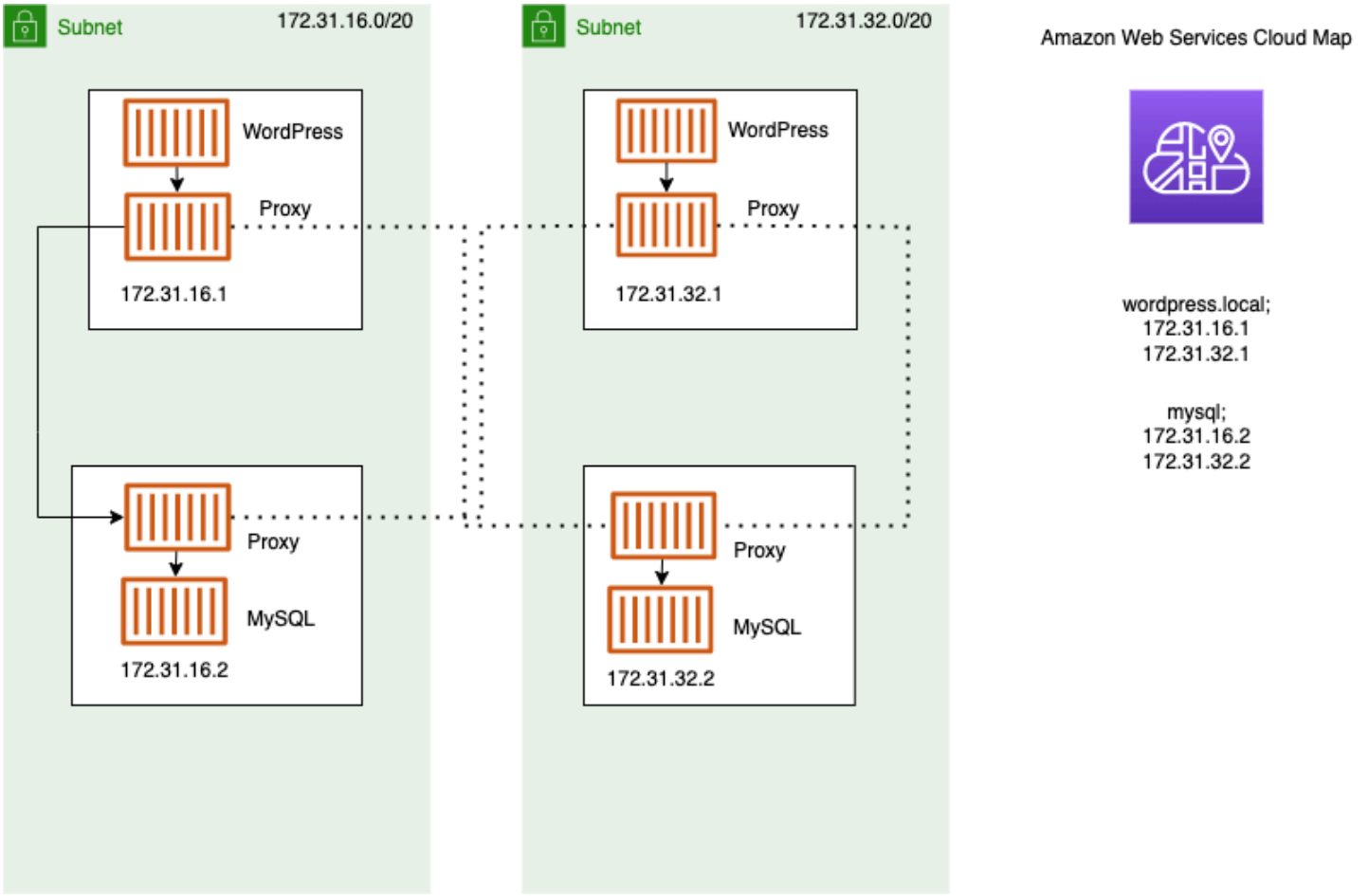
下表說明這些選項與任務網路模式之間的相容性。在資料表中，「用戶端」是指從 Amazon ECS 任務內部建立連線的應用程式。

互連線選項	橋接	awsvpc	主機
服務探索	是的，但用戶端需要知道在 DNS 中不含 hostPort 的 SRV 記錄。	是	是的，但用戶端需要知道在 DNS 中不含 hostPort 的 SRV 記錄。
Service Connect	是	是	否
VPC Lattice	是	是	是

使用 Service Connect 以短名稱連接 Amazon ECS 服務

Amazon ECS Service Connect 提供服務對服務通訊的管理作為 Amazon ECS 組態。它在 Amazon ECS 中建置服務探索和服務網格。這可在您由服務部署管理的每個服務內提供完整的組態、在命名空間中參考您服務的統一方式，而這些方式不依賴 VPC DNS 組態，以及標準化指標和日誌來監控所有應用程式。Service Connect 僅互連服務。

下圖顯示 VPC 和 2 個服務中具有 2 個子網路的 Service Connect 網路範例。一種用戶端服務，在每個子網路中執行具有 1 項任務的 WordPress。一種伺服器服務，在每個子網路中執行具有 1 項任務的 MySQL。這兩種服務都具有高度可用性，並且對任務和可用區域問題具有彈性，因為每個服務都會執行分散在 2 個子網路上的多個任務。實心箭頭顯示從 WordPress 到 MySQL 的連線。例如，從具有 IP 地址 `172.31.16.1` 任務中的 WordPress 容器內執行的 `mysql --host=mysql` CLI 命令。該命令在 MySQL 預設連接埠上使用簡短名稱 `mysql`。此名稱和連接埠會在相同任務中連線至 Service Connect Proxy。WordPress 任務中的 Proxy 會在異常值偵測中使用循環配置負載平衡，以及任何先前的失敗資訊，以挑選要連接的 MySQL 任務。如圖中的實心箭頭所示，Proxy 連線至具 IP 地址 `172.31.16.2` MySQL 任務中的第二個 Proxy。在同一任務中，第二個 Proxy 連線至本機 MySQL 伺服器。這兩個 Proxy 都會回報在 Amazon ECS 和 Amazon CloudWatch 主控台中以圖形顯示的連線效能，讓您以相同的方式從各種應用程式取得效能指標。



以下術語會與 Service Connect 搭配使用。

連接埠名稱

將名稱指派給特定連接埠映射的 Amazon ECS 任務定義組態。此組態僅供 Amazon ECS Service Connect 使用。

用戶端別名

Amazon ECS 服務組態，用於指派端點中使用的連接埠號碼。此外，用戶端別名可以指派端點的 DNS 名稱，以覆寫探索名稱。如果 Amazon ECS 服務中未提供探索名稱，則用戶端別名會覆寫連接埠名稱作為端點名稱。如需端點範例，請參閱端點的定義。您可以將多個用戶端別名指派給 Amazon ECS 服務。此組態僅供 Amazon ECS Service Connect 使用。

探索名稱

此為選用的中繼名稱，您可以從任務定義為指定的連接埠建立此名稱。此名稱用於建立 AWS Cloud Map 服務。如果未提供此名稱，則會使用任務定義中的連接埠名稱。您可以將多個探索名稱指派給 Amazon ECS 服務的特定連接埠。此組態僅供 Amazon ECS Service Connect 使用。

AWS Cloud Map 服務名稱在命名空間中必須是唯一的。由於此限制，對於每個命名空間中的特定任務定義，如果未提供探索名稱，您只能有一個 Service Connect 組態。

端點

連線到 API 或網站的 URL。URL 包含協定、DNS 名稱和連接埠。如需一般端點的更多資訊，請參閱 Amazon Web Services 一般參考內AWS 詞彙表中的端點。

Service Connect 可建立與 Amazon ECS 服務連線的端點，並將 Amazon ECS 服務中的任務設定為連線到端點。URL 包含協定、DNS 名稱和連接埠。您可以在任務定義中選取協定和連接埠名稱，因為連接埠必須比對容器映像內的應用程式。在服務中，您可以依名稱選取每個連接埠，並且可以指派 DNS 名稱。如果您未在 Amazon ECS 服務組態中指定 DNS 名稱，則會依預設使用任務定義中的連接埠名稱。例如，Service Connect 端點可以是 `http://blog:80`、`grpc://checkout:8080` 或 `http://_db.production.internal:99`。

Service Connect 服務

Amazon ECS 服務中單一端點的組態。這是 Service Connect 組態的一部分，包含主控台中 Service Connect and discovery name configuration (Service Connect 和探索名稱組態) 中的單一列，或是 Amazon ECS 服務 JSON 組態內 `services` 清單中的某個物件。此組態僅供 Amazon ECS Service Connect 使用。

如需詳細資訊，請參閱 Amazon Elastic Container Service API Reference (《Amazon Elastic Container Service API 參考》) 中的 ServiceConnectService。

命名空間

用於 Service Connect 之 AWS Cloud Map 命名空間的簡短名稱或完整 Amazon Resource Name (ARN)。命名空間必須與 Amazon ECS 服務和叢集 AWS 區域 位於相同位置。中的命名空間類型 AWS Cloud Map 不會影響 Service Connect。

Service Connect 使用 AWS Cloud Map 命名空間做為彼此通訊的 Amazon ECS 任務邏輯群組。每個 Amazon ECS 服務只能屬於一個命名空間。命名空間內的服務可以分散到相同 AWS 區域 內的不同 Amazon ECS 叢集 AWS 帳戶。您可以依據任何條件自由組織服務。

用戶端服務

執行網路用戶端應用程式的服務。此服務必須設定命名空間。服務中的每個任務都可以透過 Service Connect Proxy 容器探索，並連線到命名空間中的所有端點。

如果任務中任何容器需要從命名空間中的服務連線到端點，請選擇用戶端服務。如果前端、反向 Proxy 或負載平衡器應用程式透過其他方法 (例如 Elastic Load Balancing) 接收外部流量，則可使用此類型的 Service Connect 組態。

用戶端-伺服器服務

執行網路或 Web 服務應用程式的 Amazon ECS 服務。此服務必須具有命名空間，並且至少已設定一個端點。您可以使用端點連上服務中的每項任務。Service Connect Proxy 容器會接聽端點名稱和連接埠，以將流量引導至任務中的應用程式容器。

如果有任何容器在連接埠上公開並接聽網路流量，請選擇用戶端-伺服器服務。這些應用程式不需要連線到相同命名空間中的其他用戶端伺服器服務，但需要用戶端組態。後端、中介軟體、商業層或大多數微服務都可以使用這種類型的 Service Connect 組態。如果您希望前端、反向 Proxy 或負載平衡器應用程式接收來自相同命名空間中使用 Service Connect 設定的其他服務流量，則這些服務應使用此類型的 Service Connect 組態。

Service Connect 功能會建立相關服務的虛擬網路。您可以跨多個不同的命名空間使用相同的服務組態，以執行獨立但相同的應用程式集合。Service Connect 會定義 Amazon ECS 服務中的 Proxy 容器。如此一來，您可以使用相同的任務定義，在具有不同 Service Connect 組態的不同命名空間中執行相同的應用程式。服務執行的每個任務都會在任務中執行代理容器。

Service Connect 適用於相同命名空間內 Amazon ECS 服務之間的連線。對於下列應用程式，您需要使用額外的互連線方法，才能連線到使用 Service Connect 設定的 Amazon ECS 服務：

- 在其他命名空間中設定的任務
- 未針對 Service Connect 設定的任務
- Amazon ECS 以外的其他應用程式

這些應用程式可以透過 Service Connect Proxy 連線，但無法解析 Service Connect 端點名稱。

若要讓這些應用程式解析 Amazon ECS 任務的 IP 地址，您需要使用另一個互連方法。

定價

Amazon ECS Service Connect 定價取決於您是否使用 AWS Fargate 或 Amazon EC2 基礎設施來託管容器化工作負載。在 上使用 Amazon ECS 時 AWS Outposts，定價會遵循直接使用 Amazon EC2 時所使用的相同模型。如需詳細資訊，請參閱 Amazon ECS 定價。

AWS Cloud Map 當 Service Connect 使用它時，用量完全免費。

Amazon ECS Service Connect 元件

當您使用 Amazon ECS Service Connect 時，您可以將每個 Amazon ECS 服務設定為執行接收網路請求的伺服器應用程式（用戶端伺服器服務），或執行發出請求的用戶端應用程式（用戶端服務）。

當您準備開始使用 Service Connect 時，請先從用戶端-伺服器服務開始。您可以將 Service Connect 組態新增至新服務或現有服務。Amazon ECS 在命名空間中建立 Service Connect 端點。此外，Amazon ECS 會在服務中建立新部署來取代目前正在執行的任務。

現有任務和其他應用程式可以繼續連線到現有端點和外部應用程式。如果用戶端-伺服器服務透過向外擴展來新增任務，則來自用戶端的新連線在所有任務之間會保持平衡。如果更新了用戶端-伺服器服務，來自用戶端的新連線將在新版本 的任務之間取得平衡。

現有任務無法解析並連線到新端點。只有具有相同命名空間中 Service Connect 組態且在此部署之後開始執行的新任務，才能解析並連線至此端點。

這意味著用戶端應用程式的運算子會判斷其應用程式的組態何時變更，即使伺服器應用程式的運算子可以隨時變更其組態也是如此。每次部署命名空間中的任何服務時，命名空間中的端點清單都會變更。現有任務和替換任務的行為會繼續與最近部署之後的行為相同。

請考慮下列範例。

首先，假設您正在建立可在單一 AWS CloudFormation 範本和單一 AWS CloudFormation 堆疊中供公有網際網路使用的應用程式。公有探索和連線能力應由 持續建立 AWS CloudFormation，包括前端用戶端服務。需要按照此順序建立該服務，以防止在一段時間內，前端用戶端服務正在執行並可公開使用，但後端用戶端服務不是。如此可避免在該期間向大眾傳送錯誤訊息。在 中 AWS CloudFormation，您必須使用 `depends0n` 來向 AWS CloudFormation 指出多個 Amazon ECS 服務無法平行或同時進行。您應該為用戶端任務連線的每個後端用戶端-伺服器服務，將 `depends0n` 新增至前端用戶端服務。

其次，假設前端服務存在，但沒有 Service Connect 組態。任務正在連線到現有的後端服務。請先使用 DNS 中或前端使用的 `clientAlias` 相同名稱，將用戶端-伺服器 Service Connect 組態新增至後端服務。這會建立新的部署，因此所有部署復原偵測 AWS Management Console或 AWS SDKs AWS CLI和其他方法來復原後端服務，並將後端服務還原至先前的部署和組態。如果您對後端服務的效能和行為感到滿意，請將用戶端或用戶端-伺服器 Service Connect 組態新增至前端服務。只有新部署中的任務會使用在這些新任務中新增的 Service Connect Proxy。如果此組態有問題，您可以使用部署復原偵測或 AWS Management Console、 AWS CLI、 AWS SDKs和其他方法來復原和還原至先前的組態，並將後端服務還原至先前的部署和組態。如果您使用另一個以 DNS 為基礎的服務探索系統，而非 Service Connect，則任何前端或用戶端應用程式會在本機 DNS 快取到期後開始使用新端點和變更的端點組態，該過程通常需要數小時。

聯網

根據預設，Service Connect 代理`containerPort`會從任務定義連接埠映射接聽 。您的安全群組規則必須允許從用戶端將執行的子網路傳入（傳入）流量到此連接埠。

即使您在 Service Connect 服務組態中設定連接埠號碼，也不會變更 Service Connect Proxy 所接聽用戶端-伺服器服務的連接埠。當您設定此連接埠號碼時，Amazon ECS 會在這些任務內的 Service Connect Proxy 上變更用戶端服務連線到其中的端點連接埠。用戶端服務中的 Proxy 會使用 `containerPort` 連線至用戶端-伺服器服務中的 Proxy。

如果您想要變更 Service Connect Proxy 接聽的連接埠，請變更用戶端-伺服器服務 Service Connect 組態中的 `ingressPort0verride`。如果您變更此連接埠號碼，則必須允許流量使用此服務的此連接埠上的傳入流量。

應用程式傳送到針對 Service Connect 設定的 Amazon ECS 服務的流量，會要求 Amazon VPC 和子網路具有路由表規則和網路 ACL 規則，以允許您正在使用的 `containerPort` 和 `ingressPort0verride` 連接埠號碼。

您可以使用 Service Connect 在 VPCs之間傳送流量。路由表規則、網路 ACLs 和安全群組的相同需求適用於這兩個 VPCs。

例如，兩個叢集會在不同的 VPC 中建立任務。每個叢集中的服務會設定為使用相同的命名空間。這兩個服務中的應用程式無需任何 VPC DNS 組態，即可解析命名空間中的每個端點。不過，除非 VPC 對等互連、VPC 或子網路路由表，以及 VPC 網路 ACLs 允許 `containerPort`和`ingressPort0verride`連接埠號碼上的流量，否則代理無法連線。

對於使用`bridge`聯網模式的任務，您必須使用允許上動態連接埠範圍流量的傳入規則來建立安全群組。然後，將安全群組指派給 Service Connect 叢集中的所有 EC2 執行個體。

Service Connect Proxy

如果您使用 Service Connect 組態建立或更新服務，Amazon ECS 會在啟動時為每個新任務新增一個容器。這種使用單獨容器的模式稱為 `sidecar`。此容器在任務定義中不存在，您無法對其進行設定。Amazon ECS 會在 服務中管理容器組態。這可讓您在沒有 Service Connect 的情況下，在多個服務、命名空間和任務之間重複使用相同的任務定義。

Proxy 資源

- 對於任務定義，您必須設定 CPU 和記憶體參數。

  建議您將額外的 256 個 CPU 單位和至少 64 MiB 的記憶體新增至 Service Connect 代理容器的任務 CPU 和記憶體。在 AWS Fargate 上，您可以設定的最低記憶體容量為 512 MiB。在 Amazon EC2 上，需要任務定義記憶體。
- 對於 服務，您可以在 Service Connect 組態中設定日誌組態。

- 如果您希望此服務中的任務在尖峰負載時每秒接收的請求數超過 500 個，建議在此 Service Connect Proxy 容器的任務定義中，將 512 個 CPU 單元新增至任務 CPU。
- 如果您希望在命名空間中建立超過 100 個 Service Connect 服務，或在命名空間內所有 Amazon ECS 服務中建立總共 2000 個任務，建議在 Service Connect Proxy 容器的任務記憶體中新增 128 MiB 記憶體。您應該在命名空間中所有 Amazon ECS 服務使用的每個任務定義中執行此操作。

代理組態

您的應用程式會以與應用程式所在的相同任務連線至附屬容器中的 Proxy。Amazon ECS 會設定任務和容器，讓應用程式只有在應用程式連線到相同命名空間中的端點名稱時，才會連線到代理。所有其他流量都不會使用 Proxy。其他流量包含相同 VPC、 AWS 服務端點和外部流量中的 IP 地址。

負載平衡

服務連線會將 Proxy 設定為使用循環配置策略，在 Service Connect 端點中的任務之間進行負載平衡。位於連線來源任務中的本機 Proxy，會挑選提供端點之用戶端-伺服器服務中的其中一項任務。

例如，請考慮在名為本機的命名空間中設定為用戶端服務的服務中執行 WordPress 的任務。有另一個包含具有執行 MySQL 資料庫 2 個任務的服務。此服務設定為在相同命名空間中透過 Service Connect 提供名為 `mysql` 的端點。在 WordPress 任務中，WordPress 應用程式使用端點名稱連線至資料庫。與此名稱的連線會前往在相同任務中的附屬容器中執行的代理。然後，Proxy 可以使用循環配置策略連線至任一 MySQL 任務。

負載平衡策略：循環配置

異常值偵測

此功能會使用 Proxy 先前連線失敗的相關資料，以避免傳送新的連線至連線失敗的主機。Service Connect 會設定 Proxy 的異常值偵測功能，以提供被動運作狀態檢查。

使用先前的範例，代理可以連線至任一 MySQL 任務。如果 Proxy 與特定 MySQL 任務進行了多個連線，並且在過去的 30 秒內有 5 次以上的連線失敗，那麼 Proxy 會在 30 到 300 秒內避免該 MySQL 任務。

重試

Service Connect 會將 Proxy 設定為重試透過 Proxy 傳遞且失敗的連線，而第二次嘗試則避免使用先前連線的主機。這可確保透過 Service Connect 的每個連線都不會因為一次性原因而失敗。

重試次數：2

逾時

Service Connect 會設定 Proxy 等待用戶端-伺服器應用程式回應的最長時間。預設逾時值為 15 秒，但可以更新。

選用參數：

idleTimeout - 連線在閒置時保持作用中的時間秒數。的值會0停用 idleTimeout。

HTTP/HTTP2/GRPC 的idleTimeout預設值為 5 分鐘。

的idleTimeout預設值TCP為 1 小時。

perRequestTimeout - 等待上游以每個請求的完整回應的時間量。的值0會關閉 perRequestTimeout。這只能在應用程式容器appProtocol的 為 HTTP/HTTP2/ 時設定GRPC。預設值為 15 秒。

> **Note**
>
> 如果 idleTimeout 設定為小於 的時間perRequestTimeout，連線會在達到 idleTimeout 時關閉，而不是 perRequestTimeout。

考量事項

使用 Service Connect 時，請考慮下列事項：

- 在 Fargate 中執行的任務必須使用 Fargate Linux 平台版本 1.4.0或更新版本，才能使用 Service Connect。
- 容器執行個體上的 Amazon ECS 代理程式版本必須1.67.2或更高版本。
- 容器執行個體必須執行 Amazon ECS 最佳化 Amazon Linux 2023 AMI 版本 20230428 或更高版本，或 Amazon ECS 最佳化 Amazon Linux 2 AMI 版本 2.0.20221115，才能使用 Service Connect。除 Amazon ECS 容器代理程式外，這些版本也具有 Service Connect 代理程式。如需有關 Service Connect 代理程式的更多資訊，請參閱 GitHub 上的 Amazon ECS Service Connect 代理程式。
- 容器執行個體必須具有資源 arn:aws:ecs:*region*:*0123456789012*:task-set/*cluster*/* 的 ecs:Poll 許可。如果您使用的是 ecsInstanceRole，則不需要新增其他許可。AmazonEC2ContainerServiceforEC2Role 受管政策具有必要的許可。如需詳細資訊，請參閱Amazon ECS 容器執行個體 IAM 角色。

- Service Connect 只支援使用滾動部署的服務。
- 使用`bridge`網路模式和 Service Connect 的任務不支援`hostname`容器定義參數。
- 任務定義必須設定任務記憶體限制，才能使用 Service Connect。如需詳細資訊，請參閱Service Connect Proxy。
- 不支援設定容器記憶體限制的任務定義。

  您可以在容器上設定容器記憶體限制，但是您必須將任務記憶體限制設為大於容器記憶體限制總和的數字。任務限制中未在容器限制中配置的其他 CPU 和記憶體，會由 Service Connect Proxy 容器和其他未設定容器限制的容器使用。如需詳細資訊，請參閱Service Connect Proxy。
- 您可以設定 Service Connect 在相同區域中使用任何 AWS Cloud Map 命名空間 AWS 帳戶。
- 每個服務只能屬於一個命名空間。
- 僅支援 服務建立的任務。
- 所有端點在命名空間內必須不重複。
- 所有探索名稱在命名空間內必須不重複。
- 您必須先重新部署現有的 服務，應用程式才能解析新的端點。不會將最近部署後新增至命名空間的新端點新增至任務組態。如需詳細資訊，請參閱the section called “Service Connect 元件”。
- 刪除叢集時，Service Connect 不會刪除命名空間。您必須在 中刪除命名空間 AWS Cloud Map。
- Application Load Balancer 流量預設為在`awsvpc`網路模式下透過 Service Connect 代理程式路由。如果您希望非服務流量繞過 Service Connect 代理程式，請在 Service Connect 服務組態中使用 `ingressPortOverride` 參數。

Service Connect 不支援下列項目：

- Windows 容器
- HTTP 1.0
- 獨立任務
- 使用藍/綠和外部部署類型的服務
- Service Connect 不支援 Amazon ECS Anywhere 的 `External` 容器執行個體。
- PPv2

提供 Service Connect 的地區

Amazon ECS Service Connect 可在下列 AWS 區域使用：

區域名稱	區域
美國東部 (俄亥俄)	us-east-2
美國東部 (維吉尼亞北部)	us-east-1
美國西部 (加利佛尼亞北部)	us-west-1
美國西部 (奧勒岡)	us-west-2
非洲 (開普敦)	af-south-1
亞太區域 (香港)	ap-east-1
亞太區域 (雅加達)	ap-southeast-3
亞太區域 (孟買)	ap-south-1
亞太區域 (海德拉巴)	ap-south-2
亞太區域 (大阪)	ap-northeast-3
亞太區域 (首爾)	ap-northeast-2
亞太區域 (新加坡)	ap-southeast-1
亞太區域 (雪梨)	ap-southeast-2
亞太區域 (墨爾本)	ap-southeast-4
亞太地區 (馬來西亞)	ap-southeast-5
亞太區域 (東京)	ap-northeast-1
加拿大 (中部)	ca-central-1
加拿大西部 (卡加利)	ca-west-1
中國 (北京)	cn-north-1 （注意：此區域無法使用 TLS for Service Connect。）

區域名稱	區域
中國 (寧夏)	cn-northwest-1 （注意：此區域無法使用 TLS for Service Connect。）
歐洲 (法蘭克福)	eu-central-1
歐洲 (愛爾蘭)	eu-west-1
歐洲 (倫敦)	eu-west-2
歐洲 (巴黎)	eu-west-3
歐洲 (米蘭)	eu-south-1
歐洲 (西班牙)	eu-south-2
歐洲 (斯德哥爾摩)	eu-north-1
歐洲 (蘇黎世)	eu-central-2
以色列 (特拉維夫)	il-central-1
中東 (巴林)	me-south-1
中東 (阿拉伯聯合大公國)	me-central-1
南美洲 (聖保羅)	sa-east-1

Amazon ECS Service Connect 組態概觀

當您使用 Service Connect 時，您需要在 資源中設定參數。

下表說明 Amazon ECS 資源的組態參數。

參數位置	應用程式類型	描述	必要
任務定義	用戶端	用戶端任務定義中的 Service Connect 沒有可用的變更。	N/A

參數位置	應用程式類型	描述	必要
任務定義	用戶端-伺服器	伺服器必須將 name 欄位新增至容器 portMappings 中的連接埠。如需詳細資訊，請參閱 portMappings	是
任務定義	用戶端-伺服器	伺服器可以選擇性地提供應用程式協定 (例如 HTTP)，以接收其伺服器應用程式 (例如 HTTP 5xx) 的協定特定指標。	否
服務定義	用戶端	用戶端服務必須新增 serviceConnectConfiguration ，才能設定要加入的命名空間。此命名空間必須包含此服務需要探索的所有伺服器服務。如需詳細資訊，請參閱「serviceConnectConfiguration 」。	是
服務定義	用戶端-伺服器	伺服器服務必須新增 serviceConnectConfiguration ，才能設定可從中使用服務的 DNS 名稱、連接埠號碼和命名空間。如需詳細資訊，請參閱「serviceConnectConfiguration 」。	是
叢集	用戶端	叢集可以新增預設的 Service Connect 命名空間。在服務中設定 Service Connect 時，叢集中的新服務會繼承命名空間。	否
叢集	用戶端-伺服器	在套用至伺服器服務的叢集中，Service Connect 沒有可用的變更。伺服器任務定義和服務必須設定相應的組態。	N/A

Service Connect 的設定步驟概觀

下列步驟提供如何設定 Service Connect 的概觀。

> **Important**
>
> - Service Connect 會在您的帳戶中建立 AWS Cloud Map 服務。透過手動註冊/取消註冊執行個體、變更執行個體屬性或刪除服務來修改這些 AWS Cloud Map 資源，可能會導致應用程式流量或後續部署產生非預期的行為。
> - Service Connect 不支援任務定義中的連結。

1. 將連接埠名稱新增至任務定義中的連接埠對映。此外，您可以識別應用程式的第 7 層協定，以取得其他指標。
2. 使用 AWS Cloud Map 命名空間建立叢集，或分別建立命名空間。對於簡單的組織，請使用您要用於命名空間的名稱來建立叢集，並指定與命名空間相同的名稱。在這種情況下，Amazon ECS 會使用必要的組態建立新的 HTTP 命名空間。Service Connect 不會在 Amazon Route 53 中使用或建立 DNS 託管區域。
3. 設定服務以在命名空間內建立 Service Connect 端點。
4. 部署服務以建立端點。Amazon ECS 會將 Service Connect Proxy 容器新增至每項任務，並在 AWS Cloud Map中建立 Service Connect 端點。此容器未在任務定義中設定，而且任務定義可以重複使用而不需要修改，以在相同的命名空間或多個命名空間中建立多個服務。
5. 將用戶端應用程式部署為服務，以連線到端點。Amazon ECS 會在每項任務中透過 Service Connect Proxy，將這些應用程式連線至 Service Connect 端點。

   應用程式僅使用 Proxy 來連線至 Service Connect 端點。沒有其他設定可使用 Proxy。Proxy 會執行循環配置負載平衡、異常值偵測和重試。如需 Proxy 的詳細資訊，請參閱 Service Connect Proxy。
6. 透過 Amazon CloudWatch 中的 Service Connect Proxy 監控流量。

叢集組態

您可以在建立或更新叢集時設定 Service Connect 的預設命名空間。如果您指定的命名空間名稱在相同的 AWS 區域 和帳戶中不存在，則會建立新的 HTTP 命名空間。

如果您建立叢集並指定預設的 Service Connect 命名空間，則叢集會在 Amazon ECS 建立命名空間時在 `PROVISIONING` 狀態中等待。您可以看到 `attachment` 所處的叢集狀態顯示命名空間的狀態。根據預設，附件不會顯示於 中 AWS CLI，您必須新增 `--include ATTACHMENTS`才能查看。

服務組態

Service Connect 針對最低組態要求而設計。您必須為要在任務定義中與 Service Connect 搭配使用的每個連接埠映射設定名稱。在服務中，您需要開啟 Service Connect，然後選取命名空間來建立用戶端服務。若要建立用戶端-伺服器服務，您需要新增符合其中一個連接埠映射名稱的單一 Service Connect 服務組態。Amazon ECS 會重複使用任務定義中的連接埠號碼和連接埠名稱，以定義 Service Connect 服務和端點。若要覆寫這些值，您可以在主控台中使用 Discovery (探索)、DNS 和 Port (連接埠) 等其他參數，或分別在 Amazon ECS API 中使用 `discoveryName` 和 `clientAliases`。

加密 Amazon ECS Service Connect 流量

Amazon ECS Service Connect 支援使用 Amazon ECS 服務的 Transport Layer Security (TLS) 憑證進行自動流量加密。當您將 Amazon ECS 服務指向 AWS Private Certificate Authority (AWS Private CA) 時，Amazon ECS 會自動佈建 TLS 憑證，以加密 Amazon ECS Service Connect 服務之間的流量。Amazon ECS 會產生、輪換和分發用於流量加密的 TLS 憑證。

Service Connect 的自動流量加密使用業界領先的加密功能來保護您的服務間通訊，協助您滿足您的安全需求。它支援使用 AWS Private Certificate Authority `256-bit ECDSA`和 `2048-bit RSA`加密的 TLS 憑證。根據預設，支援 TLS 1.3，但不支援 TLS 1.0 - 1.2。您也可以完全控制私有憑證和簽署金鑰，協助您符合合規要求。

> **Note**
>
> 若要使用 TLS 1.3，您必須在目標的接聽程式上啟用它。
> 只有透過 Amazon ECS 代理程式傳遞的傳入和傳出流量才會加密。

AWS Private Certificate Authority 憑證和 Service Connect

發行憑證需要額外的 IAM 許可。Amazon ECS 提供受管資源信任政策，概述一組許可。如需此政策的詳細資訊，請參閱 AmazonECSInfrastructureRolePolicyForServiceConnectTransportLayerSecurity。

AWS Private Certificate Authority Service Connect 模式

AWS Private Certificate Authority 可以兩種模式執行：一般用途和短期。

- 一般用途 - 可設定任何過期日期的憑證。
- 短期 - 最長有效性為七天的憑證。

雖然 Amazon ECS 支援這兩種模式，但我們建議您使用短期憑證。根據預設，憑證每五天輪換一次，在短期模式下執行可大幅節省一般用途的成本。

Service Connect 不支援憑證撤銷，而是利用短期憑證，頻繁輪換憑證。您有權在 Secrets Manager 中使用受管輪換來修改輪換頻率、停用或刪除秘密，但這樣做可能會帶來下列可能後果。 https://docs.aws.amazon.com/secretsmanager/latest/userguide/intro.html

- 較短的輪換頻率 - 由於 AWS Private CA AWS KMS 和 Secrets Manager，以及 Auto Scaling 發生旋轉的工作負載增加，較短的輪換頻率會產生較高的成本。
- 較長輪換頻率 - 如果輪換頻率超過七天，您應用程式的通訊會失敗。
- 刪除秘密 - 刪除秘密會導致輪換失敗，並影響客戶應用程式通訊。

如果您的秘密輪換失敗，則會在 中發佈`RotationFailed`事件AWS CloudTrail。您也可以為 設定 CloudWatch 警示`RotationFailed`。

> **Important**
>
> 請勿將複本區域新增至秘密。這樣做可防止 Amazon ECS 刪除秘密，因為 Amazon ECS 沒有從複寫中移除區域的許可。如果您已新增複寫，請執行下列命令。
>
> ```
> aws secretsmanager remove-regions-from-replication \
>  --secret-id SecretId \
>  --remove-replica-regions region-name
> ```

次級憑證授權機構

您可以將任何 AWS Private CA根憑證或次級憑證帶到 Service Connect TLS，以發行服務的終端實體憑證。提供的發行者會被視為各地的簽署者和信任根。您可以從不同的次級 CAs 為應用程式的不同部分發行終端實體憑證。使用 時 AWS CLI，請提供 CA 的 Amazon Resource Name (ARN) 來建立信任鏈。

內部部署憑證授權機構

若要使用內部部署 CA，您可以在其中建立和設定次級 CA AWS Private Certificate Authority。這可確保為 Amazon ECS 工作負載發行的所有 TLS 憑證與您在內部部署執行的工作負載共用信任鏈，並且能夠安全地連線。

> **Important**
>
> 在 `AmazonECSManaged : true`中新增必要的標籤 AWS Private CA。

基礎設施即程式碼

使用 Service Connect TLS 搭配 Infrastructure as Code (IaC) 工具時，請務必正確設定相依性以避免問題，例如服務卡在耗盡中。如果 AWS KMS 提供金鑰，則應在 Amazon ECS 服務之後刪除 IAM 角色和 AWS Private CA 相依性。

Service Connect 和 Secrets Manager

搭配使用 Amazon ECS Service Connect 與 TLS 加密時，服務會以下列方式與 Secrets Manager 互動：

Service Connect 會使用提供的基礎設施角色，在 Secrets Manager 中建立秘密。這些秘密用於存放 TLS 憑證的相關私有金鑰，以加密 Service Connect 服務之間的流量。

> **Warning**
>
> Service Connect 自動建立和管理這些秘密可簡化為您的服務實作 TLS 加密的程序。不過，請務必注意潛在的安全性影響。具有 Secrets Manager 讀取存取權的其他 IAM 角色，可能可以存取這些自動建立的秘密。如果未正確設定存取控制，這可能會將敏感的密碼編譯材料公開給未經授權的對象。
> 若要降低此風險，請遵循下列最佳實務：
>
> - 小心管理和限制對 Secrets Manager 的存取，尤其是 Service Connect 建立的秘密。
> - 定期稽核 IAM 角色及其許可，以確保維持最低權限的原則。

授予 Secrets Manager 讀取存取權時，請考慮排除 Service Connect 建立的 TLS 私有金鑰。您可以在 IAM 政策中使用 條件來排除符合模式ARNs 的秘密，藉此達成此目的：

```
"arn:aws:secretsmanager:::secret:ecs-sc!"
```

一個範例 IAM 政策，以 `ecs-sc!`字首拒絕所有秘密`GetSecretValue`的動作：

```
{
    "Version": "2012-10-17",
```

```
    "Statement": [
        {
            "Effect": "Deny",
            "Action": "secretsmanager:GetSecretValue",
            "Resource": "arn:aws:secretsmanager:*:*:secret:ecs-sc!*"
        }
    ]
}
```

> **Note**
>
> 這是一般範例，可能需要根據您的特定使用案例和 AWS 帳戶組態進行調整。請務必徹底測試您的 IAM 政策，以確保它們提供預期的存取，同時維護安全性。

透過了解 Service Connect 如何與 Secrets Manager 互動，您可以更好地管理 Amazon ECS 服務的安全性，同時利用自動 TLS 加密的優勢。

Service Connect 和 AWS Key Management Service

您可以使用 AWS Key Management Service 來加密和解密 Service Connect 資源。 AWS KMS 是一項由 管理的服務，您可以在 AWS 其中建立和管理密碼編譯金鑰來保護您的資料。

AWS KMS 搭配 Service Connect 使用 時，您可以選擇使用為您 AWS 管理的 AWS 擁有金鑰，也可以選擇現有的 AWS KMS 金鑰。您也可以建立要使用的新 AWS KMS 金鑰。

提供您自己的加密金鑰

您可以提供自己的金鑰材料，也可以透過將自己的金鑰 AWS Key Management Service 匯入 來使用外部金鑰存放區 AWS KMS，然後在 Amazon ECS Service Connect 中指定該金鑰的 Amazon Resource Name (ARN)。

以下是範例 AWS KMS 政策。將*#####*值取代為您自己的值。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "id",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::111122223333:role/role-name"
```

```
      },
      "Action": [
        "kms:Encrypt",
        "kms:Decrypt",
        "kms:GenerateDataKey",
        "kms:GenerateDataKeyPair"
      ],
      "Resource": "*"
    }
  ]
}
```

如需金鑰政策的詳細資訊，請參閱《 AWS Key Management Service 開發人員指南》中的建立金鑰政策。

> **Note**
>
> Service Connect 僅支援對稱加密 AWS KMS 金鑰。您無法使用任何其他類型的 AWS KMS 金鑰來加密 Service Connect 資源。如需判斷金鑰是否為 AWS KMS 對稱加密金鑰的協助，請參閱識別非對稱 KMS 金鑰。

如需 AWS Key Management Service 對稱加密金鑰的詳細資訊，請參閱《 AWS Key Management Service 開發人員指南》中的對稱加密 AWS KMS 金鑰。

為 Amazon ECS Service Connect 啟用 TLS

您在建立或更新 Service Connect 服務時啟用流量加密。

使用 啟用現有命名空間中服務的流量加密 AWS Management Console

1. 發行憑證需要額外的 IAM 許可。Amazon ECS 提供受管資源信任政策，概述一組許可。如需此政策的詳細資訊，請參閱 AmazonECSInfrastructureRolePolicyForServiceConnectTransportLayerSecurity。
2. 開啟主控台，網址為 https://console.aws.amazon.com/ecs/v2。
3. 在導覽窗格中，選擇 Namespaces (命名空間)。
4. 選擇具有您要為其啟用流量加密之服務的命名空間。
5. 選擇您要啟用流量加密的服務。
6. 選擇右上角的更新服務，然後向下捲動至 Service Connect 區段。

7. 選擇在您的服務資訊下開啟流量加密以啟用 TLS。
8. 針對 Service Connect TLS 角色，選擇現有角色或建立新的角色。
9. 針對簽署者憑證授權機構，選擇現有的憑證授權機構或建立新的憑證授權機構。
10. 對於選擇 AWS KMS key，選擇 AWS 擁有和管理的金鑰，或者您可以選擇不同的金鑰。您也可以選擇建立新的。

如需使用 AWS CLI 為您的服務設定 TLS 的範例，請參閱使用 設定 Amazon ECS Service Connect AWS CLI。

驗證 Amazon ECS Service Connect 已啟用 TLS

Service Connect 在 Service Connect 代理程式啟動 TLS，並在目的地代理程式終止它。因此，應用程式程式碼永遠不會看到 TLS 互動。使用下列步驟來驗證 TLS 是否已啟用。

1. 在您的應用程式映像中包含 `openssl` CLI。
2. 在您的服務上啟用 ECS Exec，透過 SSM 連線至您的任務。或者，您可以在與服務相同的 Amazon VPC 中啟動 Amazon EC2 執行個體。
3. 從您要驗證的服務擷取任務的 IP 和連接埠。您可以在 AWS Cloud Map 主控台中擷取任務 IP 地址。資訊位於 命名空間的服務詳細資訊頁面上。
4. 使用 登入您的任何任務，`execute-command`如下列範例所示。或者，登入步驟 Amazon EC22 執行個體。

```
$ aws ecs execute-command --cluster cluster-name \
    --task task-id  \
    --container container-name \
    --interactive \
    --command "/bin/sh"
```

> **Note**
>
> 直接呼叫 DNS 名稱不會顯示憑證。

5. 在連線的 Shell 中，使用 `openssl` CLI 來驗證和檢視連接到任務的憑證。

   範例：

```
openssl s_client -connect 10.0.147.43:6379 < /dev/null 2> /dev/null \
```

```
| openssl x509 -noout -text
```

回應範例：

```
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            <serial-number>
        Signature Algorithm: ecdsa-with-SHA256
        Issuer: <issuer>
        Validity
            Not Before: Jan 23 21:38:12 2024 GMT
            Not After : Jan 30 22:38:12 2024 GMT
        Subject: <subject>
        Subject Public Key Info:
            Public Key Algorithm: id-ecPublicKey
                Public-Key: (256 bit)
                pub:
                    <pub>
                ASN1 OID: prime256v1
                NIST CURVE: P-256
        X509v3 extensions:
            X509v3 Subject Alternative Name:
                DNS:redis.yelb-cftc
            X509v3 Basic Constraints:
                CA:FALSE
            X509v3 Authority Key Identifier:
                keyid:<key-id>

            X509v3 Subject Key Identifier:
                1D:<id>
            X509v3 Key Usage: critical
                Digital Signature, Key Encipherment
            X509v3 Extended Key Usage:
                TLS Web Server Authentication, TLS Web Client Authentication
    Signature Algorithm: ecdsa-with-SHA256
        <hash>
```

使用 設定 Amazon ECS Service Connect AWS CLI

您可以為搭配 使用 Service Connect 的 Fargate 任務建立 Amazon ECS 服務 AWS CLI。

必要條件

以下是 Service Connect 先決條件：

- 確認 區域支援 Service Connect。如需詳細資訊，請參閱Regions with Service Connect。
- 確認 AWS CLI 已安裝並設定最新版本的 。如需詳細資訊，請參閱安裝或更新至最新版本的 AWS CLI。
- 您的 AWS 使用者具有 IAM AmazonECS_FullAccess 政策範例中指定的必要許可。
- 您已建立要使用的 VPC、子網路、路由表和安全群組。如需詳細資訊，請參閱the section called “建立 Virtual Private Cloud”。
- 您具有名為 `ecsTaskExecutionRole` 的任務執行角色，且 `AmazonECSTaskExecutionRolePolicy` 受管政策已附加至該角色。此角色允許 Fargate 將 NGINX 應用程式日誌和 Service Connect Proxy 日誌寫入 Amazon CloudWatch Logs。如需詳細資訊，請參閱建立任務執行 角色。

步驟 1：建立叢集

使用下列步驟建立 Amazon ECS 叢集和命名空間。

建立 Amazon ECS 叢集和 AWS Cloud Map 命名空間

1. 建立要使用的名為 `tutorial` 的 Amazon ECS 叢集。參數 `--service-connect-defaults` 會設定叢集的預設命名空間。在範例輸出中，名稱的 AWS Cloud Map 命名空間`service-connect`不存在於此帳戶中 AWS 區域，因此命名空間是由 Amazon ECS 建立。該命名空間是在帳戶的 AWS Cloud Map 中建立，並且對所有其他命名空間都可見，因此請使用可指示此用途的名稱。

   ```
   aws ecs create-cluster --cluster-name tutorial --service-connect-defaults
    namespace=service-connect
   ```

   輸出：

   ```
   {
       "cluster": {
           "clusterArn": "arn:aws:ecs:us-west-2:123456789012:cluster/tutorial",
           "clusterName": "tutorial",
           "serviceConnectDefaults": {
               "namespace": "arn:aws:servicediscovery:us-
   west-2:123456789012:namespace/ns-EXAMPLE"
   ```

```
        },
        "status": "PROVISIONING",
        "registeredContainerInstancesCount": 0,
        "runningTasksCount": 0,
        "pendingTasksCount": 0,
        "activeServicesCount": 0,
        "statistics": [],
        "tags": [],
        "settings": [
            {
                "name": "containerInsights",
                "value": "disabled"
            }
        ],
        "capacityProviders": [],
        "defaultCapacityProviderStrategy": [],
        "attachments": [
            {
                "id": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
                "type": "sc",
                "status": "ATTACHING",
                "details": []
            }
        ],
        "attachmentsStatus": "UPDATE_IN_PROGRESS"
    }
}
}
```

2. 確認已建立叢集：

```
aws ecs describe-clusters --clusters tutorial
```

輸出：

```
{
    "clusters": [
        {
            "clusterArn": "arn:aws:ecs:us-west-2:123456789012:cluster/tutorial",
            "clusterName": "tutorial",
            "serviceConnectDefaults": {
                "namespace": "arn:aws:servicediscovery:us-
west-2:123456789012:namespace/ns-EXAMPLE"
```

```
            },
            "status": "ACTIVE",
            "registeredContainerInstancesCount": 0,
            "runningTasksCount": 0,
            "pendingTasksCount": 0,
            "activeServicesCount": 0,
            "statistics": [],
            "tags": [],
            "settings": [],
            "capacityProviders": [],
            "defaultCapacityProviderStrategy": []
        }
    ],
    "failures": []
}
```

3. （選用）確認命名空間已在其中建立 AWS Cloud Map。您可以在建立時，使用 AWS Management Console 或一般 AWS CLI 組態 AWS Cloud Map。

   例如，使用 AWS CLI：

   ```
   aws servicediscovery --region us-west-2 get-namespace --id ns-EXAMPLE
   ```

   輸出：

   ```
   {
       "Namespace": {
           "Id": "ns-EXAMPLE",
           "Arn": "arn:aws:servicediscovery:us-west-2:123456789012:namespace/ns-
   EXAMPLE",
           "Name": "service-connect",
           "Type": "HTTP",
           "Properties": {
               "DnsProperties": {
                   "SOA": {}
               },
               "HttpProperties": {
                   "HttpName": "service-connect"
               }
           },
           "CreateDate": 1661749852.422,
           "CreatorRequestId": "service-connect"
   ```

```
    }
}
```

步驟 2：建立伺服器的 服務

Service Connect 功能適用於在 Amazon ECS 上進行多個應用程式的互連線。這些應用程式中至少有一個需要提供要連線的 Web 服務。在此步驟中，您要建立如下項目：

- 會使用未修改的官方 NGINX 容器映像且包含 Service Connect 組態的任務定義。
- Amazon ECS 服務定義，可設定 Service Connect 為此服務的流量提供服務探索和服務網格代理。此組態會重複使用叢集組態中的預設命名空間，以減少您為每個服務所做的服務組態數量。
- Amazon ECS 服務。該服務會使用任務定義執行一項任務，並為 Service Connect Proxy 插入額外的容器。Proxy 會在任務定義中容器連接埠映射的連接埠上接聽。在 Amazon ECS 中執行的用戶端應用程式中，用戶端任務中的 Proxy 會接聽任務定義連接埠名稱、服務探索名稱或服務用戶端別名名稱的輸出連線，以及來自用戶端別名的連接埠號碼。

使用 Service Connect 建立 Web 服務

1. 註冊與 Fargate 相容的任務定義，並使用 `awsvpc` 網路模式。請遵循下列步驟：

   a. 使用以下任務定義的內容，建立名為 `service-connect-nginx.json` 的檔案：

      此任務定義透過將 `name` 和 `appProtocol` 參數新增至連接埠映射來設定 Service Connect。使用多個連接埠時，連接埠名稱有助於您識別服務組態中的此連接埠。依預設，連接埠名稱也會用作可探索的名稱，以供命名空間中的其他應用程式使用。

      任務定義包含任務 IAM 角色，因為服務已啟用 ECS Exec。

      > **Important**
      >
      > 此任務定義使用 `logConfiguration`，將 nginx 輸出從 `stdout` 和 `stderr` 傳送到 Amazon CloudWatch Logs。此任務執行角色沒有所需的額外許可，無法建立 CloudWatch Logs 日誌群組。使用 AWS Management Console 或 在 CloudWatch Logs 中建立日誌群組 AWS CLI。如果不想將 nginx 日誌傳送到 CloudWatch Logs，您可以移除 `logConfiguration`。
      > 將任務執行角色中的 AWS 帳戶 ID 取代為您的 AWS 帳戶 ID。

```
{
    "family": "service-connect-nginx",
    "executionRoleArn": "arn:aws:iam::123456789012:role/ecsTaskExecutionRole",
    "taskRoleArn": "arn:aws:iam::123456789012:role/ecsTaskRole",
    "networkMode": "awsvpc",
    "containerDefinitions": [
        {
        "name": "webserver",
        "image": "public.ecr.aws/docker/library/nginx:latest",
        "cpu": 100,
        "portMappings": [
            {
                "name": "nginx",
                "containerPort": 80,
                "protocol": "tcp",
                "appProtocol": "http"
            }
        ],
        "essential": true,
        "logConfiguration": {
            "logDriver": "awslogs",
            "options": {
                "awslogs-group": "/ecs/service-connect-nginx",
                "awslogs-region": "region",
                "awslogs-stream-prefix": "nginx"
            }
        }
        }
    ],
    "cpu": "256",
    "memory": "512"
}
```

b. 使用 service-connect-nginx.json 檔案註冊任務定義：

```
aws ecs register-task-definition --cli-input-json file://service-connect-nginx.json
```

2. 建立 服務：

a. 使用您要建立的 Amazon ECS 服務的內容，建立名為 `service-connect-nginx-service.json` 的檔案。此範例使用上個步驟中建立的任務定義。需要 `awsvpcConfiguration`，因為任務定義範例使用 `awsvpc` 網路模式。

   當您建立 ECS 服務時，指定 Fargate 啟動類型以及支援 Service Connect 的 `LATEST` 平台版本。`securityGroups` 和 `subnets` 所屬的 VPC 必須具有使用 Amazon ECS 的需求。您可以從 Amazon VPC 主控台取得安全群組和子網路識別碼。

   此服務透過新增 `serviceConnectConfiguration` 參數來設定 Service Connect。不需要命名空間，因為叢集已設定預設命名空間。在命名空間的 ECS 中執行的用戶端應用程式使用 `portName` 和 `clientAliases` 中的連接埠連線至此服務。例如，由於 nginx 在根位置 `/` 提供歡迎頁面，因此可以使用 `http://nginx:80/` 連上此服務。未在 Amazon ECS 中執行或不在相同命名空間中的外部應用程式，可以使用任務的 IP 地址和任務定義中的連接埠號碼，透過 Service Connect Proxy 連上此應用程式。針對您的`tls`組態，`arn`為您的 `awsPcaAuthorityArn`、 `kmsKey`和 IAM `roleArn` 角色新增憑證。

   此服務會使用 `logConfiguration`，將 Service Connect Proxy 輸出從 `stdout` 和 `stderr` 傳送至 Amazon CloudWatch Logs。此任務執行角色沒有所需的額外許可，無法建立 CloudWatch Logs 日誌群組。使用 AWS Management Console 或 在 CloudWatch Logs 中建立日誌群組 AWS CLI。我們建議您建立此日誌群組，並將 Proxy 日誌儲存在 CloudWatch Logs 中。如果不想將 Proxy 日誌傳送到 CloudWatch Logs，您可以移除 `logConfiguration`。

```
{
    "cluster": "tutorial",
    "deploymentConfiguration": {
        "maximumPercent": 200,
        "minimumHealthyPercent": 0
    },
    "deploymentController": {
        "type": "ECS"
    },
    "desiredCount": 1,
    "enableECSManagedTags": true,
    "enableExecuteCommand": true,
    "launchType": "FARGATE",
    "networkConfiguration": {
        "awsvpcConfiguration": {
            "assignPublicIp": "ENABLED",
```

```
            "securityGroups": [
                "sg-EXAMPLE"
            ],
            "subnets": [
                "subnet-EXAMPLE",
                "subnet-EXAMPLE",
                "subnet-EXAMPLE"
            ]
           }
    },
    "platformVersion": "LATEST",
    "propagateTags": "SERVICE",
    "serviceName": "service-connect-nginx-service",
    "serviceConnectConfiguration": {
        "enabled": true,
        "services": [
            {
                "portName": "nginx",
                "clientAliases": [
                    {
                        "port": 80
                    }
                ],
                "tls": {
                   "issuerCertificateAuthority": {
                      "awsPcaAuthorityArn": "certificateArn"
                   },
                   "kmsKey": "kmsKey",
                   "roleArn": "iamRoleArn"
                }
            }
        ],
        "logConfiguration": {
            "logDriver": "awslogs",
            "options": {
                "awslogs-group": "/ecs/service-connect-proxy",
                "awslogs-region": "region",
                "awslogs-stream-prefix": "service-connect-proxy"
            }
        }
    },
    "taskDefinition": "service-connect-nginx"
}
```

b. 使用 service-connect-nginx-service.json 檔案建立服務：

```
aws ecs create-service --cluster tutorial --cli-input-json file://service-
connect-nginx-service.json
```

輸出：

```
{
    "service": {
        "serviceArn": "arn:aws:ecs:us-west-2:123456789012:service/tutorial/
service-connect-nginx-service",
        "serviceName": "service-connect-nginx-service",
        "clusterArn": "arn:aws:ecs:us-west-2:123456789012:cluster/tutorial",
        "loadBalancers": [],
        "serviceRegistries": [],
        "status": "ACTIVE",
        "desiredCount": 1,
        "runningCount": 0,
        "pendingCount": 0,
        "launchType": "FARGATE",
        "platformVersion": "LATEST",
        "platformFamily": "Linux",
        "taskDefinition": "arn:aws:ecs:us-west-2:123456789012:task-definition/
service-connect-nginx:1",
        "deploymentConfiguration": {
            "deploymentCircuitBreaker": {
                "enable": false,
                "rollback": false
            },
            "maximumPercent": 200,
            "minimumHealthyPercent": 0
        },
        "deployments": [
            {
                "id": "ecs-svc/3763308422771520962",
                "status": "PRIMARY",
                "taskDefinition": "arn:aws:ecs:us-west-2:123456789012:task-
definition/service-connect-nginx:1",
                "desiredCount": 1,
                "pendingCount": 0,
```

```
                "runningCount": 0,
                "failedTasks": 0,
                "createdAt": 1661210032.602,
                "updatedAt": 1661210032.602,
                "launchType": "FARGATE",
                "platformVersion": "1.4.0",
                "platformFamily": "Linux",
                "networkConfiguration": {
                    "awsvpcConfiguration": {
                        "assignPublicIp": "ENABLED",
                        "securityGroups": [
                            "sg-EXAMPLE"
                        ],
                        "subnets": [
                            "subnet-EXAMPLEf",
                            "subnet-EXAMPLE",
                            "subnet-EXAMPLE"
                        ]
                    }
                },
                "rolloutState": "IN_PROGRESS",
                "rolloutStateReason": "ECS deployment ecs-
svc/3763308422771520962 in progress.",
                "failedLaunchTaskCount": 0,
                "replacedTaskCount": 0,
                "serviceConnectConfiguration": {
                    "enabled": true,
                    "namespace": "service-connect",
                    "services": [
                        {
                            "portName": "nginx",
                            "clientAliases": [
                                {
                                    "port": 80
                                }
                            ]
                        }
                    ],
                    "logConfiguration": {
                        "logDriver": "awslogs",
                        "options": {
                            "awslogs-group": "/ecs/service-connect-proxy",
                            "awslogs-region": "us-west-2",
                            "awslogs-stream-prefix": "service-connect-proxy"
```

```
                        },
                        "secretOptions": []
                    }
                },
                "serviceConnectResources": [
                    {
                        "discoveryName": "nginx",
                        "discoveryArn": "arn:aws:servicediscovery:us-
west-2:123456789012:service/srv-EXAMPLE"
                    }
                ]
            }
        ],
        "roleArn": "arn:aws:iam::123456789012:role/aws-service-role/
ecs.amazonaws.com/AWSServiceRoleForECS",
        "version": 0,
        "events": [],
        "createdAt": 1661210032.602,
        "placementConstraints": [],
        "placementStrategy": [],
        "networkConfiguration": {
            "awsvpcConfiguration": {
                "assignPublicIp": "ENABLED",
                "securityGroups": [
                    "sg-EXAMPLE"
                ],
                "subnets": [
                    "subnet-EXAMPLE",
                    "subnet-EXAMPLE",
                    "subnet-EXAMPLE"
                ]
            }
        },
        "schedulingStrategy": "REPLICA",
        "enableECSManagedTags": true,
        "propagateTags": "SERVICE",
        "enableExecuteCommand": true
    }
}
```

您提供的 serviceConnectConfiguration 會出現在輸出的第一個部署中。當您以需要對任務進行變更的方式對 ECS 服務進行變更時，Amazon ECS 會建立新的部署。

步驟 3：確認您是否可以連線

若要確認 Service Connect 已設定且正常運作，請依照下列步驟從外部應用程式連線至 Web 服務。然後，請參閱 CloudWatch Service Connect Proxy 中建立的其他指標。

從外部應用程式連線至 Web 服務

- 使用任務 IP 地址連線至任務 IP 地址和容器連接埠

  使用 AWS CLI 取得任務 ID，方法是使用 `aws ecs list-tasks --cluster tutorial`。

  如果子網路和安全群組允許任務定義的連接埠上來自公用網際網路的流量，您可以從電腦連線到公用 IP。不過，`describe-tasks` 不提供公有 IP，因此這些步驟涉及前往 Amazon EC2 AWS Management Console 或 AWS CLI 以取得彈性網路介面的詳細資訊。

  在此範例中，相同 VPC 中的 Amazon EC2 執行個體使用的是任務的私有 IP。該應用程式是 nginx，但 `server: envoy` 標題顯示使用的是 Service Connect Proxy。Service Connect Proxy 正在任務定義中容器連接埠上接聽。

```
$ curl -v 10.0.19.50:80/
*   Trying 10.0.19.50:80...
* Connected to 10.0.19.50 (10.0.19.50) port 80 (#0)
> GET / HTTP/1.1
> Host: 10.0.19.50
> User-Agent: curl/7.79.1
> Accept: */*
>
* Mark bundle as not supporting multiuse
< HTTP/1.1 200 OK
< server: envoy
< date: Tue, 23 Aug 2022 03:53:06 GMT
< content-type: text/html
< content-length: 612
< last-modified: Tue, 16 Apr 2019 13:08:19 GMT
< etag: "5cb5d3c3-264"
< accept-ranges: bytes
< x-envoy-upstream-service-time: 0
<
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
```

```
<style>
    body {
        width: 35em;
        margin: 0 auto;
        font-family: Tahoma, Verdana, Arial, sans-serif;
    }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>

<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>

<p><em>Thank you for using nginx.</em></p>
</body>
</html>
```

檢視 Service Connect 指標

Service Connect Proxy 會在 CloudWatch 指標中建立應用程式 (HTTP、HTTP2、gRPC 或 TCP 連線) 指標。當您使用 CloudWatch 主控台時，請參閱 Amazon ECS 命名空間下 DiscoveryName、(DiscoveryName、ServiceName、ClusterName)、TargetDiscoveryName 和 (TargetDiscoveryName、ServiceName、ClusterName) 的其他指標維度。如需這些指標和維度的詳細資訊，請參閱《Amazon CloudWatch Logs 使用者指南》中的檢視可用的指標。

使用服務探索將 Amazon ECS 服務與 DNS 名稱連線

您可以選擇性地設定 Amazon ECS 服務，以使用 Amazon ECS 服務探索。服務探索使用 AWS Cloud Map API 動作來管理 Amazon ECS 服務的 HTTP 和 DNS 命名空間。如需詳細資訊，請參閱《 AWS Cloud Map 開發人員指南》中的什麼是 AWS Cloud Map。

服務探索可在下列 AWS 區域使用：

區域名稱	區域
美國東部 (維吉尼亞北部)	us-east-1
美國東部 (俄亥俄)	us-east-2
美國西部 (加州北部)	us-west-1
美國西部 (奧勒岡)	us-west-2
非洲 (開普敦)	af-south-1
亞太區域 (香港)	ap-east-1
亞太區域 (孟買)	ap-south-1
亞太區域 (海德拉巴)	ap-south-2
亞太區域 (東京)	ap-northeast-1
亞太區域 (首爾)	ap-northeast-2
亞太區域 (大阪)	ap-northeast-3
亞太區域 (新加坡)	ap-southeast-1
亞太區域 (雪梨)	ap-southeast-2
亞太區域 (雅加達)	ap-southeast-3
亞太區域 (墨爾本)	ap-southeast-4
加拿大 (中部)	ca-central-1
加拿大西部 (卡加利)	ca-west-1
中國 (北京)	cn-north-1
中國 (寧夏)	cn-northwest-1
歐洲 (法蘭克福)	eu-central-1

區域名稱	區域
歐洲 (蘇黎世)	eu-central-2
歐洲 (愛爾蘭)	eu-west-1
歐洲 (倫敦)	eu-west-2
歐洲 (巴黎)	eu-west-3
歐洲 (米蘭)	eu-south-1
歐洲 (斯德哥爾摩)	eu-north-1
以色列 (特拉維夫)	il-central-1
歐洲 (西班牙)	eu-south-2
中東 (阿拉伯聯合大公國)	me-central-1
中東 (巴林)	me-south-1
南美洲 (聖保羅)	sa-east-1
AWS GovCloud （美國東部）	us-gov-east-1
AWS GovCloud （美國西部）	us-gov-west-1

服務探索概念

服務探索包含下列元件：

- 服務探索命名空間：擁有相同網域名稱 (例如 `example.com`) 的服務探索服務的邏輯群組。此為您要向其路由流量的網域名稱。您可以使用 `aws servicediscovery create-private-dns-namespace`命令或在 Amazon ECS 主控台中建立命名空間。您可以使用 `aws servicediscovery list-namespaces` 命令來檢視由目前帳戶所建立的命名空間摘要資訊。如需 服務探索命令的詳細資訊，請參閱 AWS Cloud Map （服務探索） AWS CLI 參考指南`list-namespaces`中的 `create-private-dns-namespace`和 。
- 服務探索服務：存在於服務探索命名空間中，由命名空間的服務名稱和 DNS 設定組成。它提供以下核心元件：

  - 服務登錄檔：可讓您透過 DNS 或 AWS Cloud Map API 動作來查詢服務，並傳回可用於連線至服務的一或多個可用端點。
- 服務探索執行個體：存在於服務探索服務內，由服務目錄中每個 Amazon ECS 服務相關聯的屬性組成。
  - 執行個體屬性：針對每個設定為使用的服務探索的 Amazon ECS 服務，以下中繼資料會新增為自訂屬性：
    - **AWS_INSTANCE_IPV4** – 對於A記錄，Route 53 傳回以回應 DNS 查詢的 IPv4 地址，並在探索執行個體詳細資訊時 AWS Cloud Map 傳回，例如 `192.0.2.44`。
    - **AWS_INSTANCE_PORT**：與服務探索服務相關聯的連接埠值。
    - **AVAILABILITY_ZONE**：在其中啟動任務的可用區域。對於使用 EC2 啟動類型的任務，這是容器執行個體所在的可用區域。對於使用 Fargate 啟動類型的任務，這是彈性網路介面所在的可用區域。
    - **REGION**：任務所在的區域。
    - **ECS_SERVICE_NAME**：任務所屬的 Amazon ECS 服務名稱。
    - **ECS_CLUSTER_NAME**：任務所屬的 Amazon ECS 叢集名稱。
    - **EC2_INSTANCE_ID**：放置任務的容器執行個體的 ID。如果任務使用 Fargate 啟動類型，則不會新增這個自訂屬性。
    - **ECS_TASK_DEFINITION_FAMILY**：任務使用的任務定義系列。
    - **ECS_TASK_SET_EXTERNAL_ID**：如果任務集合是為外部部署建立，並與服務探索登錄關聯，則 `ECS_TASK_SET_EXTERNAL_ID` 屬性將包含任務集合的外部 ID。
- Amazon ECS 運作狀態檢查：Amazon ECS 會定期執行容器層級的運作狀態檢查。端點若未通過運作狀態檢查，則會從 DNS 路由中移除並標記為運作狀態不佳。

服務探索考量

使用服務探索時應考慮以下事項：

- 服務探索支援在 Fargate 上使用 1.1.0 或更新平台版本的任務。如需詳細資訊，請參閱適用於 Amazon ECS 的 Fargate 平台版本。
- 被設定為使用服務探索的服務有每個服務 1,000 項任務的限制。這是由於 Route 53 服務配額所致。
- Amazon ECS 主控台的建立服務工作流程僅支援將服務註冊到私有 DNS 命名空間。建立 AWS Cloud Map 私有 DNS 命名空間時，會自動建立 Route 53 私有託管區域。
- 必須設定 VPC DNS 屬性，以成功進行 DNS 解析。如需有關如何設定屬性的詳細資訊，請參閱 Amazon VPC 使用者指南中的 VPC 中的 DNS 支援。

- 為服務探索服務建立的 DNS 記錄總是會使用任務的私有 IP 地址註冊，而非公有 IP 地址 (即使使用公有命名空間)。
- 服務探索要求任務指定 `awsvpc`、`bridge` 或 `host` 網路模式 (不支援 none)。
- 如果您的服務任務定義使用 `awsvpc` 網路模式，您可以為每個服務任務建立任意組合的 `A` 或 SRV 紀錄。如果您使用 SRV 記錄，必須具備連接埠。
- 如果服務任務定義使用 `bridge` 或 `host` 網路模式，則 SRV 記錄是唯一受支援的 DNS 記錄類型。為每個服務任務建立 SRV 記錄。SRV 紀錄必須從任務定義中指定容器名稱和容器連接埠組合。
- 您可以在 VPC 內查詢服務探索服務的 DNS 記錄。其採用的格式為：`<service discovery service name>.<service discovery namespace>`。
- 針對服務名稱執行 DNS 查詢時，`A` 記錄會傳回一組對應到您的任務的 IP 地址。SRV 記錄會針對每個任務傳回一組 IP 地址和連接埠。
- 如果您有八個以下正常運作的記錄，Route 53 會使用所有正常記錄回應所有 DNS 查詢。
- 當所有記錄都狀況不良，Route 53 會使用最多八個狀況不良的記錄來回應 DNS 查詢。
- 您可以為負載平衡器後方的服務設定服務探索，但會將服務探索流量一律路由到任務，而不是負載平衡器。
- 服務探索不支援使用 Classic Load Balancer。
- 針對您的服務探索服務，我們建議您使用由 Amazon ECS 管理的容器層級運作狀態檢查。
  - HealthCheckCustomConfig—Amazon ECS 代替您管理運作狀態檢查。Amazon ECS 利用來自容器和運作狀態檢查的資訊和您的任務狀態，透過 AWS Cloud Map更新運作狀態。當您建立服務探索服務時，會使用 `--health-check-custom-config` 參數來指定。如需詳細資訊，請參閱 AWS Cloud Map API 參考中的 HealthCheckCustomConfig。
- 使用服務探索時建立 AWS Cloud Map 的資源必須手動清除。
- 任務和執行個體會註冊為 ，`UNHEALTHY`直到容器運作狀態檢查傳回值為止。如果運作狀態檢查通過，狀態會更新為 `HEALTHY`。如果容器運作狀態檢查失敗，則會取消註冊服務探索執行個體。

服務探索定價

使用 Amazon ECS 服務探索的客戶需要支付 Route 53 資源和 AWS Cloud Map 探索 API 操作的費用。其中包括建立 Route 53 託管區域和查詢服務登錄檔的費用。如需詳細資訊，請參閱 AWS Cloud Map 開發人員指南中的 AWS Cloud Map 定價。

Amazon ECS 會執行容器層級運作狀態檢查，並將其公開至 AWS Cloud Map 自訂運作狀態檢查 API 操作。這目前提供給客戶免費使用。如果您針對公開的任務設定其他網路運作狀態檢查，則會向您收取其費用。

建立使用 Service Discovery 的 Amazon ECS 服務

了解如何建立包含 Fargate 任務的服務，該任務搭配 使用服務探索 AWS CLI。

如需 AWS 區域 支援服務探索的清單，請參閱使用服務探索將 Amazon ECS 服務與 DNS 名稱連線。

如需支援 Fargate 的區域的資訊，請參閱the section called “AWS Fargate 區域”。

必要條件

在您開始教學課程之前，請務必先達成以下先決條件：

- AWS CLI 已安裝並設定最新版本的 。如需詳細資訊，請參閱安裝或更新至最新版本的 AWS CLI。
- 設定以使用 Amazon ECS。 中所述的步驟已完成。
- 您的 AWS 使用者具有 IAM AmazonECS_FullAccess 政策範例中指定的必要許可。
- 您已建立至少一個 VPC 和一個安全群組。如需詳細資訊，請參閱the section called “建立 Virtual Private Cloud”。

步驟 1：在 中建立 Service Discovery 資源 AWS Cloud Map

請依照下列步驟建立服務探索命名空間和服務探索服務。

1. 建立私有雲端資源服務探索命名空間。此範例會建立名為 `tutorial` 的命名空間。將 *vpc-abcd1234* 替換為您現有其中一個 VPC 的識別碼。

```
aws servicediscovery create-private-dns-namespace \
      --name tutorial \
      --vpc vpc-abcd1234
```

以下是此命令的輸出：

```
{
    "OperationId": "h2qe3s6dxftvvt7riu6lfy2f6c3jlhf4-je6chs2e"
}
```

2. 使用上一步輸出的 `OperationId`，驗證私有命名空間是否已成功建立。請記下命名空間 ID，因為您在後續命令中使用它。

```
aws servicediscovery get-operation \
```

```
      --operation-id h2qe3s6dxftvvt7riu6lfy2f6c3jlhf4-je6chs2e
```

輸出如下。

```
{
    "Operation": {
        "Id": "h2qe3s6dxftvvt7riu6lfy2f6c3jlhf4-je6chs2e",
        "Type": "CREATE_NAMESPACE",
        "Status": "SUCCESS",
        "CreateDate": 1519777852.502,
        "UpdateDate": 1519777856.086,
        "Targets": {
           "NAMESPACE": "ns-uejictsjen2i4eeg"
        }
    }
}
```

3. 使用上一步輸出中的 NAMESPACE ID，建立服務探索服務。此範例會建立名為 myapplication 的服務。請記下服務 ID 和 ARN，因為您在後續命令中使用它們。

```
aws servicediscovery create-service \
      --name myapplication \
      --dns-config "NamespaceId="ns-
uejictsjen2i4eeg",DnsRecords=[{Type="A",TTL="300"}]" \
      --health-check-custom-config FailureThreshold=1
```

輸出如下。

```
{
    "Service": {
       "Id": "srv-utcrh6wavdkggqtk",
        "Arn": "arn:aws:servicediscovery:region:aws_account_id:service/srv-
utcrh6wavdkggqtk",
        "Name": "myapplication",
        "DnsConfig": {
            "NamespaceId": "ns-uejictsjen2i4eeg",
            "DnsRecords": [
                {
                    "Type": "A",
                    "TTL": 300
                }
            ]
```

```
        },
        "HealthCheckCustomConfig": {
            "FailureThreshold": 1
        },
        "CreatorRequestId": "e49a8797-b735-481b-a657-b74d1d6734eb"
    }
}
```

步驟 2：建立 Amazon ECS 資源

請依照下列步驟建立您的 Amazon ECS 叢集、任務定義和服務。

1. 建立 Amazon ECS 叢集 此範例會建立名稱為 `tutorial` 的叢集。

```
aws ecs create-cluster \
      --cluster-name tutorial
```

2. 註冊與 Fargate 相容的任務定義，並使用 `awsvpc` 網路模式。請遵循下列步驟：

   a. 使用以下任務定義的內容，建立名為 `fargate-task.json` 的檔案：

```
{
    "family": "tutorial-task-def",
        "networkMode": "awsvpc",
        "containerDefinitions": [
            {
                "name": "sample-app",
                "image": "httpd:2.4",
                "portMappings": [
                    {
                        "containerPort": 80,
                        "hostPort": 80,
                        "protocol": "tcp"
                    }
                ],
                "essential": true,
                "entryPoint": [
                    "sh",
                    "-c"
                ],
                "command": [
```

```
                    "/bin/sh -c \"echo '<html> <head> <title>Amazon ECS Sample
 App</title> <style>body {margin-top: 40px; background-color: #333;} </style>
 </head><body> <div style=color:white;text-align:center> <h1>Amazon ECS Sample
 App</h1> <h2>Congratulations!</h2> <p>Your application is now running on a
 container in Amazon ECS.</p> </div></body></html>' >  /usr/local/apache2/
htdocs/index.html && httpd-foreground\""
                ]
            }
        ],
        "requiresCompatibilities": [
            "FARGATE"
        ],
        "cpu": "256",
        "memory": "512"
}
```

b. 使用 fargate-task.json 註冊任務定義。

```
aws ecs register-task-definition \
      --cli-input-json file://fargate-task.json
```

3. 依照下列步驟建立 ECS 服務：

a. 使用您要建立的 ECS 服務的內容，建立名為 ecs-service-discovery.json 的檔案。此範例使用上個步驟中建立的任務定義。需要 awsvpcConfiguration，因為任務定義範例使用 awsvpc 網路模式。

當您建立 ECS 服務時，指定 Fargate 啟動類型以及 LATEST 支援服務探索的平台版本。在 AWS Cloud Map 中建立服務探索服務時，registryArn 是傳回的 ARN。securityGroups 和 subnets 必須屬於用來建立 Cloud Map 命名空間的 VPC。您可以從 Amazon VPC 主控台取得安全群組和子網路識別碼。

```
{
    "cluster": "tutorial",
    "serviceName": "ecs-service-discovery",
    "taskDefinition": "tutorial-task-def",
    "serviceRegistries": [
       {
          "registryArn":
 "arn:aws:servicediscovery:region:aws_account_id:service/srv-utcrh6wavdkggqtk"
       }
    ],
```

```
    "launchType": "FARGATE",
    "platformVersion": "LATEST",
    "networkConfiguration": {
       "awsvpcConfiguration": {
          "assignPublicIp": "ENABLED",
          "securityGroups": [ "sg-abcd1234" ],
          "subnets": [ "subnet-abcd1234" ]
       }
    },
    "desiredCount": 1
}
```

b. 使用 ecs-service-discovery.json 建立您的 ECS 服務。

```
aws ecs create-service \
      --cli-input-json file://ecs-service-discovery.json
```

步驟 3：驗證 中的服務探索 AWS Cloud Map

您可以查詢服務探索資訊，以確認一切都已正確建立。設定服務探索後，您可以使用 AWS Cloud Map API 操作，或從 VPC dig 中的執行個體呼叫 。請遵循下列步驟：

1. 使用服務探索的服務 ID，來列出服務探索執行個體。請記下執行個體 ID (以粗體標示) 以進行資源清理。

```
aws servicediscovery list-instances \
      --service-id srv-utcrh6wavdkggqtk
```

輸出如下。

```
{
    "Instances": [
        {
            "Id": "16becc26-8558-4af1-9fbd-f81be062a266",
            "Attributes": {
                "AWS_INSTANCE_IPV4": "172.31.87.2"
                "AWS_INSTANCE_PORT": "80",
                "AVAILABILITY_ZONE": "us-east-1a",
                "REGION": "us-east-1",
                "ECS_SERVICE_NAME": "ecs-service-discovery",
                "ECS_CLUSTER_NAME": "tutorial",
```

```
                "ECS_TASK_DEFINITION_FAMILY": "tutorial-task-def"
            }
        }
    ]
}
```

2. 使用服務探索命名空間、服務和其他參數如 ECS 叢集名稱，來查詢服務探索執行個體的詳細資訊。

```
aws servicediscovery discover-instances \
      --namespace-name tutorial \
      --service-name myapplication \
      --query-parameters ECS_CLUSTER_NAME=tutorial
```

3. 在 Route 53 託管區域為服務探索服務建立的 DNS 記錄，可使用以下 AWS CLI 命令查詢：

   a. 使用命名空間 ID 取得命名空間的相關資訊，其中包括 Route 53 託管區域 ID。

```
aws servicediscovery \
      get-namespace --id ns-uejictsjen2i4eeg
```

   輸出如下。

```
{
    "Namespace": {
        "Id": "ns-uejictsjen2i4eeg",
        "Arn": "arn:aws:servicediscovery:region:aws_account_id:namespace/ns-
uejictsjen2i4eeg",
        "Name": "tutorial",
        "Type": "DNS_PRIVATE",
        "Properties": {
             "DnsProperties": {
                "HostedZoneId": "Z35JQ4ZFDRYPLV"
            }
        },
        "CreateDate": 1519777852.502,
        "CreatorRequestId": "9049a1d5-25e4-4115-8625-96dbda9a6093"
    }
}
```

   b. 使用上一步中的 Route 53 託管區域 ID (請參閱粗體文字)，取得託管區域的資源紀錄集。

```
aws route53 list-resource-record-sets \
      --hosted-zone-id Z35JQ4ZFDRYPLV
```

4. 您還可以使用 dig 從您的 VPC 內的執行個體查詢 DNS。

```
dig +short myapplication.tutorial
```

步驟 4：清理

當您完成此教學課程時，清除相關聯的資源以避免未使用的資源產生費用。請遵循下列步驟：

1. 使用您之前提到的服務 ID 和執行個體 ID，取消註冊服務探索服務執行個體。

```
aws servicediscovery deregister-instance \
      --service-id srv-utcrh6wavdkggqtk \
      --instance-id 16becc26-8558-4af1-9fbd-f81be062a266
```

輸出如下。

```
{
    "OperationId": "xhu73bsertlyffhm3faqi7kumsmx274n-jh0zimzv"
}
```

2. 使用來自上一步輸出的 OperationId，請驗證已成功取消註冊服務探索服務。

```
aws servicediscovery get-operation \
      --operation-id xhu73bsertlyffhm3faqi7kumsmx274n-jh0zimzv
```

```
{
  "Operation": {
        "Id": "xhu73bsertlyffhm3faqi7kumsmx274n-jh0zimzv",
        "Type": "DEREGISTER_INSTANCE",
        "Status": "SUCCESS",
        "CreateDate": 1525984073.707,
        "UpdateDate": 1525984076.426,
        "Targets": {
            "INSTANCE": "16becc26-8558-4af1-9fbd-f81be062a266",
            "ROUTE_53_CHANGE_ID": "C5NSRG1J4I1FH",
            "SERVICE": "srv-utcrh6wavdkggqtk"
```

```
        }
    }
}
```

3. 使用服務 ID 刪除服務探索服務。

```
aws servicediscovery delete-service \
      --id srv-utcrh6wavdkggqtk
```

4. 使用命名空間 ID 刪除服務探索命名空間 。

```
aws servicediscovery delete-namespace \
      --id ns-uejictsjen2i4eeg
```

輸出如下。

```
{
    "OperationId": "c3ncqglftesw4ibgj5baz6ktaoh6cg4t-jh0ztysj"
}
```

5. 使用上一步輸出的 OperationId，確認服務探索命名空間已成功刪除。

```
aws servicediscovery get-operation \
      --operation-id c3ncqglftesw4ibgj5baz6ktaoh6cg4t-jh0ztysj
```

輸出如下。

```
{
    "Operation": {
        "Id": "c3ncqglftesw4ibgj5baz6ktaoh6cg4t-jh0ztysj",
        "Type": "DELETE_NAMESPACE",
        "Status": "SUCCESS",
        "CreateDate": 1525984602.211,
        "UpdateDate": 1525984602.558,
        "Targets": {
            "NAMESPACE": "ns-rymlehshst7hhukh",
            "ROUTE_53_CHANGE_ID": "CJP2A2M86XW3O"
        }
    }
}
```

6. 將 Amazon ECS 服務所需的計數更新為 0。您必須執行此操作，才能在後續步驟中刪除服務。

```
aws ecs update-service \
      --cluster tutorial \
      --service ecs-service-discovery \
      --desired-count 0
```

7. 刪除 Amazon ECS 服務：

```
aws ecs delete-service \
      --cluster tutorial \
      --service ecs-service-discovery
```

8. 刪除 Amazon ECS 叢集：

```
aws ecs delete-cluster \
      --cluster tutorial
```

使用 Amazon VPC Lattice 連線、觀察和保護您的 Amazon ECS 服務

Amazon VPC Lattice 是一種受管應用程式聯網服務，Amazon ECS 客戶可用來觀察、保護和監控跨 AWS 運算服務、VPCs 和帳戶建置的應用程式，而無需修改其程式碼。

VPC Lattice 使用目標群組，這是運算資源的集合。這些目標會執行您的應用程式或服務，可以是 Amazon EC2 執行個體、IP 地址、Lambda 函數和 Application Load Balancer。透過將 Amazon ECS 服務與 VPC Lattice 目標群組建立關聯，客戶現在可以在 VPC Lattice 中啟用 Amazon ECS 任務做為 IP 目標。當已註冊服務的任務啟動時，Amazon ECS 會自動將任務註冊到 VPC Lattice 目標群組。

> **Note**
>
> 使用五個 VPC Lattice 組態時，您的部署時間可能會比使用較少組態時稍長。

符合條件時，接聽程式規則會用來將流量轉送至指定的目標群組。接聽程式會使用您設定的連接埠上的通訊協定來檢查連線請求。服務會根據您在設定接聽程式時定義的規則，將請求路由到其已註冊的目標。

根據 VPC Lattice 運作狀態檢查，如果任務狀態不佳，Amazon ECS 也會自動取代任務。一旦與 VPC Lattice 建立關聯，Amazon ECS 客戶也可以利用 VPC Lattice 中的許多其他跨運算連線、安全性和可觀測性功能，例如跨叢集、VPCs 和 帳戶的 服務連線 AWS Resource Access Manager、授權和身分驗證的 IAM 整合，以及進階流量管理功能。

Amazon ECS 客戶可以透過下列方式受益於 VPC Lattice。

- 提高開發人員生產力 - VPC Lattice 可讓您專注於建置功能，進而提高開發人員的生產力，而 VPC Lattice 則以統一的方式處理所有運算平台之間的聯網、安全和可觀測性挑戰。
- 更好的安全狀態 - VPC Lattice 可讓您的開發人員輕鬆驗證和保護跨應用程式和運算平台的通訊、強制執行傳輸中的加密，並使用 VPC Lattice Auth 政策套用精細存取控制。這可讓您採用更強大的安全狀態，以符合業界領先的法規和合規要求。
- 改善應用程式可擴展性和彈性 - VPC Lattice 可讓您建立部署應用程式的網路，並具有路徑、標頭和方法型路由、身分驗證、授權和監控等功能。這些優點在工作負載上沒有資源額外負荷，並且可以支援每秒產生數百萬個請求的多叢集部署，而不會增加重大延遲。
- 異質基礎設施的部署彈性 - VPC Lattice 提供所有運算服務的一致功能，例如 Amazon ECS、Fargate、Amazon EC2、Amazon EKS 和 Lambda，並讓您的組織能夠靈活地為每個應用程式選擇適當的基礎設施。

VPC Lattice 如何與其他 Amazon ECS 服務搭配使用

搭配 Amazon ECS 使用 VPC Lattice 可能會改變您使用其他 Amazon ECS 服務的方式，而其他服務則保持不變。

Application Load Balancer

您不再需要建立特定的 Application Load Balancer，以搭配 VPC Lattice 中的 Application Load Balancer 目標群組類型使用，然後連結至 Amazon ECS 服務。您只需使用 VPC Lattice 目標群組設定 Amazon ECS 服務即可。您也可以選擇同時使用 Application Load Balancer 與 Amazon ECS。

Amazon ECS 滾動部署

只有 Amazon ECS 滾動部署可與 VPC Lattice 搭配使用，Amazon ECS 會在部署期間安全地將任務帶入 ，並將其從 服務中移除。不支援程式碼部署和藍/綠部署。

若要進一步了解 VPC Lattice，請參閱 Amazon VPC Lattice 使用者指南。

建立使用 VPC Lattice 的服務

您可以使用 AWS Management Console 或 AWS CLI 來建立具有 VPC Lattice 的服務。

必要條件

在您開始教學課程之前，請務必先達成以下先決條件：

- AWS CLI 已安裝並設定最新版本的 。如需詳細資訊，請參閱安裝 AWS Command Line Interface。
- 設定以使用 Amazon ECS。 中所述的步驟已完成。
- 您的 AWS 使用者具有 IAM AmazonECS_FullAccess 政策範例中指定的必要許可。

建立搭配 使用 VPC Lattice 的服務 AWS Management Console

請依照下列步驟，使用 建立具有 VPC Lattice 的服務 AWS Management Console。

1. 開啟主控台，網址為 https://console.aws.amazon.com/ecs/v2。
2. 在導覽頁面中，選擇叢集)。
3. 在叢集頁面上，選擇要在其中建立服務的叢集。
4. 在 Services (服務) 索引標籤上，選擇 Create (建立)。

   如果您之前從未建立服務，請遵循使用主控台建立 Amazon ECS 服務中的步驟，然後在到達 VPC Lattice 區段時繼續執行這些步驟。
5. 選擇透過檢查按鈕來開啟 VPC Lattice。
6. 選擇您建立 VPC Lattice 目標群組時，已建立的 Amazon ECS 的 ECS 基礎設施角色，或選擇建立 ECS 基礎設施角色。
7. 選擇 VPC。

   VPC 取決於您在註冊任務定義時選取的聯網模式。如果您使用 `host`或 `network` 模式搭配 EC2 啟動類型，請在此處選擇您的 VPC。

   對於 `awsvpc` 模式，會根據您在網路中選擇的 VPC 自動選取 VPC，且無法變更。
8. 在目標群組下，選擇目標群組。您需要選擇至少一個目標群組，最多可以有五個目標群組。選擇新增目標群組以新增其他目標群組。為您選擇的每個目標群組選擇連接埠名稱、通訊協定和連接埠。若要刪除目標群組，請選擇移除。

   > **Note**
   >
   > - 如果您想要新增現有的目標群組，則需要使用 AWS CLI。如需如何使用 新增目標群組的說明 AWS CLI，請參閱 AWS Command Line Interface 參考中的註冊目標。
   > - 雖然 VPC Lattice 服務可以有多個目標群組，但每個目標群組只能新增至一個服務。
9. 此時，您會導覽至 VPC Lattice 主控台以繼續設定。您可以在此處將新的目標群組包含在接聽程式預設動作或現有 VPC Lattice 服務的規則中。

如需詳細資訊，請參閱 VPC Lattice 服務的接聽程式規則。

> **Important**
>
> 您需要允許安全群組或任務的傳入規則`vpc-lattice`字首，且運作狀態檢查可能會失敗。

建立搭配 使用 VPC Lattice 的服務 AWS CLI

使用 AWS CLI 建立具有 VPC Lattice 的服務。將每個*#########*替換為自己的資訊。

1. 建立目標群組組態檔案。下列範例已命名為 `tg-config.json`

```
{
    "ipAddressType": "IPV4",
    "port": 443,
    "protocol": "HTTPS",
    "protocolVersion": "HTTP1",
    "vpcIdentifier": "vpc-f1663d9868EXAMPLE"
}
```

2. 使用下列命令來建立 VPC Lattice 目標群組。

```
aws vpc-lattice create-target-group \
    --name my-lattice-target-group-ip \
    --type IP \
    --config file://tg-config.json
```

輸出範例：

```
{
    "arn": "arn:aws:vpc-lattice:us-east-2:123456789012:targetgroup/
tg-0eaa4b9ab4EXAMPLE",
    "config": {
        "healthCheck": {
            "enabled": true,
            "healthCheckIntervalSeconds": 30,
            "healthCheckTimeoutSeconds": 5,
            "healthyThresholdCount": 5,
            "matcher": {
```

```
                "httpCode": "200"
            },
            "path": "/",
            "protocol": "HTTPS",
            "protocolVersion": "HTTP1",
            "unhealthyThresholdCount": 2
        },
        "ipAddressType": "IPV4",
        "port": 443,
        "protocol": "HTTPS",
        "protocolVersion": "HTTP1",
        "vpcIdentifier": "vpc-f1663d9868EXAMPLE"
    },
    "id": "tg-0eaa4b9ab4EXAMPLE",
    "name": "my-lattice-target-group-ip",
    "status": "CREATE_IN_PROGRESS",
    "type": "IP"
}
```

3. 下列名為 *ecs-service-vpc-lattice.json # json* 檔案是用來將 Amazon ECS 服務連接至 VPC Lattice 目標群組的範例。以下範例中portName的 與您在任務定義中定義的相同。

```
{
    "serviceName": "ecs-service-vpc-lattice",
    "taskDefinition": "ecs-task-def",
        "vpcLatticeConfigurations": [
        {
            "targetGroupArn": "arn:aws:vpc-lattice:us-
west-2:123456789012:targetgroup/tg-0eaa4b9ab4EXAMPLE",
            "portName": "testvpclattice",
            "roleArn": "arn:aws:iam::123456789012:role/
ecsInfrastructureRoleVpcLattice"
        }
    ],
    "desiredCount": 5,
    "role": "ecsServiceRole"
}
```

使用下列命令來建立 Amazon ECS 服務，並使用上述 json 範例將其連接至 VPC Lattice 目標群組。

```
aws ecs create-service \
```

```
    --cluster clusterName \
    --serviceName ecs-service-vpc-lattice \
    --cli-input-json file://ecs-service-vpc-lattice.json
```

> **Note**
>
> VPC Lattice 不支援 Amazon ECS Anywhere。

保護您的 Amazon ECS 任務不會因縮減事件而終止

您可以使用 Amazon ECS 任務縮減保護，保護您的任務免於因服務自動擴展或部署而遭到縮減事件終止。

某些應用程式需要一種機制來保護關鍵任務，避免任務在使用率較低時或服務部署期間因縮減事件終止。例如：

- 您擁有佇列處理非同步應用程式，例如影片轉碼工作，即使累積服務使用率很低，其中部分任務仍需要執行數小時。
- 您有一個遊戲應用程式，可將遊戲伺服器作為 Amazon ECS 任務執行，即使所有使用者都已登出，以減少伺服器重新啟動的啟動延遲，仍需要繼續執行。
- 部署新的程式碼版本時，您需要繼續執行任務，因為重新處理的費用很昂貴。

若要保護屬於服務的任務不會在縮減事件中終止，請將 `protectionEnabled` 屬性設定為 `true`。當您`protectionEnabled`設為 true 時，任務預設會受到保護 2 小時。然後，您可以使用 `expiresInMinutes` 屬性自訂保護期。任務的保護時間最短可至 1 分鐘，最長可達 2880 分鐘 (48 小時)。

任務完成必要的工作之後，您可以將 `protectionEnabled` 屬性設定為 `false`，以便隨後的縮減事件終止任務。

任務縮減保護機制

您可以使用 Amazon ECS 容器代理程式端點或 Amazon ECS API 來設定和取得任務縮減保護。

- Amazon ECS 容器代理程式端點

我們建議將 Amazon ECS 容器代理程式端點，用於可自行判斷是否需要保護的任務。針對以佇列為基礎或工作處理的工作負載使用此方法。

當容器開始處理工作時 (例如使用 SQS 訊息)，您可以從容器內透過任務縮減保護端點路徑 `$ECS_AGENT_URI/task-protection/v1/state` 來設定 `ProtectionEnabled` 屬性。Amazon ECS 不會在縮減事件期間終止此任務。任務完成後，您可以使用相同的端點清除`ProtectionEnabled`屬性，使任務符合在後續縮減事件期間終止的資格。

如需 Amazon ECS 容器代理程式端點的詳細資訊，請參閱 Amazon ECS 任務縮減保護端點。

- Amazon ECS API

  如果您的應用程式具有可追蹤作用中任務狀態的元件，您可以使用 Amazon ECS API 來設定和擷取任務縮減保護。使用 `UpdateTaskProtection` 將一或多個任務標記為受保護。使用 `GetTaskProtection` 擷取保護狀態。

  如果應用程式以 Amazon ECS 任務的形式託管遊戲伺服器工作階段，則可使用此方法。當使用者登入伺服器 (任務) 上的工作階段時，您可以將任務標示為受保護。使用者登出後，您可以清除專門針對此任務的保護設定，或根據您希望伺服器保持閒置的需求，定期清除不再具有作用中工作階段的類似任務的保護設定。

  如需詳細資訊，請參閱《Amazon Elastic Container Service API 參考》中的 UpdateTaskProtection 和 GetTaskProtection。

您可以將這兩種方法結合起來使用。例如，使用 Amazon ECS 代理程式端點，從容器內設定任務保護，並使用 Amazon ECS API，從外部控制器服務移除對任務保護的設定。

考量事項

使用任務縮減保護之前，請考量下列幾點：

- 建議使用 Amazon ECS 容器代理程式端點，因為 Amazon ECS 代理程式具有內建的重試機制和更簡單的介面。
- 您可以為已開啟保護的任務呼叫 `UpdateTaskProtection`，以重設任務縮減保護過期期間。
- 判斷任務需要多久才能完成其必要的工作，並相應地設定 `expiresInMinutes` 屬性。如果您設定的保護過期期間超過必要的時間，則會產生費用，並會在部署新任務時面臨延遲。
- Amazon ECS 容器代理程式 `1.65.0` 或更新版本支援任務縮減保護。

使用較舊版本的 Amazon ECS 容器代理程式可在 Amazon EC2 執行個體上新增此功能的支援，方法是將代理程式更新為最新版本。如需詳細資訊，請參閱更新 Amazon ECS 容器代理程式。

- 部署考量事項：
  - 如果服務使用滾動更新，則會建立新任務，但在 protectionEnabled 設定清除或過期之前執行舊版本的任務都不會終止。您可以將部署組態中的 maximumPercentage 參數調整為可允許在舊任務受保護時建立新任務的值。
  - 如果套用了藍/綠更新，若任務有 protectionEnabled，則不會移除具有受保護任務的藍色部署。流量將轉移到出現的新任務，而較舊的任務只有在protectionEnabled清除或過期時才會移除。視 CodeDeploy 或 CloudFormation 更新的逾時而定，部署可能會逾時，而較舊的藍色任務可能仍會存在。
  - 如果您使用 CloudFormation，則更新堆疊會有 3 小時的逾時。因此，如果您將任務保護設定為 3 小時以上，則 CloudFormation 部署可能會導致失敗和復原。

    在您的舊任務受到保護期間，CloudFormation 堆疊會顯示 UPDATE_IN_PROGRESS。如果任務縮減保護遭移除或在 3 小時視窗內過期，部署將會成功並變為 UPDATE_COMPLETE 狀態。如果部署停滯在 UPDATE_IN_PROGRESS 的時間超過 3 小時，其將失敗並顯示 UPDATE_FAILED 狀態，然後會復原至舊任務集。
  - 當受保護的任務導致部署 (滾動或藍/綠) 無法進入穩定狀態，Amazon ECS 會傳送服務事件，以便您採取補救措施。嘗試更新任務的受保護狀態時，如果您收到 DEPLOYMENT_BLOCKED 錯誤訊息，則表示該服務的受保護任務比服務所需的任務計數更多。若要解決此錯誤，請執行以下其中一個動作：
    - 等待目前的任務保護過期。然後設定任務保護。
    - 判斷可以停止哪些任務。然後使用 UpdateTaskProtection，並將 protectionEnabled 選項設為 false 來完成這些任務。
    - 將服務的所需任務計數提高到超過受保護任務的數目。

任務縮減保護所需的 IAM 許可

任務必須具有具有下列許可的 Amazon ECS 任務角色：

- ecs:GetTaskProtection：允許 Amazon ECS 容器代理程式呼叫 GetTaskProtection。
- ecs:UpdateTaskProtection：允許 Amazon ECS 容器代理程式呼叫 UpdateTaskProtection。

Amazon ECS 任務縮減保護端點

Amazon ECS 容器代理程式會自動將 ECS_AGENT_URI 環境變體注入 Amazon ECS 任務的容器中，以提供與容器代理程式 API 端點互動的方法。

我們建議將 Amazon ECS 容器代理程式端點，用於可自行判斷是否需要保護的任務。

當容器開始處理工作時，您可以使用$ECS_AGENT_URI/task-protection/v1/state容器中的任務縮減保護端點路徑來設定 protectionEnabled 屬性。

在容器內使用此 URI 的 PUT 請求來設定任務縮減保護。此 URI 的 GET 請求會傳回任務的目前保護狀態。

任務縮減保護請求參數

您可以使用具有下列請求參數的 ${ECS_AGENT_URI}/task-protection/v1/state 端點來設定任務縮減保護。

ProtectionEnabled

指定 true來標記任務以進行保護。指定 false以移除保護，並讓任務符合終止資格。

類型：布林值

必要：是

ExpiresInMinutes

任務受到保護的分鐘數。您可以指定最短 1 分鐘到最長 2,880 分鐘 (48 小時)。在此期間，您的任務不會因 Service Auto Scaling 或部署中的縮減事件而終止。在此時段過後，將 protectionEnabled 參數設為 false。

如果您未指定時間，則會自動保護任務 120 分鐘 (2 小時)。

類型：整數

必要：否

下列範例示範如何設定具有不同時段的任務保護。

如何使用預設時段保護任務的範例

此範例示範如何保護預設時段為 2 小時的任務。

```
curl --request PUT --header 'Content-Type: application/json' ${ECS_AGENT_URI}/task-
protection/v1/state --data '{"ProtectionEnabled":true}'
```

如何保護任務 60 分鐘的範例

此範例示範如何使用 expiresInMinutes 參數保護任務 60 分鐘。

```
curl --request PUT --header 'Content-Type: application/json' ${ECS_AGENT_URI}/task-
protection/v1/state --data '{"ProtectionEnabled":true,"ExpiresInMinutes":60}'
```

如何保護任務 24 小時的範例

此範例示範如何使用 expiresInMinutes 參數保護任務 24 小時。

```
curl --request PUT --header 'Content-Type: application/json' ${ECS_AGENT_URI}/task-
protection/v1/state --data '{"ProtectionEnabled":true,"ExpiresInMinutes":1440}'
```

PUT 請求會傳回下列回應。

```
{
  "protection": {
    "ExpirationDate": "2023-12-20T21:57:44.837Z",
    "ProtectionEnabled": true,
    "TaskArn": "arn:aws:ecs:us-west-2:111122223333:task/1234567890abcdef0"
  }
}
```

任務縮減保護回應參數

JSON 回應中的任務縮減保護端點 ${ECS_AGENT_URI}/task-protection/v1/state 會傳回下列資訊。

ExpirationDate

任務保護將到期的時間 (以 Epoch 時間表示)。如果任務未受保護，則此值為 null。

ProtectionEnabled

任務的保護狀態。如果已針對任務啟用縮減保護，則值為 true。否則為 false。

`TaskArn`

容器所屬任務的完整 Amazon Resource Name (ARN)。

下列範例顯示了為受保護任務傳回的詳細資訊。

```
curl --request GET ${ECS_AGENT_URI}/task-protection/v1/state
```

```
{
    "protection":{
        "ExpirationDate":"2023-12-20T21:57:44Z",
        "ProtectionEnabled":true,
        "TaskArn":"arn:aws:ecs:us-west-2:111122223333:task/1234567890abcdef0"
    }
}
```

發生失敗時會傳回下列資訊。

`Arn`

任務的完整 Amazon Resource Name (ARN)。

`Detail`

與失敗相關的詳細資訊。

`Reason`

失敗的原因。

下列範例顯示了為未保護任務傳回的詳細資訊。

```
{
    "failure":{
        "Arn":"arn:aws:ecs:us-west-2:111122223333:task/1234567890abcdef0",
        "Detail":null,
        "Reason":"TASK_NOT_VALID"
    }
}
```

發生例外狀況時會傳回下列資訊。

`requestID`

造成例外狀況的 Amazon ECS API 呼叫 AWS 請求 ID。

`Arn`

任務或服務的完整 Amazon Resource Name (ARN)。

`Code`

錯誤代碼。

`Message`

錯誤訊息。

> **Note**
>
> 如果出現 `RequestError` 或 `RequestTimeout` 錯誤，則可能是聯網問題。嘗試對 Amazon ECS 使用 VPC 端點。

以下範例顯示了發生錯誤時傳回的詳細資訊。

```
{
    "requestID":"12345-abc-6789-0123-abc",
    "error":{
        "Arn":"arn:aws:ecs:us-west-2:555555555555:task/my-cluster-
name/1234567890abcdef0",
        "Code":"AccessDeniedException",
        "Message":"User: arn:aws:sts::444455556666:assumed-role/my-ecs-task-
role/1234567890abcdef0 is not authorized to perform: ecs:GetTaskProtection on resource:
 arn:aws:ecs:us-west-2:555555555555:task/test/1234567890abcdef0 because no identity-
based policy allows the ecs:GetTaskProtection action"
    }
}
```

如果 Amazon ECS 代理程式因為網路問題或 Amazon ECS 控制平面關閉等原因而無法從 Amazon ECS 端點獲得回應，則會出現下列錯誤。

```
{
  "error": {
    "Arn": "arn:aws:ecs:us-west-2:555555555555:task/my-cluster-name/1234567890abcdef0",
    "Code": "RequestCanceled",
```

```
    "Message": "Timed out calling Amazon ECS Task Protection API"
  }
}
```

當 Amazon ECS 代理程式從 Amazon ECS 中取得限流例外狀況時，會出現下列錯誤。

```
{
  "requestID": "12345-abc-6789-0123-abc",
  "error": {
    "Arn": "arn:aws:ecs:us-west-2:555555555555:task/my-cluster-name/1234567890abcdef0",
    "Code": "ThrottlingException",
    "Message": "Rate exceeded"
  }
}
```

搭配 Amazon ECS 和 Fargate 工作負載使用錯誤注入

客戶可以在 Amazon EC2 和 Fargate 上使用 Amazon ECS 進行故障注入，以測試其應用程式對特定損害案例的回應。這些測試提供可用來最佳化應用程式效能和彈性的資訊。

啟用錯誤注入時，Amazon ECS 容器代理程式允許任務存取新的錯誤注入端點。客戶必須選擇加入，才能使用故障注入，方法是將選用`enableFaultInjection`參數新增至其任務定義，並將值設定為 `true`。預設值為 `false`。

```
{
    ...
   "enableFaultInjection": true
}
```

> **Note**
>
> 故障注入僅適用於使用 `awsvpc`或 `host` 網路模式的任務。
> 故障注入不適用於 Windows。

如需如何在 中啟用故障注入的資訊 AWS Management Console，請參閱使用 主控台建立 Amazon ECS 任務定義。

您需要在 中啟用測試功能 AWS Fault Injection Service。如需詳細資訊，請參閱使用 AWS FIS aws：ecs：task 動作。

> **Note**
>
> 未使用新的 Amazon ECS 最佳化 AMIs 或使用自訂 AMIs 的客戶，需要安裝下列相依性才能使用故障注入功能：
>
> - `tc`
> - `sch_netem` 核心模組

Amazon ECS 故障注入端點

Amazon ECS 容器代理程式會自動將 `ECS_AGENT_URI` 環境變體注入 Amazon ECS 任務的容器中，以提供與容器代理程式 API 端點互動的方法。每個端點都包含 `/start`、 `/stop`和 `/status`端點。端點只接受來自已啟用故障注入之任務的請求，且每個端點每個容器每 5 秒有 1 個請求的速率限制。超過此限制會導致錯誤。

> **Note**
>
> Amazon ECS Agent `version 1.88.0+` 需要使用故障注入端點。

搭配故障注入使用的三個端點為：

- 網路黑洞連接埠端點
- 網路封包遺失端點
- 網路延遲端點

成功的請求會產生回應碼 `200`，並在您呼叫`/start`端點、`stopped`端點`/stop`和 `running`或`/status`端點`running`時顯示訊息 `not-running` 。

```
{
    "Status": <string>
}
```

失敗的請求會傳回下列其中一個錯誤代碼：

- `400` - 錯誤請求
- `409` - 錯誤注入請求與另一個執行中的錯誤衝突

- 429 - 請求已調節
- 500 - 伺服器發生意外錯誤

```
{
 "Error":  <string message>
}
```

> **Note**
>
> 一次可以注入一個網路延遲故障或一個網路封包遺失故障。嘗試注入多個結果會導致請求遭拒。

網路黑洞連接埠端點

{ECS_AGENT_URI}/fault/v1/network-blackhole-port 端點會捨棄任務網路命名空間中特定連接埠和通訊協定的傳入或傳出流量，並與兩種模式相容：

- awsvpc - 變更會套用至任務網路命名空間
- 主機 - 變更會套用至預設網路命名空間容器執行個體

{ECS_AGENT_URI}/fault/v1/network-blackhole-port/start

此端點會啟動網路黑洞連接埠故障注入，並具有下列參數：

連線埠

用於黑孔連接埠故障注入的指定連接埠。

類型：整數

必要：是

通訊協定

用於黑孔連接埠故障注入的通訊協定。

類型：字串

有效值：tcp | udp

必要：是

TrafficType

故障注入所使用的流量類型。

類型：字串

有效值：`ingress | egress`

必要：是

SourcesToFilter

受保護免於故障的 IPv4 地址或 CIDR 區塊的 JSON 陣列。

類型：字串陣列

必要：否

以下是使用`start`端點的範例請求（使用您自己的值取代*##*值）：

```
Endpoint: ${ECS_AGENT_URI}/fault/v1/network-blackhole-port/start

Http method:POST

Request payload:
{
    "Port": 1234,
    "Protocol": "tcp|udp",
    "TrafficType": "ingress|egress"
    "SourcesToFilter": ["${IP1}", "${IP2}", ...],
}
```

{ECS_AGENT_URI}/fault/v1/network-blackhole-port/stop

此端點會停止請求中指定的錯誤。此端點具有下列參數：

連線埠

受到應停止之故障影響的連接埠。

類型：整數

必要：是

通訊協定

用來停止故障的通訊協定。

類型：字串

有效值：`tcp | udp`

必要：是

TrafficType

故障注入所使用的流量類型。

類型：字串

有效值：`ingress | egress`

必要：是

以下是使用 `stop` 端點的範例請求 （使用您自己的值取代*##*值）：

```
Endpoint: ${ECS_AGENT_URI}/fault/v1/network-blackhole-port/stop

Http method: POST

Request payload:
{
    "Port": 1234,
    "Protocol": "tcp|udp",
    "TrafficType": "ingress|egress",
}
```

{ECS_AGENT_URI}/fault/v1/network-blackhole-port/status

此端點用於檢查故障注入的狀態。此端點具有下列參數：

連線埠

要檢查故障狀態的受影響連接埠。

類型：整數

必要：是

通訊協定

檢查故障狀態時使用的通訊協定。

類型：字串

有效值：`tcp | udp`

必要：是

TrafficType

故障注入所使用的流量類型。

類型：字串

有效值：`ingress | egress`

必要：是

以下是使用`status`端點的範例請求（使用您自己的值取代*##*值）：

```
Endpoint: ${ECS_AGENT_URI}/fault/v1/network-blackhole-port/status

Http method: POST

Request payload:
{
   "Port": 1234,
   "Protocol": "tcp|udp",
   "TrafficType": "ingress|egress",
}
```

網路延遲端點

`{ECS_AGENT_URI}/fault/v1/network-latency` 端點將延遲和抖動新增至任務的網路介面，以將流量傳送至特定來源。端點與兩種模式相容：

- awsvpc - 變更會套用至任務網路介面
- 主機 - 變更會套用至預設網路介面

{ECS_AGENT_URI}/fault/v1/network-latency/start

此`/start`端點會開始網路延遲故障注入，並具有下列參數：

DelayMilliseconds

要新增至網路介面以用於故障注入的延遲毫秒數。

類型：整數

必要：是

JitterMilliseconds

要新增至網路介面以用於故障注入的抖動毫秒數。

類型：整數

必要：是

來源

IPv4 地址或 CIDR 區塊的 JSON 陣列，其目的地為 ，可與故障注入搭配使用。

類型：字串陣列

必要：是

SourcesToFilter

受保護免於故障的 IPv4 地址或 CIDR 區塊的 JSON 陣列。 SourcesToFilter 的優先順序高於 Sources。

類型：字串陣列

必要：否

以下是使用 /start 端點的範例請求 （使用您自己的值取代##值）：

```
Endpoint: ${ECS_AGENT_URI}/fault/v1/network-latency/start

Http method: POST

Request payload:
{
    "DelayMilliseconds": 123,
    "JitterMilliseconds": 123,
    "Sources": ["${IP1}", "${IP2}", ...],
    "SourcesToFilter": ["${IP1}", "${IP2}", ...],
}
```

{ECS_AGENT_URI}/fault/v1/network-latency/stop 和 /status

{ECS_AGENT_URI}/fault/v1/network-latency/stop 端點會停止故障，會{ECS_AGENT_URI}/fault/v1/network-latency/status檢查故障的狀態。

以下是使用 /stop和 /status端點的兩個範例請求。兩者都使用 POST HTTP方法。

```
Endpoint: ${ECS_AGENT_URI}/fault/v1/network-latency/stop
```

```
Endpoint: ${ECS_AGENT_URI}/fault/v1/network-latency/status
```

網路封包遺失端點

{ECS_AGENT_URI}/fault/v1/network-packet-loss 端點會將封包遺失新增至指定的網路介面。此端點與兩種模式相容：

- awsvpc - 變更會套用至任務網路介面
- 主機 - 變更會套用至預設網路介面

{ECS_AGENT_URI}/fault/v1/network-packet-loss/start

此/start端點會開始網路封包遺失故障注入，並具有下列參數：

LossPercent

封包遺失的百分比

類型：整數

必要：是

來源

用於故障注入測試的 IPv4 地址或 CIDR 區塊的 JSON 陣列。

類型：字串陣列

必要：是

SourcesToFilter

受保護免於故障的 IPv4 地址或 CIDR 區塊的 JSON 陣列。 SourcesToFilter 的優先順序高於 Sources。

類型：字串陣列

必要：否

以下是使用start端點的範例請求 （ 使用您自己的值取代##值 ）：

```
Endpoint: ${ECS_AGENT_URI}/fault/v1/network-packet-loss/start

Http method: POST

{
    "LossPercent": 6,
    "Sources": ["${IP1}", "${IP2}", ...],
    "SourcesToFilter": ["${IP1}", "${IP2}", ...],
}
```

{ECS_AGENT_URI}/fault/v1/network-packet-loss/stop 和 /status

{ECS_AGENT_URI}/fault/v1/network-packet-loss/stop 端點會停止故障，並{ECS_AGENT_URI}/fault/v1/network-packet-loss/status檢查故障的狀態。一次僅支援每種類型的故障之一。

以下是使用 /stop和 /status端點的兩個範例請求。兩者都使用 POST HTTP方法。

```
Endpoint: ${ECS_AGENT_URI}/fault/v1/network-packet-loss/stop
```

```
Endpoint: ${{ECS_AGENT_URI}/fault/v1/network-packet-loss/status
```

Amazon ECS 服務調節邏輯

Amazon ECS 服務排程器包含的邏輯，可在任務重複啟動失敗的情況下，針對服務任務啟動的頻率進行調節。

如果服務的任務重複無法進入 RUNNING 狀態 （ 直接從 進入 PENDING STOPPED 狀態 ），則後續重新啟動嘗試之間的時間會遞增，最多增加 27 分鐘。此最長期間將來可能會有所變更。此行為能降低失敗的任務對 Amazon ECS 叢集資源或 Fargate 基礎設施成本所造成的影響。如果您的服務啟動了限流邏輯，您會收到以下的服務事件訊息：

```
(service service-name) is unable to consistently start tasks successfully.
```

Amazon ECS 永遠不會阻止失敗的服務重試。除了增加重新啟動的時間間隔，它也不會嘗試對其進行任何修改。服務限流邏輯沒有任何可供使用者調校的參數。

如果您更新服務使用新的任務定義，您的服務就會立即回復到正常、非調節的狀態。如需詳細資訊，請參閱使用主控台更新 Amazon ECS 服務。

以下是啟動此邏輯的一些常見原因。我們建議您採取手動動作來解決問題：

- 您的叢集中缺乏裝載任務的資源，例如連接埠、記憶體或 CPU 單位。在這種情況下，您也會看到「資源不足服務事件訊息」。
- Amazon ECS 容器代理程式無法提取您的任務 Docker 映像檔。這可能是因為錯誤的容器映像名稱、映像、或標籤，或缺乏私有登錄檔身分驗證或許可。在這種情況下，您也會在「CannotPullContainerError停止的任務錯誤」中看到。
- 您的容器執行個體磁碟空間不足以建立容器。在這種情況下，您也會在「CannotCreateContainerError停止的任務錯誤」中看到。如需詳細資訊，請參閱在 Amazon ECS API error (500): devmapper中對 Docker 進行故障診斷。

> **Important**
>
> 到達 RUNNING 狀態之後停止的任務不會啟動限流邏輯或相關聯的服務事件訊息。例如，假設服務的 Elastic Load Balancing 運作狀態檢查失敗導致任務標示為運作狀態不佳，而 Amazon ECS 取消註冊並停止了任務。這樣並不會觸發限流。即使任務的容器指令立即以非零結束代碼結束，任務仍已轉換為 RUNNING 狀態。因為指令錯誤而立即失敗的任務不會觸發限流或服務事件訊息。

Amazon ECS 服務定義參數

服務定義會定義您的 Amazon ECS 服務的執行方式。下列參數可以在服務定義中指定。

啟動類型

`launchType`

類型：字串

有效值：`EC2` | `FARGATE` | `EXTERNAL`

必要：否

您服務執行所在的啟動類型。如果未指定啟動類型，預設會使用預設的 `capacityProviderStrategy`。如需詳細資訊，請參閱Amazon ECS啟動類型。

若有指定 `launchType`，則必須省略 `capacityProviderStrategy` 參數。

容量提供者策略

`capacityProviderStrategy`

類型： 物件陣列

必要：否

要用於服務的容量提供者策略。

容量提供者策略包含一個或多個容量提供者，以及指派給這些容量提供者的 `base` 與 `weight`。容量提供者必須與要在容量提供者策略中使用的叢集相關聯。PutClusterCapacityProviders API 是用於將容量提供者與叢集相關聯。只能使用具有 `ACTIVE` 或 `UPDATING` 狀態的容量提供者。

若有指定 `capacityProviderStrategy`，則必須省略 `launchType` 參數。如果沒有指定 `capacityProviderStrategy` 或 `launchType`，則會使用叢集的 `defaultCapacityProviderStrategy`。

如果您想要指定使用 Auto Scaling 群組的容量提供者，則必須已經建立該容量提供者。可以使用 CreateCapacityProvider API 操作來建立新的容量提供者。

若要使用 AWS Fargate 容量提供者，請指定 `FARGATE`或 `FARGATE_SPOT` 容量提供者。 AWS Fargate 容量提供者適用於所有帳戶，且只需要與要使用的叢集建立關聯。

PutClusterCapacityProviders API 操作是用來在建立叢集之後，更新叢集的可用容量提供者清單。

`capacityProvider`

類型：字串

必要：是

容量提供者的簡短名稱或完整 Amazon Resource Name (ARN)。

`weight`

類型：整數

有效範圍：介於 0 到 1,000 之間的整數。

必要：否

加權值會指定應使用指定容量提供者其已啟動任務總數的相對百分比。

例如，假設您的策略包含兩個容量提供者，且兩者的加權值都是一。當基本條件符合時，工作會平均分配給兩個容量提供者。以此類推，假設您為 capacityProviderA 指定了加權值 1；並為 capacityProviderB 指定了加權值 4。那麼，每有一個任務使用 capacityProviderA 執行，就會有四個任務是使用 capacityProviderB 執行。

`base`

類型：整數

有效範圍：介於 0 到 100,000 之間的整數。

必要：否

基礎值指定至少要在指定容量提供者上執行多少任務數量。容量提供者策略中只有一個容量提供者可以定義基礎。

任務定義

`taskDefinition`

類型：字串

必要：否

`family` 和 `revision` (`family:revision`) 或在您服務中執行的任務定義完整 Amazon Resource Name (ARN)。如果 `revision` 未指定，則會使用指定系列的最新 `ACTIVE` 修訂版。

在使用循環更新 (ECS) 部署控制器時，必須指定任務定義。

平台作業系統

`platformFamily`

類型：字串

必要：有條件

預設：Linux

在 Fargate 上託管的 Amazon ECS 服務需要此參數。

Amazon EC2 上託管的 Amazon EC2 服務會忽略此參數。

執行服務的容器上的作業系統。有效值為 LINUX、WINDOWS_SERVER_2019_FULL、WINDOWS_SERVER_2019_CORE、WINDOWS_SERVER_2022_ 與 WINDOWS_SERVER_2022_CORE。

為服務指定的每個任務的 platformFamily 值必須與服務 platformFamily 值相符。例如，如果您將 platformFamily 設定為 WINDOWS_SERVER_2019_FULL，則所有任務的 platformFamily 值都必須為 WINDOWS_SERVER_2019_FULL。

平台版本

`platformVersion`

類型：字串

必要：否

您服務中任務正在其上執行的平台版本。平台版本僅會為使用 Fargate 啟動類型的任務指定。如果尚未指定，預設會使用最新版本 (LATEST)。

AWS Fargate 平台版本用於參考 Fargate 任務基礎設施的特定執行期環境。在您執行任務或建立服務時指定 LATEST 平台版本，即可取得任務所能使用的最新平台版本。當您擴展服務規模時，這些任務會收到於服務的目前部署上所指定的平台版本。如需詳細資訊，請參閱適用於 Amazon ECS 的 Fargate 平台版本。

> **Note**
>
> 並未針對使用 EC2 啟動類型的任務指定平台版本。

叢集

`cluster`

類型：字串

必要：否

您服務執行所在之叢集的 Amazon Resource Name (ARN) 簡稱或全稱。若您未指定叢集，即假設您會使用 `default` 叢集。

服務名稱

`serviceName`

類型：字串

必要：是

您的服務名稱。可以包含最多可達 255 個字元 (大小寫)、數字、連字號和底線。叢集中不得有相同的服務名稱，但一個區域內或多個區域間的多個叢集中可以有類似的服務名稱。

排程策略

`schedulingStrategy`

類型：字串

有效值：`REPLICA` | `DAEMON`

必要：否

使用的排程策略。如果未指定排程策略，則會使用 `REPLICA` 策略。如需詳細資訊，請參閱Amazon ECS 服務。

現已推出兩個服務排程器策略概念：

- `REPLICA` - 複本排程策略會置放並在整個叢集中維持所需的任務數量。根據預設，服務排程器會將任務分散至各個可用區域。您可以使用任務置放策略和限制條件，來自訂任務置放決策。如需詳細資訊，請參閱複本策略。
- `DAEMON` - 常駐程式排程策略會在每個符合您於叢集中所指定所有任務置放限制條件的作用中容器執行個體上，準確地部署一個任務。使用這項策略時，不需指定所需的任務數量、任務置放策略或使用 Service Auto Scaling 政策。如需詳細資訊，請參閱協助程式策略。

> **Note**
>
> Fargate 任務不支援 `DAEMON` 排程策略。

所需計數

`desiredCount`

類型：整數

必要：否

於指定的任務定義中，要放置在您服務上並保持執行的實例數量。

如果使用 REPLICA 排程策略，則需要此參數。如果該服務使用 DAEMON 排程策略，則此參數是選擇性的。

部署組態

`deploymentConfiguration`

類型：物件

必要：否

選用的部署參數，可控制在部署期間執行多少任務以及停止和啟動任務的順序。

`maximumPercent`

類型：整數

必要：否

如果服務使用滾動更新 (ECS) 部署類型，則 `maximumPercent` 參數代表在部署期間，服務中允許處於 RUNNING、STOPPING 或 PENDING 狀態的任務數量上限。以百分比表示 `desiredCount` 即無條件捨去至最接近的整數。您可以使用此參數來定義部署批次大小。例如，如果您的服務使用 REPLICA服務排程器，且有四個任務`desiredCount`的 ，且`maximumPercent`值為 200%，則排程器會在停止四個較舊任務之前啟動四個新任務。前提是有執行此操作所需的叢集資源。使用 REPLICA 服務排程器的服務，其預設 `maximumPercent` 值為 200%。

Amazon ECS 排程器會使用此參數來取代運作狀態不佳的任務，方法是先啟動替代任務，然後停止運作狀態不佳的任務，只要有可用的叢集資源即可啟動替代任務。如需排程器如何取代運作狀態不佳任務的詳細資訊，請參閱Amazon ECS 服務。

如果您的服務使用 DAEMON 服務排程器類型，maximumPercent 應該維持 100%。這是預設值。

部署期間的任務最大數量等於 desiredCount 乘以 maximumPercent/100，無條件捨去到最接近的整數值。

如果服務使用藍/綠 (CODE_DEPLOY) 或使用 EC2 啟動類型的EXTERNAL部署類型和任務，則最大百分比值會設為預設值。值用於定義在容器執行個體處於 RUNNING 狀態時，服務中保持 DRAINING 狀態的任務數量上限。

> **Note**
>
> 您無法為使用藍/綠 (CODE_DEPLOY) 或EXTERNAL部署類型，且具有使用 EC2 啟動類型的任務的服務指定自訂maximumPercent值。

如果服務使用藍/綠 (CODE_DEPLOY) 或EXTERNAL部署類型，且服務中的任務使用 Fargate 啟動類型，則不會使用最大百分比值。描述您的服務時，仍會傳回該值。

minimumHealthyPercent

類型：整數

必要：否

如果服務使用滾動更新 (ECS) 部署類型，則 minimumHealthyPercent 表示部署期間必須保持在 RUNNING 狀態的服務任務數量下限。這表示為無條件進位到最接近整數的 desiredCount 的百分比。您可以使用此參數，而無須使用額外的叢集容量進行部署。

例如，如果您的服務有四個任務desiredCount的 ，即 50% minimumHealthyPercent的 和 100% maximumPercent的 ，則服務排程器會停止兩個現有任務，以在啟動兩個新任務之前釋放叢集容量。

如果任何任務運作狀態不佳，且如果 maximumPercent 不允許 Amazon ECS 排程器啟動替代任務，排程器會one-by-one停止運作狀態不佳的任務，使用 minimumHealthyPercent做為限制條件，以清除啟動替代任務的容量。如需排程器如何取代運作狀態不佳任務的詳細資訊，請參閱Amazon ECS 服務。

對於不使用負載平衡器的服務，請考慮下列事項：

- 如果服務中任務內的所有必要容器都通過了運作狀態檢查，則服務就會被視為狀態良好。

- 如果工作沒有已定義運作狀態檢查的必要容器，則服務排程器會在任務達到 RUNNING 狀態時等待 40 秒，然後再將任務計入運作狀態最低百分比總計。
- 如果任務有一或多個已定義運作狀態檢查的必要容器，服務排程器會等待任務達到正常狀態，然後將其計算為運作狀態最低百分比總計。當任務內的所有必要容器都通過其運作狀態檢查時，工作就會被視為狀態良好。服務排程器可以等待的時間，是由容器運作狀態檢查設定所決定。如需詳細資訊，請參閱運作狀態檢查。

對於會使用負載平衡器的服務，請考慮下列事項：

- 如果任務沒有定義運作狀態檢查的必要容器，則服務排程器會等待負載平衡器目標群組運作狀態檢查回傳狀況良好狀態，然後將任務計算為良好狀態最低百分比總計。
- 如果工作具有已定義運作狀態檢查的必要容器，服務排程器會等待工作達到正常狀態，以及負載平衡器目標群組運作狀態檢查回傳狀況良好狀態，然後再將工作計算為最低良好狀態百分比總計。

`minimumHealthyPercent` 的複本服務預設值是 100%。使用服務排程DAEMON的服務`minimumHealthyPercent`預設值為 0% AWS CLI、 AWS SDKs 和 APIs，以及 50% AWS Management Console。

部署期間運作良好任務的最低數量等於 `desiredCount` 乘以 `minimumHealthyPercent/100`，無條件進位到最接近的整數值。

如果服務使用藍/綠 (`CODE_DEPLOY`) 或`EXTERNAL`部署類型，且正在執行使用 EC2 啟動類型的任務，則運作狀態最低百分比值會設為預設值。值用於定義在容器執行個體處於 `RUNNING` 狀態時，服務中保持 `DRAINING` 狀態之任務數量的下限。

> **Note**
>
> 您無法為使用藍/綠 (`CODE_DEPLOY`) 或`EXTERNAL`部署類型，且具有使用 EC2 啟動類型的任務的服務指定自訂`maximumPercent`值。

若服務使用的是藍/綠 (`CODE_DEPLOY`) 或 `EXTERNAL` 部署類型，並且執行使用 Fargate 啟動類型的任務，則不使用運作良好百分比下限值，雖然描述服務時會傳回此值。

部署控制器

`deploymentController`

類型：物件

必要：否

用於服務的部署控制器。如果沒有指定部署控制器，則使用 ECS 控制器。如需詳細資訊，請參閱Amazon ECS 服務。

type

類型：字串

有效值：ECS | CODE_DEPLOY | EXTERNAL

必要：是

要使用的部署控制器類型。部署控制器有三種類型：

ECS

輪流更新 (ECS) 部署類型涉及以最新版本替換目前執行的容器版本。調整在服務部署期間允許的運作良好任務的最小和最大數量，可以控制 Amazon ECS 在滾動式更新期間新增或移除的容器數量，如同 deploymentConfiguration 中所指定。

CODE_DEPLOY

藍/綠部署類型 (CODE_DEPLOY) 使用 CodeDeploy 支援的藍/綠部署模式，讓您可以在傳送生產流量之前驗證新部署的服務。

EXTERNAL

當您想要使用任何第三方部署控制器，以完全控制 Amazon ECS 服務的部署程序時，請使用外部部署類型。

任務置放

placementConstraints

類型： 物件陣列

必要：否

您服務的任務要使用的置放限制物件陣列。每項任務您最多可以指定 10 項限制。此限制包含任務定義中的限制以及執行時間指定的限制。若您使用的是 Fargate 啟動類型，則不支援任務置放限制條件。

type

類型：字串

必要：否

限制類型。使用 distinctInstance 以確保特定群組中的每個任務執行於不同的容器執行個體。使用 memberOf，以將選擇限制於特定群組或有效的待選項目。任務定義並不支援 distinctInstance 值。

expression

類型：字串

必要：否

限制所套用的叢集查詢語言運算式。如果限制條件類型為 distinctInstance，則無法指定運算式。如需詳細資訊，請參閱建立表達式以定義 Amazon ECS 任務的容器執行個體。

placementStrategy

類型： 物件陣列

必要：否

您服務的任務要使用的置放策略物件。每項服務您最多可以指定四項策略規則。

type

類型：字串

有效值：random | spread | binpack

必要：否

置放策略類型。random 置放策略會將任務隨機置放於各個可用的待選項目。spread 置放策略根據 field 參數，以平均分散的方式置放於各個可用的待選項目。binpack 策略會根據以 field 參數指定的結果，將任務置放於資源最少的可用待選項目。例如，如果您在記憶體上進行 binpack，任務會放置在剩餘記憶體最少，但仍足以執行任務的執行個體上。

field

類型：字串

必要：否

套用置放策略的欄位。若是 spread 置放策略，有效的值為 instanceId (或擁有相同效果的 host)，或任何平台以及套用至 attribute:ecs.availability-zone 等容器執行個體的自訂屬性。若是 binpack 置放策略，有效值為 cpu 與 memory。若是 random 置放策略，此欄位並不使用。

標籤

tags

類型： 物件陣列

必要：否

您套用到服務以協助您分類和組織的中繼資料。每個標籤皆包含由您定義的一個金鑰與一個選用值。刪除服務時，也會一併刪除標籤。服務最多可套用 50 個標籤。如需詳細資訊，請參閱標記 Amazon ECS 資源。

key

類型：字串

長度限制：長度下限為 1。長度上限為 128。

必要：否

組成標籤的鍵值組的一部分。索引鍵是一般標籤，作用就像更特定標籤值的類別。

value

類型：字串

長度限制：長度下限為 0。長度上限為 256。

必要：否

組成標籤的鍵值組的選用部分。值就像標籤類別 (索引鍵) 內的描述項。

enableECSManagedTags

類型：布林值

有效值：`true | false`

必要：否

指定是否對服務內的任務使用 Amazon ECS 受管標籤。如未指定任何值，則預設值為 `false`。如需詳細資訊，請參閱使用標籤計費。

`propagateTags`

類型：字串

有效值：`TASK_DEFINITION | SERVICE`

必要：否

指定是否將標籤從任務定義或服務複製到服務中的任務。如果沒有指定值，則不會複製標籤。標籤只能在建立服務期間複製到服務內的任務。若要在建立服務或任務後對任務新增標籤，請使用 `TagResource` API 動作。

網路組態

`networkConfiguration`

類型：物件

必要：否

服務的網路組態。使用 `awsvpc` 網路模式的任務定義需要此參數，才能接收其自己的彈性網路介面，其他網路模式不支援此參數。如果使用 Fargate 啟動類型，則需要 `awsvpc` 網路模式。如需 Amazon EC2 啟動類型聯網的詳細資訊，請參閱EC2 啟動類型的 Amazon ECS 任務聯網選項。如需 Fargate 啟動類型聯網的詳細資訊，請參閱 Fargate 啟動類型的 Amazon ECS 任務聯網選項。

`awsvpcConfiguration`

類型：物件

必要：否

代表任務或服務之子網路和安全群組的物件。

`subnets`

類型：字串陣列

必要：是

與任務或服務相關聯的子網路。根據 awsvpcConfiguration 限制僅能指定 16 個子網路。

securityGroups

類型：字串陣列

必要：否

與任務或服務相關聯的安全群組。如果您未指定安全群組，則會使用預設的 VPC 安全群組。根據 awsvpcConfiguration 限制僅能指定五個安全群組。

assignPublicIP

類型：字串

有效值：ENABLED | DISABLED

必要：否

無論任務的彈性網路介面是否收到公有 IP 地址。如果未指定任何值，則會使用 DISABLED 的預設值。

healthCheckGracePeriodSeconds

類型：整數

必要：否

Amazon ECS 服務排程器在任務進入 RUNNING 狀態後，應忽略運作狀態不佳的 Elastic Load Balancing 目標運作狀態檢查、容器運作狀態檢查、VPC Lattice 運作狀態檢查和 Route 53 運作狀態檢查的期間，以秒為單位。這僅在您的服務已設定為使用負載平衡器時才會有效。如果您的服務已定義負載平衡器，且您未指定運作狀態檢查寬限期值，這時將會使用 0 的預設值。

若您的服務任務啟動及回應運作狀態檢查需要一些時間，您可以指定運作狀態檢查之寬限期間最高達 2,147,483,647 秒。在此期間，ECS 服務排程器會忽略運作狀態檢查狀態。此寬限期可防止 ECS 服務排程器將任務標示為狀況不良並在其來得及標示前加以停止。

如果您不使用 Elastic Load Balancing，建議您在任務定義運作狀態檢查參數中使用 startPeriod。如需詳細資訊，請參閱使用容器運作狀態檢查判斷 Amazon ECS 任務運作狀態。

`loadBalancers`

類型： 物件陣列

必要：否

用來代表您的服務所使用之負載平衡器的負載平衡器物件。針對使用 Application Load Balancer 或 Network Load Balancer 的任務，您最多僅能連接五個目標群組到一個服務。

建立服務之後，負載平衡器組態無法從 AWS Management Console變更。您可以使用 AWS Copilot AWS CloudFormation AWS CLI 或 SDK 來修改ECS滾動部署控制器的負載平衡器組態，而不是 AWS CodeDeploy 藍/綠或外部。當您新增、更新或移除負載平衡器組態時，Amazon ECS 會使用更新後的 Elastic Load Balancing 組態啟動新的部署。這會導致任務向負載平衡器註冊和取消註冊。我們建議您在更新 Elastic Load Balancing 組態之前，先在測試環境中驗證。如需有關如何修改組態的資訊，請參閱 Amazon Elastic Container Service API 參考 中的 UpdateService。

針對 Application Load Balancer 和 Network Load Balancer，此物件必須包含負載平衡器目標群組 ARN、容器名稱 (這會出現在容器定義中) 和容器連接埠，以從負載平衡器存取。當此服務的任務置於容器執行個體，該容器執行個體和連接埠組合就會註冊為指定之目標群組中的目標。

`targetGroupArn`

類型：字串

必要：否

與服務相關聯的 Elastic Load Balancing 目標群組的完整 Amazon Resource Name (ARN)。

僅在使用 Application Load Balancer 或Network Load Balancer時，才指定目標群組 ARN。

`loadBalancerName`

類型：字串

必要：否

要與服務建立關聯的負載平衡器名稱。

如果您使用 Application Load Balancer 或 Network Load Balancer，請略去負載平衡器名稱參數。

`containerName`

類型：字串

必要：否

要與負載平衡器建立關聯的容器 (這會出現在容器定義中) 名稱。

`containerPort`

類型：整數

必要：否

要與負載平衡器建立關聯之容器的連接埠。此連接埠必須對應至服務中任務所使用之任務定義中的 `containerPort`。針對使用 EC2 啟動類型的任務，容器執行個體必須允許傳入連接埠對應的 `hostPort` 的流量。

`role`

類型：字串

必要：否

IAM 角色的簡稱或完整 ARN，此角色允許 Amazon ECS 代您呼叫您的負載平衡器。只有在您對服務使用具有單一目標群組的負載平衡器，且您的任務定義沒有使用 `awsvpc` 網路模式時，才允許此參數。如果您指定 `role` 參數，您還必須以 `loadBalancers` 參數指定負載平衡器物件。

如果您指定的角色有 `/` 以外的路徑，則您必須指定完整的角色 ARN (建議項目) 或在角色名稱前加上路徑。例如，如果名稱為 `bar` 的角色有路徑 `/foo/`，則您可指定 `/foo/bar` 為角色名稱。如需詳細資訊，請參閱 IAM 使用者指南中的易記名稱和路徑。

> **Important**
>
> 若您的帳戶已建立 Amazon ECS 服務連結角色，則根據預設會為您的服務使用該角色，除非您在此處另行指定。若您的任務定義使用 awsvpc 網路模式，則服務連結角色為必要項目，此時您便不應在此處指定角色。如需詳細資訊，請參閱使用 Amazon ECS 的服務連結角色。

`serviceConnectConfiguration`

類型：物件

必要：否

可讓此服務探索及連線至服務，並可供命名空間內其他服務探索及連線的組態。

如需詳細資訊，請參閱使用 Service Connect 以短名稱連接 Amazon ECS 服務。

`enabled`

類型：布林值

必要：是

指定是否要搭配此服務使用 Service Connect。

`namespace`

類型：字串

必要：否

用於 Service Connect 之 AWS Cloud Map 命名空間的簡短名稱或完整 Amazon Resource Name (ARN)。命名空間必須位於與 Amazon ECS 服務和叢集相同的 AWS 區域 。命名空間的類型不會影響 Service Connect。如需詳細資訊 AWS Cloud Map，請參閱《 AWS Cloud Map 開發人員指南》中的使用 服務。

`services`

類型： 物件陣列

必要：否

Service Connect 服務物件的陣列。這些是其他 Amazon ECS 服務用來連線到此服務的名稱和別名 (又稱為端點)。

對於屬於命名空間成員且僅連線至命名空間內其他服務的「用戶端」Amazon ECS 服務來說，此欄位不是必要欄位。以接受傳入請求的前端應用程式為例，這些請求由連接到服務的負載平衡器傳入或透過其他方式傳入。

物件會從任務定義中選取連接埠、指派 AWS Cloud Map 服務的名稱，以及用戶端應用程式的別名陣列（也稱為端點）和連接埠，以參考此服務。

`portName`

類型：字串

必要：是

portName 必須符合此 Amazon ECS 服務任務定義中來自所有容器的其中一個 portMappings 的 name。

discoveryName

類型：字串

必要：否

discoveryName 是 Amazon ECS 為此 Amazon ECS AWS Cloud Map 服務建立的新服務的名稱。此名稱在 AWS Cloud Map 命名空間中必須不重複。

若沒有指定此欄位，則使用 portName。

clientAliases

類型： 物件陣列

必要：否

此 Service Connect 服務的用戶端別名清單。您可以使用這些別名，指派用戶端應用程式可使用的名稱。您可以在此清單中擁有的用戶端別名數量上限為 1。

每個別名 (「端點」) 都是其他 Amazon ECS 服務 (「用戶端」) 可用來連線至此服務的 DNS 名稱和連接埠號碼。

每個名稱和連接埠組合在命名空間內必須不重複。

這些名稱在用戶端服務的每個任務中設定，而不是在 AWS Cloud Map中設定。解析這些名稱的 DNS 請求不會離開任務，也不會計入每個彈性網路介面每秒對 DNS 請求的配額。

port

類型：整數

必要：是

Service Connect Proxy 的接聽連接埠號碼。此連接埠可在相同命名空間內的所有任務中使用。

若要避免在用戶端 Amazon ECS 服務中變更應用程式，請將此設定為用戶端應用程式預設使用的相同連接埠。

dnsName

類型：字串

必要：否

dnsName 是您在用戶端任務應用程式中用來連線到此服務的名稱。名稱必須是有效的 DNS 標籤。

如果未指定欄位，則預設值為 discoveryName.namespace。如果未指定 discoveryName，則會使用來自任務定義的 portName。

若要避免在用戶端 Amazon ECS 服務中變更應用程式，請將此設定為用戶端應用程式預設使用的相同名稱。例如，一些常見名稱為 database、db，或資料庫的小寫名稱，例如 mysql 或 redis。

ingressPortOverride

類型：整數

必要：否

(選用) Service Connect Proxy 接聽所用的連接埠號碼。

使用此欄位的值以略過此應用程式任務定義中指定之 portMapping 所指定的連接埠號碼上流量的 Proxy，然後在 Amazon VPC 安全群組中使用此值，以允許流量進入此 Amazon ECS 服務的 Proxy。

在 awsvpc 模式中，預設值是在此應用程式任務定義中指定之 portMapping 所指定的容器連接埠號碼。在 bridge 模式中，預設值是 Service Connect Proxy 的動態暫時性連接埠。

logConfiguration

類型：LogConfiguration 物件

必要：否

這會定義 Service Connect Proxy 日誌的發佈位置。在非預期事件期間使用日誌進行偵錯。此組態會在此 Amazon ECS 服務的每個任務中，設定 Service Connect Proxy 容器中的 logConfiguration 參數。未在任務定義中指定 Proxy 容器。

我們建議您使用與此 Amazon ECS 服務任務定義的應用程式容器相同的日誌組態。對於 FireLens，這是應用程式容器的日誌組態。這不是使用 fluent-bit 或 fluentd 容器映像的 FireLens 日誌路由器容器。

`serviceRegistries`

類型： 物件陣列

必要：否

您的服務探索組態詳細資訊。如需詳細資訊，請參閱使用服務探索將 Amazon ECS 服務與 DNS 名稱連線。

`registryArn`

類型：字串

必要：否

服務登錄檔的 Amazon Resource Name (ARN)。目前支援的服務登錄檔為 AWS Cloud Map。如需詳細資訊，請參閱 AWS Cloud Map 開發人員指南中的使用服務。

`port`

類型：整數

必要：否

如果您的服務探索服務指定了 SRV 記錄，則使用該連接埠值。如果已使用 `awsvpc` 網路模式和 SRV 記錄，則此欄位為必要項目。

`containerName`

類型：字串

必要：否

容器名稱值用於您的「服務探索」服務。該值已於任務定義中指定。如果服務任務指定的任務定義是使用 `bridge` 或 `host` 網路模式，您必須透過任務定義指定 `containerName` 與 `containerPort` 組合。如果服務任務指定的任務定義使用 `awsvpc`網路模式，並已使用類型 SRV DNS 記錄，您必須指定 `containerName` 和 `containerPort` 組合或 `port` 值，但不得同時指定兩者。

`containerPort`

類型：整數

必要：否

用於您的「服務探索」服務的連接埠值。該值已於任務定義中指定。如果服務任務指定的任務定義是使用 bridge 或 host 網路模式，您必須透過任務定義指定 containerName 與 containerPort 組合。如果服務任務指定的任務定義使用 awsvpc 網路模式，並已使用類型 SRV DNS 記錄，您必須指定 containerName 和 containerPort 組合或 port 值，但不得同時指定兩者。

用戶端字符

clientToken

類型：字串

必要：否

由您提供的唯一且區分大小寫的識別符，以確保請求的等冪性。長度上限為 32 個 ASCII 字元。

可用區域重新平衡

availabilityZoneRebalancing

類型：字串

必要：否

指出服務是否使用可用區域重新平衡。有效值為 ENABLED和 DISABLED。如需可用區域重新平衡的詳細資訊，請參閱在可用區域之間平衡 Amazon ECS 服務。

磁碟區組態

volumeConfigurations

類型：物件

必要：否

用來為由 服務管理的任務建立磁碟區組態。服務中的每個任務都會建立一個磁碟區。只有任務定義configuredAtLaunch中標記為 的磁碟區才能使用此物件來設定。此物件是將 Amazon EBS 資料磁碟區連接至由 服務管理的任務時的必要物件。如需詳細資訊，請參閱 Amazon EBS 磁碟區。

`name`

類型：字串

必要：是

建立或更新服務時設定的磁碟區名稱。最多允許 255 個字母（大寫和小寫）、數字、底線 (_) 和連字號 (-)。此值必須符合任務定義中指定的磁碟區名稱。

`managedEBSVolume`

類型：物件

必要：否

在建立或更新服務時，連接至由服務管理之任務的 Amazon EBS 磁碟區的磁碟區組態。

`encrypted`

類型：布林值

必要：否

有效值：`true|false`

指定是否加密連接至由 服務管理之任務的 Amazon EBS 磁碟區。如果您已為您的帳戶預設開啟 Amazon EBS 加密，此設定將會覆寫，且磁碟區將會加密。如需預設 EBS 加密的詳細資訊，請參閱《Amazon EBS 使用者指南》中的預設啟用 Amazon EBS 加密。

`kmsKeyId`

類型：字串

必要：否

用於 Amazon EBS 加密的 AWS Key Management Service (AWS KMS) 金鑰識別碼。如果未指定此參數，則會使用適用於 Amazon EBS AWS KMS key 的 。如果指定 `KmsKeyId`，加密狀態必須是 `true`。

您可以使用下列任一項來指定 KMS 金鑰：

- 金鑰 ID – 例如，`1234abcd-12ab-34cd-56ef-1234567890ab`。
- 金鑰別名 – 例如 `alias/ExampleAlias`。

- 金鑰 ARN – 例如，`arn:aws:kms:us-east-1:012345678910:key/1234abcd-12ab-34cd-56ef-1234567890ab`。
- 別名 ARN – 例如 `arn:aws:kms:us-east-1:012345678910:alias/ExampleAlias`。

> **Important**
>
> AWS 會以非同步方式驗證 KMS 金鑰。因此，如果您指定無效的 ID、別名或 ARN，動作可能會看起來成功，但最終會失敗。如需詳細資訊，請參閱疑難排解 Amazon EBS 磁碟區連接問題。

`volumeType`

類型：字串

必要：否

有效值：`gp2|gp3|io1|io2|sc1st1|standard`

EBS 磁碟區類型。如需磁碟區類型的詳細資訊，請參閱《Amazon EBS 使用者指南》中的 Amazon EBS 磁碟區類型。 預設磁碟區類型為 gp3。

> **Note**
>
> `standard` 為連接至 Fargate 任務而設定的 Amazon EBS 磁碟區不支援磁碟區類型。

`sizeInGiB`

類型：整數

必要：否

有效範圍：介於 1 到 16，384 之間的整數

EBS 磁碟區的大小，以 GB (GiB) 為單位。如果您沒有提供快照 ID 來設定連接磁碟區，則必須提供大小值。如果您使用快照設定連接磁碟區，預設值為快照大小。然後，您可以指定大於或等於快照大小的大小。

對於 gp2和 gp3磁碟區類型，有效範圍為 1-16，384。

對於 io1和 io2磁碟區類型，有效範圍為 4-16，384。

對於 st1和 sc1磁碟區類型，有效範圍為 125-16，384。

對於standard磁碟區類型，有效範圍為 1-1，024。

`snapshotId`

類型：字串

必要：否

現有 EBS 磁碟區的快照 ID，用於建立新的磁碟區，並連接到 ECS 任務。

`iops`

類型：整數

必要：否

每秒 I/O 操作的次數 (IOPS)。對 gp3、io1 和 io2 磁碟區而言，這代表針對磁碟區佈建的 IOPS 數量。對於gp2磁碟區，此值代表磁碟區的基準效能，以及磁碟區累積 I/O 點數爆量的速率。這是 io1 和 io2 磁碟區的必要參數。gp2、 st1sc1或 standard磁碟區不支援此參數。

對於磁碟gp3區，有效值範圍為 3，000 到 16，000。

對於磁碟io1區，有效值範圍為 100 到 64，000。

對於磁碟io2區，有效值範圍為 100 到 64，000。

`throughput`

類型：整數

必要：否

連接至由 服務管理之任務之磁碟區的佈建輸送量。

> **Important**
>
> 此參數僅支援 gp3磁碟區。

`roleArn`

類型：字串

必要：是

基礎設施 (IAM) 角色的 Amazon Resource ARN AWS Identity and Access Management (ARN)，提供 Amazon ECS 許可來管理任務的 Amazon EBS 資源。如需詳細資訊，請參閱Amazon ECS 基礎設施 IAM 角色。

`tagSpecifications`

類型：物件

必要：否

要套用至 serivce 受管 Amazon EBS 磁碟區的標籤規格。

`resourceType`

類型：字串

必要：是

有效值：`volume`

建立時要標記的資源類型。

`tags`

類型： 物件陣列

必要：否

您套用至磁碟區的中繼資料，可協助您分類和組織這些磁碟區。每個標籤都包含一個索引鍵和一個選用值，您定義兩者。 `AmazonECSCreated`和 `AmazonECSManaged`都是 Amazon ECS 代表您新增的預留標籤，因此您最多可以指定自己的 48 個標籤。刪除磁碟區時，也會刪除標籤。如需詳細資訊，請參閱標記 Amazon ECS 資源。

`key`

類型：字串

長度限制：長度下限為 1。長度上限為 128。

必要：否

組成標籤的鍵值對的一部分。索引鍵是一般標籤，作用就像更特定標籤值的類別。

value

類型：字串

長度限制：長度下限為 0。長度上限為 256。

必要：否

組成標籤之鍵值對的選用部分。值就像標籤類別 (索引鍵) 內的描述項。

propagateTags

類型：字串

有效值：TASK_DEFINITION | SERVICE | NONE

必要：否

指定要將標籤從任務定義或服務複製到磁碟區。如果NONE已指定 或沒有指定值，則不會複製標籤。

fileSystemType

類型：字串

必要：否

有效值：xfs|ext3|ext4|NTFS

磁碟區上的檔案系統類型。磁碟區的檔案系統類型會決定如何在磁碟區中存放和擷取資料。對於從快照建立的磁碟區，您必須指定建立快照時磁碟區所使用的相同檔案系統類型。如果檔案系統類型不相符，任務將無法啟動。

Linux 的有效值為 xfs、ext3, and ext4。連接到 Linux 任務的磁碟區預設為 XFS。

Windows 的有效值為 NTFS。連接到 Windows 任務的磁碟區預設為 NTFS。

服務定義範本

以下顯示 Amazon ECS 服務定義的 JSON 表示法。

Amazon EC2 啟動類型

```
{
    "cluster": "",
    "serviceName": "",
    "taskDefinition": "",
    "loadBalancers": [
        {
            "targetGroupArn": "",
            "loadBalancerName": "",
            "containerName": "",
            "containerPort": 0
        }
    ],
    "serviceRegistries": [
        {
            "registryArn": "",
            "port": 0,
            "containerName": "",
            "containerPort": 0
        }
    ],
    "desiredCount": 0,
    "clientToken": "",
    "launchType": "EC2",
    "capacityProviderStrategy": [
        {
            "capacityProvider": "",
            "weight": 0,
            "base": 0
        }
    ],
    "platformVersion": "",
    "role": "",
    "deploymentConfiguration": {
        "deploymentCircuitBreaker": {
            "enable": true,
            "rollback": true
        },
        "maximumPercent": 0,
        "minimumHealthyPercent": 0,
        "alarms": {
            "alarmNames": [
                ""
```

```
            ],
            "enable": true,
            "rollback": true
        }
    },
    "placementConstraints": [
        {
            "type": "distinctInstance",
            "expression": ""
        }
    ],
    "placementStrategy": [
        {
            "type": "binpack",
            "field": ""
        }
    ],
    "networkConfiguration": {
        "awsvpcConfiguration": {
            "subnets": [
                ""
            ],
            "securityGroups": [
                ""
            ],
            "assignPublicIp": "DISABLED"
        }
    },
    "healthCheckGracePeriodSeconds": 0,
    "schedulingStrategy": "REPLICA",
    "deploymentController": {
        "type": "EXTERNAL"
    },
    "tags": [
        {
            "key": "",
            "value": ""
        }
    ],
    "enableECSManagedTags": true,
    "propagateTags": "TASK_DEFINITION",
    "enableExecuteCommand": true,
    "availabilityZoneRebalancing": "ENABLED",
    "serviceConnectConfiguration": {
```

```
        "enabled": true,
        "namespace": "",
        "services": [
            {
                "portName": "",
                "discoveryName": "",
                "clientAliases": [
                    {
                        "port": 0,
                        "dnsName": ""
                    }
                ],
                "ingressPortOverride": 0
            }
        ],
        "logConfiguration": {
            "logDriver": "journald",
            "options": {
                "KeyName": ""
            },
            "secretOptions": [
                {
                    "name": "",
                    "valueFrom": ""
                }
            ]
        }
    },
    "volumeConfigurations": [
        {
            "name": "",
            "managedEBSVolume": {
                "encrypted": true,
                "kmsKeyId": "",
                "volumeType": "",
                "sizeInGiB": 0,
                "snapshotId": "",
                "iops": 0,
                "throughput": 0,
                "tagSpecifications": [
                    {
                        "resourceType": "volume",
                        "tags": [
                            {
```

```
                                "key": "",
                                "value": ""
                            }
                        ],
                        "propagateTags": "NONE"
                    }
                ],
                "roleArn": "",
                "filesystemType": ""
            }
        }
    ]
}
```

Fargate 啟動類型

```
{
    "cluster": "",
    "serviceName": "",
    "taskDefinition": "",
    "loadBalancers": [
        {
            "targetGroupArn": "",
            "loadBalancerName": "",
            "containerName": "",
            "containerPort": 0
        }
    ],
    "serviceRegistries": [
        {
            "registryArn": "",
            "port": 0,
            "containerName": "",
            "containerPort": 0
        }
    ],
    "desiredCount": 0,
    "clientToken": "",
    "launchType": "FARGATE",
    "capacityProviderStrategy": [
        {
            "capacityProvider": "",
            "weight": 0,
```

```
            "base": 0
        }
    ],
    "platformVersion": "",
    "platformFamily": "",
    "role": "",
    "deploymentConfiguration": {
        "deploymentCircuitBreaker": {
            "enable": true,
            "rollback": true
        },
        "maximumPercent": 0,
        "minimumHealthyPercent": 0,
        "alarms": {
            "alarmNames": [
                ""
            ],
            "enable": true,
            "rollback": true
        }
    },
    "placementStrategy": [
        {
            "type": "binpack",
            "field": ""
        }
    ],
    "networkConfiguration": {
        "awsvpcConfiguration": {
            "subnets": [
                ""
            ],
            "securityGroups": [
                ""
            ],
            "assignPublicIp": "DISABLED"
        }
    },
    "healthCheckGracePeriodSeconds": 0,
    "schedulingStrategy": "REPLICA",
    "deploymentController": {
        "type": "EXTERNAL"
    },
    "tags": [
```

```
        {
            "key": "",
            "value": ""
        }
    ],
    "enableECSManagedTags": true,
    "propagateTags": "TASK_DEFINITION",
    "enableExecuteCommand": true,
    "availabilityZoneRebalancing": "ENABLED",
    "serviceConnectConfiguration": {
        "enabled": true,
        "namespace": "",
        "services": [
            {
                "portName": "",
                "discoveryName": "",
                "clientAliases": [
                    {
                        "port": 0,
                        "dnsName": ""
                    }
                ],
                "ingressPortOverride": 0
            }
        ],
        "logConfiguration": {
            "logDriver": "journald",
            "options": {
                "KeyName": ""
            },
            "secretOptions": [
                {
                    "name": "",
                    "valueFrom": ""
                }
            ]
        }
    },
    "volumeConfigurations": [
        {
            "name": "",
            "managedEBSVolume": {
                "encrypted": true,
                "kmsKeyId": "",
```

```
                "volumeType": "",
                "sizeInGiB": 0,
                "snapshotId": "",
                "iops": 0,
                "throughput": 0,
                "tagSpecifications": [
                    {
                        "resourceType": "volume",
                        "tags": [
                            {
                                "key": "",
                                "value": ""
                            }
                        ],
                        "propagateTags": "NONE"
                    }
                ],
                "roleArn": "",
                "filesystemType": ""
            }
        }
    ]
}
```

您可以使用下列 AWS CLI 命令建立此服務定義範本。

```
aws ecs create-service --generate-cli-skeleton
```

標記 Amazon ECS 資源

為了協助您管理 Amazon ECS 資源，您可以選擇使用標籤 將自己的中繼資料指派給每個資源。每個標籤皆包含索引鍵與選用值。

您可以使用標籤，以不同方式分類 Amazon ECS 資源，例如依用途、擁有者或環境。這在您擁有許多相同類型的資源時很有用。您可以根據您指派給資源的標籤快速識別特定資源。例如，您可以為帳戶的Amazon ECS容器執行個體定義一組標籤。這可以協助您追蹤每個執行個體的擁有者和堆疊層級。

您可以將標籤用於成本和用量報告。您可以使用這些報告來分析 Amazon ECS 資源的成本和用量。如需詳細資訊，請參閱the section called “用量報告”。

> **Warning**
>
> 有許多APIs傳回標籤索引鍵及其值。拒絕存取 `DescribeTags`不會自動拒絕存取其他 傳回的標籤APIs。根據最佳實務，建議您不要在標籤中包含敏感資料。

我們建議您為每種資源類型建立符合您需求的標籤金鑰。您可以使用一致的標籤索引鍵組，讓您更輕鬆的管理您的資源。您可以根據您新增的標籤搜尋和篩選資源。

標籤對 Amazon 沒有任何語義意義，ECS並嚴格解譯為一串字元。您可以編輯標籤金鑰和值，並且可以隨時從資源移除標籤。您可以將標籤的值設為空白字串，但您無法將標籤的值設為 Null。若您將與現有標籤具有相同鍵的標籤新增到該資源，則新值會覆寫早前的值。刪除資源時，也會刪除資源的任何標籤。

如果您使用 AWS Identity and Access Management （ IAM ），您可以控制 AWS 帳戶中哪些使用者具有管理標籤的許可。

如何標記資源

標記 Amazon ECS任務、服務、任務定義和叢集的方式有多種：

- 使用者使用 AWS Management Console、Amazon ECS API、 AWS CLI或 手動標記資源 AWS SDK。
- 使用者建立服務或執行獨立任務，並選取 Amazon ECS受管標籤選項。

  Amazon ECS會自動標記所有新啟動的任務。如需詳細資訊，請參閱the section called “Amazon ECS受管標籤”。

- 使用者可使用主控台建立資源。主控台會自動標記資源。

  這些標籤會傳回 AWS CLI、 和 AWS SDK 回應，並顯示在主控台中。您無法修改或刪除這些標籤。

  如需新增標籤的相關資訊，請參閱 Amazon ECS 資源標籤支援資料表中的主控台欄自動新增的標籤。

如果您在建立資源時指定標籤，且標籤無法套用，Amazon 會ECS復原建立程序。這可確保資源不是具有標籤建立，就是不會建立，因此無論何時都不會有不具有標籤的資源。藉由在建立時為資源建立標籤，您可以消除在資源建立後執行自訂標籤指令碼的必要。

下表說明支援標記的 Amazon ECS 資源。

資源	支援標籤	支援標籤傳播	主控台自動新增的標籤
Amazon ECS任務	是	是，從任務定義。	索引鍵：`aws:ecs:clusterName` Value (值)：`cluster-name`
Amazon ECS服務	是	是，從任務定義或服務到服務中的任務。	索引鍵：`ecs:service:stackId` 值`arn:aws:cloudformation: arn`
Amazon ECS任務集	是	否	N/A
Amazon ECS任務定義	是	否	索引鍵：`ecs:taskDefinition:createdFrom` Value (值)：`ecs-console-v2`

資源	支援標籤	支援標籤傳播	主控台自動新增的標籤
Amazon ECS叢集	是	否	索引鍵：`aws:cloudformation:logical-id` Value (值)：`ECSCluster` 索引鍵：`aws:cloudformation:stack-id` Value (值)：`arn:aws:cloudformation:` *`arn`* 索引鍵：`aws:cloudformation:stack-name` Value (值)：`ECS-Console-V2-Cluster-` *`EXAMPLE`*
Amazon ECS容器執行個體	是	是，來自 Amazon EC2執行個體。如需詳細資訊，請參閱將標籤新增至 Amazon ECS容器執行個體。	N/A
Amazon ECS外部執行個體	是	否	N/A

資源	支援標籤	支援標籤傳播	主控台自動新增的標籤
Amazon ECS容量提供者	是。 您無法標記預先定義的 FARGATE 和 FARGATE_SPOT 容量提供者。	否	N/A

在建立期間標記資源

下列資源支援使用 Amazon ECS API AWS CLI、 或 AWS 建立時標記SDK：

- Amazon ECS任務
- Amazon ECS服務
- Amazon ECS任務定義
- Amazon ECS任務集
- Amazon ECS叢集
- Amazon ECS容器執行個體
- Amazon ECS容量提供者

Amazon ECS可以選擇使用標記授權來建立資源。 AWS 帳戶 設定 以標記授權時，使用者必須具有建立資源之動作的許可，例如 `ecsCreateCluster`。如果您在資源建立動作中指定標籤，AWS 會執行其他授權，以驗證使用者或角色是否具有建立標籤的許可。因此，您必須授予使用 `ecs:TagResource` 動作的明確許可。如需詳細資訊，請參閱the section called “在建立期間標記資源”。如需有關如何設定 選項的資訊，請參閱 the section called “標記授權”。

限制

以下限制適用於標籤：

- 資源最多可與 50 個標籤建立關聯。
- 單一資源的標籤索引鍵不能重複。每個標籤索引鍵都必須是唯一的，而且只能有一個值。

- 索引鍵長度上限為 128 個字元，長度為 UTF-8。
- 值長度上限為 256 個字元，長度為 UTF-8。
- 如果多個 AWS 服務 和資源使用您的標記結構描述，請限制您使用的字元類型。某些服務可能對允許的字元設有限制。通常允許的字元為：字母、數字和空格，以及下列字元：**+ - = . _ : / @**。
- 標籤鍵與值皆區分大小寫。
- 您無法使用 `aws:`、`AWS:` 或任何大小寫組合作為索引鍵或值的字首。這些僅供保留 AWS 使用。您不可編輯或刪除具此字首的標籤金鑰或值。具有此字首的標籤不會計入您的 tags-per-resource限制。

Amazon ECS受管標籤

當您使用 Amazon ECS受管標籤時，Amazon ECS會自動將所有新啟動的任務和連接至任務的任何 Amazon EBS磁碟區，標記叢集資訊，以及使用者新增的任務定義標籤或服務標籤。以下說明新增的標籤：

- 獨立任務 – 索引鍵為 `aws:ecs:clusterName` 且值設定為叢集名稱的標籤。使用者新增的所有任務定義標籤。連接至獨立任務的 Amazon EBS磁碟區將收到金鑰為 的標籤，`aws:ecs:clusterName`並將值設定為叢集名稱。如需 Amazon EBS磁碟區標記的詳細資訊，請參閱標記 Amazon EBS磁碟區 。
- 屬於服務一部分的任務 – 索引鍵為 `aws:ecs:clusterName` 且值設定為叢集名稱的標籤。索引鍵為 `aws:ecs:serviceName` 且值設定為服務名稱的標籤。來自下列其中一項資源的標籤：
  - 任務定義 – 使用者新增的所有任務定義標籤。
  - 服務 – 使用者新增的所有服務標籤。

  連接至 任務的 Amazon EBS磁碟區會接收一個標籤，其金鑰為 `aws:ecs:clusterName`，並將值設定為 叢集名稱，以及將金鑰設定為 `aws:ecs:serviceName`，並將值設定為 服務名稱。如需 Amazon EBS磁碟區標記的詳細資訊，請參閱標記 Amazon EBS磁碟區 。

對此功能而言，下列選項為必要：

- 您必須選擇加入新的 Amazon Resource Name （ ARN ） 和資源識別碼 （ ID ） 格式。如需詳細資訊，請參閱Amazon Resource Names (ARNs) 和 IDs。
- 當您使用 APIs來建立服務或執行任務時，您必須`enableECSManagedTags`將 設定為 `true` `run-task`和 `create-service`。如需詳細資訊，請參閱 AWS Command Line Interface API 參考 中的建立服務和執行任務。

- Amazon ECS使用受管標籤來判斷啟用某些功能的時間，例如叢集 Auto Scaling 。建議您不要手動修改標籤，以便 Amazon ECS可以有效地管理功能。

使用標籤計費

AWS 提供名為 Cost Explorer 的報告工具，可用來分析 Amazon ECS 資源的成本和用量。

您可以使用 Cost Explorer 來檢視用量和成本的圖表。您可以檢視過去 13 個月以來的資料，並預測未來三個月的可能花費。您可以使用 Cost Explorer 來查看在一段時間內的 AWS 資源支出模式。例如，您可以用它來找出需進一步調查的領域，以及查看您可用來了解成本的趨勢。您也可以指定資料的時間範圍，以及根據天或月檢視時間資料。

您可以使用 Amazon ECS受管標籤或使用者新增標籤作為成本和用量報告。如需詳細資訊，請參閱Amazon ECS用量報告。

若想要查看合併資源的成本，您可根據具有相同標籤金鑰值的資源來整理您的帳單資訊。例如，您可以使用特定應用程式名稱來標記數個資源，然後整理帳單資訊以查看該應用程式跨數項服務的總成本。如需有關使用標籤設定成本分配報告的詳細資訊，請參閱《AWS Billing 使用者指南》中的每月成本分配報告。

此外，您可以開啟分割成本分配資料，以取得成本CPU和用量報告中的任務層級和記憶體用量資料。如需詳細資訊，請參閱任務層級成本和用量報告。

> **Note**
>
> 若您開啟報告，目前月份的資料會在 24 小時之後提供檢視。

將標籤新增至 Amazon ECS 資源

您可以標記新的或現有的任務、服務、任務定義或叢集。如需有關標記容器執行個體的資訊，請參閱將標籤新增至 Amazon ECS容器執行個體。

> **Warning**
>
> 請勿在標籤中新增個人識別資訊（PII）或其他機密或敏感資訊。許多 AWS 服務都可以存取標籤，包括計費。標籤不適用於私人或敏感資料。

建立資源時，您可以使用以下資源來指定標籤。

任務	主控台	AWS CLI	API 動作
執行一或多個任務。	以 Amazon ECS 任務執行應用程式	run-task	RunTask
建立服務。	使用 主控台建立 Amazon ECS 服務	create-service	CreateService
建立任務集。	使用第三方控制器部署 Amazon ECS 服務	create-task-set	CreateTaskSet
註冊任務定義。	the section called “使用主控台建立任務定義”	register-task-definition	RegisterTaskDefinition
建立叢集。	為 Fargate 啟動類型建立 Amazon ECS 叢集	create-cluster	CreateCluster
執行一或多個容器執行個體。	啟動 Amazon ECS Linux 容器執行個體	run-instances	RunInstances

將標籤新增至現有資源 （ Amazon ECS主控台 ）

您可以直接從資源頁面新增或刪除與叢集、服務、任務和任務定義相關聯的標籤。

修改個別資源的標籤

1. 在 https://console.aws.amazon.com/ecs/v2 開啟主控台。
2. 從導覽列中，選取要 AWS 區域 使用的 。
3. 在導覽窗格中，選取資源類型 (例如，Clusters (叢集))。

4. 從資源清單選取資源，選擇 Tags (標籤) 索引標籤，然後選擇 Manage tags (管理標籤)。
5. 設定標籤。

   [新增標籤] 選擇新增標籤，然後執行下列操作：

   - 在索引鍵中，輸入索引鍵名稱。
   - 在值中，進入索引鍵值。
6. 選擇 Save (儲存)。

將標籤新增至現有資源 （ AWS CLI ）

您可以使用 或 AWS CLI 來新增或覆寫一或多個標籤API。

- AWS CLI - tag-resource
- API 動作 - TagResource

將標籤新增至 Amazon ECS容器執行個體

您可以使用以下方法其中之一，將標籤與您的容器執行個體建立關聯：

- 方法 1 – 使用 Amazon EC2 API、 CLI或 主控台建立容器執行個體時，請使用容器代理程式組態參數 將使用者資料傳遞至執行個體來指定標籤ECS_CONTAINER_INSTANCE_TAGS。這只會建立與 ECS Amazon 中容器執行個體相關聯的標籤，無法使用 Amazon EC2 列出這些標籤API。如需詳細資訊，請參閱引導 Amazon ECS Linux 容器執行個體以傳遞資料。

  > **Important**
  >
  > 如果您使用 Amazon EC2 Auto Scaling 群組啟動容器執行個體，則應該使用 ECS_CONTAINER_INSTANCE_TAGS 代理程式組態參數來新增標籤。這是因為標籤新增至使用 Auto Scaling 群組啟動的 Amazon EC2執行個體的方式。

  以下的使用者資料指令碼範例將標籤與您的容器執行個體建立關聯：

```
#!/bin/bash
cat <<'EOF' >> /etc/ecs/ecs.config
ECS_CLUSTER=MyCluster
```

```
ECS_CONTAINER_INSTANCE_TAGS={"tag_key": "tag_value"}
EOF
```

- 方法 2 – 當您使用 Amazon EC2 API、 CLI或 主控台建立容器執行個體時，請先使用 `TagSpecification.N` 參數指定標籤。然後，使用容器代理程式組態參數 `ECS_CONTAINER_INSTANCE_PROPAGATE_TAGS_FROM` 將使用者資料傳遞至執行個體。這樣做會將它們從 Amazon 傳播EC2到 Amazon ECS。

  以下是使用者資料指令碼的範例，該指令碼會傳播與 Amazon EC2執行個體相關聯的標籤，並將執行個體註冊到名為 的叢集`MyCluster`。

  ```
  #!/bin/bash
  cat <<'EOF' >> /etc/ecs/ecs.config
  ECS_CLUSTER=MyCluster
  ECS_CONTAINER_INSTANCE_PROPAGATE_TAGS_FROM=ec2_instance
  EOF
  ```

  若要提供允許容器執行個體標籤從 Amazon 傳播EC2至 Amazon 的存取權ECS，請手動將下列許可作為內嵌政策新增至 Amazon ECS容器執行個體IAM角色。如需詳細資訊，請參閱新增和移除IAM政策。

  - `ec2:DescribeTags`

  以下為用來新增這些許可的政策範例。

  ```
  {
      "Version": "2012-10-17",
      "Statement": [
          {
              "Effect": "Allow",
              "Action": [
               "ec2:DescribeTags"
              ],
              "Resource": "*"
          }
      ]
  }
  ```

外部容器執行個體

您可以使用以下其中一種方法，將標籤與您的外部容器執行個體建立關聯。

- 方法 1 – 在執行安裝指令碼向叢集註冊外部執行個體之前，請在 建立或編輯 Amazon ECS容器代理程式組態檔案，`/etc/ecs/ecs.config`然後新增`ECS_CONTAINER_INSTANCE_TAGS`容器代理程式組態參數。這將會建立與外部執行個體相關聯的標籤。

  以下為範例語法。

  ```
  ECS_CONTAINER_INSTANCE_TAGS={"tag_key": "tag_value"}
  ```

- 方法 2 – 將外部執行個體註冊至叢集後，您可以使用 AWS Management Console 來新增標籤。如需詳細資訊，請參閱將標籤新增至現有資源（Amazon ECS主控台）。

Amazon ECS用量報告

AWS 提供名為 Cost Explorer 的報告工具，可用來分析 Amazon ECS 資源的成本和用量。

您可以使用 Cost Explorer 來檢視用量和成本的圖表。您可以檢視過去 13 個月以來的資料，並預測未來三個月的可能花費。您可以使用 Cost Explorer 來查看在一段時間內的 AWS 資源支出模式。例如，您可以用它來找出需進一步調查的領域，以及查看您可用來了解成本的趨勢。您也可以指定資料的時間範圍，以及根據天或月檢視時間資料。

成本和用量報告中的計量資料會顯示所有 Amazon ECS任務的用量。計量資料包含與執行`GB-Hours`中每個任務相同的CPU用量`vCPU-Hours`和記憶體用量。資料的呈現方式取決於任務的啟動類型。

針對使用 Fargate 啟動類型的任務，`lineItem/Operation` 欄會顯示 `FargateTask`，而且您會看到與每項任務相關聯的費用。

對於使用EC2啟動類型的任務，資料`lineItem/Operation`欄會顯示 ，`ECSTask-EC2`且任務沒有與其相關聯的直接成本。報告中顯示的計量資料，如記憶體用量，代表任務在指定計費期間預留的總資源。您可以使用此資料來判斷 Amazon EC2執行個體基礎叢集的成本。Amazon EC2執行個體的成本和用量資料會分別列在 Amazon EC2服務下。

您也可以使用 Amazon ECS受管標籤來識別每個任務所屬的服務或叢集。如需詳細資訊，請參閱使用標籤計費。

> **Important**
>
> 只有在 2018 年 11 月 16 日或之後啟動的任務，才有計量資料可供檢視。在此日期之前啟動的任務不會顯示計量資料。

以下是 Cost Explorer 中可以用來排序成本分配資料的一些欄位範例。

- 叢集名稱
- 服務名稱
- 資源標籤
- 啟動類型
- AWS 區域
- 用量類型

如需建立 AWS 成本和用量報告的詳細資訊，請參閱 AWS Billing 使用者指南 中的AWS 成本和用量報告。

任務層級成本和用量報告

AWS Cost Management 可以在 中 AWS Cost and Usage Report 為 Amazon 上的每個任務提供 CPU 和 記憶體用量資料ECS，包括 Fargate 上的任務和 上的任務EC2。此資料稱為拆分成本分配資料。您可以使用此資料來分析應用程式的成本和用量。此外，您可以拆分成本並將成本分配給具有成本分配標籤和成本類別的個別業務單位和團隊。如需分割成本分配資料 的詳細資訊，請參閱 AWS Cost and Usage Report 使用者指南中的了解分割成本分配資料。

您可以選擇加入 AWS Cost Management Console帳戶的任務層級拆分成本分配資料。如果您有一個管理 (付款人) 帳戶，您可以從付款人帳戶中選擇加入此組態，並將其套用至每個連結的帳戶。

設定分割成本分配資料 後，報告中的splitLineItem標頭下會有額外的資料欄。如需詳細資訊，請參閱 AWS Cost and Usage Report 使用者指南中的分割明細項目詳細資訊

對於 上的任務EC2，此資料會根據資源用量或保留以及EC2執行個體上的剩餘資源來分割執行個體的成本。

以下是先決條件：

- 將 `ECS_DISABLE_METRICS` Amazon ECS代理程式組態參數設定為 `false`。

  當此設定為 時`false`，Amazon ECS代理程式會將指標傳送至 Amazon CloudWatch。在 Linux 上，此設定`false`預設為 ，指標會傳送至 CloudWatch。在 Windows 上，此設定`true`預設為 ，因此您必須將設定變更為 `false`，才能將指標 CloudWatch傳送至 AWS Cost Management 以供 使用。如需ECS代理程式組態的詳細資訊，請參閱 Amazon ECS 容器代理程式組態。

- 可靠指標的最低 Docker 版本為 Docker v20.10.13 及更新版本，其包含在 Amazon ECS最佳化AMI 的 20220607 及更新版本中。

若要使用拆分成本分配資料，您必須建立報表，然後選取 Split cost allocation data (拆分成本分配資料)。如需詳細資訊，請參閱 AWS Cost and Usage Report 使用者指南中的建立成本和用量報告。

AWS Cost Management 會使用任務CPU和記憶體用量來計算分割成本分配資料。 AWS Cost Management 如果無法使用， 可以使用任務CPU和記憶體保留，而不是用量。如果您看到 CUR 使用保留，請檢查您的容器執行個體是否符合先決條件，且任務資源用量指標會顯示在 中 CloudWatch。

監控 Amazon ECS

監控是維護 Amazon ECS 和您的 AWS 解決方案可靠性、可用性和效能的重要部分。您應該從 AWS 解決方案的所有部分收集監控資料，以便在發生多點失敗時更輕鬆地偵錯。開始監控 Amazon ECS 之前，請建立監控計畫，其中包含下列問題的答案：

- 監控目標是什麼？
- 要監控哪些資源？
- 監控這些資源的頻率為何？
- 要使用哪些監控工具？
- 誰將執行監控任務？
- 發生問題時應該通知誰？

提供的指標依據叢集中的任務與服務的啟動類型而定。如果您的服務使用的是 Fargate 啟動類型，將提供 CPU 與記憶體使用率指標，以協助您監控服務。對於 Amazon EC2 啟動類型，您擁有且需要監控構成基礎設施的 EC2 執行個體。叢集、服務和任務提供其他 CPU 和記憶體保留和使用率指標。

下一步是在各個時間點和不同的負載條件下測量效能，以在您的環境中確立 Amazon ECS 正常效能的基準。當您監控 Amazon ECS 時，請存放歷史記錄監控資料，如此才能與目前的效能資料做比較、辨識正常效能模式和效能異常狀況、規劃問題處理方式。

若要建立基準，您至少必須監控下列項目：

- 您的 Amazon ECS 叢集的 CPU 和記憶體預留及使用率指標
- 您的 Amazon ECS 服務的 CPU 及記憶體使用率指標

  如需詳細資訊，請參閱檢視 Amazon ECS 指標。

監控 Amazon ECS 的最佳實務

使用下列最佳實務來監控 Amazon ECS。

- 將監控視為優先事項，在小問題變得大問題之前解決
- 建立監控計畫，其中包含下列問題的答案
  - 監控目標是什麼？
  - 要監控哪些資源？

  - 監控這些資源的頻率為何？
  - 要使用哪些監控工具？
  - 誰將執行監控任務？
  - 發生問題時應該通知誰？
- 盡可能自動化監控。
- 檢查 Amazon ECS 日誌檔案。如需詳細資訊，請參閱檢視 Amazon ECS 容器代理程式日誌。
- 使用執行期監控協助保護您的帳戶、容器、工作負載和 AWS 環境中的資料。如需詳細資訊，請參閱使用執行期監控識別未經授權的行為。

Amazon ECS 的監控工具

AWS 提供各種可用來監控 Amazon ECS 的工具。您可以設定其中一些工具來進行監控，但有些工具需要手動介入。建議您盡可能自動化監控任務。

自動化監控工具

您可以使用下列自動化監控工具來監看 Amazon ECS，並在發生錯誤時進行回報：

- Amazon CloudWatch 警示：監看指定時段內的單一指標，並根據與多個時段內給定之閾值相對的指標值來執行一或多個動作。此動作是傳送到 Amazon Simple Notification Service (Amazon SNS) 主題或 Amazon EC2 Auto Scaling 政策的通知。CloudWatch 警示不會只因處於特定狀態就叫用動作，狀態必須已變更並已維持一段指定的時間。如需詳細資訊，請參閱使用 CloudWatch 監控 Amazon ECS。

  如果服務的任務是使用 Fargate 啟動類型，您可以使用 CloudWatch 警示，以根據 CloudWatch 指標 (例如 CPU 和記憶體使用率) 來擴展和縮減服務中的任務。如需詳細資訊，請參閱自動擴展 Amazon ECS 服務。

  如果叢集的任務或服務是使用 EC2 啟動類型，您可以使用 CloudWatch 警示，以根據 CloudWatch 指標 (例如叢集記憶體保留) 來縮減和擴展容器執行個體。

  對於利用 Amazon ECS 最佳化 Amazon Linux AMI 啟動的容器執行個體，您可以使用 CloudWatch Logs 在一個便利位置檢視容器執行個體的不同日誌。您必須在容器執行個體上安裝 CloudWatch 代理程式。如需詳細資訊，請參閱 Amazon CloudWatch 使用者指南中的使用命令列下載並設定 CloudWatch 代理程式。您還必須將 `ECS-CloudWatchLogs` 政策新增至 `ecsInstanceRole` 角色。如需詳細資訊，請參閱監控容器執行個體許可。

- Amazon CloudWatch Logs：透過在任務定義中指定 `awslogs` 日誌驅動程式，藉以監控、存放及存取您的 Amazon ECS 任務中容器的日誌檔案。如需詳細資訊，請參閱將 Amazon ECS 日誌傳送至 CloudWatch 。

  您也可以從 Amazon ECS 容器執行個體監控、存放及存取作業系統和 Amazon ECS 容器代理程式日誌檔案。這種存取日誌的方法可用於使用 EC2 啟動類型的容器。
- Amazon CloudWatch Events：匹配事件並將它們路由至一或多個目標函式或串流以進行變更、擷取狀態資訊，以及採取修正動作。如需詳細資訊，請參閱本指南使用 EventBridge 自動化對 Amazon ECS 錯誤的回應中的 和 EventBridge 是 Amazon EventBridge 使用者指南中的 Amazon CloudWatch Events 的演變。 EventBridge
- Container Insights – 從容器化應用程式和微服務收集、彙總和摘要指標和日誌。Container Insights 會使用內嵌指標格式將資料收集為效能日誌事件。這些效能日誌事件是使用結構化 JSON 結構描述的項目，允許大規模擷取和存放高基數資料。從這些資料中，CloudWatch 會在叢集、任務和服務層級建立彙總指標，做為 CloudWatch 指標。您可在 CloudWatch 自動儀表板取得 Container Insights 收集的指標，也可在 CloudWatch 主控台的指標區段進行檢視。
- AWS CloudTrail 日誌監控 – 在帳戶之間共用日誌檔案、透過將日誌檔案傳送至 CloudWatch Logs 即時監控 CloudTrail 日誌檔案、在 Java 中寫入日誌處理應用程式，以及驗證您的日誌檔案在 CloudTrail 交付後並未變更。如需詳細資訊，請參閱本指南中的 使用 記錄 Amazon ECS API 呼叫 AWS CloudTrail，以及《AWS CloudTrail 使用者指南》中的使用 CloudTrail 記錄檔案。
- 執行期監控 – 偵測環境中 AWS 叢集和容器的威脅。執行期監控使用 GuardDuty 安全代理程式，可增加個別 Amazon ECS 工作負載的執行期可見性，例如檔案存取、程序執行和網路連線。

手動監控工具

監控 Amazon ECS 的另一個重要部分是手動監控 CloudWatch 警示未涵蓋的項目。CloudWatch Trusted Advisor和其他 AWS 主控台儀表板提供 AWS 環境狀態的at-a-glance。我們建議您也檢查您容器執行個體上的日誌檔，以及您任務中的容器。

- Amazon ECS 主控台：
  - EC2 啟動類型的叢集指標
  - 服務指標
  - 服務運作狀態
  - 服務部署事件
- CloudWatch 首頁：
  - 目前警示與狀態

- 警示與資源的圖表
- 服務運作狀態

此外，您可以使用 CloudWatch 執行下列動作：

- 建立自訂儀表板來監控您關心的服務。
- 用於疑難排解問題以及探索驅勢的圖形指標資料。
- 搜尋和瀏覽您的所有 AWS 資源指標。
- 建立與編輯要通知發生問題的警示。

- 容器運作狀態檢查 - 這些命令會在容器本機上執行，並驗證應用程式運作狀態和可用性。您可以在任務定義中為每個容器設定這些值。
- AWS Trusted Advisor 可協助您監控 AWS 資源，以提高效能、可靠性、安全性和成本效益。所有使用者皆可使用四個 Trusted Advisor 檢查；具有商業或企業支援計劃的使用者可使用超過 50 個檢查。如需詳細資訊，請參閱AWS Trusted Advisor。

  Trusted Advisor 具有與 Amazon ECS 相關的這些檢查：

  - 容錯能力，指出您在單一可用區域中有執行中的服務。
  - 容錯能力，指出您尚未針對多個可用區域使用分散置放策略。
- AWS Compute Optimizer 是一種 服務，可分析 AWS 資源的組態和使用率指標。這會報告您的資源是否已為最佳化，並產生最佳化建議，以降低成本並改善工作負載的效能。

  如需詳細資訊，請參閱AWS Compute Optimizer Amazon ECS 的建議。

使用 CloudWatch 監控 Amazon ECS

您可以使用 Amazon CloudWatch 來監控 Amazon ECS 資源，前者會收集來自 Amazon ECS 的原始資料，並處理為可讀且近乎即時的指標。這些統計資料記錄會保留兩週，讓您可存取歷史資訊，且能更清楚叢集或服務的執行方式。Amazon ECS 指標資料會自動以 1 分鐘的期間傳送到 CloudWatch。如需有關 CloudWatch 的詳細資訊，請參閱 Amazon CloudWatch 使用者指南。

Amazon ECS 為叢集和服務提供免費指標。您可以為叢集開啟 Amazon ECS CloudWatch Container Insights 以取得每個任務的指標，包括 CPU、記憶體和 EBS 檔案系統使用率，但需額外付費。如需更多 Container Insights 的相關資訊，請參閱 使用具有增強可觀測性的 Container Insights 監控 Amazon ECS 容器。

考量事項

使用 Amazon ECS CloudWatch 指標時，應考慮下列事項。

- Fargate 上託管的任何 Amazon ECS 服務都會自動擁有 CloudWatch CPU 和記憶體使用率指標，因此您不需要採取任何手動步驟。

- 對於 Amazon EC2 執行個體上託管的任何 Amazon ECS 任務或服務，Amazon EC2 執行個體需要版本 `1.4.0`或更新版本 (Linux) `1.0.0`或更新版本 (Windows) 的容器代理程式，才能產生 CloudWatch 指標。不過，我們建議您使用最新版的容器代理程式。如需檢查代理程式版本及更新至最新版本的資訊，請參閱「更新 Amazon ECS 容器代理程式」。
- 可靠 CloudWatch 指標的最低 Docker 版本是 Docker 版本`20.10.13`和更新版本。
- 您的 Amazon EC2 執行個體也需要您啟動 Amazon EC2 執行個體之 IAM 角色的`ecs:StartTelemetrySession`許可。如果您在 CloudWatch 指標可用於 Amazon ECS 之前建立 Amazon ECS 容器執行個體 IAM 角色，您可能需要新增此許可。如需容器執行個體 IAM 角色和連接容器執行個體之受管 IAM 政策的相關資訊，請參閱 Amazon ECS 容器執行個體 IAM 角色。
- 您可以在 Amazon ECS 容器代理程式組態中設定 ，在 Amazon EC2 執行個體上停用 CloudWatch 指標集合。 `ECS_DISABLE_METRICS=true`如需詳細資訊，請參閱Amazon ECS 容器代理程式組態。

建議的指標

Amazon ECS 提供免費的 CloudWatch 指標，可讓您用來監控資源。CPU 和記憶體保留，以及整個叢集的 CPU、記憶體和 EBS 檔案系統使用率，以及叢集中服務的 CPU、記憶體和 EBS 檔案系統使用率，都可以使用這些指標進行測量。針對 GPU 工作負載，您可以量測叢集整體的 GPU 保留。

叢集中 Amazon ECS 任務託管的基礎設施會決定哪些指標可用。對於 Fargate 基礎設施上託管的任務，Amazon ECS 提供 CPU、記憶體和 EBS 檔案系統使用率指標，以協助監控您的服務。對於 EC2 執行個體上託管的任務，Amazon ECS 會在叢集和服務層級提供 CPU、記憶體和 GPU 保留指標，以及 CPU 和記憶體使用率指標。您需要監控單獨構成基礎設施的 Amazon EC2 執行個體。如需監控 Amazon EC2 執行個體的詳細資訊，請參閱《Amazon EC2 使用者指南》中的監控 Amazon EC2。

如需搭配 Amazon ECS 使用的建議警示相關資訊，請參閱《Amazon CloudWatch Logs 使用者指南》中的下列其中一項：

- Amazon ECS
- Amazon ECS 搭配 Container Insights

檢視 Amazon ECS 指標

在叢集中執行資源之後，您可以在 Amazon ECS 和 CloudWatch 主控台上檢視指標。Amazon ECS 主控台提供您的叢集和服務指標 24 小時的最小值、最大值和平均值檢視。CloudWatch 主控台提供微調和可自訂的資源顯示，以及服務中所執行任務的數量。

Amazon ECS 主控台

Amazon ECS 服務 CPU 和記憶體使用率指標皆可在 Amazon ECS 主控台中取得。提供的服務指標檢視會顯示前 24 小時期間的平均值、最小值、最大值，並提供 5 分鐘間隔的資料點。如需詳細資訊，請參閱Amazon ECS 服務使用率指標。

1. 開啟主控台，網址為 https://console.aws.amazon.com/ecs/v2。
2. 選取您要檢視其指標的叢集。
3. 決定要檢視的指標。

從 檢視指標	步驟
叢集	在叢集詳細資訊頁面上，選擇指標索引標籤。也提供連結給 CloudWatch 主控台，以便在開啟 CloudWatch Container Insights 指標時檢視這些指標。
服務	在叢集詳細資訊頁面上的服務索引標籤上，選取服務。然後，您可以在運作狀態和指標索引標籤上取得這些指標。

CloudWatch 主控台

對於 Fargate 啟動類型，也可以在 CloudWatch 主控台上檢視 Amazon ECS 服務指標。主控台提供最詳細的 Amazon ECS 指標檢視，您可以修改這些檢視以符合您的需求。您可以檢視服務使用率和服務 RUNNING 任務計數。

對於 EC2 啟動類型，也可以在 CloudWatch 主控台上檢視 Amazon ECS 叢集和服務指標。主控台提供最詳細的 Amazon ECS 指標檢視，您可以修改這些檢視以符合您的需求。

如需如何檢視指標的詳細資訊，請參閱《Amazon CloudWatch 使用者指南》中的檢視可用指標。

Amazon ECS CloudWatch 指標

您可以使用 CloudWatch 用量指標來提供您帳戶的資源用量可見度。使用這些指標，以 CloudWatch 圖表和儀表板視覺化目前的服務使用狀況。

`CPUReservation`

在叢集或服務中預留的 CPU 單位百分比。

CPU 保留 (篩選`ClusterName`) 是以叢集上 Amazon ECS 任務預留的總 CPU 單位數，除以叢集中註冊的所有 Amazon EC2 執行個體的總 CPU 單位數。只有 `ACTIVE`或 `DRAINING` 狀態中的 Amazon EC2 執行個體會影響 CPU 保留指標。指標僅支援託管在 Amazon EC2 執行個體上的任務。

有效維度：`ClusterName`。

實用統計資料：平均值、最小值、最大值

單位：百分比。

`CPUUtilization`

叢集或服務使用的 CPU 單位百分比。

叢集層級 CPU 使用率 (篩選`ClusterName`) 的測量方式為叢集上 Amazon ECS 任務正在使用的總 CPU 單位，除以叢集中註冊的所有 Amazon EC2 執行個體的總 CPU 單位。只有 `ACTIVE`或 `DRAINING` 狀態中的 Amazon EC2 執行個體會影響 CPU 保留指標。叢集層級指標僅支援託管在 Amazon EC2 執行個體上的任務。

服務層級 CPU 使用率 (篩選， `ServiceName`) `ClusterName`是以屬於服務的任務正在使用的 CPU 單位總數，除以屬於服務的任務預留的 CPU 單位總數。Amazon EC2 執行個體和 Fargate 上託管的任務支援服務層級指標。

有效維度：`ClusterName`、`ServiceName`。

實用統計資料：平均值、最小值、最大值

單位：百分比。

`MemoryReservation`

由叢集內執行中任務保留的記憶體的百分比。

叢集記憶體保留的測量方式為叢集上 Amazon ECS 任務預留的總記憶體，除以叢集中註冊的所有 Amazon EC2 執行個體的記憶體總量。此指標只能由 篩選ClusterName。只有 ACTIVE或 DRAINING 狀態中的 Amazon EC2 執行個體會影響記憶體保留指標。叢集層級記憶體保留指標僅支援 Amazon EC2 執行個體上託管的任務。

> **Note**
>
> 計算記憶體使用率時，如果MemoryReservation已指定 ，則會在計算中使用它，而不是總記憶體。

有效維度：ClusterName。

實用統計資料：平均值、最小值、最大值

單位：百分比。

MemoryUtilization

叢集或服務使用的記憶體百分比。

叢集層級記憶體使用率 （ 由 篩選ClusterName) 的測量方式為叢集上 Amazon ECS 任務正在使用的總記憶體，除以叢集中註冊的所有 Amazon EC2 執行個體的總記憶體。只有 ACTIVE或 DRAINING 狀態中的 Amazon EC2 執行個體會影響記憶體使用率指標。叢集層級指標僅支援託管在 Amazon EC2 執行個體上的任務。

服務層級記憶體使用率 （ 由 篩選ClusterName，ServiceName) 的測量方式是屬於服務的任務正在使用的總記憶體，除以屬於服務的任務預留的總記憶體。Amazon EC2 執行個體和 Fargate 上託管的任務支援服務層級指標。

有效維度：ClusterName、ServiceName。

實用統計資料：平均值、最小值、最大值

單位：百分比。

EBSFilesystemUtilization

服務中任務所使用的 Amazon EBS 檔案系統百分比。

服務層級 EBS 檔案系統使用率指標 （ 由 篩選ServiceName) ClusterName是以屬於服務的任務正在使用的 EBS 檔案系統總量，除以分配給屬於服務之所有任務的 EBS 檔案系統儲存總量。服務

層級 EBS 檔案系統使用率指標僅適用於託管在 Amazon EC2 執行個體 （使用容器代理程式版本 1.79.0) 和 Fargate （使用平台版本 1.4.0) 上且已連接 EBS 磁碟區的任務。

> **Note**
>
> 對於託管在 Fargate 上的任務，磁碟上有只有 Fargate 使用的空間。Fargate 使用的空間沒有成本，但您會使用 等工具看到此額外的儲存df體。

有效維度：ClusterName, ServiceName。

實用統計資料：平均值、最小值、最大值

單位：百分比。

GPUReservation

由叢集內執行中任務保留的 GPU 可用總數的百分比。

叢集層級 GPU 保留指標的測量方式為叢集上 Amazon ECS 任務預留的 GPUs 數量，除以叢集中已註冊 GPUs 的所有 Amazon EC2 執行個體上可用的 GPUs 總數。只有 ACTIVE或 DRAINING 狀態中的 Amazon EC2 執行個體會影響 GPU 保留指標。

有效維度：ClusterName。

實用統計資料：平均值、最小值、最大值

所有統計資料：平均值、最小值、最大值、總和、範例計數。

單位：百分比。

ActiveConnectionCount

從用戶端到在共享所選 DiscoveryName 的任務中執行之 Amazon ECS Service Connect Proxy 的作用中同時連線總數。

僅當您已設定 Amazon ECS Service Connect 時，才能使用此指標。

有效維度：DiscoveryName 和 DiscoveryName, ServiceName, ClusterName。

實用統計資料：平均、最小、最大、總和。

單位：計數。

`NewConnectionCount`

從用戶端到在共享所選 `DiscoveryName` 的任務中執行之 Amazon ECS Service Connect Proxy 的新建立連線總數。

僅當您已設定 Amazon ECS Service Connect 時，才能使用此指標。

有效維度：`DiscoveryName` 和 `DiscoveryName, ServiceName, ClusterName`。

實用統計資料：平均、最小、最大、總和。

單位：計數。

`ProcessedBytes`

Service Connect Proxy 處理的輸入流量總位元組數。

僅當您已設定 Amazon ECS Service Connect 時，才能使用此指標。

有效維度：`DiscoveryName` 和 `DiscoveryName, ServiceName, ClusterName`。

實用統計資料：平均、最小、最大、總和。

單位：位元組。

`RequestCount`

Service Connect Proxy 處理的輸入流量請求數。

僅當您已設定 Amazon ECS Service Connect 時，才能使用此指標。

您也需要在任務定義的`appProtocol`連接埠映射中設定 。

有效維度：`DiscoveryName` 和 `DiscoveryName, ServiceName, ClusterName`。

實用統計資料：平均、最小、最大、總和。

單位：計數。

`GrpcRequestCount`

Service Connect Proxy 處理的 gRPC 輸入流量請求數。

僅當您已設定 Amazon ECS Service Connect 且 `appProtocol` 在任務定義的連接埠映射中為 `GRPC` 時，才能使用此指標。

有效維度：`DiscoveryName` 和 `DiscoveryName, ServiceName, ClusterName`。

實用統計資料：平均、最小、最大、總和。

單位：計數。

`HTTPCode_Target_2XX_Count`

這些任務中應用程式所產生的編號 200 至 299 的 HTTP 回應代碼數。這些任務是目標。此指標只會計算應用程式在這些任務中傳送至 Service Connect Proxy 的回應，而非直接傳送的回應。

僅當您已設定 Amazon ECS Service Connect 且 `appProtocol` 在任務定義的連接埠映射中為 `HTTP` 或 `HTTP2` 時，才能使用此指標。

有效維度：`TargetDiscoveryName` 和 `TargetDiscoveryName, ServiceName, ClusterName`。

實用統計資料：平均、最小、最大、總和。

單位：計數。

`HTTPCode_Target_3XX_Count`

這些任務中應用程式所產生的編號 300 至 399 的 HTTP 回應代碼數。這些任務是目標。此指標只會計算應用程式在這些任務中傳送至 Service Connect Proxy 的回應，而非直接傳送的回應。

僅當您已設定 Amazon ECS Service Connect 且 `appProtocol` 在任務定義的連接埠映射中為 `HTTP` 或 `HTTP2` 時，才能使用此指標。

有效維度：`TargetDiscoveryName` 和 `TargetDiscoveryName, ServiceName, ClusterName`。

實用統計資料：平均、最小、最大、總和。

單位：計數。

`HTTPCode_Target_4XX_Count`

這些任務中應用程式所產生的編號 400 至 499 的 HTTP 回應代碼數。這些任務是目標。此指標只會計算應用程式在這些任務中傳送至 Service Connect Proxy 的回應，而非直接傳送的回應。

僅當您已設定 Amazon ECS Service Connect 且 `appProtocol` 在任務定義的連接埠映射中為 `HTTP` 或 `HTTP2` 時，才能使用此指標。

有效維度：`TargetDiscoveryName` 和 `TargetDiscoveryName, ServiceName, ClusterName`。

實用統計資料：平均值、最小值、最大值、總和

單位：計數。

`HTTPCode_Target_5XX_Count`

這些任務中應用程式所產生的編號 500 至 599 的 HTTP 回應代碼數。這些任務是目標。此指標只會計算應用程式在這些任務中傳送至 Service Connect Proxy 的回應，而非直接傳送的回應。

僅當您已設定 Amazon ECS Service Connect 且 `appProtocol` 在任務定義的連接埠映射中為 `HTTP` 或 `HTTP2` 時，才能使用此指標。

實用統計資料：平均、最小、最大、總和。

單位：計數。

`RequestCountPerTarget`

共享所選 `DiscoveryName` 的每個目標接收的平均請求數。

僅當您已設定 Amazon ECS Service Connect 時，才能使用此指標。

有效維度：`TargetDiscoveryName` 和 `TargetDiscoveryName, ServiceName, ClusterName`。

實用統計資料：平均值。

單位：計數。

`TargetProcessedBytes`

Service Connect Proxy 處理的總位元組數。

僅當您已設定 Amazon ECS Service Connect 時，才能使用此指標。

有效維度：`TargetDiscoveryName` 和 `TargetDiscoveryName, ServiceName, ClusterName`。

實用統計資料：平均、最小、最大、總和。

單位：位元組。

`TargetResponseTime`

應用程式請求處理的延遲。從請求到達目標任務中的 Service Connect Proxy，直到 Proxy 收到目標應用程式回應所經過的時間 (以毫秒為單位)。

僅當您已設定 Amazon ECS Service Connect 時，才能使用此指標。

有效維度：TargetDiscoveryName 和 TargetDiscoveryName, ServiceName, ClusterName。

實用統計資料：平均值、最小值、最大值。

所有統計資料：平均值、最小值、最大值、總和、範例計數。

單位：毫秒。

ClientTLSNegotiationErrorCount

TLS 連線失敗的總次數。此指標只會在啟用 TLS 時使用。

僅當您已設定 Amazon ECS Service Connect 時，才能使用此指標。

有效維度： DiscoveryName和 DiscoveryName、ServiceName、ClusterName。

實用統計資料：平均、最小、最大、總和。

單位：計數。

TargetTLSNegotiationErrorCount

TLS 連線因缺少用戶端憑證、 AWS Private CA 驗證失敗或 SAN 驗證失敗而失敗的次數。此指標只會在啟用 TLS 時使用。

僅當您已設定 Amazon ECS Service Connect 時，才能使用此指標。

有效維度：ServiceName、 ClusterNameTargetDiscoveryName和 TargetDiscoveryName。

實用統計資料：平均、最小、最大、總和。

單位：計數。

Amazon ECS 指標的維度

Amazon ECS 指標使用 AWS/ECS 命名空間，並提供下列維度的指標。Amazon ECS 只會針對 RUNNING 狀態中有任務的資源傳送指標。舉例來說，如果您有叢集包含一個服務，但該服務沒有處於 RUNNING 狀態的任務，則不會傳送任何指標至 CloudWatch。如果您有兩個服務，其中一個有執行中的任務，而另一個沒有，則只會傳送有執行中任務之服務的指標。

`ClusterName`

此維度可篩選您為指定叢集中所有資源請求的資料。所有 Amazon ECS 指標皆以 `ClusterName` 篩選。

`ServiceName`

此維度可篩選您為指定叢集中的指定服務的所有資源請求的資料。

`DiscoveryName`

此維度會篩選您為流量指標請求，且傳送至所有 Amazon ECS 叢集內指定 Service Connect 探索名稱的資料。

請注意，執行中容器內的特定連接埠可以擁有多個探索名稱。

`DiscoveryName, ServiceName, ClusterName`

此維度會篩選您為流量指標請求，且傳送至具有此探索名稱並由此叢集中此服務所建立之任務內指定 Service Connect 探索名稱的資料。

如果您在不同命名空間的多個服務中重複使用相同的探索名稱，請使用此維度來查看特定服務的輸入流量指標。

請注意，執行中容器內的特定連接埠可以擁有多個探索名稱。

`TargetDiscoveryName`

此維度會篩選您為流量指標請求，且傳送至所有 Amazon ECS 叢集內指定 Service Connect 探索名稱的資料。

與 `DiscoveryName` 不同，這些流量指標僅會測量傳送至此 `DiscoveryName`，且來自在此命名空間中具有 Service Connect 組態的其他 Amazon ECS 任務的輸入流量。這包括服務使用僅限用戶端或用戶端-伺服器 Service Connect 組態建立的任務。

請注意，執行中容器內的特定連接埠可以擁有多個探索名稱。

`TargetDiscoveryName, ServiceName, ClusterName`

此維度會篩選您為流量指標請求，且傳送至指定 Service Connect 探索名稱的資料，但僅會計算來自此叢集中此服務所建立任務的流量。

使用此維度來查看來自另一服務中特定用戶端的輸入流量指標。

與 DiscoveryName, ServiceName, ClusterName 不同，這些流量指標僅會測量傳送至此 DiscoveryName，且來自在此命名空間中具有 Service Connect 組態的其他 Amazon ECS 任務的輸入流量。這包括服務使用僅限用戶端或用戶端-伺服器 Service Connect 組態建立的任務。

請注意，執行中容器內的特定連接埠可以擁有多個探索名稱。

AWS Fargate 用量指標

您可以使用 CloudWatch 用量指標來提供您帳戶的資源用量可見度。使用這些指標，以 CloudWatch 圖表和儀表板視覺化目前的服務使用狀況。

AWS Fargate 用量指標對應至 AWS 服務配額。您可以設定警示，在您的用量接近服務配額時發出警示。如需 Fargate 的服務配額詳細資訊，請參閱 AWS Fargate 服務配額。

AWS Fargate 會在AWS/Usage命名空間中發佈下列指標。

指標	描述
ResourceCount	您的帳戶中正在執行的特定資源總數。資源由與指標相關聯的維度定義。

以下維度用於強化 AWS Fargate發佈的用量指標。

維度	描述
Service	包含 資源 AWS 的服務名稱。對於 AWS Fargate 用量指標，此維度的值為 Fargate。
Type	正在報告的實體類型。目前， AWS Fargate 用量指標的唯一有效值是 Resource。
Resource	正在執行的資源類型。正在執行的資源類型。目前， AWS Fargate 用量指標的唯一有效值是vCPU傳回執行中執行個體的相關資訊。
Class	正在追蹤的資源類別。正在追蹤的資源類別。對於使用 vCPU 做為資源維度值的 AWS Fargate 用量指標，有效值為 Standard/OnDemand 和 Standard/Spot 。

您可以使用 Service Quotas 主控台，在圖形上視覺化您的用量，並設定警示，在 AWS Fargate 用量接近服務配額時提醒您。如需有關如何建立 CloudWatch 警示，以便在接近配額值閾值時通知您的資訊，請參閱Service Quotas 《 使用者指南》中的 Service Quotas 和 Amazon CloudWatch 警示

.

Amazon ECS 叢集保留指標

相較於叢集中每個作用中之容器執行個體所登錄之彙總的 CPU、記憶體和 GPU，叢集保留指標是依叢集上所有 Amazon ECS 任務保留的 CPU、記憶體和 GPU 百分比計量。只有處於 ACTIVE 或 DRAINING 狀態的容器執行個體會影響叢集保留指標。此指標僅用於在 EC2 執行個體上託管任務或服務所在的叢集。在託管任務的叢集上不支援 AWS Fargate。

```
                              (Total CPU units reserved by tasks in cluster) x 100
Cluster CPU reservation =
 ---------------------------------------------------------------
                           (Total CPU units registered by container instances in
 cluster)
```

```
                                 (Total MiB of memory reserved by tasks in cluster x
 100)
Cluster memory reservation =
 ---------------------------------------------------------------------
                              (Total MiB of memory registered by container instances in
 cluster)
```

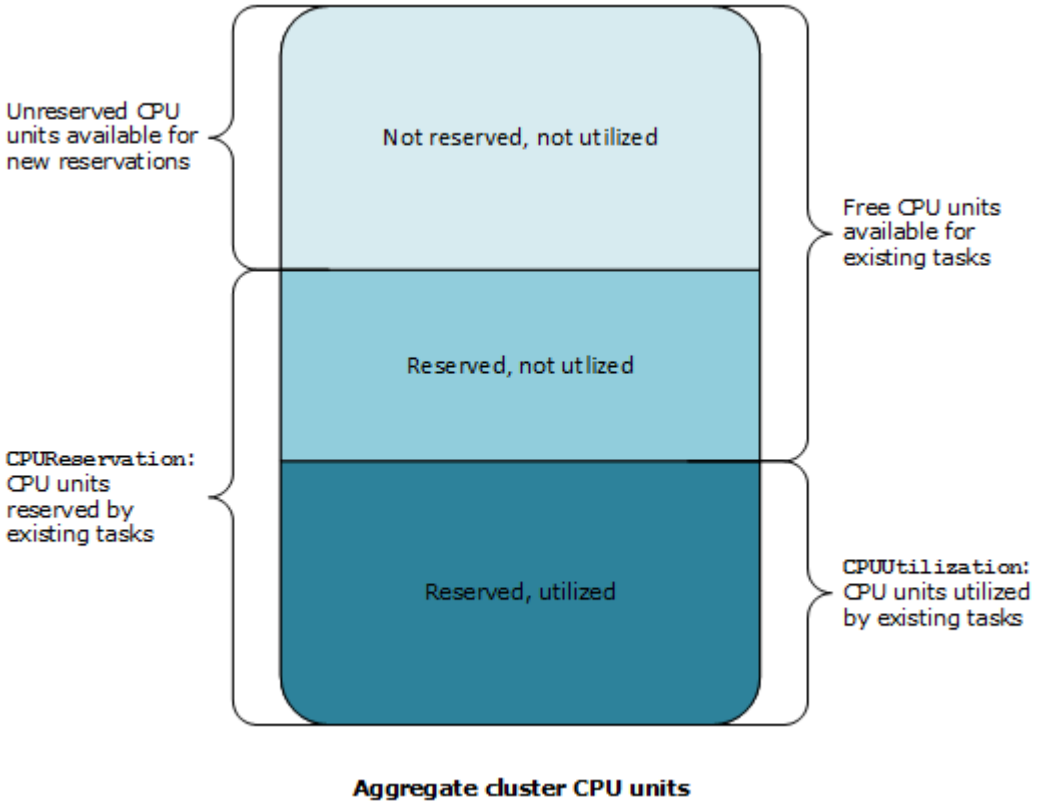
```
                                (Total GPUs reserved by tasks in cluster x 100)
Cluster GPU reservation =
 ---------------------------------------------------------------------
                              (Total GPUs registered by container instances in cluster)
```

當您在叢集中執行任務時，Amazon ECS 會剖析其任務定義，並保留其容器定義中指定的彙總 CPU 單位、記憶體 MiB 和 GPU。Amazon ECS 每分鐘都會計算目前在叢集中，每項執行中之任務保留的 CPU 單位、記憶體 MiB 和 GPU 的數量。為叢集上所有執行中之任務保留的 CPU、記憶體和 GPU 總數量皆會計算，這些數字會以叢集總登錄資源百分比的形式回報給 CloudWatch。如果您指定任務定義中的軟性限制 (memoryReservation)，則其將用於計算保留的記憶體數量。否則，會使用硬性限制 (memory)。叢集中由任務所保留的總 MiB 記憶體也包括暫存檔案系統 (tmpfs) 磁碟區大小以及 sharedMemorySize (若任務定義中有定義)。如需硬性和軟性限制、共用記憶體大小及 tmpfs 磁碟區大小的詳細資訊，請參閱任務定義參數。

例如，叢集已登錄兩個作用中的容器執行個體：c4.4xlarge 執行個體和 c4.large 執行個體。c4.4xlarge 執行個體在叢集中登錄為 16,384 個 CPU 單位和 30,158 MiB 記憶體。c4.large 執行個體登錄為 2,048 個 CPU 單位和 3,768 MiB 記憶體。此叢集的彙總資源為 18,432 個 CPU 單位和 33,926 MiB 記憶體。

如果任務定義保留 1,024 個 CPU 單位和 2,048 MiB 記憶體，在此叢集上有十項任務使用此任務定義啟動 (目前未執行任何其他任務)，總共保留 10,240 個 CPU 單位和 20,480 MiB 記憶體。此會回報到 CloudWatch，做為叢集的 55% CPU 保留和 60% 記憶體保留。

下圖顯示叢集中的總登錄 CPU 單位，及其保留和使用率對現有的任務和新任務置放所代表的意義。下方 (已保留、已使用) 和中間 (已保留、未使用) 區塊代表為叢集上現有執行中任務所保留的總 CPU 單位，或 CPUReservation CloudWatch 指標。下方區塊代表叢集上執行中任務實際使用的保留 CPU 單位或 CPUUtilization CloudWatch 指標。上方的區塊表示現有任務未保留的 CPU 單位，這些 CPU 單位可供新任務置放使用。現有的任務也可以利用這些未保留的 CPU 單位，如果其 CPU 資源需求增加。如需詳細資訊，請參閱「cpu」任務定義參數文件。



Amazon ECS 叢集使用率指標

叢集使用率指標適用於 CPU、記憶體，以及當 EBS 磁碟區連接至您的任務時，EBS 檔案系統使用率。這些指標僅適用於在 Amazon EC2 執行個體上託管任務或服務所在的叢集。在託管任務的叢集上不支援它們 AWS Fargate。

Amazon ECS 叢集層級 CPU 和記憶體使用率指標

CPU 和記憶體使用率是以叢集上所有任務使用的 CPU 和記憶體百分比來衡量，與註冊到叢集的每個作用中 Amazon EC2 執行個體所註冊的彙總 CPU 和記憶體相比。只有 ACTIVE或 DRAINING 狀態中的 Amazon EC2 執行個體會影響叢集使用率指標。

```
                                  (Total CPU units used by tasks in cluster) x 100
Cluster CPU utilization =
 ------------------------------------------------------------
                           (Total CPU units registered by container instances in
 cluster)
```

```
                                     (Total MiB of memory used by tasks in cluster x
 100)
Cluster memory utilization =
 ------------------------------------------------------------------
                              (Total MiB of memory registered by container instances in
 cluster)
```

每一分鐘，每個 Amazon EC2 執行個體上的 Amazon ECS 容器代理程式都會計算目前用於該 Amazon EC2 執行個體上執行之每個任務的 CPU 單位和記憶體 MiB 數量，並將此資訊回報給 Amazon ECS。叢集上所有執行中之任務使用的 CPU 和記憶體總數量皆會計算，這些數字會以在叢集總登錄資源中所佔的百分比回報給 CloudWatch。

例如，叢集有兩個作用中的 Amazon EC2 執行個體已註冊，一個c4.4xlarge執行個體和一個c4.large執行個體。c4.4xlarge 執行個體會使用 16,384 CPU 單位和記憶體的 30,158 MiB 註冊到叢集。c4.large 執行個體會向 2,048 CPU 單位和記憶體的 3,768 MiB 註冊。此叢集的彙總資源是 18,432 CPU 單位和記憶體的 33,926 MiB。

如果在此叢集上執行十個任務，且每個任務耗用 1,024 CPU 單位和 MiB 2,048 記憶體，則叢集上總共會使用 10,240 CPU 單位和 MiB 20,480 記憶體。此會回報到 CloudWatch，做為叢集的 55% CPU 使用率和 60% 記憶體使用率。

Amazon ECS 叢集層級 Amazon EBS 檔案系統使用率

叢集層級 EBS 檔案系統使用率指標的測量方式為叢集上執行的任務正在使用的 EBS 檔案系統總量，除以叢集中所有任務配置的 EBS 檔案系統儲存總量。

```
                                        (Total GB of EBS filesystem used by tasks in
 cluster x 100)
```

```
Cluster EBS filesystem utilization =
 ------------------------------------------------------------
                                      (Total GB of EBS filesystem allocated to tasks
 in cluster)
```

Amazon ECS 服務使用率指標

服務使用率指標適用於 CPU、記憶體，以及當 EBS 磁碟區連接至您的任務時，EBS 檔案系統使用率。服務層級指標支援在 Amazon EC2 執行個體和 Fargate 上託管任務的服務。

服務層級 CPU 和記憶體使用率

與服務任務定義中指定的 CPU 和記憶體相比，CPU 和記憶體使用率是以叢集上屬於服務的 Amazon ECS 任務所使用的 CPU 和記憶體百分比來衡量。

```
                                    (Total CPU units used by tasks in service) x 100
Service CPU utilization =
 ------------------------------------------------------------------------------
                          (Total CPU units specified in task definition) x (number of
 tasks in service)
```

```
                                    (Total MiB of memory used by tasks in service) x
 100
Service memory utilization =
 ------------------------------------------------------------------------------
                             (Total MiB of memory specified in task definition) x
 (number of tasks in service)
```

Amazon ECS 容器代理程式每分鐘會計算目前用於服務所擁有之每個任務的 CPU 單位和記憶體 MiB 數量，並將此資訊回報給 Amazon ECS。叢集上執行中的服務擁有之所有任務使用的 CPU 和記憶體總量皆會計算，這些數字會以在服務的任務定義中，為服務指定的總資源中所佔的百分比，回報給 CloudWatch。如果您指定軟性限制 (`memoryReservation`)，則會用以計算保留的記憶體數量。否則，會使用硬性限制 (`memory`)。如需硬性與軟性限制的詳細資訊，請參閱任務大小。

例如，服務的任務定義為其所有容器指定總共 512 個 CPU 單位和 1,024 MiB 記憶體 (使用硬性限制 `memory` 參數)。服務的所需計數為 1 項執行中的任務，此服務正使用 1 個 `c4.large` 容器執行個體 (總共 2,048 個 CPU 單位和 3,768 MiB 記憶體) 在叢集上執行，且叢集上目前未執行其他任務。雖然任務指定的是 512 CPU 單位，因為其是在具有 2,048 CPU 單位的容器執行個體上唯一的執行中任務，該任務可以使用指定數量 (2,048/512) 多達四次。然而，指定記憶體 1,024 MiB 是硬性限制且不得被超過，因此在此情況下，服務記憶體使用率不得超過 100%。

如果之前的範例使用軟性限制 `memoryReservation` 而非硬性限制 `memory` 參數，則服務的任務如有需要，可以使用超過所指定的 1,024 MiB 記憶體。在這種情況下，服務的記憶體使用率可能超過 100%。

如果您的應用程式短時間記憶體使用率驟升，由於 Amazon ECS 每分鐘會收集多個資料點，然後將這些資料點匯集至傳送到 CloudWatch 的單一資料點，因此您會發現服務記憶體使用率並未增加。

如果此任務在某期間內執行 CPU 密集型工作，並使用所有 2,048 個可用的 CPU 單位和 512 MiB 的記憶體，則服務會報告 400% 的 CPU 使用率和 50% 的記憶體使用率。如果任務閒置並使用 128 個 CPU 單位和 128 MiB 記憶體，則服務會回報 25% 的 CPU 使用率和 12.5% 的記憶體使用率。

> **Note**
>
> 在此範例中，只有當 CPU 單位在容器層級定義時，CPU 使用率會超過 100%。若您在任務層級定義 CPU 單位，使用率將不會高於定義的任務層級限制。

服務層級 EBS 檔案系統使用率

服務層級 EBS 檔案系統使用率是以屬於服務的任務所使用的 EBS 檔案系統總量，除以分配給屬於服務之所有任務的 EBS 檔案系統儲存總量。

```
                                    (Total GB of EBS filesystem used by tasks in the
 service x 100)
Service EBS filesystem utilization =
 ------------------------------------------------------------------
                                    (Total GB of EBS filesystem allocated to tasks
 in the service)
```

服務 **RUNNING** 任務計數

您可以使用 CloudWatch 指標來檢視您服務中 RUNNING 狀態的任務數量。例如，您可以為此指標設定 CloudWatch 警示，在服務的執行中之任務數量低於指定值時提醒您。

Amazon ECS CloudWatch Container Insights 中的服務 **RUNNING** 任務計數

當您使用 Amazon ECS CloudWatch Container Insights 時，每個叢集和每個服務都有一個「正在執行的任務數」(`RunningTaskCount`) 指標。您可以選擇加入 `containerInsights` 帳戶設定可對所有建立的新叢集啟用容器洞見、在個別叢集上可於建立叢集期間開啟叢集設定，或可在現有叢集上使用 UpdateClusterSettings API。CloudWatch Container Insights 收集的指標會以自訂指標的方式計費。如需了解有關 CloudWatch 定價的詳細資訊，請參閱 CloudWatch 定價。

如要查看此指標，請參閱《Amazon CloudWatch 使用者指南》中的 Amazon ECS Container Insights 指標。

使用 EventBridge 自動化對 Amazon ECS 錯誤的回應

使用 Amazon EventBridge，您可以自動化您的 AWS 服務，並自動回應系統事件，例如應用程式可用性問題或資源變更。 AWS 服務的事件會以接近即時的方式傳送到 EventBridge。您可編寫簡單的規則，來指示您在意的事件，以及當事件符合規則時所要自動執行的動作。可以自動設定的動作如下：

- 將事件新增至 CloudWatch Logs 中的日誌群組
- 叫用 AWS Lambda 函數
- 調用 Amazon EC2 執行命令
- 將事件轉傳至 Amazon Kinesis Data Streams
- 啟用 AWS Step Functions 狀態機器
- 通知 Amazon SNS 主題或 Amazon Simple Queue Service (Amazon SQS) 佇列

如需詳細資訊，請參閱 《Amazon EventBridge 使用者指南》中的 Amazon EventBridge 入門。

您可以使用適用於 EventBridge 的 Amazon ECS 事件，接收有關您的 Amazon ECS 叢集目前狀態的近乎即時通知。如果您的任務使用的 EC2 啟動類型，則可以看到容器執行個體的狀態，以及這些容器執行個體上執行之所有任務的目前狀態。如果您的任務使用 Fargate 啟動類型，您可以查看容器執行個體的狀態。

您可以使用 EventBridge，在 Amazon ECS 上建置自訂排程器，而自訂排程器負責協調叢集之間的任務，並以近乎即時的方式監控叢集狀態。您不需要排定和監控持續輪詢 Amazon ECS 服務以處理狀態變更的程式碼，而是改為使用任意 EventBridge 目標，以非同步的方式處理 Amazon ECS 狀態變更。目標可能包括 AWS Lambda Amazon Simple Queue Service、Amazon Simple Notification Service 或 Amazon Kinesis Data Streams。

Amazon ECS 事件資料流可確保每個事件會至少交付一次。如果傳送重複的事件，則事件會提供足夠的資訊來識別重複項目。如需詳細資訊，請參閱處理 Amazon ECS 事件。

由於事件的排序具有相對性，因此您可以清楚得知事件發生的時間 (相對於其他事件)。

主題

- Amazon ECS 事件
- 處理 Amazon ECS 事件

Amazon ECS 事件

Amazon ECS 會追蹤每個任務和服務的狀態。如果任務或服務的狀態變更，就會產生事件並將事件傳送到 Amazon EventBridge。系統會將這些事件分類為任務狀態變更事件和服務動作事件。下列各節將更詳細地說明這些事件及其可能原因。

Amazon ECS 會產生下列類型的事件並將其傳送至 EventBridge：容器執行個體狀態變更事件、任務狀態變更事件、服務動作和服務部署狀態變更事件。

- 容器執行個體狀態變更
- 任務狀態變更
- Deployment state change (部署狀態變更)
- 服務動作

> **Note**
>
> Amazon ECS 日後可能會新增其他事件類型、來源和詳細資訊。如果您要在程式碼中取消序列化事件 JSON 資料，請確定您的應用程式已準備好處理未知屬性，以避免新增這些其他屬性時發生問題。

在某些情況下，會針對相同的活動產生多個事件。例如，在容器執行個體開始任務時，即會為新的任務產生任務狀態變更事件。系統會針對容器執行個體上可用資源 (例如，CPU、記憶體和可用連接埠) 中的變更，對帳戶產生容器執行個體狀態變更事件。同樣地，如果終止容器執行個體，則會針對容器執行個體、容器代理連線狀態以及在容器執行個體上執行的每個任務產生事件。

容器狀態變更和任務狀態變更事件包含兩個 `version` 欄位：一個位於事件本體，另一個位於事件的 `detail` 物件中。以下說明這兩個欄位之間的差異：

- 事件本體的 `version` 欄位在所有事件中皆設為 `0`。如需 EventBridge 參數的詳細資訊，請參閱《Amazon EventBridge 使用者指南》中的AWS 服務事件中繼資料。
- 事件之 `detail` 物件中的 `version` 欄位說明相關資源的版本。每次資源變更狀態時，此版本都會遞增。因為事件可以傳送多次，所以此欄位可讓您識別重複的事件。重複事件在 `detail` 物件中具有相同版本。如果您使用 EventBridge 複寫 Amazon ECS 容器執行個體和任務狀態，您可以將 Amazon ECS APIs 所回報的資源版本與 EventBridge 中回報的資源版本（在 `detail` 物件內）進行比較，以確認事件串流中的版本是否為最新版本。

服務動作事件只包含本體中的 version 欄位。

服務動作事件在 2 個不同欄位中指定服務：

- 對於 產生的事件create-service，服務位於 serviceName 欄位中。
- 對於 產生的事件update-service，服務位於 service 欄位中。

如果您針對服務事件使用自動化工具，則需要為這兩個欄位編寫程式碼。

您可以使用下列 EventBridge 規則修補來考慮來自 create-service、 和 的服務動作事件update-service。

```
{
  "source": ["aws.ecs"],
  "detail-type": ["ECS Service Action"],
  "detail": {
    "eventName": ["CreateService", "UpdateService"],
    "serviceName": ["your-service-name"],
    "service": ["your-service-name"]
  }
}
```

如需有關如何整合 Amazon ECS 與 EventBridge 的其他資訊，請參閱整合 Amazon EventBridge 與 Amazon ECS。

Amazon ECS 容器執行個體狀態變更事件

下列案例會引發容器執行個體狀態變更事件：

您可以呼叫 StartTask、RunTask 或 StopTask API 操作 (可直接或使用 AWS Management Console 或軟體開發套件執行此動作)。

在容器執行個體上放置或停用任務會修改容器執行個體上的可用資源 (例如 CPU、記憶體和可用連接埠)。

Amazon ECS 服務排程器會啟動或停止任務。

在容器執行個體上放置或停用任務會修改容器執行個體上的可用資源 (例如 CPU、記憶體和可用連接埠)。

Amazon ECS 容器代理程式會呼叫 SubmitTaskStateChange API 操作取得所需狀態為 RUNNING 之任務的 STOPPED 狀態。

Amazon ECS 容器代理程式會監控容器執行個體上的任務狀態，並報告任何狀態變更。如果應為 RUNNING 的任務轉換為 STOPPED，則代理程式會釋出配給至已停止任務的資源 (例如 CPU、記憶體和可用連接埠)。

您可以使用 DeregisterContainerInstance API 操作取消註冊容器執行個體，無論是直接，還是使用 AWS Management Console 或 SDKs。

取消登錄容器代理程式會變更容器執行個體的狀態以及 Amazon ECS 容器代理程式的連線狀態。

在 EC2 執行個體停止時，任務即停止。

當您停止容器執行個體時，在其上執行的任務會轉換為 STOPPED 狀態。

Amazon ECS 容器代理程式第一次登錄容器執行個體。

Amazon ECS 容器代理程式第一次登錄容器執行個體時 (在啟動時，或手動初次執行時)，這會建立執行個體的狀態變更事件。

Amazon ECS 容器代理會與 Amazon ECS 連線或中斷連線。

Amazon ECS 容器代理程式與 Amazon ECS 後端連線或中斷連線時，會變更容器執行個體的 agentConnected 狀態。

> **Note**
>
> Amazon ECS 容器代理程式在正常操作的過程中，每小時會中斷連線並重新連線數次，因此應預期會發生代理程式連線事件。這些事件並不表示容器代理程式或容器執行個體發生問題。

升級執行個體上的 Amazon ECS 容器代理程式。

容器執行個體詳細資訊包含容器代理版本的物件。如果您升級代理，則此版本資訊會變更並產生事件。

Example 容器執行個體狀態變更事件

容器執行個體狀態變更事件以下列格式交付。以下 detail 區段類似於從 Amazon Elastic Container Service API 參考中的 DescribeContainerInstances API 操作所返回的 ContainerInstance 物件。如需

EventBridge 參數的詳細資訊，請參閱《Amazon EventBridge 使用者指南》中的AWS 服務事件中繼資料。

```
{
  "version": "0",
  "id": "8952ba83-7be2-4ab5-9c32-6687532d15a2",
  "detail-type": "ECS Container Instance State Change",
  "source": "aws.ecs",
  "account": "111122223333",
  "time": "2016-12-06T16:41:06Z",
  "region": "us-east-1",
  "resources": [
    "arn:aws:ecs:us-east-1:111122223333:container-instance/
b54a2a04-046f-4331-9d74-3f6d7f6ca315"
  ],
  "detail": {
    "agentConnected": true,
    "attributes": [
      {
        "name": "com.amazonaws.ecs.capability.logging-driver.syslog"
      },
      {
        "name": "com.amazonaws.ecs.capability.task-iam-role-network-host"
      },
      {
        "name": "com.amazonaws.ecs.capability.logging-driver.awslogs"
      },
      {
        "name": "com.amazonaws.ecs.capability.logging-driver.json-file"
      },
      {
        "name": "com.amazonaws.ecs.capability.docker-remote-api.1.17"
      },
      {
        "name": "com.amazonaws.ecs.capability.privileged-container"
      },
      {
        "name": "com.amazonaws.ecs.capability.docker-remote-api.1.18"
      },
      {
        "name": "com.amazonaws.ecs.capability.docker-remote-api.1.19"
      },
      {
```

```
        "name": "com.amazonaws.ecs.capability.ecr-auth"
      },
      {
        "name": "com.amazonaws.ecs.capability.docker-remote-api.1.20"
      },
      {
        "name": "com.amazonaws.ecs.capability.docker-remote-api.1.21"
      },
      {
        "name": "com.amazonaws.ecs.capability.docker-remote-api.1.22"
      },
      {
        "name": "com.amazonaws.ecs.capability.docker-remote-api.1.23"
      },
      {
        "name": "com.amazonaws.ecs.capability.task-iam-role"
      }
    ],
    "clusterArn": "arn:aws:ecs:us-east-1:111122223333:cluster/default",
    "containerInstanceArn": "arn:aws:ecs:us-east-1:111122223333:container-instance/
b54a2a04-046f-4331-9d74-3f6d7f6ca315",
    "ec2InstanceId": "i-f3a8506b",
    "registeredResources": [
      {
        "name": "CPU",
        "type": "INTEGER",
        "integerValue": 2048
      },
      {
        "name": "MEMORY",
        "type": "INTEGER",
        "integerValue": 3767
      },
      {
        "name": "PORTS",
        "type": "STRINGSET",
        "stringSetValue": [
          "22",
          "2376",
          "2375",
          "51678",
          "51679"
        ]
      },
```

```
      {
        "name": "PORTS_UDP",
        "type": "STRINGSET",
        "stringSetValue": []
      }
    ],
    "remainingResources": [
      {
        "name": "CPU",
        "type": "INTEGER",
        "integerValue": 1988
      },
      {
        "name": "MEMORY",
        "type": "INTEGER",
        "integerValue": 767
      },
      {
        "name": "PORTS",
        "type": "STRINGSET",
        "stringSetValue": [
          "22",
          "2376",
          "2375",
          "51678",
          "51679"
        ]
      },
      {
        "name": "PORTS_UDP",
        "type": "STRINGSET",
        "stringSetValue": []
      }
    ],
    "status": "ACTIVE",
    "version": 14801,
    "versionInfo": {
      "agentHash": "aebcbca",
      "agentVersion": "1.13.0",
      "dockerVersion": "DockerVersion: 1.11.2"
    },
    "updatedAt": "2016-12-06T16:41:06.991Z"
  }
```

```
}
```

Amazon ECS 任務狀態變更事件

下列案例會引發任務狀態變更事件：

您可以呼叫 StartTask、RunTask 或 StopTask API 操作 (可直接或使用 AWS Management Console、 AWS CLI或軟體開發套件執行此動作)。

啟動或停止任務會建立新的任務資源，或修改現有任務資源的狀態。

Amazon ECS 服務排程器會啟動或停止任務。

啟動或停止任務會建立新的任務資源，或修改現有任務資源的狀態。

Amazon ECS 容器代理程序會呼叫 SubmitTaskStateChange API 操作。

對於 Amazon EC2 啟動類型，Amazon ECS 容器代理程式會監控容器執行個體上的任務狀態。Amazon ECS 容器代理程式會報告任何狀態變更。狀態變更可能包括從 PENDING 到 RUNNING 或從 RUNNING 到 STOPPED 的變更。

您可以使用 DeregisterContainerInstance API 操作和 force旗標強制取消基礎容器執行個體的註冊，無論是直接，還是使用 AWS Management Console 或 SDKs。

取消登錄容器代理程式會變更容器執行個體的狀態以及 Amazon ECS 容器代理程式的連線狀態。如果任務是在容器執行個體上執行，則必須將 force 旗標設定為允許取消登錄。這會停止執行個體上的所有任務。

停止或終止基礎容器執行個體。

當您停止或終止容器執行個體時，在其上執行的任務會轉換為 STOPPED 狀態。

任務中的容器變更狀態。

Amazon ECS 容器代理程序會監控任務內的容器狀態。例如，如果任務內的執行容器停止，則此容器狀態變更會產生事件。

使用 Fargate Spot 容量提供者的任務會收到終止通知。

當任務使用 FARGATE_SPOT 容量提供者且因 Spot 中斷而停止時，會產生任務狀態變更事件。

Example 任務狀態變更事件

任務狀態變更事件以下列格式交付。以下 `detail` 區段類似於從 Amazon Elastic Container Service API 參考中的 DescribeTasks API 操作所返回的 Task 物件。如果您的容器使用 Amazon ECR, 託管的映像，則會傳回 `imageDigest` 欄位。

> **Note**
>
> `createdAt`、`connectivityAt`、`pullStartedAt`、`startedAt`、`pullStoppedAt` 和 `updatedAt` 欄位的值是 `DescribeTasks` 動作之回應中的 UNIX 時間戳記，而在任務狀態變更事件中，這些值是 ISO 字串時間戳記。

如需 EventBridge 參數的詳細資訊，請參閱《Amazon EventBridge 使用者指南》中的AWS 服務事件中繼資料。

如需如何設定 Amazon EventBridge 事件規則的相關資訊，該規則只會擷取任務因其中一個基本容器終止而停止執行的任務事件，請參閱 傳送 Amazon ECS 任務已停止事件的 Amazon Simple Notification Service 提醒

```
{
    "version": "0",
    "id": "3317b2af-7005-947d-b652-f55e762e571a",
    "detail-type": "ECS Task State Change",
    "source": "aws.ecs",
    "account": "111122223333",
    "time": "2020-01-23T17:57:58Z",
    "region": "us-west-2",
    "resources": [
        "arn:aws:ecs:us-west-2:111122223333:task/FargateCluster/
c13b4cb40f1f4fe4a2971f76ae5a47ad"
    ],
    "detail": {
        "attachments": [
            {
                "id": "1789bcae-ddfb-4d10-8ebe-8ac87ddba5b8",
                "type": "eni",
                "status": "ATTACHED",
                "details": [
                    {
                        "name": "subnetId",
                        "value": "subnet-abcd1234"
```

```
                    },
                    {
                        "name": "networkInterfaceId",
                        "value": "eni-abcd1234"
                    },
                    {
                        "name": "macAddress",
                        "value": "0a:98:eb:a7:29:ba"
                    },
                    {
                        "name": "privateIPv4Address",
                        "value": "10.0.0.139"
                    }
                ]
            }
        ],
        "availabilityZone": "us-west-2c",
        "clusterArn": "arn:aws:ecs:us-west-2:111122223333:cluster/FargateCluster",
        "containers": [
            {
                "containerArn": "arn:aws:ecs:us-west-2:111122223333:container/
cf159fd6-3e3f-4a9e-84f9-66cbe726af01",
                "lastStatus": "RUNNING",
                "name": "FargateApp",
                "image": "111122223333.dkr.ecr.us-west-2.amazonaws.com/hello-
repository:latest",
                "imageDigest":
 "sha256:74b2c688c700ec95a93e478cdb959737c148df3fbf5ea706abe0318726e885e6",
                "runtimeId":
 "ad64cbc71c7fb31c55507ec24c9f77947132b03d48d9961115cf24f3b7307e1e",
                "taskArn": "arn:aws:ecs:us-west-2:111122223333:task/FargateCluster/
c13b4cb40f1f4fe4a2971f76ae5a47ad",
                "networkInterfaces": [
                    {
                        "attachmentId": "1789bcae-ddfb-4d10-8ebe-8ac87ddba5b8",
                        "privateIpv4Address": "10.0.0.139"
                    }
                ],
                "cpu": "0"
            }
        ],
        "createdAt": "2020-01-23T17:57:34.402Z",
        "launchType": "FARGATE",
        "cpu": "256",
```

```
        "memory": "512",
        "desiredStatus": "RUNNING",
        "group": "family:sample-fargate",
        "lastStatus": "RUNNING",
        "overrides": {
            "containerOverrides": [
                {
                    "name": "FargateApp"
                }
            ]
        },
        "connectivity": "CONNECTED",
        "connectivityAt": "2020-01-23T17:57:38.453Z",
        "pullStartedAt": "2020-01-23T17:57:52.103Z",
        "startedAt": "2020-01-23T17:57:58.103Z",
        "pullStoppedAt": "2020-01-23T17:57:55.103Z",
        "updatedAt": "2020-01-23T17:57:58.103Z",
        "taskArn": "arn:aws:ecs:us-west-2:111122223333:task/FargateCluster/
c13b4cb40f1f4fe4a2971f76ae5a47ad",
        "taskDefinitionArn": "arn:aws:ecs:us-west-2:111122223333:task-definition/
sample-fargate:1",
        "version": 4,
        "platformVersion": "1.3.0"
    }
}
```

Amazon ECS 服務動作事件

Amazon ECS 會傳送具有 ECS Service Action (ECS 服務動作) 詳細類型的服務動作事件。與容器執行個體和任務狀態變更事件不同，服務動作事件不會在 `details` 回應欄位中包含版本號碼。以下這個事件模式會用來建立 Amazon ECS 服務動作事件的 EventBridge 規則。如需詳細資訊，請參閱《Amazon EventBridge 使用者指南》中的 EventBridge 入門。 EventBridge

```
{
    "source": [
        "aws.ecs"
    ],
    "detail-type": [
        "ECS Service Action"
    ]
}
```

Amazon ECS 會傳送具有 INFO、WARN 和 ERROR 事件類型的事件。以下是服務動作事件。

具有 **INFO** 事件類型的服務動作事件

SERVICE_STEADY_STATE

服務運作狀況良好且有所需的任務數量，因而達到穩定狀態。服務排程器會定期報告狀態，因此您可能會多次收到此訊息。

TASKSET_STEADY_STATE

任務集運作狀況良好且有所需的任務數量，因而達到穩定的狀態。

CAPACITY_PROVIDER_STEADY_STATE

與服務相關聯的容量提供者會達到穩定狀態。

SERVICE_DESIRED_COUNT_UPDATED

服務排程器更新適用於服務或任務集的計算所需計數時。當使用者手動更新所需的計數時，不會傳送此事件。

TASKS_STOPPED

服務已停止執行中的任務。

SERVICE_DEPLOYMENT_IN_PROGRESS

服務部署正在進行中。服務部署可以是復原或新的服務修訂。

SERVICE_DEPLOYMENT_COMPLETED

服務部署處於穩定狀態且已完成。服務部署可以是復原，也可以是部署更新的服務修訂。

具有 **WARN** 事件類型的服務動作事件

SERVICE_TASK_START_IMPAIRED

此服務無法成功持續啟動任務。

SERVICE_DISCOVERY_INSTANCE_UNHEALTHY

使用服務探索的服務包含運作狀況不良的任務。服務排程器會偵測到服務登錄中的任務運作狀況不良。

VPC_LATTICE_TARGET_UNHEALTHY

使用 VPC Lattice 的服務已偵測到 VPC Lattice 的其中一個目標運作狀態不佳。

具有 **ERROR** 事件類型的服務動作事件

`SERVICE_DAEMON_PLACEMENT_CONSTRAINT_VIOLATED`

使用 DAEMON 服務排程器策略的服務中的任務不再符合此服務的放置限制策略。

`ECS_OPERATION_THROTTLED`

由於 Amazon ECS API 節流限制，服務排程器已經受到節流。

`SERVICE_DISCOVERY_OPERATION_THROTTLED`

由於 AWS Cloud Map API 節流限制，服務排程器已調節。這可能會在設定為使用服務探索的服務上發生。

`SERVICE_TASK_PLACEMENT_FAILURE`

服務排程器無法放置任務。原因的描述位於 `reason` 欄位中。

產生此服務事件的常見原因是因為叢集缺乏放置任務的資源。例如，可用的容器執行個體沒有足夠的 CPU 或記憶體容量，或沒有可用的容器執行個體。另一個常見原因是，若容器執行個體上的 Amazon ECS 容器代理程式連線中斷，會造成排程器無法放置任務。

`SERVICE_TASK_CONFIGURATION_FAILURE`

由於組態錯誤，服務排程器無法放置任務。原因的描述位於 `reason` 欄位中。

產生此服務事件的常見原因是因為標籤正在套用至服務，但使用者或角色尚未在區域中選擇使用新的 Amazon Resource Name (ARN) 格式。如需詳細資訊，請參閱Amazon Resource Names (ARNs) 和 IDs。另一個常見的原因是 Amazon ECS 無法擔任所提供的任務 IAM 角色。

`SERVICE_HEALTH_UNKNOWN`

服務無法描述任務的運作狀態資料。

`SERVICE_DEPLOYMENT_FAILED`

服務部署未達到穩定。當 CloudWatch 觸發或斷路器偵測到服務部署失敗時，就會發生這種情況。

Example 服務穩定狀態事件

任務穩定狀態事件以下列格式交付。如需 EventBridge 參數的詳細資訊，請參閱《Amazon EventBridge 使用者指南》中的 EventBridge 中的事件。 EventBridge

```
{
    "version": "0",
```

```
    "id": "af3c496d-f4a8-65d1-70f4-a69d52e9b584",
    "detail-type": "ECS Service Action",
    "source": "aws.ecs",
    "account": "111122223333",
    "time": "2019-11-19T19:27:22Z",
    "region": "us-west-2",
    "resources": [
        "arn:aws:ecs:us-west-2:111122223333:service/default/servicetest"
    ],
    "detail": {
        "eventType": "INFO",
        "eventName": "SERVICE_STEADY_STATE",
        "clusterArn": "arn:aws:ecs:us-west-2:111122223333:cluster/default",
        "createdAt": "2019-11-19T19:27:22.695Z"
    }
}
```

Example 容量提供者穩定狀態事件

容量提供者穩定狀態事件以下列格式交付。

```
{
    "version": "0",
    "id": "b9baa007-2f33-0eb1-5760-0d02a572d81f",
    "detail-type": "ECS Service Action",
    "source": "aws.ecs",
    "account": "111122223333",
    "time": "2019-11-19T19:37:00Z",
    "region": "us-west-2",
    "resources": [
        "arn:aws:ecs:us-west-2:111122223333:service/default/servicetest"
    ],
    "detail": {
        "eventType": "INFO",
        "eventName": "CAPACITY_PROVIDER_STEADY_STATE",
        "clusterArn": "arn:aws:ecs:us-west-2:111122223333:cluster/default",
        "capacityProviderArns": [
            "arn:aws:ecs:us-west-2:111122223333:capacity-provider/ASG-tutorial-
capacity-provider"
        ],
        "createdAt": "2019-11-19T19:37:00.807Z"
    }
}
```

Example 服務任務啟動受損事件

服務任務啟動受損事件以下列格式交付。

```
{
    "version": "0",
    "id": "57c9506e-9d21-294c-d2fe-e8738da7e67d",
    "detail-type": "ECS Service Action",
    "source": "aws.ecs",
    "account": "111122223333",
    "time": "2019-11-19T19:55:38Z",
    "region": "us-west-2",
    "resources": [
        "arn:aws:ecs:us-west-2:111122223333:service/default/servicetest"
    ],
    "detail": {
        "eventType": "WARN",
        "eventName": "SERVICE_TASK_START_IMPAIRED",
        "clusterArn": "arn:aws:ecs:us-west-2:111122223333:cluster/default",
        "createdAt": "2019-11-19T19:55:38.725Z"
    }
}
```

Example 服務任務放置失敗事件

服務任務放置失敗事件以下列格式交付。如需詳細資訊，請參閱《Amazon EventBridge 使用者指南》中的 EventBridge。 EventBridge

在下列範例中，此任務嘗試使用 FARGATE_SPOT 容量提供者，但服務排程器無法取得任何 Fargate Spot 容量。

```
{
    "version": "0",
    "id": "ddca6449-b258-46c0-8653-e0e3a6d0468b",
    "detail-type": "ECS Service Action",
    "source": "aws.ecs",
    "account": "111122223333",
    "time": "2019-11-19T19:55:38Z",
    "region": "us-west-2",
    "resources": [
        "arn:aws:ecs:us-west-2:111122223333:service/default/servicetest"
    ],
    "detail": {
```

```
        "eventType": "ERROR",
        "eventName": "SERVICE_TASK_PLACEMENT_FAILURE",
        "clusterArn": "arn:aws:ecs:us-west-2:111122223333:cluster/default",
        "capacityProviderArns": [
            "arn:aws:ecs:us-west-2:111122223333:capacity-provider/FARGATE_SPOT"
        ],
        "reason": "RESOURCE:FARGATE",
        "createdAt": "2019-11-06T19:09:33.087Z"
    }
}
```

在下列 EC2 啟動類型範例中，嘗試在容器容器執行個體 2dd1b186f39845a584488d2ef155c131 上啟動任務，但由於 CPU 不足，服務排程器無法放置任務。

```
{
  "version": "0",
  "id": "ddca6449-b258-46c0-8653-e0e3a6d0468b",
  "detail-type": "ECS Service Action",
  "source": "aws.ecs",
  "account": "111122223333",
  "time": "2019-11-19T19:55:38Z",
  "region": "us-west-2",
  "resources": [
    "arn:aws:ecs:us-west-2:111122223333:service/default/servicetest"
  ],
  "detail": {
    "eventType": "ERROR",
    "eventName": "SERVICE_TASK_PLACEMENT_FAILURE",
    "clusterArn": "arn:aws:ecs:us-west-2:111122223333:cluster/default",
    "containerInstanceArns": [
    "arn:aws:ecs:us-west-2:111122223333:container-instance/
default/2dd1b186f39845a584488d2ef155c131"
    ],
    "reason": "RESOURCE:CPU",
    "createdAt": "2019-11-06T19:09:33.087Z"
  }
}
```

Amazon ECS 服務部署狀態變更事件

Amazon ECS 會傳送具有 ECS Deployment State Change (ECS 部署狀態變更) 詳細類型的服務部署變更狀態事件。以下這個事件模式會用來建立 Amazon ECS 服務部署狀態變更事件的 EventBridge

規則。如需建立 EventBridge 規則的詳細資訊，請參閱《Amazon EventBridge 使用者指南》中的 Amazon EventBridge 入門。

```
{
    "source": [
        "aws.ecs"
    ],
    "detail-type": [
        "ECS Deployment State Change"
    ]
}
```

Amazon ECS 會傳送具有 INFO 和 ERROR 事件類型的事件。以下是服務部署狀態變更事件。

`SERVICE_DEPLOYMENT_IN_PROGRESS`

服務部署正在進行。此事件會被傳送用於初始部署和回復部署。

`SERVICE_DEPLOYMENT_COMPLETED`

服務部署完成。一旦服務在部署後達到穩定狀態，此事件將被傳送。

`SERVICE_DEPLOYMENT_FAILED`

服務部署失敗。此事件會被傳送用於開啟部署斷路器邏輯的服務。

Example 服務部署正在進行事件

服務部署正在進行事件會在初始和回復部署均已開始時交付。兩者之間的區別位於 `reason` 欄位。如需 EventBridge 參數的詳細資訊，請參閱《Amazon EventBridge 使用者指南》中的AWS 服務事件中繼資料。

下列顯示初始部署開始的輸出範例。

```
{
   "version": "0",
   "id": "ddca6449-b258-46c0-8653-e0e3a6EXAMPLE",
   "detail-type": "ECS Deployment State Change",
   "source": "aws.ecs",
   "account": "111122223333",
   "time": "2020-05-23T12:31:14Z",
   "region": "us-west-2",
```

```
   "resources": [
        "arn:aws:ecs:us-west-2:111122223333:service/default/servicetest"
   ],
   "detail": {
        "eventType": "INFO",
        "eventName": "SERVICE_DEPLOYMENT_IN_PROGRESS",
        "deploymentId": "ecs-svc/123",
        "updatedAt": "2020-05-23T11:11:11Z",
        "reason": "ECS deployment deploymentId in progress."
   }
}
```

下列顯示回復部署開始的輸出範例。reason 欄位提供服務轉返至的部署 ID。

```
{
   "version": "0",
   "id": "ddca6449-b258-46c0-8653-e0e3aEXAMPLE",
   "detail-type": "ECS Deployment State Change",
   "source": "aws.ecs",
   "account": "111122223333",
   "time": "2020-05-23T12:31:14Z",
   "region": "us-west-2",
   "resources": [
        "arn:aws:ecs:us-west-2:111122223333:service/default/servicetest"
   ],
   "detail": {
        "eventType": "INFO",
        "eventName": "SERVICE_DEPLOYMENT_IN_PROGRESS",
        "deploymentId": "ecs-svc/123",
        "updatedAt": "2020-05-23T11:11:11Z",
        "reason": "ECS deployment circuit breaker: rolling back to
 deploymentId deploymentID."
   }
}
```

Example 服務部署完成事件

任務部署完成狀態事件以下列格式交付。如需詳細資訊，請參閱透過取代任務來部署 Amazon ECS 服務。

```
{
   "version": "0",
```

```
    "id": "ddca6449-b258-46c0-8653-e0e3aEXAMPLE",
    "detail-type": "ECS Deployment State Change",
    "source": "aws.ecs",
    "account": "111122223333",
    "time": "2020-05-23T12:31:14Z",
    "region": "us-west-2",
    "resources": [
         "arn:aws:ecs:us-west-2:111122223333:service/default/servicetest"
    ],
    "detail": {
         "eventType": "INFO",
         "eventName": "SERVICE_DEPLOYMENT_COMPLETED",
         "deploymentId": "ecs-svc/123",
         "updatedAt": "2020-05-23T11:11:11Z",
         "reason": "ECS deployment deploymentID completed."
    }
}
```

Example 服務部署失敗事件

任務部署失敗狀態事件以下列格式交付。服務部署失敗狀態事件只會被傳送用於開啟部署斷路器邏輯的服務。如需詳細資訊，請參閱透過取代任務來部署 Amazon ECS 服務。

```
{
    "version": "0",
    "id": "ddca6449-b258-46c0-8653-e0e3aEXAMPLE",
    "detail-type": "ECS Deployment State Change",
    "source": "aws.ecs",
    "account": "111122223333",
    "time": "2020-05-23T12:31:14Z",
    "region": "us-west-2",
    "resources": [
         "arn:aws:ecs:us-west-2:111122223333:service/default/servicetest"
    ],
    "detail": {
         "eventType": "ERROR",
         "eventName": "SERVICE_DEPLOYMENT_FAILED",
         "deploymentId": "ecs-svc/123",
         "updatedAt": "2020-05-23T11:11:11Z",
         "reason": "ECS deployment circuit breaker: task failed to start."
    }
}
```

處理 Amazon ECS 事件

Amazon ECS 會傳送事件至少一次。這表示您可能會收到指定事件的多個副本。此外，可能不會依事件發生順序將事件交付至事件接聽程式。

為了正確排序事件，每個事件的 `detail` 區段都會包含 `version` 屬性。每次資源變更狀態時，此 `version` 都會遞增。重複事件在 `detail` 物件中具有相同的 `version`。如果您使用 EventBridge 複寫 Amazon ECS 容器執行個體和任務狀態，您可以將 Amazon ECS APIs 所回報的資源版本與 EventBridge 中`version`回報的資源版本進行比較，以驗證事件串流中的版本是否為最新版本。較高版本屬性編號的事件應該視為比較低版本編號的事件晚發生。

範例：處理 AWS Lambda 函式中的事件

下列範例顯示使用 Python 3.9 撰寫的 Lambda 函數，以同時擷取任務和容器執行個體狀態變更事件，並將其儲存至兩個 Amazon DynamoDB 資料表中的其中一個：

- ECSCtrInstanceState：存放容器執行個體的最新狀態。資料表 ID 是容器執行個體的 `containerInstanceArn` 值。
- ECSTaskState：存放任務的最新狀態。資料表 ID 是任務的 `taskArn` 值。

```
import json
import boto3

def lambda_handler(event, context):
    id_name = ""
    new_record = {}

    # For debugging so you can see raw event format.
    print('Here is the event:')
    print((json.dumps(event)))

    if event["source"] != "aws.ecs":
       raise ValueError("Function only supports input from events with a source type
 of: aws.ecs")

    # Switch on task/container events.
    table_name = ""
    if event["detail-type"] == "ECS Task State Change":
        table_name = "ECSTaskState"
        id_name = "taskArn"
        event_id = event["detail"]["taskArn"]
```

```
    elif event["detail-type"] == "ECS Container Instance State Change":
        table_name = "ECSCtrInstanceState"
        id_name =  "containerInstanceArn"
        event_id = event["detail"]["containerInstanceArn"]
    else:
        raise ValueError("detail-type for event is not a supported type. Exiting
without saving event.")

    new_record["cw_version"] = event["version"]
    new_record.update(event["detail"])

    # "status" is a reserved word in DDB, but it appears in containerPort
    # state change messages.
    if "status" in event:
        new_record["current_status"] = event["status"]
        new_record.pop("status")


    # Look first to see if you have received a newer version of an event ID.
    # If the version is OLDER than what you have on file, do not process it.
    # Otherwise, update the associated record with this latest information.
    print("Looking for recent event with same ID...")
    dynamodb = boto3.resource("dynamodb", region_name="us-east-1")
    table = dynamodb.Table(table_name)
    saved_event = table.get_item(
        Key={
            id_name : event_id
        }
    )
    if "Item" in saved_event:
        # Compare events and reconcile.
        print(("EXISTING EVENT DETECTED: Id " + event_id + " - reconciling"))
        if saved_event["Item"]["version"] < event["detail"]["version"]:
            print("Received event is a more recent version than the stored event -
updating")
            table.put_item(
                Item=new_record
            )
        else:
            print("Received event is an older version than the stored event -
ignoring")
    else:
        print(("Saving new event - ID " + event_id))
```

```
        table.put_item(
            Item=new_record
        )
```

下列 Fargate 範例顯示以 Python 3.9 編寫的 Lambda 函數，可擷取任務狀態變更事件並將其儲存至下列 Amazon DynamoDB 資料表：

```
import json
import boto3

def lambda_handler(event, context):
    id_name = ""
    new_record = {}

    # For debugging so you can see raw event format.
    print('Here is the event:')
    print((json.dumps(event)))

    if event["source"] != "aws.ecs":
       raise ValueError("Function only supports input from events with a source type
 of: aws.ecs")

    # Switch on task/container events.
    table_name = ""
    if event["detail-type"] == "ECS Task State Change":
        table_name = "ECSTaskState"
        id_name = "taskArn"
        event_id = event["detail"]["taskArn"]
    else:
        raise ValueError("detail-type for event is not a supported type. Exiting
 without saving event.")

    new_record["cw_version"] = event["version"]
    new_record.update(event["detail"])

    # "status" is a reserved word in DDB, but it appears in containerPort
    # state change messages.
    if "status" in event:
        new_record["current_status"] = event["status"]
        new_record.pop("status")


    # Look first to see if you have received a newer version of an event ID.
```

```
    # If the version is OLDER than what you have on file, do not process it.
    # Otherwise, update the associated record with this latest information.
    print("Looking for recent event with same ID...")
    dynamodb = boto3.resource("dynamodb", region_name="us-east-1")
    table = dynamodb.Table(table_name)
    saved_event = table.get_item(
        Key={
            id_name : event_id
        }
    )
    if "Item" in saved_event:
        # Compare events and reconcile.
        print(("EXISTING EVENT DETECTED: Id " + event_id + " - reconciling"))
        if saved_event["Item"]["version"] < event["detail"]["version"]:
            print("Received event is a more recent version than the stored event -
updating")
            table.put_item(
                Item=new_record
            )
        else:
            print("Received event is an older version than the stored event -
ignoring")
    else:
        print(("Saving new event - ID " + event_id))

        table.put_item(
            Item=new_record
        )
```

使用具有增強可觀測性的 Container Insights 監控 Amazon ECS 容器

Container Insights 會從容器化應用程式和微服務收集、彙總和摘要指標和日誌。 提供所有 Container Insights 指標，以及額外的任務和容器指標。

Container Insights 探索叢集中所有執行中的容器，並在效能堆疊的每一層收集效能資料。營運資料的收集形式為「效能日誌事件」。這些項目使用的是結構化的 JSON 結構描述，可大規模擷取並存放高基數資料。CloudWatch 可透過這些資料，建立在叢集、服務和任務層級更高層級彙總的指標並做為 CloudWatch 指標。指標包含 CPU、記憶體、磁碟和網路這類資源的使用率。在 CloudWatch 自動儀表板中可取得這些指標。

指標只會反映在指定時間範圍內執行任務的資源。舉例來說，如果您有叢集包含一個服務，但該服務沒有處於 RUNNING 狀態的任務，則不會傳送任何指標至 CloudWatch。如果您有兩個服務，其中一個有執行中的任務，而另一個沒有，則只會傳送有執行中任務之服務的指標。

在 2024 年 12 月 2 日， AWS 發行了 Container Insights 並增強了 Amazon ECS 的可觀測性。此版本支援使用 Amazon EC2 和 Fargate 啟動類型的 Amazon ECS 叢集增強型可觀測性 在 Amazon ECS 上設定 Container Insights 增強型可觀測性之後，Container Insights 會自動收集從叢集層級到環境中容器層級的詳細基礎設施遙測，並在已整理的儀表板中顯示這些關鍵效能資料，消除可觀測性設定中的繁重負荷。如需如何設定具有增強可觀測性之 Container Insights 的詳細資訊，請參閱 具有增強可觀測性的容器洞見。

我們建議您使用具有增強可觀測性的 Container Insights，而不是 Container Insights，因為它可在您的容器環境中提供詳細的可見性，從而縮短解決的平均時間。如需詳細資訊，請參閱《Amazon CloudWatch 使用者指南》中的具有增強型可觀測性指標的 Amazon ECS Container Insights。
Amazon CloudWatch

您可以檢視的事件包括 Amazon ECS 傳送至 Amazon EventBridge 的事件。如需詳細資訊，請參閱 Amazon ECS 事件。

> **Important**
>
> CloudWatch Container Insights 收集的指標會以自訂指標的方式計費。如需了解有關 CloudWatch 定價的詳細資訊，請參閱 CloudWatch 定價。

使用容器運作狀態檢查判斷 Amazon ECS 任務運作狀態

建立任務定義時，您可以設定容器的運作狀態檢查。運作狀態檢查是在容器上執行的命令，可驗證應用程式運作狀態和可用性。

Amazon ECS 容器代理程式只會監控並報告任務定義中指定的運作狀態檢查。Amazon ECS 不會監控嵌入在容器映像中但未在容器定義中指定的 Docker 運作狀態檢查。容器定義中指定的運作狀態檢查參數，會覆寫任何存在於容器影像中的 Docker 運作狀態檢查。

在任務定義中定義運作狀態檢查時，容器會在容器內執行運作狀態檢查程序，然後評估結束程式碼以判斷應用程式的運作狀態。

運作狀態檢查包含下列參數：

- 命令 – 容器執行的命令，用於判斷它是否正常運作。此字串陣列的開頭可以是 CMD，如此能直接執行命令引數；或是 CMD-SHELL，藉以使用容器預設的 shell 來執行命令。
- 間隔 – 每次運作狀態檢查之間的時間 （以秒為單位）。
- 逾時 – 在被視為失敗之前等待運作狀態檢查成功的期間 （以秒為單位）。
- 重試 – 在容器被視為運作狀態不佳之前重試失敗運作狀態檢查的次數。
- 開始期間 – 在運作狀態檢查失敗之前，提供容器開機時間的選用寬限期會計入重試次數上限。

  如果運作狀態檢查在 startPeriod 內成功，則代表容器運作狀態良好，之後的任何故障都會計入，累積至重試次數上限。

如需有關如何在任務定義中指定運作狀態檢查的資訊，請參閱 運作狀態檢查。

以下說明容器的可能運作狀態值：

- HEALTHY - 容器運作狀態檢查已成功通過。
- UNHEALTHY - 容器運作狀態檢查失敗。
- UNKNOWN - 正在評估容器運作狀態檢查，沒有定義容器運作狀態檢查，或 Amazon ECS 沒有容器的運作狀態。

運作狀態檢查命令會在容器上執行。因此，您必須在容器映像中包含命令。

運作狀態檢查會透過容器的回送界面連接至應用程式，網址為 localhost或 127.0.0.1。的結束碼0表示成功，非零結束碼表示失敗。

使用容器運作狀態檢查時，請考慮下列事項：

- 容器運作狀態檢查需要 1.17.0 版或更新版本的 Amazon ECS 容器代理程式。
- 如果您使用 Linux 平台版本 1.1.0或更新版本或 Windows 平台版本 1.1.0或更新版本，則 Fargate 任務支援容器運作狀態檢查

Amazon ECS 如何判斷任務運作狀態

必要且任務定義中具有運作狀態檢查命令的容器，是唯一可視為決定任務運作狀態的容器。

下列規則會依序評估：

1. 如果一個必要容器的狀態為 UNHEALTHY，則任務狀態為 UNHEALTHY。

2. 如果一個必要容器的狀態為 UNKNOWN，則任務狀態為 UNKNOWN。

3. 如果所有必要容器的狀態為 HEALTHY，則任務狀態為 HEALTHY。

請考慮以下任務運作狀態範例，其中包含 2 個必要容器。

容器 1 運作狀態	容器 2 運作狀態	任務運作狀態
UNHEALTHY	UNKNOWN	UNHEALTHY
UNHEALTHY	HEALTHY	UNHEALTHY
HEALTHY	UNKNOWN	UNKNOWN
HEALTHY	HEALTHY	HEALTHY

請考慮以下具有 3 個容器的任務運作狀態範例。

容器 1 運作狀態	容器 2 運作狀態	容器 3 運作狀態	任務運作狀態
UNHEALTHY	UNKNOWN	UNKNOWN	UNHEALTHY
UNHEALTHY	UNKNOWN	HEALTHY	UNHEALTHY
UNHEALTHY	HEALTHY	HEALTHY	UNHEALTHY
HEALTHY	UNKNOWN	HEALTHY	UNKNOWN
HEALTHY	UNKNOWN	UNKNOWN	UNKNOWN
HEALTHY	HEALTHY	HEALTHY	HEALTHY

客服人員中斷連線對運作狀態檢查的影響

如果 Amazon ECS 容器代理程式與 Amazon ECS 服務中斷連線，這不會導致容器轉換為 UNHEALTHY 狀態。這是根據設計，以確保容器在代理程式重新啟動或暫時無法使用期間持續執行。運作狀態檢查狀態是來自 Amazon ECS 代理程式的「上次接聽」回應，因此如果在中斷連線HEALTHY之前考慮容器，

則該狀態會一直保留，直到代理程式重新連線並發生另一個運作狀態檢查為止。沒有對容器運作狀態檢查的狀態做出任何假設。

檢視 Amazon ECS 容器運作狀態

您可以在 主控台中檢視容器運作狀態，並在DescribeTasks回應中使用 API。如需詳細資訊，請參閱《Amazon Elastic Container Service API 參考》中的 DescribeTasks。

如果您使用容器的記錄，例如 Amazon CloudWatch Logs，您可以設定運作狀態檢查命令，將容器運作狀態輸出轉送至您的日誌。請務必使用 2&1 來擷取 stdout和 stderr資訊。

```
"command": [
     "CMD-SHELL",
     "curl -f http://localhost/ >> /proc/1/fd/1 2>&1  || exit 1"
   ],
```

監控 Amazon ECS 容器執行個體運作狀態

Amazon ECS 提供容器執行個體運作狀態監控。您可以快速判斷 Amazon ECS 是否偵測到任何可能阻礙您的容器執行個體執行容器的問題。Amazon ECS 對每個具有代理程式版本 1.57.0 或更新版本的正在執行的容器執行個體，執行自動檢查以找出問題。有關驗證容器執行個體代理程式版本的詳細資訊，請參閱 更新 Amazon ECS 容器代理程式。

您必須使用 AWS CLI 版本 1.22.3或更新版本，或是 AWS CLI 版本 2.3.6 或更新版本。如需有關如何更新 的資訊 AWS CLI，請參閱《 使用者指南第 2 版》中的安裝或更新最新版本 AWS CLI的 。AWS Command Line Interface

若要檢視容器執行個體的運作狀態，describe-container-instances請使用 CONTAINER_INSTANCE_HEALTH選項執行 。

以下是 overallStatus 的有效值：

- OK
- IMPAIRED
- INSUFFICIENT_DATA
- INITIALIZING

以下是如何執行 describe-container-instances 的範例。

```
aws ecs describe-container-instances \
     --cluster cluster_name \
     --container-instances 47279cd2cadb41cbaef2dcEXAMPLE \
     --include CONTAINER_INSTANCE_HEALTH
```

下列為輸出中運作狀態物件的範例。

```
"healthStatus": {
 "overallStatus": "OK",
 "details": [{
  "type": "CONTAINER_RUNTIME",
  "status": "OK",
  "lastUpdated": "2021-11-10T03:30:26+00:00",
  "lastStatusChange": "2021-11-10T03:26:41+00:00"
 }]
}
```

容器執行個體運作狀態問題

當 以外的`overallStatus`任何狀態時`OK`，請嘗試下列動作：

- 等待，然後執行 `describe-container-instances`
- 在 EC2 主控台或使用 CLI 檢視您的容器執行個體運作狀態。
- 檢閱 CloudWatch 指標。如需詳細資訊，請參閱 使用 CloudWatch 監控 Amazon ECS
- 檢查 AWS Health Dashboard 以查看服務是否有任何問題。

使用應用程式追蹤資料識別 Amazon ECS 最佳化機會

Amazon ECS 與 AWS Distro for OpenTelemetry 整合，以從您的應用程式收集追蹤資料。Amazon ECS 使用 AWS Distro for OpenTelemetry 附屬容器來收集和路由追蹤資料 AWS X-Ray。如需詳細資訊，請參閱在 AWS Amazon ECS 中設定 Distro for OpenTelemetry Collector。然後，您可以使用 AWS X-Ray 來識別錯誤和例外狀況、分析效能瓶頸和回應時間。

若要讓 AWS Distro for OpenTelemetry Collector 將追蹤資料傳送至其中 AWS X-Ray，您的應用程式必須設定為建立追蹤資料。如需詳細資訊，請參閱《AWS X-Ray 開發人員指南》中的檢測您的 AWS X-Ray應用程式。

AWS Distro for OpenTelemetry 與 整合所需的 IAM 許可 AWS X-Ray

Amazon ECS 與 AWS Distro for OpenTelemetry 的整合需要您建立任務角色，並在任務定義中指定角色。建議您設定 AWS Distro for OpenTelemetry 附屬項目，將容器日誌路由至 CloudWatch Logs。

> **Important**
>
> 如果您也使用 AWS Distro for OpenTelemetry 整合收集應用程式指標，請確定您的任務 IAM 角色也包含該整合所需的許可。如需詳細資訊，請參閱使用應用程式指標關聯 Amazon ECS 應用程式效能。

建立下列政策，然後將其連接至任務執行角色。

若要使用 JSON 政策編輯器來建立政策

1. 登入 AWS Management Console ，並在 https：//https://console.aws.amazon.com/iam/ 開啟 IAM 主控台。
2. 在左側的導覽窗格中，選擇 Policies (政策)。

   如果這是您第一次選擇 Policies (政策)，將會顯示 Welcome to Managed Policies (歡迎使用受管政策) 頁面。選擇 Get Started (開始使用)。
3. 在頁面頂端，選擇 Create policy (建立政策)。
4. 在政策編輯器中，選擇 JSON 選項。
5. 輸入下列 JSON 政策文件：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "logs:PutLogEvents",
                "logs:CreateLogGroup",
                "logs:CreateLogStream",
                "logs:DescribeLogStreams",
                "logs:DescribeLogGroups",
                "logs:PutRetentionPolicy",
                "xray:PutTraceSegments",
                "xray:PutTelemetryRecords",
```

```
                "xray:GetSamplingRules",
                "xray:GetSamplingTargets",
                "xray:GetSamplingStatisticSummaries",
                "ssm:GetParameters"
            ],
            "Resource": "*"
        }
    ]
}
```

6. 選擇 Next (下一步)。

> **Note**
>
> 您可以隨時切換視覺化與 JSON 編輯器選項。不過，如果您進行變更或在視覺化編輯器中選擇下一步，IAM 就可能會調整您的政策結構，以便針對視覺化編輯器進行最佳化。如需詳細資訊，請參閱 IAM 使用者指南中的調整政策結構。

7. 在檢視與建立頁面上，為您正在建立的政策輸入政策名稱與描述 (選用)。檢視此政策中定義的許可，來查看您的政策所授予的許可。
8. 選擇 Create policy (建立政策) 儲存您的新政策。

指定 AWS Distro for OpenTelemetry 附屬裝置，以 AWS X-Ray 整合您的任務定義

Amazon ECS 主控台使用使用追蹤收集選項，簡化建立 AWS Distro for OpenTelemetry 附屬容器的程序。如需詳細資訊，請參閱使用主控台建立 Amazon ECS 任務定義。

如果您不是使用 Amazon ECS 主控台，您可以將 AWS Distro for OpenTelemetry 附屬容器新增至任務定義。下列任務定義程式碼片段顯示容器定義，用於新增 AWS Distro for OpenTelemetry 附屬工具以進行 AWS X-Ray 整合。

```
{
 "family": "otel-using-xray",
 "taskRoleArn": "arn:aws:iam::111122223333:role/AmazonECS_OpenTelemetryXrayRole",
 "executionRoleArn": "arn:aws:iam::111122223333:role/ecsTaskExecutionRole",
 "containerDefinitions": [{
   "name": "aws-otel-emitter",
   "image": "application-image",
   "logConfiguration": {
```

```
    "logDriver": "awslogs",
    "options": {
     "awslogs-create-group": "true",
     "awslogs-group": "/ecs/aws-otel-emitter",
     "awslogs-region": "us-east-1",
     "awslogs-stream-prefix": "ecs"
    }
   },
   "dependsOn": [{
    "containerName": "aws-otel-collector",
    "condition": "START"
   }]
  },
  {
   "name": "aws-otel-collector",
   "image": "public.ecr.aws/aws-observability/aws-otel-collector:v0.30.0",
   "essential": true,
   "command": [
    "--config=/etc/ecs/otel-instance-metrics-config.yaml"
   ],
   "logConfiguration": {
    "logDriver": "awslogs",
    "options": {
     "awslogs-create-group": "True",
     "awslogs-group": "/ecs/ecs-aws-otel-sidecar-collector",
     "awslogs-region": "us-east-1",
     "awslogs-stream-prefix": "ecs"
    }
   }
  }
 ],
 "networkMode": "awsvpc",
 "requiresCompatibilities": [
  "FARGATE"
 ],
 "cpu": "1024",
 "memory": "3072"
}
```

使用應用程式指標關聯 Amazon ECS 應用程式效能

Fargate 上的 Amazon ECS 支援從您在 Fargate 上執行的應用程式收集指標，並將指標匯出到 Amazon CloudWatch 或 Amazon Managed Service for Prometheus。

您可以使用收集的中繼資料，將應用程式效能資料與基礎基礎設施資料建立關聯，減少解決問題的平均時間。

Amazon ECS 使用 AWS Distro for OpenTelemetry 附屬容器來收集應用程式指標並將其路由至目的地。Amazon ECS 主控台體驗可簡化在建立任務定義時新增此整合的程序。

主題

將應用程式指標匯出至 Amazon CloudWatch

Fargate 上的 Amazon ECS 支援將您的自訂應用程式指標匯出到 Amazon CloudWatch 作為自訂指標。方法是將 AWS Distro for OpenTelemetry 附屬容器新增至您的任務定義。Amazon ECS 主控台會在建立新的任務定義時新增使用指標集合選項，以簡化此程序。如需詳細資訊，請參閱使用主控台建立 Amazon ECS 任務定義。

將應用程式指標匯出至有日誌群組名稱 `/aws/ecs/application/metrics` 的 CloudWatch Logs，並且您可以在 `ECS/AWSOTel/Application` 命名空間中查看指標。您的應用程式必須使用 OpenTelemetry SDK 進行檢測。如需詳細資訊，請參閱 AWS Distro for OpenTelemetry 文件中的 AWS Distro for OpenTelemetry 簡介。

考量事項

使用 Amazon ECS on Fargate 與 AWS Distro for OpenTelemetry 整合，將應用程式指標傳送至 Amazon CloudWatch 時，應考慮下列事項。

- 此整合只會將您的自訂應用程式指標傳送至 CloudWatch。如果需要任務層級指標，可以在 Amazon ECS 叢集組態中開啟 Container Insights。如需詳細資訊，請參閱使用具有增強可觀測性的 Container Insights 監控 Amazon ECS 容器。
- Amazon EC2 執行個體上託管於 Fargate 上的 AWS Amazon ECS 工作負載和託管於 Amazon EC2 執行個體上的 Amazon ECS 工作負載支援 Distro for OpenTelemetry 整合。目前不支援外部執行個體。
- CloudWatch 支援每個指標最多 30 個維度。預設情況下，Amazon ECS 預設為指標中包含 `TaskARN`、`ClusterARN`、`LaunchType`、`TaskDefinitionFamily`，以及 `TaskDefinitionRevision` 維度。其餘 25 個維度可由您的應用程式定義。如果設定的維度超過 30 個，則 CloudWatch 將無法顯示它們。發生這種情況時，應用程式指標會顯示在 `ECS/AWSOTel/Application` CloudWatch 指標命名空間，但沒有任何維度。您可以檢測應用程式以增加其他維

度。如需詳細資訊，請參閱 AWS Distro for OpenTelemetry 文件中的使用 CloudWatch 指標搭配 AWS Distro for OpenTelemetry。

AWS Distro for OpenTelemetry 與 Amazon CloudWatch 整合所需的 IAM 許可

Amazon ECS 與 AWS Distro for OpenTelemetry 的整合需要您建立任務 IAM 角色，並在任務定義中指定角色。我們建議您也將 AWS Distro for OpenTelemetry 附屬項目設定為將容器日誌路由至 CloudWatch Logs，該日誌也需要在任務定義中建立和指定任務執行 IAM 角色。Amazon ECS 主控台會代表您處理任務執行 IAM 角色，但任務 IAM 角色必須手動建立並新增至您的任務定義。如需有關任務執行 IAM 角色的詳細資訊，請參閱 Amazon ECS 任務執行 IAM 角色。

> **Important**
>
> 如果您也使用 AWS Distro for OpenTelemetry 整合收集應用程式追蹤資料，請確定您的任務 IAM 角色也包含該整合所需的許可。如需詳細資訊，請參閱使用應用程式追蹤資料識別 Amazon ECS 最佳化機會。
> 如果您的應用程式需要任何其他許可，則應將其新增到此政策中。每個任務定義只能指定一個任務 IAM 角色。例如，如果您使用儲存在 Systems Manager 中的自訂組態檔案，則應將 `ssm:GetParameters` 許可新增至此 IAM 政策。

建立 Elastic Container Service (IAM 主控台） 的服務角色

1. 登入 AWS Management Console ，並在 https：//https://console.aws.amazon.com/iam/ 開啟 IAM 主控台。
2. 在 IAM 主控台的導覽窗格中，選擇角色，然後選擇建立角色。
3. 對於 Trusted entity type (信任的實體類型)，請選擇 AWS 服務。
4. 針對服務或使用案例，選擇彈性容器服務，然後選擇彈性容器服務任務使用案例。
5. 選擇 Next (下一步)。
6. 在新增許可區段中，搜尋 AWSDistroOpenTelemetryPolicyForXray，然後選取政策。
7. (選用) 設定許可界限。這是進階功能，可用於服務角色，而不是服務連結的角色。

   a. 開啟設定許可界限區段，然後選擇使用許可界限來控制角色許可上限。

      IAM 包含您帳戶中 AWS 受管和客戶受管政策的清單。

   b. 選取用於許可界限的政策。

8. 選擇 Next (下一步)。
9. 輸入角色名稱或角色名稱尾碼，以協助您識別角色的目的。

> **Important**
>
> 當您命名角色時，請注意下列事項：
>
> - 角色名稱在您的 中必須是唯一的 AWS 帳戶，而且無法依大小寫設為唯一的。
>
> 例如，不要同時建立名為 **PRODROLE** 和 **prodrole** 的角色。當角色名稱用於政策或 ARN 的一部分時，角色名稱會區分大小寫，但是當角色名稱在主控台中顯示給客戶時，例如在登入過程中，角色名稱不會區分大小寫。
> - 因為其他實體可能會參考角色，所以在建立角色之後，就無法編輯其名稱。

10. (選用) 在說明中，輸入角色的說明。
11. (選用) 若要編輯使用案例和角色許可，請在步驟 1：選取受信任的實體或者步驟 2：新增許可區段中選擇編輯。
12. (選用) 若要協助識別、組織或搜尋角色，請將標籤新增為索引鍵值對。如需在 IAM 中使用標籤的詳細資訊，請參閱《IAM 使用者指南》中的AWS Identity and Access Management 資源的標籤。
13. 檢閱角色，然後選擇 Create role (建立角色)。

指定您任務定義中的 AWS Distro for OpenTelemetry 附屬

Amazon ECS 主控台使用使用指標收集選項，簡化建立 AWS Distro for OpenTelemetry 附屬容器的體驗。如需詳細資訊，請參閱使用主控台建立 Amazon ECS 任務定義。

如果您不是使用 Amazon ECS 主控台，則可以手動將 AWS Distro for OpenTelemetry 附屬容器新增至任務定義。下列任務定義範例顯示為 Amazon CloudWatch 整合新增 AWS Distro for OpenTelemetry 附屬項目的容器定義。

```
{
 "family": "otel-using-cloudwatch",
 "taskRoleArn": "arn:aws:iam::111122223333:role/AmazonECS_OpenTelemetryCloudWatchRole",
 "executionRoleArn": "arn:aws:iam::111122223333:role/ecsTaskExecutionRole",
 "containerDefinitions": [
    {
   "name": "aws-otel-emitter",
   "image": "application-image",
   "logConfiguration": {
```

```
    "logDriver": "awslogs",
    "options": {
     "awslogs-create-group": "true",
     "awslogs-group": "/ecs/aws-otel-emitter",
     "awslogs-region": "us-east-1",
     "awslogs-stream-prefix": "ecs"
    }
   },
   "dependsOn": [{
    "containerName": "aws-otel-collector",
    "condition": "START"
   }]
  },
  {
   "name": "aws-otel-collector",
   "image": "public.ecr.aws/aws-observability/aws-otel-collector:v0.30.0",
   "essential": true,
   "command": [
    "--config=/etc/ecs/ecs-cloudwatch.yaml"
   ],
   "logConfiguration": {
    "logDriver": "awslogs",
    "options": {
     "awslogs-create-group": "True",
     "awslogs-group": "/ecs/ecs-aws-otel-sidecar-collector",
     "awslogs-region": "us-east-1",
     "awslogs-stream-prefix": "ecs"
    }
   }
  }
 ],
 "networkMode": "awsvpc",
 "requiresCompatibilities": [
  "FARGATE"
 ],
 "cpu": "1024",
 "memory": "3072"
}
```

將應用程式指標匯出至 Amazon Managed Service for Prometheus

Amazon ECS 支援將您的任務層級 CPU、記憶體、網路和儲存指標，以及自訂應用程式指標匯出至 Amazon Managed Service for Prometheus。方法是將 AWS Distro for OpenTelemetry 附屬容器新增

至您的任務定義。Amazon ECS 主控台會在建立新的任務定義時新增使用指標集合選項，以簡化此程序。如需詳細資訊，請參閱使用主控台建立 Amazon ECS 任務定義。

這些指標將匯出至 Amazon Managed Service for Prometheus，並可以使用 Amazon Managed Grafana 儀表板查看。您必須使用 Prometheus 程式庫或 OpenTelemetry SDK 檢測應用程式。如需使用 OpenTelemetry SDK 檢測應用程式的詳細資訊，請參閱 AWS Distro for OpenTelemetry 文件中的 AWS Distro for OpenTelemetry 簡介。

使用 Prometheus 程式庫時，您的應用程式必須公開 `/metrics` 端點，用於抓取指標資料。如需有關使用 Prometheus 程式庫檢測應用程式的詳細資訊，請參閱 Prometheus 文件中的 Prometheus 用戶端程式庫。

考量事項

使用 Amazon ECS on Fargate 與 AWS Distro for OpenTelemetry 整合，將應用程式指標傳送至 Amazon Managed Service for Prometheus 時，應考慮下列事項。

- Amazon EC2 執行個體上託管於 Fargate 上的 AWS Amazon ECS 工作負載和託管於 Amazon EC2 執行個體上的 Amazon ECS 工作負載支援 Distro for OpenTelemetry 整合。目前不支援外部執行個體。
- 根據預設， AWS Distro for OpenTelemetry 會在匯出至 Amazon Managed Service for Prometheus 時包含應用程式指標的所有可用任務層級維度。您還可以檢測應用程式以增加其他維度。如需詳細資訊，請參閱 Distro for OpenTelemetry 文件中的適用於 Amazon Managed Service for Prometheus 的 Prometheus 遠端寫入匯出器入門。 AWS

AWS Distro for OpenTelemetry 與 Amazon Managed Service for Prometheus 整合所需的 IAM 許可

使用 AWS Distro for OpenTelemetry 附屬工具的 Amazon ECS 與 Amazon Managed Service for Prometheus 整合需要您建立任務 IAM 角色，並在任務定義中指定角色。註冊任務定義之前，必須使用以下步驟手動建立此任務 IAM 角色。

我們建議您也將 AWS Distro for OpenTelemetry 附屬項目設定為將容器日誌路由至 CloudWatch Logs，該日誌也需要在任務定義中建立和指定任務執行 IAM 角色。Amazon ECS 主控台會代表您處理任務執行 IAM 角色，但任務 IAM 角色必須手動建立。如需有關為任務建立 IAM 角色的詳細資訊，請參閱 Amazon ECS 任務執行 IAM 角色。

> **Important**
>
> 如果您也使用 AWS Distro for OpenTelemetry 整合收集應用程式追蹤資料，請確定您的任務 IAM 角色也包含該整合所需的許可。如需詳細資訊，請參閱使用應用程式追蹤資料識別 Amazon ECS 最佳化機會。

建立 Elastic Container Service (IAM 主控台） 的服務角色

1. 登入 AWS Management Console ，並在 https：//https://console.aws.amazon.com/iam/ 開啟 IAM 主控台。
2. 在 IAM 主控台的導覽窗格中，選擇角色，然後選擇建立角色。
3. 對於 Trusted entity type (信任的實體類型)，請選擇 AWS 服務。
4. 針對服務或使用案例，選擇彈性容器服務，然後選擇彈性容器服務任務使用案例。
5. 選擇 Next (下一步)。
6. 在新增許可區段中，搜尋 AmazonPrometheusRemoteWriteAccess，然後選取政策。
7. (選用) 設定許可界限。這是進階功能，可用於服務角色，而不是服務連結的角色。

   a. 開啟設定許可界限區段，然後選擇使用許可界限來控制角色許可上限。

      IAM 包含您帳戶中 AWS 受管和客戶受管政策的清單。

   b. 選取用於許可界限的政策。
8. 選擇 Next (下一步)。
9. 輸入角色名稱或角色名稱尾碼，以協助您識別角色的目的。

   > **Important**
   >
   > 當您命名角色時，請注意下列事項：
   >
   > - 角色名稱在您的 中必須是唯一的 AWS 帳戶，而且無法依大小寫設為唯一的。
   >
   >   例如，不要同時建立名為 **PRODROLE** 和 **prodrole** 的角色。當角色名稱用於政策或 ARN 的一部分時，角色名稱會區分大小寫，但是當角色名稱在主控台中顯示給客戶時，例如在登入過程中，角色名稱不會區分大小寫。
   > - 因為其他實體可能會參考角色，所以在建立角色之後，就無法編輯其名稱。
10. (選用) 在說明中，輸入角色的說明。

11. (選用) 若要編輯使用案例和角色許可，請在步驟 1：選取受信任的實體或者步驟 2：新增許可區段中選擇編輯。
12. (選用) 若要協助識別、組織或搜尋角色，請將標籤新增為索引鍵值對。如需在 IAM 中使用標籤的詳細資訊，請參閱《IAM 使用者指南》中的AWS Identity and Access Management 資源的標籤。
13. 檢閱角色，然後選擇 Create role (建立角色)。

指定您任務定義中的 AWS Distro for OpenTelemetry 附屬

Amazon ECS 主控台使用使用指標收集選項，簡化建立 AWS Distro for OpenTelemetry 附屬容器的體驗。如需詳細資訊，請參閱使用主控台建立 Amazon ECS 任務定義。

如果您不是使用 Amazon ECS 主控台，則可以手動將 AWS Distro for OpenTelemetry 附屬容器新增至任務定義。下列任務定義範例顯示為 Amazon Managed Service for Prometheus 整合新增 AWS Distro for OpenTelemetry 附屬項目的容器定義。

```
{
 "family": "otel-using-cloudwatch",
 "taskRoleArn": "arn:aws:iam::111122223333:role/AmazonECS_OpenTelemetryCloudWatchRole",
 "executionRoleArn": "arn:aws:iam::111122223333:role/ecsTaskExecutionRole",
 "containerDefinitions": [{
   "name": "aws-otel-emitter",
   "image": "application-image",
   "logConfiguration": {
    "logDriver": "awslogs",
    "options": {
     "awslogs-create-group": "true",
     "awslogs-group": "/ecs/aws-otel-emitter",
     "awslogs-region": "aws-region",
     "awslogs-stream-prefix": "ecs"
    }
   },
   "dependsOn": [{
    "containerName": "aws-otel-collector",
    "condition": "START"
   }]
  },
  {
   "name": "aws-otel-collector",
   "image": "public.ecr.aws/aws-observability/aws-otel-collector:v0.30.0",
   "essential": true,
   "command": [
```

```
     "--config=/etc/ecs/ecs-amp.yaml"
    ],
    "environment": [{
     "name": "AWS_PROMETHEUS_ENDPOINT",
     "value": "https://aps-workspaces.aws-region.amazonaws.com/workspaces/
ws-a1b2c3d4-5678-90ab-cdef-EXAMPLE11111/api/v1/remote_write"
    }],
    "logConfiguration": {
     "logDriver": "awslogs",
     "options": {
      "awslogs-create-group": "True",
      "awslogs-group": "/ecs/ecs-aws-otel-sidecar-collector",
      "awslogs-region": "aws-region",
      "awslogs-stream-prefix": "ecs"
     }
    }
   }
  ],
  "networkMode": "awsvpc",
  "requiresCompatibilities": [
   "FARGATE"
  ],
  "cpu": "1024",
  "memory": "3072"
}
```

使用 記錄 Amazon ECS API 呼叫 AWS CloudTrail

Amazon Elastic Container Service 已與 整合AWS CloudTrail，此服務可提供使用者、角色或 所採取動作的記錄 AWS 服務。CloudTrail 會將 Amazon ECS 的所有 API 呼叫擷取為事件。擷取的呼叫包括從 Amazon ECS 主控台的呼叫，以及對 Amazon ECS API 操作的程式碼呼叫。使用 CloudTrail 收集的資訊，您可以判斷對 Amazon ECS 提出的請求、提出請求的 IP 地址、提出請求的時間，以及其他詳細資訊。

每一筆事件或日誌專案都會包含產生請求者的資訊。身分資訊可協助您判斷下列事項：

- 該請求是使用根使用者還是使用者憑證提出。
- 請求是否代表 IAM Identity Center 使用者提出。
- 提出該請求時，是否使用了特定角色或聯合身分使用者的暫時安全憑證。
- 該請求是否由另一項 AWS 服務服務提出。

當您建立帳戶 AWS 帳戶 時CloudTrail 會在 中處於作用中狀態，而且您會自動存取 CloudTrail 事件歷史記錄。CloudTrail 事件歷史記錄提供過去 90 天發生的已記錄 AWS 區域管理事件的可檢視、可搜尋、可下載而且不可變的記錄。如需詳細資訊，請參閱「AWS CloudTrail 使用者指南」中的使用 CloudTrail 事件歷史記錄。檢視事件歷史記錄不會產生 CloudTrail 費用。

如需 AWS 帳戶 過去 90 天內持續記錄的事件，請建立線索或 CloudTrail Lake 事件資料存放區。

CloudTrail 追蹤

線索能讓 CloudTrail 將日誌檔案交付至 Amazon S3 儲存貯體。使用 建立的所有線索 AWS Management Console 都是多區域。您可以使用 AWS CLI建立單一或多區域追蹤。建議您建立多區域追蹤，因為您擷取 AWS 區域 帳戶中所有 的活動。如果您建立單一區域追蹤，您只能檢視追蹤 AWS 區域中所記錄的事件。如需追蹤的詳細資訊，請參閱「AWS CloudTrail 使用者指南」中的為您的 AWS 帳戶建立追蹤和為組織建立追蹤。

您可以透過建立追蹤，免費將持續管理事件的一個複本從 CloudTrail 傳遞至您的 Amazon S3 儲存貯體，但這樣做會產生 Amazon S3 儲存費用。如需 CloudTrail 定價的詳細資訊，請參閱 AWS CloudTrail 定價。如需 Amazon S3 定價的相關資訊，請參閱 Amazon S3 定價。

CloudTrail Lake 事件資料存放區

CloudTrail Lake 讓您能夠對事件執行 SQL 型查詢。CloudTrail Lake 會將分列式 JSON 格式的現有事件轉換為 Apache ORC 格式。ORC 是一種單欄式儲存格式，針對快速擷取資料進行了最佳化。系統會將事件彙總到事件資料存放區中，事件資料存放區是事件的不可變集合，其依據為您透過套用進階事件選取器選取的條件。套用於事件資料存放區的選取器控制哪些事件持續存在並可供您查詢。如需 CloudTrail Lake 的詳細資訊，請參閱AWS CloudTrail 《 使用者指南》中的使用 AWS CloudTrail Lake。

CloudTrail Lake 事件資料存放區和查詢會產生費用。建立事件資料存放區時，您可以選擇要用於事件資料存放區的定價選項。此定價選項將決定擷取和儲存事件的成本，以及事件資料存放區的預設和最長保留期。如需 CloudTrail 定價的詳細資訊，請參閱 AWS CloudTrail 定價。

CloudTrail 中的 Amazon ECS 管理事件

管理事件提供在 資源上執行的管理操作的相關資訊 AWS 帳戶。這些也稱為控制平面操作。根據預設，CloudTrail 記錄管理事件。

Amazon Elastic Container Service 會將所有 Amazon ECS 控制平面操作記錄為管理事件。例如，對 `CreateService`、`RunTask` 和 `DeleteCluster` 區段的呼叫，會在 CloudTrail 日誌檔案中產生項

目。如需 Amazon ECS 記錄到 CloudTrail 的 Amazon Elastic Container Service 控制平面操作清單，請參閱 Amazon Elastic Container Service API 參考。

Amazon ECS 事件範例

一個事件代表任何來源提出的單一請求，並包含請求 API 操作的相關資訊、操作的日期和時間、請求參數等。CloudTrail 日誌檔並非依公有 API 呼叫的堆疊追蹤排序，因此事件不會以任何特定順序出現。

下列範例顯示示範 `CreateService`動作的 CloudTrail 事件。

> **Note**
>
> 此範例已格式化，以提升可讀性。在 CloudTrail 日誌檔案中，所有項目和事件會合併為單一列。此外，這個範例中受限於單一 Amazon ECS 項目。在真正的 CloudTrail 日誌檔案中，您會看到來自多個 AWS 服務的項目和事件。


```
{
    "eventVersion": "1.04",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "AIDACKCEVSQ6C2EXAMPLE:account_name",
        "arn": "arn:aws:sts::123456789012:user/Mary_Major",
        "accountId": "123456789012",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "sessionContext": {
            "attributes": {
                "mfaAuthenticated": "false",
                "creationDate": "2018-06-20T18:32:25Z"
            },
            "sessionIssuer": {
                "type": "Role",
                "principalId": "AIDACKCEVSQ6C2EXAMPLE",
                "arn": "arn:aws:iam::123456789012:role/Admin",
                "accountId": "123456789012",
                "userName": "Mary_Major"
            }
        }
    },
    "eventTime": "2018-06-20T19:04:36Z",
    "eventSource": "ecs.amazonaws.com",
    "eventName": "CreateCluster",
```

```
    "awsRegion": "us-east-1",
    "sourceIPAddress": "203.0.113.12",
    "userAgent": "console.amazonaws.com",
    "requestParameters": {
        "clusterName": "default"
    },
    "responseElements": {
        "cluster": {
            "clusterArn": "arn:aws:ecs:us-east-1:123456789012:cluster/default",
            "pendingTasksCount": 0,
            "registeredContainerInstancesCount": 0,
            "status": "ACTIVE",
            "runningTasksCount": 0,
            "statistics": [],
            "clusterName": "default",
            "activeServicesCount": 0
        }
    },
    "requestID": "cb8c167e-EXAMPLE",
    "eventID": "e3c6f4ce-EXAMPLE",
    "eventType": "AwsApiCall",
    "recipientAccountId": "123456789012"
}
```


如需有關 CloudTrail 記錄內容的詳細資訊，請參閱「AWS CloudTrail 使用者指南」中的CloudTrail 記錄內容。

使用 Amazon ECS 中繼資料監控工作負載

您可以使用任務和容器中繼資料來疑難排解工作負載，並根據執行期環境進行組態變更。

中繼資料包括下列類別：

- 任務層級屬性，提供任務執行位置的相關資訊。
- 提供 Docker ID、名稱和映像詳細資訊的容器層級屬性。

  這可提供容器的可見性。
- IP 地址、子網路和網路模式等網路設定。

  這有助於網路組態和故障診斷。

- 任務狀態和運作狀態

  這可讓您知道任務是否正在執行。

您可以透過下列任一方法檢視中繼資料：

- 容器中繼資料檔案

  從 Amazon ECS 容器代理程式 1.15.0 版開始，容器或主機容器執行個體中提供各種容器中繼資料。透過啟用此功能，您可從容器或主機容器執行個體內查詢關於任務、容器和容器執行個體的資訊。中繼資料檔案建立於主機執行個體，並在容器中掛載為 Docker 磁碟區，因此當任務託管於 AWS Fargate 時不可用。
- 任務中繼資料端點

  Amazon ECS 容器代理程式會將環境變數插入到每個容器中，稱為任務中繼資料端點，它為容器提供各種任務中繼資料和 Docker 統計資訊。
- 容器簡介

  Amazon ECS 容器代理程式提供一種 API 操作，它收集執行代理程式的容器執行個體的詳細資訊，以及在該執行個體上執行之關聯任務的詳細資訊。

Amazon ECS 容器中繼資料檔案

從 Amazon ECS 容器代理程式 1.15.0 版開始，容器或主機容器執行個體中提供各種容器中繼資料。透過啟用此功能，您可從容器或主機容器執行個體內查詢關於任務、容器和容器執行個體的資訊。中繼資料檔案是在主機執行個體上建立，並以 Docker 磁碟區的形式掛載在容器中，因此在 AWS Fargate 上託管任務時無法使用。

清除容器時，會清除主機執行個體上的容器中繼資料檔案。您可以使用 `ECS_ENGINE_TASK_CLEANUP_WAIT_DURATION` 容器代理變數，來調整執行此動作的時間。如需詳細資訊，請參閱自動 Amazon ECS 任務和映像清除。

主題

- 容器中繼資料檔案位置
- 開啟 Amazon ECS 容器中繼資料
- Amazon ECS 容器中繼資料檔案格式

容器中繼資料檔案位置

容器中繼資料檔案預設會寫入下列主機和容器路徑。

- 對於 Linux 執行個體：
  - 主機路徑：`/var/lib/ecs/data/metadata/cluster_name/task_id/container_name/ecs-container-metadata.json`

    > **Note**
    >
    > Linux 主機路徑假設在啟動代理時，會使用預設資料目錄掛載路徑 (`/var/lib/ecs/data`)。如果您並非使用 Amazon ECS 最佳化 AMI (或 `ecs-init` 套件來啟動和維護容器代理程式)，請務必將 `ECS_HOST_DATA_DIR` 代理程式組態變數設定為容器代理程式狀態檔案所在的主機路徑。如需詳細資訊，請參閱Amazon ECS 容器代理程式組態。

  - 容器路徑：`/opt/ecs/metadata/random_ID/ecs-container-metadata.json`
- 對於 Windows 執行個體：
  - 主機路徑：`C:\ProgramData\Amazon\ECS\data\metadata\task_id\container_name\ecs-container-metadata.json`
  - 容器路徑：`C:\ProgramData\Amazon\ECS\metadata\random_ID\ecs-container-metadata.json`

不過，為方便存取，容器中繼資料檔案位置設定為容器內部的 `ECS_CONTAINER_METADATA_FILE` 環境變數。您可以使用下列命令在容器內部讀取檔案內容：

- 對於 Linux 執行個體：

```
cat $ECS_CONTAINER_METADATA_FILE
```

- 對於 Windows 執行個體 (PowerShell)：

```
Get-Content -path $env:ECS_CONTAINER_METADATA_FILE
```

開啟 Amazon ECS 容器中繼資料

您可以將 `ECS_ENABLE_CONTAINER_METADATA` 容器代理變數設定為 `true`，以在容器執行個體層級開啟容器中繼資料。您可以在 `/etc/ecs/ecs.config` 組態檔案中設定此變數，並重新啟動代理。

啟動代理容器時，您也可以在執行時間將它設定為 Docker 環境變數。如需詳細資訊，請參閱Amazon ECS 容器代理程式組態。

如果 ECS_ENABLE_CONTAINER_METADATA 在代理程式啟動時設定為 true，則系統會為該時間點後建立的任何容器建立中繼資料檔案。對於在 ECS_ENABLE_CONTAINER_METADATA 容器代理程式變數設定為 true 之前所建立的容器，Amazon ECS 容器代理程式無法為其建立中繼資料檔案。為了確保所有容器都收到中繼資料檔案，您應該在容器執行個體啟動時設定此代理變數。以下是範例使用者資料指令碼，可設定此變數以及使用叢集來註冊容器執行個體。

```
#!/bin/bash
cat <<'EOF' >> /etc/ecs/ecs.config
ECS_CLUSTER=your_cluster_name
ECS_ENABLE_CONTAINER_METADATA=true
EOF
```

Amazon ECS 容器中繼資料檔案格式

下列資訊存放在容器中繼資料 JSON 檔案中。

`Cluster`

容器任務執行所在叢集的名稱。

`ContainerInstanceARN`

主機容器執行個體的完整 Amazon Resource Name (ARN)。

`TaskARN`

容器所屬任務的完整 Amazon Resource Name (ARN)。

`TaskDefinitionFamily`

容器正在使用的任務定義系列名稱。

`TaskDefinitionRevision`

容器正在使用的任務定義修訂版。

`ContainerID`

容器的 Docker 容器 ID (而不是 Amazon ECS 容器 ID)。

`ContainerName`

容器之 Amazon ECS 任務定義中的容器名稱。

`DockerContainerName`

Docker 常駐程式用於容器的容器名稱 (例如，顯示於 docker ps 命令輸出中的名稱)。

`ImageID`

用來啟動容器之 Docker 映像的 SHA 摘要。

`ImageName`

用來啟動容器之 Docker 映像的映像名稱和標籤。

`PortMappings`

與容器相關聯的任何連接埠映射。

`ContainerPort`

容器上公開的連接埠。

`HostPort`

主機容器執行個體上公開的連接埠。

`BindIp`

Docker 指派給容器的連結 IP 地址。此 IP 地址只能使用 `bridge` 網路模式予以套用，而且只能從容器執行個體存取。

`Protocol`

用於連接埠映射的網路協定。

`Networks`

容器的網路模式和 IP 地址。

`NetworkMode`

容器所屬任務的網路模式。

`IPv4Addresses`

與容器相關聯的 IP 地址。

> **Important**
>
> 如果您的任務使用的是 `awsvpc` 網路模式，就不會傳回容器的 IP 地址。在這種情況下，您可以使用下列命令讀取 /etc/hosts 檔案來擷取 IP 地址：

```
tail -1 /etc/hosts | awk '{print $1}'
```

MetadataFileStatus

中繼資料檔案的狀態。狀態為 READY 時，中繼資料檔案是最新且完整的。如果檔案尚未就緒 (例如，啟動任務時)，則會提供截斷版本的檔案格式。為了避免已啟動容器但尚未寫入中繼資料的可能競爭條件，您可以在依存中繼資料之前剖析中繼資料檔案，並等待這個參數設定為 READY。這通常可在容器啟動的 1 秒內可用。

AvailabilityZone

主機容器執行個體所在的可用區域。

HostPrivateIPv4Address

容器所屬任務的私有 IP 地址。

HostPublicIPv4Address

容器所屬任務的公有 IP 地址。

Example Amazon ECS 容器中繼資料檔案 (**READY**)

下列範例示範狀態為 READY 的容器中繼資料檔案。

```
{
    "Cluster": "default",
    "ContainerInstanceARN": "arn:aws:ecs:us-west-2:012345678910:container-instance/
default/1f73d099-b914-411c-a9ff-81633b7741dd",
    "TaskARN": "arn:aws:ecs:us-west-2:012345678910:task/default/2b88376d-
aba3-4950-9ddf-bcb0f388a40c",
    "TaskDefinitionFamily": "console-sample-app-static",
    "TaskDefinitionRevision": "1",
    "ContainerID": "aec2557997f4eed9b280c2efd7afccdcedfda4ac399f7480cae870cfc7e163fd",
    "ContainerName": "simple-app",
    "CreatedAt": "2023-10-08T20:09:11.44527186Z",
    "StartedAt": "2023-10-08T20:09:11.44527186Z",
    "DockerContainerName": "/ecs-console-sample-app-static-1-simple-app-
e4e8e495e8baa5de1a00",
    "ImageID":
 "sha256:2ae34abc2ed0a22e280d17e13f9c01aaf725688b09b7a1525d1a2750e2c0d1de",
```

```
    "ImageName": "httpd:2.4",
    "PortMappings": [
        {
            "ContainerPort": 80,
            "HostPort": 80,
            "BindIp": "0.0.0.0",
            "Protocol": "tcp"
        }
    ],
    "Networks": [
        {
            "NetworkMode": "bridge",
            "IPv4Addresses": ["192.0.2.0"]
        }
    ],
    "MetadataFileStatus": "READY",
    "AvailabilityZone": "us-east-1b",
    "HostPrivateIPv4Address": "192.0.2.0",
    "HostPublicIPv4Address": "203.0.113.0"
}
```

Example 未完成的 Amazon ECS 容器中繼資料檔案 (尚未 **READY**)

下列範例示範尚未達到 READY 狀態的容器中繼資料檔案。檔案中的資訊僅限於任務定義中已知的數個參數。容器中繼資料檔案應該會在容器啟動的 1 秒內就緒。

```
{
    "Cluster": "default",
    "ContainerInstanceARN": "arn:aws:ecs:us-west-2:012345678910:container-instance/
default/1f73d099-b914-411c-a9ff-81633b7741dd",
    "TaskARN": "arn:aws:ecs:us-west-2:012345678910:task/default/
d90675f8-1a98-444b-805b-3d9cabb6fcd4",
    "ContainerName": "metadata"
}
```

EC2 上 Amazon ECS 任務可用的任務中繼資料

Amazon ECS 容器代理程式提供方法來擷取各種任務中繼資料和 Docker 統計資訊。這稱為任務中繼資料端點。以下為可用的版本：

- 任務中繼資料端點版本 4 — 為容器提供各種中繼資料和 Docker 統計資訊。也提供網路速率資料。適用於在執行 Amazon ECS 容器代理程式至少 `1.39.0` 版本的 Amazon EC2 Linux 執行個體上啟動

的 Amazon ECS 任務。對於使用 awsvpc 網路模式的 Amazon EC2 Windows 執行個體，Amazon ECS 容器代理程式必須至少為 1.54.0 版。如需詳細資訊，請參閱Amazon ECS 任務中繼資料端點第 4 版。

- 任務中繼資料端點版本 3 — 為容器提供各種中繼資料和 Docker 統計資訊。適用於在執行 Amazon ECS 容器代理程式至少 1.21.0 版本的 Amazon EC2 Linux 執行個體上啟動的 Amazon ECS 任務。對於使用 awsvpc 網路模式的 Amazon EC2 Windows 執行個體，Amazon ECS 容器代理程式必須至少為 1.54.0 版。如需詳細資訊，請參閱Amazon ECS 任務中繼資料端點第 3 版。
- 任務中繼資料端點版本 2 — 適用於在執行 Amazon ECS 容器代理程式至少 1.17.0 版本的 Amazon EC2 Linux 執行個體上啟動的 Amazon ECS 任務。對於使用 awsvpc 網路模式的 Amazon EC2 Windows 執行個體，Amazon ECS 容器代理程式必須至少為 1.54.0 版。如需詳細資訊，請參閱Amazon ECS 任務中繼資料端點第 2 版。

如果在 Amazon EC2 上託管您的 Amazon ECS 任務，您也可以使用執行個體中繼資料服務 (IMDS) 端點存取任務主機中繼資料。當從託管任務的執行個體內執行時，下列命令會列出主機執行個體的 ID。

```
curl http://169.254.169.254/latest/meta-data/instance-id
```

您可以從端點取得的資訊分為幾個類別，例如 *instance-id*。如需有關您可以使用端點取得之不同類別主機執行個體中繼資料的詳細資訊，請參閱執行個體中繼資料類別 。

Amazon ECS 任務中繼資料端點第 4 版

Amazon ECS 容器代理程式會將環境變數插入到每個容器中，稱為任務中繼資料端點，它為容器提供各種任務中繼資料和 Docker 統計資訊。

任務中繼資料和網路速率統計資訊會傳送至 CloudWatch Container Insights ，並可在 AWS Management Console中檢視。如需詳細資訊，請參閱使用具有增強可觀測性的 Container Insights 監控 Amazon ECS 容器。

> **Note**
>
> Amazon ECS 提供任務中繼資料端點的較早版本。為了避免未來需要建立新的任務中繼資料端點版本，可以將其他中繼資料新增至第 4 版輸出。我們不會移除任何現有中繼資料或變更中繼資料欄位名稱。

環境變數會預設插入至 Amazon ECS 任務的容器中，這些任務在執行 Amazon ECS 容器代理程式至少 1.39.0 版的 Amazon EC2 Linux 執行個體上啟動。對於使用 awsvpc 網路模式的 Amazon

EC2 Windows 執行個體，Amazon ECS 容器代理程式必須至少為 1.54.0 版。如需詳細資訊，請參閱Amazon ECS Linux 容器執行個體管理。

> **Note**
>
> 使用較舊版本的 Amazon ECS 容器代理程式可在 Amazon EC2 執行個體上新增此功能的支援，方法是將代理程式更新為最新版本。如需詳細資訊，請參閱更新 Amazon ECS 容器代理程式。

任務中繼資料端點版本 4 路徑

下列任務中繼資料端點路徑可供容器使用。

`${ECS_CONTAINER_METADATA_URI_V4}`

此路徑傳回容器的中繼資料。

`${ECS_CONTAINER_METADATA_URI_V4}/task`

此路徑傳回任務的中繼資料，包括與任務相關聯之所有容器的容器 ID 和名稱清單。如需此端點之回應的詳細資訊，請參閱「Amazon ECS 任務中繼資料 V4 JSON 回應」。

`${ECS_CONTAINER_METADATA_URI_V4}/taskWithTags`

除了可使用 `ListTagsForResource` API 擷取的任務和容器執行個體標籤之外，此路徑還會傳回包含在 `/task` 端點內的任務的中繼資料。擷取標籤中繼資料時收到的任何錯誤都會包含在回應的 `Errors` 欄位中。

> **Note**
>
> `Errors` 欄位僅位於執行容器代理程式至少 1.50.0 版本的 Amazon EC2 Linux 執行個體中託管的任務回應中。對於使用 `awsvpc` 網路模式的 Amazon EC2 Windows 執行個體，Amazon ECS 容器代理程式必須至少為 1.54.0 版。
> 此端點需要 `ecs.ListTagsForResource` 許可。

`${ECS_CONTAINER_METADATA_URI_V4}/stats`

此路徑傳回特定容器的 Docker 統計資訊。如需每個所傳回統計資訊的詳細資訊，請參閱 Docker API 文件中的 ContainerStats。

對於使用 awsvpc 或 bridge 網路模式、託管於執行容器代理程式至少 1.43.0 版本的 Amazon EC2 Linux 執行個體中的 Amazon ECS 任務，回應中會包含額外的網路速率統計資訊。對於所有其他任務，回應只會包含累積的網路統計資訊。

`${ECS_CONTAINER_METADATA_URI_V4}/task/stats`

此路徑傳回與任務相關聯之所有容器的 Docker 統計資訊。附屬容器可使用該資訊來擷取網路指標。如需每個所傳回統計資訊的詳細資訊，請參閱 Docker API 文件中的 ContainerStats。

對於使用 awsvpc 或 bridge 網路模式、託管於執行容器代理程式至少 1.43.0 版本的 Amazon EC2 Linux 執行個體中的 Amazon ECS 任務，回應中會包含額外的網路速率統計資訊。對於所有其他任務，回應只會包含累積的網路統計資訊。

Amazon ECS 任務中繼資料 V4 JSON 回應

任務中繼資料端點 (`${ECS_CONTAINER_METADATA_URI_V4}/task`) JSON 回應會傳回下列資訊。這包括除了任務內每個容器的中繼資料之外，與任務相關聯的中繼資料。

`Cluster`

任務所屬 Amazon ECS 叢集的 Amazon Resource Name (ARN) 或簡短名稱。

`ServiceName`

任務所屬的服務名稱。如果任務與服務關聯，Amazon EC2 和 Amazon ECS Anywhere 容器執行個體都會顯示 ServiceName。

> **Note**
>
> 只有在使用 Amazon ECS 容器代理程式版本 1.63.1 或更新版本時，才包括該 ServiceName 中繼資料。

`VPCID`

Amazon EC2 容器執行個體的 VPC ID。此欄位僅適用於 Amazon EC2 執行個體。

> **Note**
>
> 只有在使用 Amazon ECS 容器代理程式版本 1.63.1 或更新版本時，才包括該 VPCID 中繼資料。

`TaskARN`

容器所屬任務的完整 Amazon Resource Name (ARN)。

`Family`

任務的 Amazon ECS 任務定義系列。

`Revision`

任務的 Amazon ECS 任務定義修訂。

`DesiredStatus`

Amazon ECS 中任務的所需狀態。

`KnownStatus`

Amazon ECS 中任務的已知狀態。

`Limits`

在任務層級指定的資源限制，例如 CPU (以 vCPU 表示) 和記憶體。如果未定義資源限制，則會省略此參數。

`PullStartedAt`

第一個容器映像提取的開始時間戳記。

`PullStoppedAt`

最後一個容器映像提取的完成時間戳記。

`AvailabilityZone`

任務所在的可用區域。

> **Note**
>
> 可用區域中繼資料僅適用於使用平台第 1.4 版或更新版本 (Linux) 或 1.0.0 (Windows) 的 Fargate 任務。

`LaunchType`

任務使用的啟動類型。使用叢集容量提供者時，這會指出任務是使用 Fargate 還是 EC2 基礎設施。

> **Note**
>
> 只有在使用 Amazon ECS Linux 容器代理程式版本 1.45.0 或更新版本 (Linux) 或 1.0.0 或更新版本 (Windows) 時，才包括此 LaunchType 中繼資料。

`Containers`

與任務相關聯之每個容器的容器中繼資料清單。

`DockerId`

容器的 Docker ID。

當您使用 Fargate 時，ID 為 32 位十六進制，後跟 10 位數字。

`Name`

任務定義中指定的容器名稱。

`DockerName`

提供給 Docker 的容器名稱。Amazon ECS 容器代理程式會產生容器的唯一名稱，以避免在單一執行個體上執行相同任務定義的多個複本時，發生名稱衝突。

`Image`

容器的映像。

`ImageID`

映像的 SHA-256 摘要。

`Ports`

向容器開放的任何連接埠。如果未開放連接埠，則會省略此參數。

`Labels`

任何套用至容器的標籤。如果未套用標籤，則會省略此參數。

`DesiredStatus`

Amazon ECS 中容器的所需狀態。

`KnownStatus`

Amazon ECS 中容器的已知狀態。

`ExitCode`

容器的結束代碼。如果容器尚未結束，則會省略此參數。

`Limits`

在容器層級指定的資源限制，例如 CPU (以 CPU 單位表示) 和記憶體。如果未定義資源限制，則會省略此參數。

`CreatedAt`

容器的建立時間戳記。如果尚未建立容器，則會省略此參數。

`StartedAt`

容器的啟動時間戳記。如果尚未啟動容器，則會省略此參數。

`FinishedAt`

容器的停止時間戳記。如果尚未停止容器，則會省略此參數。

`Type`

容器的類型。任務定義中指定的容器類型為 `NORMAL`。您可以忽略其他容器類型，這些是 Amazon ECS 容器代理程式用來佈建內部任務資源的容器類型。

`LogDriver`

容器正在使用的日誌驅動程式。

> **Note**
>
> 只有在使用 Amazon ECS Linux 容器代理程式版本 `1.45.0` 或更新版本時，才包括此 `LogDriver` 中繼資料。

`LogOptions`

為容器定義的日誌驅動程式選項。

> **Note**
>
> 只有在使用 Amazon ECS Linux 容器代理程式版本 `1.45.0` 或更新版本時，才包括此 `LogOptions` 中繼資料。

`ContainerARN`

容器的完整 Amazon Resource Name (ARN)。

> **Note**
>
> 只有在使用 Amazon ECS Linux 容器代理程式版本 `1.45.0` 或更新版本時，才包括此 `ContainerARN` 中繼資料。

`Networks`

容器的網路資訊，例如網路模式和 IP 地址。如果未定義網路資訊，則會省略此參數。

`RestartCount`

容器重新啟動的次數。

> **Note**
>
> 只有在容器啟用重新啟動政策時，才會包含`RestartCount`中繼資料。如需詳細資訊，請參閱使用容器重新啟動政策重新啟動 Amazon ECS 任務中的個別容器。

`ExecutionStoppedAt`

任務 `DesiredStatus` 移至 `STOPPED` 時的時間戳記。這會在基本容器移至 `STOPPED` 時發生。

Amazon ECS 任務中繼資料 v4 範例

以下範例顯示來自每個任務中繼資料端點的範例輸出。

容器中繼資料回應範例

查詢 `${ECS_CONTAINER_METADATA_URI_V4}` 端點時，只會傳回關於容器本身的中繼資料。以下為範例輸出。

```
{
    "DockerId": "ea32192c8553fbff06c9340478a2ff089b2bb5646fb718b4ee206641c9086d66",
    "Name": "curl",
    "DockerName": "ecs-curltest-24-curl-cca48e8dcadd97805600",
    "Image": "111122223333.dkr.ecr.us-west-2.amazonaws.com/curltest:latest",
```

```
    "ImageID":
 "sha256:d691691e9652791a60114e67b365688d20d19940dde7c4736ea30e660d8d3553",
    "Labels": {
        "com.amazonaws.ecs.cluster": "default",
        "com.amazonaws.ecs.container-name": "curl",
        "com.amazonaws.ecs.task-arn": "arn:aws:ecs:us-west-2:111122223333:task/
default/8f03e41243824aea923aca126495f665",
        "com.amazonaws.ecs.task-definition-family": "curltest",
        "com.amazonaws.ecs.task-definition-version": "24"
    },
    "DesiredStatus": "RUNNING",
    "KnownStatus": "RUNNING",
    "Limits": {
        "CPU": 10,
        "Memory": 128
    },
    "CreatedAt": "2020-10-02T00:15:07.620912337Z",
    "StartedAt": "2020-10-02T00:15:08.062559351Z",
    "Type": "NORMAL",
    "LogDriver": "awslogs",
    "LogOptions": {
        "awslogs-create-group": "true",
        "awslogs-group": "/ecs/metadata",
        "awslogs-region": "us-west-2",
        "awslogs-stream": "ecs/curl/8f03e41243824aea923aca126495f665"
    },
    "ContainerARN": "arn:aws:ecs:us-west-2:111122223333:container/0206b271-
b33f-47ab-86c6-a0ba208a70a9",
    "Networks": [
        {
            "NetworkMode": "awsvpc",
            "IPv4Addresses": [
                "10.0.2.100"
            ],
            "AttachmentIndex": 0,
            "MACAddress": "0e:9e:32:c7:48:85",
            "IPv4SubnetCIDRBlock": "10.0.2.0/24",
            "PrivateDNSName": "ip-10-0-2-100.us-west-2.compute.internal",
            "SubnetGatewayIpv4Address": "10.0.2.1/24"
        }
    ]
}
```

範例任務中繼資料回應

查詢 ${ECS_CONTAINER_METADATA_URI_V4}/task 端點時，除了任務內每個容器的中繼資料外，只會傳回容器所屬任務的中繼資料。以下為範例輸出。

```
{
    "Cluster": "default",
    "TaskARN": "arn:aws:ecs:us-west-2:111122223333:task/
default/158d1c8083dd49d6b527399fd6414f5c",
    "Family": "curltest",
    "ServiceName": "MyService",
    "Revision": "26",
    "DesiredStatus": "RUNNING",
    "KnownStatus": "RUNNING",
    "PullStartedAt": "2020-10-02T00:43:06.202617438Z",
    "PullStoppedAt": "2020-10-02T00:43:06.31288465Z",
    "AvailabilityZone": "us-west-2d",
    "VPCID": "vpc-1234567890abcdef0",
    "LaunchType": "EC2",
    "Containers": [
        {
            "DockerId":
 "598cba581fe3f939459eaba1e071d5c93bb2c49b7d1ba7db6bb19deeb70d8e38",
            "Name": "~internal~ecs~pause",
            "DockerName": "ecs-curltest-26-internalecspause-e292d586b6f9dade4a00",
            "Image": "amazon/amazon-ecs-pause:0.1.0",
            "ImageID": "",
            "Labels": {
                "com.amazonaws.ecs.cluster": "default",
                "com.amazonaws.ecs.container-name": "~internal~ecs~pause",
                "com.amazonaws.ecs.task-arn": "arn:aws:ecs:us-west-2:111122223333:task/
default/158d1c8083dd49d6b527399fd6414f5c",
                "com.amazonaws.ecs.task-definition-family": "curltest",
                "com.amazonaws.ecs.task-definition-version": "26"
            },
            "DesiredStatus": "RESOURCES_PROVISIONED",
            "KnownStatus": "RESOURCES_PROVISIONED",
            "Limits": {
                "CPU": 0,
                "Memory": 0
            },
            "CreatedAt": "2020-10-02T00:43:05.602352471Z",
            "StartedAt": "2020-10-02T00:43:06.076707576Z",
            "Type": "CNI_PAUSE",
```

```
            "Networks": [
                {
                    "NetworkMode": "awsvpc",
                    "IPv4Addresses": [
                        "10.0.2.61"
                    ],
                    "AttachmentIndex": 0,
                    "MACAddress": "0e:10:e2:01:bd:91",
                    "IPv4SubnetCIDRBlock": "10.0.2.0/24",
                    "PrivateDNSName": "ip-10-0-2-61.us-west-2.compute.internal",
                    "SubnetGatewayIpv4Address": "10.0.2.1/24"
                }
            ]
        },
        {
            "DockerId":
 "ee08638adaaf009d78c248913f629e38299471d45fe7dc944d1039077e3424ca",
            "Name": "curl",
            "DockerName": "ecs-curltest-26-curl-a0e7dba5aca6d8cb2e00",
            "Image": "111122223333.dkr.ecr.us-west-2.amazonaws.com/curltest:latest",
            "ImageID":
 "sha256:d691691e9652791a60114e67b365688d20d19940dde7c4736ea30e660d8d3553",
            "Labels": {
                "com.amazonaws.ecs.cluster": "default",
                "com.amazonaws.ecs.container-name": "curl",
                "com.amazonaws.ecs.task-arn": "arn:aws:ecs:us-west-2:111122223333:task/
default/158d1c8083dd49d6b527399fd6414f5c",
                "com.amazonaws.ecs.task-definition-family": "curltest",
                "com.amazonaws.ecs.task-definition-version": "26"
            },
            "DesiredStatus": "RUNNING",
            "KnownStatus": "RUNNING",
            "Limits": {
                "CPU": 10,
                "Memory": 128
            },
            "CreatedAt": "2020-10-02T00:43:06.326590752Z",
            "StartedAt": "2020-10-02T00:43:06.767535449Z",
            "Type": "NORMAL",
            "LogDriver": "awslogs",
            "LogOptions": {
                "awslogs-create-group": "true",
                "awslogs-group": "/ecs/metadata",
                "awslogs-region": "us-west-2",
```

```
                "awslogs-stream": "ecs/curl/158d1c8083dd49d6b527399fd6414f5c"
            },
            "ContainerARN": "arn:aws:ecs:us-west-2:111122223333:container/
abb51bdd-11b4-467f-8f6c-adcfe1fe059d",
            "Networks": [
                {
                    "NetworkMode": "awsvpc",
                    "IPv4Addresses": [
                        "10.0.2.61"
                    ],
                    "AttachmentIndex": 0,
                    "MACAddress": "0e:10:e2:01:bd:91",
                    "IPv4SubnetCIDRBlock": "10.0.2.0/24",
                    "PrivateDNSName": "ip-10-0-2-61.us-west-2.compute.internal",
                    "SubnetGatewayIpv4Address": "10.0.2.1/24"
                }
            ]
        }
    ]
}
```

具有標籤中繼資料回應的任務範例

查詢 ${ECS_CONTAINER_METADATA_URI_V4}/taskWithTags 端點時，會傳回關於任務的中繼資料，包括任務和容器執行個體標籤。以下為範例輸出。

```
{
    "Cluster": "default",
    "TaskARN": "arn:aws:ecs:us-west-2:111122223333:task/
default/158d1c8083dd49d6b527399fd6414f5c",
    "Family": "curltest",
    "ServiceName": "MyService",
    "Revision": "26",
    "DesiredStatus": "RUNNING",
    "KnownStatus": "RUNNING",
    "PullStartedAt": "2020-10-02T00:43:06.202617438Z",
    "PullStoppedAt": "2020-10-02T00:43:06.31288465Z",
    "AvailabilityZone": "us-west-2d",
    "VPCID": "vpc-1234567890abcdef0",
    "TaskTags": {
        "tag-use": "task-metadata-endpoint-test"
    },
    "ContainerInstanceTags":{
```

```
        "tag_key":"tag_value"
    },
    "LaunchType": "EC2",
    "Containers": [
        {
            "DockerId":
 "598cba581fe3f939459eaba1e071d5c93bb2c49b7d1ba7db6bb19deeb70d8e38",
            "Name": "~internal~ecs~pause",
            "DockerName": "ecs-curltest-26-internalecspause-e292d586b6f9dade4a00",
            "Image": "amazon/amazon-ecs-pause:0.1.0",
            "ImageID": "",
            "Labels": {
                "com.amazonaws.ecs.cluster": "default",
                "com.amazonaws.ecs.container-name": "~internal~ecs~pause",
                "com.amazonaws.ecs.task-arn": "arn:aws:ecs:us-west-2:111122223333:task/
default/158d1c8083dd49d6b527399fd6414f5c",
                "com.amazonaws.ecs.task-definition-family": "curltest",
                "com.amazonaws.ecs.task-definition-version": "26"
            },
            "DesiredStatus": "RESOURCES_PROVISIONED",
            "KnownStatus": "RESOURCES_PROVISIONED",
            "Limits": {
                "CPU": 0,
                "Memory": 0
            },
            "CreatedAt": "2020-10-02T00:43:05.602352471Z",
            "StartedAt": "2020-10-02T00:43:06.076707576Z",
            "Type": "CNI_PAUSE",
            "Networks": [
                {
                    "NetworkMode": "awsvpc",
                    "IPv4Addresses": [
                        "10.0.2.61"
                    ],
                    "AttachmentIndex": 0,
                    "MACAddress": "0e:10:e2:01:bd:91",
                    "IPv4SubnetCIDRBlock": "10.0.2.0/24",
                    "PrivateDNSName": "ip-10-0-2-61.us-west-2.compute.internal",
                    "SubnetGatewayIpv4Address": "10.0.2.1/24"
                }
            ]
        },
        {
```

```
            "DockerId":
 "ee08638adaaf009d78c248913f629e38299471d45fe7dc944d1039077e3424ca",
            "Name": "curl",
            "DockerName": "ecs-curltest-26-curl-a0e7dba5aca6d8cb2e00",
            "Image": "111122223333.dkr.ecr.us-west-2.amazonaws.com/curltest:latest",
            "ImageID":
 "sha256:d691691e9652791a60114e67b365688d20d19940dde7c4736ea30e660d8d3553",
            "Labels": {
                "com.amazonaws.ecs.cluster": "default",
                "com.amazonaws.ecs.container-name": "curl",
                "com.amazonaws.ecs.task-arn": "arn:aws:ecs:us-west-2:111122223333:task/
default/158d1c8083dd49d6b527399fd6414f5c",
                "com.amazonaws.ecs.task-definition-family": "curltest",
                "com.amazonaws.ecs.task-definition-version": "26"
            },
            "DesiredStatus": "RUNNING",
            "KnownStatus": "RUNNING",
            "Limits": {
                "CPU": 10,
                "Memory": 128
            },
            "CreatedAt": "2020-10-02T00:43:06.326590752Z",
            "StartedAt": "2020-10-02T00:43:06.767535449Z",
            "Type": "NORMAL",
            "LogDriver": "awslogs",
            "LogOptions": {
                "awslogs-create-group": "true",
                "awslogs-group": "/ecs/metadata",
                "awslogs-region": "us-west-2",
                "awslogs-stream": "ecs/curl/158d1c8083dd49d6b527399fd6414f5c"
            },
            "ContainerARN": "arn:aws:ecs:us-west-2:111122223333:container/
abb51bdd-11b4-467f-8f6c-adcfe1fe059d",
            "Networks": [
                {
                    "NetworkMode": "awsvpc",
                    "IPv4Addresses": [
                        "10.0.2.61"
                    ],
                    "AttachmentIndex": 0,
                    "MACAddress": "0e:10:e2:01:bd:91",
                    "IPv4SubnetCIDRBlock": "10.0.2.0/24",
                    "PrivateDNSName": "ip-10-0-2-61.us-west-2.compute.internal",
                    "SubnetGatewayIpv4Address": "10.0.2.1/24"
```

```
                }
            ]
        }
    ]
}
```

具有標籤和錯誤中繼資料回應的任務範例

查詢 ${ECS_CONTAINER_METADATA_URI_V4}/taskWithTags 端點時，會傳回關於任務的中繼資料，包括任務和容器執行個體標籤。如果擷取標籤資料時發生錯誤，則會在回應中傳回錯誤。以下是與容器執行個體相關聯的 IAM 角色沒有允許 ecs:ListTagsForResource 許可時的輸出範例。

```
{
    "Cluster": "default",
    "TaskARN": "arn:aws:ecs:us-west-2:111122223333:task/
default/158d1c8083dd49d6b527399fd6414f5c",
    "Family": "curltest",
    "ServiceName": "MyService",
    "Revision": "26",
    "DesiredStatus": "RUNNING",
    "KnownStatus": "RUNNING",
    "PullStartedAt": "2020-10-02T00:43:06.202617438Z",
    "PullStoppedAt": "2020-10-02T00:43:06.31288465Z",
    "AvailabilityZone": "us-west-2d",
    "VPCID": "vpc-1234567890abcdef0",
    "Errors": [
        {
            "ErrorField": "ContainerInstanceTags",
            "ErrorCode": "AccessDeniedException",
            "ErrorMessage": "User: arn:aws:sts::111122223333:assumed-
role/ecsInstanceRole/i-0744a608689EXAMPLE is not authorized to perform:
 ecs:ListTagsForResource on resource: arn:aws:ecs:us-west-2:111122223333:container-
instance/default/2dd1b186f39845a584488d2ef155c131",
            "StatusCode": 400,
            "RequestId": "cd597ef0-272b-4643-9bd2-1de469870fa6",
            "ResourceARN": "arn:aws:ecs:us-west-2:111122223333:container-instance/
default/2dd1b186f39845a584488d2ef155c131"
        },
        {
            "ErrorField": "TaskTags",
            "ErrorCode": "AccessDeniedException",
            "ErrorMessage": "User: arn:aws:sts::111122223333:assumed-
role/ecsInstanceRole/i-0744a608689EXAMPLE is not authorized to perform:
```

```
 ecs:ListTagsForResource on resource: arn:aws:ecs:us-west-2:111122223333:task/
default/9ef30e4b7aa44d0db562749cff4983f3",
            "StatusCode": 400,
            "RequestId": "862c5986-6cd2-4aa6-87cc-70be395531e1",
            "ResourceARN": "arn:aws:ecs:us-west-2:111122223333:task/
default/9ef30e4b7aa44d0db562749cff4983f3"
        }
    ],
    "LaunchType": "EC2",
    "Containers": [
        {
            "DockerId":
 "598cba581fe3f939459eaba1e071d5c93bb2c49b7d1ba7db6bb19deeb70d8e38",
            "Name": "~internal~ecs~pause",
            "DockerName": "ecs-curltest-26-internalecspause-e292d586b6f9dade4a00",
            "Image": "amazon/amazon-ecs-pause:0.1.0",
            "ImageID": "",
            "Labels": {
                "com.amazonaws.ecs.cluster": "default",
                "com.amazonaws.ecs.container-name": "~internal~ecs~pause",
                "com.amazonaws.ecs.task-arn": "arn:aws:ecs:us-west-2:111122223333:task/
default/158d1c8083dd49d6b527399fd6414f5c",
                "com.amazonaws.ecs.task-definition-family": "curltest",
                "com.amazonaws.ecs.task-definition-version": "26"
            },
            "DesiredStatus": "RESOURCES_PROVISIONED",
            "KnownStatus": "RESOURCES_PROVISIONED",
            "Limits": {
                "CPU": 0,
                "Memory": 0
            },
            "CreatedAt": "2020-10-02T00:43:05.602352471Z",
            "StartedAt": "2020-10-02T00:43:06.076707576Z",
            "Type": "CNI_PAUSE",
            "Networks": [
                {
                    "NetworkMode": "awsvpc",
                    "IPv4Addresses": [
                        "10.0.2.61"
                    ],
                    "AttachmentIndex": 0,
                    "MACAddress": "0e:10:e2:01:bd:91",
                    "IPv4SubnetCIDRBlock": "10.0.2.0/24",
                    "PrivateDNSName": "ip-10-0-2-61.us-west-2.compute.internal",
```

```
                    "SubnetGatewayIpv4Address": "10.0.2.1/24"
                }
            ]
        },
        {
            "DockerId":
 "ee08638adaaf009d78c248913f629e38299471d45fe7dc944d1039077e3424ca",
            "Name": "curl",
            "DockerName": "ecs-curltest-26-curl-a0e7dba5aca6d8cb2e00",
            "Image": "111122223333.dkr.ecr.us-west-2.amazonaws.com/curltest:latest",
            "ImageID":
 "sha256:d691691e9652791a60114e67b365688d20d19940dde7c4736ea30e660d8d3553",
            "Labels": {
                "com.amazonaws.ecs.cluster": "default",
                "com.amazonaws.ecs.container-name": "curl",
                "com.amazonaws.ecs.task-arn": "arn:aws:ecs:us-west-2:111122223333:task/
default/158d1c8083dd49d6b527399fd6414f5c",
                "com.amazonaws.ecs.task-definition-family": "curltest",
                "com.amazonaws.ecs.task-definition-version": "26"
            },
            "DesiredStatus": "RUNNING",
            "KnownStatus": "RUNNING",
            "Limits": {
                "CPU": 10,
                "Memory": 128
            },
            "CreatedAt": "2020-10-02T00:43:06.326590752Z",
            "StartedAt": "2020-10-02T00:43:06.767535449Z",
            "Type": "NORMAL",
            "LogDriver": "awslogs",
            "LogOptions": {
                "awslogs-create-group": "true",
                "awslogs-group": "/ecs/metadata",
                "awslogs-region": "us-west-2",
                "awslogs-stream": "ecs/curl/158d1c8083dd49d6b527399fd6414f5c"
            },
            "ContainerARN": "arn:aws:ecs:us-west-2:111122223333:container/
abb51bdd-11b4-467f-8f6c-adcfe1fe059d",
            "Networks": [
                {
                    "NetworkMode": "awsvpc",
                    "IPv4Addresses": [
                        "10.0.2.61"
                    ],
```

```
                    "AttachmentIndex": 0,
                    "MACAddress": "0e:10:e2:01:bd:91",
                    "IPv4SubnetCIDRBlock": "10.0.2.0/24",
                    "PrivateDNSName": "ip-10-0-2-61.us-west-2.compute.internal",
                    "SubnetGatewayIpv4Address": "10.0.2.1/24"
                }
            ]
        }
    ]
}
```

容器統計資訊回應範例

查詢 ${ECS_CONTAINER_METADATA_URI_V4}/stats 端點時，會傳回容器的網路指標。對於使用 awsvpc 或 bridge 網路模式、託管於執行容器代理程式至少 1.43.0 版本的 Amazon EC2 執行個體中的 Amazon ECS 任務，回應中會包含額外的網路速率統計資訊。對於所有其他任務，回應只會包含累積的網路統計資訊。

以下是 Amazon EC2 上使用 bridge 網路模式的 Amazon ECS 任務的輸出範例。

```
{
    "read": "2020-10-02T00:51:13.410254284Z",
    "preread": "2020-10-02T00:51:12.406202398Z",
    "pids_stats": {
        "current": 3
    },
    "blkio_stats": {
        "io_service_bytes_recursive": [

        ],
        "io_serviced_recursive": [

        ],
        "io_queue_recursive": [

        ],
        "io_service_time_recursive": [

        ],
        "io_wait_time_recursive": [

        ],
        "io_merged_recursive": [
```

```
        ],
        "io_time_recursive": [

        ],
        "sectors_recursive": [

        ]
    },
    "num_procs": 0,
    "storage_stats": {

    },
    "cpu_stats": {
        "cpu_usage": {
            "total_usage": 360968065,
            "percpu_usage": [
                182359190,
                178608875
            ],
            "usage_in_kernelmode": 40000000,
            "usage_in_usermode": 290000000
        },
        "system_cpu_usage": 13939680000000,
        "online_cpus": 2,
        "throttling_data": {
            "periods": 0,
            "throttled_periods": 0,
            "throttled_time": 0
        }
    },
    "precpu_stats": {
        "cpu_usage": {
            "total_usage": 360968065,
            "percpu_usage": [
                182359190,
                178608875
            ],
            "usage_in_kernelmode": 40000000,
            "usage_in_usermode": 290000000
        },
        "system_cpu_usage": 13937670000000,
        "online_cpus": 2,
        "throttling_data": {
```

```
            "periods": 0,
            "throttled_periods": 0,
            "throttled_time": 0
        }
    },
    "memory_stats": {
        "usage": 1806336,
        "max_usage": 6299648,
        "stats": {
            "active_anon": 606208,
            "active_file": 0,
            "cache": 0,
            "dirty": 0,
            "hierarchical_memory_limit": 134217728,
            "hierarchical_memsw_limit": 268435456,
            "inactive_anon": 0,
            "inactive_file": 0,
            "mapped_file": 0,
            "pgfault": 4185,
            "pgmajfault": 0,
            "pgpgin": 2926,
            "pgpgout": 2778,
            "rss": 606208,
            "rss_huge": 0,
            "total_active_anon": 606208,
            "total_active_file": 0,
            "total_cache": 0,
            "total_dirty": 0,
            "total_inactive_anon": 0,
            "total_inactive_file": 0,
            "total_mapped_file": 0,
            "total_pgfault": 4185,
            "total_pgmajfault": 0,
            "total_pgpgin": 2926,
            "total_pgpgout": 2778,
            "total_rss": 606208,
            "total_rss_huge": 0,
            "total_unevictable": 0,
            "total_writeback": 0,
            "unevictable": 0,
            "writeback": 0
        },
        "limit": 134217728
    },
```

```
    "name": "/ecs-curltest-26-curl-c2e5f6e0cf91b0bead01",
    "id": "5fc21e5b015f899d22618f8aede80b6d70d71b2a75465ea49d9462c8f3d2d3af",
    "networks": {
        "eth0": {
            "rx_bytes": 84,
            "rx_packets": 2,
            "rx_errors": 0,
            "rx_dropped": 0,
            "tx_bytes": 84,
            "tx_packets": 2,
            "tx_errors": 0,
            "tx_dropped": 0
        }
    },
    "network_rate_stats": {
        "rx_bytes_per_sec": 0,
        "tx_bytes_per_sec": 0
    }
}
```

任務統計資訊回應範例

查詢 ${ECS_CONTAINER_METADATA_URI_V4}/task/stats 端點時，會傳回關於容器所屬任務的網路指標。以下為範例輸出。

```
{
    "01999f2e5c6cf4df3873f28950e6278813408f281c54778efec860d0caad4854": {
        "read": "2020-10-02T00:51:32.51467703Z",
        "preread": "2020-10-02T00:51:31.50860463Z",
        "pids_stats": {
            "current": 1
        },
        "blkio_stats": {
            "io_service_bytes_recursive": [

            ],
            "io_serviced_recursive": [

            ],
            "io_queue_recursive": [

            ],
            "io_service_time_recursive": [
```

```
            ],
            "io_wait_time_recursive": [

            ],
            "io_merged_recursive": [

            ],
            "io_time_recursive": [

            ],
            "sectors_recursive": [

            ]
        },
        "num_procs": 0,
        "storage_stats": {

        },
        "cpu_stats": {
            "cpu_usage": {
                "total_usage": 177232665,
                "percpu_usage": [
                    13376224,
                    163856441
                ],
                "usage_in_kernelmode": 0,
                "usage_in_usermode": 160000000
            },
            "system_cpu_usage": 13977820000000,
            "online_cpus": 2,
            "throttling_data": {
                "periods": 0,
                "throttled_periods": 0,
                "throttled_time": 0
            }
        },
        "precpu_stats": {
            "cpu_usage": {
                "total_usage": 177232665,
                "percpu_usage": [
                    13376224,
                    163856441
                ],
```

```
                "usage_in_kernelmode": 0,
                "usage_in_usermode": 160000000
            },
            "system_cpu_usage": 13975800000000,
            "online_cpus": 2,
            "throttling_data": {
                "periods": 0,
                "throttled_periods": 0,
                "throttled_time": 0
            }
        },
        "memory_stats": {
            "usage": 532480,
            "max_usage": 6279168,
            "stats": {
                "active_anon": 40960,
                "active_file": 0,
                "cache": 0,
                "dirty": 0,
                "hierarchical_memory_limit": 9223372036854771712,
                "hierarchical_memsw_limit": 9223372036854771712,
                "inactive_anon": 0,
                "inactive_file": 0,
                "mapped_file": 0,
                "pgfault": 2033,
                "pgmajfault": 0,
                "pgpgin": 1734,
                "pgpgout": 1724,
                "rss": 40960,
                "rss_huge": 0,
                "total_active_anon": 40960,
                "total_active_file": 0,
                "total_cache": 0,
                "total_dirty": 0,
                "total_inactive_anon": 0,
                "total_inactive_file": 0,
                "total_mapped_file": 0,
                "total_pgfault": 2033,
                "total_pgmajfault": 0,
                "total_pgpgin": 1734,
                "total_pgpgout": 1724,
                "total_rss": 40960,
                "total_rss_huge": 0,
                "total_unevictable": 0,
```

```
                "total_writeback": 0,
                "unevictable": 0,
                "writeback": 0
            },
            "limit": 4073377792
        },
        "name": "/ecs-curltest-26-internalecspause-a6bcc3dbadfacfe85300",
        "id": "01999f2e5c6cf4df3873f28950e6278813408f281c54778efec860d0caad4854",
        "networks": {
            "eth0": {
                "rx_bytes": 84,
                "rx_packets": 2,
                "rx_errors": 0,
                "rx_dropped": 0,
                "tx_bytes": 84,
                "tx_packets": 2,
                "tx_errors": 0,
                "tx_dropped": 0
            }
        },
        "network_rate_stats": {
            "rx_bytes_per_sec": 0,
            "tx_bytes_per_sec": 0
        }
    },
    "5fc21e5b015f899d22618f8aede80b6d70d71b2a75465ea49d9462c8f3d2d3af": {
        "read": "2020-10-02T00:51:32.512771349Z",
        "preread": "2020-10-02T00:51:31.510597736Z",
        "pids_stats": {
            "current": 3
        },
        "blkio_stats": {
            "io_service_bytes_recursive": [

            ],
            "io_serviced_recursive": [

            ],
            "io_queue_recursive": [

            ],
            "io_service_time_recursive": [

            ],
```

```
            "io_wait_time_recursive": [

            ],
            "io_merged_recursive": [

            ],
            "io_time_recursive": [

            ],
            "sectors_recursive": [

            ]
        },
        "num_procs": 0,
        "storage_stats": {

        },
        "cpu_stats": {
            "cpu_usage": {
                "total_usage": 379075681,
                "percpu_usage": [
                    191355275,
                    187720406
                ],
                "usage_in_kernelmode": 60000000,
                "usage_in_usermode": 310000000
            },
            "system_cpu_usage": 13977800000000,
            "online_cpus": 2,
            "throttling_data": {
                "periods": 0,
                "throttled_periods": 0,
                "throttled_time": 0
            }
        },
        "precpu_stats": {
            "cpu_usage": {
                "total_usage": 378825197,
                "percpu_usage": [
                    191104791,
                    187720406
                ],
                "usage_in_kernelmode": 60000000,
                "usage_in_usermode": 310000000
```

```
            },
            "system_cpu_usage": 13975800000000,
            "online_cpus": 2,
            "throttling_data": {
                "periods": 0,
                "throttled_periods": 0,
                "throttled_time": 0
            }
        },
        "memory_stats": {
            "usage": 1814528,
            "max_usage": 6299648,
            "stats": {
                "active_anon": 606208,
                "active_file": 0,
                "cache": 0,
                "dirty": 0,
                "hierarchical_memory_limit": 134217728,
                "hierarchical_memsw_limit": 268435456,
                "inactive_anon": 0,
                "inactive_file": 0,
                "mapped_file": 0,
                "pgfault": 5377,
                "pgmajfault": 0,
                "pgpgin": 3613,
                "pgpgout": 3465,
                "rss": 606208,
                "rss_huge": 0,
                "total_active_anon": 606208,
                "total_active_file": 0,
                "total_cache": 0,
                "total_dirty": 0,
                "total_inactive_anon": 0,
                "total_inactive_file": 0,
                "total_mapped_file": 0,
                "total_pgfault": 5377,
                "total_pgmajfault": 0,
                "total_pgpgin": 3613,
                "total_pgpgout": 3465,
                "total_rss": 606208,
                "total_rss_huge": 0,
                "total_unevictable": 0,
                "total_writeback": 0,
                "unevictable": 0,
```

```
                "writeback": 0
            },
            "limit": 134217728
        },
        "name": "/ecs-curltest-26-curl-c2e5f6e0cf91b0bead01",
        "id": "5fc21e5b015f899d22618f8aede80b6d70d71b2a75465ea49d9462c8f3d2d3af",
        "networks": {
            "eth0": {
                "rx_bytes": 84,
                "rx_packets": 2,
                "rx_errors": 0,
                "rx_dropped": 0,
                "tx_bytes": 84,
                "tx_packets": 2,
                "tx_errors": 0,
                "tx_dropped": 0
            }
        },
        "network_rate_stats": {
            "rx_bytes_per_sec": 0,
            "tx_bytes_per_sec": 0
        }
    }
}
```

Amazon ECS 任務中繼資料端點第 3 版

> **Important**
>
> 不再積極維護任務中繼資料第 3 版端點。建議您更新任務中繼資料第 4 版端點，以取得最新的中繼資料端點資訊。如需詳細資訊，請參閱the section called “任務中繼資料端點第 4 版”。如果您使用託管在 上的 Amazon ECS 任務 AWS Fargate，請參閱 Amazon ECS 任務中繼資料端點第 3 版，以取得 Fargate 上的任務。

從 Amazon ECS 容器代理程式 1.21.0 版開始，代理程式會將稱為 `ECS_CONTAINER_METADATA_URI` 的環境變數插入任務中的每個容器。當您查詢任務中繼資料第 3 版端點時，有各種任務中繼資料和 Docker 統計資訊可供任務使用。對於使用 `bridge` 網路模式的工作，查詢 `/stats` 端點時，可使用網路指標。

根據預設，對於在平台版本 1.3.0 或更新版本上使用 Fargate 啟動類型的任務，以及使用 EC2 啟動類型的任務，會啟用任務中繼資料端點第 3 版功能，並在至少執行 Amazon ECS 容器代理程式 1.21.0 版的 Amazon EC2 Linux 基礎設施，或者在至少執行 Amazon ECS 容器代理程式 `1.54.0` 版的 Amazon EC2 Windows 基礎設施上啟動該功能，並使用 `awsvpc` 網路模式。如需詳細資訊，請參閱Amazon ECS Linux 容器執行個體管理。

您可以將代理更新為最新版本，以在舊容器執行個體上新增此功能的支援。如需詳細資訊，請參閱更新 Amazon ECS 容器代理程式。

> **Important**
>
> 對於使用 Fargate 啟動類型和低於 1.3.0 版平台版本的任務，支援任務中繼資料第 2 版端點。如需詳細資訊，請參閱Amazon ECS 任務中繼資料端點第 2 版。

任務中繼資料端點第 3 版路徑

下列任務中繼資料端點可供容器使用：

`${ECS_CONTAINER_METADATA_URI}`

此路徑傳回容器的 JSON 中繼資料。

`${ECS_CONTAINER_METADATA_URI}/task`

此路徑傳回任務的中繼資料 JSON，包括與任務相關聯之所有容器的容器 ID 和名稱清單。如需此端點之回應的詳細資訊，請參閱「Amazon ECS 任務中繼資料 v3 JSON 回應」。

`${ECS_CONTAINER_METADATA_URI}/taskWithTags`

除了可使用 `ListTagsForResource` API 擷取的任務和容器執行個體標籤之外，此路徑還會傳回包含在 `/task` 端點內的任務的中繼資料。

`${ECS_CONTAINER_METADATA_URI}/stats`

此路徑傳回特定 Docker 容器的 Docker 統計資訊 JSON。如需每個所傳回統計資訊的詳細資訊，請參閱 Docker API 文件中的 ContainerStats。

`${ECS_CONTAINER_METADATA_URI}/task/stats`

此路徑傳回與任務相關聯之所有容器的 Docker 統計資訊 JSON。如需每個所傳回統計資訊的詳細資訊，請參閱 Docker API 文件中的 ContainerStats。

Amazon ECS 任務中繼資料 v3 JSON 回應

任務中繼資料端點 (`${ECS_CONTAINER_METADATA_URI}/task`) JSON 回應會傳回下列資訊。

`Cluster`

任務所屬 Amazon ECS 叢集的 Amazon Resource Name (ARN) 或簡短名稱。

`TaskARN`

容器所屬任務的完整 Amazon Resource Name (ARN)。

`Family`

任務的 Amazon ECS 任務定義系列。

`Revision`

任務的 Amazon ECS 任務定義修訂。

`DesiredStatus`

Amazon ECS 中任務的所需狀態。

`KnownStatus`

Amazon ECS 中任務的已知狀態。

`Limits`

在任務層級指定的資源限制，例如 CPU (以 vCPU 表示) 和記憶體。如果未定義資源限制，則會省略此參數。

`PullStartedAt`

第一個容器映像提取的開始時間戳記。

`PullStoppedAt`

最後一個容器映像提取的完成時間戳記。

`AvailabilityZone`

任務所在的可用區域。

> **Note**
>
> 可用區域中繼資料僅適用於使用平台第 1.4 版或更新版本 (Linux) 或 1.0.0 或更新版本 (Windows) 的 Fargate 任務。

`Containers`

與任務相關聯之每個容器的容器中繼資料清單。

`DockerId`

容器的 Docker ID。

`Name`

任務定義中指定的容器名稱。

`DockerName`

提供給 Docker 的容器名稱。Amazon ECS 容器代理程式會產生容器的唯一名稱，以避免在單一執行個體上執行相同任務定義的多個複本時，發生名稱衝突。

`Image`

容器的映像。

`ImageID`

映像的 SHA-256 摘要。

`Ports`

向容器開放的任何連接埠。如果未開放連接埠，則會省略此參數。

`Labels`

任何套用至容器的標籤。如果未套用標籤，則會省略此參數。

`DesiredStatus`

Amazon ECS 中容器的所需狀態。

`KnownStatus`

Amazon ECS 中容器的已知狀態。

`ExitCode`

容器的結束代碼。如果容器尚未結束，則會省略此參數。

`Limits`

在容器層級指定的資源限制，例如 CPU (以 CPU 單位表示) 和記憶體。如果未定義資源限制，則會省略此參數。

`CreatedAt`

容器的建立時間戳記。如果尚未建立容器，則會省略此參數。

`StartedAt`

容器的啟動時間戳記。如果尚未啟動容器，則會省略此參數。

`FinishedAt`

容器的停止時間戳記。如果尚未停止容器，則會省略此參數。

`Type`

容器的類型。任務定義中指定的容器類型為 NORMAL。您可以忽略其他容器類型，這些是 Amazon ECS 容器代理程式用來佈建內部任務資源的容器類型。

`Networks`

容器的網路資訊，例如網路模式和 IP 地址。如果未定義網路資訊，則會省略此參數。

`ClockDrift`

關於參考時間和系統時間之間差異的資訊。這適用於 Linux 作業系統。此功能使用 Amazon Time Sync Service 來測量時鐘準確性，並提供容器的時鐘錯誤界限。如需詳細資訊，請參閱《Amazon EC2 Linux 執行個體使用者指南》中的設定 Linux 執行個體的時間。

`ReferenceTime`

時鐘精確度的基礎。Amazon ECS 透過 NTP 使用世界協調時間 (UTC) 的全球標準，例如 `2021-09-07T16:57:44Z`。

`ClockErrorBound`

時鐘誤差的測量值，定義為 UTC 的偏移。此誤差是參考時間和系統時間之間的毫秒差異。

`ClockSynchronizationStatus`

指出系統時間和參考時間之間最近的同步嘗試是否成功。

有效值為 SYNCHRONIZED 和 NOT_SYNCHRONIZED。

`ExecutionStoppedAt`

任務 DesiredStatus 移至 STOPPED 時的時間戳記。這會在基本容器移至 STOPPED 時發生。

Amazon ECS 任務中繼資料 v3 範例

以下範例顯示來自任務中繼資料端點的範例輸出。

容器中繼資料回應範例

查詢 ${ECS_CONTAINER_METADATA_URI} 端點時，只會傳回關於容器本身的中繼資料。以下為範例輸出。

```
{
    "DockerId": "43481a6ce4842eec8fe72fc28500c6b52edcc0917f105b83379f88cac1ff3946",
    "Name": "nginx-curl",
    "DockerName": "ecs-nginx-5-nginx-curl-ccccb9f49db0dfe0d901",
    "Image": "nrdlngr/nginx-curl",
    "ImageID":
 "sha256:2e00ae64383cfc865ba0a2ba37f61b50a120d2d9378559dcd458dc0de47bc165",
    "Labels": {
        "com.amazonaws.ecs.cluster": "default",
        "com.amazonaws.ecs.container-name": "nginx-curl",
        "com.amazonaws.ecs.task-arn": "arn:aws:ecs:us-
east-2:012345678910:task/9781c248-0edd-4cdb-9a93-f63cb662a5d3",
        "com.amazonaws.ecs.task-definition-family": "nginx",
        "com.amazonaws.ecs.task-definition-version": "5"
    },
    "DesiredStatus": "RUNNING",
    "KnownStatus": "RUNNING",
    "Limits": {
        "CPU": 512,
        "Memory": 512
    },
    "CreatedAt": "2018-02-01T20:55:10.554941919Z",
    "StartedAt": "2018-02-01T20:55:11.064236631Z",
    "Type": "NORMAL",
    "Networks": [
        {
            "NetworkMode": "awsvpc",
            "IPv4Addresses": [
                "10.0.2.106"
            ]
        }
    ]
}
```

範例任務中繼資料回應

查詢 ${ECS_CONTAINER_METADATA_URI}/task 端點時，只會傳回容器所屬任務的中繼資料。以下為範例輸出。

下列為單一容器任務的 JSON 回應。

```
{
  "Cluster": "default",
  "TaskARN": "arn:aws:ecs:us-east-2:012345678910:task/9781c248-0edd-4cdb-9a93-
f63cb662a5d3",
  "Family": "nginx",
  "Revision": "5",
  "DesiredStatus": "RUNNING",
  "KnownStatus": "RUNNING",
  "Containers": [
    {
      "DockerId": "731a0d6a3b4210e2448339bc7015aaa79bfe4fa256384f4102db86ef94cbbc4c",
      "Name": "~internal~ecs~pause",
      "DockerName": "ecs-nginx-5-internalecspause-acc699c0cbf2d6d11700",
      "Image": "amazon/amazon-ecs-pause:0.1.0",
      "ImageID": "",
      "Labels": {
        "com.amazonaws.ecs.cluster": "default",
        "com.amazonaws.ecs.container-name": "~internal~ecs~pause",
        "com.amazonaws.ecs.task-arn": "arn:aws:ecs:us-
east-2:012345678910:task/9781c248-0edd-4cdb-9a93-f63cb662a5d3",
        "com.amazonaws.ecs.task-definition-family": "nginx",
        "com.amazonaws.ecs.task-definition-version": "5"
      },
      "DesiredStatus": "RESOURCES_PROVISIONED",
      "KnownStatus": "RESOURCES_PROVISIONED",
      "Limits": {
        "CPU": 0,
        "Memory": 0
      },
      "CreatedAt": "2018-02-01T20:55:08.366329616Z",
      "StartedAt": "2018-02-01T20:55:09.058354915Z",
      "Type": "CNI_PAUSE",
      "Networks": [
        {
          "NetworkMode": "awsvpc",
          "IPv4Addresses": [
            "10.0.2.106"
          ]
        }
      ]
    },
    {
```

```
      "DockerId": "43481a6ce4842eec8fe72fc28500c6b52edcc0917f105b83379f88cac1ff3946",
      "Name": "nginx-curl",
      "DockerName": "ecs-nginx-5-nginx-curl-ccccb9f49db0dfe0d901",
      "Image": "nrdlngr/nginx-curl",
      "ImageID":
 "sha256:2e00ae64383cfc865ba0a2ba37f61b50a120d2d9378559dcd458dc0de47bc165",
      "Labels": {
        "com.amazonaws.ecs.cluster": "default",
        "com.amazonaws.ecs.container-name": "nginx-curl",
        "com.amazonaws.ecs.task-arn": "arn:aws:ecs:us-
east-2:012345678910:task/9781c248-0edd-4cdb-9a93-f63cb662a5d3",
        "com.amazonaws.ecs.task-definition-family": "nginx",
        "com.amazonaws.ecs.task-definition-version": "5"
      },
      "DesiredStatus": "RUNNING",
      "KnownStatus": "RUNNING",
      "Limits": {
        "CPU": 512,
        "Memory": 512
      },
      "CreatedAt": "2018-02-01T20:55:10.554941919Z",
      "StartedAt": "2018-02-01T20:55:11.064236631Z",
      "Type": "NORMAL",
      "Networks": [
        {
          "NetworkMode": "awsvpc",
          "IPv4Addresses": [
            "10.0.2.106"
          ]
        }
      ]
    }
  ],
  "PullStartedAt": "2018-02-01T20:55:09.372495529Z",
  "PullStoppedAt": "2018-02-01T20:55:10.552018345Z",
  "AvailabilityZone": "us-east-2b"
}
```

Amazon ECS 任務中繼資料端點第 2 版

> **Important**
>
> 不再積極維護任務中繼資料第 2 版端點。建議您更新任務中繼資料第 4 版端點，以取得最新的中繼資料端點資訊。如需詳細資訊，請參閱the section called “任務中繼資料端點第 4 版”。

從 Amazon ECS 容器代理程式 1.17.0 版開始，在 Amazon ECS 容器代理提供之 HTTP 端點使用 awsvpc 網路模式的任務，皆可使用各種任務中繼資料和 Docker 統計資訊。

屬於使用 awsvpc 網路模式啟動之任務的所有容器都接收預先定義連結本機地址範圍內的本機 IPv4 地址。當容器查詢中繼資料端點時，Amazon ECS 容器代理程式可以根據任務的唯一 IP 地址，判斷容器屬於哪些任務，並傳回該任務的中繼資料和統計資料。

啟用任務中繼資料

對於下列項目，預設會啟用任務中繼資料第 2 版功能：

- 使用 Fargate 啟動類型且使用平台 1.1.0 版或更新版本的任務。如需詳細資訊，請參閱適用於 Amazon ECS 的 Fargate 平台版本。
- 使用 EC2 啟動類型的任務，該任務也會使用 awsvpc 網路模式，且在至少執行 Amazon ECS 容器代理程式 1.17.0 版的 Amazon EC2 Linux 基礎設施上啟動，或在至少執行 1.54.0 版 Amazon ECS 容器代理程式的 Amazon EC2 Windows 基礎設施上啟動。如需詳細資訊，請參閱Amazon ECS Linux 容器執行個體管理。

您可以將代理更新為最新版本，以在舊容器執行個體上新增此功能的支援。如需詳細資訊，請參閱更新 Amazon ECS 容器代理程式。

任務中繼資料端點路徑

容器可使用下列 API 端點：

`169.254.170.2/v2/metadata`

此端點會傳回任務的中繼資料 JSON，包括與任務相關聯之所有容器的容器 ID 和名稱清單。如需此端點之回應的詳細資訊，請參閱「任務中繼資料 JSON 回應」。

`169.254.170.2/v2/metadata/<container-id>`

此端點會傳回指定 Docker 容器 ID 的中繼資料 JSON。

`169.254.170.2/v2/metadata/taskWithTags`

除了可使用 `ListTagsForResource` API 擷取的任務和容器執行個體標籤之外，此路徑還會傳回包含在 `/task` 端點內的任務的中繼資料。

`169.254.170.2/v2/stats`

此端點會傳回與任務相關聯之所有容器的 Docker 統計資訊 JSON。如需每個所傳回統計資訊的詳細資訊，請參閱 Docker API 文件中的 ContainerStats。

`169.254.170.2/v2/stats/<container-id>`

此端點會傳回指定 Docker 容器 ID 的 Docker 統計資訊 JSON。如需每個所傳回統計資訊的詳細資訊，請參閱 Docker API 文件中的 ContainerStats。

任務中繼資料 JSON 回應

任務中繼資料端點 (`169.254.170.2/v2/metadata`) JSON 回應會傳回下列資訊。

`Cluster`

任務所屬 Amazon ECS 叢集的 Amazon Resource Name (ARN) 或簡短名稱。

`TaskARN`

容器所屬任務的完整 Amazon Resource Name (ARN)。

`Family`

任務的 Amazon ECS 任務定義系列。

`Revision`

任務的 Amazon ECS 任務定義修訂。

`DesiredStatus`

Amazon ECS 中任務的所需狀態。

`KnownStatus`

Amazon ECS 中任務的已知狀態。

`Limits`

在任務層級指定的資源限制，例如 CPU (以 vCPU 表示) 和記憶體。如果未定義資源限制，則會省略此參數。

`PullStartedAt`

第一個容器映像提取的開始時間戳記。

`PullStoppedAt`

最後一個容器映像提取的完成時間戳記。

`AvailabilityZone`

任務所在的可用區域。

> **Note**
>
> 可用區域中繼資料僅適用於使用平台第 1.4 版或更新版本 (Linux) 或 1.0.0 或更新版本 (Windows) 的 Fargate 任務。

`Containers`

與任務相關聯之每個容器的容器中繼資料清單。

`DockerId`

容器的 Docker ID。

`Name`

任務定義中指定的容器名稱。

`DockerName`

提供給 Docker 的容器名稱。Amazon ECS 容器代理程式會產生容器的唯一名稱，以避免在單一執行個體上執行相同任務定義的多個複本時，發生名稱衝突。

`Image`

容器的映像。

`ImageID`

映像的 SHA-256 摘要。

`Ports`

向容器開放的任何連接埠。如果未開放連接埠，則會省略此參數。

`Labels`

任何套用至容器的標籤。如果未套用標籤，則會省略此參數。

`DesiredStatus`

Amazon ECS 中容器的所需狀態。

`KnownStatus`

Amazon ECS 中容器的已知狀態。

`ExitCode`

容器的結束代碼。如果容器尚未結束，則會省略此參數。

`Limits`

在容器層級指定的資源限制，例如 CPU (以 CPU 單位表示) 和記憶體。如果未定義資源限制，則會省略此參數。

`CreatedAt`

容器的建立時間戳記。如果尚未建立容器，則會省略此參數。

`StartedAt`

容器的啟動時間戳記。如果尚未啟動容器，則會省略此參數。

`FinishedAt`

容器的停止時間戳記。如果尚未停止容器，則會省略此參數。

`Type`

容器的類型。任務定義中指定的容器類型為 `NORMAL`。您可以忽略其他容器類型，這些是 Amazon ECS 容器代理程式用來佈建內部任務資源的容器類型。

`Networks`

容器的網路資訊，例如網路模式和 IP 地址。如果未定義網路資訊，則會省略此參數。

`ClockDrift`

關於參考時間和系統時間之間差異的資訊。這適用於 Linux 作業系統。此功能使用 Amazon Time Sync Service 來測量時鐘準確性，並提供容器的時鐘錯誤界限。如需詳細資訊，請參閱《Amazon EC2 Linux 執行個體使用者指南》中的設定 Linux 執行個體的時間。

`ReferenceTime`

時鐘精確度的基礎。Amazon ECS 透過 NTP 使用世界協調時間 (UTC) 的全球標準，例如 `2021-09-07T16:57:44Z`。

`ClockErrorBound`

時鐘誤差的測量值，定義為 UTC 的偏移。此誤差是參考時間和系統時間之間的毫秒差異。

`ClockSynchronizationStatus`

指出系統時間和參考時間之間最近的同步嘗試是否成功。

有效值為 `SYNCHRONIZED` 和 `NOT_SYNCHRONIZED`。

`ExecutionStoppedAt`

任務 `DesiredStatus` 移至 `STOPPED` 時的時間戳記。這會在基本容器移至 `STOPPED` 時發生。

範例任務中繼資料回應

下列為單一容器任務的 JSON 回應。

```
{
  "Cluster": "default",
  "TaskARN": "arn:aws:ecs:us-east-2:012345678910:task/9781c248-0edd-4cdb-9a93-
f63cb662a5d3",
  "Family": "nginx",
  "Revision": "5",
  "DesiredStatus": "RUNNING",
  "KnownStatus": "RUNNING",
  "Containers": [
    {
      "DockerId": "731a0d6a3b4210e2448339bc7015aaa79bfe4fa256384f4102db86ef94cbbc4c",
      "Name": "~internal~ecs~pause",
      "DockerName": "ecs-nginx-5-internalecspause-acc699c0cbf2d6d11700",
      "Image": "amazon/amazon-ecs-pause:0.1.0",
      "ImageID": "",
      "Labels": {
        "com.amazonaws.ecs.cluster": "default",
        "com.amazonaws.ecs.container-name": "~internal~ecs~pause",
        "com.amazonaws.ecs.task-arn": "arn:aws:ecs:us-
east-2:012345678910:task/9781c248-0edd-4cdb-9a93-f63cb662a5d3",
        "com.amazonaws.ecs.task-definition-family": "nginx",
        "com.amazonaws.ecs.task-definition-version": "5"
```

```
      },
      "DesiredStatus": "RESOURCES_PROVISIONED",
      "KnownStatus": "RESOURCES_PROVISIONED",
      "Limits": {
        "CPU": 0,
        "Memory": 0
      },
      "CreatedAt": "2018-02-01T20:55:08.366329616Z",
      "StartedAt": "2018-02-01T20:55:09.058354915Z",
      "Type": "CNI_PAUSE",
      "Networks": [
        {
          "NetworkMode": "awsvpc",
          "IPv4Addresses": [
            "10.0.2.106"
          ]
        }
      ]
    },
    {
      "DockerId": "43481a6ce4842eec8fe72fc28500c6b52edcc0917f105b83379f88cac1ff3946",
      "Name": "nginx-curl",
      "DockerName": "ecs-nginx-5-nginx-curl-ccccb9f49db0dfe0d901",
      "Image": "nrdlngr/nginx-curl",
      "ImageID":
 "sha256:2e00ae64383cfc865ba0a2ba37f61b50a120d2d9378559dcd458dc0de47bc165",
      "Labels": {
        "com.amazonaws.ecs.cluster": "default",
        "com.amazonaws.ecs.container-name": "nginx-curl",
        "com.amazonaws.ecs.task-arn": "arn:aws:ecs:us-
east-2:012345678910:task/9781c248-0edd-4cdb-9a93-f63cb662a5d3",
        "com.amazonaws.ecs.task-definition-family": "nginx",
        "com.amazonaws.ecs.task-definition-version": "5"
      },
      "DesiredStatus": "RUNNING",
      "KnownStatus": "RUNNING",
      "Limits": {
        "CPU": 512,
        "Memory": 512
      },
      "CreatedAt": "2018-02-01T20:55:10.554941919Z",
      "StartedAt": "2018-02-01T20:55:11.064236631Z",
      "Type": "NORMAL",
      "Networks": [
```

```
        {
          "NetworkMode": "awsvpc",
          "IPv4Addresses": [
            "10.0.2.106"
          ]
        }
      ]
    }
  ],
  "PullStartedAt": "2018-02-01T20:55:09.372495529Z",
  "PullStoppedAt": "2018-02-01T20:55:10.552018345Z",
  "AvailabilityZone": "us-east-2b"
}
```

Amazon ECS 任務中繼資料可用於 Fargate 上的任務

Amazon ECS on Fargate 提供一種方法，可擷取關於您容器和任務的各種中繼資料、網路指標，以及 Docker 統計資料。這稱為任務中繼資料端點。以下任務中繼資料端點版本可用於 Amazon ECS on Fargate 任務：

- 任務中繼資料端點第 4 版：適用於使用平台版本 1.4.0 或更新版本的任務。
- 任務中繼資料端點第 3 版：適用於使用平台版本 1.1.0 或更新版本的任務。

屬於使用 `awsvpc` 網路模式啟動之任務的所有容器都接收預先定義連結本機地址範圍內的本機 IPv4 地址。當容器查詢中繼資料端點時，容器代理程式可以根據任務的唯一 IP 地址，判斷容器屬於哪些任務，並傳回該任務的中繼資料和統計資料。

主題

- Fargate 上任務的 Amazon ECS 任務中繼資料端點第 4 版
- Fargate 上任務的 Amazon ECS 任務中繼資料端點第 3 版

Fargate 上任務的 Amazon ECS 任務中繼資料端點第 4 版

> **Important**
>
> 如果您使用託管在 Amazon EC2 執行個體上的 Amazon ECS 任務，請參閱 Amazon ECS 任務中繼資料端點。

從 Fargate 平台版本 1.4.0 開始，名為 ECS_CONTAINER_METADATA_URI_V4 的環境變數會插入任務中的每個容器。當您查詢任務中繼資料端點第 4 版時，有各種任務中繼資料和 Docker 統計資訊可供任務使用。

任務中繼資料端點第 4 版功能與第 3 版端點類似，但第 4 版為您的容器和任務提供其他網路中繼資料。查詢 /stats 端點時，也可以使用其他網路指標。

使用平台版本 1.4.0或更新版本執行的所有 Amazon ECS 任務 AWS Fargate ，其任務中繼資料端點預設為開啟。

> **Note**
>
> 為了避免未來需要建立新的任務中繼資料端點版本，可以將其他中繼資料新增至第 4 版輸出。我們不會移除任何現有中繼資料或變更中繼資料欄位名稱。

Fargate 任務中繼資料端點第 4 版路徑

下列任務中繼資料端點可供容器使用：

`${ECS_CONTAINER_METADATA_URI_V4}`

此路徑傳回容器的中繼資料。

`${ECS_CONTAINER_METADATA_URI_V4}/task`

此路徑傳回任務的中繼資料，包括與任務相關聯之所有容器的容器 ID 和名稱清單。如需此端點之回應的詳細資訊，請參閱「Fargate 上任務的 Amazon ECS 任務中繼資料 v4 JSON 回應」。

`${ECS_CONTAINER_METADATA_URI_V4}/stats`

此路徑傳回 Docker 容器的 Docker 統計資訊。如需每個所傳回統計資訊的詳細資訊，請參閱 Docker API 文件中的 ContainerStats。

> **Note**
>
> 上的 Amazon ECS 任務 AWS Fargate 需要容器執行約 1 秒，才能傳回容器統計資料。

`${ECS_CONTAINER_METADATA_URI_V4}/task/stats`

此路徑傳回與任務相關聯之所有容器的 Docker 統計資訊。如需每個所傳回統計資訊的詳細資訊，請參閱 Docker API 文件中的 ContainerStats。

> **Note**
>
> 上的 Amazon ECS 任務 AWS Fargate 需要容器執行約 1 秒，才能傳回容器統計資料。

Fargate 上任務的 Amazon ECS 任務中繼資料 v4 JSON 回應

任務中繼資料端點 (`${ECS_CONTAINER_METADATA_URI_V4}/task`) JSON 回應會傳回下列中繼資料。

`Cluster`

任務所屬 Amazon ECS 叢集的 Amazon Resource Name (ARN) 或簡短名稱。

`VPCID`

Amazon EC2 容器執行個體的 VPC ID。此欄位僅適用於 Amazon EC2 執行個體。

> **Note**
>
> 只有在使用 Amazon ECS 容器代理程式版本 `1.63.1` 或更新版本時，才包括該 `VPCID` 中繼資料。

`TaskARN`

容器所屬任務的完整 Amazon Resource Name (ARN)。

`Family`

任務的 Amazon ECS 任務定義系列。

`Revision`

任務的 Amazon ECS 任務定義修訂。

`DesiredStatus`

Amazon ECS 中任務的所需狀態。

`KnownStatus`

Amazon ECS 中任務的已知狀態。

`Limits`

在任務層級指定的資源限制，例如 CPU (以 vCPU 表示) 和記憶體。如果未定義資源限制，則會省略此參數。

`PullStartedAt`

第一個容器映像提取的開始時間戳記。

`PullStoppedAt`

最後一個容器映像提取的完成時間戳記。

`AvailabilityZone`

任務所在的可用區域。

> **Note**
>
> 可用區域中繼資料僅適用於使用平台第 1.4 版或更新版本 (Linux) 或 1.0.0 (Windows) 的 Fargate 任務。

`LaunchType`

任務使用的啟動類型。使用叢集容量提供者時，這會指出任務是使用 Fargate 還是 EC2 基礎設施。

> **Note**
>
> 只有在使用 Amazon ECS Linux 容器代理程式版本 `1.45.0` 或更新版本 (Linux) 或 1.0.0 或更新版本 (Windows) 時，才包括此 `LaunchType` 中繼資料。

`EphemeralStorageMetrics`

此任務暫時性儲存的保留大小和目前使用量。

> **Note**
>
> Fargate 會在磁盤上保留空間。此空間僅由 Fargate 使用。我們不會向您收費。它不會顯示在這些指標中。但是，您可以在其他工具 (例如 `df`) 中看到此額外儲存空間。

`Utilized`

此任務目前的暫時性儲存使用量 (MiB)。

`Reserved`

此任務的保留暫時性儲存裝置 (MiB)。在執行中的任務中，無法變更暫時性儲存的大小。您可以在任務定義中指定 `ephermalStorage` 物件來變更暫時性儲存量。`ephermalStorage` 是以 GiB 為單位，而不是以 MiB 為單位。此 `ephermalStorage` 與 `EphemeralStorageMetrics` 僅適用於 Fargate Linux 平台版本 1.4.0 或更高版本。

`Containers`

與任務相關聯之每個容器的容器中繼資料清單。

`DockerId`

容器的 Docker ID。

當您使用 Fargate 時，ID 為 32 位十六進制，後跟 10 位數字。

`Name`

任務定義中指定的容器名稱。

`DockerName`

提供給 Docker 的容器名稱。Amazon ECS 容器代理程式會產生容器的唯一名稱，以避免在單一執行個體上執行相同任務定義的多個複本時，發生名稱衝突。

`Image`

容器的映像。

`ImageID`

映像的 SHA-256 摘要。

`Ports`

向容器開放的任何連接埠。如果未開放連接埠，則會省略此參數。

`Labels`

任何套用至容器的標籤。如果未套用標籤，則會省略此參數。

`DesiredStatus`

Amazon ECS 中容器的所需狀態。

`KnownStatus`

Amazon ECS 中容器的已知狀態。

`ExitCode`

容器的結束代碼。如果容器尚未結束，則會省略此參數。

`Limits`

在容器層級指定的資源限制，例如 CPU (以 CPU 單位表示) 和記憶體。如果未定義資源限制，則會省略此參數。

`CreatedAt`

容器的建立時間戳記。如果尚未建立容器，則會省略此參數。

`StartedAt`

容器的啟動時間戳記。如果尚未啟動容器，則會省略此參數。

`FinishedAt`

容器的停止時間戳記。如果尚未停止容器，則會省略此參數。

`Type`

容器的類型。任務定義中指定的容器類型為 `NORMAL`。您可以忽略其他容器類型，這些是 Amazon ECS 容器代理程式用來佈建內部任務資源的容器類型。

`LogDriver`

容器正在使用的日誌驅動程式。

> **Note**
>
> 只有在使用 Amazon ECS Linux 容器代理程式版本 `1.45.0` 或更新版本時，才包括此 `LogDriver` 中繼資料。

`LogOptions`

為容器定義的日誌驅動程式選項。

> **Note**
>
> 只有在使用 Amazon ECS Linux 容器代理程式版本 `1.45.0` 或更新版本時，才包括此 `LogOptions` 中繼資料。

`ContainerARN`

容器的完整 Amazon Resource Name (ARN)。

> **Note**
>
> 只有在使用 Amazon ECS Linux 容器代理程式版本 `1.45.0` 或更新版本時，才包括此 `ContainerARN` 中繼資料。

`Networks`

容器的網路資訊，例如網路模式和 IP 地址。如果未定義網路資訊，則會省略此參數。

`Snapshotter`

containerd 使用 snapshotter 來下載此容器映像。有效值為 `overlayfs` (為預設值) 和 `soci`，並且在使用 SOCI 索引延遲載入時使用。此參數僅適用於在 Linux 平台版本 `1.4.0` 上執行的任務。

`RestartCount`

容器重新啟動的次數。

> **Note**
>
> 只有在容器啟用重新啟動政策時，才會包含`RestartCount`中繼資料。如需詳細資訊，請參閱使用容器重新啟動政策重新啟動 Amazon ECS 任務中的個別容器。

`ClockDrift`

關於參考時間和系統時間之間差異的資訊。此功能使用 Amazon Time Sync Service 來測量時鐘準確性，並提供容器的時鐘錯誤界限。如需詳細資訊，請參閱《Amazon EC2 Linux 執行個體使用者指南》中的設定 Linux 執行個體的時間。 Amazon EC2

`ReferenceTime`

時鐘精確度的基礎。Amazon ECS 透過 NTP 使用世界協調時間 (UTC) 的全球標準，例如 `2021-09-07T16:57:44Z`。

`ClockErrorBound`

時鐘誤差的測量值，定義為 UTC 的偏移。此誤差是參考時間和系統時間之間的毫秒差異。

`ClockSynchronizationStatus`

指出系統時間和參考時間之間最近的同步嘗試是否成功。

有效值為 `SYNCHRONIZED` 和 `NOT_SYNCHRONIZED`。

`ExecutionStoppedAt`

任務 `DesiredStatus` 移至 `STOPPED` 時的時間戳記。這會在基本容器移至 `STOPPED` 時發生。

Fargate 上任務的 Amazon ECS 任務中繼資料 v4 範例

以下範例顯示來自在 AWS Fargate上執行之 Amazon ECS 任務的任務中繼資料端點範例輸出。

在容器中，您可以使用 curl 後跟任務中繼資料端點來查詢端點，例如 `curl ${ECS_CONTAINER_METADATA_URI_V4}/task`。

容器中繼資料回應範例

查詢 `${ECS_CONTAINER_METADATA_URI_V4}` 端點時，只會傳回關於容器本身的中繼資料。以下為範例輸出。

```
{
    "DockerId": "cd189a933e5849daa93386466019ab50-2495160603",
    "Name": "curl",
    "DockerName": "curl",
    "Image": "111122223333.dkr.ecr.us-west-2.amazonaws.com/curltest:latest",
    "ImageID":
 "sha256:25f3695bedfb454a50f12d127839a68ad3caf91e451c1da073db34c542c4d2cb",
    "Labels": {
        "com.amazonaws.ecs.cluster": "arn:aws:ecs:us-west-2:111122223333:cluster/
default",
        "com.amazonaws.ecs.container-name": "curl",
        "com.amazonaws.ecs.task-arn": "arn:aws:ecs:us-west-2:111122223333:task/default/
cd189a933e5849daa93386466019ab50",
        "com.amazonaws.ecs.task-definition-family": "curltest",
```

```
        "com.amazonaws.ecs.task-definition-version": "2"
    },
    "DesiredStatus": "RUNNING",
    "KnownStatus": "RUNNING",
    "Limits": {
        "CPU": 10,
        "Memory": 128
    },
    "CreatedAt": "2020-10-08T20:09:11.44527186Z",
    "StartedAt": "2020-10-08T20:09:11.44527186Z",
    "Type": "NORMAL",
    "Networks": [
        {
            "NetworkMode": "awsvpc",
            "IPv4Addresses": [
                "192.0.2.3"
            ],
            "AttachmentIndex": 0,
            "MACAddress": "0a:de:f6:10:51:e5",
            "IPv4SubnetCIDRBlock": "192.0.2.0/24",
            "DomainNameServers": [
                "192.0.2.2"
            ],
            "DomainNameSearchList": [
                "us-west-2.compute.internal"
            ],
            "PrivateDNSName": "ip-10-0-0-222.us-west-2.compute.internal",
            "SubnetGatewayIpv4Address": "192.0.2.0/24"
        }
    ],
    "ContainerARN": "arn:aws:ecs:us-west-2:111122223333:container/05966557-
f16c-49cb-9352-24b3a0dcd0e1",
    "LogOptions": {
        "awslogs-create-group": "true",
        "awslogs-group": "/ecs/containerlogs",
        "awslogs-region": "us-west-2",
        "awslogs-stream": "ecs/curl/cd189a933e5849daa93386466019ab50"
    },
    "LogDriver": "awslogs",
    "Snapshotter": "overlayfs"
}
```

Fargate 上任務的 Amazon ECS 任務中繼資料 v4 範例

查詢 ${ECS_CONTAINER_METADATA_URI_V4}/task 端點時，只會傳回容器所屬任務的中繼資料。以下為範例輸出。

```
{
  "Cluster": "arn:aws:ecs:us-east-1:123456789012:cluster/clusterName",
  "TaskARN": "arn:aws:ecs:us-east-1:123456789012:task/MyEmptyCluster/
bfa2636268144d039771334145e490c5",
  "Family": "sample-fargate",
  "Revision": "5",
  "DesiredStatus": "RUNNING",
  "KnownStatus": "RUNNING",
  "Limits": {
    "CPU": 0.25,
    "Memory": 512
  },
  "PullStartedAt": "2023-07-21T15:45:33.532811081Z",
  "PullStoppedAt": "2023-07-21T15:45:38.541068435Z",
  "AvailabilityZone": "us-east-1d",
  "Containers": [
    {
      "DockerId": "bfa2636268144d039771334145e490c5-1117626119",
      "Name": "curl-image",
      "DockerName": "curl-image",
      "Image": "curlimages/curl",
      "ImageID":
 "sha256:daf3f46a2639c1613b25e85c9ee4193af8a1d538f92483d67f9a3d7f21721827",
      "Labels": {
        "com.amazonaws.ecs.cluster": "arn:aws:ecs:us-east-1:123456789012:cluster/
MyEmptyCluster",
        "com.amazonaws.ecs.container-name": "curl-image",
        "com.amazonaws.ecs.task-arn": "arn:aws:ecs:us-east-1:123456789012:task/
MyEmptyCluster/bfa2636268144d039771334145e490c5",
        "com.amazonaws.ecs.task-definition-family": "sample-fargate",
        "com.amazonaws.ecs.task-definition-version": "5"
      },
      "DesiredStatus": "RUNNING",
      "KnownStatus": "RUNNING",
      "Limits": { "CPU": 128 },
      "CreatedAt": "2023-07-21T15:45:44.91368314Z",
      "StartedAt": "2023-07-21T15:45:44.91368314Z",
      "Type": "NORMAL",
      "Networks": [
```

```
        {
          "NetworkMode": "awsvpc",
          "IPv4Addresses": ["172.31.42.189"],
          "AttachmentIndex": 0,
          "MACAddress": "0e:98:9f:33:76:d3",
          "IPv4SubnetCIDRBlock": "172.31.32.0/20",
          "DomainNameServers": ["172.31.0.2"],
          "DomainNameSearchList": ["ec2.internal"],
          "PrivateDNSName": "ip-172-31-42-189.ec2.internal",
          "SubnetGatewayIpv4Address": "172.31.32.1/20"
        }
      ],
      "ContainerARN": "arn:aws:ecs:us-east-1:123456789012:container/MyEmptyCluster/
bfa2636268144d039771334145e490c5/da6cccf7-1178-400c-afdf-7536173ee209",
      "Snapshotter": "overlayfs"
    },
    {
      "DockerId": "bfa2636268144d039771334145e490c5-3681984407",
      "Name": "fargate-app",
      "DockerName": "fargate-app",
      "Image": "public.ecr.aws/docker/library/httpd:latest",
      "ImageID":
 "sha256:8059bdd0058510c03ae4c808de8c4fd2c1f3c1b6d9ea75487f1e5caa5ececa02",
      "Labels": {
        "com.amazonaws.ecs.cluster": "arn:aws:ecs:us-east-1:123456789012:cluster/
MyEmptyCluster",
        "com.amazonaws.ecs.container-name": "fargate-app",
        "com.amazonaws.ecs.task-arn": "arn:aws:ecs:us-east-1:123456789012:task/
MyEmptyCluster/bfa2636268144d039771334145e490c5",
        "com.amazonaws.ecs.task-definition-family": "sample-fargate",
        "com.amazonaws.ecs.task-definition-version": "5"
      },
      "DesiredStatus": "RUNNING",
      "KnownStatus": "RUNNING",
      "Limits": { "CPU": 2 },
      "CreatedAt": "2023-07-21T15:45:44.954460255Z",
      "StartedAt": "2023-07-21T15:45:44.954460255Z",
      "Type": "NORMAL",
      "Networks": [
        {
          "NetworkMode": "awsvpc",
          "IPv4Addresses": ["172.31.42.189"],
          "AttachmentIndex": 0,
          "MACAddress": "0e:98:9f:33:76:d3",
```

```
          "IPv4SubnetCIDRBlock": "172.31.32.0/20",
          "DomainNameServers": ["172.31.0.2"],
          "DomainNameSearchList": ["ec2.internal"],
          "PrivateDNSName": "ip-172-31-42-189.ec2.internal",
          "SubnetGatewayIpv4Address": "172.31.32.1/20"
        }
      ],
      "ContainerARN": "arn:aws:ecs:us-east-1:123456789012:container/MyEmptyCluster/
bfa2636268144d039771334145e490c5/f65b461d-aa09-4acb-a579-9785c0530cbc",
      "Snapshotter": "overlayfs"
    }
  ],
  "LaunchType": "FARGATE",
  "ClockDrift": {
    "ClockErrorBound": 0.446931,
    "ReferenceTimestamp": "2023-07-21T16:09:17Z",
    "ClockSynchronizationStatus": "SYNCHRONIZED"
  },
  "EphemeralStorageMetrics": {
    "Utilized": 261,
    "Reserved": 20496
  }
}
```

任務統計資訊回應範例

查詢 ${ECS_CONTAINER_METADATA_URI_V4}/task/stats 端點時，會傳回關於容器所屬任務的網路指標。以下為範例輸出。

```
{
  "3d1f891cded94dc795608466cce8ddcf-464223573": {
    "read": "2020-10-08T21:24:44.938937019Z",
    "preread": "2020-10-08T21:24:34.938633969Z",
    "pids_stats": {},
    "blkio_stats": {
      "io_service_bytes_recursive": [
        {
          "major": 202,
          "minor": 26368,
          "op": "Read",
          "value": 638976
        },
        {
```

```
          "major": 202,
          "minor": 26368,
          "op": "Write",
          "value": 0
        },
        {
          "major": 202,
          "minor": 26368,
          "op": "Sync",
          "value": 638976
        },
        {
          "major": 202,
          "minor": 26368,
          "op": "Async",
          "value": 0
        },
        {
          "major": 202,
          "minor": 26368,
          "op": "Total",
          "value": 638976
        }
      ],
      "io_serviced_recursive": [
        {
          "major": 202,
          "minor": 26368,
          "op": "Read",
          "value": 12
        },
        {
          "major": 202,
          "minor": 26368,
          "op": "Write",
          "value": 0
        },
        {
          "major": 202,
          "minor": 26368,
          "op": "Sync",
          "value": 12
        },
        {
```

```
          "major": 202,
          "minor": 26368,
          "op": "Async",
          "value": 0
        },
        {
          "major": 202,
          "minor": 26368,
          "op": "Total",
          "value": 12
        }
      ],
      "io_queue_recursive": [],
      "io_service_time_recursive": [],
      "io_wait_time_recursive": [],
      "io_merged_recursive": [],
      "io_time_recursive": [],
      "sectors_recursive": []
    },
    "num_procs": 0,
    "storage_stats": {},
    "cpu_stats": {
      "cpu_usage": {
        "total_usage": 1137691504,
        "percpu_usage": [
          696479228,
          441212276,
          0,
          0,
          0,
          0,
          0,
          0,
          0,
          0,
          0,
          0,
          0,
          0,
          0,
          0
        ],
        "usage_in_kernelmode": 80000000,
        "usage_in_usermode": 810000000
      },
```

```
      "system_cpu_usage": 9393210000000,
      "online_cpus": 2,
      "throttling_data": {
        "periods": 0,
        "throttled_periods": 0,
        "throttled_time": 0
      }
    },
    "precpu_stats": {
      "cpu_usage": {
        "total_usage": 1136624601,
        "percpu_usage": [
          695639662,
          440984939,
          0,
          0,
          0,
          0,
          0,
          0,
          0,
          0,
          0,
          0,
          0,
          0,
          0,
          0
        ],
        "usage_in_kernelmode": 80000000,
        "usage_in_usermode": 810000000
      },
      "system_cpu_usage": 9373330000000,
      "online_cpus": 2,
      "throttling_data": {
        "periods": 0,
        "throttled_periods": 0,
        "throttled_time": 0
      }
    },
    "memory_stats": {
      "usage": 6504448,
      "max_usage": 8458240,
      "stats": {
        "active_anon": 1675264,
```

```
        "active_file": 557056,
        "cache": 651264,
        "dirty": 0,
        "hierarchical_memory_limit": 536870912,
        "hierarchical_memsw_limit": 9223372036854772000,
        "inactive_anon": 0,
        "inactive_file": 3088384,
        "mapped_file": 430080,
        "pgfault": 11034,
        "pgmajfault": 5,
        "pgpgin": 8436,
        "pgpgout": 7137,
        "rss": 4669440,
        "rss_huge": 0,
        "total_active_anon": 1675264,
        "total_active_file": 557056,
        "total_cache": 651264,
        "total_dirty": 0,
        "total_inactive_anon": 0,
        "total_inactive_file": 3088384,
        "total_mapped_file": 430080,
        "total_pgfault": 11034,
        "total_pgmajfault": 5,
        "total_pgpgin": 8436,
        "total_pgpgout": 7137,
        "total_rss": 4669440,
        "total_rss_huge": 0,
        "total_unevictable": 0,
        "total_writeback": 0,
        "unevictable": 0,
        "writeback": 0
      },
      "limit": 9223372036854772000
    },
    "name": "curltest",
    "id": "3d1f891cded94dc795608466cce8ddcf-464223573",
    "networks": {
      "eth1": {
        "rx_bytes": 2398415937,
        "rx_packets": 1898631,
        "rx_errors": 0,
        "rx_dropped": 0,
        "tx_bytes": 1259037719,
        "tx_packets": 428002,
```

```
        "tx_errors": 0,
        "tx_dropped": 0
      }
    },
    "network_rate_stats": {
      "rx_bytes_per_sec": 43.298687872232854,
      "tx_bytes_per_sec": 215.39347269466413
    }
  }
}
```


Fargate 上任務的 Amazon ECS 任務中繼資料端點第 3 版

> **Important**
>
> 不再積極維護任務中繼資料第 3 版端點。建議您更新任務中繼資料第 4 版端點，以取得最新的中繼資料端點資訊。如需詳細資訊，請參閱the section called “Fargate 上任務的任務中繼資料端點第 4 版”。

從 Fargate 平台版本 `1.1.0` 開始，名為 `ECS_CONTAINER_METADATA_URI` 的環境變數會插入任務中的每個容器。當您查詢任務中繼資料第 3 版端點時，有各種任務中繼資料和 Docker 統計資訊可供任務使用。

對於託管於 Fargate 的 Amazon ECS 任務 (使用平台版本 `1.1.0` 或更新版本)，預設會啟用任務中繼資料端點功能。如需詳細資訊，請參閱適用於 Amazon ECS 的 Fargate 平台版本。

Fargate 上任務的任務中繼資料端點路徑

容器可使用下列 API 端點：

`${ECS_CONTAINER_METADATA_URI}`

此路徑傳回容器的 JSON 中繼資料。

`${ECS_CONTAINER_METADATA_URI}/task`

此路徑傳回任務的中繼資料 JSON，包括與任務相關聯之所有容器的容器 ID 和名稱清單。如需此端點之回應的詳細資訊，請參閱「Fargate 上任務的 Amazon ECS 任務中繼資料 v3 JSON 回應」。

`${ECS_CONTAINER_METADATA_URI}/stats`

此路徑傳回特定 Docker 容器的 Docker 統計資訊 JSON。如需每個所傳回統計資訊的詳細資訊，請參閱 Docker API 文件中的 ContainerStats。

`${ECS_CONTAINER_METADATA_URI}/task/stats`

此路徑傳回與任務相關聯之所有容器的 Docker 統計資訊 JSON。如需每個所傳回統計資訊的詳細資訊，請參閱 Docker API 文件中的 ContainerStats。

Fargate 上任務的 Amazon ECS 任務中繼資料 v3 JSON 回應

任務中繼資料端點 (`${ECS_CONTAINER_METADATA_URI}/task`) JSON 回應會傳回下列資訊。

`Cluster`

任務所屬 Amazon ECS 叢集的 Amazon Resource Name (ARN) 或簡短名稱。

`TaskARN`

容器所屬任務的完整 Amazon Resource Name (ARN)。

`Family`

任務的 Amazon ECS 任務定義系列。

`Revision`

任務的 Amazon ECS 任務定義修訂。

`DesiredStatus`

Amazon ECS 中任務的所需狀態。

`KnownStatus`

Amazon ECS 中任務的已知狀態。

`Limits`

在任務層級指定的資源限制，例如 CPU (以 vCPU 表示) 和記憶體。如果未定義資源限制，則會省略此參數。

`PullStartedAt`

第一個容器映像提取的開始時間戳記。

`PullStoppedAt`

最後一個容器映像提取的完成時間戳記。

`AvailabilityZone`

任務所在的可用區域。

> **Note**
>
> 可用區域中繼資料僅適用於使用平台第 1.4 版或更新版本 (Linux) 或 1.0.0 或更新版本 (Windows) 的 Fargate 任務。

`Containers`

與任務相關聯之每個容器的容器中繼資料清單。

`DockerId`

容器的 Docker ID。

`Name`

任務定義中指定的容器名稱。

`DockerName`

提供給 Docker 的容器名稱。Amazon ECS 容器代理程式會產生容器的唯一名稱，以避免在單一執行個體上執行相同任務定義的多個複本時，發生名稱衝突。

`Image`

容器的映像。

`ImageID`

映像的 SHA-256 摘要。

`Ports`

向容器開放的任何連接埠。如果未開放連接埠，則會省略此參數。

`Labels`

任何套用至容器的標籤。如果未套用標籤，則會省略此參數。

`DesiredStatus`

Amazon ECS 中容器的所需狀態。

`KnownStatus`

Amazon ECS 中容器的已知狀態。

`ExitCode`

容器的結束代碼。如果容器尚未結束，則會省略此參數。

`Limits`

在容器層級指定的資源限制，例如 CPU (以 CPU 單位表示) 和記憶體。如果未定義資源限制，則會省略此參數。

`CreatedAt`

容器的建立時間戳記。如果尚未建立容器，則會省略此參數。

`StartedAt`

容器的啟動時間戳記。如果尚未啟動容器，則會省略此參數。

`FinishedAt`

容器的停止時間戳記。如果尚未停止容器，則會省略此參數。

`Type`

容器的類型。任務定義中指定的容器類型為 `NORMAL`。您可以忽略其他容器類型，這些是 Amazon ECS 容器代理程式用來佈建內部任務資源的容器類型。

`Networks`

容器的網路資訊，例如網路模式和 IP 地址。如果未定義網路資訊，則會省略此參數。

`ClockDrift`

關於參考時間和系統時間之間差異的資訊。這適用於 Linux 作業系統。此功能使用 Amazon Time Sync Service 來測量時鐘準確性，並提供容器的時鐘錯誤界限。如需詳細資訊，請參閱《Amazon EC2 Linux 執行個體使用者指南》中的設定 Linux 執行個體的時間。

`ReferenceTime`

時鐘精確度的基礎。Amazon ECS 透過 NTP 使用世界協調時間 (UTC) 的全球標準，例如 `2021-09-07T16:57:44Z`。

ClockErrorBound

時鐘誤差的測量值，定義為 UTC 的偏移。此誤差是參考時間和系統時間之間的毫秒差異。

ClockSynchronizationStatus

指出系統時間和參考時間之間最近的同步嘗試是否成功。

有效值為 SYNCHRONIZED 和 NOT_SYNCHRONIZED。

ExecutionStoppedAt

任務 DesiredStatus 移至 STOPPED 時的時間戳記。這會在基本容器移至 STOPPED 時發生。

Fargate 上任務的 Amazon ECS 任務中繼資料 v3 範例

下列為單一容器任務的 JSON 回應。

```
{
  "Cluster": "default",
  "TaskARN": "arn:aws:ecs:us-east-2:012345678910:task/9781c248-0edd-4cdb-9a93-
f63cb662a5d3",
  "Family": "nginx",
  "Revision": "5",
  "DesiredStatus": "RUNNING",
  "KnownStatus": "RUNNING",
  "Containers": [
    {
      "DockerId": "731a0d6a3b4210e2448339bc7015aaa79bfe4fa256384f4102db86ef94cbbc4c",
      "Name": "~internal~ecs~pause",
      "DockerName": "ecs-nginx-5-internalecspause-acc699c0cbf2d6d11700",
      "Image": "amazon/amazon-ecs-pause:0.1.0",
      "ImageID": "",
      "Labels": {
        "com.amazonaws.ecs.cluster": "default",
        "com.amazonaws.ecs.container-name": "~internal~ecs~pause",
        "com.amazonaws.ecs.task-arn": "arn:aws:ecs:us-
east-2:012345678910:task/9781c248-0edd-4cdb-9a93-f63cb662a5d3",
        "com.amazonaws.ecs.task-definition-family": "nginx",
        "com.amazonaws.ecs.task-definition-version": "5"
      },
      "DesiredStatus": "RESOURCES_PROVISIONED",
      "KnownStatus": "RESOURCES_PROVISIONED",
      "Limits": {
        "CPU": 0,
```

```
        "Memory": 0
      },
      "CreatedAt": "2018-02-01T20:55:08.366329616Z",
      "StartedAt": "2018-02-01T20:55:09.058354915Z",
      "Type": "CNI_PAUSE",
      "Networks": [
        {
          "NetworkMode": "awsvpc",
          "IPv4Addresses": [
            "10.0.2.106"
          ]
        }
      ]
    },
    {
      "DockerId": "43481a6ce4842eec8fe72fc28500c6b52edcc0917f105b83379f88cac1ff3946",
      "Name": "nginx-curl",
      "DockerName": "ecs-nginx-5-nginx-curl-ccccb9f49db0dfe0d901",
      "Image": "nrdlngr/nginx-curl",
      "ImageID":
 "sha256:2e00ae64383cfc865ba0a2ba37f61b50a120d2d9378559dcd458dc0de47bc165",
      "Labels": {
        "com.amazonaws.ecs.cluster": "default",
        "com.amazonaws.ecs.container-name": "nginx-curl",
        "com.amazonaws.ecs.task-arn": "arn:aws:ecs:us-
east-2:012345678910:task/9781c248-0edd-4cdb-9a93-f63cb662a5d3",
        "com.amazonaws.ecs.task-definition-family": "nginx",
        "com.amazonaws.ecs.task-definition-version": "5"
      },
      "DesiredStatus": "RUNNING",
      "KnownStatus": "RUNNING",
      "Limits": {
        "CPU": 512,
        "Memory": 512
      },
      "CreatedAt": "2018-02-01T20:55:10.554941919Z",
      "StartedAt": "2018-02-01T20:55:11.064236631Z",
      "Type": "NORMAL",
      "Networks": [
        {
          "NetworkMode": "awsvpc",
          "IPv4Addresses": [
            "10.0.2.106"
          ]
```

```
        }
      ]
    }
  ],
  "PullStartedAt": "2018-02-01T20:55:09.372495529Z",
  "PullStoppedAt": "2018-02-01T20:55:10.552018345Z",
  "AvailabilityZone": "us-east-2b"
}
```

Amazon ECS 容器簡介

Amazon ECS 容器代理程式提供一種 API 操作，它收集執行代理程式的容器執行個體的詳細資訊，以及在該執行個體上執行之關聯任務的詳細資訊。您可以在容器執行個體內使用 curl 命令來查詢 Amazon ECS 容器代理程式 (連接埠 51678)，並傳回容器執行個體的中繼資料或任務資訊。

> **Important**
>
> 您的容器執行個體必須擁有允許存取 Amazon ECS 以擷取中繼資料的 IAM 角色。如需詳細資訊，請參閱Amazon ECS 容器執行個體 IAM 角色。

若要檢視容器執行個體中繼資料，請透過 SSH 登入您的容器執行個體，並執行以下命令。中繼資料包含容器執行個體 ID、註冊容器執行個體的 Amazon ECS 叢集，以及 Amazon ECS 容器代理程式版本資訊。

```
curl -s http://localhost:51678/v1/metadata | python3 -mjson.tool
```

輸出：

```
{
    "Cluster": "cluster_name",
    "ContainerInstanceArn": "arn:aws:ecs:region:aws_account_id:container-
instance/cluster_name/container_instance_id",
    "Version": "Amazon ECS Agent - v1.30.0 (02ff320c)"
}
```

若要檢視所有在容器執行個體上執行之任務的資訊，請透過 SSH 登入您的容器執行個體，然後執行以下命令：

```
curl http://localhost:51678/v1/tasks
```

輸出：

```
{
    "Tasks": [
        {
            "Arn": "arn:aws:ecs:us-west-2:012345678910:task/default/example5-58ff-46c9-
ae05-543f8example",
            "DesiredStatus": "RUNNING",
            "KnownStatus": "RUNNING",
            "Family": "hello_world",
            "Version": "8",
            "Containers": [
                {
          "DockerId":
 "9581a69a761a557fbfce1d0f6745e4af5b9dbfb86b6b2c5c4df156f1a5932ff1",
          "DockerName": "ecs-hello_world-8-mysql-fcae8ac8f9f1d89d8301",
          "Name": "mysql",
          "CreatedAt": "2023-10-08T20:09:11.44527186Z",
          "StartedAt": "2023-10-08T20:09:11.44527186Z",
          "ImageID":
 "sha256:2ae34abc2ed0a22e280d17e13f9c01aaf725688b09b7a1525d1a2750e2c0d1de"
        },
        {
          "DockerId":
 "bf25c5c5b2d4dba68846c7236e75b6915e1e778d31611e3c6a06831e39814a15",
          "DockerName": "ecs-hello_world-8-wordpress-e8bfddf9b488dff36c00",
          "Name": "wordpress"
        }
    ]
}
    ]
}
```

您可以檢視在該容器執行個體上執行之特定任務的資訊。若要指定特定任務或容器，請將以下其中一項附加到請求：

- 任務 ARN (?taskarn=*task_arn*)
- 容器的 Docker ID (?dockerid=*docker_id*)

若要使用容器的 Docker ID 取得任務資訊，請透過 SSH 登入您的容器執行個體，然後執行以下命令。

> **Note**
>
> 1.14.2 版之前的 Amazon ECS 容器代理程式需要完整的 Docker 容器 ID 以用於自我檢查 API，而非 docker ps 顯示的簡短版本。您可以藉由在容器執行個體上執行 docker ps --no-trunc 命令，取得容器完整的 Docker ID。

```
curl http://localhost:51678/v1/tasks?dockerid=79c796ed2a7f
```

輸出：

```
{
    "Arn": "arn:aws:ecs:us-west-2:012345678910:task/default/e01d58a8-151b-40e8-
bc01-22647b9ecfec",
    "Containers": [
        {
            "DockerId":
 "79c796ed2a7f864f485c76f83f3165488097279d296a7c05bd5201a1c69b2920",
            "DockerName": "ecs-nginx-efs-2-nginx-9ac0808dd0afa495f001",
            "Name": "nginx",
            "CreatedAt": "2023-10-08T20:09:11.44527186Z",
            "StartedAt": "2023-10-08T20:09:11.44527186Z",
            "ImageID":
 "sha256:2ae34abc2ed0a22e280d17e13f9c01aaf725688b09b7a1525d1a2750e2c0d1de"
        }
    ],
    "DesiredStatus": "RUNNING",
    "Family": "nginx-efs",
    "KnownStatus": "RUNNING",
    "Version": "2"
}
```

使用執行期監控識別未經授權的行為

Amazon GuardDuty 是一種威脅偵測服務，可協助保護您的帳戶、容器、工作負載和 AWS 環境中的資料。GuardDuty 使用機器學習 (ML) 模型，以及異常和威脅偵測功能，持續監控不同的日誌來源和執行時間活動，以識別環境中的潛在安全風險和惡意活動，並排定其優先順序。

GuardDuty 中的執行期監控透過持續監控 AWS 日誌和聯網活動來識別惡意或未經授權的行為，來保護 Fargate 和 EC2 容器執行個體上執行的工作負載。執行期監控使用輕量且全受管的 GuardDuty 安全代理程式，可分析主機上行為，例如檔案存取、程序執行和網路連線。這涵蓋的問題包括權限升級、使用公開的登入資料，或與惡意 IP 地址、網域的通訊，以及 Amazon EC2 執行個體和容器工作負載上存在惡意軟體。如需詳細資訊，請參閱《GuardDuty 使用者指南》中的 GuardDuty 執行期監控。 GuardDuty

您的安全管理員會針對 AWS Organizations GuardDuty 中的單一或多個帳戶啟用執行期監控。他們也會選取當您使用 Fargate 時，GuardDuty 是否自動部署 GuardDuty 安全代理程式。所有叢集都會自動受到保護，GuardDuty 會代表您管理安全代理程式。

您也可以在下列情況下手動設定 GuardDuty 安全代理程式：

- 您可以使用 EC2 容器執行個體
- 您需要精細的控制，才能在叢集層級啟用執行期監控

若要使用執行期監控，您必須設定受保護的叢集，並在 EC2 容器執行個體上安裝和管理 GuardDuty 安全代理程式。

執行期監控如何與 Amazon ECS 搭配使用

執行期監控使用輕量型 GuardDuty 安全代理程式，可監控應用程式請求、存取和使用基礎系統資源的方式。

對於 Fargate 任務，GuardDuty 安全代理程式會做為每個任務的附屬容器執行。

對於 EC2 容器執行個體，GuardDuty 安全代理程式會在執行個體上執行程序。

GuardDuty 安全代理程式會從下列資源收集資料，然後將資料傳送至 GuardDuty 進行處理。您可以在 GuardDuty 主控台中檢視調查結果。您也可以將它們傳送到其他 AWS Security Hub， AWS 服務 例如或第三方安全廠商，以進行彙總和修復。如需有關如何檢視和管理問題清單的資訊，請參閱《Amazon GuardDuty 使用者指南》中的管理 Amazon GuardDuty 問題清單。 Amazon GuardDuty

- 來自下列 Amazon ECS API 呼叫的回應：
  - DescribeClusters

    當您使用 `--include TAGS`選項時，回應參數包含執行期監控標籤 （設定標籤時）。
  - DescribeTasks

對於 Fargate 啟動類型，回應參數包含 GuardDuty 附屬容器。

- ListAccountSettings

  回應參數包含執行期監控帳戶設定，由您的安全管理員設定。

- 容器代理程式簡介資料。如需詳細資訊，請參閱Amazon ECS 容器簡介。
- 啟動類型的任務中繼資料端點：
  - Amazon ECS 任務中繼資料端點第 4 版
  - Fargate 上任務的 Amazon ECS 任務中繼資料端點第 4 版

考量事項

使用執行期監控時，請考慮下列事項：

- 執行期監控具有與其相關聯的成本。如需詳細資訊，請參閱 Amazon GuardDuty 定價。
- Amazon ECS Anywhere 不支援執行期監控。
- Windows 作業系統不支援執行期監控。
- 當您在 Fargate 上使用 Amazon ECS Exec 時，您必須指定容器名稱，因為 GuardDuty 安全代理程式會以附屬容器的形式執行。
- 您無法在 GuardDuty 安全代理程式附屬容器上使用 Amazon ECS Exec。
- 在叢集層級控制執行期監控的 IAM 使用者必須具有適當的 IAM 許可，才能進行標記。如需詳細資訊，請參閱《IAM 使用者指南》中的 IAM 教學課程：定義根據標籤存取 AWS 資源的許可。
- Fargate 任務必須使用任務執行角色。此角色會授予任務許可，以代表您擷取、更新和管理存放在 Amazon ECR 私有儲存庫中的 GuardDuty 安全代理程式。

資源使用率

您新增至叢集的標籤會計入叢集標籤配額。

GuardDuty 代理程式附屬容器不會計入每個任務定義配額的容器。

與大多數安全軟體一樣，GuardDuty 有些微的額外負荷。如需有關 Fargate 記憶體限制的資訊，請參閱《GuardDuty 使用者指南》中的 CPU 和記憶體限制。如需 Amazon EC2 記憶體限制的資訊，請參閱 GuardDuty 代理程式的 CPU 和記憶體限制。

Amazon ECS Fargate 工作負載的執行期監控

如果您使用 EC2 容器執行個體，則必須手動設定執行期監控。如需詳細資訊，請參閱Amazon ECS 上 EC2 工作負載的執行期監控。

您可以讓 GuardDuty 管理容器執行個體上的安全代理程式。此選項僅適用於 Fargate。此選項 (GuardDuty 代理程式管理) 可在 GuardDuty 中使用

當您使用 GuardDuty 代理程式管理時，GuardDuty 會執行下列操作：

- 為託管叢集的每個 VPC 建立 GuardDuty 的 VPC 端點。
- 擷取並安裝最新的 GuardDuty 安全代理程式，做為所有新獨立 Fargate 任務和新服務部署上的附屬容器。

  新服務部署會在您第一次啟動 服務，或您使用強制新部署選項更新現有服務時發生。

開啟 Amazon ECS 的執行期監控

您可以設定 GuardDuty 來自動管理所有 Fargate 叢集的安全 Agen。

必要條件

以下是使用執行期監控的先決條件：

- Linux 的 Fargate 平台版本必須為 `1.4.0` 或更新版本。
- Amazon ECS 的 IAM 角色和許可：
  - Fargate 任務必須使用任務執行角色。此角色會授予任務許可，以代表您擷取、更新和管理 GuardDuty 安全代理程式。如需詳細資訊，請參閱 Amazon ECS 任務執行 IAM 角色。
  - 您可以使用預先定義的標籤來控制叢集的執行期監控。如果您的存取政策根據標籤限制存取，您必須將明確許可授予 IAM 使用者以標記叢集。如需詳細資訊，請參閱《IAM 使用者指南》中的 IAM 教學課程：定義根據標籤存取 AWS 資源的許可。
- 連線至 Amazon ECR 儲存庫：

  GuardDuty 安全代理程式存放在 Amazon ECR 儲存庫中。每個獨立 和服務任務都必須具有儲存庫的存取權。您可以使用下列其中一個選項：
  - 對於公有子網路中的任務，您可以使用任務的公有 IP 地址，或在任務執行的子網路中為 Amazon ECR 建立 VPC 端點。如需詳細資訊，請參閱 《Amazon Elastic Container Registry 使用者指南》中的 Amazon ECR 介面 VPC 端點 (AWS PrivateLink)。

- 對於私有子網路中的任務，您可以使用網路地址轉譯 (NAT) 閘道，或在任務執行的子網路中建立 Amazon ECR 的 VPC 端點。

  如需詳細資訊，請參閱私有子網路和 NAT 閘道。

- 您必須具有 GuardDuty `AWSServiceRoleForAmazonGuardDuty`的角色。如需詳細資訊，請參閱《Amazon GuardDuty 使用者指南》中的 GuardDuty 的服務連結角色許可。 Amazon GuardDuty
- 您希望透過執行期監控保護的任何檔案都必須由根使用者存取。如果您手動變更檔案的許可，則必須將其設定為 `755`。

以下是在 EC2 容器執行個體上使用執行期監控的先決條件：

- 您必須使用 或更新版本`20230929`的 Amazon ECS-AMI。
- 您必須在容器執行個體上執行 Amazon ECS 代理程式至版本 `1.77`或更新版本。
- 您必須使用核心版本 `5.10`或更新版本。
- 如需有關支援的 Linux 作業系統和架構的資訊，請參閱 GuardDuty Runtime Monitoring 支援哪些操作模型和工作負載。
- 您可以使用 Systems Manager 來管理您的容器執行個體。如需詳細資訊，請參閱AWS Systems Manager Session Manager 《 使用者指南》中的為 EC2 執行個體設定 Systems Manager。

程序

您可以在 GuardDuty 中啟用執行期監控。如需如何啟用此功能的資訊，請參閱《Amazon GuardDuty 使用者指南》中的啟用執行期監控。

將執行期監控新增至現有的 Amazon ECS Fargate 任務

當您開啟執行期監控時，叢集中的所有新獨立任務和新服務部署都會自動受到保護。為了保留不可變性限制，現有的任務不會受到影響。

必要條件

1. 開啟執行期監控。如需詳細資訊，請參閱開啟 Amazon ECS 的執行期監控。
2. Fargate 任務必須使用任務執行角色。此角色會授予任務許可，以代表您擷取、更新和管理 GuardDuty 安全代理程式。如需詳細資訊，請參閱 Amazon ECS 任務執行 IAM 角色。

程序

- 若要立即保護任務，您需要執行下列其中一個動作：
  - 對於獨立任務，請停止任務，然後啟動它們。如需詳細資訊，請參閱 停止 Amazon ECS 任務 和 以 Amazon ECS 任務執行應用程式
  - 對於屬於 服務的任務，請使用「強制新部署」選項來更新服務。如需詳細資訊，請參閱使用主控台更新 Amazon ECS 服務。

從 Amazon ECS 叢集移除執行期監控

您可能想要將某些叢集排除在保護之外，例如用於測試的叢集。這會導致 GuardDuty 對叢集中的資源執行下列操作：

- 不再將 GuardDuty 安全代理程式部署到新的獨立 Fargate 任務或新的服務部署。

  為了保留不可變性限制，已啟用執行期監控的現有任務和部署不會受到影響。
- 停止計費，不再接受任務的執行時間事件。

程序

執行下列步驟，從叢集移除執行期監控。

1. 使用 Amazon ECS 主控台或 將叢集上的`GuardDutyManaged`標籤金鑰 AWS CLI 設定為 `false`。如需詳細資訊，請參閱使用 CLI 或 API 更新叢集或處理標籤。 https://docs.aws.amazon.com/AmazonECS/latest/developerguide/ecs-using-tags.html#tag-resources-api-sdk針對標籤使用下列值。

   > **Note**
   >
   > 索引鍵和值區分大小寫，且必須與字串完全相符。

   索引鍵 = `GuardDutyManaged`，值 = `false`
2. 刪除叢集的 GuardDuty VPC 端點。如需如何刪除 VPC 端點的詳細資訊，請參閱AWS PrivateLink《 使用者指南》中的刪除介面端點。

從 帳戶移除 Amazon ECS 的執行期監控

當您不想再使用執行期監控時，請停用 GuardDuty 中的功能。如需如何停用此功能的資訊，請參閱《Amazon GuardDuty 使用者指南》中的啟用執行期監控。

GuardDuty 會執行下列操作：

- 刪除託管叢集之每個 VPC 的 GuardDuty VPC 端點。
- 不再將 GuardDuty 安全代理程式部署到新的獨立 Fargate 任務或新的服務部署。

  為了保留不可變性限制，現有的任務和部署在停止、複寫或擴展之前不會受到影響。
- 停止計費，不再接受任務的執行時間事件。

Amazon ECS 上 EC2 工作負載的執行期監控

當您針對容量使用 EC2 執行個體，或當您需要在 Fargate 上的叢集層級精細控制執行期監控時，請使用此選項。

您可以新增預先定義的標籤，為執行期監控佈建叢集。

對於 EC2 容器執行個體，您可以下載、安裝和管理 GuardDuty 安全代理程式。

對於 Fargate，GuardDuty 會代表您管理安全代理程式。

開啟 Amazon ECS 的執行期監控

您可以為具有 EC2 執行個體的叢集開啟執行期監控，或當您需要在 Fargate 上的叢集層級精細控制執行期監控時。

以下是使用執行期監控的先決條件：

- Linux 的 Fargate 平台版本必須為 `1.4.0` 或更新版本。
- Amazon ECS 的 IAM 角色和許可：
  - Fargate 任務必須使用任務執行角色。此角色會授予任務許可，以代表您擷取、更新和管理 GuardDuty 安全代理程式。如需詳細資訊，請參閱 Amazon ECS 任務執行 IAM 角色。
  - 您可以使用預先定義的標籤來控制叢集的執行期監控。如果您的存取政策根據標籤限制存取，您必須將明確許可授予 IAM 使用者以標記叢集。如需詳細資訊，請參閱《IAM 使用者指南》中的 IAM 教學課程：定義根據標籤存取 AWS 資源的許可。

- 連線至 Amazon ECR 儲存庫：

  GuardDuty 安全代理程式存放在 Amazon ECR 儲存庫中。每個獨立 和服務任務都必須具有儲存庫的存取權。您可以使用下列其中一個選項：

  - 對於公有子網路中的任務，您可以使用任務的公有 IP 地址，或在任務執行的子網路中為 Amazon ECR 建立 VPC 端點。如需詳細資訊，請參閱 《Amazon Elastic Container Registry 使用者指南》中的 Amazon ECR 介面 VPC 端點 (AWS PrivateLink)。
  - 對於私有子網路中的任務，您可以使用網路地址轉譯 (NAT) 閘道，或在任務執行的子網路中建立 Amazon ECR 的 VPC 端點。

    如需詳細資訊，請參閱私有子網路和 NAT 閘道。
- 您必須具有 GuardDuty `AWSServiceRoleForAmazonGuardDuty`的角色。如需詳細資訊，請參閱《Amazon GuardDuty 使用者指南》中的 GuardDuty 的服務連結角色許可。 Amazon GuardDuty
- 您希望透過執行期監控保護的任何檔案都必須由根使用者存取。如果您手動變更檔案的許可，則必須將其設定為 `755`。

以下是在 EC2 容器執行個體上使用執行期監控的先決條件：

- 您必須使用 或更新版本`20230929`的 Amazon ECS-AMI。
- 您必須在容器執行個體上執行 Amazon ECS 代理程式至版本 `1.77`或更新版本。
- 您必須使用核心版本 `5.10`或更新版本。
- 如需有關支援的 Linux 作業系統和架構的資訊，請參閱 GuardDuty Runtime Monitoring 支援哪些操作模型和工作負載。
- 您可以使用 Systems Manager 來管理您的容器執行個體。如需詳細資訊，請參閱AWS Systems Manager Session Manager 《 使用者指南》中的為 EC2 執行個體設定 Systems Manager。

您可以在 GuardDuty 中開啟執行期監控。如需如何啟用功能的資訊，請參閱《Amazon GuardDuty 使用者指南》中的啟用執行期監控。

新增執行期監控 Amazon ECS 叢集

設定叢集的執行期監控，然後在 EC2 容器執行個體上安裝 GuardDuty 安全代理程式。

必要條件

1. 開啟執行期監控。如需詳細資訊，請參閱開啟 Amazon ECS 的執行期監控。

2. 您可以使用預先定義的標籤來控制叢集的執行期監控。如果您的存取政策根據標籤限制存取，您必須將明確許可授予 IAM 使用者以標記叢集。如需詳細資訊，請參閱《IAM 使用者指南》中的 IAM 教學課程：定義根據標籤存取 AWS 資源的許可。

程序

執行下列操作，將執行期監控新增至叢集。

1. 為每個叢集 VPC 建立 GuardDuty 的 VPC 端點。如需詳細資訊，請參閱《GuardDuty 使用者指南》中的手動建立 Amazon VPC 端點。
2. 設定 EC2 容器執行個體。
   a. 在叢集中的 EC2 容器執行個體上，將 Amazon ECS 代理程式更新為版本 `1.77`或更新版本。如需詳細資訊，請參閱 更新 Amazon ECS 容器代理程式。
   b. 在叢集的 EC2 容器執行個體上安裝 GuardDuty 安全代理程式。如需詳細資訊，請參閱《GuardDuty 使用者指南》中的在 Amazon EC2 執行個體上手動管理安全代理程式。

      由於 GuardDuty 安全代理程式會在 EC2 容器執行個體上執行程序，因此所有新的和現有的任務和部署都會立即受到保護。
3. 使用 Amazon ECS 主控台或 將叢集上的`GuardDutyManaged`標籤金鑰 AWS CLI 設定為 `true`。如需詳細資訊，請參閱使用 CLI 或 API 更新叢集或處理標籤。 https://docs.aws.amazon.com/AmazonECS/latest/developerguide/ecs-using-tags.html#tag-resources-api-sdk針對標籤使用下列值。

   > **Note**
   >
   > 索引鍵和值區分大小寫，且必須與字串完全相符。

   索引鍵 = `GuardDutyManaged`，值 = `true`

將執行期監控新增至現有的 Amazon ECS 任務

當您開啟執行期監控時，叢集中的所有新獨立任務和新服務部署都會自動受到保護。為了保留不可變性限制，現有的任務不會受到影響。

必要條件

- 開啟執行期監控。如需詳細資訊，請參閱開啟 Amazon ECS 的執行期監控。

程序

- 若要立即保護任務，您需要執行下列其中一個動作：
  - 對於獨立任務，請停止任務，然後啟動它們。如需詳細資訊，請參閱 停止 Amazon ECS 任務 和 以 Amazon ECS 任務執行應用程式。
  - 對於屬於 服務的任務，請使用「強制新部署」選項來更新服務。如需詳細資訊，請參閱使用主控台更新 Amazon ECS 服務。

從 Amazon ECS 叢集移除執行期監控

您可以從叢集移除執行期監控。這會導致 GuardDuty 停止監控叢集中的所有資源。

從叢集移除執行期監控

1. 使用 Amazon ECS 主控台或 將叢集上的`GuardDutyManaged`標籤金鑰 AWS CLI 設定為 `false`。如需詳細資訊，請參閱使用 CLI 或 API 更新叢集或處理標籤。 https://docs.aws.amazon.com/AmazonECS/latest/developerguide/ecs-using-tags.html#tag-resources-api-sdk

   > **Note**
   >
   > 索引鍵和值區分大小寫，且必須與字串完全相符。

   索引鍵 = `GuardDutyManaged`，值 = `false`
2. 在叢集的 EC2 容器執行個體上解除安裝 GuardDuty 安全代理程式。

   如需詳細資訊，請參閱《GuardDuty 使用者指南》中的手動解除安裝安全代理程式。
3. 刪除每個叢集 VPC 的 GuardDuty VPC 端點。如需如何刪除 VPC 端點的詳細資訊，請參閱AWS PrivateLink 《 使用者指南》中的刪除介面端點。

更新 Amazon ECS 容器執行個體上的 GuardDuty 安全代理程式

如需有關如何在 EC2 容器執行個體上更新 GuardDuty 安全代理程式的資訊，請參閱《Amazon GuardDuty 使用者指南》中的更新 GuardDuty 安全代理程式。 Amazon GuardDuty

從 帳戶移除 Amazon ECS 的執行期監控

當您不想再使用執行期監控時，請停用 GuardDuty 中的功能。如需有關如何停用此功能的資訊，請參閱《Amazon GuardDuty 使用者指南》中的啟用執行期監控。

從所有叢集移除執行期監控。如需詳細資訊，請參閱從 Amazon ECS 叢集移除執行期監控。

執行時間監控故障診斷FAQs

您可能需要對任務和容器上的執行時間監控已啟用和執行，進行故障診斷或驗證。

主題

- 如何判斷我的帳戶是否已啟用執行期監控？
- 如何判斷叢集上的執行期監控是否處於作用中狀態？
- 如何判斷 GuardDuty 安全代理程式是否正在 Fargate 任務上執行？
- 如何判斷 GuardDuty 安全代理程式是否在 EC2 容器執行個體上執行？
- 如果叢集上執行的任務沒有任務執行角色，會發生什麼情況？
- 如何判斷我是否具有標記叢集以進行執行期監控的正確許可？
- 沒有連線 Amazon ECR 時會發生什麼情況？
- 啟用執行期監控後，如何解決 Fargate 任務上的記憶體不足錯誤？

如何判斷我的帳戶是否已啟用執行期監控？

在 Amazon ECS 主控台中，資訊位於帳戶設定頁面上。

您也可以`list-account-settings`使用 `effective-settings`選項執行 。

```
aws ecs list-account-settings --effective-settings
```

輸出

名稱設為 `guardDutyActivate`且值設為 的 設定`on`表示帳戶已設定。您必須向您的 GuardDuty 管理員確認管理是自動還是手動。

```
{
    "setting": {
        "name": "guardDutyActivate",
        "value": "enabled",
        "principalArn": "arn:aws:iam::123456789012:root",
        "type": "aws-managed"
    }
}
```

如何判斷叢集上的執行期監控是否處於作用中狀態？

在 Amazon ECS 主控台中，資訊位於叢集詳細資訊頁面的標籤索引標籤上。

您也可以`describe-clusters`使用 `TAGS`選項執行 。

下列範例顯示預設叢集的輸出

```
aws ecs describe-clusters --cluster default --include TAGS
```

輸出

金鑰設為 `GuardDutyManaged`且值設為 的標籤`true`表示叢集已設定為執行期監控。

```
{
    "clusters": [
        {
            "clusterArn": "arn:aws:ecs:us-east-1:1234567890:cluster/default",
            "clusterName": "default",
            "status": "ACTIVE",
            "registeredContainerInstancesCount": 0,
            "runningTasksCount": 1,
            "pendingTasksCount": 0,
            "activeServicesCount": 0,
            "statistics": [],
            "tags": [
                {
                    "key": "GuardDutyManaged",
                    "value": "true"
                }
            ],
            "settings": [],
            "capacityProviders": [],
```

```
            "defaultCapacityProviderStrategy": []
        }
    ],
    "failures": []
}
```

如何判斷 GuardDuty 安全代理程式是否正在 Fargate 任務上執行？

GuardDuty 安全代理程式會做為 Fargate 任務的附屬容器執行。

在 Amazon ECS 主控台中，附屬項目會顯示在任務詳細資訊頁面上的容器下方。

您可以執行`describe-tasks`並尋找名稱設為 `aws-gd-agent`且 lastStatus 設為 的容器`RUNNING`。

下列範例顯示任務 的預設叢集輸出`aws:ecs:us-east-1:123456789012:task/0b69d5c0-d655-4695-98cd-5d2d5EXAMPLE`。

```
aws ecs describe-tasks --cluster default --tasks aws:ecs:us-
east-1:123456789012:task/0b69d5c0-d655-4695-98cd-5d2d5EXAMPLE
```

輸出

名為 的容器`gd-agent`處於 RUNNING 狀態。

```
"containers": [
      {
        "containerArn": "arn:aws:ecs:us-east-1:123456789012:container/4df26bb4-
f057-467b-a079-96167EXAMPLE",
        "taskArn": "arn:aws:ecs:us-east-1:123456789012:task/0b69d5c0-
d655-4695-98cd-5d2d5EXAMPLE",
        "lastStatus": "RUNNING",
        "healthStatus": "UNKNOWN",
        "memory": "string",
        "name": "aws-gd-agent"
      }
    ]
```

如何判斷 GuardDuty 安全代理程式是否在 EC2 容器執行個體上執行？

執行下列命令以檢視狀態：

```
sudo systemctl status amazon-guardduty-agent
```

日誌檔案位於下列位置：

```
/var/log/amzn-guardduty-agent
```

如果叢集上執行的任務沒有任務執行角色，會發生什麼情況？

對於 Fargate 任務，任務在沒有 GuardDuty 安全代理程式附屬容器的情況下啟動。GuardDuty 儀表板會在涵蓋範圍統計資料儀表板中顯示任務缺少保護。

如何判斷我是否具有標記叢集以進行執行期監控的正確許可？

若要標記叢集，您必須同時擁有 `CreateCluster`和 `ecs:TagResource`的動作`UpdateCluster`。

以下是範例政策的程式碼片段。

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
         "ecs:TagResource"
      ],
      "Resource": "*",
      "Condition": {
         "StringEquals": {
             "ecs:CreateAction" : "CreateCluster",
             "ecs:CreateAction" : "UpdateCluster",
          }
       }
    }
  ]
}
```

沒有連線 Amazon ECR 時會發生什麼情況？

對於 Fargate 任務，任務在沒有 GuardDuty 安全代理程式附屬容器的情況下啟動。GuardDuty 儀表板會在涵蓋範圍統計資料儀表板中顯示任務缺少保護。

啟用執行期監控後，如何解決 Fargate 任務上的記憶體不足錯誤？

GuardDuty 安全代理程式是一個輕量化程序。不過，程序仍會根據工作負載的大小耗用資源。我們建議您使用容器資源追蹤工具，例如 Amazon CloudWatch Container Insights，在叢集中暫存

GuardDuty 部署。這些工具可協助您探索應用程式 GuardDuty 安全代理程式的耗用設定檔。然後，您可以視需要調整 Fargate 任務大小，以避免潛在的記憶體不足情況。

使用 ECS Exec 監控 Amazon ECS 容器

使用 Amazon ECS Exec，您可以直接與容器互動，而無需先與主機容器作業系統互動、開啟傳入連接埠或管理 SSH 金鑰。您可以使用 ECS Exec 執行命令，或為在 Amazon EC2 執行個體或 AWS Fargate上執行的容器取得 shell。如此一來就可以更輕鬆地收集診斷資訊並快速對錯誤進行故障診斷。例如，在開發環境中，您可以使用 ECS Exec 輕鬆地與容器中的各種程序互動，並對應用程式進行故障診斷。在生產案例中，您可以使用它來取得容器的碎片存取，以偵錯問題。

您可以從 Amazon ECS API、 AWS Command Line Interface (AWS CLI)、 AWS SDKs或 AWS Copilot CLI 使用 ECS Exec，在執行中的 Linux 或 Windows 容器中執行命令。如需使用 ECS Exec 的詳細資訊，以及使用 AWS Copilot CLI 的影片逐步解說，請參閱 Copilot GitHub 文件。

您也可以使用 ECS Exec 來維護更嚴格的存取控制政策。透過選擇性地開啟此功能，您可以控制執行命令的人員，以及他們可以執行這些命令的任務。透過每個命令及其輸出的日誌，您可以使用 ECS Exec 檢視已執行的任務，並使用 CloudTrail 來稽核存取容器的人員。

考量事項

在本主題中，您應該熟悉下列使用 ECS Exec 的相關方面：

- 目前不支援使用 ECS Exec AWS Management Console。

  叢集詳細資訊頁面會顯示用於加密本機用戶端和容器之間資料的記錄和 AWS KMS 客戶受管金鑰。
- 在 Systems Manager 不支援的作業系統上執行時，ECS Exec 可能無法如預期般運作。如需有關支援的作業系統的資訊，請參閱AWS Systems Manager 《 使用者指南》中的作業系統類型。
- 對於在下列基礎設施上執行的任務支援 ECS Exec：
  - 任何 Amazon ECS 最佳化 AMI 中 Amazon EC2 上的 Linux 容器，包括 Bottlerocket
  - 外部執行個體上的 Linux 和 Windows 容器 (Amazon ECS Anywhere)
  - AWS Fargate 上的 Linux 和 Windows 容器
  - 下列 Windows Amazon ECS 最佳化 AMI 中 Amazon EC2 上的 Windows 容器 (使用容器代理程式版本 `1.56` 或更新版本)：
    - Amazon ECS 最佳化 Windows Server 2022 Full AMI
    - Amazon ECS 最佳化 Windows Server 2022 Core AMI

  - Amazon ECS 最佳化 Windows Server 2019 Full AMI
  - Amazon ECS 最佳化 Windows Server 2019 Core AMI
  - Amazon ECS 最佳化 Windows Server 20H2 Core AMI
- 如果您為任務設定 HTTP 代理，請將`NO_PROXY`環境變數設定為 `"NO_PROXY=169.254.169.254,169.254.170.2"` ，以略過 EC2 執行個體中繼資料和 IAM 角色流量的代理。如果您未設定`NO_PROXY`環境變數，則從容器內的中繼資料端點擷取執行個體中繼資料或 IAM 角色登入資料時，可能會發生失敗。將`NO_PROXY`環境變數設定為建議值會篩選中繼資料和 IAM 流量，讓 請求`169.254.169.254 and 169.254.170.2`不會通過HTTP代理。
- ECS Exec 和 Amazon VPC
  - 如果您將界面 Amazon VPC 端點與 Amazon ECS 搭配使用，則必須為 Systems Manager 工作階段管理工具 (`ssmmessages`) 建立界面 Amazon VPC 端點。如需 Systems Manager VPC 端點的詳細資訊，請參閱AWS Systems Manager 《 使用者指南》中的使用 AWS PrivateLink 為 Session Manager 設定 VPC 端點。
  - 如果您將界面 Amazon VPC 端點與 Amazon ECS 搭配使用，並將 AWS KMS key 用於加密，則必須為 AWS KMS key建立界面 Amazon VPC 端點。如需詳細資訊，請參閱《AWS Key Management Service 開發人員指南》中的透過 VPC 端點連線至 AWS KMS key。
  - 當您有任務在 Amazon EC2 執行個體上執行時，請使用`awsvpc`聯網模式。如果您沒有網際網路存取，例如未設定為使用 NAT 閘道），則必須為 Systems Manager Session Manager () 建立界面 Amazon VPC 端點`ssmmessages`。如需有關 `awsvpc` 網路模式考量事項的詳細資訊，請參閱考量事項。如需 Systems Manager VPC 端點的詳細資訊，請參閱AWS Systems Manager 《 使用者指南》中的使用 AWS PrivateLink 為 Session Manager 設定 VPC 端點。
- ECS Exec 和 SSM
  - 當使用者使用 ECS Exec 在容器上執行命令時，這些命令會以 `root` 使用者執行。即使您指定容器的使用者 ID，SSM 代理程式及其子處理程序也會以根身分執行。
  - SSM 代理程式需要將容器檔案系統寫入 ，才能建立所需的目錄和檔案。因此，不支援使用 `readonlyRootFilesystem` 任務定義參數或任何其他方法將根檔案系統設為唯讀。
  - 可以啟動 `execute-command` 動作以外的 SSM 工作階段時，這會導致工作階段未被記錄，會計入工作階段限制。建議您限制此存取，方法是使用 IAM 政策拒絕 `ssm:start-session` 動作。如需詳細資訊，請參閱限制存取「啟動工作階段」動作。
- 下列功能會以附屬容器的形式執行。因此，您必須指定要執行命令的容器名稱。
  - 執行期監控
  - Service Connect

- 使用者可以執行容器內容中所有可用的命令。下列動作可能會導致孤立和廢止程序：終止容器的主要程序、終止命令代理程式，以及刪除相依性。為了清理廢止程序，建議將 `initProcessEnabled` 旗標新增至您的任務定義。
- ECS Exec 使用一些 CPU 和記憶體。在任務定義中指定 CPU 和記憶體資源配置時，您需要適應這一點。
- 您必須使用 AWS CLI 版本 `1.22.3`或更新版本，或是 AWS CLI 版本 `2.3.6` 或更新版本。如需有關如何更新 的資訊 AWS CLI，請參閱《 使用者指南第 2 版》中的安裝或更新最新版本 AWS CLI的 。AWS Command Line Interface
- 每個程序 ID (PID) 命名空間僅能有一個 ECS Exec 工作階段。如果您在任務中共享 PID 命名空間，則僅能在一個容器中啟動 ECS Exec 工作階段。
- ECS Exec 工作階段的閒置逾時值為 20 分鐘。無法變更此值。
- 您無法針對現有任務開啟 ECS Exec。只能針對新任務開啟。
- 當您使用 在叢集上`run-task`啟動任務時，無法使用 ECS Exec，該叢集使用受管擴展與非同步置放（在沒有執行個體的情況下啟動任務）。
- 您無法針對 Microsoft Nano Server 容器執行 ECS Exec。

必要條件

開始使用 ECS Exec 之前，請確定您已完成下列動作：

- 安裝及設定 AWS CLI。如需詳細資訊，請參閱 入門 AWS CLI。
- 安裝 的 Session Manager 外掛程式 AWS CLI。如需詳細資訊，請參閱為 AWS CLI安裝工作階段管理工具外掛程式。
- 您必須使用具有 ECS Exec 適當許可的任務角色。如需詳細資訊，請參閱任務 IAM 角色。
- ECS Exec 具有版本要求，取決於您的任務是否託管於 Amazon EC2 或 AWS Fargate：
  - 如果您使用的是 Amazon EC2，則必須使用在 2021 年 1 月 20 日之後發行的 Amazon ECS 最佳化 AMI，且代理程式版本為 1.50.2 或更高版本。如需詳細資訊，請參閱 Amazon ECS 最佳化 AMI。
  - 如果您使用的是 AWS Fargate，則必須使用平台版本 `1.4.0` 或更新版本 (Linux) 或 `1.0.0`(Windows)。如需詳細資訊，請參閱 AWS Fargate 平台版本。

架構

ECS Exec 會使用 AWS Systems Manager (SSM) Session Manager 與執行中的容器建立連線，並使用 AWS Identity and Access Management (IAM) 政策來控制對執行中容器中執行中命令的存取。將必要的 SSM 代理程式二進位檔繫結掛載至容器，即可達成此目的。Amazon ECS 或 AWS Fargate 代理程式負責在您的應用程式程式碼中啟動容器內的 SSM 核心代理程式。如需詳細資訊，請參閱 Systems Manager 工作階段管理員。

您可以使用 中的`ExecuteCommand`事件稽核哪些使用者存取容器， AWS CloudTrail 並將每個命令（及其輸出）記錄到 Amazon S3 或 Amazon CloudWatch Logs。若要使用您自己的加密金鑰加密本機用戶端和容器之間的資料，您必須提供 AWS Key Management Service (AWS KMS) 金鑰。

使用 ECS Exec

可選任務定義變更

如果您將任務定義參數設定為 `initProcessEnabled` `true`，這會啟動容器內的初始化程序。這會移除找到的任何 zombie SSM 代理程式子程序。以下是範例。

```
{
    "taskRoleArn": "ecsTaskRole",
    "networkMode": "awsvpc",
    "requiresCompatibilities": [
        "EC2",
        "FARGATE"
    ],
    "executionRoleArn": "ecsTaskExecutionRole",
    "memory": ".5 gb",
    "cpu": ".25 vcpu",
    "containerDefinitions": [
        {
            "name": "amazon-linux",
            "image": "amazonlinux:latest",
            "essential": true,
            "command": ["sleep","3600"],
            "linuxParameters": {
                "initProcessEnabled": true
            }
        }
    ],
```

```
    "family": "ecs-exec-task"
}
```

為您的任務和服務開啟 ECS Exec

您可以在使用下列其中一個 AWS CLI 命令時指定 `--enable-execute-command`旗標，以開啟服務和獨立任務的 ECS Exec 功能：create-service、start-task、 update-service或 run-task。

例如，如果您執行下列命令，則新建立的服務在 Fargate 上執行時，ECS Exec 功能會開啟。如需建立服務的詳細資訊，請參閱 create-service。

```
aws ecs create-service \
    --cluster cluster-name \
    --task-definition task-definition-name \
    --enable-execute-command \
    --service-name service-name \
    --launch-type FARGATE \
     --network-configuration
 "awsvpcConfiguration={subnets=[subnet-12344321],securityGroups=[sg-12344321],assignPublicIp=EN
 \
    --desired-count 1
```

為任務開啟 ECS Exec 之後，您可以執行下列命令來確認任務已可供使用。如果 `ExecuteCommandAgent` 的 `lastStatus` 屬性會列為 `RUNNING` 且 `enableExecuteCommand` 屬性會設為 `true`，則您的任務已準備就緒。

```
aws ecs describe-tasks \
    --cluster cluster-name \
    --tasks task-id
```

以下輸出程式碼片段為您可能會看到的內容範例。

```
{
    "tasks": [
        {
            ...
            "containers": [
                {
                    ...
```

```
                    "managedAgents": [
                        {
                            "lastStartedAt": "2021-03-01T14:49:44.574000-06:00",
                            "name": "ExecuteCommandAgent",
                            "lastStatus": "RUNNING"
                        }
                    ]
                }
            ],
            ...
            "enableExecuteCommand": true,
            ...
        }
    ]
}
```

使用 ECS Exec 執行命令

在確認 `ExecuteCommandAgent` 正在執行後，您可以使用以下命令在容器上開啟交互式 Shell。如果您的任務包含多個容器，您必須使用 `--container` 旗標指定容器名稱。Amazon ECS 僅支援啟動互動式工作階段，因此您必須使用 `--interactive` 旗標。

下列命令將針對 ID 為 *`task-id ######## container-name`* 的容器執行互動式`/bin/sh`命令。

`task-id` 是任務的 Amazon Resource Name (ARN)。

```
aws ecs execute-command --cluster cluster-name \
    --task task-id \
    --container container-name \
    --interactive \
    --command "/bin/sh"
```

使用 ECS Exec 記錄

開啟記錄您的任務和服務

> **Important**
>
> 如需了解有關 CloudWatch 定價的詳細資訊，請參閱 CloudWatch 定價。Amazon ECS 還提供免費的監控指標。如需詳細資訊，請參閱使用 CloudWatch 監控 Amazon ECS。

Amazon ECS 提供使用 ECS Exec 執行之記錄命令的預設組態。預設為使用任務定義中設定的日誌驅動程式，將awslogs日誌傳送至 CloudWatch Logs。如果您想要提供自訂組態， AWS CLI 支援create-cluster和 update-cluster命令的--configuration旗標。容器映像需要cat安裝script和 ，才能將命令日誌正確上傳至 Amazon S3 或 CloudWatch Logs。如需建立叢集的詳細資訊，請參閱 create-cluster。

> **Note**
>
> 此組態只會處理 execute-command 工作階段的日誌記錄。它不會影響應用程式的日誌記錄。

下列範例會建立叢集，然後將輸出記錄至名為 cloudwatch-log-group-name 的 CloudWatch Logs LogGroup 和名為 s3-bucket-name 的 Amazon S3 儲存貯體。

當您將 CloudWatchEncryptionEnabled選項設定為 時，必須使用 AWS KMS 客戶受管金鑰來加密日誌群組true。如需有關如何加密日誌群組的資訊，請參閱Amazon CloudWatch Logs 《 使用者指南》中的使用 在 CloudWatch Logs 中加密日誌資料 AWS Key Management Service。

```
aws ecs create-cluster \
    --cluster-name cluster-name \
    --configuration executeCommandConfiguration="{ \
        kmsKeyId=string, \
        logging=OVERRIDE, \
        logConfiguration={ \
            cloudWatchLogGroupName=cloudwatch-log-group-name, \
            cloudWatchEncryptionEnabled=true, \
            s3BucketName=s3-bucket-name, \
            s3EncryptionEnabled=true, \
            s3KeyPrefix=demo \
        } \
    }"
```

logging 屬性決定 ECS Exec 的日誌記錄功能行為：

- NONE：記錄已關閉。
- DEFAULT：日誌會傳送至設定的awslogs驅動程式。如果未設定驅動程式，則不會儲存日誌。
- OVERRIDE：日誌會傳送至提供的 Amazon CloudWatch Logs LogGroup、Amazon S3 儲存貯體或兩者。

Amazon CloudWatch Logs 或 Amazon S3 日誌記錄所需的 IAM 許可

若要啟用日誌記錄，任務定義中參考的 Amazon ECS 任務角色需要有額外的許可。這些額外的許可可以作為政策新增至任務角色。視您是否將日誌導向至 Amazon CloudWatch Logs 或 Amazon S3 而定，它們會有所不同。

Amazon CloudWatch Logs

下列政策範例會新增必要 Amazon CloudWatch Logs 許可。

```
{
   "Version": "2012-10-17",
   "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "logs:DescribeLogGroups"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
               "logs:CreateLogStream",
               "logs:DescribeLogStreams",
               "logs:PutLogEvents"
            ],
            "Resource": "arn:aws:logs:region:account-id:log-group:/aws/
ecs/cloudwatch-log-group-name:*"
        }
   ]
}
```

Amazon S3

下列政策範例會新增必要 Amazon S3 許可。

```
{
   "Version": "2012-10-17",
   "Statement": [
        {
            "Effect": "Allow",
```

```
            "Action": [
               "s3:GetBucketLocation"
            ],
            "Resource": "*"
        },
        {
           "Effect": "Allow",
           "Action": [
               "s3:GetEncryptionConfiguration"
           ],
           "Resource": "arn:aws:s3:::s3-bucket-name"
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:PutObject"
            ],
            "Resource": "arn:aws:s3:::s3-bucket-name/*"
        }
    ]
   }
```

使用您自己的加密所需的 IAM 許可 AWS KMS key (KMS 金鑰)

根據預設，本機用戶端與容器之間傳輸的資料會使用 AWS 提供的 TLS 1.2 加密。若要使用您自己的 KMS 金鑰進一步加密資料，您必須建立 KMS 金鑰並新增 `kms:Decrypt` 許可至您的任務 IAM 角色。您的容器使用此許可來解密資料。如需建立 KMS 金鑰的詳細資訊，請參閱建立金鑰。

您可以將下列內嵌政策新增至需要 AWS KMS 許可的任務 IAM 角色。如需詳細資訊，請參閱ECS Exec 許可。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "kms:Decrypt"
            ],
            "Resource": "kms-key-arn"
        }
    ]
```

```
}
```

若要使用您自己的 KMS 金鑰加密資料，則必須向使用 `execute-command` 動作的使用者或群組授予 `kms:GenerateDataKey` 許可。

下列使用者或群組的範例政策包含使用您自己的 KMS 金鑰的必要許可。您必須指定 KMS 金鑰的 Amazon Resource Name (ARN)。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "kms:GenerateDataKey"
            ],
            "Resource": "kms-key-arn"
        }
    ]
}
```

使用 IAM 政策限制 ECS Exec 的存取

您可以使用下列一或多個 IAM 政策條件金鑰，限制使用者對執行命令 API 動作的存取：

- `aws:ResourceTag/clusterTagKey`
- `ecs:ResourceTag/clusterTagKey`
- `aws:ResourceTag/taskTagKey`
- `ecs:ResourceTag/taskTagKey`
- `ecs:container-name`
- `ecs:cluster`
- `ecs:task`
- `ecs:enable-execute-command`

使用下列範例 IAM 政策，使用者可以在任務內執行的容器中執行命令，任務標籤具有 `environment` 鍵和 `development` 值，並位於名為 `cluster-name` 的叢集中。

```
{
```

```
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ecs:ExecuteCommand",
                "ecs:DescribeTasks"
            ],
         "Resource": [
                "arn:aws:ecs:region:aws-account-id:task/cluster-name/*",
                "arn:aws:ecs:region:aws-account-id:cluster/*"
            ],
            "Condition": {
                "StringEquals": {
                    "ecs:ResourceTag/environment": "development"
                }
            }
        }
    ]
}
```

在下列 IAM 政策範例中，當容器名稱為 `production-app` 時，使用者無法使用 `execute-command` API。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Deny",
            "Action": [
                "ecs:ExecuteCommand"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "ecs:container-name": "production-app"
                }
            }
        }
    ]
}
```

使用下列 IAM 政策，使用者只能在 ECS Exec 關閉時啟動任務。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ecs:RunTask",
                "ecs:StartTask",
                "ecs:CreateService",
                "ecs:UpdateService"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "ecs:enable-execute-command": "false"
                }
            }
        }
    ]
}
```

> **Note**
>
> 由於 `execute-command` API 動作只包含請求中的任務和叢集資源，所以只會評估叢集和任務標籤。

如需有關 IAM 政策條件索引鍵的詳細資訊，請參閱《服務授權參考》中的 Amazon Elastic Container Service 的動作、資源與條件索引鍵。

限制存取「啟動工作階段」動作

雖然可以在 ECS Exec 之外的容器上啟動 SSM 工作階段，但這可能會導致工作階段未被記錄。在 ECS Exec 以外啟動的工作階段也會計入工作階段配額。建議您透過直接拒絕針對使用 IAM 政策之 Amazon ECS 任務的 `ssm:start-session` 動作來限制此存取。您可以根據使用的標籤拒絕存取所有 Amazon ECS 任務或特定任務。

以下為範例 IAM 政策，它會針對具有特定叢集名稱之所有區域中的任務，拒絕存取 `ssm:start-session` 動作。您可以選擇包含具有 *`cluster-name`* 的萬用字元。

```
{
```

```
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Deny",
            "Action": "ssm:StartSession",
            "Resource": [
                "arn:aws:ecs:region:aws-account-id:task/cluster-name/*",
                "arn:aws:ecs:region:aws-account-id:cluster/*"
            ]
        }
    ]
}
```

以下為範例 IAM 政策，它會針對所有具有標籤鍵 `Task-Tag-Key` 和標籤值 `Exec-Task` 的區域中的資源，拒絕對 `ssm:start-session` 動作的存取。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Deny",
            "Action": "ssm:StartSession",
            "Resource": "arn:aws:ecs:*:*:task/*",
            "Condition": {
                "StringEquals": {
                    "aws:ResourceTag/Task-Tag-Key": "Exec-Task"
                }
            }
        }
    ]
}
```

如需使用 Amazon ECS Exec 時可能遇到的任何問題的協助，請參閱針對 Exec 的問題進行故障診斷。

AWS Compute Optimizer Amazon ECS 的建議

AWS Compute Optimizer 會針對 Amazon ECS 任務和容器大小產生建議。如需詳細資訊，請參閱《 AWS Compute Optimizer使用者指南》中的什麼是AWS Compute Optimizer ？。

上的 Amazon ECS 服務的任務和容器大小建議 AWS Fargate

AWS Compute Optimizer 會在 上產生 Amazon ECS 服務的建議 AWS Fargate。 AWS Compute Optimizer 建議任務 CPU 和任務記憶體大小，以及容器 CPU、容器記憶體和容器記憶體保留大小。這些建議會顯示在 Compute Optimizer 主控台的以下頁面中。

- 針對 Fargate 上 Amazon ECS 服務的建議頁面
- Fargate 上的 Amazon ECS 服務詳細資訊頁面

如需詳細資訊，請參閱《AWS Compute Optimizer 使用者指南》中的檢視針對 Fargate 上 Amazon ECS 服務的建議。

Amazon ECS 故障診斷

您可能需要對負載平衡器、任務、服務或容器執行個體的問題進行故障診斷。本章會協助您從 Amazon ECS 容器代理程式、容器執行個體的 Docker 常駐程式，以及 Amazon ECS 主控台的服務事件日誌中尋找診斷資訊。

如需有關已停止任務的資訊，請參閱下列其中一項。

動作	進一步了解
解決已停止的任務錯誤。	檢視 Amazon ECS 已停止的任務錯誤
檢視已停止的任務錯誤。	解決 Amazon ECS 已停止的任務錯誤
檢閱已停止的任務錯誤代碼。	Amazon ECS 已停止任務錯誤訊息
檢閱 CannotPullContainer 任務錯誤。	Amazon ECS 中的 CannotPullContainer 任務錯誤
檢視任務 IAM 角色請求。	檢視 Amazon ECS 任務的 IAM 角色請求

如需服務錯誤的相關資訊，請參閱下列其中一項。

動作	進一步了解
檢視服務事件訊息。	檢視 Amazon ECS 服務事件訊息
檢閱服務事件訊息。	Amazon ECS 服務事件訊息
檢閱負載平衡器問題。	對 Amazon ECS 中的服務負載平衡器進行故障診斷

動作	進一步了解	
檢閱服務自動擴展問題。	對 Amazon ECS 中的服務自動擴展進行故障診斷	

如需任務定義錯誤的相關資訊，請參閱下列其中一項。

動作	進一步了解	
解決任務定義記憶體錯誤。	對 Amazon ECS 任務定義無效的 CPU 或記憶體錯誤進行故障診斷	

如需 Amazon ECS 代理程式錯誤的相關資訊，請參閱下列其中一項。

動作	進一步了解	
檢視 Amazon ECS 容器代理程式日誌。	檢視 Amazon ECS 容器代理程式日誌	
了解如何收集 Amazon ECS 日誌。	使用 Amazon ECS 日誌收集器收集容器日誌	
使用 Amazon ECS 代理程式擷取診斷詳細資訊。	使用代理程式簡介擷取 Amazon ECS 診斷詳細資訊	

如需 Docker 錯誤的相關資訊，請參閱下列其中一項。

動作	進一步了解	
使用 Docker 診斷。	Amazon ECS 中的 Docker 診斷	
開啟 Docker 偵錯模式。	在 Amazon ECS 中設定 Docker 協助程式的詳細輸出	

動作	進一步了解	
對 Docker API 錯誤 500 進行故障診斷。	在 Amazon ECS API error (500): devmapper 中對 Docker 進行故障診斷	

如需 ECS Exec 和 Amazon ECS Anywhere 錯誤的相關資訊，請參閱下列其中一項。

動作	進一步了解	
ECS Exec. 故障診斷	對 Amazon ECS Exec 問題進行故障診斷	
對 Amazon ECS Anywhere 進行故障診斷。	對 Amazon ECS Anywhere 問題進行故障診斷	

如需限流問題的相關資訊，請參閱下列其中一項。

動作	進一步了解	
了解 Fargate 調節配額。	AWS Fargate 調節配額	
了解 Amazon ECS 限流的最佳實務。	處理 Amazon ECS 限流問題	

如需有關 API 錯誤的資訊，請參閱下列其中一項。

動作	進一步了解	
解決 API 錯誤。	Amazon ECS API 失敗原因	

解決 Amazon ECS 已停止的任務錯誤

當您的任務無法啟動時，您會在主控台和`describe-tasks`輸出參數 (`stoppedReason` 和) 中看到錯誤訊息`stoppedCode`。

您可以在主控台中檢視停止的任務一小時。若要查看停止的任務，您必須變更篩選條件選項。如需詳細資訊，請參閱檢視 Amazon ECS 已停止的任務錯誤。

以下頁面提供有關已停止任務的資訊。

- 了解已停止任務錯誤訊息的變更。

  Amazon ECS 已停止任務錯誤訊息更新
- 檢視停止的任務，以便取得原因的相關資訊。

  檢視 Amazon ECS 已停止的任務錯誤
- 了解停止的任務錯誤訊息，以及可能的錯誤原因。

  Amazon ECS 已停止任務錯誤訊息
- 了解如何驗證已停止的任務連線並修正錯誤。

  驗證 Amazon ECS 已停止任務連線

Amazon ECS 已停止任務錯誤訊息更新

自 2024 年 6 月 14 日起，Amazon ECS 團隊將變更停止的任務錯誤訊息，如下表所述。`stopCode` 不會變更。如果您的應用程式依賴確切的錯誤訊息字串，您必須使用新的字串來更新應用程式。如需問題或疑問的協助，請聯絡 AWS 支援。

> **Note**
>
> 我們建議您不要依賴自動化的錯誤訊息，因為錯誤訊息可能會有所變更。

CannotPullContainerError

舊錯誤訊息	新錯誤訊息	
CannotPullContainerError：協助程式的錯誤回應：提取儲存#的存取遭拒、儲存庫不存在或可能需要 'docker login'：遭拒：使用者： *roleARN*	• CannotPullContainerError：任務無法提取映像。檢查角色是否具有從登錄檔提取映像的許可。協助程式的錯誤回應：提取儲存#的存取遭	

舊錯誤訊息	新錯誤訊息	
	拒、儲存庫不存在或可能需要 'docker login'：遭拒：使用者： *roleARN* 未獲授權執行：ecr：BatchGetImage on resource： *image* ，因為身分型政策不允許 ecr：BatchGetImage 動作。 • CannotPullContainerError：任務無法提取映像。檢查映像是否存在。協助程式的錯誤回應：提取儲存#的存取遭拒、儲存庫不存在或可能需要 'docker login'：遭拒：請求存取資源遭拒。	
CannotPullContainerError：協助程式的錯誤回應：取得 *imageURI*：net/http： 請求在等待連線時取消	CannotPullContainerError：任務無法提取映像。檢查您的網路組態。協助程式的錯誤回應：取得##：net/http： 請求在等待連線時取消 （等待標頭時超過 Client.Timeout)	
CannotPullContainerError：已重試 5 次 （無法複製） 的 ref 提取：httpReadSeeker： 無法開啟：無法執行請求： Get *registry-uri* ： dial tcp <ip>：<port> i/o 逾時	CannotPullContainerError：任務無法從登錄檔*-uri ## image-uri*。任務與登錄檔之間發生連線問題。檢查您的任務網路組態。 ： 無法複製：httpReadSeeker： 無法開啟： 無法執行請求： Get *registry-uri* ： dial tcp <ip>：<port> i/o 逾時	

ResourceNotFoundException

舊錯誤訊息	新錯誤訊息	
從 AWS Secrets Manager ##擷取秘密資料：秘密 *sercretARN* ：ResourceNotFoundException：Secrets Manager 找不到指定的秘密。	ResourceNotFoundException：任務無法從中擷取具有 ARN '*sercretARN* ' 的秘密 AWS Secrets Manager。檢查秘密是否存在於指定的區域中。ResourceNotFoundException：從 AWS Secrets Manager ##擷取秘密資料：秘密 *sercretARN* ：ResourceNotFoundException：Secrets Manager 找不到指定的秘密。	

ResourceInitializationError

舊錯誤訊息	新錯誤訊息	
ResourceInitializationError：無法提取秘密或登錄驗證：執行資源擷取失敗：無法擷取 ecr 登錄驗證：服務呼叫已重試 3 次：RequestError：傳送請求失敗，原因為：Post "https://api.ecr.us-east-1.amazonaws.com/": dial tcp <ip>：<port>：i/o 逾時。請檢查您的任務網路組態。	ResourceInitializationError：無法提取秘密或登錄驗證：任務無法從 Amazon ECR 提取登錄驗證：任務與 Amazon ECR 之間發生連線問題。檢查您的任務網路組態。RequestError：傳送請求失敗，原因為：Post "https://api.ecr.us-east-1.amazonaws.com": dial tcp <ip>：<port>：i/o 逾時	
ResourceInitializationError：無法提取秘密或登錄驗證：執行資源擷取失敗：無法從 ssm 擷取秘密：服務呼叫已重試 5	ResourceInitializationError：無法提取秘密或登錄驗證：無法從 ssm 擷取秘密：任務無法從 提取秘密 AWS	

舊錯誤訊息	新錯誤訊息	
次：RequestCanceled：請求內容因：超過內容截止日期而取消	Systems Manager。任務和 AWS Systems Manager 參數存放區之間發生連線問題。檢查您的任務網路組態。RequestCanceled：請求內容因下列原因取消：超過內容截止日期	
ResourceInitializationError：無法下載 env 檔案：檔案下載命令：非空白錯誤串流：RequestCanceled：請求內容因：超過內容截止日期而取消	ResourceInitializationError：無法下載 env 檔案：任務無法從 Amazon S3 下載環境變數檔案。任務與 Amazon S3 之間發生連線問題。檢查您的任務網路組態。服務呼叫已重試 5 次：RequestCanceled：請求內容已取消，原因為：超過內容截止日期	
ResourceInitializationError：驗證記錄器 args：：signal：killed 失敗	ResourceInitializationError：無法驗證記錄器 args：任務找不到任務定義中定義的 Amazon CloudWatch 日誌群組。任務與 Amazon CloudWatch 之間發生連線問題。檢查您的網路組態。：訊號：已終止	
ResourceInitializationError：無法提取秘密或登錄驗證：提取命令失敗：：訊號：已終止	ResourceInitializationError：無法提取秘密或登錄驗證：檢查您的任務網路組態。：訊號：已終止	

檢視 Amazon ECS 已停止的任務錯誤

如果啟動任務時有問題，您的任務可能會因為應用程式或組態錯誤而停止。例如，您執行任務，而任務顯示 PENDING 狀態，然後就消失。

如果您的任務是由 Amazon ECS 服務建立，Amazon ECS 為維護服務所採取的動作會發佈在服務事件中。您可以在 AWS Management Console、 AWS CLI、 AWS SDKs、Amazon ECS API 或使用 SDKs和 API 的工具中檢視事件。這些事件包括 Amazon ECS 停止並取代任務，因為任務中的容器已停止執行，或多次未通過 Elastic Load Balancing 的運作狀態檢查。

如果您的任務在 Amazon EC2 或外部電腦上的容器執行個體上執行，您也可以查看容器執行期和 Amazon ECS 代理程式的日誌。這些日誌位於主機 Amazon EC2 執行個體或外部電腦上。如需詳細資訊，請參閱檢視 Amazon ECS 容器代理程式日誌。

程序

Console

AWS Management Console

下列步驟可用來使用 主控台檢查已停止的任務是否發生錯誤。若要查看停止的任務，您必須變更篩選條件選項。

停止的任務只會在主控台中顯示 1 小時。

1. 開啟主控台，網址為 https://console.aws.amazon.com/ecs/v2。
2. 在導覽窗格中，選擇叢集。
3. 在叢集頁面上，選擇叢集。
4. 在 Cluster : ***name*** (叢集：名稱) 頁面上，選擇 Tasks (任務) 索引標籤。
5. 設定篩選條件以顯示已停止的任務。針對篩選條件所需的狀態，選擇已停止。

   Stopped (已停止) 選項會顯示已停止的任務，而 Any desired status (任何所需狀態) 會顯示所有任務。
6. 選擇要檢查的已停止任務。
7. 在已停止任務的列中，在 Last Status (上次狀態) 欄中，選擇 Stopped (已停止)。

   快顯視窗會顯示停止的原因。

AWS CLI

1. 列出叢集中停止的任務。輸出包含任務的 Amazon Resource Name (ARN)，您需要 Amazon Resource Name (ARN) 來描述任務。

```
aws ecs list-tasks \
     --cluster cluster_name \
     --desired-status STOPPED \
     --region region
```

2. 描述停止的任務以擷取資訊。如需詳細資訊，請參閱 AWS Command Line Interface 參考中的 describe-tasks。

```
aws ecs describe-tasks \
     --cluster cluster_name \
     --tasks arn:aws:ecs:region:account_id:task/cluster_name/task_ID \
     --region region
```

使用下列輸出參數。

- `stopCode` - 停止碼指出任務停止的原因，例如 ResourceInitializationError
- `StoppedReason` - 任務停止的原因。
- `reason`（在 `containers`結構中）- 原因提供已停止容器的其他詳細資訊。

後續步驟

檢視停止的任務，以便取得原因的相關資訊。如需詳細資訊，請參閱Amazon ECS 已停止任務錯誤訊息。

Amazon ECS 已停止任務錯誤訊息

以下是當您的任務意外停止時，您可能會收到的錯誤訊息。

若要使用 檢查已停止的任務是否有錯誤訊息 AWS Management Console，請參閱 檢視 Amazon ECS 已停止的任務錯誤。

已停止的任務錯誤代碼具有與其相關聯的類別，例如 "ResourceInitializationError"。若要取得每個類別的詳細資訊，請參閱以下內容：

類別	進一步了解
TaskFailedToStart	對 Amazon ECS TaskFailedToStart 錯誤進行故障診斷
ResourceInitializationError	對 Amazon ECS ResourceInitializationError 錯誤進行故障診斷
ResourceNotFoundException	對 Amazon ECS ResourceNotFoundException 錯誤進行故障診斷
SpotInterruptionError	對 Amazon ECS SpotInterruption 錯誤進行故障診斷
InternalError	對 Amazon ECS InternalError 錯誤進行故障診斷
OutOfMemoryError	對 Amazon ECS OutOfMemoryError 錯誤進行故障診斷
ContainerRuntimeError	對 Amazon ECS ContainerRuntimeError 錯誤進行故障診斷
ContainerRuntimeTimeoutError	對 Amazon ECS ContainerRuntimeTimeoutError 錯誤進行故障診斷
CannotStartContainerError	對 Amazon ECS CannotStartContainerError 錯誤進行故障診斷
CannotStopContainerError	對 Amazon ECS CannotStopContainerError 錯誤進行故障診斷

類別	進一步了解	
CannotInspectContainerError	對 Amazon ECS CannotInspectContainerError 錯誤進行故障診斷	
CannotCreateVolumeError	對 Amazon ECS CannotCreateVolumeError 錯誤進行故障診斷	
CannotPullContainer	Amazon ECS 中的 CannotPullContainer 任務錯誤	

對 Amazon ECS TaskFailedToStart 錯誤進行故障診斷

以下是一些`TaskFailedToStart`錯誤訊息和動作，您可以採取這些動作來修正錯誤。

若要使用 檢查已停止的任務是否有錯誤訊息 AWS Management Console，請參閱 檢視 Amazon ECS 已停止的任務錯誤。

嘗試在子網路 '***subnet-id*** 中啟用公有 IP 指派時建立網路介面時發生非預期的 EC2 錯誤

當 Fargate 任務使用`awsvpc`網路模式並在具有公有 IP 地址的子網路中執行，且子網路沒有足夠的 IP 地址時，就會發生這種情況。

可用的 IP 地址數量可在 Amazon EC2 主控台的子網路詳細資訊頁面上取得，或使用 `describe-subnets`。如需詳細資訊，請參閱《Amazon VPC 使用者指南》中的檢視子網路。

若要修正此問題，您可以建立新的子網路以執行任務。

InternalError：***<reason>***

在請求 ENI 附件時，會發生此錯誤。Amazon EC2 以非同步方式處理 ENI 的佈建。佈建程序需要時間。Amazon ECS 會出現逾時，導致等待時間較長或未報告的故障。有時會佈建 ENI，但報告會在故障逾時後傳送給 Amazon ECS。在這種情況下，Amazon ECS 會透過使用中的 ENI 查看報告的任務失敗。

選取的任務定義與選取的運算策略不相容

當您選擇具有不符合叢集容量類型的啟動類型的任務定義時，就會發生此錯誤。如需詳細資訊，請參閱Amazon ECS啟動類型。您需要選取符合指派給叢集之容量提供者的任務定義。

無法將網路介面連接至未使用的裝置索引

當使用awsvpc網路類型且任務沒有足夠的 CPU/記憶體時，會發生此錯誤。首先，檢查執行個體的 CPU。如需詳細資訊，請參閱 Amazon EC2 執行個體類型中的 Amazon EC2 執行個體類型規格。Amazon EC2 取得執行個體的 CPU 值，並將其乘以執行個體ENIs 數量。在任務定義中使用該值 e。

代理程式

您嘗試啟動任務所在之容器執行個體上的代理程式目前中斷連線。為避免延長任務置放等待時間，請求遭到拒絕。

如需如何對已中斷連線的代理程式進行故障診斷的相關資訊，請參閱 How do I troubleshoot a disconnected Amazon ECS agent (如何對已中斷連線的 Amazon ECS 代理程式進行故障診斷)。

對 Amazon ECS ResourceInitializationError 錯誤進行故障診斷

以下是一些ResourceInitialization錯誤訊息和動作，您可以採取這些動作來修正錯誤。

若要使用 檢查已停止的任務是否有錯誤訊息 AWS Management Console，請參閱 檢視 Amazon ECS 已停止的任務錯誤。

錯誤

- 任務無法從 Amazon ECR 提取登錄身分驗證。任務與 Amazon ECR 之間發生連線問題。檢查您的任務網路組態。
- 任務無法從 Amazon S3 下載環境變數檔案。任務與 Amazon S3 之間發生連線問題。檢查您的任務網路組態。
- 任務無法從 AWS Systems Manager 參數存放區提取秘密。檢查任務與 之間的網路連線 AWS Systems Manager。
- 任務無法從中提取秘密 AWS Secrets Manager。任務與 Secrets Manager 之間發生連線問題。檢查您的任務網路組態。
- 任務無法從 Secrets Manager 提取秘密。任務無法從 Secrets Manager 使用 ARN ‘secretARN' 擷取秘密。檢查秘密是否存在於指定的區域中。
- pull 命令失敗：無法提取秘密或登錄驗證 檢查您的任務網路組態。
- 任務找不到任務定義中定義的 Amazon CloudWatch 日誌群組。任務與 Amazon CloudWatch 之間發生連線問題。檢查您的網路組態。
- 無法初始化記錄驅動程式
- 無法叫用 EFS utils 命令來設定 EFS 磁碟區

任務無法從 Amazon ECR 提取登錄身分驗證。任務與 Amazon ECR 之間發生連線問題。檢查您的任務網路組態。

此錯誤表示任務無法連線至 Amazon ECR。

檢查任務與 Amazon ECR 之間的連線。如需相關資訊，請參閱 驗證 Amazon ECS 已停止任務連線。

任務無法從 Amazon S3 下載環境變數檔案。任務與 Amazon S3 之間發生連線問題。檢查您的任務網路組態。

當您的任務無法從 Amazon S3 下載環境檔案時，就會發生此錯誤。

檢查任務與 Amazon S3 VPC 端點之間的連線。如需相關資訊，請參閱 驗證 Amazon ECS 已停止任務連線。

任務無法從 AWS Systems Manager 參數存放區提取秘密。檢查任務與 之間的網路連線 AWS Systems Manager。

當您的任務無法使用 Systems Manager 中的登入資料提取任務定義中定義的映像時，就會發生此錯誤。

檢查任務與 Systems Manager VPC 端點之間的連線。如需相關資訊，請參閱 驗證 Amazon ECS 已停止任務連線。

任務無法從中提取秘密 AWS Secrets Manager。任務與 Secrets Manager 之間發生連線問題。檢查您的任務網路組態。

當您的任務無法使用 Secrets Manager 中的登入資料提取任務定義中定義的映像時，就會發生此錯誤。

錯誤表示 Systems Manager VPC 端點和任務之間存在網路連線問題。

如需有關如何驗證任務與端點之間連線的資訊，請參閱 驗證 Amazon ECS 已停止任務連線。

任務無法從 Secrets Manager 提取秘密。任務無法從 Secrets Manager 使用 ARN '***secretARN***' 擷取秘密。檢查秘密是否存在於指定的區域中。

當您的任務無法使用 Secrets Manager 中的登入資料提取任務定義中定義的映像時，就會發生此錯誤。

此問題是由下列其中一個原因所造成：

錯誤原因..	執行此作業...	
Secrets Manager VPC 端點與任務之間的網路連線問題。 當您在錯誤訊息中看到下列任何字串時，問題是網路問題： • 撥打 tcp • 撥打 udp • <ip>：<port>：i/o 逾時 • net/http：TLS 交握逾時 • 讀取：連線逾時 • 等待標頭時超過 Client.Timeout • net/http： 請求在等待連線時取消 • 訊號：已終止 • 超過內容截止日期	驗證任務與 Secrets Manager 端點之間的連線。如需詳細資訊，請參閱驗證 Amazon ECS 已停止任務連線。	
任務定義中定義的任務執行角色沒有 Secrets Manager 的許可。	將 Secrets Manager 所需的許可新增至任務執行角色。如需詳細資訊，請參閱Secrets Manager 或 Systems Manager 許可。	
秘密 ARN 不存在	檢查 ARN 是否存在於 Secrets Manager 中。如需有關檢視映像的資訊，請參閱《Secrets Manager 開發人員指南》中的在 Secrets Manager 中尋找秘密。	

pull 命令失敗：無法提取秘密或登錄驗證 檢查您的任務網路組態。

當您的任務無法連線至 Amazon ECR、Systems Manager 或 Secrets Manager 時，就會發生此錯誤。這是因為您的網路組態錯誤。

若要修正此問題，請確認任務與 Amazon ECR 之間的連線。您也需要檢查任務與存放秘密的服務 (Systems Manager 或 Secrets Manager) 之間的連線。如需詳細資訊，請參閱驗證 Amazon ECS 已停止任務連線。

任務找不到任務定義中定義的 Amazon CloudWatch 日誌群組。任務與 Amazon CloudWatch 之間發生連線問題。檢查您的網路組態。

當您的任務找不到您在任務定義中定義的 CloudWatch 日誌群組時，就會發生此錯誤。

錯誤表示 CloudWatch VPC 端點與任務之間存在網路連線問題。

如需如何驗證任務與端點之間連線的資訊，請參閱 驗證 Amazon ECS 已停止任務連線。

無法初始化記錄驅動程式

當您的任務找不到您在任務定義中定義的 CloudWatch 日誌群組時，就會發生此錯誤。

錯誤表示任務定義中的 CloudWatch 群組不存在。

使用下列步驟尋找缺少的 CloudWatch。

1. 執行下列命令以取得任務定義資訊。

```
aws ecs describe-task-definition \
    --task-definition task-def-name
```

查看每個容器的輸出，並記下 `awslogs-group`值。

```
"logConfiguration": {
                "logDriver": "awslogs",
                "options": {
                    "awslogs-group": "/ecs/example-group",
                    "awslogs-create-group": "true",
                    "awslogs-region": "us-east-1",
                    "awslogs-stream-prefix": "ecs"
                },
```

2. 如需詳細資訊，請確認群組存在於 CloudWatch 中，請參閱《Amazon CloudWatch Logs 使用者指南》中的使用日誌群組和日誌串流。

   問題是任務定義中指定的群組不正確，或日誌群組不存在。

3. 修正問題。

問題是...	執行此作業...
任務定義中指定了不正確的日誌群組。	更新任務定義以在容器定義中包含日誌群組組態。如需更新任務定義的相關資訊，請參閱 Amazon Elastic Container Service API 參考中的 使用主控台更新 Amazon ECS 任務定義或 RegisterTaskDefinition。
CloudWatch 中不存在日誌群組	建立日誌群組。如需詳細資訊，請參閱《Amazon CloudWatch Logs 使用者指南》中的在 CloudWatch Logs 中建立日誌群組。

無法叫用 EFS utils 命令來設定 EFS 磁碟區

下列問題可能會使您無法在您的 請求上掛載 Amazon EFS 磁碟區：

- Amazon EFS 檔案系統未正確設定。
- 任務沒有必要的許可。
- 網路和 VPC 組態發生問題。

如需有關如何偵錯和修正此問題的資訊，請參閱為何我無法在 re：Post 上的 AWS Fargate 任務上掛載 Amazon EFS 磁碟區。 AWS

對 Amazon ECS ResourceNotFoundException 錯誤進行故障診斷

以下是一些 `ResourceNotFoundException`錯誤訊息和動作，您可以採取這些動作來修正錯誤。

若要使用 檢查已停止的任務是否有錯誤訊息 AWS Management Console，請參閱 檢視 Amazon ECS 已停止的任務錯誤。

任務無法從中擷取具有 ARN '*sercretARN*' 的秘密 AWS Secrets Manager。檢查秘密是否存在於指定的區域中。

當任務無法從 Secrets Manager 擷取秘密時，會發生此錯誤。這表示任務定義 （並包含在錯誤訊息中） 中指定的秘密不存在於 Secrets Manager 中。

區域在錯誤訊息中。

從 AWS Secrets Manager *##*擷取秘密資料：秘密 *sercretARN*：ResourceNotFoundException：Secrets Manager 找不到指定的秘密。

如需尋找秘密的詳細資訊，請參閱AWS Secrets Manager 《 使用者指南》中的在 中尋找秘密 AWS Secrets Manager。

使用下表來判斷和解決錯誤。

問題	動作	
秘密位於與任務定義不同的區域。	a. 在與任務相同的區域中建立秘密。如需詳細資訊，請參閱建立 AWS Secrets Manager 秘密。 b. 使用新的秘密更新任務定義。如需詳細資訊，請參閱《Amazon Elastic Container Service API 參考》中的使用主控台更新 Amazon ECS 任務定義或 RegisterTaskDefinition。	
任務定義具有不正確的秘密 ARN。Secrets Manager 中存在正確的秘密。	使用正確的秘密更新任務定義。如需詳細資訊，請參閱《Amazon Elastic Container Service API 參考》中的 使用主控台更新 Amazon ECS 任	

問題	動作	
	務定義或 RegisterTaskDefinition。	
秘密不再存在。	a. 在與任務相同的區域中建立秘密。如需詳細資訊，請參閱建立 AWS Secrets Manager 秘密。 b. 使用新的秘密更新任務定義。如需詳細資訊，請參閱《Amazon Elastic Container Service API 參考》中的使用主控台更新 Amazon ECS 任務定義或 RegisterTaskDefinition。	

對 Amazon ECS SpotInterruption 錯誤進行故障診斷

`SpotInterruption` 錯誤對於 Fargate 和 EC2 啟動類型有不同的原因。

若要使用 檢查已停止的任務是否有錯誤訊息 AWS Management Console，請參閱 檢視 Amazon ECS 已停止的任務錯誤。

Fargate 啟動類型

當沒有 Fargate Spot 容量或 Fargate 取回 Spot 容量時，`SpotInterruption`就會發生錯誤。

您可以讓任務在多個可用區域中執行，以允許更多容量。

EC2 啟動類型

當沒有可用的 Spot 執行個體或 EC2 取回 Spot 執行個體容量時，就會發生此錯誤。

您可以讓執行個體在多個可用區域中執行，以允許更多容量。

對 Amazon ECS InternalError 錯誤進行故障診斷

適用於：Fargate 啟動類型

若要使用 檢查已停止的任務是否有錯誤訊息 AWS Management Console，請參閱 檢視 Amazon ECS 已停止的任務錯誤。

代理程式遇到非預期、非執行期相關內部`InternalError`錯誤時的錯誤。

只有在使用平台版本 `1.4` 或更新版本時，才會發生此錯誤。

如需有關如何偵錯和修正此問題的資訊，請參閱如何對在 re：Post 上 ECS 叢集中無法啟動的 Amazon ECS 任務進行故障診斷。 AWS

對 Amazon ECS OutOfMemoryError 錯誤進行故障診斷

以下是一些 OutOfMemoryError 錯誤訊息和動作，您可以採取這些動作來修正錯誤。

若要使用 檢查已停止的任務是否有錯誤訊息 AWS Management Console，請參閱 檢視 Amazon ECS 已停止的任務錯誤。

容器因記憶體使用量而遭到終止

當容器結束是由於容器中的處理序消耗的記憶體比任務定義中分配的記憶體更多時，就會發生此錯誤。

對 Amazon ECS ContainerRuntimeError 錯誤進行故障診斷

以下是一些 ContainerRuntimeError 錯誤訊息和動作，您可以採取這些動作來修正錯誤。

若要使用 檢查已停止的任務是否有錯誤訊息 AWS Management Console，請參閱 檢視 Amazon ECS 已停止的任務錯誤。

ContainerRuntimeError

當代理程式從 `containerd` 接收到特定執行時間操作的意外錯誤時，會發生此錯誤。此錯誤通常是由代理程式或 `containerd` 執行時間的內部失敗所造成。

只有在使用平台版本 `1.4.0` 或更新版本 (Linux)，或 `1.0.0` 或更新版本 (Windows) 時，才會發生此錯誤。

如需有關如何偵錯和修正此問題的資訊，請參閱為什麼我的 Amazon ECS 任務在 re：Post 上停止。 AWS

對 Amazon ECS ContainerRuntimeTimeoutError 錯誤進行故障診斷

以下是一些 ContainerRuntimeTimeoutError 錯誤訊息和動作，您可以採取這些動作來修正錯誤。

若要使用 檢查已停止的任務是否有錯誤訊息 AWS Management Console，請參閱 檢視 Amazon ECS 已停止的任務錯誤。

無法轉換為執行中；等待 1 公尺或 Docker 逾時錯誤後逾時

當容器在逾時期間無法轉換成 RUNNING 或 STOPPED 狀態時，會發生此錯誤。錯誤訊息中會提供原因和逾時值。

對 Amazon ECS CannotStartContainerError 錯誤進行故障診斷

以下是一些可以用來修正錯誤的 CannotStartContainerError 錯誤訊息和動作。

若要使用 檢查已停止的任務是否有錯誤訊息 AWS Management Console，請參閱 檢視 Amazon ECS 已停止的任務錯誤。

無法取得容器狀態：*<reason>*

無法啟動容器時，會發生此錯誤。

如果您的容器嘗試超過此處指定的記憶體，則容器會停止。增加提供給容器的記憶體。這是任務定義中的 memory 參數。如需詳細資訊，請參閱the section called “記憶體”。

對 Amazon ECS CannotStopContainerError 錯誤進行故障診斷

以下是您可以採取的一些 CannotStopContainerError 錯誤訊息和動作來修正錯誤。

若要使用 檢查已停止的任務是否有錯誤訊息 AWS Management Console，請參閱 檢視 Amazon ECS 已停止的任務錯誤。

CannotStopContainerError

無法停用容器時，會發生此錯誤。

如需有關如何偵錯和修正此問題的資訊，請參閱為什麼我的 Amazon ECS 任務在 re：Post 上停止。AWS

對 Amazon ECS CannotInspectContainerError 錯誤進行故障診斷

以下是一些可以用來修正錯誤的 CannotInspectContainerError 錯誤訊息和動作。

若要使用 檢查已停止的任務是否有錯誤訊息 AWS Management Console，請參閱 檢視 Amazon ECS 已停止的任務錯誤。

CannotInspectContainerError

當容器代理程式無法透過容器執行時間描述容器時，會發生此錯誤。

使用平台版本 1.3或更早版本時，Amazon ECS 代理程式會從 Docker 傳回原因。

使用平台版本 1.4.0 或更新版本 (Linux) 1.0.0或更新版本 (Windows) 時，Fargate 代理程式會從 傳回原因containerd。

如需有關如何偵錯和修正此問題的資訊，請參閱為什麼我的 Amazon ECS 任務在 re：Post 上停止。AWS

對 Amazon ECS CannotCreateVolumeError 錯誤進行故障診斷

以下是您可以採取的一些 CannotCreateVolumeError 錯誤訊息和動作來修正錯誤。

若要使用 檢查已停止的任務是否有錯誤訊息 AWS Management Console，請參閱 檢視 Amazon ECS 已停止的任務錯誤。

CannotCreateVolumeError

當代理程式無法建立任務定義中指定的磁碟區掛載時，會發生此錯誤。

只有在使用平台版本 1.4.0 或更新版本 (Linux)，或 1.0.0 或更新版本 (Windows) 時，才會發生此錯誤。

如需有關如何偵錯和修正此問題的資訊，請參閱為什麼我的 Amazon ECS 任務在 re：Post 上停止。AWS

Amazon ECS 中的 CannotPullContainer 任務錯誤

下列錯誤表示任務無法啟動，因為 Amazon ECS 無法擷取指定的容器映像。

> **Note**
>
> 1.4 Fargate 平台版本會截斷長的錯誤訊息。

若要使用 檢查已停止的任務是否有錯誤訊息 AWS Management Console，請參閱 檢視 Amazon ECS 已停止的任務錯誤。

錯誤

- 任務無法提取映像。檢查角色是否具有從登錄檔提取映像的許可。
- 任務無法從 Amazon ECR 儲存庫「儲存庫 URI」中提取「image-name」。任務與 Amazon ECR 之間發生連線問題。檢查您的任務網路組態。
- 任務無法提取映像。檢查您的網路組態
- API 錯誤 (500)：取得在等待連線時取消的 https://111122223333.dkr.ecr.us-east-1.amazonaws.com/v2/: net/http： 請求
- API 錯誤
- write /var/lib/docker/tmp/GetImageBlob111111111：裝置上沒有剩餘空間
- 錯誤：toomanyrequests：請求過多或已達到提取率限制。
- 協助程式的錯誤回應：取得 URL：net/http： 請求在等待連線時取消
- 已重試 1 次 （ 多個 ） 的參考提取：無法複製：httpReaderSeeker：無法開啟：未預期狀態碼
- 提取存取遭拒
- pull 命令失敗：驚慌：執行時間錯誤：記憶體地址無效或 nil 指標解構
- 提取映像組態時發生錯誤/提取映像組態時發生錯誤
- 內容已取消

任務無法提取映像。檢查角色是否具有從登錄檔提取映像的許可。

此錯誤表示任務因為許可問題而無法提取任務定義中指定的映像。

若要解決此問題：

1. 檢查映像是否存在於###中。如需有關檢視映像的詳細資訊，請參閱《Amazon Elastic Container Registry 使用者指南》中的在 Amazon ECR 中檢視映像詳細資訊。
2. 確認 *role-arn* 具有提取映像的正確許可。

   如需有關如何更新角色的資訊，請參閱 AWS Identity and Access Management 使用指南中的更新角色的許可。

   任務使用以下其中一個角色：

   - 對於具有 Fargate 啟動類型的任務，這是任務執行角色。如需 Amazon ECR 額外許可的相關資訊，請參閱透過介面端點許可提取 Amazon ECR 映像的 Fargate 任務。
   - 對於具有 EC2 啟動類型的任務，這是容器執行個體角色。如需 Amazon ECR 額外許可的相關資訊，請參閱Amazon ECR 許可。

任務無法從 Amazon ECR 儲存庫「*### URI*」中提取「*image-name*」。任務與 Amazon ECR 之間發生連線問題。檢查您的任務網路組態。

此錯誤表示任務無法連線至 Amazon ECR。檢查儲存*# URI* 儲存庫的連線。

如需有關如何驗證和解決問題的資訊，請參閱 驗證 Amazon ECS 已停止任務連線。

任務無法提取映像。檢查您的網路組態

此錯誤表示任務無法連線至 Amazon ECR。

如需有關如何驗證和解決問題的資訊，請參閱 驗證 Amazon ECS 已停止任務連線。

API 錯誤 (500)：取得在等待連線時取消的 https://111122223333.dkr.ecr.us-east-1.amazonaws.com/v2/: net/http： 請求

此錯誤表示連線逾時，因為網際網路的路由不存在。

要解決此問題，則可以採取下列方式：

- 針對公有子網路中的任務，您必須在啟動任務時，將 Auto-assign public IP (自動指派公有 IP) 指定為 ENABLED (啟用)。如需詳細資訊，請參閱以 Amazon ECS 任務執行應用程式。
- 針對在私有子網路中的任務，則必須在啟動任務時，將 Auto-assign public IP (自動指派公有 IP) 指定為 DISABLED (停用)，並設定 VPC 中的 NAT 閘道，以便將請求路由至網際網路。如需詳細資訊，請參閱 Amazon VPC 使用者指南中的 NAT 閘道。

API 錯誤

此錯誤表示 Amazon ECR 端點發生連線問題。

如需如何解決此問題的資訊，請參閱 支援 網站上的如何解決 Amazon ECS 中的 Amazon ECR 錯誤「CannotPullContainerError： API 錯誤」。

write /var/lib/docker/tmp/*GetImageBlob111111111*：裝置上沒有剩餘空間

此錯誤表示磁碟空間不足。

若要解決這個問題，請釋放磁碟空間。

如果您使用的是 Amazon ECS 最佳化 AMI，您可以使用下列命令來擷取檔案系統上 20 個最大的檔案：

```
du -Sh / | sort -rh | head -20
```

輸出範例：

```
5.7G    /var/lib/docker/
containers/50501b5f4cbf90b406e0ca60bf4e6d4ec8f773a6c1d2b451ed8e0195418ad0d2
1.2G    /var/log/ecs
594M    /var/lib/docker/devicemapper/mnt/
c8e3010e36ce4c089bf286a623699f5233097ca126ebd5a700af023a5127633d/rootfs/data/logs
...
```

在某些情況下，執行中的容器可能會填入根磁碟區。如果容器使用預設的 `json-file` 日誌驅動程式，而不具有 `max-size` 限制，則該日誌檔案可能佔用大部分空間。您可以使用 `docker ps` 命令，透過將目錄名稱從上面的輸出映射至容器 ID，來驗證哪個容器正在使用該空間。例如：

```
CONTAINER ID   IMAGE                                COMMAND             CREATED
 STATUS              PORTS                               NAMES
50501b5f4cbf   amazon/amazon-ecs-agent:latest   "/agent"            4 days ago
 Up 4 days                                           ecs-agent
```

在預設情況下，當使用 `json-file` 日誌驅動程式時，Docker 會擷取所有容器的標準輸出 (和標準錯誤)，並使用 JSON 格式將其寫入檔案中。您可以將 `max-size` 設定為日誌驅動程式選項，以防止日誌檔案佔用過多空間。如需詳細資訊，請參閱 Docker 文件中的 JSON 檔案記錄驅動程式。

下列是一個容器定義片段，顯示如何使用此選項：

```
{
    "log-driver": "json-file",
    "log-opts": {
        "max-size": "256m"
    }
}
```

如果您的容器日誌佔用太多磁碟空間，另一種方法是使用`awslogs`日誌驅動程式。`awslogs` 日誌驅動程式會將日誌傳送至 CloudWatch，以釋放原本要給容器執行個體上的容器日誌使用的磁碟空間。如需詳細資訊，請參閱將 Amazon ECS 日誌傳送至 CloudWatch 。

錯誤：toomanyrequests：請求過多或已達到提取率限制。

此錯誤表示有 Docker Hub 速率限制。

如果收到下列其中一項錯誤訊息，您很可能會達到 Docker Hub 速率限制：

如需 Docker Hub 速率限制的詳細資訊，請參閱了解 Docker Hub 速率限制。

如果您已提高 Docker Hub 速率限制，而且需要驗證容器執行個體的 Docker 提取，請參閱容器執行個體的私有登錄驗證。

協助程式的錯誤回應：取得 ***URL***：net/http： 請求在等待連線時取消

此錯誤表示連線逾時，因為網際網路的路由不存在。

要解決此問題，則可以採取下列方式：

- 針對公有子網路中的任務，您必須在啟動任務時，將 Auto-assign public IP (自動指派公有 IP) 指定為 ENABLED (啟用)。如需詳細資訊，請參閱以 Amazon ECS 任務執行應用程式。
- 針對在私有子網路中的任務，則必須在啟動任務時，將 Auto-assign public IP (自動指派公有 IP) 指定為 DISABLED (停用)，並設定 VPC 中的 NAT 閘道，以便將請求路由至網際網路。如需詳細資訊，請參閱 Amazon VPC 使用者指南中的 NAT 閘道。

已重試 1 次（多個）的參考提取：無法複製：httpReaderSeeker：無法開啟：未預期狀態碼

此錯誤表示複製映像時發生故障。

若要解決此問題，請檢閱下列其中一個文章：

- 如需 Fargate 任務，請參閱如何解決 Fargate 上 Amazon ECS 任務的 "cannotpullcontainererror" 錯誤。
- 如需其他任務，請參閱如何解決 Fargate 上 Amazon ECS 任務的 "cannotpullcontainererror" 錯誤。

提取存取遭拒

此錯誤表示無法存取映像。

若要解決此問題，您可能需要使用 Amazon ECR 驗證 Docker 用戶端。如需詳細資訊，請參閱《Amazon ECR 使用者指南》中的私有登錄身分驗證。

pull 命令失敗：驚慌：執行時間錯誤：記憶體地址無效或 nil 指標解構

此錯誤表示由於無效的記憶體地址或 nil 指標解引用，無法存取映像。

若要解決此問題：

- 檢查您是否擁有可連接 Amazon S3 的安全群組規則。
- 使用閘道端點時，您必須在路由表中新增路由，才能存取端點。

提取映像組態時發生錯誤/提取映像組態時發生錯誤

此錯誤表示已達到速率限制或發生網路錯誤：

若要解決此問題，請參閱我的 Amazon ECS EC2 啟動類型任務中的如何解決「CannotPullContainerError」錯誤。

內容已取消

此錯誤表示內容已取消。

此錯誤的原因通常是因為您的任務所使用的 VPC 沒有路由，而無法從 Amazon ECR 提取容器映像。

驗證 Amazon ECS 已停止任務連線

有時候任務會因為網路連線問題而停止。這可能是間歇性問題，但很有可能是因為任務無法連線至端點。

測試任務連線

您可以使用 `AWSSupport-TroubleshootECSTaskFailedToStart` Runbook 來測試任務連線。當您使用 Runbook 時，您需要下列資源資訊：

- 任務 ID

  使用最近失敗任務的 ID。
- 任務所在的叢集

如需有關如何使用 Runbook 的資訊，請參閱 AWS Systems Manager Automation Runbook 參考`AWSSupport-TroubleshootECSTaskFailedToStart`中的 。

Runbook 會分析任務。您可以在輸出區段中檢視下列問題的結果，這些問題可能會讓任務無法啟動：

- 已設定容器登錄檔的網路連線

- VPC 端點連線
- 安全群組規則組態

修正 VPC 端點問題

當 `AWSSupport-TroubleshootECSTaskFailedToStart` Runbook 結果指出 VPC 端點問題時，請檢查下列組態：

- 您建立端點的 VPC 需要使用私有 DNS。
- 請確定您擁有 服務的 AWS PrivateLink 端點，而任務無法在與任務相同的 VPC 中連線到該端點。如需詳細資訊，請參閱下列其中一項：

服務	服務的 VPC 端點資訊
Amazon ECR	Amazon ECR 介面 VPC 端點 (AWS PrivateLink)
Systems Manager	使用適用於 Systems Manager 的 VPC 端點來改善 EC2 執行個體的安全性
Secrets Manager	使用 AWS Secrets Manager VPC 端點
CloudWatch	CloudWatch VPC 端點
Amazon S3	AWS PrivateLink 適用於 Amazon S3

- 設定任務子網路的傳出規則，允許連接埠 443 DNS (TCP) 流量上的 HTTPS。如需詳細資訊，請參閱《Amazon Elastic Compute Cloud 使用者指南》中的設定安全群組規則。
- 如果您使用自訂名稱網域伺服器，請確認 DNS 查詢的設定。查詢必須具有連接埠 53 的傳出存取權，並使用 UDP 和 TCP 通訊協定。此外，它必須在連接埠 443 上具有 HTTPS 存取。如需詳細資訊，請參閱《Amazon Elastic Compute Cloud 使用者指南》中的 Coonfigure 安全群組規則。
- 如果子網路具有網路 ACL，則需要下列 ACL 規則：
  - 允許連接埠 1024-65535 流量的傳出規則。
  - 允許連接埠 443 上 TCP 流量的傳入規則。

如需有關如何設定規則的資訊，請參閱《Amazon Virtual Private Cloud 使用者指南》中的使用網路 ACLs 控制子網路的流量。

修正網路問題

當 `AWSSupport-TroubleshootECSTaskFailedToStart` Runbook 結果指出網路問題時，請檢查下列組態：

在公有子網路中使用 awsvpc 網路模式的任務

根據 Runbook 執行下列組態：

- 針對公有子網路中的任務，您必須在啟動任務時，將 Auto-assign public IP (自動指派公有 IP) 指定為 ENABLED (啟用)。如需詳細資訊，請參閱以 Amazon ECS 任務執行應用程式。
- 您需要閘道來處理網際網路流量。任務子網路的路由表需要有通往閘道的流量路由。

  如需詳細資訊，請參閱《Amazon Virtual Private Cloud 使用者指南》中的從路由表新增和移除路由。

閘道類型	路由表目的地	路由表目標
NAT	0.0.0.0/0	NAT 閘道 ID
網際網路閘道	0.0.0.0/0	網際網路閘道 ID

- 如果任務子網路具有網路 ACL，則需要下列 ACL 規則：
  - 允許連接埠 1024-65535 流量的傳出規則。
  - 允許連接埠 443 上 TCP 流量的傳入規則。

  如需有關如何設定規則的資訊，請參閱《Amazon Virtual Private Cloud 使用者指南》中的使用網路 ACLs 控制子網路的流量。

在私有子網路中使用 awsvpc 網路模式的任務

根據 Runbook 執行下列組態：

- 啟動任務時，針對自動指派公有 IP 選擇停用。

- 在 VPC 中設定 NAT 閘道，將請求路由到網際網路。如需詳細資訊，請參閱《Amazon Virtual Private Cloud 使用者指南》中的 NAT Gateways。
- 任務子網路的路由表需要有路由，才能將流量傳送到 NAT 閘道。

  如需詳細資訊，請參閱《Amazon Virtual Private Cloud 使用者指南》中的從路由表新增和移除路由。

閘道類型	路由表目的地	路由表目標
NAT	0.0.0.0/0	NAT 閘道 ID

- 如果任務子網路具有網路 ACL，則需要下列 ACL 規則：
  - 允許連接埠 1024-65535 流量的傳出規則。
  - 允許連接埠 443 上 TCP 流量的傳入規則。

  如需有關如何設定規則的資訊，請參閱《Amazon Virtual Private Cloud 使用者指南》中的使用網路 ACLs 控制子網路的流量。

在公有子網路中不使用 awsvpc 網路模式的任務

根據 Runbook 執行下列組態：

- 當您建立叢集時，請在 Amazon EC2 執行個體的聯網下選擇開啟自動指派 IP。

  此選項會將公有 IP 地址指派給執行個體主要網路介面。
- 您需要閘道來處理網際網路流量。執行個體子網路的路由表需要有通往閘道的流量路由。

  如需詳細資訊，請參閱《Amazon Virtual Private Cloud 使用者指南》中的從路由表新增和移除路由。

閘道類型	路由表目的地	路由表目標
NAT	0.0.0.0/0	NAT 閘道 ID
網際網路閘道	0.0.0.0/0	網際網路閘道 ID

- 如果執行個體子網路具有網路 ACL，則需要下列 ACL 規則：
  - 允許連接埠 1024-65535 流量的傳出規則。

- 允許連接埠 443 上 TCP 流量的傳入規則。

如需有關如何設定規則的資訊，請參閱《Amazon Virtual Private Cloud 使用者指南》中的使用網路 ACLs 控制子網路的流量。

在私有子網路中不使用 awsvpc 網路模式的任務

根據 Runbook 執行下列組態：

- 當您建立叢集時，請選擇 Networking for Amazon EC2 執行個體下的關閉自動指派 IP。
- 在 VPC 中設定 NAT 閘道，將請求路由到網際網路。如需詳細資訊，請參閱 Amazon VPC 使用者指南中的 NAT 閘道。
- 執行個體子網路的路由表需要有路由，才能將流量傳送到 NAT 閘道。

  如需詳細資訊，請參閱《Amazon Virtual Private Cloud 使用者指南》中的從路由表新增和移除路由。

閘道類型	路由表目的地	路由表目標
NAT	0.0.0.0/0	NAT 閘道 ID

- 如果任務子網路具有網路 ACL，則需要下列 ACL 規則：
  - 允許連接埠 1024-65535 流量的傳出規則。
  - 允許連接埠 443 上 TCP 流量的傳入規則。

  如需有關如何設定規則的資訊，請參閱《Amazon Virtual Private Cloud 使用者指南》中的使用網路 ACLs 控制子網路的流量。

檢視 Amazon ECS 任務的 IAM 角色請求

當您在 IAM 角色中使用 提供者做為任務登入資料時，該提供者請求會儲
存在稽核日誌中。稽核紀錄會繼承與容器代理程式紀錄相同的紀錄輪換設
定。ECS_LOG_ROLLOVER_TYPE、ECS_LOG_MAX_FILE_SIZE_MB 和 ECS_LOG_MAX_ROLL_COUNT 容器代理程式組態變數可供設定來影響稽核日誌的行為。如需詳細資訊，請參閱Amazon ECS 容器代理程式日誌組態參數。

針對 1.36.0 和更新版本的容器代理程式，稽核日誌會位在 /var/log/ecs/audit.log。紀錄輪換時，會在日誌檔名稱的結尾新增 *YYYY*-*MM*-*DD*-*HH* 格式的時間戳記。

針對 1.35.0 和更舊版本的容器代理程式，稽核日誌會位在 /var/log/ecs/audit.log.*YYYY*-*MM*-*DD*-*HH*。

日誌項目的格式如下：

- 時間戳記
- HTTP 回應代碼
- 請求來源的 IP 地址和連接埠號碼
- 登入資料提供者的相對 URI
- 提出請求的使用者代理
- 提出請求容器所屬之任務的 ARN
- GetCredentials API 名稱和版本編號
- 容器執行個體註冊對象的 Amazon ECS 叢集名稱
- 容器執行個體 ARN

您可以使用下列命令來檢視日誌檔案。

```
cat /var/log/ecs/audit.log.2016-07-13-16
```

輸出：

```
2016-07-13T16:11:53Z 200 172.17.0.5:52444 "/v1/credentials" "python-requests/2.7.0
 CPython/2.7.6 Linux/4.4.14-24.50.amzn1.x86_64" TASK_ARN GetCredentials
 1 CLUSTER_NAME CONTAINER_INSTANCE_ARN
```

檢視 Amazon ECS 服務事件訊息

對服務問題進行故障診斷時，您第一個應該檢查以取得診斷資訊的位置是服務事件日誌。您可以使用 DescribeServices API、 AWS CLI或 來檢視服務事件 AWS Management Console。

使用 Amazon ECS API 檢視服務事件訊息時，只會傳回來自服務排程器的事件。其中包括最近的任務放置和執行個體運作狀態事件。不過，Amazon ECS 主控台會顯示下列來源的服務事件。

- 來自 Amazon ECS 服務排程器的任務放置和執行個體運作狀態事件。這些事件具有 服務字首 ***(service-name)***。為了確保此事件檢視有幫助，我們只會顯示 100 個最近事件，在解決原因或經過六個小時之前，會忽略重複事件訊息。如果原因未在六小時內解決，您會收到該原因的另一個服務事件訊息。
- Service Auto Scaling 事件。這些事件具有 訊息的字首。系統會顯示 10 個最近擴展事件。只有在使用 Application Auto Scaling 擴展政策設定服務時，才會發生這些事件。

請依照下列步驟檢視您目前的服務事件訊息。

Console

1. 開啟主控台，網址為 https://console.aws.amazon.com/ecs/v2。
2. 在導覽窗格中，選擇叢集。
3. 在叢集頁面上，選擇叢集。
4. 選擇要檢查的服務。
5. 選擇 Deployments and events (部署和事件)，然後在 Events (事件) 下檢視訊息。

AWS CLI

使用 describe-services 命令來檢視指定服務的服務事件訊息。

下列 AWS CLI 範例說明##叢集中的 *service-name* 服務，可提供最新的服務事件訊息。

```
aws ecs describe-services \
    --cluster default \
    --services service-name \
    --region us-west-2
```

Amazon ECS 服務事件訊息

您會在 Amazon ECS 主控台中看到以下服務事件訊息範例：

服務 (***service-name***) 已達到穩定狀態。

服務排程器會在service (*service-name*) has reached a steady state.服務運作狀態良好且達到所需任務數量時傳送服務事件，進而達到穩定狀態。

服務排程器會定期報告狀態，因此您可能會多次收到此訊息。

服務 (*service-name*) 無法放置任務，因為沒有容器執行個體符合其所有要求。

服務排程器會在找不到可新增其他任務的可用資源時傳送此事件訊息。可能的原因如下：

您的叢集中找不到任何容器執行個體

如果您嘗試執行任務的叢集中沒有註冊容器執行個體，則會收到此錯誤。您應該將容器執行個體新增到您的叢集。如需詳細資訊，請參閱啟動 Amazon ECS Linux 容器執行個體。

連接埠不足

如果您的任務使用固定的主機連接埠映射 (例如，您的任務為 Web 伺服器使用主機的連接埠 80)，則您每項任務至少必須擁有一個容器執行個體，因為一個容器一次只能使用單一個主機連接埠。您應該新增容器執行個體到您的叢集，或降低所需任務數量。

註冊的連接埠太多

任務置放最接近的相符容器執行個體，不得超過每個容器執行個體允許 100 個主機連接埠的預留連接埠上限。使用動態主機連接埠映射可以修復此問題。

連接埠已在使用中

此任務的任務定義在其連接埠映射中使用相同的連接埠，做為已在所選容器執行個體上執行的任務。服務事件訊息會將具有所選容器執行個體 ID 作為以下訊息的一部分。

```
The closest matching container-instance is already using a port required by your
 task.
```

記憶體不足

如果您的任務定義指定 1000 MiB 的記憶體，而您叢集中的容器執行個體每個都有 1024 MiB 的記憶體，則您每個容器執行個體只能執行一個此任務複本。您可以在任務定義中試驗使用較少的記憶體，以便您每個容器執行個體可以啟動多項任務，或在您的叢集中啟動更多的容器執行個體。

> **Note**
>
> 若您嘗試盡可能為特定執行個體類型的任務提供最多的記憶體，以將資源使用率最大化，請參閱「保留 Amazon ECS Linux 容器執行個體記憶體」。

CPU 不足

容器執行個體的每個 CPU 核心都有 1,024 CPU 單位。如果您的任務定義指定 1,000 CPU 單位，而您叢集中的容器執行個體每個都有 1,024 CPU 單位，您每個容器執行個體只能執行一個此任務複本。您可以在任務定義中試驗使用較少的 CPU 單位，以便您每個容器執行個體可以啟動多項任務，或在您的叢集中啟動更多的容器執行個體。

可用的 ENI 連接點不足

使用 awsvpc 網路模式的每項任務都會收到自己的彈性網路介面 (ENI)，連接到裝載它的容器執行個體。Amazon EC2 執行個體有可以連接至其的 ENI 數目限制，而且有 ENI 容量可用的叢集中沒有容器執行個體。

個別容器執行個體的 ENI 限制取決於以下條件：

- 如果您尚未選擇使用 awsvpcTrunking 帳戶設定，則每個容器執行個體的 ENI 限制取決於執行個體類型。如需詳細資訊，請參閱《Amazon EC2 使用者指南》中的每個執行個體類型每個網路介面的 IP 地址。
- 如果您已選擇加入awsvpcTrunking帳戶設定，但在選擇加入後尚未使用支援的執行個體類型啟動新的容器執行個體，則每個容器執行個體的 ENI 限制仍為預設值。如需詳細資訊，請參閱《Amazon EC2 使用者指南》中的每個執行個體類型每個網路介面的 IP 地址。
- 如果您已選擇使用 awsvpcTrunking 帳戶設定，且已在選擇使用後，使用支援的執行個體類型啟動新的容器執行個體，則會有額外的 ENI 可用。如需詳細資訊，請參閱支援增加 Amazon ECS 容器網路介面的執行個體。

如需選擇使用的 awsvpcTrunking 帳戶設定詳細資訊，請參閱 增加 Amazon ECS Linux 容器執行個體網路介面。

您可以將容器執行個體新增到您的叢集，以提供更多可用的網路轉接器。

容器執行個體遺漏必要的屬性

有些任務定義參數需要在容器執行個體上安裝特定的 Docker 遠端 API 版本。其他，例如記錄驅動程式選項，需要容器執行個體向 ECS_AVAILABLE_LOGGING_DRIVERS 代理組態變數註冊這些日誌驅動程式。如果您的任務定義包含需要特定容器執行個體屬性的參數，而且您沒有任何可滿足此需求的可用容器執行個體，則無法放置任務。

如果您的服務使用使用awsvpc網路模式和 EC2 啟動類型的任務，則此錯誤的常見原因。您指定的叢集沒有在建立服務awsvpcConfiguration時在 中指定的相同子網路中註冊的容器執行個體。

您可以使用 AWSSupport-TroubleshootECSContainerInstance 執行手冊進行故障診斷。Runbook 會檢閱執行個體的使用者資料是否包含正確的叢集資訊、執行個體描述檔是否包含必要的許可，以

及網路組態問題。如需詳細資訊，請參閱 AWS Systems Manager Automation Runbook 參考使用者指南中的 AWSSupport-TroubleshootECSContainerInstance。

有關特定任務定義參數和代理組態變數需要哪些必要屬性的詳細資訊，請參閱「Amazon ECS 任務定義參數」和「Amazon ECS 容器代理程式組態」。

服務 (*service-name*) 無法放置任務，因為沒有容器執行個體符合其所有要求。最接近的相符容器執行個體 (*###### id*) 可用 CPU 單位不足。

最接近的任務置放相符容器執行個體不包含足夠的 CPU 單位，以滿足任務定義中的要求。檢閱任務定義之任務大小和容器定義參數的 CPU 需求。

服務 (*service-name*) 無法放置任務，因為沒有容器執行個體符合其所有要求。最接近的相符容器執行個體 *container-instance-id* 發生「代理」錯誤。

適合放置任務之最接近相符容器執行個體中的 Amazon ECS 容器代理中斷連線。如果您可以使用 SSH 連線到容器執行個體，您即可檢查代理日誌，如需詳細資訊，請參閱「Amazon ECS 容器代理程式日誌組態參數」。您也應該確認代理是否在執行個體上執行。如果您使用的是 Amazon ECS 最佳化 AMI，您可以使用下列命令嘗試停用並重新啟動代理。

- 適用於 Amazon ECS 最佳化 Amazon Linux 2 AMI 和 Amazon ECS 最佳化 Amazon Linux 2023 AMI

```
sudo systemctl restart ecs
```

- 對於 Amazon ECS 最佳化 Amazon Linux AMI

```
sudo stop ecs && sudo start ecs
```

服務 (*service-name*) (執行個體 *instance-id*) 在 (elb *elb-name*) 中運作狀態不良，是因為 (執行個體失敗的原因是幾乎沒有連續的運作狀態檢查 UnhealthyThreshold 數。)

此服務已在負載平衡器註冊，但負載平衡器的運作狀態檢查失敗。如需詳細資訊，請參閱對 Amazon ECS 中的服務負載平衡器進行故障診斷。

服務 (*service-name*) 無法成功持續啟動任務。

此服務包含在連續嘗試後無法啟動的任務。此時，服務排程器開始逐漸增加重試之間的時間。您應該對任務無法啟動的原因進行故障診斷。如需詳細資訊，請參閱Amazon ECS 服務調節邏輯。

服務更新之後，例如使用更新的任務定義，服務排程器就會恢復正常的行為。

服務 (*service-name*) 操作正在受到調節。將稍後再試。

由於 API 調節限制，此服務無法啟動更多任務。一旦服務排程器能夠啟動更多任務，就會繼續執行。

若要請求提高 API 比率限制配額，請開啟 AWS 支援 Center (中心) 頁面，並視需要登入，然後選擇 Create case (建立案例)。選擇 Service limit increase (提高服務限制)。填妥並提交表格。

服務 (*service-name*) 在部署期間因服務部署組態而無法停止或啟動任務。更新 minimumHealthyPercent 或 maximumPercent 值，然後再試一次。

由於部署組態，此服務無法在服務部署期間停止或啟動任務。部署組態包含 `minimumHealthyPercent`和 `maximumPercent`值，這些值會在建立服務時定義。這些值也可以在現有服務上更新。

`minimumHealthyPercent` 代表部署期間或容器執行個體耗盡時，應為服務執行的任務數量下限。這是服務所需任務數量的百分比。此值會向上捨入。例如，如果運作狀態百分比下限為 `50`，且所需的任務計數為 4，則排程器可以在啟動兩個新任務之前停止兩個現有任務。同樣地，如果最小運作狀態良好的百分比為 75%，而所需的任務計數為 2，則排程器無法停止任何任務，因為產生的值也是 2。

`maximumPercent` 代表部署期間或容器執行個體耗盡時，應為服務執行的任務數量上限。這是服務所需任務數量的百分比。此值會向下捨去。例如，如果最大百分比為 `200`，而所需的任務計數為 4，則排程器可以在停止四個現有任務之前啟動四個新任務。同樣地，如果最大百分比為 125，且所需的任務計數為 3，則排程器無法停止任何任務，因為產生的值也是 3。

設定最小最小運作狀態良好的百分比或最大百分比時，您應確定排程器在觸發部署時，至少可以停止或啟動一項任務。

服務 (*service-name*) 無法放置任務。原因：您已達到可以同時執行的任務數量限制

您可以請求提升導致錯誤的資源配額。如需詳細資訊，請參閱Service Quotas。若要請求增加配額，請參閱《Service Quotas 使用者指南》中的請求提高配額。

服務 (*service-name*) 無法放置任務。原因：內部錯誤。

以下是此錯誤的可能原因：

由於子網路位於不受支持的可用區域中，該服務無法開始任務。

如需支援的區域的 Fargate 和可用區域的相關資訊，請參閱 the section called “AWS Fargate 區域”。

如需如何檢視子網路可用區域的相關資訊，請參閱《Amazon VPC 使用者指南》中的檢視您的子網路。

服務 (*service-name*) 無法放置任務。原因：請求的 CPU 組態超過您的限制。

您可以請求提升導致錯誤的資源配額。如需詳細資訊，請參閱Service Quotas。若要請求增加配額，請參閱《Service Quotas 使用者指南》中的請求提高配額。

服務 (*service-name*) 無法放置任務。原因：請求的 MEMORY 組態超過您的限制。

您可以請求提升導致錯誤的資源配額。如需詳細資訊，請參閱Service Quotas。若要請求增加配額，請參閱《Service Quotas 使用者指南》中的請求提高配額。

服務 (*service-name*) 無法放置任務。原因：您已達到可以同時執行的 vCPU 數量限制

AWS Fargate 正在從任務計數型配額轉換為 vCPU 型配額。

對於 Fargate vCPU 型配額，您可請求提高配額。如需詳細資訊，請參閱Service Quotas。若要請求提高 Fargate 配額，請參閱《Service Quotas 使用者指南》中的請求提高配額。

服務 (*service-name*) 無法達到穩定狀態，因為任務集 (*taskSet-ID*) 無法縮減。原因：受保護任務的數量超過所需的任務數量

服務比所需任務數量有更多受保護的任務。您可以執行下列任一動作：

- 等待目前任務的保護到期，然後終止這些任務。
- 決定哪些任務可以停止，並使用 `UpdateTaskProtection` API 搭配 設定為 `protectionEnabled`的選項`false`，以取消設定這些任務的保護。
- 將服務的所需任務計數提高到超過受保護任務的數目。

服務 (*service-name*) 無法達到穩定狀態。原因：在容量提供者中找不到容器執行個體。

服務排程器會在找不到可新增其他任務的可用資源時傳送此事件訊息。可能的原因如下：

沒有與叢集相關聯的容量提供者

使用 `describe-services` 來驗證您擁有與叢集相關聯的容量提供者 您可以更新服務的容量提供者策略。

確認容量提供者中有可用的容量，在 EC2 啟動類型的情況下，請確定容器執行個體符合任務定義要求。

您的叢集中找不到任何容器執行個體

如果您嘗試執行任務的叢集中沒有註冊容器執行個體，則會收到此錯誤。您應該將容器執行個體新增到您的叢集。如需詳細資訊，請參閱啟動 Amazon ECS Linux 容器執行個體。

連接埠不足

如果您的任務使用固定主機連接埠映射 （ 例如，您的任務在 Web 伺服器主機上使用連接埠 80)，則每個任務必須至少有一個容器執行個體。一次只能有一個容器使用單一主機連接埠。您應該新增容器執行個體到您的叢集，或降低所需任務數量。

註冊的連接埠太多

任務置放最接近的相符容器執行個體，不得超過每個容器執行個體允許 100 個主機連接埠的預留連接埠上限。使用動態主機連接埠映射可以修復此問題。

連接埠已在使用中

此任務的任務定義在其連接埠映射中使用相同的連接埠，做為已在所選容器執行個體上執行的任務。服務事件訊息會將具有所選容器執行個體 ID 作為以下訊息的一部分。

```
The closest matching container-instance is already using a port required by your
 task.
```

記憶體不足

如果您的任務定義指定 1000 MiB 的記憶體，而您叢集中的容器執行個體每個都有 1024 MiB 的記憶體，則您每個容器執行個體只能執行一個此任務複本。您可以在任務定義中試驗使用較少的記憶體，以便您每個容器執行個體可以啟動多項任務，或在您的叢集中啟動更多的容器執行個體。

> **Note**
>
> 若您嘗試盡可能為特定執行個體類型的任務提供最多的記憶體，以將資源使用率最大化，請參閱「保留 Amazon ECS Linux 容器執行個體記憶體」。

可用的 ENI 連接點不足

使用 awsvpc 網路模式的每項任務都會收到自己的彈性網路介面 (ENI)，連接到裝載它的容器執行個體。Amazon EC2 執行個體的 ENIs 數量有限制，而且叢集中沒有可用 ENI 容量的容器執行個體。

個別容器執行個體的 ENI 限制取決於以下條件：

- 如果您尚未選擇使用 awsvpcTrunking 帳戶設定，則每個容器執行個體的 ENI 限制取決於執行個體類型。如需詳細資訊，請參閱《Amazon EC2 使用者指南》中的每個執行個體類型每個網路介面的 IP 地址。
- 如果您已選擇加入awsvpcTrunking帳戶設定，但在選擇加入後尚未使用支援的執行個體類型啟動新的容器執行個體，則每個容器執行個體的 ENI 限制仍為預設值。如需詳細資訊，請參閱《Amazon EC2 使用者指南》中的每個執行個體類型每個網路介面的 IP 地址。
- 如果您已選擇使用 awsvpcTrunking 帳戶設定，且已在選擇使用後，使用支援的執行個體類型啟動新的容器執行個體，則會有額外的 ENI 可用。如需詳細資訊，請參閱支援增加 Amazon ECS 容器網路介面的執行個體。

如需選擇使用的 awsvpcTrunking 帳戶設定詳細資訊，請參閱 增加 Amazon ECS Linux 容器執行個體網路介面。

您可以將容器執行個體新增到您的叢集，以提供更多可用的網路轉接器。

容器執行個體遺漏必要的屬性

有些任務定義參數需要在容器執行個體上安裝特定的 Docker 遠端 API 版本。其他，例如記錄驅動程式選項，需要容器執行個體向 ECS_AVAILABLE_LOGGING_DRIVERS 代理組態變數註冊這些日誌驅動程式。如果您的任務定義包含需要特定容器執行個體屬性的參數，而且您沒有任何可滿足此需求的可用容器執行個體，則無法放置任務。

如果您的服務使用使用awsvpc網路模式和 EC2 啟動類型的任務，且您指定的叢集沒有在建立服務awsvpcConfiguration時在 中指定的相同子網路中註冊容器執行個體，則此錯誤的常見原因是。

您可以使用 AWSSupport-TroubleshootECSContainerInstance 執行手冊進行故障診斷。Runbook 會檢閱執行個體的使用者資料是否包含正確的叢集資訊、執行個體描述檔是否包含必要的許可，以及網路組態問題。如需詳細資訊，請參閱 AWS Systems Manager Automation Runbook 參考使用者指南中的 AWSSupport-TroubleshootECSContainerInstance。

有關特定任務定義參數和代理組態變數需要哪些必要屬性的詳細資訊，請參閱「Amazon ECS 任務定義參數」和「Amazon ECS 容器代理程式組態」。

服務 (*service-name*) 無法放置任務。原因：目前無法使用容量。請稍後再試一次，或在不同的可用區域中嘗試。

目前沒有可用的容量可用來執行您的服務。

您可以執行下列任一動作：

- 等到 Fargate 容量或 EC2 容器執行個體可以使用。
- 重新啟動服務並指定其他子網路。

service (*service-name*) 部署失敗：任務無法啟動。

服務中的任務無法啟動。

如需如何偵錯已停止任務的詳細資訊，請參閱Amazon ECS 已停止任務錯誤訊息。

service (*service-name*) 等待 Amazon ECS 代理程式啟動逾時。請檢查位於 /var/log/ecs/ecs-agent.log 的日誌。

適合放置任務之最接近相符容器執行個體中的 Amazon ECS 容器代理中斷連線。如果您可以使用 SSH 連線到容器執行個體，您可以檢查代理程式日誌。如需詳細資訊，請參閱Amazon ECS 容器代理程式日誌組態參數。您也應該確認代理是否在執行個體上執行。如果您使用的是 Amazon ECS 最佳化 AMI，您可以使用下列命令嘗試停用並重新啟動代理。

- 對於 Amazon ECS 最佳化 Amazon Linux 2 AMI

```
sudo systemctl restart ecs
```

- 對於 Amazon ECS 最佳化 Amazon Linux AMI

```
sudo stop ecs && sudo start ecs
```

服務 (*service-name*) 任務集 (*taskSet-ID*) 在目標群組 (*targetGroup-ARN*) 中運作不良），因為 **TARGET GROUP IS NOT FOUND**。

服務的任務集運作狀態檢查失敗，因為找不到目標群組。您應該刪除並重新建立服務。除非已刪除對應的 Amazon ECS 服務，否則請勿刪除任何 Elastic Load Balancing 目標群組。

服務 (*service-name*) 任務集 (*taskSet-ID*) 在目標群組 (*targetGroup-ARN)* 中運作不良），因為 **TARGET IS NOT FOUND**。

服務的任務集運作狀態檢查失敗，因為找不到目標。

Amazon ECS 可用區域服務重新平衡服務事件訊息

以下是您可能會看到的服務事件訊息範例。

service *(service-name*) 與*#### 1######*、*#### 2* 中的*####*和*#### 3* 中的*#####*不平衡。AZ 重新平衡進行中。

當任務數量未平均分散到可用區域時，服務排程器會傳送service (*service-name*) is not AZ balanced服務事件。無需採取任何動作。這是資訊事件。

service (*service-name*) 在*#### 1* 中與*####*任務、*#### 2* 中的*####*任務，以及*#####3 ######*任務之間取得 AZ 平衡。

服務排程器會在可用區域service (*service-name*) is AZ balanced服務重新平衡完成時傳送服務事件。無需採取任何動作。這是資訊事件。

service-name 已在*#####*開始將 ***number-tasks*** 任務重新平衡至 AZ： ***task-ids***。

由於服務重新平衡，服務排程器在*#####*啟動任務時，已傳送 service*-name*/*task-set-name* 已在可用區域服務事件中啟動*##*任務。無需採取任何動作。這是資訊事件。

service-name 已停止在*#####*執行任務的 ***number-tasks***，因為 AZ 重新平衡：***task-id***。

服務排程器傳送 *service-name*/*task-set-name* 已在*####*服務事件中因服務重新平衡而停止可用區域中的任務時，開始在可用區域中執行*##*任務。無需採取任何動作。這是資訊事件。

service (*service-name*) 無法將任務放置在*#####*，因為沒有容器執行個體符合其所有需求。

服務排程器傳送*####*無法在*####*服務事件中放置任務，因為沒有容器執行個體符合其所有需求。若要解決問題，請在可用區域中啟動執行個體。

service (*service-name*) 無法在*#####*放置任務。

當您使用 Fargate 啟動類型且沒有可用容量時，服務排程器傳送*#####*無法在*####*服務事件中放置任務。

您可以在錯誤訊息的可用區域中新增其他子網路，或聯絡 支援 以取得額外的容量。

service (*service-name*) 無法重新平衡 AZ，因為 *task-set-name* 因*#*無法縮減規模。

服務排程器傳送*#####*無法重新平衡 AZ，因為使用任務縮減保護時，任務*task-set-name*因服務事件*###*縮減。

您可以執行下列任一動作：

- 等待目前任務的保護到期，然後終止這些任務。
- 決定哪些任務可以停止，並使用 `UpdateTaskProtection` API 搭配 設定為 `protectionEnabled`的選項`false`，以取消設定這些任務的保護。
- 將服務的所需任務計數提高到超過受保護任務的數目。

service (*service-name*) 已停止 AZ 重新平衡。

當可用區域重新平衡操作停止時，服務排程器會傳送*####*已停止的 AZ 重新平衡服務事件。這是資訊事件。Amazon ECS 會傳送提供詳細資訊的其他事件。

對 Amazon ECS 中的服務負載平衡器進行故障診斷

Amazon ECS 服務可在 Elastic Load Balancing 負載平衡器註冊任務。負載平衡器組態錯誤是常見的任務停止原因。如果您的停止任務是服務使用負載平衡器所啟動，請考慮以下可能的原因。

Amazon ECS 服務連結角色不存在

Amazon ECS 服務連結的角色可讓 Amazon ECS 服務向 Elastic Load Balancing 負載平衡器註冊容器執行個體。必須在您的帳戶中建立服務連結角色。如需詳細資訊，請參閱使用 Amazon ECS 的服務連結角色。

容器執行個體安全群組

如果您的容器映射到您容器執行個體的連接埠 80，則您的容器執行個體安全群組必須允許連接埠 80 傳入流量，以讓負載平衡器的運作狀態檢查通過。

Elastic Load Balancing 負載平衡器未針對所有可用區域設定

您的負載平衡器應該設定使用一個區域中的所有可用區域，或至少是您容器執行個體所在的所有可用區域。如果服務使用負載平衡器，並在位於負載平衡器未設定使用之可用區域中的容器執行個體上啟動任務，則任務永遠不會通過運作狀態檢查。這會導致任務遭到刪除。

Elastic Load Balancing 負載平衡器運作狀態檢查設定錯誤

負載平衡器的運作狀態檢查參數可能太過侷限或指向不存在的資源。如果容器執行個體判定運作狀態不佳，則會從負載平衡器中移除。請務必確認為您的服務負載平衡器正確設定以下參數。

Ping 連接埠

負載平衡器運作狀態檢查的 Ping Port (Ping 連接埠) 值，是負載平衡器檢查判斷運作狀態是否良好之容器執行個體的連接埠。如果此連接埠設定錯誤，負載平衡器可能會從本身取消註冊您的容器執行個體。此連接埠應設定使用您處理運作狀態檢查之服務任務定義容器的 `hostPort` 值。

Ping 路徑

這是負載平衡器運作狀態檢查的一部分。這是您應用程式上的端點，可在應用程式運作狀態良好時傳回成功的狀態碼 (例如 200)。此值通常設為 `index.html`，但如果您的服務不回應該請求，則運作狀態檢查會失敗。如果您的容器沒有 `index.html` 檔案，您可以將此設定為 `/`，以容器執行個體的基本 URL 為目標。

回應逾時

這是您的容器必須傳回運作狀態檢查 ping 回應的時間。如果此值低於回應所需的時間，則運作狀態檢查會失敗。

運作狀態檢查間隔

這是運作狀態檢查 ping 之間的時間。運作狀態檢查的間隔愈短，您的容器執行個體就愈快到達 Unhealthy Threshold (狀態不良閾值)。

狀態不良閾值

這是您的運作狀態檢查失敗前，您的容器執行個體視為運作狀態不良的次數。如果您的運作狀態不佳閾值為 2，且運作狀態檢查間隔為 30 秒，則您的任務有 60 秒的時間回應運作狀態檢查 ping，之後才會假設運作狀態不佳。您可以提高運作狀態不良閾值或運作狀態檢查間隔，讓您的任務有更多的時間回應。

無法更新服務## : 負載平衡器容器名稱或任務定義中變更的連接埠

如果您的服務使用負載平衡器，您可以使用 AWS CLI 或 SDK 來修改負載平衡器組態。如需有關如何修改組態的資訊，請參閱 Amazon Elastic Container Service API 參考 中的 UpdateService。如

果您為服務更新任務定義，則在負載平衡器組態中指定的容器名稱和容器連接埠必須保留在任務定義中。

您已達到可同時執行的任務數量限制。

若為新帳戶，您的配額可能低於服務配額。您可以在 Service Quotas 主控台檢視您的帳戶的服務配額。若要請求提高配額，請參閱《Service Quotas 使用者指南》中的請求提高配額。

對 Amazon ECS 中的服務自動擴展進行故障診斷

Application Auto Scaling 會在 Amazon ECS 部署進行中時關閉縮減程序，並在部署完成後恢復。不過，除非在部署期間暫停，否則向外擴展程序會繼續發生。如需詳細資訊，請參閱暫停和繼續擴展 Application Auto Scaling。

對 Amazon ECS 任務定義無效的 CPU 或記憶體錯誤進行故障診斷

使用 Amazon ECS API 註冊任務定義時 AWS CLI，或者，如果您指定無效的 `cpu`或 `memory`值，則會傳回下列錯誤。

```
An error occurred (ClientException) when calling the RegisterTaskDefinition operation:
 Invalid 'cpu' setting for task.
```

> **Note**
>
> 使用 Terraform 時，可能會傳回下列錯誤。
>
> ```
> Error: ClientException: No Fargate configuration exists for given values.
> ```

若要解決這個問題，您必須為您任務定義中的任務 CPU 和記憶體指定受支援的值。此`cpu`值可以在任務定義中以 CPU 單位或 vCPUs表示。它會轉換為整數，表示在註冊任務定義時 CPU 單位。`memory`值可以在任務定義中以 MiB 或 GB 表示。註冊任務定義時，它會轉換為表示 MiB 的整數。

對於`FARGATE`為 `requiresCompatibilities` 參數指定的任務定義 （即使`EC2`也指定了），您必須使用下表中的其中一個值。這些值決定 CPU 和記憶體參數支援的數值範圍。

對於在 Fargate 上託管的任務，下表顯示了有效的 CPU 和記憶體組合。JSON 檔案中的記憶體值是以 MiB 為單位。您可以將 GB 值乘以 1024，將 GB值 轉換為 MiB。例如 1 GB = 1024 MiB。

CPU 數值	記憶體數值	AWS Fargate 支援的作業系統
256 (.25 vCPU)	512 MiB、1 GB、2 GB	Linux
512 (.5 vCPU)	1 GB、2 GB、3 GB、4 GB	Linux
1024 (1 vCPU)	2 GB、3 GB、4 GB、5 GB、6 GB、7 GB、8 GB	Linux、Windows
2048 (2 vCPU)	介於 4 GB 與 16 GB 之間，以 1 GB 為單位遞增	Linux、Windows
4096 (4 vCPU)	介於 8 GB 與 30 GB 之間，以 1 GB 為單位遞增	Linux、Windows
8192 (8 vCPU) **Note** 此選項需要 Linux 平台 `1.4.0` 或更新版本。	介於 16 GB 與 60 GB 之間，以 4 GB 為單位遞增	Linux
16384 (16vCPU) **Note** 此選項需要 Linux 平台 `1.4.0` 或更新版本。	介於 32 GB 與 120 GB 之間，以 8 GB 為單位遞增	Linux

對於託管在 Amazon EC2 上的任務，支援的任務 CPU 值介於 0.25 vCPUs和 10 vCPUs之間。

Amazon ECS 使用 CPU 期間和 CPU 配額來控制任務大小的 CPU 硬性限制。當您在任務定義中指定 vCPU 時，Amazon ECS 會將值轉譯為 CPU 期間，以及套用至 的 CPU 配額設定`cgroup`。

CPU 配額控制在特定 CPU `cgroup`期間授予 的 CPU 時間量。這兩個設定都以微秒為單位。當 CPU 配額等於 CPU 期間時，表示 `cgroup`可在一個 vCPU 上執行高達 100% 的 （或多個 vCPUs 總計為 100% 的任何其他分數）。CPU 配額上限為 1000000us，CPU 期間則為 1ms。您可以使用這些值來設

定 CPU 計數的限制。當您在不變更 CPU 配額的情況下變更 CPU 期間時，您具有與任務定義中所指定不同的有效限制。

100 毫秒期間允許 0.125 到 10 範圍內vCPUs。

> **Note**
>
> Windows 容器會忽略任務層級的 CPU 和記憶體參數。

檢視 Amazon ECS 容器代理程式日誌

Amazon ECS 會將日誌存放在您容器執行個體的 `/var/log/ecs` 資料夾中。您可以從 Amazon ECS 容器代理程式和 `ecs-init` 服務中取得日誌，該服務控制容器執行個體上的代理程式狀態 (開始/停止)。您可以使用 SSH 連線到容器執行個體來檢視這些日誌檔。

> **Note**
>
> 如果您不確定如何收集您容器執行個體中的所有日誌，您可以使用 Amazon ECS 日誌收集器。如需詳細資訊，請參閱使用 Amazon ECS 日誌收集器收集容器日誌 。

Linux 作業系統

`ecs-init` 程序會將日誌存放在 `/var/log/ecs/ecs-init.log`。

`ecs-init.log` 檔案包含容器代理程式生命週期管理、組態和引導的相關資訊。

您可以使用下列命令來檢視日誌檔案。

```
cat /var/log/ecs/ecs-init.log
```

輸出：

```
2018-02-16T18:13:54Z [INFO] pre-start
2018-02-16T18:13:56Z [INFO] start
2018-02-16T18:13:56Z [INFO] No existing agent container to remove.
2018-02-16T18:13:56Z [INFO] Starting Amazon Elastic Container Service Agent
```

Windows 作業系統

您可以使用適用於 Windows 的 Amazon ECS 日誌收集器。如需詳細資訊，請參閱 Github 上的 Amazon ECS Logs Collector for Windows。

1. 連線到您的執行個體。
2. 開啟 PowerShell，然後使用管理權限執行下列命令。命令會下載指令碼並收集日誌。

```
Invoke-WebRequest -OutFile ecs-logs-collector.ps1 https://
raw.githubusercontent.com/awslabs/aws-ecs-logs-collector-for-windows/master/ecs-
logs-collector.ps1
.\ecs-logs-collector.ps1
```

您可以開啟 Amazon ECS 代理程式和 Docker 協助程式的偵錯記錄。此選項允許指令碼在開啟偵錯模式之前收集日誌。指令碼會重新啟動 Docker 協助程式和 Amazon ECS 代理程式，然後終止執行個體上執行的所有容器。在執行下列命令之前，請耗盡容器執行個體，並將任何重要任務移至其他容器執行個體。

執行下列命令以開啟記錄。

```
.\ecs-logs-collector.ps1 -RunMode debug
```

使用 Amazon ECS 日誌收集器收集容器日誌

如果您不確定如何收集您容器執行個體中的所有各種日誌，您可以使用 Amazon ECS 日誌收集器。此工具的 Linux 和 Windows 版本都可在 GitHub 上取得。指令碼會收集一般作業系統日誌，以及 Docker 和 Amazon ECS 容器代理程式日誌，這有助於對 AWS 支援 案例進行故障診斷。然後，它會將所收集的資訊壓縮並封存為單一檔案，輕鬆共用於診斷目的。也支援為 Docker 常駐程式和 Amazon Linux 變體上的 Amazon ECS 容器代理程式 (例如 Amazon ECS 最佳化 AMI) 啟用除錯模式。目前，Amazon ECS 日誌收集器支援下列作業系統：

- Amazon Linux
- Red Hat Enterprise Linux 7
- Debian 8
- Ubuntu 14.04
- Ubuntu 16.04
- Ubuntu 18.04

- Windows Server 2016

> **Note**
>
> Amazon ECS 日誌收集器的原始程式碼可在 GitHub 取得，同時提供 Linux 和 Windows 版本。我們鼓勵您為想要進行的變更提交提取請求。然而，Amazon Web Services 目前不支援執行此軟體經修改的複本。

下載並執行 Linux 版的 Amazon ECS 日誌收集器

1. 連線到您的容器執行個體。
2. 下載 Amazon ECS 日誌收集器指令碼。

```
curl -O https://raw.githubusercontent.com/awslabs/ecs-logs-collector/master/ecs-logs-collector.sh
```

3. 執行指令碼以收集日誌並建立封存。

> **Note**
>
> 若要啟用 Docker 協助程式和 Amazon ECS 容器代理程式的偵錯模式，請將 `--mode=enable-debug`選項新增至下列命令。這可能會重新啟動 Docker 協助程式，這會刪除執行個體上執行的所有容器。請考慮耗盡容器執行個體，並將任何重要的任務先移到其他容器執行個體，再啟用除錯模式。如需詳細資訊，請參閱耗盡 Amazon ECS 容器執行個體。

```
[ec2-user ~]$ sudo bash ./ecs-logs-collector.sh
```

> **Important**
>
> 建議您編輯日誌，並從檔案中移除所有敏感資料。您可以搜尋已知資料，也可以在 `AWS_SESSION_TOKEN`檔案中搜尋環境變數，例如 `AWS_ACCESS_KEY_ID`、`AWS_SECRET_ACCESS_KEY` 和 。

在您執行指令碼之後，您可以在指令碼建立的 `collect` 資料夾中檢查收集到的日誌。`collect.tgz` 檔案是所有日誌的壓縮封存檔，您可以與 共用以 AWS 支援 尋求診斷協助。

下載並執行 Windows 版的 Amazon ECS 日誌收集器

1. 連線到您的容器執行個體。如需詳細資訊，請參閱《Amazon EC2 使用者指南》中的使用 RDP 連線至 Windows 執行個體。
2. 使用 PowerShell 下載 Amazon ECS 日誌收集器指令碼。

```
Invoke-WebRequest -OutFile ecs-logs-collector.ps1 https://
raw.githubusercontent.com/awslabs/aws-ecs-logs-collector-for-windows/master/ecs-
logs-collector.ps1
```

3. 執行指令碼以收集日誌並建立封存。

> **Note**
>
> 若要啟用 Docker 協助程式和 Amazon ECS 容器代理程式的偵錯模式，請將 `-RunMode debug`選項新增至下列命令。這會重新啟動 Docker 常駐程式，它會終止在執行個體上執行的所有容器。請考慮耗盡容器執行個體，並將任何重要的任務先移到其他容器執行個體，再啟用除錯模式。如需詳細資訊，請參閱耗盡 Amazon ECS 容器執行個體。

```
.\ecs-logs-collector.ps1
```

> **Important**
>
> 建議您編輯日誌，並從檔案中移除所有敏感資料。您可以搜尋已知資料，也可以在 `AWS_SESSION_TOKEN`檔案中搜尋環境變數，例如 `AWS_ACCESS_KEY_ID`、`AWS_SECRET_ACCESS_KEY` 和 。

在您執行指令碼之後，您可以在指令碼建立的 `collect` 資料夾中檢查收集到的日誌。`collect.tgz` 檔案是所有日誌的壓縮封存檔，您可以與 AWS Support 共用，以取得診斷協助。

使用代理程式簡介擷取 Amazon ECS 診斷詳細資訊

Amazon ECS 代理程式自我檢查 API 提供有關 Amazon ECS 代理程式和容器執行個體整體狀態的資訊。

您可以使用 代理程式自我檢查 API 來取得任務中容器的 Docker ID。您可以使用 SSH 連線到容器執行個體，使用代理自我檢查 API。

Important

您的容器執行個體必須擁有允許存取 Amazon ECS 以連接自我檢查 API 的 IAM 角色。如需詳細資訊，請參閱Amazon ECS 容器執行個體 IAM 角色。

下列範例顯示兩個任務，一個目前正在執行，另一個已停止。

Note

下列命令會透過 輸送python -mjson.tool，以提高可讀性。

```
curl http://localhost:51678/v1/tasks | python -mjson.tool
```

輸出：

```
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  1095  100  1095    0     0   117k      0 --:--:-- --:--:-- --:--:--  133k
{
    "Tasks": [
        {
            "Arn": "arn:aws:ecs:us-west-2:aws_account_id:task/090eff9b-1ce3-4db6-848a-
a8d14064fd24",
            "Containers": [
                {
                    "DockerId":
 "189a8ff4b5f04affe40e5160a5ffadca395136eb5faf4950c57963c06f82c76d",
                    "DockerName": "ecs-console-sample-app-static-6-simple-
app-86caf9bcabe3e9c61600",
                    "Name": "simple-app"
```

```
                },
                {
                    "DockerId":
 "f7f1f8a7a245c5da83aa92729bd28c6bcb004d1f6a35409e4207e1d34030e966",
                    "DockerName": "ecs-console-sample-app-static-6-busybox-
ce83ce978a87a890ab01",
                    "Name": "busybox"
                }
            ],
            "Family": "console-sample-app-static",
            "KnownStatus": "STOPPED",
            "Version": "6"
        },
        {
            "Arn": "arn:aws:ecs:us-west-2:aws_account_id:task/1810e302-eaea-4da9-
a638-097bea534740",
            "Containers": [
                {
                    "DockerId":
 "dc7240fe892ab233dbbcee5044d95e1456c120dba9a6b56ec513da45c38e3aeb",
                    "DockerName": "ecs-console-sample-app-static-6-simple-app-
f0e5859699a7aecfb101",
                    "Name": "simple-app"
                },
                {
                    "DockerId":
 "096d685fb85a1ff3e021c8254672ab8497e3c13986b9cf005cbae9460b7b901e",
                    "DockerName": "ecs-console-sample-app-static-6-
busybox-92e4b8d0ecd0cce69a01",
                    "Name": "busybox"
                }
            ],
            "DesiredStatus": "RUNNING",
            "Family": "console-sample-app-static",
            "KnownStatus": "RUNNING",
            "Version": "6"
        }
    ]
}
```

在上述範例中，停止的任務 (*090eff9b-1ce3-4db6-848a-a8d14064fd24*) 有兩個容器。您可以使用 docker inspect ***container-ID*** 以檢視每個容器的詳細資訊。如需詳細資訊，請參閱Amazon ECS 容器簡介。

Amazon ECS 中的 Docker 診斷

Docker 提供數種診斷工具，可協助您對容器和任務問題進行故障診斷。如需所有可用 Docker 命令列公用程式的詳細資訊，請參閱 Docker 文件中的 Docker CLI 參考。您可以使用 SSH 連線到容器執行個體，存取 Docker 命令列公用程式。

Docker 容器回報的結束代碼，也可以提供一些診斷資訊 (例如，結束代碼 137 表示容器收到 SIGKILL 訊號)。如需詳細資訊，請參閱 Docker 文件中的「結束狀態」。

在 Amazon ECS 中列出 Docker 容器

您可以在您的容器執行個體中使用 docker ps 命令列出執行中的容器。在下列範例中，只有 Amazon ECS 容器代理程式正在執行。如需詳細資訊，請參閱 Docker 文件中的 docker ps。

```
docker ps
```

輸出：

```
CONTAINER ID        IMAGE                            COMMAND             CREATED
      STATUS              PORTS                             NAMES
cee0d6986de0        amazon/amazon-ecs-agent:latest   "/agent"            22 hours ago
      Up 22 hours         127.0.0.1:51678->51678/tcp   ecs-agent
```

您可以使用 docker ps -a 命令查看所有容器 (甚至是停止或終止的容器)。這有助於列出未預期停止的容器。在下列範例中，容器 f7f1f8a7a245 已在 9 秒前結束，所以 -a 旗標會出現在 docker ps 輸出中。

```
docker ps -a
```

輸出：

```
CONTAINER ID        IMAGE                                              COMMAND
 CREATED             STATUS                          PORTS                          NAMES
db4d48e411b1        amazon/ecs-emptyvolume-base:autogenerated   "not-applicable"
  19 seconds ago                                                                     ecs-
console-sample-app-static-6-internalecs-emptyvolume-source-c09288a6b0cba8a53700
f7f1f8a7a245        busybox:buildroot-2014.02                   "\"sh -c '/bin/sh -c
  22 hours ago        Exited (137) 9 seconds ago                                     ecs-
console-sample-app-static-6-busybox-ce83ce978a87a890ab01
```

```
189a8ff4b5f0        httpd:2                                              "httpd-foreground"
  22 hours ago        Exited (137) 40 seconds ago                                      ecs-
console-sample-app-static-6-simple-app-86caf9bcabe3e9c61600
0c7dca9321e3        amazon/ecs-emptyvolume-base:autogenerated   "not-applicable"
  22 hours ago                                                                         ecs-
console-sample-app-static-6-internalecs-emptyvolume-source-90fefaa68498a8a80700
cee0d6986de0        amazon/amazon-ecs-agent:latest              "/agent"
  22 hours ago        Up 22 hours                    127.0.0.1:51678->51678/tcp   ecs-
agent
```

在 Amazon ECS 中檢視 Docker 日誌

您可以使用 docker logs 命令檢視容器的 STDOUT 和 STDERR 串流。在此範例中，會顯示 *dc7240fe892a* 容器的日誌，並為簡潔起見使用 head 命令以管道輸送。如需詳細資訊，請前往 Docker 文件中的 docker logs。

> **Note**
>
> 若您使用預設的 `json` 日誌驅動程式，則 Docker 日誌僅適用於容器執行個體。如果您已使用 `awslogs` 日誌驅動程式設定任務，即可在 CloudWatch Logs 中找到容器日誌。如需詳細資訊，請參閱將 Amazon ECS 日誌傳送至 CloudWatch 。

```
docker logs dc7240fe892a | head
```

輸出：

```
AH00558: httpd: Could not reliably determine the server's fully qualified domain name,
 using 172.17.0.11. Set the 'ServerName' directive globally to suppress this message
AH00558: httpd: Could not reliably determine the server's fully qualified domain name,
 using 172.17.0.11. Set the 'ServerName' directive globally to suppress this message
[Thu Apr 23 19:48:36.956682 2015] [mpm_event:notice] [pid 1:tid 140327115417472]
 AH00489: Apache/2.4.12 (Unix) configured -- resuming normal operations
[Thu Apr 23 19:48:36.956827 2015] [core:notice] [pid 1:tid 140327115417472] AH00094:
 Command line: 'httpd -D FOREGROUND'
10.0.1.86 - - [23/Apr/2015:19:48:59 +0000] "GET / HTTP/1.1" 200 348
10.0.0.154 - - [23/Apr/2015:19:48:59 +0000] "GET / HTTP/1.1" 200 348
10.0.1.86 - - [23/Apr/2015:19:49:28 +0000] "GET / HTTP/1.1" 200 348
10.0.0.154 - - [23/Apr/2015:19:49:29 +0000] "GET / HTTP/1.1" 200 348
10.0.1.86 - - [23/Apr/2015:19:49:50 +0000] "-" 408 -
```

```
10.0.0.154 - - [23/Apr/2015:19:49:50 +0000] "-" 408 -
10.0.1.86 - - [23/Apr/2015:19:49:58 +0000] "GET / HTTP/1.1" 200 348
10.0.0.154 - - [23/Apr/2015:19:49:59 +0000] "GET / HTTP/1.1" 200 348
10.0.1.86 - - [23/Apr/2015:19:50:28 +0000] "GET / HTTP/1.1" 200 348
10.0.0.154 - - [23/Apr/2015:19:50:29 +0000] "GET / HTTP/1.1" 200 348
time="2015-04-23T20:11:20Z" level="fatal" msg="write /dev/stdout: broken pipe"
```

在 Amazon ECS 中檢查 Docker 容器

如果您有容器的 Docker ID，您可以使用 docker inspect 命令來檢查該 Docker ID。檢查容器可提供最詳細的容器啟動環境檢視。如需詳細資訊，請參閱 Docker 文件中的 docker inspect。

```
docker inspect dc7240fe892a
```

輸出：

```
[{
    "AppArmorProfile": "",
    "Args": [],
    "Config": {
        "AttachStderr": false,
        "AttachStdin": false,
        "AttachStdout": false,
        "Cmd": [
            "httpd-foreground"
        ],
        "CpuShares": 10,
        "Cpuset": "",
        "Domainname": "",
        "Entrypoint": null,
        "Env": [
            "PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/
local/apache2/bin",
            "HTTPD_PREFIX=/usr/local/apache2",
            "HTTPD_VERSION=2.4.12",
            "HTTPD_BZ2_URL=https://www.apache.org/dist/httpd/httpd-2.4.12.tar.bz2"
        ],
        "ExposedPorts": {
            "80/tcp": {}
        },
        "Hostname": "dc7240fe892a",
...
```

在 Amazon ECS 中設定 Docker 協助程式的詳細輸出

如果您無法使用 Docker 容器或映像，您可以在 Docker 協助程式上開啟偵錯模式。使用偵錯可提供更多常駐程式的詳細輸出，您可以使用它來擷取從容器登錄檔傳送的錯誤訊息，例如 Amazon ECR。

> **Important**
>
> 此程序是針對 Amazon ECS 最佳化 Amazon Linux AMI 所寫入。如需其他作業系統，請參閱 Docker 文件中的啟用偵錯和控制，以及Docker使用 設定 systemd 。

在 Amazon ECS 最佳化 Amazon Linux AMI 上使用 Docker 協助程式偵錯模式

1. 連線到您的容器執行個體。
2. 使用文字編輯器開啟 Docker 選項檔案，例如 vi。對於 Amazon ECS 最佳化 Amazon Linux AMI，Docker 選項檔案位於 `/etc/sysconfig/docker`。
3. 尋找 Docker 選項陳述式，並在字串中新增以引號括住的 `-D` 選項。

   > **Note**
   >
   > 如果 Docker 選項陳述式以 # 開頭，請移除該字元以取消陳述式註解並啟用選項。

   對於 Amazon ECS 最佳化 Amazon Linux AMI，Docker 選項陳述式稱為 `OPTIONS`。例如：

   ```
   # Additional startup options for the Docker daemon, for example:
   # OPTIONS="--ip-forward=true --iptables=true"
   # By default we limit the number of open files per container
   OPTIONS="-D --default-ulimit nofile=1024:4096"
   ```
4. 儲存檔案並結束您的文字編輯器。
5. 重新啟動 Docker 常駐程式。

   ```
   sudo service docker restart
   ```

   其輸出如下：

   ```
   Stopping docker:                                           [  OK  ]
   ```

```
Starting docker: .                                         [  OK  ]
```

6. 重新啟動 Amazon ECS 代理。

```
sudo service ecs restart
```

您的 Docker 日誌現在應該會顯示更多的詳細輸出。

```
time="2015-12-30T21:48:21.907640838Z" level=debug msg="Unexpected response from
 server: \"{\\\"errors\\\":[{\\\"code\\\":\\\"DENIED\\\",\\\"message\\\":\\\"User:
 arn:aws:sts::1111:assumed-role/ecrReadOnly/i-abcdefg is not authorized to perform:
 ecr:InitiateLayerUpload on resource: arn:aws:ecr:us-east-1:1111:repository/nginx_test
\\\"}]}\\n\" http.Header{\"Connection\":[]string{\"keep-alive\"}, \"Content-Type\":
[]string{\"application/json; charset=utf-8\"}, \"Date\":[]string{\"Wed, 30 Dec 2015
 21:48:21 GMT\"}, \"Docker-Distribution-Api-Version\":[]string{\"registry/2.0\"},
 \"Content-Length\":[]string{\"235\"}}"
```

在 Amazon ECS **API error (500): devmapper**中對 Docker 進行故障診斷

以下 Docker 錯誤指出，您容器執行個體上的精簡集區儲存已滿，Docker 常駐程式無法建立新的容器：

```
CannotCreateContainerError: API error (500): devmapper: Thin Pool has 4350 free data
 blocks which is less than minimum required 4454 free data blocks. Create more free
 space in thin pool or use dm.min_free_space option to change behavior
```

根據預設，2015.09.d 版和更新版本的 Amazon ECS 最佳化 Amazon Linux AMI 會使用作業系統的 8 GiB 磁碟區啟動，此作業系統連接於 /dev/xvda，而且掛載為檔案系統的根目錄。另有一個 22 GiB 的磁碟區連接在 /dev/xvdcz，Docker 用於儲存映像和中繼資料。如果這個儲存空間已滿，則 Docker 常駐程式無法建立新的容器。

要將儲存體新增到您的容器執行個體，最簡單方式是終止現有的執行個體，並啟動有較大資料儲存體磁碟區的新執行個體。不過，如果您無法執行此作業，您可以遵循 Amazon ECS 最佳化 Linux AMIs 中的程序，在 Docker 所用的磁碟區群組新增儲存，並擴展其邏輯磁碟區。

如果您的容器執行個體儲存體太快填滿，您可以採取幾個動作降低此效果：

- 若要檢視精簡輪詢資訊，請在您的容器執行個體上執行下列命令：

```
docker info
```

- (Amazon ECS 容器代理程式 1.8.0 及更新版本） 您可以減少容器執行個體上停止或結束容器的時間。ECS_ENGINE_TASK_CLEANUP_WAIT_DURATION 代理組態變數會將時間設為從任務停止等到 Docker 容器移除 (此值預設為 3 小時)。這會移除 Docker 容器的資料。如果此值設定過低，您可能無法在移除日誌之前檢查已停止的容器或檢視日誌。如需詳細資訊，請參閱Amazon ECS 容器代理程式組態。
- 您可以從容器執行個體中移除未執行的容器和未使用的映像。您可以使用以下範例命令，手動移除已停止的容器和未使用的映像。容器在刪除之後即無法檢查，而已刪除的映像必須得再次提取，才能從中啟動新的容器。

  若要移除非執行中的容器，請在您的容器執行個體上執行以下命令：

```
docker rm $(docker ps -aq)
```

  若要移除未使用的映像，請在您的容器執行個體上執行以下命令：

```
docker rmi $(docker images -q)
```

- 您可以在容器內移除未使用的資料區塊。您可以使用下列命令在任何執行中的容器上執行 fstrim，並捨棄容器檔案系統未使用的任何資料區塊。

```
sudo sh -c "docker ps -q | xargs docker inspect --format='{{ .State.Pid }}' | xargs -IZ fstrim /proc/Z/root/"
```

對 Amazon ECS Exec 問題進行故障診斷

以下是故障診斷注意事項，可協助診斷您在使用 ECS Exec 時可能會出現錯誤的原因。

使用 Exec Checker 驗證

ECS Exec Checker 指令碼提供一種方法來驗證和驗證您的 Amazon ECS 叢集和任務是否符合使用 ECS Exec 功能的先決條件。ECS Exec Checker 指令碼會代表您呼叫各種 APIs，以驗證您的 AWS CLI 環境和叢集和任務是否已準備好接受 ECS Exec。工具需要最新版本的 AWS CLI 和 `jq` 可供使用。如需詳細資訊，請參閱 GitHub 上的 ECS Exec Checker。

呼叫 execute-command 時發生錯誤

如果發生 The execute command failed 錯誤，則可能的原因如下。

- 此任務沒有必要的許可。確認用來啟動任務的任務定義已定義任務 IAM 角色，且該角色具有必要的許可。如需詳細資訊，請參閱ECS Exec 許可。
- SSM 代理程式未安裝或未執行。
- Amazon ECS 有一個界面 Amazon VPC 端點，但沒有 Systems Manager Session Manager 的界面。

對 Amazon ECS Anywhere 問題進行故障診斷

Amazon ECS Anywhere 支援將內部部署伺服器或虛擬機器 (VM) 等外部執行個體註冊到您的 Amazon ECS 叢集。以下是您可能會遇到的常見問題，及其一般故障診斷建議。

主題

外部執行個體註冊問題

在 Amazon ECS 叢集註冊外部執行個體時，必須符合以下要求：

- 必須擷取由啟用 ID 和啟用碼組成的 AWS Systems Manager 啟用。您可以使用它來將外部執行個體註冊為 Systems Manager 受管執行個體。請求 Systems Manager 啟用時，請指定註冊限制和過期日期。註冊限制指定使用啟用時可註冊的執行個體數目上限。註冊限制的預設值為1執行個體。過期日期指定啟用的過期時間。預設值為 24 小時。如果您用來註冊外部執行個體的 Systems Manager 啟用無效，則請求新的執行個體。如需詳細資訊，請參閱將外部執行個體註冊至 Amazon ECS 叢集。
- IAM 政策用於為您的外部執行個體提供與 AWS API 操作通訊所需的許可。如果未正確建立此受管政策，且不包含必要的許可，則外部執行個體註冊會失敗。如需詳細資訊，請參閱Amazon ECS Anywhere IAM 角色。
- Amazon ECS 提供安裝指令碼，可在外部執行個體上安裝 Docker、Amazon ECS 容器代理程式和 Systems Manager Agent。如果安裝指令碼失敗，則指令碼很可能無法在不發生錯誤的情形下在同一

個執行個體上再次執行。如果發生這種情況，請依照清除程序從執行個體清除 AWS 資源，以便再次執行安裝指令碼。如需詳細資訊，請參閱取消註冊 Amazon ECS 外部執行個體。

> **Note**
>
> 請注意，如果成功請求安裝指令碼並使用 Systems Manager 啟用，則第二次執行安裝指令碼會再次使用 Systems Manager 啟用。這轉而可能會導致您達到該啟用的註冊限制。如果達到此限制，則您必須建立新的啟用。

- 在 GPU 工作負載的外部執行個體上執行安裝指令碼時，如果未正確偵測到或設定 NVIDIA 驅動程式，將發生錯誤。安裝指令碼使用 `nvidia-smi` 命令來確認 NVIDIA 驅動程式的存在。

外部執行個體網路問題

若要傳達任何變更，您的外部執行個體需要 AWS網路連線。如果您的外部執行個體失去其網路連線 AWS，則執行個體上執行的任務仍會繼續執行，除非手動停止。 AWS 還原與 的連線後，Amazon ECS 容器代理程式和 Systems Manager 代理程式在外部執行個體上使用的 AWS 登入資料會自動續約。如需用於外部執行個體與 之間通訊之 AWS 網域的詳細資訊 AWS，請參閱 聯網 。

在外部執行個體上執行任務的問題

如果您的任務或容器無法在外部執行個體上執行，最常見的原因是網路或許可相關問題。如果您的容器從 Amazon ECR 提取其映像，或者設定為將容器日誌傳送到 CloudWatch Logs，則您的任務定義必須指定有效的任務執行 IAM 角色。如果沒有有效的任務執行 IAM 角色，則您的容器將無法啟動。如需網路相關問題的詳細資訊，請參閱 外部執行個體網路問題。

> **Important**
>
> Amazon ECS 提供了 Amazon ECS 日誌收集工具。您可以將其用於從外部執行個體收集日誌，以進行故障診斷。如需詳細資訊，請參閱使用 Amazon ECS 日誌收集器收集容器日誌 。

AWS Fargate 調節配額

AWS Fargate 會針對每個區域的每個 AWS 帳戶，使用字符儲存貯體演算法將 Amazon ECS 任務和 Amazon EKS Pod 啟動率限制為配額 （ 先前稱為限制 ）。使用此演算法，您的帳戶擁有儲存特定數量字符的儲存貯體。儲存貯體中的字符數量代表您在任何指定秒數的速率配額。每個客戶帳戶都有一個任

務和 Pod 字符儲存貯體，會根據客戶帳戶啟動的任務和 Pod 的數量耗盡。此字符儲存貯體具有允許您定期提出更多請求的儲存貯體最大值，以及允許您在需要時保持穩定的請求速率的重新填滿速率。

例如，Fargate 客戶帳戶的任務和 Pod 字符儲存貯體大小為 100 個字符，重新填滿速率為每秒 20 個字符。因此，您可以立即啟動 Amazon ECS 任務和 Amazon EKS Pod (每個客戶帳戶最多 100 個)，持續啟動速率為每秒 20 個 Amazon ECS 任務和 Amazon EKS Pod。

動作	儲存貯體容量上限 (或爆量率)	儲存貯體重新填滿速率 (或持續速率)
隨需型 Amazon ECS 任務和 Amazon EKS Pod 的 Fargate 資源速率配額[1]	100	20
Spot Amazon ECS 任務的 Fargate 資源速率配額	100	20

[1]使用 Amazon EKS 平台版本中提到的平台版本時，僅啟動 Amazon EKS Pod 的帳戶爆量率為 20，Pod 的持續啟動速率為每秒 20 次。

在 Fargate 中調節 **RunTask** API

此外，Fargate 在以單獨配額使用 Amazon ECS RunTask API 啟動任務時會限制請求速率。Fargate 會依區域限制每個 AWS 帳戶的 Amazon ECS RunTask API 請求。您每提出一個請求，就會從儲存貯體中刪除一個字符。我們這樣做是為了協助服務效能，並確保所有 Fargate 客戶的公平使用。無論 API 呼叫來自 Amazon Elastic Container Service 主控台、命令列工具還是第三方應用程式，API 呼叫都會受限於請求配額。呼叫 Amazon ECS RunTask API 的速率配額是每秒 20 次呼叫 (爆量和持續)。不過，每次呼叫此 API 最多可啟動 10 個任務。這表示您可以在一秒鐘內對此 API 進行 10 次呼叫，請求在每次呼叫中啟動 10 個任務，從而實現在一秒鐘內啟動 100 個任務。同理，您也可以對此 API 進行 20 次呼叫，請求在每次呼叫中啟動 5 個任務。如需 Amazon ECS RunTask API 的 API 限流詳細資訊，請參閱《Amazon ECS API 參考》中的 API 請求限流。

實際上，任務和 Pod 啟動速率還取決於其他注意事項，例如要下載和解壓縮的容器映像、運作狀態檢查以及啟用的其他整合 (例如向負載平衡器註冊任務或 Pod)。客戶會根據客戶啟用的功能，看到任務和 Pod 啟動率與先前表示的配額相比有所差異。

在 Fargate 中調整速率配額

您可為 AWS 帳戶請求增加 Fargate 速率調節配額。如需詳細資訊，請參閱《Service Quotas 使用者指南》中的請求增加配額。

處理 Amazon ECS 限流問題

調節錯誤分為兩個主要類別：同步調節和非同步調節。

同步調節

同步調節發生時，您立即收到來自 Amazon ECS 的錯誤回應。當您在執行任務或建立服務時呼叫 Amazon ECS APIs 時，通常會發生此類別。如需相關限流和相關限流限制的詳細資訊，請參閱請求 Amazon ECS API 的限流。

當您的應用程式啟動 API 請求時，例如，使用 AWS CLI 或 AWS SDK，您可以修復 API 限流。您可以透過建構應用程式來處理錯誤，或實作指數退避和抖動策略，搭配 API 呼叫的重試邏輯來執行此操作。如需詳細資訊，請參閱逾時、重試和抖動退避。

如果您使用 AWS SDK，則自動重試邏輯是內建且可設定的。

非同步調節

非同步限流是因為非同步工作流程造成，其中 Amazon ECS 或 AWS CloudFormation 可能代表您呼叫 APIs來佈建資源。請務必了解 Amazon ECS 代您叫用哪些 AWS APIs。例如，針對使用`awsvpc`網路模式的任務叫用 `CreateNetworkInterface` API，並在對註冊到負載平衡器的任務執行運作狀態檢查時叫用 `DescribeTargetHealth` API。

當您的工作負載達到相當大的規模時，這些 API 操作可能會受到調節。也就是說，它們可能會受到足夠限流，以違反 Amazon ECS 或正在呼叫 AWS 服務 的 強制執行的限制。例如，如果您部署數百個 服務，則每個服務都有數百個使用`awsvpc`網路模式並行的任務，Amazon ECS 會叫用 Amazon EC2 API 操作，例如 `CreateNetworkInterface`和 Elastic Load Balancing API 操作，例如 `RegisterTarget`或 `DescribeTargetHealth`，分別註冊彈性網路介面和負載平衡器。這些 API 呼叫可能會超過 API 限制，導致限流錯誤。以下是包含在服務事件訊息中的 Elastic Load Balancing 調節錯誤範例。

```
{
   "userIdentity":{
      "arn":"arn:aws:sts::111122223333:assumed-role/AWSServiceRoleForECS/ecs-service-
scheduler",
```

```
        "eventTime":"2022-03-21T08:11:24Z",
        "eventSource":"elasticloadbalancing.amazonaws.com",
        "eventName":" DescribeTargetHealth ",
        "awsRegion":"us-east-1",
        "sourceIPAddress":"ecs.amazonaws.com",
        "userAgent":"ecs.amazonaws.com",
        "errorCode":"ThrottlingException",
        "errorMessage":"Rate exceeded",
        "eventID":"0aeb38fc-229b-4912-8b0d-2e8315193e9c"
    }
}
```

當這些 API 呼叫與您帳戶中的其他 API 流量共用限制時，即使它們是以服務事件發出，它們仍可能難以監控。

監控限流

請務必識別哪些 API 請求受到限流，以及誰發出這些請求。您可以使用 AWS CloudTrail 來監控限流，並與 CloudWatch、Amazon Athena 和 Amazon EventBridge 整合。您可以設定 CloudTrail 將特定事件傳送至 CloudWatch Logs。CloudWatch Logs 日誌洞察會剖析和分析事件。這可識別限流事件中的詳細資訊，例如進行呼叫的使用者或 IAM 角色，以及進行 API 呼叫的數量。如需詳細資訊，請參閱使用 CloudWatch Logs 監控 CloudTrail 日誌檔案 CloudWatch。

如需 CloudWatch Logs 洞見的詳細資訊，以及如何查詢日誌檔案的說明，請參閱使用 CloudWatch Logs Insights 分析日誌資料。

使用 Amazon Athena，您可以使用標準 SQL 建立查詢和分析資料。例如，您可以建立 Athena 資料表來剖析 CloudTrail 事件。如需詳細資訊，請參閱使用 CloudTrail 主控台為 CloudTrail 日誌建立 Athena 資料表。

建立 Athena 資料表之後，您可以使用 SQL 查詢，例如下列查詢來調查`ThrottlingException`錯誤。

將 *`user-input`* 取代為您的值。

```
select eventname, errorcode,eventsource,awsregion, useragent,COUNT(*) count
FROM cloudtrail_table-name
where errorcode = 'ThrottlingException'
AND eventtime between '2024-09-24T00:00:08Z' and '2024-09-23T23:15:08Z'
group by errorcode, awsregion, eventsource, useragent, eventname
order by count desc;
```

Amazon ECS 也會向 Amazon EventBridge 發出事件通知。有資源狀態變更事件和服務動作事件。其中包括 API 限流事件，例如 ECS_OPERATION_THROTTLED和 SERVICE_DISCOVERY_OPERATION_THROTTLED。如需詳細資訊，請參閱Amazon ECS 服務動作事件。

這些事件可由 等服務取用 AWS Lambda ，以執行動作來回應。如需詳細資訊，請參閱處理 Amazon ECS 事件。

如果您執行獨立任務，則某些 API 操作，例如 RunTask是非同步的，而且不會自動執行重試操作。在這種情況下，您可以使用 等服務 AWS Step Functions 搭配 EventBridge 整合，重試調節或失敗的操作。如需詳細資訊，請參閱管理容器任務 (Amazon ECS、Amazon SNS)。

使用 CloudWatch 監控限流

CloudWatch 提供依 AWS 資源下Usage命名空間的 API 用量監控。這些指標會以類型 API 和指標名稱 CallCount 記錄。您可以建立警示，以便在這些指標達到特定閾值時啟動。如需詳細資訊，請參閱視覺化您的服務配額和設定警示。

CloudWatch 也提供異常偵測。此功能使用機器學習，根據您啟用指標的特定行為來分析和建立基準。如果有不尋常的 API 活動，您可以將此功能與 CloudWatch 警示搭配使用。如需詳細資訊，請參閱使用 CloudWatch 異常偵測。

透過主動監控限流錯誤，您可以聯絡 支援 來增加相關的限流限制，並收到您唯一應用程式需求的指引。

Amazon ECS API 失敗原因

當您透過 Amazon ECS API、 主控台或 AWS CLI 結束並顯示failures錯誤訊息時，下列操作可能有助於對原因進行故障診斷。失敗會傳回與失敗相關聯的資源原因和 Amazon Resource Name (ARN)。

許多資源是區域限定的，所以在使用主控台時，請務必為您的資源設定正確的區域。使用 時 AWS CLI，請確定您的 AWS CLI 命令已使用 `--region` *`region`* 參數傳送至正確的區域。

如需有關 Failure 資料類型結構的詳細資訊，請參閱《Amazon Elastic Container Service API 參考》中的故障。

以下是您在執行 API 命令時可能收到的失敗訊息範例。

API 動作	失敗原因或停止原因	原因
DescribeClusters	MISSING	找不到指定的叢集。請確認叢集名稱的拼字。
DescribeInstances	MISSING	找不到指定的容器執行個體。確認您指定的叢集容器執行個體已註冊，且容器執行個體 ARN 或 ID 都是正確的。
DescribeServices	MISSING	找不到指定的服務。請確認已指定正確的叢集或區域，且服務 ARN 或名稱有效。
DescribeTasks	MISSING	找不到指定的任務。請確認已指定正確的叢集或區域，而且任務 ARN 或 ID 皆有效。
DescribeTasks	TaskFailedToStart: RESOURCE:*	如果 RESOURCE:CPU 發生錯誤，無法在容器執行個體上取得任務請求的 CPU 數量。這通常發生在任務定義中的 CPU 單位需求大於映射到容量提供者的 Auto Scaling 群組中定義的 Amazon EC2 執行個體的 CPU 大小時。您需要檢查容量提供者組態。 如果 RESOURCE:MEMORY 發生錯誤，無法在容器執行個體上取得任務請求的記憶體數量。這通常發生在任務定義中的記憶體數量需求大於 Auto Scaling 群組中映射到容量提供者的 Amazon EC2 執行個體上支援的記憶體時。您需要檢查容量提供者組態。

API 動作	失敗原因或停止原因	原因
	TaskFailedToStart: AGENT	您嘗試啟動任務所在之容器執行個體上的代理程式目前中斷連線。為避免延長任務置放等待時間，請求遭到拒絕。 如需如何對已中斷連線的代理程式進行故障診斷的相關資訊，請參閱 How do I troubleshoot a disconnected Amazon ECS agent (如何對已中斷連線的 Amazon ECS 代理程式進行故障診斷)。
	TaskFailedToStart: MemberOf placement constraint unsatisfied	沒有容器執行個體符合任務定義中定義的置放限制條件。
	TaskFailedToStart: ATTRIBUTE	您的任務定義包含需要特定容器執行個體屬性的參數，但您的容器執行個體不提供此屬性。例如，如果您的任務使用 awsvpc 網路模式，但您使用 ecs.capability.task-eni 屬性指定的子網路中沒有任何執行個體。如需特定任務定義參數和代理組態變數需要哪些必要屬性的詳細資訊，請參閱「Amazon ECS 任務定義參數」和「Amazon ECS 容器代理程式組態」。

API 動作	失敗原因或停止原因	原因
	TaskFailedToStart: NO ACTIVE INSTANCES	容量提供者中沒有作用中的執行個體。如需有關如何管理 Auto Scaling 群組的相關資訊，請參閱《Amazon EC2 Auto Scaling 使用者指南》中的 Auto Scaling 群組。
	TaskFailedToStart: EMPTY CAPACITY PROVIDER	您的叢集中沒有執行個體。這很可能是因為空白容量提供者，或因為容量提供者中的執行個體未註冊至叢集。如需有關如何管理 Auto Scaling 群組的相關資訊，請參閱《Amazon EC2 Auto Scaling 使用者指南》中的 Auto Scaling 群組。
GetTaskProtection	MISSING	找不到指定的任務。確認叢集名稱或 ARN 以及任務 ARN 或 ID 是否有效。
	TASK_NOT_VALID	指定的任務不屬於 Amazon ECS 服務。只有 Amazon ECS 服務受管任務才能受到保護。請確認任務 ARN 或 ID，然後再試一次。

API 動作	失敗原因或停止原因	原因
RunTask 或 StartTask	RESOURCE:*	無法在叢集的容器執行個體上取得任務請求的一或多項資源。如果資源是 CPU、記憶體、連接埠或彈性網路介面，您可能需要新增其他容器執行個體到您的叢集。 若為 RESOURCE:ENI 錯誤，您的叢集不會有任何可用的彈性網路介面連接點，但這對使用 awsvpc 網路模式的任務是必要的。Amazon EC2 執行個體有可以連接到它們的網路界面數量限制，主要網路界面包含在內。如需每個執行個體類型支援多少網路介面的詳細資訊，請參閱《Amazon EC2 使用者指南》中的每個執行個體類型的每個網路介面 IP 地址。 對於 RESOURCE:GPU 錯誤，任務請求的 GPU 數目不可用，而且您可能需要將啟用 GPU 的容器執行個體新增至叢集。如需詳細資訊，請參閱GPU 工作負載的 Amazon ECS 任務定義。

API 動作	失敗原因或停止原因	原因
	AGENT	您嘗試啟動任務所在之容器執行個體上的代理程式目前中斷連線。為避免延長任務置放等待時間，請求遭到拒絕。 如需如何對已中斷連線的代理程式進行故障診斷的相關資訊，請參閱 How do I troubleshoot a disconnected Amazon ECS agent (如何對已中斷連線的 Amazon ECS 代理程式進行故障診斷)。
	LOCATION	您嘗試啟動任務的容器執行個體，位於與您在 awsVpcConfiguration 中指定的子網路不同的可用區域。
	ATTRIBUTE	您的任務定義包含需要特定容器執行個體屬性的參數，但您的容器執行個體不提供此屬性。例如，如果您的任務使用 awsvpc 網路模式，但您使用 ecs.capability.task-eni 屬性指定的子網路中沒有任何執行個體。如需特定任務定義參數和代理組態變數需要哪些必要屬性的詳細資訊，請參閱「Amazon ECS 任務定義參數」和「Amazon ECS 容器代理程式組態」。

API 動作	失敗原因或停止原因	原因
StartTask	MISSING	找不到您嘗試啟動任務的容器執行個體。檢查是否已指定錯誤的叢集或區域，或容器執行個體 ARN 或 ID 是否拼寫錯誤。
	INACTIVE	您嘗試啟動任務所在之容器執行個體，之前已向 Amazon ECS 取消註冊，無法使用。
UpdateTaskProtection	DEPLOYMENT_BLOCKED	無法設定任務保護，因為一或多個受保護的任務阻止服務部署達到穩定狀態。取消現有任務的任務保護設定，或等到任務保護到期。
	MISSING	找不到指定的任務。確認叢集名稱或 ARN 以及任務 ARN 或 ID 是否有效。
	TASK_NOT_VALID	指定的任務不屬於 Amazon ECS 服務。只有 Amazon ECS 服務受管任務才能受到保護。請確認任務 ARN 或 ID，然後再試一次。

> **Note**
>
> 除了此處所述的失敗案例之外，API 操作也可能因為例外狀況而失敗，導致錯誤回應。如需此類例外狀況的清單，請參閱常見錯誤。

Amazon Elastic Container Service 的安全性

的雲端安全性 AWS 是最高優先順序。身為 AWS 客戶，您可以從資料中心和網路架構中受益，該架構旨在滿足最安全敏感組織的需求。

安全性是 AWS 和 之間的共同責任。共同責任模型 將此描述為雲端的安全和雲端內的安全：

- 雲端的安全性 – AWS 負責保護在 Cloud AWS 中執行 AWS 服務的基礎設施。 AWS 也提供您可以安全使用的服務。在 AWS 合規計畫中，第三方稽核員會定期測試並驗證我們的安全功效。若要了解適用於 Amazon Elastic Container Service 的合規計劃，請參閱合規計劃範圍內的AWS 服務。
- 雲端安全性 – 您的責任取決於您使用 AWS 的服務。您也必須對其他因素負責，包括資料的機密性、您公司的要求和適用法律和法規。

本文件有助於您了解如何在使用 Amazon ECS 時套用共同責任模型。下列主題說明如何設定 Amazon ECS 以符合您的安全與合規目標。您也會了解如何使用其他 AWS 服務來協助您監控和保護 Amazon ECS 資源。

主題

- Amazon Elastic Container Service 的身分和存取管理
- 在 Amazon Elastic Container Service 中記錄和監控
- Amazon Elastic Container Service 的合規驗證
- AWS Fargate 聯邦資訊處理標準 (FIPS-140)
- Amazon Elastic Container Service 的基礎設施安全性
- AWS Amazon ECS 的共同責任模型
- Amazon ECS 的網路安全最佳實務
- Amazon ECS 任務和容器安全最佳實務

Amazon Elastic Container Service 的身分和存取管理

AWS Identity and Access Management (IAM) 是一種 AWS 服務 ，可協助管理員安全地控制對 AWS 資源的存取。IAM 管理員可以控制應對誰進行身分驗證 (已登入) 和授權 (具有許可) 以使用 Amazon ECS 資源。IAM 是 AWS 服務 您可以免費使用的 。

主題

目標對象

使用方式 AWS Identity and Access Management (IAM) 會有所不同，取決於您在 Amazon ECS 中執行的工作。

服務使用者 – 如果您使用 Amazon ECS 服務執行任務，您的管理員會為您提供您需要的憑證和許可。隨著您為了執行作業而使用的 Amazon ECS 功能數量變多，您可能會需要額外的許可。了解存取許可的管理方式可協助您向管理員請求正確的許可。如果您無法存取 Amazon ECS 中的功能，請參閱對 Amazon Elastic Container Service 身分和存取進行故障診斷。

服務管理員：若您在公司負責管理 Amazon ECS 資源，您應該擁有 Amazon ECS 的完整存取權。您的任務是判斷服務使用者應存取的 Amazon ECS 功能和資源。接著，您必須將請求提交給您的 IAM 管理員，來變更您服務使用者的許可。檢閱此頁面上的資訊，了解 IAM 的基本概念。若要進一步了解貴公司可搭配 Amazon ECS 使用 IAM 的方式，請參閱Amazon Elastic Container Service 如何與 IAM 搭配使用。

IAM 管理員：如果您是 IAM 管理員，建議您掌握如何撰寫政策以管理 Amazon ECS 存取權的詳細資訊。若要檢視您可以在 IAM 中使用的基於 Amazon ECS 身分的政策範例，請參閱Amazon Elastic Container Service 身分型政策範例。

使用身分驗證

身分驗證是您 AWS 使用身分憑證登入 的方式。您必須以 AWS 帳戶根使用者身分、IAM 使用者身分或擔任 IAM 角色身分進行身分驗證 （ 登入 AWS)。

您可以使用透過身分來源提供的憑證，以聯合身分 AWS 身分身分身分登入 。 AWS IAM Identity Center (IAM Identity Center) 使用者、您公司的單一登入身分驗證，以及您的 Google 或 Facebook 登入資料，都是聯合身分的範例。您以聯合身分登入時，您的管理員先前已設定使用 IAM 角色的聯合身分。當您使用聯合 AWS 身分存取 時，您會間接擔任角色。

視您身分的使用者類型而定，您可以登入 AWS Management Console 或 AWS 存取入口網站。如需登入的詳細資訊 AWS，請參閱AWS 登入 《 使用者指南》中的如何登入您的 AWS 帳戶 。

如果您以 AWS 程式設計方式存取 ， AWS 會提供軟體開發套件 (SDK) 和命令列界面 (CLI)，以使用您的登入資料以密碼編譯方式簽署您的請求。如果您不使用 AWS 工具，則必須自行簽署請求。如需使用建議的方法自行簽署請求的詳細資訊，請參閱《IAM 使用者指南》中的適用於 API 請求的AWS Signature 第 4 版。

無論您使用何種身分驗證方法，您可能都需要提供額外的安全性資訊。例如， AWS 建議您使用多重要素驗證 (MFA) 來提高帳戶的安全性。如需更多資訊，請參閱《AWS IAM Identity Center 使用者指南》中的多重要素驗證和《IAM 使用者指南》中的 IAM 中的AWS 多重要素驗證。

AWS 帳戶 根使用者

當您建立 時 AWS 帳戶，您會從一個登入身分開始，該身分可完整存取 帳戶中的所有 AWS 服務 和資源。此身分稱為 AWS 帳戶 Theroot 使用者，可透過使用您用來建立帳戶的電子郵件地址和密碼登入來存取。強烈建議您不要以根使用者處理日常任務。保護您的根使用者憑證，並將其用來執行只能由根使用者執行的任務。如需這些任務的完整清單，了解需以根使用者登入的任務，請參閱 IAM 使用者指南中的需要根使用者憑證的任務。

聯合身分

最佳實務是， 要求人類使用者，包括需要管理員存取權的使用者，使用臨時登入資料 AWS 服務 來使用與身分提供者的聯合來存取 。

聯合身分是來自您的企業使用者目錄、Web 身分提供者、 AWS Directory Service、身分中心目錄，或是使用透過身分來源提供的憑證 AWS 服務 存取的任何使用者。當聯合身分存取時 AWS 帳戶，它們會擔任角色，而角色會提供臨時憑證。

對於集中式存取權管理，我們建議您使用 AWS IAM Identity Center。您可以在 IAM Identity Center 中建立使用者和群組，或者您可以連接並同步到您自己的身分來源中的一組使用者和群組，以用於所有

AWS 帳戶 和 應用程式。如需 IAM Identity Center 的詳細資訊，請參閱 AWS IAM Identity Center 使用者指南中的什麼是 IAM Identity Center？。

IAM 使用者和群組

IAM 使用者是 中具有單一人員或應用程式特定許可 AWS 帳戶 的身分。建議您盡可能依賴臨時憑證，而不是擁有建立長期憑證 (例如密碼和存取金鑰) 的 IAM 使用者。但是如果特定使用案例需要擁有長期憑證的 IAM 使用者，建議您輪換存取金鑰。如需更多資訊，請參閱 IAM 使用者指南中的為需要長期憑證的使用案例定期輪換存取金鑰。

IAM 群組是一種指定 IAM 使用者集合的身分。您無法以群組身分簽署。您可以使用群組來一次為多名使用者指定許可。群組可讓管理大量使用者許可的程序變得更為容易。例如，您可以擁有一個名為 IAMAdmins 的群組，並給予該群組管理 IAM 資源的許可。

使用者與角色不同。使用者只會與單一人員或應用程式建立關聯，但角色的目的是在由任何需要它的人員取得。使用者擁有永久的長期憑證，但角色僅提供臨時憑證。如需更多資訊，請參閱《IAM 使用者指南》中的 IAM 使用者的使用案例。

IAM 角色

IAM 角色是 中具有特定許可 AWS 帳戶 的身分。它類似 IAM 使用者，但不與特定的人員相關聯。若要暫時在 中擔任 IAM 角色 AWS Management Console，您可以從使用者切換至 IAM 角色（主控台）。您可以透過呼叫 AWS CLI 或 AWS API 操作或使用自訂 URL 來擔任角色。如需使用角色的方法詳細資訊，請參閱《IAM 使用者指南》中的擔任角色的方法。

使用臨時憑證的 IAM 角色在下列情況中非常有用：

- 聯合身分使用者存取 — 如需向聯合身分指派許可，請建立角色，並為角色定義許可。當聯合身分進行身分驗證時，該身分會與角色建立關聯，並獲授予由角色定義的許可。如需有關聯合角色的相關資訊，請參閱《IAM 使用者指南》中的為第三方身分提供者 (聯合) 建立角色。如果您使用 IAM Identity Center，則需要設定許可集。為控制身分驗證後可以存取的內容，IAM Identity Center 將許可集與 IAM 中的角色相關聯。如需有關許可集的資訊，請參閱 AWS IAM Identity Center 使用者指南中的許可集。
- 暫時 IAM 使用者許可 – IAM 使用者或角色可以擔任 IAM 角色來暫時針對特定任務採用不同的許可。
- 跨帳戶存取權：您可以使用 IAM 角色，允許不同帳戶中的某人 (信任的主體) 存取您帳戶的資源。角色是授予跨帳戶存取權的主要方式。不過，對於某些 AWS 服務，您可以直接將政策連接到資源（而不是使用角色做為代理）。如需了解使用角色和資源型政策進行跨帳戶存取之間的差異，請參閱《IAM 使用者指南》中的 IAM 中的跨帳戶資源存取。

- 跨服務存取 – 有些 AWS 服務 使用其他 中的功能 AWS 服務。例如，當您在服務中進行呼叫時，該服務通常會在 Amazon EC2 中執行應用程式或將物件儲存在 Amazon Simple Storage Service (Amazon S3) 中。服務可能會使用呼叫主體的許可、使用服務角色或使用服務連結角色來執行此作業。
  - 轉送存取工作階段 (FAS) – 當您使用 IAM 使用者或角色在 中執行動作時 AWS，您會被視為主體。使用某些服務時，您可能會執行某個動作，進而在不同服務中啟動另一個動作。FAS 使用呼叫 的委託人許可 AWS 服務，並結合 AWS 服務 請求向下游服務提出請求。只有當服務收到需要與其他 AWS 服務 或 資源互動才能完成的請求時，才會提出 FAS 請求。在此情況下，您必須具有執行這兩個動作的許可。如需提出 FAS 請求時的政策詳細資訊，請參閱《轉發存取工作階段》。
  - 服務角色 – 服務角色是服務擔任的 IAM 角色，可代表您執行動作。IAM 管理員可以從 IAM 內建立、修改和刪除服務角色。如需詳細資訊，請參閱「IAM 使用者指南」中的建立角色以委派許可權給 AWS 服務。
  - 服務連結角色 – 服務連結角色是一種連結至 的服務角色。 AWS 服務服務可以擔任代表您執行動作的角色。服務連結角色會出現在您的 中 AWS 帳戶 ，並由服務擁有。IAM 管理員可以檢視，但不能編輯服務連結角色的許可。
- 在 Amazon EC2 上執行的應用程式 – 您可以使用 IAM 角色來管理在 EC2 執行個體上執行之應用程式的臨時登入資料，以及提出 AWS CLI 或 AWS API 請求。這是在 EC2 執行個體內儲存存取金鑰的較好方式。若要將 AWS 角色指派給 EC2 執行個體，並將其提供給其所有應用程式，您可以建立連接至執行個體的執行個體描述檔。執行個體設定檔包含該角色，並且可讓 EC2 執行個體上執行的程式取得臨時憑證。如需詳細資訊，請參閱《IAM 使用者指南》中的使用 IAM 角色來授予許可權給 Amazon EC2 執行個體上執行的應用程式。

使用政策管理存取權

您可以透過建立政策並將其連接至身分或資源 AWS 來控制 AWS 中的存取。政策是 中的物件，當與身分或資源相關聯時， AWS 會定義其許可。當委託人（使用者、根使用者或角色工作階段）發出請求時， 會 AWS 評估這些政策。政策中的許可決定是否允許或拒絕請求。大多數政策會以 JSON 文件 AWS 的形式存放在 中。如需 JSON 政策文件結構和內容的詳細資訊，請參閱 IAM 使用者指南中的 JSON 政策概觀。

管理員可以使用 AWS JSON 政策來指定誰可以存取內容。也就是說，哪個主體在什麼條件下可以對什麼資源執行哪些動作。

預設情況下，使用者和角色沒有許可。若要授予使用者對其所需資源執行動作的許可，IAM 管理員可以建立 IAM 政策。然後，管理員可以將 IAM 政策新增至角色，使用者便能擔任這些角色。

IAM 政策定義該動作的許可，無論您使用何種方法來執行操作。例如，假設您有一個允許 `iam:GetRole` 動作的政策。具有該政策的使用者可以從 AWS Management Console AWS CLI、 或 API AWS 取得角色資訊。

身分型政策

身分型政策是可以附加到身分 (例如 IAM 使用者、使用者群組或角色) 的 JSON 許可政策文件。這些政策可控制身分在何種條件下能對哪些資源執行哪些動作。如需了解如何建立身分型政策，請參閱《IAM 使用者指南》中的透過客戶管理政策定義自訂 IAM 許可。

身分型政策可進一步分類成內嵌政策或受管政策。內嵌政策會直接內嵌到單一使用者、群組或角色。受管政策是獨立的政策，您可以連接到 中的多個使用者、群組和角色 AWS 帳戶。受管政策包括 AWS 受管政策和客戶受管政策。如需了解如何在受管政策及內嵌政策之間選擇，請參閱《IAM 使用者指南》中的在受管政策和內嵌政策間選擇。

資源型政策

資源型政策是連接到資源的 JSON 政策文件。資源型政策的最常見範例是 IAM 角色信任政策和 Amazon S3 儲存貯體政策。在支援資源型政策的服務中，服務管理員可以使用它們來控制對特定資源的存取權限。對於附加政策的資源，政策會定義指定的主體可以對該資源執行的動作以及在何種條件下執行的動作。您必須在資源型政策中指定主體。委託人可以包括帳戶、使用者、角色、聯合身分使用者或 AWS 服務。

資源型政策是位於該服務中的內嵌政策。您無法在資源型政策中使用來自 IAM 的 AWS 受管政策。

存取控制清單 (ACL)

存取控制清單 (ACL) 可控制哪些主體 (帳戶成員、使用者或角色) 擁有存取某資源的許可。ACL 類似於資源型政策，但它們不使用 JSON 政策文件格式。

Amazon S3 AWS WAF和 Amazon VPC 是支援 ACLs的服務範例。如需進一步了解 ACL，請參閱 Amazon Simple Storage Service 開發人員指南中的存取控制清單 (ACL) 概觀。

其他政策類型

AWS 支援其他較不常見的政策類型。這些政策類型可設定較常見政策類型授予您的最大許可。

- 許可界限 – 許可範圍是一種進階功能，可供您設定身分型政策能授予 IAM 實體 (IAM 使用者或角色) 的最大許可。您可以為實體設定許可界限。所產生的許可會是實體的身分型政策和其許可界限的交集。會在 `Principal` 欄位中指定使用者或角色的資源型政策則不會受到許可界限限制。所有這類政

策中的明確拒絕都會覆寫該允許。如需許可界限的詳細資訊，請參閱 IAM 使用者指南中的 IAM 實體許可界限。

- 服務控制政策 SCPs) – SCPs是 JSON 政策，可指定 in. 中組織或組織單位 (OU) 的最大許可 AWS Organizations。 AWS Organizations 是一種用於分組和集中管理您企業擁有 AWS 帳戶 的多個的服務。若您啟用組織中的所有功能，您可以將服務控制政策 (SCP) 套用到任何或所有帳戶。SCP 會限制成員帳戶中實體的許可，包括每個實體 AWS 帳戶根使用者。如需 Organizations 和 SCP 的詳細資訊，請參閱《AWS Organizations 使用者指南》中的服務控制政策。
- 資源控制政策 (RCP) - RCP 是 JSON 政策，可用來設定您帳戶中資源的可用許可上限，採取這種方式就不需要更新附加至您所擁有的每個資源的 IAM 政策。RCP 會限制成員帳戶中資源的許可，並可能影響身分的有效許可，包括 AWS 帳戶根使用者，無論它們是否屬於您的組織。如需 Organizations 和 RCPs的詳細資訊，包括支援 RCPs AWS 服務 的 清單，請參閱AWS Organizations 《 使用者指南》中的資源控制政策 RCPs)。
- 工作階段政策 – 工作階段政策是一種進階政策，您可以在透過撰寫程式的方式建立角色或聯合使用者的暫時工作階段時，做為參數傳遞。所產生工作階段的許可會是使用者或角色的身分型政策和工作階段政策的交集。許可也可以來自資源型政策。所有這類政策中的明確拒絕都會覆寫該允許。如需詳細資訊，請參閱 IAM 使用者指南中的工作階段政策。

多種政策類型

將多種政策類型套用到請求時，其結果形成的許可會更為複雜、更加難以理解。若要了解如何 AWS 在涉及多種政策類型時決定是否允許請求，請參閱《IAM 使用者指南》中的政策評估邏輯。

Amazon Elastic Container Service 如何與 IAM 搭配使用

在您使用 IAM 管理對 Amazon ECS 的存取權之前，請瞭解哪些 IAM 功能可以與 Amazon ECS 搭配使用。

IAM 功能	Amazon ECS 支援
身分型政策	是
資源型政策	否
政策動作	是
政策資源	部分

IAM 功能	Amazon ECS 支援
政策條件索引鍵	是
ACL	否
ABAC (政策中的標籤)	是
暫時性憑證	是
轉送存取工作階段 (FAS)	是
服務角色	是
服務連結角色	是

若要深入了解 Amazon ECS 和其他 AWS 服務如何與大多數 IAM 功能搭配使用，請參閱《IAM 使用者指南》中的AWS 與 IAM 搭配使用的 服務。

Amazon ECS 身分型政策

支援身分型政策：是

身分型政策是可以附加到身分 (例如 IAM 使用者、使用者群組或角色) 的 JSON 許可政策文件。這些政策可控制身分在何種條件下能對哪些資源執行哪些動作。如需了解如何建立身分型政策，請參閱《IAM 使用者指南》中的透過客戶管理政策定義自訂 IAM 許可。

使用 IAM 身分型政策，您可以指定允許或拒絕的動作和資源，以及在何種條件下允許或拒絕動作。您無法在身分型政策中指定主體，因為這會套用至連接的使用者或角色。如要了解您在 JSON 政策中使用的所有元素，請參閱《IAM 使用者指南》中的 IAM JSON 政策元素參考。

Amazon ECS 的身分型政策範例

若要檢視 Amazon ECS 身分型政策的範例，請參閱Amazon Elastic Container Service 身分型政策範例。

Amazon ECS 中的資源型政策

支援資源型政策：否

資源型政策是附加到資源的 JSON 政策文件。資源型政策的最常見範例是 IAM 角色信任政策和 Amazon S3 儲存貯體政策。在支援資源型政策的服務中，服務管理員可以使用它們來控制對特定資源的存取權限。對於附加政策的資源，政策會定義指定的主體可以對該資源執行的動作以及在何種條件下執行的動作。您必須在資源型政策中指定主體。委託人可以包括帳戶、使用者、角色、聯合身分使用者或 AWS 服務。

如需啟用跨帳戶存取權，您可以指定在其他帳戶內的所有帳戶或 IAM 實體，做為資源型政策的主體。新增跨帳戶主體至資源型政策，只是建立信任關係的一半。當主體和資源位於不同的位置時 AWS 帳戶，信任帳戶中的 IAM 管理員也必須授予主體實體（使用者或角色）存取資源的許可。其透過將身分型政策連接到實體來授與許可。不過，如果資源型政策會為相同帳戶中的主體授予存取，這時就不需要額外的身分型政策。如需詳細資訊，請參閱《IAM 使用者指南》中的 IAM 中的快帳戶資源存取。

Amazon ECS 的政策動作

支援政策動作：是

管理員可以使用 AWS JSON 政策來指定誰可以存取內容。也就是說，哪個主體在什麼条件下可以對什麼資源執行哪些動作。

JSON 政策的 `Action` 元素描述您可以用來允許或拒絕政策中存取的動作。政策動作通常具有與相關聯 AWS API 操作相同的名稱。有一些例外狀況，例如沒有相符的 API 操作的僅限許可動作。也有一些作業需要政策中的多個動作。這些額外的動作稱為相依動作。

政策會使用動作來授予執行相關聯動作的許可。

若要查看 Amazon ECS 動作的清單，請參閱《服務授權參考》中的 Amazon Elastic Container Service 定義的動作。

Amazon ECS 中的政策動作會在動作之前使用下列字首：

```
ecs
```

如需在單一陳述式中指定多個動作，請用逗號分隔。

```
"Action": [
      "ecs:action1",
      "ecs:action2"
         ]
```

您也可以使用萬用字元 (*) 來指定多個動作。例如，若要指定開頭是 Describe 文字的所有動作，請包含以下動作：

```
"Action": "ecs:Describe*"
```

若要檢視 Amazon ECS 身分型政策的範例，請參閱Amazon Elastic Container Service 身分型政策範例。

Amazon ECS 的政策資源

支援政策資源：部分

管理員可以使用 AWS JSON 政策來指定誰可以存取內容。也就是說，哪個主體在什麼條件下可以對什麼資源執行哪些動作。

Resource JSON 政策元素可指定要套用動作的物件。陳述式必須包含 Resource 或 NotResource 元素。最佳實務是使用其 Amazon Resource Name (ARN) 來指定資源。您可以針對支援特定資源類型的動作 (稱為資源層級許可) 來這麼做。

對於不支援資源層級許可的動作 (例如列出操作)，請使用萬用字元 (*) 來表示陳述式適用於所有資源。

```
"Resource": "*"
```

若要查看 Amazon ECS 資源類型及其 ARN 的清單，請參閱《服務授權參考》中的 Amazon Elastic Container Service 定義的資源。若要瞭解您可以使用哪些動作指定每個資源的 ARN，請參閱 Amazon Elastic Container Service 定義的動作。

部分 Amazon ECS API 動作支援多個資源。例如，呼叫 DescribeClusters API 動作時可以參考多個叢集。若要在單一陳述式中指定多項資源，請使用逗號分隔 ARN。

```
"Resource": [
      "EXAMPLE-RESOURCE-1",
      "EXAMPLE-RESOURCE-2"
```

例如，Amazon ECS 叢集資源具有以下 ARN：

```
arn:${Partition}:ecs:${Region}:${Account}:cluster/${clusterName}
```

若要在您的陳述式中指定 my-cluster-1 和 my-cluster-2 叢集，請使用以下 ARN：

```
"Resource": [
        "arn:aws:ecs:us-east-1:123456789012:cluster/my-cluster-1",
        "arn:aws:ecs:us-east-1:123456789012:cluster/my-cluster-2"
```

若要指定所有屬於特定帳戶的叢集，請使用萬用字元 (*)：

```
"Resource": "arn:aws:ecs:us-east-1:123456789012:cluster/*"
```

對於任務定義，您可以指定最新版本或特定修訂版。

若要指定任務定義的所有修訂，請使用萬用字元 (*)：

```
"Resource:arn:${Partition}:ecs:${Region}:${Account}:task-definition/
${TaskDefinitionFamilyName}:*"
```

若要指定特定任務定義修訂版，請使用 ${TaskDefinitionRevisionNumber}：

```
"Resource:arn:${Partition}:ecs:${Region}:${Account}:task-definition/
${TaskDefinitionFamilyName}:${TaskDefinitionRevisionNumber}"
```

若要檢視 Amazon ECS 身分型政策的範例，請參閱Amazon Elastic Container Service 身分型政策範例。

Amazon ECS 的政策條件索引鍵

支援服務特定政策條件金鑰：是

管理員可以使用 AWS JSON 政策來指定誰可以存取內容。也就是說，哪個主體在什麼條件下可以對什麼資源執行哪些動作。

Condition 元素 (或 Condition 區塊) 可讓您指定使陳述式生效的條件。Condition 元素是選用項目。您可以建立使用條件運算子的條件運算式 (例如等於或小於)，來比對政策中的條件和請求中的值。

若您在陳述式中指定多個 Condition 元素，或是在單一 Condition 元素中指定多個索引鍵， AWS 會使用邏輯 AND 操作評估他們。如果您為單一條件索引鍵指定多個值， 會使用邏輯OR操作 AWS 來評估條件。必須符合所有條件，才會授與陳述式的許可。

您也可以在指定條件時使用預留位置變數。例如，您可以只在使用者使用其 IAM 使用者名稱標記時，將存取資源的許可授予該 IAM 使用者。如需更多資訊，請參閱 IAM 使用者指南中的 IAM 政策元素：變數和標籤。

AWS 支援全域條件金鑰和服務特定的條件金鑰。若要查看所有 AWS 全域條件索引鍵，請參閱《IAM 使用者指南》中的AWS 全域條件內容索引鍵。

Amazon ECS 支援以下服務特定條件索引鍵，您可以使用這些條件索引鍵來為您的 IAM 政策提供精細篩選條件：

條件索引鍵	描述	評估類型
aws:RequestTag/${TagKey}	內容金鑰的格式為 "aws:RequestTag/ *tag-key*":"*tag-value* "，其中 *tag-key* 和 *tag-value* 是標籤金鑰和值對。 檢查標籤鍵/值對是否存在於 AWS 請求中。例如，您可以檢查請求是否包含標籤金鑰 "Dept" 並且它具有值 "Accounting" 。	字串
aws:ResourceTag/${TagKey}	內容金鑰的格式為 "aws:ResourceTag/ *tag-key*":"*tag-value* "，其中 *tag-key* 和 *tag-value* 是標籤金鑰和值對。 檢查連接至身分資源 (使用者或角色) 的標籤是否符合指定的鍵名稱和值。	字串
aws:TagKeys	此內容鍵的格式為 "aws:TagKeys":" *tag-key*"，其中 *tag-key* 是沒有值的標籤鍵清單 (例如，["Dept","Cost-Center"])。 檢查 AWS 請求中存在的標籤索引鍵。	字串
ecs:ResourceTag/${TagKey}	內容金鑰的格式為 "ecs:ResourceTag/ *tag-key*":"*tag-value* "，其中 *tag-key* 和 *tag-value* 是標籤金鑰和值對。 檢查連接至身分資源 (使用者或角色) 的標籤是否符合指定的鍵名稱和值。	字串

條件索引鍵	描述	評估類型
ecs:cluster	內容金鑰的格式為 "ecs:cluster":" *cluster-arn* "，其中 *cluster-arn* 是 Amazon ECS 叢集的 ARN。	ARN, Null
ecs:container-instances	內容金鑰的格式為 "ecs:container-instances":" *container-instance-arns* "，其中 *container-instance-arns* 為一或多個容器執行個體的 ARN。	ARN, Null
ecs:container-name	內容索引鍵的格式為 "ecs:container-name":" *container-name* "，其中 *container-instance-* 是在任務定義中定義的 Amazon ECS 容器名稱。	字串
ecs:enable-execute-command	內文索引鍵已被格式化為 "ecs:enable-execute-command":" *value*"，其中 *value-* 是 "true" 或 "false"。	字串
ecs:enable-service-connect	內容索引鍵的格式為 "ecs:enable-service-connect":" *value*"，其中 *value* 是 "true" 或 "false"。	字串
ecs：enable-ebs-volumes	內容索引鍵的格式為 "ecs:enable-ebs-volumes":" *value*"，其中 *value* 是 "true" 或 "false"。	字串
ecs:namespace	內容索引鍵的格式為 "ecs:namespace":" *namespace-arn* "，其中 *namespace-arn* 是 AWS Cloud Map 命名空間的 ARN。	ARN, Null
ecs:service	內容金鑰的格式為 "ecs:service":" *service-arn* "，其中 *service-arn* 是 Amazon ECS 服務的 ARN。	ARN, Null
ecs:task-definition	內容金鑰的格式為 "ecs:task-definition":" *task-definition-arn* "，其中 *task-definition-arn* 為 Amazon ECS 任務定義的 ARN。	ARN, Null

條件索引鍵	描述	評估類型
ecs:account-setting	內容索引鍵的格式為 "ecs:account-setting":" *account-setting* "，其中 *account-setting* 為 Amazon ECS 帳戶設定的名稱。	字串

若要查看 Amazon ECS 條件索引鍵的清單，請參閱《服務授權參考》中的 Amazon Elastic Kubernetes Service 的條件索引鍵。若要了解您可以搭配哪些動作和資源使用條件索引鍵，請參閱 Amazon Elastic Container Service 定義的動作。

若要檢視 Amazon ECS 身分型政策的範例，請參閱Amazon Elastic Container Service 身分型政策範例。

Amazon ECS 中的存取控制清單 (ACL)

支援 ACL：否

存取控制清單 (ACL) 可控制哪些主體 (帳戶成員、使用者或角色) 擁有存取某資源的許可。ACL 類似於資源型政策，但它們不使用 JSON 政策文件格式。

搭配 Amazon ECS 的屬性型存取控制 (ABAC)

> **Important**
>
> Amazon ECS 支援所有 Amazon ECS 資源的屬性型存取控制。若要判斷是否可以使用屬性來限定動作範圍，請使用《服務授權參考》中由 Amazon ECS 表格定義的動作。首先驗證 Resource (資源) 欄位中是否有資源。然後，使用 Condition keys (條件索引鍵) 欄位查看動作/資源組合的索引鍵。

支援 ABAC (政策中的標籤)：是

屬性型存取控制 (ABAC) 是一種授權策略，可根據屬性來定義許可。在 中 AWS，這些屬性稱為標籤。您可以將標籤連接至 IAM 實體 （使用者或角色） 和許多 AWS 資源。為實體和資源加上標籤是 ABAC 的第一步。您接著要設計 ABAC 政策，允許在主體的標籤與其嘗試存取的資源標籤相符時操作。

ABAC 在成長快速的環境中相當有幫助，並能在政策管理變得繁瑣時提供協助。

如需根據標籤控制存取，請使用 aws:ResourceTag/*key-name*、aws:RequestTag/*key-name* 或 aws:TagKeys 條件索引鍵，在政策的條件元素中，提供標籤資訊。

如果服務支援每個資源類型的全部三個條件金鑰，則對該服務而言，值為 Yes。如果服務僅支援某些資源類型的全部三個條件金鑰，則值為 Partial。

如需 ABAC 的詳細資訊，請參閱《IAM 使用者指南》中的使用 ABAC 授權定義許可。如要查看含有設定 ABAC 步驟的教學課程，請參閱 IAM 使用者指南中的使用屬性型存取控制 (ABAC)。

如需標記 Amazon ECS 資源的詳細資訊，請參閱 標記 Amazon ECS 資源。

若要檢視身分型政策範例，以根據該資源上的標籤來限制存取資源，請參閱根據標籤描述 Amazon ECS 服務。

將暫時憑證與 Amazon ECS 搭配使用

支援臨時憑證：是

當您使用臨時憑證登入時，有些 AWS 服務 無法使用。如需詳細資訊，包括哪些 AWS 服務 使用臨時登入資料，請參閱《AWS 服務 IAM 使用者指南》中的使用 IAM 的 。

如果您 AWS Management Console 使用使用者名稱和密碼以外的任何方法登入 ，則會使用臨時登入資料。例如，當您 AWS 使用公司的單一登入 (SSO) 連結存取 時，該程序會自動建立臨時登入資料。當您以使用者身分登入主控台，然後切換角色時，也會自動建立臨時憑證。如需切換角色的詳細資訊，請參閱《IAM 使用者指南》中的從使用者切換至 IAM 角色 (主控台)。

您可以使用 AWS CLI 或 AWS API 手動建立臨時登入資料。然後，您可以使用這些臨時登入資料來存取 AWS。 AWS 建議您動態產生臨時登入資料，而不是使用長期存取金鑰。如需詳細資訊，請參閱 IAM 中的暫時性安全憑證。

轉送 Amazon ECS 的存取工作階段

支援轉寄存取工作階段 (FAS)：是

當您使用 IAM 使用者或角色在 中執行動作時 AWS，您會被視為委託人。使用某些服務時，您可能會執行某個動作，進而在不同服務中啟動另一個動作。FAS 使用呼叫 的委託人許可 AWS 服務，並結合 AWS 服務 請求向下游服務提出請求。只有當服務收到需要與其他 AWS 服務 或 資源互動才能完成的請求時，才會提出 FAS 請求。在此情況下，您必須具有執行這兩個動作的許可。如需提出 FAS 請求時的政策詳細資訊，請參閱《轉發存取工作階段》。

Amazon ECS 的服務角色

支援服務角色：是

服務角色是服務擔任的 IAM 角色，可代您執行動作。IAM 管理員可以從 IAM 內建立、修改和刪除服務角色。如需詳細資訊，請參閱「IAM 使用者指南」中的建立角色以委派許可權給 AWS 服務。

> **Warning**
>
> 變更服務角色的許可可能會中斷 Amazon ECS 功能。只有在 Amazon ECS 提供指引時，才能編輯服務角色。

Amazon ECS 的服務連結角色

支援服務連結角色：是

服務連結角色是連結至 的服務角色類型 AWS 服務。服務可以擔任代表您執行動作的角色。服務連結角色會出現在您的 中 AWS 帳戶 ，並由服務擁有。IAM 管理員可以檢視，但不能編輯服務連結角色的許可。

如需建立或管理 Amazon ECS 服務連結角色的詳細資訊，請參閱使用 Amazon ECS 的服務連結角色。

Amazon Elastic Container Service 身分型政策範例

根據預設，使用者和角色不具備建立或修改 Amazon ECS 資源的許可。他們也無法使用 AWS Management Console、 AWS Command Line Interface (AWS CLI) 或 AWS API 來執行任務。若要授予使用者對其所需資源執行動作的許可，IAM 管理員可以建立 IAM 政策。然後，管理員可以將 IAM 政策新增至角色，使用者便能擔任這些角色。

如需了解如何使用這些範例 JSON 政策文件建立 IAM 身分型政策，請參閱《IAM 使用者指南》中的建立 IAM 政策 (主控台)。

如需 Amazon ECS 所定義之動作和資源類型的詳細資訊，包括每種資源類型的 ARN 格式，請參閱服務授權參考中的《適用於 Amazon Elastic Container Service 的動作、資源和條件索引鍵》。

主題

- Amazon ECS 政策最佳實務
- 允許 Amazon ECS 使用者檢視自己的許可
- Amazon ECS 叢集範例
- Amazon ECS 容器執行個體範例
- Amazon ECS 任務定義範例

- 執行 Amazon ECS 任務範例
- 啟動 Amazon ECS 任務範例
- 列出並描述 Amazon ECS 任務範例
- 建立 Amazon ECS 服務範例
- 更新 Amazon ECS 服務範例
- 根據標籤描述 Amazon ECS 服務
- 拒絕 Amazon ECS Service Connect 命名空間覆寫範例

Amazon ECS 政策最佳實務

身分型政策會判斷您帳戶中的某個人員是否可以建立、存取或刪除 Amazon ECS 資源。這些動作可能會讓您的 AWS 帳戶產生費用。當您建立或編輯身分型政策時，請遵循下列準則及建議事項：

- 開始使用 AWS 受管政策並邁向最低權限許可 – 若要開始將許可授予您的使用者和工作負載，請使用 AWS 受管政策，將許可授予許多常見使用案例。它們可在您的 中使用 AWS 帳戶。我們建議您定義特定於使用案例 AWS 的客戶受管政策，進一步減少許可。如需更多資訊，請參閱 IAM 使用者指南中的 AWS 受管政策或任務職能的AWS 受管政策。
- 套用最低權限許可 – 設定 IAM 政策的許可時，請僅授予執行任務所需的許可。為實現此目的，您可以定義在特定條件下可以對特定資源採取的動作，這也稱為最低權限許可。如需使用 IAM 套用許可的更多相關資訊，請參閱 IAM 使用者指南中的 IAM 中的政策和許可。
- 使用 IAM 政策中的條件進一步限制存取權 – 您可以將條件新增至政策，以限制動作和資源的存取。例如，您可以撰寫政策條件，指定必須使用 SSL 傳送所有請求。如果透過特定 使用服務動作，您也可以使用條件來授予存取服務動作的權限 AWS 服務，例如 AWS CloudFormation。如需詳細資訊，請參閱 IAM 使用者指南中的 IAM JSON 政策元素：條件。
- 使用 IAM Access Analyzer 驗證 IAM 政策，確保許可安全且可正常運作 – IAM Access Analyzer 驗證新政策和現有政策，確保這些政策遵從 IAM 政策語言 (JSON) 和 IAM 最佳實務。IAM Access Analyzer 提供 100 多項政策檢查及切實可行的建議，可協助您撰寫安全且實用的政策。如需詳細資訊，請參閱《IAM 使用者指南》中的使用 IAM Access Analyzer 驗證政策。
- 需要多重要素驗證 (MFA)：如果您的案例需要 IAM 使用者或 中的根使用者 AWS 帳戶，請開啟 MFA 以增加安全性。如需在呼叫 API 操作時請求 MFA，請將 MFA 條件新增至您的政策。如需詳細資訊，請參閱《IAM 使用者指南》https://docs.aws.amazon.com/IAM/latest/UserGuide/id_credentials_mfa_configure-api-require.html中的透過 MFA 的安全 API 存取。

如需 IAM 中最佳實務的相關資訊，請參閱 IAM 使用者指南中的 IAM 安全最佳實務。

允許 Amazon ECS 使用者檢視自己的許可

此範例會示範如何建立政策，允許 IAM 使用者檢視連接到他們使用者身分的內嵌及受管政策。此政策包含在主控台上完成此動作的許可，或使用 AWS CLI 或 AWS API 以程式設計方式完成此動作的許可。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ViewOwnUserInfo",
            "Effect": "Allow",
            "Action": [
                "iam:GetUserPolicy",
                "iam:ListGroupsForUser",
                "iam:ListAttachedUserPolicies",
                "iam:ListUserPolicies",
                "iam:GetUser"
            ],
            "Resource": ["arn:aws:iam::*:user/${aws:username}"]
        },
        {
            "Sid": "NavigateInConsole",
            "Effect": "Allow",
            "Action": [
                "iam:GetGroupPolicy",
                "iam:GetPolicyVersion",
                "iam:GetPolicy",
                "iam:ListAttachedGroupPolicies",
                "iam:ListGroupPolicies",
                "iam:ListPolicyVersions",
                "iam:ListPolicies",
                "iam:ListUsers"
            ],
            "Resource": "*"
        }
    ]
}
```

Amazon ECS 叢集範例

以下 IAM 政策允許建立和列出叢集的許可。CreateCluster 和 ListClusters 動作不接受任何資源，因此資源定義已針對所有資源設為 *。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ecs:CreateCluster",
                "ecs:ListClusters"
            ],
            "Resource": ["*"]
        }
    ]
}
```

以下 IAM 政策允許說明和刪除指定叢集的許可。DescribeClusters 和 DeleteCluster 動作接受叢集 ARN 做為資源。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ecs:DescribeClusters",
                "ecs:DeleteCluster"
            ],
            "Resource": ["arn:aws:ecs:us-east-1:<aws_account_id>:cluster/
<cluster_name>"]
        }
    ]
}
```

以下 IAM 政策可連接到使用者或群組，只允許該使用者或群組在特定叢集上執行操作。

```
{
    "Version": "2012-10-17",
    "Statement": [
```

```
        {
            "Action": [
                "ecs:Describe*",
                "ecs:List*"
            ],
            "Effect": "Allow",
            "Resource": "*"
        },
        {
            "Action": [
                "ecs:DeleteCluster",
                "ecs:DeregisterContainerInstance",
                "ecs:ListContainerInstances",
                "ecs:RegisterContainerInstance",
                "ecs:SubmitContainerStateChange",
                "ecs:SubmitTaskStateChange"
            ],
            "Effect": "Allow",
            "Resource": "arn:aws:ecs:us-east-1:<aws_account_id>:cluster/default"
        },
        {
            "Action": [
                "ecs:DescribeContainerInstances",
                "ecs:DescribeTasks",
                "ecs:ListTasks",
                "ecs:UpdateContainerAgent",
                "ecs:StartTask",
                "ecs:StopTask",
                "ecs:RunTask"
            ],
            "Effect": "Allow",
            "Resource": "*",
            "Condition": {
                "ArnEquals": {"ecs:cluster": "arn:aws:ecs:us-
east-1:<aws_account_id>:cluster/default"}
            }
        }
    ]
}
```

Amazon ECS 容器執行個體範例

容器執行個體的註冊由 Amazon ECS 代理程式處理，但有時候您可能會想要讓使用者能從叢集手動取消註冊執行個體。也許容器執行個體意外註冊到錯誤的叢集，或執行個體上尚有任務執行中時遭到終止。

以下 IAM 政策可讓使用者列出和取消註冊指定叢集中的容器執行個體：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ecs:DeregisterContainerInstance",
                "ecs:ListContainerInstances"
            ],
            "Resource": ["arn:aws:ecs:<region>:<aws_account_id>:cluster/
<cluster_name>"]
        }
    ]
}
```

以下 IAM 政策可讓使用者說明指定叢集中的指定容器執行個體。若要開啟此許可給叢集中的所有容器執行個體，您可以使用 * 取代容器執行個體的 UUID。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": ["ecs:DescribeContainerInstances"],
            "Condition": {
                "ArnEquals": {"ecs:cluster":
 "arn:aws:ecs:<region>:<aws_account_id>:cluster/<cluster_name>"}
            },
            "Resource": ["arn:aws:ecs:<region>:<aws_account_id>:container-instance/
<cluster_name>/<container_instance_UUID>"]
        }
    ]
}
```

Amazon ECS 任務定義範例

任務定義 IAM 政策不支援資源層級許可，但以下 IAM 政策允許使用者註冊、列出和說明任務定義：

如果您使用主控台，則必須新增 `CloudFormation: CreateStack` 作為 `Action`。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ecs:RegisterTaskDefinition",
                "ecs:ListTaskDefinitions",
                "ecs:DescribeTaskDefinition"
            ],
            "Resource": ["*"]
        }
    ]
}
```

執行 Amazon ECS 任務範例

`RunTask` 的資源為任務定義。若要限制使用者可執行任務定義的叢集，您可以在 `Condition` 區塊中加以指定。優點是您不需要在您的資源中列出任務定義和叢集，也能允許適當的存取。您可以套用其中一項，或同時套用兩者。

以下 IAM 政策允許在特定叢集上執行任何特定任務定義之修訂的許可：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": ["ecs:RunTask"],
            "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>:*"]
        }
```

```
    ]
}
```

啟動 Amazon ECS 任務範例

StartTask 的資源為任務定義。若要限制使用者可啟動任務定義的叢集和容器執行個體，您可以在 Condition 區塊中加以指定。優點是您不需要在您的資源中列出任務定義和叢集，也能允許適當的存取。您可以套用其中一項，或同時套用兩者。

以下 IAM 政策允許在特定叢集與特定容器執行個體上，啟動任何特定任務定義修訂的許可。

> **Note**
>
> 在此範例中，當您使用 AWS CLI 或其他 AWS SDK 呼叫 StartTask API 時，您必須指定任務定義修訂，以便Resource映射相符。

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

列出並描述 Amazon ECS 任務範例

以下 IAM 政策允許使用者列出指定叢集的任務：

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

以下 IAM 政策可讓使用者說明指定叢集中的指定任務：

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

建立 Amazon ECS 服務範例

以下 IAM 政策可讓使用者在 AWS Management Console中建立 Amazon ECS 服務：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
```

```
            "Action": [
                "application-autoscaling:Describe*",
                "application-autoscaling:PutScalingPolicy",
                "application-autoscaling:RegisterScalableTarget",
                "cloudwatch:DescribeAlarms",
                "cloudwatch:PutMetricAlarm",
                "ecs:List*",
                "ecs:Describe*",
                "ecs:CreateService",
                "elasticloadbalancing:Describe*",
                "iam:GetPolicy",
                "iam:GetPolicyVersion",
                "iam:GetRole",
                "iam:ListAttachedRolePolicies",
                "iam:ListRoles",
                "iam:ListGroups",
                "iam:ListUsers"
            ],
            "Resource": ["*"]
        }
    ]
}
```

更新 Amazon ECS 服務範例

以下 IAM 政策可讓使用者在 AWS Management Console中更新 Amazon ECS 服務：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "application-autoscaling:Describe*",
                "application-autoscaling:PutScalingPolicy",
                "application-autoscaling:DeleteScalingPolicy",
                "application-autoscaling:RegisterScalableTarget",
                "cloudwatch:DescribeAlarms",
                "cloudwatch:PutMetricAlarm",
                "ecs:List*",
                "ecs:Describe*",
                "ecs:UpdateService",
                "iam:GetPolicy",
```

```
                "iam:GetPolicyVersion",
                "iam:GetRole",
                "iam:ListAttachedRolePolicies",
                "iam:ListRoles",
                "iam:ListGroups",
                "iam:ListUsers"
            ],
            "Resource": ["*"]
        }
    ]
}
```

根據標籤描述 Amazon ECS 服務

您可以在基於身分的政策中使用條件，根據標籤控制 Amazon ECS 資源的存取權。此範例會示範如何建立會允許描述您的服務的政策。但是，只有在服務標籤 Owner 的值是該使用者的使用者名稱時，才會授予該許可。此政策也會授予在主控台完成此動作的必要許可。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "DescribeServices",
            "Effect": "Allow",
            "Action": "ecs:DescribeServices",
            "Resource": "*"
        },
        {
            "Sid": "ViewServiceIfOwner",
            "Effect": "Allow",
            "Action": "ecs:DescribeServices",
            "Resource": "arn:aws:ecs:*:*:service/*",
            "Condition": {
                "StringEquals": {"ecs:ResourceTag/Owner": "${aws:username}"}
            }
        }
    ]
}
```

您可以將此政策連接到您帳戶中的 IAM 使用者。如果名為 `richard-roe` 的使用者嘗試描述 Amazon ECS 服務，則該服務必須標記為 `Owner=richard-roe` 或 `owner=richard-roe`。否則他便會被拒

絕存取。條件標籤鍵 Owner 符合 Owner 和 owner，因為條件索引鍵名稱不區分大小寫。如需詳細資訊，請參閱《IAM 使用者指南》中的 IAM JSON 政策元素：條件。

拒絕 Amazon ECS Service Connect 命名空間覆寫範例

下列 IAM 政策會拒絕使用者覆寫服務組態中的預設 Service Connect 命名空間。預設命名空間在叢集中設定。不過，您可以在服務組態中覆寫命名空間。為了保持一致性，請考慮將所有新服務設定為使用相同的命名空間。使用下列內容索引鍵可要求服務使用特定的命名空間。在以下範例中，將 <region>、<aws_account_id>、<cluster_name> 和 <namespace_id> 取代為自訂文字。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ecs:CreateService",
                "ecs:UpdateService"
            ],
            "Condition": {
                "ArnEquals": {
                    "ecs:cluster": "arn:aws:ecs:<region>:<aws_account_id>:cluster/
<cluster_name>",
                    "ecs:namespace":
 "arn:aws:servicediscovery:<region>:<aws_account_id>:namespace/<namespace_id>"
                }
            },
            "Resource": "*"
        }
    ]
}
```

AWS Amazon Elastic Container Service 的 受管政策

若要將許可新增至使用者、群組和角色，使用 AWS 受管政策比自行撰寫政策更容易。建立 IAM 客戶受管政策需要時間和專業知識，而受管政策可為您的團隊提供其所需的許可。若要快速開始使用，您可以使用我們的 AWS 受管政策。這些政策涵蓋常見的使用案例，並且可在您的帳戶中使用 AWS 。如需受 AWS 管政策的詳細資訊，請參閱《IAM 使用者指南》中的AWS 受管政策。

AWS 服務會維護和更新 AWS 受管政策。您無法變更 AWS 受管政策中的許可。服務偶爾會將其他許可新增至 AWS 受管政策，以支援新功能。此類型的更新會影響已連接政策的所有身分識別 (使用者、群組和角色)。服務最有可能在新功能啟動或新操作可用時更新 AWS 受管政策。服務不會從 AWS 受管政策中移除許可，因此政策更新不會破壞您現有的許可。

此外， AWS 支援跨多個 服務之任務函數的受管政策。例如，ReadOnlyAccess AWS 受管政策提供所有 AWS 服務和資源的唯讀存取權。當服務啟動新功能時， 會為新操作和資源 AWS 新增唯讀許可。如需任務職能政策的清單和說明，請參閱 IAM 使用者指南中有關任務職能的AWS 受管政策。

Amazon ECS 和 Amazon ECR 提供多個受管政策和信任關係，您可將其連接至使用者、群組、角色、Amazon EC2 執行個體和 Amazon ECS 任務，允許對資源和 API 操作進行不同層級的控制。您可以直接套用這些政策，也可以使用它們開始建立您自己的政策。如需 Amazon ECR 受管政策的詳細資訊，請參閱 Amazon ECR 受管政策。

AmazonECS_FullAccess

您可將 `AmazonECS_FullAccess` 政策連接到 IAM 身分。此政策會授予 Amazon ECS 資源的管理存取權，並授予 IAM 身分（例如使用者、群組或角色）存取權給與 Amazon ECS 整合 AWS 的服務，以使用所有 Amazon ECS 功能。使用此政策可存取 AWS Management Console中可用的所有 Amazon ECS 功能。

若要檢視此政策的許可，請參閱 AWS 受管政策參考中的 AmazonECS_FullAccess。

AmazonECSInfrastructureRolePolicyForVolumes

您可以將 `AmazonECSInfrastructureRolePolicyForVolumes` 受管政策連接至 IAM 實體。

此政策會授予 Amazon ECS 代表您進行 AWS API 呼叫所需的許可。您可以在啟動 Amazon ECS 任務和服務時，將此政策連接至您隨磁碟區組態提供的 IAM 角色。此角色可讓 Amazon ECS 管理連接至任務的磁碟區。如需詳細資訊，請參閱 Amazon ECS 基礎設施 IAM 角色。

若要檢視此政策的許可，請參閱 AWS 受管政策參考中的 AmazonECSInfrastructureRolePolicyForVolumes。

AmazonEC2ContainerServiceforEC2Role

您可將 `AmazonEC2ContainerServiceforEC2Role` 政策連接到 IAM 身分。此政策會授予管理許可，允許 Amazon ECS 容器執行個體 AWS 代表您呼叫 。如需詳細資訊，請參閱Amazon ECS 容器執行個體 IAM 角色。

Amazon ECS 會將此政策連接到服務角色，讓 Amazon ECS 能夠代表您對 Amazon EC2 執行個體或外部執行個體執行動作。

若要檢視此政策的許可，請參閱 AWS 受管政策參考中的 AmazonEC2ContainerServiceforEC2Role。

考量事項

使用 AmazonEC2ContainerServiceforEC2Role 受管 IAM 政策時，您應考慮下列建議和考量。

- 遵循授予最低權限的標準安全建議，您可以修改 AmazonEC2ContainerServiceforEC2Role 受管政策，以符合您的特定需求。如果您的使用案例不需要受管政策中授予的任何許可，請建立自訂政策並僅新增您需要的許可。例如，專為 Spot 執行個體耗盡而提供的 UpdateContainerInstancesState 許可。如果您的使用案例不需要該許可，請使用自訂政策將其排除。
- 在您的容器執行個體上執行的容器可透過「執行個體中繼資料」，存取所有提供給容器執行個體角色的許可。我們建議您將容器執行個體角色中的許可限制為受管 AmazonEC2ContainerServiceforEC2Role 政策所提供的最小許可清單。若您任務中的容器需要未列出的額外許可，我們建議您使用其專屬的 IAM 角色提供這些任務。如需詳細資訊，請參閱Amazon ECS 任務 IAM 角色。

  您可以防止 docker0 橋接器上的容器存取容器執行個體角色提供的許可。您可以執行此操作，同時仍允許 Amazon ECS 任務 IAM 角色 提供的許可，方法是在容器執行個體上執行下列 iptables 命令。此規則生效時，容器無法查詢執行個體中繼資料。此命令假設使用預設 Docker 橋接器組態，並且將無法使用採用 host 網路模式的容器。如需詳細資訊，請參閱網路模式。

  ```
  sudo yum install -y iptables-services; sudo iptables --insert DOCKER USER 1 --in-interface docker+ --destination 169.254.169.254/32 --jump DROP
  ```

  您必須將此 iptables 規則儲存在您的容器執行個體上，才能避免在重新開機時遭刪除。對於 Amazon ECS 最佳化 AMI，請使用下列命令。至於其他作業系統，請參閱該作業系統文件。

  - 對於 Amazon ECS 最佳化 Amazon Linux 2 AMI：

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

  - 對於 Amazon ECS 最佳化 Amazon Linux AMI：

    ```
    sudo service iptables save
    ```

AmazonEC2ContainerServiceEventsRole

您可將 AmazonEC2ContainerServiceEventsRole 政策連接到 IAM 身分。此政策授予許可，允許 Amazon EventBridge (之前為 CloudWatch Events) 代您執行任務。此政策可連接至在建立排程任務時指定的 IAM 角色。如需詳細資訊，請參閱Amazon ECS EventBridge IAM 角色。

若要檢視此政策的許可，請參閱 AWS 受管政策參考中的 AmazonEC2ContainerServiceEventsRole。

AmazonECSTaskExecutionRolePolicy

AmazonECSTaskExecutionRolePolicy 受管 IAM 政策會授予 Amazon ECS 容器代理程式和 AWS Fargate 容器代理程式代表您進行 AWS API 呼叫所需的許可。此政策可新增至您的任務執行 IAM 角色。如需詳細資訊，請參閱Amazon ECS 任務執行 IAM 角色。

若要檢視此政策的許可，請參閱 AWS 受管政策參考中的 AmazonECSTaskExecutionRolePolicy。

AmazonECSServiceRolePolicy

AmazonECSServiceRolePolicy 受管 IAM 政策可讓 Amazon Elastic Container Service 管理您的叢集。此政策可新增至您的任務執行 IAM 角色。如需詳細資訊，請參閱Amazon ECS 任務執行 IAM 角色。

若要檢視此政策的許可，請參閱 AWS 受管政策參考中的 AmazonECSServiceRolePolicy。

AmazonECSInfrastructureRolePolicyForServiceConnectTransportLayerSecur

您可將 AmazonECSInfrastructureRolePolicyForServiceConnectTransportLayerSecurity 政策附加至 IAM 實體。此政策會授予管理存取權 AWS Private Certificate Authority、Secrets Manager 和其他必要 AWS 服務，以代表您管理 Amazon ECS Service Connect TLS 功能。

若要檢視此政策的許可，請參閱 AWS 受管政策參考中的 AmazonECSInfrastructureRolePolicyForServiceConnectTransportLayerSecurity。

AWSApplicationAutoscalingECSServicePolicy

您無法將 AWSApplicationAutoscalingECSServicePolicy 連接至您的 IAM 實體。此政策會連接到服務連結角色，允許 Application Auto Scaling 代表您執行動作。如需詳細資訊，請參閱 Application Auto Scaling 的服務連結角色。

若要檢視此政策的許可，請參閱 AWS 受管政策參考中的 AWSApplicationAutoscalingECSServicePolicy。

AWSCodeDeployRoleForECS

您無法將 AWSCodeDeployRoleForECS 連接至您的 IAM 實體。此政策會連接到服務連結角色，可讓 CodeDeploy 代表您執行動作。如需詳細資訊，請參閱 《AWS CodeDeploy 使用者指南》中的建立 CodeDeploy 的服務角色。

若要檢視此政策的許可，請參閱 AWS 受管政策參考中的 AWSCodeDeployRoleForECS。

AWSCodeDeployRoleForECSLimited

您無法將 AWSCodeDeployRoleForECSLimited 連接至您的 IAM 實體。此政策會連接到服務連結角色，可讓 CodeDeploy 代表您執行動作。如需詳細資訊，請參閱 《AWS CodeDeploy 使用者指南》中的建立 CodeDeploy 的服務角色。

若要檢視此政策的許可，請參閱 AWS 受管政策參考中的 AWSCodeDeployRoleForECSLimited。

AmazonECSInfrastructureRolePolicyForVpcLattice

您可將 AmazonECSInfrastructureRolePolicyForVpcLattice 政策附加至 IAM 實體。此政策提供代表您管理 Amazon ECS 工作負載中 VPC Lattice 功能所需的其他 AWS 服務資源的存取權。

若要檢視此政策的許可，請參閱 AWS 受管政策參考中的 AmazonECSInfrastructureRolePolicyForVpcLattice。

提供代表您管理 Amazon ECS 工作負載中 VPC Lattice 功能所需的其他 AWS 服務資源的存取權。

受 AWS 管政策的 Amazon ECS 更新

檢視自此服務開始追蹤這些變更以來，Amazon ECS AWS 受管政策更新的詳細資訊。如需有關此頁面變更的自動提醒，請訂閱 Amazon ECS 文件歷史記錄頁面上的 RSS 摘要。

變更	描述	日期
新增 AmazonECSInfrastructureRolePolicyForVpcLattice	提供代表您管理 Amazon ECS 工作負載中 VPC Lattice 功能所需的其他 AWS 服務資源的存取權。	2024 年 11 月 18 日

變更	描述	日期
將許可新增至 AmazonECSInfrastructureRolePolicyForVolumes	AmazonECSInfrastructureRolePolicyForVolumes 已更新政策，讓客戶能夠從快照建立 Amazon EBS 磁碟區。	2024 年 10 月 10 日
新增許可至 the section called “AmazonECS_FullAccess”	已更新AmazonECS_FullAccess 政策，為名為 的角色新增 IAM 角色的iam:PassRole 許可ecsInfrastructureRole 。這是 建立的預設 IAM 角色 AWS Management Console ，旨在用作允許 Amazon ECS 管理連接到 ECS 任務的 Amazon EBS 磁碟區的 ECS 基礎設施角色。	2024 年 8 月 13 日
新增 AmazonECSInfrastructureRolePolicyForServiceConnectTransportLayerSecurity 政策	新增了新的 AmazonECSInfrastructureRolePolicyForServiceConnectTransportLayerSecurity 政策 AWS KMS，該政策提供對 Secrets Manager 的管理存取權 AWS Private Certificate Authority，並可讓 Amazon ECS Service Connect TLS 功能正常運作。	2024 年 1 月 22 日

變更	描述	日期
新增政策 AmazonECSInfrastructureRolePolicyForVolumes	已新增AmazonECSInfrastructureRolePolicyForVolumes 政策。此政策會授予 Amazon ECS 進行 AWS API 呼叫所需的許可，以管理與 Amazon ECS 工作負載相關聯的 Amazon EBS 磁碟區。	2024 年 1 月 11 日
新增許可至 AmazonECSServiceRolePolicy	AmazonECSServiceRolePolicy 受管 IAM 政策已更新為新的events許可和其他 autoscaling 和 autoscaling-plans 許可。	2023 年 12 月 4 日
將許可新增至 AmazonEC2ContainerServiceEventsRole	受AmazonECSServiceRolePolicy 管 IAM 政策已更新，以允許存取 DiscoverInstancesRevision API AWS Cloud Map 操作。	2023 年 10 月 4 日
新增許可至 AmazonEC2ContainerServiceforEC2Role	AmazonEC2ContainerServiceforEC2Role 政策已修改為新增ecs:TagResource 許可，其中包括將許可限制為新建立的叢集和已註冊的容器執行個體的條件。	2023 年 3 月 6 日

變更	描述	日期
新增許可至 the section called “AmazonECS_FullAccess”	AmazonECS_FullAccess 政策已修改為新增elasticloadbalancing:AddTags 許可，其中包括限制許可的條件，僅限於新建立的負載平衡器、目標群組、規則和建立的接聽程式。此許可不允許將標籤新增至任何已建立的 Elastic Load Balancing 資源。	2023 年 1 月 4 日
Amazon ECS 開始追蹤變更	Amazon ECS 開始追蹤其 AWS 受管政策的變更。	2021 年 6 月 8 日

Amazon Elastic Container Service 的逐步淘汰 AWS 受管 IAM 政策

下列 AWS 受管 IAM 政策會逐步淘汰。這些政策現在會由更新的政策取代。我們建議您更新使用者或角色，以使用更新的政策。

AmazonEC2ContainerServiceFullAccess

> **Important**
>
> AmazonEC2ContainerServiceFullAccess 受管 IAM 政策已於 2021 年 1 月 29 日淘汰，以使用 iam:passRole 許可回應安全問題。此許可授予帳戶中的角色對所有資源的存取權，包括憑證。因為政策已逐步淘汰，您無法將政策連接至任何新的使用者或角色。已連接政策的所有使用者或角色都可以繼續使用它。但我們建議您更新使用者或角色，以使用 AmazonECS_FullAccess 受管政策。如需詳細資訊，請參閱遷移至 AmazonECS_FullAccess 受管政策。

AmazonEC2ContainerServiceRole

> **Important**
>
> AmazonEC2ContainerServiceRole 受管 IAM 政策已逐步淘汰。現在已由 Amazon ECS 服務連結角色取代。如需詳細資訊，請參閱使用 Amazon ECS 的服務連結角色。

AmazonEC2ContainerServiceAutoscaleRole

> **Important**
>
> AmazonEC2ContainerServiceAutoscaleRole 受管 IAM 政策已逐步淘汰。現在已由 Amazon ECS 的 Application Auto Scaling 服務連結角色取代。如需詳細資訊，請參閱「Application Auto Scaling 使用者指南」中的適用於 Application Auto Scaling 的服務連結角色。

遷移至 AmazonECS_FullAccess 受管政策

AmazonEC2ContainerServiceFullAccess 受管 IAM 政策已於 2021 年 1 月 29 日淘汰，以使用 iam:passRole 許可回應安全問題。此許可授予帳戶中的角色對所有資源的存取權，包括憑證。因為政策已逐步淘汰，您無法將政策連接至任何新的群組、使用者或角色。已連接政策的所有群組、使用者或角色都可以繼續使用它。但我們建議您更新群組、使用者或角色，以使用 AmazonECS_FullAccess 受管政策。

AmazonECS_FullAccess 政策授予的許可包含以系統管理員身分使用 ECS 所需的完整許可清單。如果您目前使用政策中未授予AmazonEC2ContainerServiceFullAccess的許可AmazonECS_FullAccess，您可以將它們新增至內嵌政策陳述式。如需詳細資訊，請參閱AWS Amazon Elastic Container Service 的 受管政策。

使用下列步驟來判斷您是否有任何群組、使用者或角色目前正在使用 AmazonEC2ContainerServiceFullAccess 受管 IAM 政策。然後，更新它們以分開先前的政策並連接 AmazonECS_FullAccess 政策。

更新群組、使用者或角色以使用 AmazonECS_FullAccess 政策 (AWS Management Console)

1. 開啟位於 https://console.aws.amazon.com/iam/ 的 IAM 主控台。

2. 在導覽窗格中，選擇 Policies (政策)，然後搜尋並選取 AmazonEC2ContainerServiceFullAccess 政策。
3. 選擇 Policy usage (政策使用) 索引標籤，它顯示目前正在使用此政策的所有 IAM 角色。
4. 針對目前使用AmazonEC2ContainerServiceFullAccess政策的每個 IAM 角色，選取角色，並使用下列步驟來分離淘汰政策並連接AmazonECS_FullAccess政策。

   a. 在 Permissions (許可) 標籤中，選擇 AmazonEC2ContainerServiceFullAccess 政策旁邊的 X。
   b. 選擇新增許可。
   c. 選擇 Attach existing policies directly (直接連接現有政策)，搜尋並選取 AmazonECS_FullAccess 政策，然後選擇 Next: Review (下一步：檢閱)。
   d. 檢閱變更，然後選擇 Add permissions (新增許可)。
   e. 對使用 AmazonEC2ContainerServiceFullAccess 政策的每個群組、使用者或角色重複這些步驟。

更新群組、使用者或角色以使用 **AmazonECS_FullAccess** 政策 (AWS CLI)

1. 使用 generate-service-last-accessed-details命令來產生報告，其中包含上次使用淘汰政策時的詳細資訊。

```
aws iam generate-service-last-accessed-details \
     --arn arn:aws:iam::aws:policy/AmazonEC2ContainerServiceFullAccess
```

   輸出範例：

```
{
    "JobId": "32bb1fb0-1ee0-b08e-3626-ae83EXAMPLE"
}
```

2. 使用上一個輸出的任務 ID 搭配 get-service-last-accessed-details命令，以擷取服務的上次存取報告。此報告會顯示上次使用淘汰政策之 IAM 實體的 Amazon Resource Name (ARN)。

```
aws iam get-service-last-accessed-details \
      --job-id 32bb1fb0-1ee0-b08e-3626-ae83EXAMPLE
```

3. 請使用下列其中一個命令，將 AmazonEC2ContainerServiceFullAccess 政策從群組、使用者或角色中分開。

- detach-group-policy
- detach-role-policy
- detach-user-policy

4. 請使用下列其中一個命令，將 AmazonECS_FullAccess 政策連接至群組、使用者或角色。

- attach-group-policy
- attach-role-policy
- attach-user-policy

使用 Amazon ECS 的服務連結角色

Amazon Elastic Container Service 使用 AWS Identity and Access Management (IAM) 服務連結角色。服務連結角色是直接連結至 Amazon ECS 的一種特殊 IAM 角色類型。服務連結角色由 Amazon ECS 預先定義，內含該服務代您呼叫其他 AWS 服務所需的所有許可。

服務連結角色可讓設定 Amazon ECS 更為簡單，因為您不必手動新增必要的許可。Amazon ECS 定義其服務連結角色的許可，除非另有定義，否則僅有 Amazon ECS 可以擔任其角色。定義的許可包括信任政策和許可政策，且該許可政策無法附加至其他 IAM 實體。

如需支援服務連結角色的其他 服務的資訊，請參閱服務連結角色欄中AWS 與 IAM 搭配使用的服務，並尋找具有是的服務。選擇具有連結的是，以檢視該服務的服務連結角色文件。

Amazon ECS 的服務連結角色許可

Amazon ECS 使用名稱為 AWSServiceRoleForECS 的服務連結角色。

AWSServiceRoleForECS 服務連結角色信任下列服務可擔任該角色：

- ecs.amazonaws.com

名稱為 AmazonECSServiceRolePolicy 的角色許可政策允許 Amazon ECS 針對指定資源完成下列動作：

- 動作：使用適用於 Amazon ECS 任務的 awsvpc 網路模式時，Amazon ECS 會管理與任務關聯的彈性網路介面的生命週期。這也包括 Amazon ECS 新增至彈性網路介面的標籤。
- 動作：將負載平衡器與 Amazon ECS 服務搭配使用時，Amazon ECS 會使用負載平衡器來管理資源的註冊和取消註冊。

- 動作：使用 Amazon ECS 服務探索時，Amazon ECS 會管理所需的 Route 53 AWS Cloud Map 和資源，讓服務探索運作。
- 動作：使用 Amazon ECS 服務自動擴展時，Amazon ECS 會管理必要的自動擴展資源。
- 動作：Amazon ECS 會建立和管理 CloudWatch 警示和日誌串流，以協助監控 Amazon ECS 資源。
- 動作：使用 Amazon ECS Exec 時，Amazon ECS 會管理必要的許可，以便對您的任務啟動 Amazon ECS Exec 工作階段。
- 動作：使用 Amazon ECS Service Connect 時，Amazon ECS 會管理必要的 AWS Cloud Map 資源，以便使用此功能。
- 動作：使用 Amazon ECS 容量提供者時，Amazon ECS 會管理必要的許可，以便修改 Auto Scaling 群組及其 Amazon EC2 執行個體。

您必須設定許可，IAM 實體 (如使用者、群組或角色) 才可建立、編輯或刪除服務連結角色。如需詳細資訊，請參閱 IAM 使用者指南中的服務連結角色許可。

建立 Amazon ECS 的服務連結角色

在大多數情況下，您不需要手動建立一個服務連結角色。當您在 AWS Management Console、 或 AWS API 中建立叢集或建立 AWS CLI或更新服務時，Amazon ECS 會為您建立服務連結角色。如果您在建立叢集之後看不到 AWSServiceRoleForECS 角色，請執行下列步驟來修正問題：

- 驗證和設定許可，以允許 Amazon ECS 代表您建立、編輯或刪除服務連結角色。如需詳細資訊，請參閱 IAM 使用者指南中的服務連結角色許可。
- 重試叢集建立作業，或手動建立服務連結角色。

  您可以使用 IAM 主控台來建立 AWSServiceRoleForECS 服務連結角色。在 AWS CLI 或 AWS API 中，使用服務名稱建立`ecs.amazonaws.com`服務連結角色。如需詳細資訊，請參閱《IAM 使用者指南》中的「建立服務連結角色」。

> **Important**
>
> 此服務連結角色可以顯示在您的帳戶，如果您於其他服務中完成一項動作時，可以使用支援此角色的功能。

若您刪除此服務連結角色，之後需要再次建立，您可以在帳戶中使用相同程序重新建立角色。在建立叢集，或者建立或更新服務時，Amazon ECS 會再次為您建立服務連結角色。

如果您刪除這個服務連結角色，則可使用相同的 IAM 程序再次建立該角色。

編輯 Amazon ECS 的服務連結角色

Amazon ECS 不允許您編輯 AWSServiceRoleForECS 服務連結角色。因為有各種實體可能會參考服務連結角色，所以您無法在建立角色之後變更角色名稱。然而，您可使用 IAM 來編輯角色描述。如需詳細資訊，請參閱《IAM 使用者指南》中的更新服務連結角色。

刪除 Amazon ECS 的服務連結角色

若您不再使用需要服務連結角色的功能或服務，我們建議您刪除該角色。如此一來，您就沒有未主動監控或維護的未使用實體。然而，在手動刪除服務連結角色之前，您必須先清除資源。

> **Note**
>
> 若 Amazon ECS 服務在您試圖刪除資源時正在使用該角色，刪除可能會失敗。若此情況發生，請等待數分鐘後並再次嘗試操作。

檢查服務連結角色是否有作用中工作階段

1. 開啟位於 https://console.aws.amazon.com/iam/ 的 IAM 主控台。
2. 在導覽窗格中，選擇 Roles (角色)，然後選擇 AWSServiceRoleForECS 名稱 (而不是核取方塊)。
3. 在 Summary (摘要) 頁面上，選擇 Access Advisor (存取 Advisor)，然後檢閱服務連結角色的近期活動。

> **Note**
>
> 如果您不確定 Amazon ECS 是否正在使用 AWSServiceRoleForECS 角色，可嘗試刪除該角色。如果服務正在使用該角色，則刪除會失敗，而您可以檢視正在使用該角色的區域。如果服務正在使用該角色，您必須先等到工作階段結束，才能刪除該角色。您無法撤銷服務連結角色的工作階段。

若要移除 AWSServiceRoleForECS 服務連結角色使用的 Amazon ECS 資源

您必須刪除所有 AWS 區域中的所有 Amazon ECS 叢集，才能刪除 AWSServiceRoleForECS 角色。

1. 將所有區域中所有 Amazon ECS 服務的所需計數都縮減到 0，然後刪除該服務。如需詳細資訊，請參閱 使用主控台更新 Amazon ECS 服務 和 使用主控台刪除 Amazon ECS 服務。

2. 強制將所有容器執行個體從所有區域中的所有叢集取消註冊。如需詳細資訊，請參閱取消註冊 Amazon ECS 容器執行個體。
3. 刪除所有區域中的所有 Amazon ECS 叢集。如需詳細資訊，請參閱刪除 Amazon ECS 叢集。

使用 IAM 手動刪除服務連結角色

使用 IAM 主控台 AWS CLI、 或 AWS API 來刪除 AWSServiceRoleForECS 服務連結角色。如需詳細資訊，請參閱「IAM 使用者指南」中的刪除服務連結角色。

Amazon ECS 服務連結角色的支援區域

Amazon ECS 在所有提供服務的區域中支援使用服務連結角色。如需詳細資訊，請參閱 AWS 區域與端點。

Amazon ECS 的 IAM 角色

IAM 角色是您可以在帳戶中建立的另一種 IAM 身分，具有特定的許可。在 Amazon ECS 中，您可以建立角色，以授予許可給 Amazon ECS 資源，例如容器或服務。

Amazon ECS 所需的角色取決於任務定義啟動類型和您使用的功能。使用下表來判斷您需要哪些 Amazon ECS 的 IAM 角色。

角色	定義	需要時	其他資訊
任務執行角色	此角色允許 Amazon ECS 代表您使用其他 AWS 服務。	您的任務託管在 AWS Fargate外部執行個體上或外部執行個體上，並且： • 從 Amazon ECR 私有儲存庫提取容器映像。 • 從執行任務的帳戶的不同帳戶中的 Amazon ECR 私有儲存庫提取容器映像。	Amazon ECS 任務執行 IAM 角色

角色	定義	需要時	其他資訊
		• 使用 `awslogs` 日誌驅動程式將容器日誌傳送到 CloudWatch Logs。 您的任務託管在 AWS Fargate 或 Amazon EC2 執行個體上，並且： • 使用私有登錄驗證。 • 使用執行期監控。 • 任務定義會使用 Secrets Manager 秘密或 AWS Systems Manager 參數存放區參數來參考敏感資料。	
任務角色	此角色可讓您的應用程式程式碼（在容器上）使用其他 AWS 服務。	您的應用程式會存取其他服務 AWS ，例如 Amazon S3。	Amazon ECS 任務 IAM 角色
容器執行個體角色	此角色可讓您的 EC2 執行個體或外部執行個體向叢集註冊。	您的任務託管在 Amazon EC2 執行個體或外部執行個體上。	Amazon ECS 容器執行個體 IAM 角色
Amazon ECS Anywhere 角色	此角色可讓您的外部執行個體存取 AWS APIs。	您的任務託管在外部執行個體上。	Amazon ECS Anywhere IAM 角色

角色	定義	需要時	其他資訊
Amazon ECS CodeDeploy 角色	此角色允許 CodeDeploy 更新您的服務。	您可以使用 CodeDeploy 藍/綠部署類型來部署 服務。	Amazon ECS CodeDeploy IAM 角色
Amazon ECS EventBridge 角色	此角色允許 EventBridge 更新 服務。	您可以使用 EventBridge 規則和目標來排程任務。	Amazon ECS EventBridge IAM 角色
Amazon ECS 基礎設施角色	此角色允許 Amazon ECS 管理叢集中的基礎設施資源。	• 您想要將 Amazon EBS 磁碟區連接至 Fargate 或 EC2 啟動類型 Amazon ECS 任務。基礎設施角色允許 Amazon ECS 為您的任務管理 Amazon EBS 磁碟區。 • 您想要使用 Transport Layer Security (TLS) 來加密 Amazon ECS Service Connect 服務之間的流量。 • 您想要建立 VPC Lattice 目標群組。	Amazon ECS 基礎設施 IAM 角色

Amazon ECS 中 IAM 角色的最佳實務

建議您指派任務角色。其角色可以與執行 Amazon EC2 執行個體所使用的角色有所區別。為每項任務指派一個角色的做法符合最低權限存取原則，並且允許對動作和資源進行更精細的控制。

為任務指派 IAM 角色時，您必須使用下列信任政策，以便每個任務都可以承擔與 EC2 執行個體所使用角色不同的 IAM 角色。如此一來，您的任務就不會繼承 EC2 執行個體的角色。

```
{
```

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

當您將任務角色新增至任務定義時，Amazon ECS 容器代理程式會自動為該任務建立具有唯一憑證 ID 的字符 (例如 12345678-90ab-cdef-1234-567890abcdef)。然後，此字符和角色憑證會新增至代理程式的內部快取。代理程式會以憑證 ID 的 URI 填入容器中的環境變數 AWS_CONTAINER_CREDENTIALS_RELATIVE_URI (例如 /v2/credentials/12345678-90ab-cdef-1234-567890abcdef)。

您可以透過將環境變數附加至 Amazon ECS 容器代理程式的 IP 地址，然後在產生的字串上執行 curl 命令，從容器內手動擷取暫時性角色憑證。

```
curl 169.254.170.2$AWS_CONTAINER_CREDENTIALS_RELATIVE_URI
```

預期的輸出如下：

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

較新版本的 AWS SDKs 在進行 AWS API 呼叫時，會自動從AWS_CONTAINER_CREDENTIALS_RELATIVE_URI環境變數擷取這些登入資料。如需有關如何續約登入資料的資訊，請參閱在 rePost 上續約 AWS 登入資料。

輸出包含存取金鑰對，其中包含私密存取金鑰 ID 和您的應用程式用來存取 AWS 資源的私密金鑰。它也包含權杖， AWS 用於驗證登入資料是否有效。依預設，指派給使用任務角色之任務的憑證有效期為六小時。之後，Amazon ECS 容器代理程式會自動輪換。

任務執行角色

任務執行角色用於授予 Amazon ECS 容器代理程式代表您呼叫特定 AWS API 動作的許可。例如，當您使用 AWS Fargate時，Fargate 需要 IAM 角色，以允許其從 Amazon ECR 提取映像並將日誌寫入 CloudWatch Logs。當任務參考存放在 中的秘密時 AWS Secrets Manager，例如映像提取秘密，也需要 IAM 角色。

> **Note**
>
> 如果您以經身份驗證的使用者提取映像，則不太可能受到 Docker Hub 提取速率限制所發生的變更影響。如需詳細資訊，請參閱容器執行個體的私有登錄檔身分驗證。
> 透過使用 Amazon ECR 和 Amazon ECR Public，您可以避免 Docker 施加的限制。如果您從 Amazon ECR 提取映像，這也有助於縮短網路提取時間，並減少流量離開 VPC 時的資料傳輸變更。

> **Important**
>
> 當您使用 Fargate 時，您必須使用 `repositoryCredentials` 對私有映像登錄檔進行驗證。無法為 Fargate 上託管的任務設定 Amazon ECS 容器代理程式環境變數 `ECS_ENGINE_AUTH_TYPE` 或 `ECS_ENGINE_AUTH_DATA` 或修改 `ecs.config` 檔案。如需詳細資訊，請參閱任務的私有登錄檔身分驗證。

容器執行個體角色

`AmazonEC2ContainerServiceforEC2Role` 受管 IAM 政策包含下列許可。遵循授予最低權限的標準安全建議，`AmazonEC2ContainerServiceforEC2Role` 受管政策可作為指南。如果您的使用案例不需要受管政策中授予的任何許可，請建立自訂政策並僅新增您需要的許可。

- `ec2:DescribeTags` –)選用 ） 允許主體描述與 Amazon EC2 執行個體相關聯的標籤。Amazon ECS 容器代理程式會使用此許可，以支援資源標籤傳播。如需詳細資訊，請參閱如何標記資源。
- `ecs:CreateCluster` – （ 選用 ） 允許主體建立 Amazon ECS 叢集。Amazon ECS 容器代理程式會使用此許可來建立 `default` 叢集 (如果叢集尚不存在)。

- ecs:DeregisterContainerInstance –（選用）允許主體從叢集取消註冊 Amazon ECS 容器執行個體。Amazon ECS 容器代理程式不會呼叫此 API 操作，但此許可仍然有助於確保回溯相容性。
- ecs:DiscoverPollEndpoint –（必要）此動作會傳回 Amazon ECS 容器代理程式用來輪詢更新的端點。
- ecs:Poll –（必要）允許 Amazon ECS 容器代理程式與 Amazon ECS 控制平面通訊，以報告任務狀態變更。
- ecs:RegisterContainerInstance –（必要）允許主體向叢集註冊容器執行個體。Amazon ECS 容器代理程式使用此許可向叢集註冊 Amazon EC2 執行個體，並支援資源標籤傳播。
- ecs:StartTelemetrySession –（選用）允許 Amazon ECS 容器代理程式與 Amazon ECS 控制平面通訊，以報告每個容器和任務的健康資訊和指標。

  雖然不需要此許可，但建議您新增它，以允許容器執行個體指標開始擴展動作，並接收與運作狀態檢查命令相關的報告。
- ecs:TagResource –（選用）允許 Amazon ECS 容器代理程式在建立時標記叢集，並在容器執行個體註冊至叢集時標記容器執行個體。
- ecs:UpdateContainerInstancesState - 允許委託人修改 Amazon ECS 容器執行個體的狀態。Amazon ECS 容器代理程式使用此許可，用於 Spot 執行個體耗盡。
- ecs:Submit* –（必要）這包括 SubmitAttachmentStateChanges、SubmitContainerStateChange和 SubmitTaskStateChange API 動作。Amazon ECS 容器代理程式會使用它們，將每個資源的狀態變更報告給 Amazon ECS 控制平面。Amazon ECS 容器代理程式不會再使用該SubmitContainerStateChange許可，但仍保留此許可，以協助確保回溯相容性。
- ecr:GetAuthorizationToken –（選用）允許委託人擷取授權字符。授權字符代表您的 IAM 身分驗證憑證，且可用來存取 IAM 委託人有存取權限的任何 Amazon ECR 登錄檔。收到的授權字符的有效期為 12 小時。
- ecr:BatchCheckLayerAvailability –（選用）將容器映像推送至 Amazon ECR 私有儲存庫時，會檢查每個映像層，以確認是否已推送。如果已推送，則會略過映像層。
- ecr:GetDownloadUrlForLayer –（選用）從 Amazon ECR 私有儲存庫提取容器映像時，此 API 會針對尚未快取的每個映像層呼叫一次。
- ecr:BatchGetImage –（選用）從 Amazon ECR 私有儲存庫提取容器映像時，會呼叫此 API 一次以擷取映像資訊清單。
- logs:CreateLogStream –（選用）允許主體為指定的日誌群組建立 CloudWatch Logs 日誌串流。

- `logs:PutLogEvents` – （選用） 允許委託人將一批日誌事件上傳到指定的日誌串流。

下列政策包含必要的許可。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ecs:DiscoverPollEndpoint",
                "ecs:Poll",
                "ecs:RegisterContainerInstance",
                "ecs:UpdateContainerInstancesState",
                "ecs:Submit*"
            ],
            "Resource": "*"
        }
    ]
}
```

服務連結角色

您可以使用 Amazon ECS 的服務連結角色來授予 Amazon ECS 服務代表您呼叫其他服務 API 的許可。Amazon ECS 需要許可，才能使用目標群組建立和刪除網路介面、註冊和取消註冊目標。其也需要必要的許可，才能建立和刪除擴展政策。透過任務連結角色授予許可。在您第一次使用服務時代表您建立此角色。

> **Note**
>
> 如果您不小心刪除服務連結角色，您可以重新建立角色。如需指示，請參閱建立服務連結角色。

角色建議

建議您在設定任務 IAM 角色和政策時執行下列動作。

封鎖 Amazon EC2 中繼資料的存取

當您在 Amazon EC2 執行個體上執行任務時，強烈建議您封鎖 Amazon EC2 中繼資料的存取，以防止容器繼承指派給這些執行個體的角色。如果您的應用程式必須呼叫 AWS API 動作，請改用任務的 IAM 角色。

若要防止以橋接模式執行的任務存取 Amazon EC2 中繼資料，請執行以下命令或更新執行個體的使用者資料。如需更新執行個體使用者資料的詳細指示，請參閱此 AWS 支援文章。如需有關任務定義橋接模式的詳細資訊，請參閱任務定義網路模式。

```
sudo yum install -y iptables-services; sudo iptables --insert FORWARD 1 --in-interface
 docker+ --destination 169.254.169.254/32 --jump DROP
```

如需在重新開機後保留此變更，請執行下列 Amazon Machine Image (AMI) 專用的命令：

- Amazon Linux 2

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

- Amazon Linux

  ```
  sudo service iptables save
  ```

對於使用 awsvpc 網路模式的任務，請在 /etc/ecs/ecs.config 檔案中將環境變數 ECS_AWSVPC_BLOCK_IMDS 設定為 true。

您應該在 ecs-agent config 檔案中將 ECS_ENABLE_TASK_IAM_ROLE_NETWORK_HOST 變數設定為 false，以防止在 host 網路中執行的容器存取 Amazon EC2 中繼資料。

使用**awsvpc**網路模式

使用網路 awsvpc 網路模式來限制不同任務之間的流量，或是您的任務與 Amazon VPC 中執行的其他服務之間的流量。此新增其他的安全層。awsvpc 網路模式可為 Amazon EC2 上執行的任務提供任務層級的網路隔離。這是 上的預設模式 AWS Fargate。它是唯一可以用來將安全群組指派給任務的網路模式。

使用上次存取的資訊來精簡角色

建議您移除任何從未使用或已有一段時間未使用的動作。這樣可以防止不必要的存取發生。若要執行此作業，請檢閱 IAM 提供的上次存取資訊，然後移除從未使用或最近尚未使用的動作。您可以按照下列步驟執行此操作。

執行下列命令來產生報告，其中顯示參考政策的上次存取資訊：

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

使用輸出中的 `JobId` 來執行以下命令。執行這項操作之後，您可以檢視報告的結果。

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

如需詳細資訊，請參閱AWS 使用上次存取資訊在 中縮小許可範圍。

監控 AWS CloudTrail 可疑活動

您可以監控 AWS CloudTrail 任何可疑活動。大多數 AWS API 呼叫都會記錄 AWS CloudTrail 為事件。它們由 AWS CloudTrail Insights 進行分析，並且會提醒您任何與 `write` API 呼叫相關聯的可疑行為。這可能包括呼叫量的激增。這些警示包括異常活動發生的時間和為 API 貢獻的頂端身份 ARN 等資訊。

您可以查看事件的 `userIdentity` 屬性，來識別 AWS CloudTrail 中具有 IAM 角色的任務所執行的動作。在下列範例中，`arn` 包含擔任角色的名稱 `s3-write-go-bucket-role`,後面接著任務名稱 `7e9894e088ad416eb5cab92afExample`。

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

> **Note**
>
> 當擔任角色的任務在 Amazon EC2 容器執行個體上執行時，Amazon ECS 容器代理程式會將請求記錄到位於 `/var/log/ecs/audit.log.YYYY-MM-DD-HH` 格式地址的代理程式稽核日誌中。如需詳細資訊，請參閱任務 IAM 角色日誌和記錄追蹤的 Insights 事件。

Amazon ECS 任務執行 IAM 角色

任務執行角色會授予 Amazon ECS 容器和 Fargate 代理程式許可，以代表您進行 AWS API 呼叫。視任務需求而定，任務執行 IAM 角色是必要的項目。您可擁有多個任務執行角色，以用於與帳戶相關聯的不同用途和服務。

> **Note**
>
> 任務中的容器無法存取這些許可。有關應用程式需要執行的 IAM 許可，請參閱 Amazon ECS 任務 IAM 角色。

以下是任務執行 IAM 角色的常用案例：

- 您的任務託管在 AWS Fargate外部執行個體上或外部執行個體上，並且：
  - 從 Amazon ECR 私有儲存庫提取容器映像。
  - 從執行任務之帳戶的不同帳戶中的 Amazon ECR 私有儲存庫提取容器映像。
  - 使用 `awslogs` 日誌驅動程式將容器日誌傳送到 CloudWatch Logs。如需詳細資訊，請參閱將 Amazon ECS 日誌傳送至 CloudWatch 。
- 您的任務託管在 AWS Fargate 或 Amazon EC2 執行個體上，並且：
  - 使用私有登錄驗證。如需詳細資訊，請參閱私有登錄驗證許可。
  - 使用執行期監控。
  - 任務定義會使用 Secrets Manager 秘密或 AWS Systems Manager 參數存放區參數來參考敏感資料。如需詳細資訊，請參閱Secrets Manager 或 Systems Manager 許可。

> **Note**
>
> Amazon ECS 容器代理程式 1.16.0 版和更新版本支援任務執行角色。

Amazon ECS 提供名為 AmazonECSTaskExecutionRolePolicy 的受管政策，其中包含上述常見使用案例所需的許可。如需詳細資訊，請參閱《 AWS 受管政策參考指南》中的 AmazonECSTaskExecutionRolePolicy。針對特殊使用案例，可能需要將內嵌政策新增至您的任務執行角色

Amazon ECS 主控台會建立任務執行角色。您可以手動連接任務的 受管 IAM 政策，以允許 Amazon ECS 在引入任務時新增未來功能和增強功能的許可。您可以使用 IAM 主控台搜尋，`ecsTaskExecutionRole`並查看您的帳戶是否已有任務執行角色。如需詳細資訊，請參閱《IAM 使用者指南》中的 IAM 主控台搜尋。

如果您將映像提取為已驗證的使用者，則不太可能受到 Docker Hub 提取速率限制所發生的變更影響。如需詳細資訊，請參閱容器執行個體的私有登錄檔身分驗證。

透過使用 Amazon ECR 和 Amazon ECR Public，您可以避免 Docker 施加的限制。如果您從 Amazon ECR 提取映像，這也有助於縮短網路提取時間，並減少流量離開 VPC 時的資料傳輸變更。

當您使用 Fargate 時，您必須使用 `repositoryCredentials` 對私有映像登錄檔進行驗證。無法為 Fargate 上託管的任務設定 Amazon ECS 容器代理程式環境變數 `ECS_ENGINE_AUTH_TYPE` 或 `ECS_ENGINE_AUTH_DATA` 或修改 `ecs.config` 檔案。如需詳細資訊，請參閱任務的私有登錄檔身分驗證。

建立任務執行 角色

如果您的帳戶尚未擁有任務執行角色，請使用下列步驟來建立角色。

AWS Management Console

建立 Elastic Container Service (IAM 主控台) 的服務角色

1. 登入 AWS Management Console ，並在 https：//https://console.aws.amazon.com/iam/ 開啟 IAM 主控台。
2. 在 IAM 主控台的導覽窗格中，選擇角色，然後選擇建立角色。
3. 對於 Trusted entity type (信任的實體類型)，請選擇 AWS 服務。
4. 針對服務或使用案例，選擇彈性容器服務，然後選擇彈性容器服務任務使用案例。
5. 選擇 Next (下一步)。
6. 在新增許可區段中，搜尋 AmazonECSTaskExecutionRolePolicy，然後選取政策。
7. 選擇 Next (下一步)。
8. 針對角色名稱，輸入 ecsTaskExecutionRole。

9. 檢閱角色，然後選擇 Create role (建立角色)。

AWS CLI

將*#######*取代為您自己的資訊。

1. 建立名為 ecs-tasks-trust-policy.json 的檔案，其中包含用於 IAM 角色的信任政策。檔案應包含以下內容：

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

2. 使用在上一個步驟中建立的信任政策，建立名為 ecsTaskExecutionRole 的 IAM 角色。

```
aws iam create-role \
      --role-name ecsTaskExecutionRole \
      --assume-role-policy-document file://ecs-tasks-trust-policy.json
```

3. 將 AWS 受管AmazonECSTaskExecutionRolePolicy政策連接至 ecsTaskExecutionRole角色。

```
aws iam attach-role-policy \
      --role-name ecsTaskExecutionRole \
      --policy-arn arn:aws:iam::aws:policy/service-role/
AmazonECSTaskExecutionRolePolicy
```

建立角色之後，請為角色新增下列功能的其他許可。

功能	額外許可
使用 Secrets Manager 登入資料來存取您的容器映像私有儲存庫	私有登錄驗證許可
使用 Systems Manager 或 Secrets Manager 傳遞敏感資料	Secrets Manager 或 Systems Manager 許可
讓 Fargate 任務在介面端點上提取 Amazon ECR 映像	透過介面端點許可提取 Amazon ECR 映像的 Fargate 任務
Amazon S3 儲存貯體中的主機組態檔案	Amazon S3 檔案儲存許可
設定 Container Insights 以檢視 Amazon ECS 生命週期事件	設定 Container Insights 以檢視 Amazon ECS 生命週期事件所需的許可
在 Container Insights 中檢視 Amazon ECS 生命週期事件	在 Container Insights 中檢視 Amazon ECS 生命週期事件所需的許可

私有登錄驗證許可

若要將存取提供給您建立的秘密，請將以下許可字作為內嵌政策，新增到任務執行角色。如需詳細資訊，請參閱新增和移除 IAM 政策。

- `secretsmanager:GetSecretValue`
- `kms:Decrypt` - 只有在您的金鑰使用自訂 KMS 金鑰而非預設金鑰時，才需要此項目。您的自訂金鑰的 Amazon Resource Name (ARN) 必須新增為資源。

下列為新增許可的內嵌政策範例。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "kms:Decrypt",
                "secretsmanager:GetSecretValue"
            ],
```

```
            "Resource": [
                "arn:aws:secretsmanager:<region>:<aws_account_id>:secret:secret_name",
                "arn:aws:kms:<region>:<aws_account_id>:key/key_id"
            ]
        }
    ]
}
```

Secrets Manager 或 Systems Manager 許可

允許容器代理程式提取必要 AWS Systems Manager 或 Secrets Manager 資源的許可。如需詳細資訊，請參閱將敏感資料傳遞至 Amazon ECS 容器。

使用 Secrets Manager

若要允許存取您建立的 Secrets Manager 秘密，請將以下許可新增到任務執行角色。如需有關如何管理許可的相關資訊，請參閱《IAM 使用者指南》中的新增和移除 IAM 身分許可。

- `secretsmanager:GetSecretValue` – 如果參考 Secrets Manager 秘密，則此項目為必要。新增從 Secrets Manager 擷取密碼的許可。

下列政策範例新增必要許可。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "secretsmanager:GetSecretValue"
      ],
      "Resource": [
        "arn:aws:secretsmanager:region:aws_account_id:secret:secret_name"
      ]
    }
  ]
}
```

使用 Systems Manager

> **Important**
>
> 對於使用 EC2 啟動類型的任務，您必須使用 ECS 代理程式組態變數 `ECS_ENABLE_AWSLOGS_EXECUTIONROLE_OVERRIDE=true`，才能使用此功能。您可以在建立容器執行個體期間將其新增至 `./etc/ecs/ecs.config` 檔案，也可以將其新增至現有的執行個體，然後重新啟動 ECS 代理程式。如需詳細資訊，請參閱Amazon ECS 容器代理程式組態。

若要允許存取您建立的 Systems Manager Parameter Store 參數，請將以下許可當作政策，手動新增到任務執行角色。如需有關如何管理許可的相關資訊，請參閱《IAM 使用者指南》中的新增和移除 IAM 身分許可。

- `ssm:GetParameters` – 如果參考任務定義中的 Systems Manager Parameter Store 參數，則此項目為必要。新增擷取 Systems Manager 參數的許可。
- `secretsmanager:GetSecretValue` – 如果直接參考 Secrets Manager 秘密，或者 Systems Manager Parameter Store 參數參考任務定義中的 Secrets Manager 秘密，則此項目為必要。新增從 Secrets Manager 擷取密碼的許可。
- `kms:Decrypt` – 只有在秘密使用客戶受管金鑰而非預設金鑰時，此項目為必要。您的自訂金鑰的 ARN 應該新增為資源。新增解密客戶受管金鑰的許可。

下列政策範例新增必要許可：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ssm:GetParameters",
        "secretsmanager:GetSecretValue",
        "kms:Decrypt"
      ],
      "Resource": [
        "arn:aws:ssm:region:aws_account_id:parameter/parameter_name",
        "arn:aws:secretsmanager:region:aws_account_id:secret:secret_name",
        "arn:aws:kms:region:aws_account_id:key/key_id"
      ]
    }
```

```
  ]
}
```

透過介面端點許可提取 Amazon ECR 映像的 Fargate 任務

當 Amazon ECR 設定為使用介面 VPC 端點時，只要您啟動使用 Fargate 啟動類型的任務時 (此類型可從 Amazon ECR 中提取映像)，您就可以限制任務存取特定的 VPC 或 VPC 端點。建立任務執行角色，讓任務使用利用 IAM 條件金鑰的角色，即可完成此操作。

使用以下 IAM 全域條件金鑰，限制存取特定 VPC 或 VPC 端點。如需詳細資訊，請參閱 AWS 全域條件內容金鑰。

- `aws:SourceVpc` - 限制存取特定 VPC。您可以將 VPC 限制為託管任務和端點的 VPC。
- `aws:SourceVpce` - 限制存取特定 VPC 端點。

以下任務執行角色政策提供新增條件金鑰的範例：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ecr:GetAuthorizationToken",
                "logs:CreateLogStream",
                "logs:PutLogEvents"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ecr:BatchCheckLayerAvailability",
                "ecr:GetDownloadUrlForLayer",
                "ecr:BatchGetImage"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:sourceVpce": "vpce-xxxxxx",
                    "aws:sourceVpc": "vpc-xxxxx"
                }
```

```
            }
        }
    ]
}
```

Amazon S3 檔案儲存許可

當您指定託管在 Amazon S3 中的組態檔案時，任務執行角色必須包含組態檔案的`s3:GetObject`許可，以及檔案所在的 Amazon S3 儲存貯體的`s3:GetBucketLocation`許可。如需詳細資訊，請參閱《Amazon Simple Storage Service 使用者指南》中的 Amazon S3 政策動作。

下列範例政策新增從 Amazon S3 擷取檔案所需的許可。指定 Amazon S3 儲存貯體的名稱和組態檔案名稱。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject"
      ],
      "Resource": [
        "arn:aws:s3:::amzn-s3-demo-bucket/folder_name/config_file_name"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetBucketLocation"
      ],
      "Resource": [
        "arn:aws:s3:::amzn-s3-demo-bucket"
      ]
    }
  ]
}
```

設定 Container Insights 以檢視 Amazon ECS 生命週期事件所需的許可

任務角色需要下列許可，才能設定生命週期事件：

- events:PutRule

- events:PutTargets
- logs:CreateLogGroup

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "events:PutRule",
        "events:PutTargets",
        "logs:CreateLogGroup"
      ],
      "Resource": "*"
    }
  ]
}
```

在 Container Insights 中檢視 Amazon ECS 生命週期事件所需的許可

檢視生命週期事件需要以下許可: 將以下許可作為內嵌政策新增至任務執行角色。如需詳細資訊，請參閱新增和移除 IAM 政策。

- events:DescribeRule
- events:ListTargetsByRule
- logs:DescribeLogGroups

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "events:DescribeRule",
        "events:ListTargetsByRule",
        "logs:DescribeLogGroups"
      ],
      "Resource": "*"
    }
  ]
```

```
}
```

Amazon ECS 任務 IAM 角色

您的 Amazon ECS 任務可以擁有與其關聯的 IAM 角色。在 IAM 角色中授予的許可是由任務中執行的容器承繼。此角色可讓您的應用程式程式碼 （在容器上） 使用其他 AWS 服務。當您的應用程式存取其他 AWS 服務時，例如 Amazon S3，即需要任務角色。

> **Note**
>
> Amazon ECS 容器和 Fargate 代理程式無法存取這些許可。如需 Amazon ECS 提取容器映像和執行任務所需的 IAM 許可，請參閱 Amazon ECS 任務執行 IAM 角色。

以下是使用任務角色的好處：

- 登入資料隔離：容器只擷取所屬任務定義中定義的 IAM 角色登入資料，容器從未能存取屬於其他任務之其他容器的登入資料。
- 授權：未經授權的容器無法存取為其他任務定義的 IAM 角色登入資料。
- 稽核：可透過 CloudTrail 存取和事件記錄，以確保回溯性稽核。任務憑證有連接到工作階段的 `taskArn` 內容，所以 CloudTrail 日誌會顯示哪個任務使用哪個角色。

> **Note**
>
> 當您為任務指定 IAM 角色時， AWS CLI 或該任務容器中的其他 SDKs 只會使用任務角色提供的 AWS 登入資料，而不會再從其執行的 Amazon EC2 或外部執行個體繼承任何 IAM 許可。

建立任務 IAM 角色

為任務建立要使用的 IAM 政策時，該政策必須包含您希望任務中容器擔任的許可。您可以使用現有的 AWS 受管政策，也可以從頭開始建立自訂政策，以符合您的特定需求。如需詳細資訊，請參閱「IAM 使用者指南」中的建立 IAM 政策。

> **Important**
>
> 針對 Amazon ECS 任務 (適用於所有啟動類型)，建議在任務中使用 IAM 政策和角色。這些登入資料可讓任務提出 AWS API 請求，而無需呼叫 `sts:AssumeRole`擔任與任務已關聯的相同

角色。如果任務需要自己擔任的角色，則必須建立信任政策，明確允許該角色擔任它自己。如需詳細資訊，請參閱《IAM 使用者指南》中的更新角色信任政策。

在建立 IAM 政策後，您可以建立 IAM 角色，其中包含在 Amazon ECS 任務定義中參考的政策。您可以在 IAM 主控台中使用 Elastic Container Service Task (Elastic Container Service 任務) 使用案例來建立角色。然後，您可以將特定 IAM 政策連接至角色，為任務中的容器提供所需的許可。以下程序說明如何執行此操作。

如果您有多個任務定義或服務需要 IAM 許可，您應該考慮為每個特定任務定義或服務建立具有操作任務所需最低許可的角色，以便將您為每項任務提供的存取降至最低。

如需 區域服務端點的相關資訊，請參閱《 Amazon Web Services 一般參考 指南》中的服務端點。

IAM 任務角色必須具有信任政策，該政策會指定 `ecs-tasks.amazonaws.com` 服務。`sts:AssumeRole` 許可允許您的任務承繼與 Amazon EC2 執行個體使用角色不同的 IAM 角色。如此一來，您的任務就不會沿用與 Amazon EC2 執行個體相關聯的角色。信任政策範例如下。取代區域識別符，並指定您在啟動任務時使用 AWS 的帳號。

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Principal":{
            "Service":[
               "ecs-tasks.amazonaws.com"
            ]
         },
         "Action":"sts:AssumeRole",
         "Condition":{
            "ArnLike":{
            "aws:SourceArn":"arn:aws:ecs:us-west-2:111122223333:*"
            },
            "StringEquals":{
               "aws:SourceAccount":"111122223333"
            }
         }
      }
   ]
}
```

> **Important**
>
> 建立任務 IAM 角色時，建議您在信任關係`aws:SourceAccount`或與該角色相關聯的 IAM 政策中使用 或 `aws:SourceArn`條件索引鍵，進一步限制許可範圍，以防止混淆代理人安全問題。使用 `aws:SourceArn` 條件索引鍵來指定目前不受支援的特定叢集時，您應使用萬用字元來指定所有叢集。若要進一步了解混淆代理人問題以及如何保護 AWS 您的帳戶，請參閱《IAM 使用者指南》中的混淆代理人問題。

下列程序說明如何建立政策，以使用範例政策從 Amazon S3 擷取物件。將*#######*取代為您自己的值。

AWS Management Console

若要使用 JSON 政策編輯器來建立政策

1. 登入 AWS Management Console ，並在 https：//https://console.aws.amazon.com/iam/ 開啟 IAM 主控台。
2. 在左側的導覽窗格中，選擇 Policies (政策)。

   如果這是您第一次選擇 Policies (政策)，將會顯示 Welcome to Managed Policies (歡迎使用受管政策) 頁面。選擇 Get Started (開始使用)。
3. 在頁面頂端，選擇 Create policy (建立政策)。
4. 在政策編輯器中，選擇 JSON 選項。
5. 輸入下列 JSON 政策文件：

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":[
            "s3:GetObject"
         ],
         "Resource":[
            "arn:aws:s3:::my-task-secrets-bucket/*"
         ],
         "Condition":{
            "ArnLike":{
```

```
            "aws:SourceArn":"arn:aws:ecs:region:123456789012:*"
            },
            "StringEquals":{
               "aws:SourceAccount":"123456789012"
            }
         }
      }
   ]
}
```

6. 選擇 Next (下一步)。

> **Note**
>
> 您可以隨時切換視覺化與 JSON 編輯器選項。不過，如果您進行變更或在視覺化編輯器中選擇下一步，IAM 就可能會調整您的政策結構，以便針對視覺化編輯器進行最佳化。如需詳細資訊，請參閱 IAM 使用者指南中的調整政策結構。

7. 在檢視與建立頁面上，為您正在建立的政策輸入政策名稱與描述 (選用)。檢視此政策中定義的許可，來查看您的政策所授予的許可。
8. 選擇 Create policy (建立政策) 儲存您的新政策。

AWS CLI

將*******取代為您自己的值。

1. 建立稱為 s3-policy.json 的檔案，其中具有以下內容。

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":[
            "s3:GetObject"
         ],
         "Resource":[
            "arn:aws:s3:::my-task-secrets-bucket/*"
         ],
         "Condition":{
            "ArnLike":{
```

```
            "aws:SourceArn":"arn:aws:ecs:region:123456789012:*"
            },
            "StringEquals":{
               "aws:SourceAccount":"123456789012"
            }
         }
      }
   ]
}
```

2. 使用以下命令，使用 JSON 政策文件檔案建立 IAM 政策。

```
aws iam create-policy \
      --policy-name taskRolePolicy \
      --policy-document file://s3-policy.json
```

下列程序說明如何透過連接您建立的 IAM 政策來建立任務 IAM 角色。

AWS Management Console

建立 Elastic Container Service (IAM 主控台) 的服務角色

1. 登入 AWS Management Console ，並在 https：//https://console.aws.amazon.com/iam/ 開啟 IAM 主控台。
2. 在 IAM 主控台的導覽窗格中，選擇角色，然後選擇建立角色。
3. 對於 Trusted entity type (信任的實體類型)，請選擇 AWS 服務。
4. 針對服務或使用案例，選擇彈性容器服務，然後選擇彈性容器服務任務使用案例。
5. 選擇 Next (下一步)。
6. 針對新增許可，搜尋並選擇您建立的政策。
7. 選擇 Next (下一步)。
8. 針對 Role name (角色名稱)，輸入您的角色名稱。在此範例中，輸入 `AmazonECSTaskS3BucketRole` 來命名角色。
9. 檢閱角色，然後選擇 Create role (建立角色)。

AWS CLI

將*******取代為您自己的值。

1. 建立名為 的檔案ecs-tasks-trust-policy.json，其中包含要用於任務 IAM 角色的信任政策。檔案應包含下列項目。取代區域識別符，並指定您在啟動任務時使用 AWS 的帳號。

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Principal":{
            "Service":[
               "ecs-tasks.amazonaws.com"
            ]
         },
         "Action":"sts:AssumeRole",
         "Condition":{
            "ArnLike":{
            "aws:SourceArn":"arn:aws:ecs:us-west-2:111122223333:*"
            },
            "StringEquals":{
               "aws:SourceAccount":"111122223333"
            }
         }
      }
   ]
}
```

2. 使用在上一個步驟中建立的信任政策，建立名為 ecsTaskRole 的 IAM 角色。

```
aws iam create-role \
      --role-name ecsTaskRole \
      --assume-role-policy-document file://ecs-tasks-trust-policy.json
```

3. 擷取您使用以下命令建立的 IAM 政策 ARN。將 *taskRolePolicy* 取代為您建立的政策名稱。

```
aws iam list-policies --scope Local --query 'Policies[?
PolicyName==`taskRolePolicy`].Arn'
```

4. 將您建立的 IAM 政策連接至ecsTaskRole角色。policy-arn 將 取代為您建立之政策的 ARN。

```
aws iam attach-role-policy \
```

```
    --role-name ecsTaskRole \
    --policy-arn arn:aws:iam:111122223333:aws:policy/taskRolePolicy
```

建立角色之後，請為角色新增下列功能的其他許可。

功能	額外許可
使用 ECS Exec	ECS Exec 許可
使用 EC2 執行個體 (Windows 和 Linux)	Amazon EC2 執行個體額外組態
使用外部執行個體	外部執行個體額外組態
使用 Windows EC2 執行個體	Amazon EC2 Windows 執行個體額外組態

ECS Exec 許可

ECS Exec 功能需要任務 IAM 角色，才能授予容器在受管 SSM 代理程式 (execute-command 代理程式） 與 SSM 服務之間通訊所需的許可。您應將下列許可新增至任務 IAM 角色，並在任務定義中包含任務 IAM 角色。如需詳細資訊，請參閱《IAM 使用者指南》中的新增和移除 IAM 政策。

針對您的任務 IAM 角色使用下列政策來新增所需的 SSM 許可。

```
{
   "Version": "2012-10-17",
   "Statement": [
       {
       "Effect": "Allow",
       "Action": [
            "ssmmessages:CreateControlChannel",
            "ssmmessages:CreateDataChannel",
            "ssmmessages:OpenControlChannel",
            "ssmmessages:OpenDataChannel"
       ],
      "Resource": "*"
      }
   ]
}
```

Amazon EC2 執行個體額外組態

我們建議您將容器執行個體角色中的許可限制為 AmazonEC2ContainerServiceforEC2Role 受管 IAM 政策所使用的最小許可清單。

您的 Amazon EC2 執行個體至少需要 1.11.0 版本的容器代理程式才能使用任務角色；不過，我們建議您使用最新的容器代理程式版本。如需檢查代理程式版本及更新至最新版本的資訊，請參閱「更新 Amazon ECS 容器代理程式」。如果您使用 Amazon ECS 最佳化 AMI，您的執行個體至少需要 1.11.0-1 ecs-init套件。如果執行個體使用最新的 Amazon ECS 最佳化 AMI，則它們會包含所需的容器代理程式和 ecs-init。如需詳細資訊，請參閱Amazon ECS 最佳化 Linux AMIs。

如果您不是使用容器執行個體的 Amazon ECS 最佳化 AMI，請將 --net=host選項新增至啟動代理程式的docker run命令，以及所需組態的下列代理程式組態變數 （ 如需詳細資訊，請參閱 Amazon ECS 容器代理程式組態)：

ECS_ENABLE_TASK_IAM_ROLE=true

為具有 bridge 和 default 網路模式之容器的任務使用 IAM 角色。

ECS_ENABLE_TASK_IAM_ROLE_NETWORK_HOST=true

為具有 host 網路模式之容器的任務使用 IAM 角色。僅代理 1.12.0 版和更新版本支援此變數。

如需執行命令範例，請參閱「手動更新 Amazon ECS 容器代理程式 (適用於非 Amazon ECS 最佳化 AMI)」。您也需要在容器執行個體上設定下列聯網命令，以便任務中的容器可以擷取其 AWS 憑證：

```
sudo sysctl -w net.ipv4.conf.all.route_localnet=1
sudo iptables -t nat -A PREROUTING -p tcp -d 169.254.170.2 --dport 80 -j DNAT --to-
destination 127.0.0.1:51679
sudo iptables -t nat -A OUTPUT -d 169.254.170.2 -p tcp -m tcp --dport 80 -j REDIRECT --
to-ports 51679
```

您必須將這些 iptables 規則儲存在您的容器執行個體上，才能避免在重新開機時將其刪除。您可以使用 iptables-save 和 iptables-restore 命令來儲存 iptables 規則以及在啟動時將其還原。如需詳細資訊，請參閱特定的作業系統文件。

針對任務中採用 awsvpc 網路模式的容器，如欲避免其存取提供給 Amazon EC2 執行個體設定檔的憑證資訊 (同時仍然允許任務角色所提供的許可)，請在代理程式組態檔案中將 ECS_AWSVPC_BLOCK_IMDS 代理程式組態變數設定為 true，並重新啟動代理程式。如需詳細資訊，請參閱Amazon ECS 容器代理程式組態。

至於任務中採用 bridge 網路模式的容器，則可在 Amazon EC2 執行個體上執行下列 iptables 命令，藉此避免其存取提供給 Amazon EC2 執行個體設定檔的憑證資訊 (同時仍然允許任務角色所提供的許可)。此命令不會影響任務中採用 host 或 awsvpc 網路模式的容器。如需詳細資訊，請參閱網路模式。

- ```
  sudo yum install -y iptables-services; sudo iptables --insert DOCKER-USER 1 --in-interface docker+ --destination 169.254.169.254/32 --jump DROP
  ```

  您必須將此 iptables 規則儲存在您的 Amazon EC2 執行個體上，才能避免在重新開機時遭刪除。使用 Amazon ECS 最佳化 AMI 時，您可以使用下列命令。至於其他作業系統，請參閱該作業系統的文件。

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

外部執行個體額外組態

您的外部執行個體至少需要 1.11.0 版本的容器代理程式才能使用任務 IAM 角色；不過，我們建議您使用最新的容器代理程式版本。如需檢查代理程式版本及更新至最新版本的資訊，請參閱「更新 Amazon ECS 容器代理程式」。如果使用的是 Amazon ECS 最佳化 AMI，您的執行個體至少需要 ecs-init 套件的 1.11.0-1 版。如果執行個體使用最新的 Amazon ECS 最佳化 AMI，則它們會包含所需的容器代理程式和 ecs-init。如需詳細資訊，請參閱Amazon ECS 最佳化 Linux AMIs。

如果您不是使用容器執行個體的 Amazon ECS 最佳化 AMI，請將 --net=host選項新增至啟動代理程式的docker run命令，以及所需組態的下列代理程式組態變數 （ 如需詳細資訊，請參閱 Amazon ECS 容器代理程式組態) ：

ECS_ENABLE_TASK_IAM_ROLE=true

為具有 bridge 和 default 網路模式之容器的任務使用 IAM 角色。

ECS_ENABLE_TASK_IAM_ROLE_NETWORK_HOST=true

為具有 host 網路模式之容器的任務使用 IAM 角色。僅代理 1.12.0 版和更新版本支援此變數。

如需執行命令範例，請參閱「手動更新 Amazon ECS 容器代理程式 (適用於非 Amazon ECS 最佳化 AMI)」。您也需要在容器執行個體上設定下列聯網命令，以便任務中的容器可以擷取其 AWS 憑證：

```
sudo sysctl -w net.ipv4.conf.all.route_localnet=1
```
```

```
sudo iptables -t nat -A PREROUTING -p tcp -d 169.254.170.2 --dport 80 -j DNAT --to-
destination 127.0.0.1:51679
sudo iptables -t nat -A OUTPUT -d 169.254.170.2 -p tcp -m tcp --dport 80 -j REDIRECT --
to-ports 51679
```

您必須將這些 iptables 規則儲存在您的容器執行個體上，才能避免在重新開機時將其刪除。您可以使用 iptables-save 和 iptables-restore 命令來儲存 iptables 規則以及在啟動時將其還原。如需詳細資訊，請參閱特定的作業系統文件。

Amazon EC2 Windows 執行個體額外組態

> **Important**
>
> 這僅適用於使用任務角色的 EC2 上的 Windows 容器。

具有 Windows 功能的任務角色需要在 EC2 上進行其他組態。

- 當您啟動容器執行個體時，您必須在容器執行個體使用者資料指令碼中設定 `-EnableTaskIAMRole` 選項。`EnableTaskIAMRole` 會開啟任務的任務 IAM 角色功能。例如：

  ```
  <powershell>
  Import-Module ECSTools
  Initialize-ECSAgent -Cluster 'windows' -EnableTaskIAMRole
  </powershell>
  ```

- 您必須使用「Amazon ECS 容器引導指令碼」中提供的聯網命令引導您的容器。
- 您必須為您的任務建立 IAM 角色和政策。如需詳細資訊，請參閱建立任務 IAM 角色。
- 任務登入資料供應者的 IAM 角色在容器執行個體上使用連接埠 80。因此，如果您在容器執行個體上設定任務 IAM 角色，您的容器無法在任何連接埠映射中為主機連接埠使用連接埠 80。若要在連接埠 80 公開您的容器，我們建議您為其設定使用負載平衡的服務。您可以在負載平衡器上使用連接埠 80。這樣做可以將流量路由到容器執行個體上的另一個主機連接埠。如需詳細資訊，請參閱使用負載平衡來分發 Amazon ECS 服務流量。
- 如果 Windows 執行個體已重新啟動，您必須刪除代理界面並再次初始化 Amazon ECS 容器代理程式，以重新啟動登入資料代理。

Amazon ECS 容器引導指令碼

容器必須使用所需的聯網命令加以引導，才能夠存取容器執行個體的登入資料代理，取得登入資料。以下程式碼範例指令碼應該在您的容器啟動時於其上執行。

> **Note**
>
> 當您在 Windows 上使用 awsvpc 網路模式時，您不需要執行此指令碼。

如果您執行包含 Powershell 的 Windows 容器，請使用下列指令碼：

```
# Copyright Amazon.com Inc. or its affiliates. All Rights Reserved.
#
# Licensed under the Apache License, Version 2.0 (the "License"). You may
# not use this file except in compliance with the License. A copy of the
# License is located at
#
# http://aws.amazon.com/apache2.0/
#
# or in the "license" file accompanying this file. This file is distributed
# on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either
# express or implied. See the License for the specific language governing
# permissions and limitations under the License.

$gateway = (Get-NetRoute | Where { $_.DestinationPrefix -eq '0.0.0.0/0' } | Sort-Object
 RouteMetric | Select NextHop).NextHop
$ifIndex = (Get-NetAdapter -InterfaceDescription "Hyper-V Virtual Ethernet*" | Sort-
Object | Select ifIndex).ifIndex
New-NetRoute -DestinationPrefix 169.254.170.2/32 -InterfaceIndex $ifIndex -NextHop
 $gateway -PolicyStore ActiveStore # credentials API
New-NetRoute -DestinationPrefix 169.254.169.254/32 -InterfaceIndex $ifIndex -NextHop
 $gateway -PolicyStore ActiveStore # metadata API
```

如果您執行只有命令 shell 的 Windows 容器，請使用下列指令碼：

```
# Copyright Amazon.com Inc. or its affiliates. All Rights Reserved.
#
# Licensed under the Apache License, Version 2.0 (the "License"). You may
# not use this file except in compliance with the License. A copy of the
# License is located at
#
```

```
# http://aws.amazon.com/apache2.0/
#
# or in the "license" file accompanying this file. This file is distributed
# on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either
# express or implied. See the License for the specific language governing
# permissions and limitations under the License.

for /f "tokens=1" %i  in ('netsh interface ipv4 show interfaces ^| findstr /x /r
 ".*vEthernet.*"') do set interface=%i
for /f "tokens=3" %i  in ('netsh interface ipv4 show addresses %interface% ^| findstr /
x /r ".*Default.Gateway.*"') do set gateway=%i
netsh interface ipv4 add route prefix=169.254.170.2/32 interface="%interface%"
 nexthop="%gateway%" store=active # credentials API
netsh interface ipv4 add route prefix=169.254.169.254/32 interface="%interface%"
 nexthop="%gateway%" store=active # metadata API
```

Amazon ECS 容器執行個體 IAM 角色

Amazon ECS 容器執行個體 (包括 Amazon EC2 和外部執行個體) 會執行 Amazon ECS 容器代理程式，並要求 IAM 角色，才能讓服務知道該代理程式為您所有。在您啟動容器執行個體並將它們註冊到叢集之前，您必須先為要使用的容器執行個體建立 IAM 角色。角色會在您用來登入主控台或執行 AWS CLI 命令的帳戶中建立。

> **Important**
>
> 如果您要向叢集註冊外部執行個體，則您使用的 IAM 角色也需要 Systems Manager 許可。如需詳細資訊，請參閱Amazon ECS Anywhere IAM 角色。

Amazon ECS 提供 AmazonEC2ContainerServiceforEC2Role 受管 IAM 政策，其中包含使用完整 Amazon ECS 功能集所需的許可。此受管政策可連接至 IAM 角色，並與您的容器執行個體相關聯。或者，您可以在建立要使用的自訂政策時，使用受管政策作為指南。容器執行個體角色提供 Amazon ECS 容器代理程式和 Docker 協助程式代表您呼叫 AWS APIs所需的許可。如需受管政策的詳細資訊，請參閱 AmazonEC2ContainerServiceforEC2Role。

Amazon ECS 支援使用受支援的 Amazon EC2 執行個體類型，以更高的 ENI 密度來啟動容器執行個體。當您使用此功能時，我們建議您建立 2 個容器執行個體角色。為一個角色啟用awsvpcTrunking帳戶設定，並將該角色用於需要 ENI 中繼的任務。如需awsvpcTrunking帳戶設定的相關資訊，請參閱使用帳戶設定存取 Amazon ECS功能。

建立容器執行個體角色

> **Important**
>
> 如果您要向叢集註冊外部執行個體，請參閱 Amazon ECS Anywhere IAM 角色。

您可手動建立角色並連接容器執行個體的受管 IAM 政策，讓 Amazon ECS 為日後推出的功能和改進新增許可。如有需要，請使用下列程序連接 受管 IAM 政策。

AWS Management Console

建立 Elastic Container Service (IAM 主控台） 的服務角色

1. 登入 AWS Management Console ，並在 https：//https://console.aws.amazon.com/iam/ 開啟 IAM 主控台。
2. 在 IAM 主控台的導覽窗格中，選擇角色，然後選擇建立角色。
3. 對於 Trusted entity type (信任的實體類型)，請選擇 AWS 服務。
4. 針對服務或使用案例，選擇彈性容器服務，然後選擇彈性容器服務的 EC2 角色使用案例。
5. 選擇 Next (下一步)。
6. 在許可政策區段中，確認已選取 AmazonEC2ContainerServiceforEC2Role 政策。

   > **Important**
   >
   > AmazonEC2ContainerServiceforEC2Role 受管政策應連接至容器執行個體 IAM 角色，否則您會使用 來建立叢集 AWS Management Console 而收到錯誤。

7. 選擇 Next (下一步)。
8. 針對角色名稱，輸入 ecsInstanceRole
9. 檢閱角色，然後選擇 Create role (建立角色)。

AWS CLI

將*******取代為您自己的值。

1. 建立稱為 `instance-role-trust-policy.json` 的檔案，其中具有以下內容。

```
{
```

```
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": { "Service": "ec2.amazonaws.com"},
      "Action": "sts:AssumeRole"
    }
  ]
}
```

2. 使用以下命令，使用信任政策文件建立執行個體 IAM 角色。

```
aws iam create-role \
    --role-name ecsInstanceRole \
    --assume-role-policy-document file://instance-role-trust-policy.json
```

3. 使用 create-instance-profile 命令建立名為 ecsInstanceRole-profile 的執行個體設定檔。

```
aws iam create-instance-profile --instance-profile-name ecsInstanceRole-profile
```

回應範例

```
{
    "InstanceProfile": {
        "InstanceProfileId": "AIPAJTLBPJLEGREXAMPLE",
        "Roles": [],
        "CreateDate": "2022-04-12T23:53:34.093Z",
        "InstanceProfileName": "ecsInstanceRole-profile",
        "Path": "/",
        "Arn": "arn:aws:iam::123456789012:instance-profile/ecsInstanceRole-
profile"
    }
}
```

4. 將 *ecsInstanceRole* 角色新增至 *ecsInstanceRole-profile* 執行個體設定檔。

```
aws iam add-role-to-instance-profile \
    --instance-profile-name ecsInstanceRole-profile \
    --role-name ecsInstanceRole
```

5. 使用下列命令將 AmazonEC2ContainerServiceRoleForEC2Role 受管政策連接至角色。

```
aws iam attach-role-policy \
    --policy-arn arn:aws:iam::aws:policy/service-role/
AmazonEC2ContainerServiceforEC2Role \
    --role-name ecsInstanceRole
```

建立角色之後，請為角色新增下列功能的其他許可。

功能	額外許可
Amazon ECR 具有容器映像	Amazon ECR 許可
讓 CloudWatch Logs 監控容器執行個體	監控容器執行個體許可
Amazon S3 儲存貯體中的主機組態檔案	Amazon S3 唯讀存取

Amazon ECR 許可

搭配容器執行個體使用的 Amazon ECS 容器執行個體角色，必須具有下列 Amazon ECR 的 IAM 政策許可。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ecr:BatchCheckLayerAvailability",
                "ecr:BatchGetImage",
                "ecr:GetDownloadUrlForLayer",
                "ecr:GetAuthorizationToken"
            ],
            "Resource": "*"
        }
    ]
}
```

如果您的容器執行個體使用 AmazonEC2ContainerServiceforEC2Role 受管政策，則您的角色具有適當的許可。若要檢查您的角色是否支援 Amazon ECR，請參閱 Amazon Elastic Container Service 開發人員指南中的 Amazon ECS 容器執行個體 IAM 角色。

Amazon S3 唯讀存取

在 Amazon S3 中的私有儲存貯體內存放組態資訊，並對容器執行個體 IAM 角色授予唯讀存取權，這是一種允許容器執行個體在啟動時進行設定的安全且便利方法。您可以在私有儲存貯體中存放ecs.config檔案的複本，使用 Amazon EC2 使用者資料安裝 ， AWS CLI 然後在執行個體啟動/etc/ecs/ecs.config時將您的組態資訊複製到 。

如需建立 ecs.config 檔案、將其存放在 Amazon S3 以及使用此組態啟動執行個體的詳細資訊，請參閱 在 Amazon S3 中存放 Amazon ECS 容器執行個體組態。

您可以使用下列 AWS CLI 命令來允許容器執行個體角色的 Amazon S3 唯讀存取。將 *ecsInstanceRole* 取代為您建立的角色名稱。

```
aws iam attach-role-policy \
      --role-name ecsInstanceRole \
      --policy-arn arn:aws:iam::aws:policy/AmazonS3ReadOnlyAccess
```

您也可以使用 IAM 主控台將 Amazon S3 唯讀存取 (AmazonS3ReadOnlyAccess) 新增至您的角色。如需詳細資訊，請參閱AWS Identity and Access Management 《 使用者指南》中的更新角色的許可。

監控容器執行個體許可

在容器執行個體將日誌資料傳送至 CloudWatch Logs 之前，您必須建立 IAM 政策，以允許 Amazon ECS 代理程式將客戶的應用程式日誌寫入 CloudWatch （通常透過awslogs驅動程式處理）。建立政策之後，請將該政策連接至 ecsInstanceRole。

AWS Management Console

若要使用 JSON 政策編輯器來建立政策

1. 登入 AWS Management Console ，並在 https：//https://console.aws.amazon.com/iam/ 開啟 IAM 主控台。
2. 在左側的導覽窗格中，選擇 Policies (政策)。

   如果這是您第一次選擇 Policies (政策)，將會顯示 Welcome to Managed Policies (歡迎使用受管政策) 頁面。選擇 Get Started (開始使用)。
3. 在頁面頂端，選擇 Create policy (建立政策)。

4. 在政策編輯器中，選擇 JSON 選項。
5. 輸入下列 JSON 政策文件：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogGroup",
                "logs:CreateLogStream",
                "logs:PutLogEvents",
                "logs:DescribeLogStreams"
            ],
            "Resource": ["arn:aws:logs:*:*:*"]
        }
    ]
}
```

6. 選擇 Next (下一步)。

> **Note**
>
> 您可以隨時切換視覺化與 JSON 編輯器選項。不過，如果您進行變更或在視覺化編輯器中選擇下一步，IAM 就可能會調整您的政策結構，以便針對視覺化編輯器進行最佳化。如需詳細資訊，請參閱 IAM 使用者指南中的調整政策結構。

7. 在檢視與建立頁面上，為您正在建立的政策輸入政策名稱與描述 (選用)。檢視此政策中定義的許可，來查看您的政策所授予的許可。
8. 選擇 Create policy (建立政策) 儲存您的新政策。

建立政策之後，請將政策連接至容器執行個體角色。如需有關如何將政策連接至角色的資訊，請參閱AWS Identity and Access Management 《 使用者指南》中的更新角色的許可。

AWS CLI

1. 建立稱為 `instance-cw-logs.json` 的檔案，其中具有以下內容。

```
{
    "Version": "2012-10-17",
    "Statement": [
```

```
        {
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogGroup",
                "logs:CreateLogStream",
                "logs:PutLogEvents",
                "logs:DescribeLogStreams"
            ],
            "Resource": ["arn:aws:logs:*:*:*"]
        }
    ]
}
```

2. 使用以下命令，使用 JSON 政策文件檔案建立 IAM 政策。

```
aws iam create-policy \
      --policy-name cwlogspolicy \
      --policy-document file://instance-cw-logs.json
```

3. 擷取您使用以下命令建立的 IAM 政策 ARN。將 *cwlogspolicy* 取代為您建立的政策名稱。

```
aws iam list-policies --scope Local --query 'Policies[?
PolicyName==`cwlogspolicy`].Arn'
```

4. 使用下列命令，使用政策 ARN 將政策連接至容器執行個體 IAM 角色。

```
aws iam attach-role-policy \
      --role-name ecsInstanceRole \
      --policy-arn arn:aws:iam:111122223333:aws:policy/cwlogspolicy
```

Amazon ECS Anywhere IAM 角色

當您向叢集註冊現場部署伺服器或虛擬機器 (VM) 時，伺服器或 VM 需要 IAM 角色才能與 AWS APIs 通訊。您只需要為每個 AWS 帳戶建立此 IAM 角色一次。不過，此 IAM 角色必須與您向叢集註冊的每個伺服器或虛擬機器產生關聯。此角色為 ECSAnywhereRole。您可以手動建立此角色。或者，當您在 AWS Management Console中註冊外部執行個體時，Amazon ECS 可以代表您建立角色。您可以使用 IAM 主控台搜尋來搜尋 ecsAnywhereRole，並查看您的帳戶是否已有 角色。如需詳細資訊，請參閱《IAM 使用者指南》中的 IAM 主控台搜尋。

AWS 提供兩個 受管 IAM 政策，可用於建立 ECS Anywhere IAM 角色、AmazonSSMManagedInstanceCore和 AmazonEC2ContainerServiceforEC2Role政

策。AmazonEC2ContainerServiceforEC2Role 政策包含的許可提供的存取權很可能比您需要的更多。因此，視您的特定使用案例而定，建議您建立自訂政策，僅在該政策中新增所需許可。如需詳細資訊，請參閱 Amazon ECS 容器執行個體 IAM 角色。

任務執行 IAM 角色會授予 Amazon ECS 容器代理程式許可，以代表您進行 AWS API 呼叫。使用任務執行 IAM 角色時，必須在您的任務定義中指定它。如需詳細資訊，請參閱Amazon ECS 任務執行 IAM 角色。

如果以下任何條件適用，則需要任務執行角色：

- 使用 awslogs 日誌驅動程式將容器日誌傳送到 CloudWatch Logs。
- 您的任務定義會指定託管在 Amazon ECR 私有儲存庫中的容器映像。不過，如果與外部執行個體相關聯的ECSAnywhereRole角色也包含從 Amazon ECR 提取映像所需的許可，則您的任務執行角色不需要包含它們。

建立 Amazon ECS Anywhere 角色

使用您自己的資訊取代*#######*。

1. 使用下列信任政策建立名為 ssm-trust-policy.json的本機檔案。

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

2. 使用下列 AWS CLI 命令建立角色並連接信任政策。

```
aws iam create-role --role-name ecsAnywhereRole --assume-role-policy-document
 file://ssm-trust-policy.json
```

3. 使用以下命令連接 AWS 受管政策。

```
aws iam attach-role-policy --role-name ecsAnywhereRole --policy-arn
 arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore
```

```
aws iam attach-role-policy --role-name ecsAnywhereRole --policy-arn
 arn:aws:iam::aws:policy/service-role/AmazonEC2ContainerServiceforEC2Role
```

您也可以使用 IAM 自訂信任政策工作流程來建立角色。如需詳細資訊，請參閱《IAM 使用者指南》中的使用自訂信任政策（主控台）建立角色。

Amazon ECS 基礎設施 IAM 角色

Amazon ECS 基礎設施 IAM 角色允許 Amazon ECS 代表您管理叢集中的基礎設施資源，並在下列情況使用：

- 您想要將 Amazon EBS 磁碟區連接至 Fargate 或 EC2 啟動類型 Amazon ECS 任務。基礎設施角色允許 Amazon ECS 為您的任務管理 Amazon EBS 磁碟區。
- 您想要使用 Transport Layer Security (TLS) 來加密 Amazon ECS Service Connect 服務之間的流量。
- 您想要建立 Amazon VPC Lattice 目標群組。

當 Amazon ECS 擔任此角色代表您採取動作時，事件將顯示於 中 AWS CloudTrail。如果 Amazon ECS 使用 角色來管理連接至任務的 Amazon EBS 磁碟區，CloudTrail 日誌`roleSessionName`將是 `ECSTaskVolumesForEBS`。如果角色用於加密 Service Connect 服務之間的流量，CloudTrail 日誌`roleSessionName`將為 `ECSServiceConnectForTLS`。如果角色用於為 VPC Lattice 建立目標群組，CloudTrail 日誌`roleSessionName`將為 `ECSNetworkingWithVPCLattice`。您可以使用此名稱，透過篩選使用者名稱，在 CloudTrail 主控台中搜尋事件。

Amazon ECS 提供受管政策，其中包含磁碟區連接和 TLS 所需的許可。如需詳細資訊，請參閱 AWS 受管政策參考指南中的 AmazonECSInfrastructureRolePolicyForVolumes、AmazonECSInfrastructureRolePolicyForServiceConnectT 和 AmazonECSInfrastructureRolePolicyForVpcLattice。

建立 Amazon ECS 基礎設施角色

使用您自己的資訊取代*#######*。

1. 建立名為 `ecs-infrastructure-trust-policy.json` 的檔案，其中包含用於 IAM 角色的信任政策。檔案應包含以下內容：

```
{
  "Version": "2012-10-17",
```

```
  "Statement": [
    {
      "Sid": "AllowAccessToECSForInfrastructureManagement",
      "Effect": "Allow",
      "Principal": {
        "Service": "ecs.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

2. 使用以下 AWS CLI 命令，ecsInfrastructureRole使用您在上一個步驟中建立的信任政策來建立名為 的角色。

```
aws iam create-role \
      --role-name ecsInfrastructureRole \
      --assume-role-policy-document file://ecs-infrastructure-trust-policy.json
```

3. 根據您的使用案例，將 AWS 受管 AmazonECSInfrastructureRolePolicyForVolumes、AmazonECSInfrastructureRolePolicyForServiceConnectTransportLayerSecurity或AmazonECSInfrastructureRolePolicyForVpcLattice政策連接至ecsInfrastructureRole角色。

```
aws iam attach-role-policy \
      --role-name ecsInfrastructureRole \
      --policy-arn arn:aws:iam::aws:policy/service-role/
AmazonECSInfrastructureRolePolicyForVolumes
```

```
aws iam attach-role-policy \
      --role-name ecsInfrastructureRole \
      --policy-arn arn:aws:iam::aws:policy/service-role/
AmazonECSInfrastructureRolePolicyForServiceConnectTransportLayerSecurity
```

您也可以使用 IAM 主控台的自訂信任政策工作流程來建立角色。如需詳細資訊，請參閱《IAM 使用者指南》中的使用自訂信任政策（主控台）建立角色。

Important

如果 Amazon ECS 正在使用 ECS 基礎設施角色來管理連接到任務的 Amazon EBS 磁碟區，請在停止使用 Amazon EBS 磁碟區的任務之前確保下列事項。

- 不會刪除角色。
- 角色的信任政策不會修改為移除 Amazon ECS 存取 (`ecs.amazonaws.com`)。
- `AmazonECSInfrastructureRolePolicyForVolumes` 未移除受管政策。如果您必須修改角色的許可，請保留至少 `ec2:DetachVolume`、 `ec2:DeleteVolume`和 `ec2:DescribeVolumes`以進行磁碟區刪除。

在停止具有連接 Amazon EBS 磁碟區的任務之前刪除或修改角色，將導致任務卡在其中，`DEPROVISIONING`且相關聯的 Amazon EBS 磁碟區無法刪除。Amazon ECS 將定期自動重試以停止任務並刪除磁碟區，直到還原必要的許可為止。您可以使用 DescribeTasks API 檢視任務的磁碟區連接狀態和相關聯的狀態原因。

建立檔案之後，您必須授予使用者將角色傳遞給 Amazon ECS 的許可。

將基礎設施角色傳遞至 Amazon ECS 的許可

若要使用 ECS 基礎設施 IAM 角色，您必須授予使用者將角色傳遞至 Amazon ECS 的許可。將下列`iam:PassRole`許可連接至您的使用者。將 *`ecsInfrastructureRole`* 取代為您建立的基礎設施角色名稱。

```
{
    "Version": "2012-10-17",
    "Statement": [

        {
            "Action": "iam:PassRole",
            "Effect": "Allow",
            "Resource": ["arn:aws:iam::*:role/ecsInfrastructureRole"],
            "Condition": {
                "StringEquals": {"iam:PassedToService": "ecs.amazonaws.com"}
            }
        }
    ]
}
```

如需有關 `iam:Passrole` 和更新使用者許可的詳細資訊，請參閱《 AWS Identity and Access Management 使用者指南》中的授予使用者將角色傳遞至 AWS 服務的許可，以及變更 IAM 使用者的許可。

Amazon ECS CodeDeploy IAM 角色

在您搭配使用 CodeDeploy 藍/綠部署類型與 Amazon ECS 之前，CodeDeploy 服務需要許可來代您更新 Amazon ECS 服務。這些許可由 CodeDeploy IAM 角色 (`ecsCodeDeployRole`) 提供。

> **Note**
>
> 使用者也需要許可來使用 CodeDeploy；所需的 IAM 許可 中描述了這些許可。

提供兩個受管政策。如需詳細資訊，請參閱 AWS 受管政策參考指南中的下列其中一項：

- AWSCodeDeployRoleForECS - 授予 CodeDeploy 使用相關聯動作更新任何資源的許可。
- AWSCodeDeployRoleForECSLimited - 給予 CodeDeploy 更多有限許可。

建立 CodeDeploy 角色

您可以使用下列程序為 Amazon ECS 建立 CodeDeploy 角色

AWS Management Console

建立 CodeDeploy 的服務角色 (IAM 主控台)

1. 登入 AWS Management Console ，並在 https：//https://console.aws.amazon.com/iam/ 開啟 IAM 主控台。
2. 在 IAM 主控台的導覽窗格中，選擇角色，然後選擇建立角色。
3. 對於 Trusted entity type (信任的實體類型)，請選擇 AWS 服務。
4. 針對服務或使用案例，選擇 CodeDeploy，然後選擇 CodeDeploy - ECS 使用案例。
5. 選擇 Next (下一步)。
6. 在連接許可政策區段中，確定已選取 AWSCodeDeployRoleForECS 政策。
7. 選擇 Next (下一步)。
8. 針對角色名稱，輸入 ecsCodeDeployRole。
9. 檢閱角色，然後選擇 Create role (建立角色)。

AWS CLI

將*#######*取代為您自己的資訊。

1. 建立名為 的檔案codedeploy-trust-policy.json，其中包含要用於 CodeDeploy IAM 角色的信任政策。

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

2. 使用在上一個步驟中建立的信任政策，建立名為 ecsCodedeployRole 的 IAM 角色。

```
aws iam create-role \
      --role-name ecsCodedeployRole \
      --assume-role-policy-document file://codedeploy-trust-policy.json
```

3. 將 AWSCodeDeployRoleForECS或 AWSCodeDeployRoleForECSLimited受管政策連接至ecsTaskRole角色。

```
aws iam attach-role-policy \
      --role-name ecsCodedeployRole \
      --policy-arn arn:aws:iam::aws:policy/AWSCodeDeployRoleForECS
```

```
aws iam attach-role-policy \
      --role-name ecsCodedeployRole \
      --policy-arn arn:aws:iam::aws:policy/AWSCodeDeployRoleForECSLimited
```

當您服務中的任務需要任務執行角色時，您必須將每個任務執行角色或任務角色覆寫的iam:PassRole許可新增至 CodeDeploy 角色做為政策。

任務執行角色許可

當您服務中的任務需要任務執行角色時，您必須將每個任務執行角色或任務角色覆寫的`iam:PassRole`許可新增至 CodeDeploy 角色做為政策。如需詳細資訊，請參閱 Amazon ECS 任務執行 IAM 角色 和 Amazon ECS 任務 IAM 角色。然後，您將該政策連接至 CodeDeploy 角色

建立 政策

AWS Management Console

若要使用 JSON 政策編輯器來建立政策

1. 登入 AWS Management Console ，並在 https：//https://console.aws.amazon.com/iam/ 開啟 IAM 主控台。
2. 在左側的導覽窗格中，選擇 Policies (政策)。

   如果這是您第一次選擇 Policies (政策)，將會顯示 Welcome to Managed Policies (歡迎使用受管政策) 頁面。選擇 Get Started (開始使用)。
3. 在頁面頂端，選擇 Create policy (建立政策)。
4. 在政策編輯器中，選擇 JSON 選項。
5. 輸入下列 JSON 政策文件：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": ["arn:aws:iam::<aws_account_id>:role/
<ecsCodeDeployRole>"]
        }
    ]
}
```

6. 選擇 Next (下一步)。

> **Note**
>
> 您可以隨時切換視覺化與 JSON 編輯器選項。不過，如果您進行變更或在視覺化編輯器中選擇下一步，IAM 就可能會調整您的政策結構，以便針對視覺化編輯器進行最佳化。如需詳細資訊，請參閱 IAM 使用者指南中的調整政策結構。

7. 在檢視與建立頁面上，為您正在建立的政策輸入政策名稱與描述 (選用)。檢視此政策中定義的許可，來查看您的政策所授予的許可。
8. 選擇 Create policy (建立政策) 儲存您的新政策。

建立政策之後，請將政策連接至 CodeDeploy 角色。如需有關如何將政策連接至角色的資訊，請參閱AWS Identity and Access Management 《使用者指南》中的更新角色的許可。

AWS CLI

使用您自己的資訊取代*#######*。

1. 建立稱為 `blue-green-iam-passrole.json` 的檔案，其中具有以下內容。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": ["arn:aws:iam::<aws_account_id>:role/
<ecsCodeDeployRole>"]
        }
    ]
}
```

2. 使用以下命令，使用 JSON 政策文件檔案建立 IAM 政策。

```
aws iam create-policy \
      --policy-name cdTaskExecutionPolicy \
      --policy-document file://blue-green-iam-passrole.json
```

3. 擷取您使用以下命令建立的 IAM 政策 ARN。

```
aws iam list-policies --scope Local --query 'Policies[?
PolicyName==`cdTaskExecutionPolicy`].Arn'
```

4. 使用以下命令將政策連接至 CodeDeploy IAM 角色。

```
aws iam attach-role-policy \
      --role-name ecsCodedeployRole \
      --policy-arn arn:aws:iam:111122223333:aws:policy/cdTaskExecutionPolicy
```

Amazon ECS EventBridge IAM 角色

在您可以將 Amazon ECS 排程任務與 EventBridge 規則和目標搭配使用之前，EventBridge 服務需要代表您執行 Amazon ECS 任務的許可。這些許可是由 EventBridge IAM 角色 () 提供`ecsEventsRole`。

`AmazonEC2ContainerServiceEventsRole` 政策顯示如下。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": ["ecs:RunTask"],
            "Resource": ["*"]
        },
        {
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": ["*"],
            "Condition": {
                "StringLike": {"iam:PassedToService": "ecs-tasks.amazonaws.com"}
            }
        },
        {
            "Effect": "Allow",
            "Action": "ecs:TagResource",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "ecs:CreateAction": ["RunTask"]
                }
```

```
            }
        }
    ]
}
```

如果您的排程任務需要使用任務執行角色、任務角色或任務角色覆寫，則必須將每個任務執行角色、任務角色或任務角色覆寫的`iam:PassRole`許可新增至 EventBridge IAM 角色。如需任務執行角色的詳細資訊，請參閱「Amazon ECS 任務執行 IAM 角色」。

> **Note**
>
> 指定您任務執行角色或任務角色覆寫的完整 ARN。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": ["arn:aws:iam::<aws_account_id>:role/
<ecsTaskExecutionRole_or_TaskRole_name>"]
        }
    ]
}
```

設定排程任務時，您可以選擇讓 為您 AWS Management Console 建立 EventBridge 角色。如需詳細資訊，請參閱使用 Amazon EventBridge 排程器來排程 Amazon ECS 任務 。

建立 EventBridge 角色

將#######取代為您自己的資訊。

1. 建立名為 `eventbridge-trust-policy.json` 的檔案，其中包含用於 IAM 角色的信任政策。檔案應包含以下內容：

   ```
   {
     "Version": "2012-10-17",
     "Statement": [
       {
         "Sid": "",
   ```

```
      "Effect": "Allow",
      "Principal": {
        "Service": "events.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

2. 使用以下命令，ecsEventsRole使用您在上一個步驟中建立的信任政策來建立名為 的 IAM 角色。

```
aws iam create-role \
      --role-name ecsEventsRole \
      --assume-role-policy-document file://eventbridge-trust-policy.json
```

3. 使用下列命令 將 AWS 受管 AmazonEC2ContainerServiceEventsRole連接至ecsEventsRole角色。

```
aws iam attach-role-policy \
      --role-name ecsEventsRole \
      --policy-arn arn:aws:iam::aws:policy/service-role/
AmazonEC2ContainerServiceEventsRole
```

您也可以使用 IAM 主控台的自訂信任政策工作流程 (https://console.aws.amazon.com/iam/：//) 來建立角色。如需詳細資訊，請參閱《IAM 使用者指南》中的使用自訂信任政策 （ 主控台 ） 建立角色。

將政策連接至 **ecsEventsRole** 角色

您可以使用下列程序，將任務執行角色的許可新增至 EventBridge IAM 角色。

AWS Management Console

若要使用 JSON 政策編輯器來建立政策

1. 登入 AWS Management Console ，並在 https：//https://console.aws.amazon.com/iam/ 開啟 IAM 主控台。
2. 在左側的導覽窗格中，選擇 Policies (政策)。

   如果這是您第一次選擇 Policies (政策)，將會顯示 Welcome to Managed Policies (歡迎使用受管政策) 頁面。選擇 Get Started (開始使用)。

3. 在頁面頂端，選擇 Create policy (建立政策)。
4. 在政策編輯器中，選擇 JSON 選項。
5. 輸入下列 JSON 政策文件：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": ["arn:aws:iam::<aws_account_id>:role/
<ecsTaskExecutionRole_or_TaskRole_name>"]
        }
    ]
}
```

6. 選擇 Next (下一步)。

> **Note**
>
> 您可以隨時切換視覺化與 JSON 編輯器選項。不過，如果您進行變更或在視覺化編輯器中選擇下一步，IAM 就可能會調整您的政策結構，以便針對視覺化編輯器進行最佳化。如需詳細資訊，請參閱 IAM 使用者指南中的調整政策結構。

7. 在檢視與建立頁面上，為您正在建立的政策輸入政策名稱與描述 (選用)。檢視此政策中定義的許可，來查看您的政策所授予的許可。
8. 選擇 Create policy (建立政策) 儲存您的新政策。

建立政策後，請將政策連接至 EventBridge 角色。如需有關如何將政策連接至角色的資訊，請參閱AWS Identity and Access Management 《使用者指南》中的更新角色的許可。

AWS CLI

將*******取代為您自己的資訊。

1. 建立稱為 `ev-iam-passrole.json` 的檔案，其中具有以下內容。

```
{
    "Version": "2012-10-17",
    "Statement": [
```

```
        {
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": ["arn:aws:iam::<aws_account_id>:role/
<ecsTaskExecutionRole_or_TaskRole_name>"]
        }
    ]
}
```

2. 使用以下 AWS CLI 命令，使用 JSON 政策文件檔案建立 IAM 政策。

```
aws iam create-policy \
      --policy-name eventsTaskExecutionPolicy \
      --policy-document file://ev-iam-passrole.json
```

3. 擷取您使用以下命令建立的 IAM 政策 ARN。

```
aws iam list-policies --scope Local --query 'Policies[?
PolicyName==`eventsTaskExecutionPolicy`].Arn'
```

4. 使用以下命令，使用政策 ARN 將政策連接至 EventBridge IAM 角色。

```
aws iam attach-role-policy \
      --role-name ecsEventsRole \
      --policy-arn arn:aws:iam:111122223333:aws:policy/eventsTaskExecutionPolicy
```

Amazon ECS 主控台必要的許可

遵循授予最低權限的最佳實務，您可以使用 AmazonECS_FullAccess 受管政策作為範本，以建立您自己的自訂政策。如此一來，您就可以根據特定需求，從受管政策中取消或新增許可。如需詳細資訊，請參閱 AWS 受管政策參考中的 AmazonECS_FullAccess。

建立 IAM 角色的許可

下列動作需要其他許可才能完成操作：

- 註冊外部執行個體 - 如需詳細資訊，請參閱 Amazon ECS Anywhere IAM 角色
- 註冊任務定義 - 如需詳細資訊，請參閱 Amazon ECS 任務執行 IAM 角色
- 建立要用於排程任務的 EventBridge 規則 - 如需詳細資訊，請參閱 Amazon ECS EventBridge IAM 角色

您可以先在 IAM 中建立角色來新增這些許可，然後在 Amazon ECS 主控台中使用這些許可。如果您不建立角色，Amazon ECS 主控台會代表您建立角色。

將外部執行個體註冊至叢集所需的許可

當您將外部執行個體註冊至叢集並想要建立新的外部執行個體 (ecsExternalInstanceRole) 角色時，您需要其他許可。

以下是所需的其他許可：

- iam - 允許委託人建立並列出 IAM 角色及其連接的政策。
- ssm – 允許主體使用 Systems Manager 註冊外部執行個體。

> **Note**
>
> 若要選擇現有 ecsExternalInstanceRole，您必須擁有 iam:GetRole 和 iam:PassRole 許可。

下列政策包含必要的許可，並將動作限制為 ecsExternalInstanceRole 角色。

```
{
  "Statement": [
      {
          "Effect": "Allow",
            "Action": [
              "iam:AttachRolePolicy",
              "iam:CreateRole",
              "iam:CreateInstanceProfile",
              "iam:AddRoleToInstanceProfile",
              "iam:ListInstanceProfilesForRole",
              "iam:GetRole"
            ],
            "Resource": "arn:aws:iam::*:role/ecsExternalInstanceRole"
        },
        {
            "Effect": "Allow",
            "Action": ["iam:PassRole","ssm:CreateActivation"],
            "Resource": "arn:aws:iam::*:role/ecsExternalInstanceRole"
        }
```

```
    ]
}
```

註冊任務定義所需的許可

當您註冊任務定義並想要建立新的任務執行 (ecsTaskExecutionRole) 角色時，您需要其他許可。

以下是所需的其他許可：

- iam - 允許委託人建立並列出 IAM 角色及其連接的政策。

> **Note**
>
> 若要選擇現有 ecsTaskExecutionRole，您必須擁有 iam:GetRole 許可。

下列政策包含必要的許可，並將動作限制為 ecsTaskExecutionRole 角色。

```
{
  "Statement": [
      {
          "Effect": "Allow",
            "Action": [
              "iam:AttachRolePolicy",
              "iam:CreateRole",
              "iam:GetRole"
            ],
            "Resource": "arn:aws:iam::*:role/ecsTaskExecutionRole"
        }
    ]
}
```

為排程任務建立 EventBridge 規則所需的許可

當您排程任務且想要建立新的 CloudWatch Events (ecsEventsRole) 角色時，您需要其他許可。

以下是所需的其他許可：

- iam - 允許委託人建立並列出 IAM 角色及其附加政策，並允許 Amazon ECS 將角色傳遞給其他服務以擔任該角色。

> **Note**
>
> 若要選擇現有 ecsEventsRole，您必須擁有 iam:GetRole 和 iam:PassRole 許可。

下列政策包含必要的許可，並將動作限制為 ecsEventsRole 角色。

```
{
  "Statement": [
      {
          "Effect": "Allow",
            "Action": [
              "iam:AttachRolePolicy",
              "iam:CreateRole",
              "iam:GetRole",
              "iam: PassRole"
            ],
            "Resource": "arn:aws:iam::*:role/ecsEventsRole"
        }
    ]
}
```

檢視服務部署所需的許可

當您遵循授予最低權限的最佳實務時，您需要新增其他許可，才能在 主控台中檢視服務部署。

您需要存取下列動作：

- ListServiceDeployments
- DescribeServiceDeployments
- DescribeServiceRevisions

您需要存取下列資源：

- 服務
- 服務部署
- 服務修訂

下列範例政策包含必要的許可，並將動作限制為指定的服務。

將 account、 cluster-name和 取代service-name為您的值。

```
{
  "Statement": [
      {
          "Effect": "Allow",
            "Action": [
              "ecs:ListServiceDeployments",
              "ecs:DescribeServiceDeployments",
              "ecs:DescribeServiceRevisions"
            ],
            "Resource": [
             "arn:aws:ecs:us-east-1:123456789012:service/cluster-name/service-name",
             "arn:aws:ecs:us-east-1:123456789012:service-deployment/cluster-name/
service-name/*",
             "arn:aws:ecs:us-east-1:123456789012:service-revision/cluster-name/service-
name/*"
            ]
        }
    ]
}
```

使用 的 Amazon ECS 主控台所需的許可 AWS CloudFormation

Amazon ECS 主控台由 提供動力 AWS CloudFormation ，在下列情況下需要額外的 IAM 許可：

- 建立叢集
- 建立服務
- 建立容量提供者

您可以為其他許可建立政策，然後將其連接至用於存取主控台的 IAM 角色。如需詳細資訊，請參閱「IAM 使用者指南」中的建立 IAM 政策。

建立叢集所需的許可

當您在 主控台中建立叢集時，您需要其他許可來授予您管理 AWS CloudFormation 堆疊的許可。

以下是所需的其他許可：

- `cloudformation` - 允許委託人建立和管理 AWS CloudFormation 堆疊。在使用 AWS Management Console 建立 Amazon ECS 叢集以及後續管理這些叢集時，這為必要項。

- ssm – 允許 AWS CloudFormation 參考最新的 Amazon ECS 最佳化 AMI。使用 建立 Amazon ECS 叢集時，這是必要的 AWS Management Console。

下列政策包含必要的 AWS CloudFormation 許可，並將動作限制為在 Amazon ECS 主控台中建立的資源。

```
{
  "Statement": [
      {
          "Effect": "Allow",
          "Action": [
                "cloudformation:CreateStack",
                "cloudformation:DeleteStack",
                "cloudformation:DescribeStack*",
                "cloudformation:UpdateStack"
             ],
            "Resource": [
                "arn:*:cloudformation:*:*:stack/Infra-ECS-Cluster-*"
            ]
      },
      {
          "Effect": "Allow",
          "Action": "ssm:GetParameters",
          "Resource": [
            "arn:aws:ssm:*:*:parameter/aws/service/ecs/optimized-ami/amazon-linux-2*/
*",
            "arn:aws:ssm:*:*:parameter/aws/service/ecs/optimized-ami/amazon-
linux-2023*/*"
          ]
      }
   ]
}
```

如果您尚未建立 Amazon ECS 容器執行個體角色 (ecsInstanceRole)，並且正建立使用 Amazon EC2 執行個體的叢集，則主控台將代表您建立角色。

此外，如果您使用 Auto Scaling 群組，則需要額外的許可，以便主控台在使用叢集自動擴展功能時，可以將標籤新增至自動擴展群組。

以下是所需的其他許可：

- `autoscaling` – 允許主控台標記 Amazon EC2 Auto Scaling 群組。使用叢集自動擴展功能管理 Amazon EC2 Auto Scaling 群組時，這為必要項。標籤是 ECS 受管標籤，主控台會自動新增至群組，表示已在主控台中建立。
- `iam` - 允許委託人列出 IAM 角色及其連接的政策。委託人還可列出 Amazon EC2 執行個體可用的執行個體設定檔。

以下政策包含必要的 IAM 許可，並將動作限制為 `ecsInstanceRole` 角色。

Auto Scaling 許可不受限制。

```
{
  "Statement": [
      {
          "Effect": "Allow",
            "Action": [
              "iam:AttachRolePolicy",
              "iam:CreateRole",
              "iam:CreateInstanceProfile",
              "iam:AddRoleToInstanceProfile",
              "iam:ListInstanceProfilesForRole",
              "iam:GetRole"
            ],
            "Resource": "arn:aws:iam::*:role/ecsInstanceRole"
        },
        {
            "Effect": "Allow",
            "Action": "autoscaling:CreateOrUpdateTags",
            "Resource": "*"
        }
    ]
}
```

建立 服務所需的許可

當您在 主控台中建立服務時，您需要其他許可，以授予您管理 AWS CloudFormation 堆疊的許可。以下是所需的其他許可：

- `cloudformation` - 允許委託人建立和管理 AWS CloudFormation 堆疊。在使用 AWS Management Console 建立 Amazon ECS 服務以及後續管理這些服務時，這為必要項。

下列政策包含必要的許可，並將動作限制為在 Amazon ECS 主控台中建立的資源。

```
{
  "Statement": [
      {
          "Effect": "Allow",
          "Action": [
                "cloudformation:CreateStack",
                "cloudformation:DeleteStack",
                "cloudformation:DescribeStack*",
                "cloudformation:UpdateStack"
             ],
            "Resource": [
                "arn:*:cloudformation:*:*:stack/ECS-Console-V2-Service-*"
            ]
      }
   ]
}
```

Amazon ECS 服務自動擴展所需的 IAM 許可

透過結合 Amazon ECS、CloudWatch 和 Application Auto Scaling API 可實現服務自動擴展。服務是使用 Amazon ECS 建立與更新的，警示是使用 CloudWatch 建立的，而擴展政策是使用 Application Auto Scaling 建立的。

除了建立和更新服務的標準 IAM 許可之外，還需要下列許可才能與 Service Auto Scaling 設定互動，如下列範例政策所示。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "application-autoscaling:*",
                "ecs:DescribeServices",
                "ecs:UpdateService",
                "cloudwatch:DescribeAlarms",
                "cloudwatch:PutMetricAlarm",
                "cloudwatch:DeleteAlarms",
                "cloudwatch:DescribeAlarmHistory",
                "cloudwatch:DescribeAlarmsForMetric",
```

```
                "cloudwatch:GetMetricStatistics",
                "cloudwatch:ListMetrics",
                "cloudwatch:DisableAlarmActions",
                "cloudwatch:EnableAlarmActions",
                "iam:CreateServiceLinkedRole",
                "sns:CreateTopic",
                "sns:Subscribe",
                "sns:Get*",
                "sns:List*"
            ],
            "Resource": ["*"]
        }
    ]
}
```

建立 Amazon ECS 服務範例 和 更新 Amazon ECS 服務範例 IAM 政策範例示範在 AWS Management Console中使用 Service Auto Scaling 所需的許可。

Application Auto Scaling 服務也需要描述 Amazon ECS 服務和 CloudWatch 警示的許可，以及代您修改服務所需計數的許可。此 `sns:` 權限許可用於 CloudWatch 在超出閾值時向 Amazon SNS 主題傳送的通知。如果您為 Amazon ECS 服務使用自動擴展，則會建立其名為 `AWSServiceRoleForApplicationAutoScaling_ECSService` 的服務連結角色。此服務連結角色會授與 Application Auto Scaling 許可以描述政策的警示、監控服務目前執行的任務計數，以及修改服務需要的計數。Application Auto Scaling 的原始受管 Amazon ECS 角色為 `ecsAutoscaleRole`，但已不再需要。服務連結的角色預設為 Application Auto Scaling。如需詳細資訊，請參閱「Application Auto Scaling 使用者指南」中的適用於 Application Auto Scaling 的服務連結角色。

如果您已在 Amazon ECS 可使用 CloudWatch 指標之前建立您的 Amazon ECS 容器執行個體角色，則您可能需要新增 `ecs:StartTelemetrySession` 許可。如需詳細資訊，請參閱考量事項。

在建立時授予標記資源的許可

以下的標籤建立 Amazon ECS API 動作允許您在建立資源時指定標籤。如果在資源建立動作中指定標籤， AWS 會執行其他授權，以確認指派正確的許可來建立標籤。

- `CreateCapacityProvider`
- `CreateCluster`
- `CreateService`
- `CreateTaskSet`
- `RegisterContainerInstance`

- RegisterTaskDefinition
- RunTask
- StartTask

您可以使用資源標籤來實作以屬性為基礎的控制 (ABAC)。如需詳細資訊，請參閱 the section called "使用資源標籤控制對 Amazon ECS 資源的存取" 和 標記 資源。

若要允許在建立時標記，請建立或修改政策，以同時包含使用建立資源之動作的許可，例如 ecs:CreateCluster 或 ecs:RunTask 和 ecs:TagResource 動作。

下列範例示範 政策，允許使用者在叢集建立期間建立叢集和新增標籤。使用者沒有標記現有資源的權限 (他們不能直接呼叫 ecs:TagResource 動作)。

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
         "ecs:CreateCluster"
      ],
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": [
         "ecs:TagResource"
      ],
      "Resource": "*",
      "Condition": {
         "StringEquals": {
                  "ecs:CreateAction": [
                      "CreateCluster",
                      "CreateCapacityProvider",
                      "CreateService",
                      "CreateTaskSet",
                      "RegisterContainerInstance",
                      "RegisterTaskDefinition",
                      "RunTask",
                      "StartTask"
                  ]
          }
```

```
        }
      }
    ]
}
```

只有在資源建立動作中套用了標籤時，才評估 ecs:TagResource 動作。因此，在沒有標記條件的情況下，若請求中未指定標籤，則具備資源建立許可的使用者不需要使用 ecs:TagResource 動作的許可。然而，若該使用者試圖建立具有標籤的資源卻未具備使用 ecs:TagResource 動作的許可，則該請求會失敗。

Amazon ECS 控制對特定標籤的存取

您可以在 IAM 政策的 Condition 元素中使用其他條件，來控制可套用至資源的標籤金鑰索和值。

下列條件金鑰機可與前一節中的範例搭配使用：

- aws:RequestTag：表示請求中必須存在特定標籤金鑰或標籤金鑰與值。請求內亦可指定其他標籤。
  - 搭配 StringEquals 條件運算子使用，以強制結合特定標籤金鑰與值，例如強制執行標籤 cost-center=cc123：

    ```
    "StringEquals": { "aws:RequestTag/cost-center": "cc123" }
    ```

  - 搭配 StringLike 條件運算子使用，以在請求中強制執行特定標籤金鑰，例如強制執行標籤金鑰 purpose：

    ```
    "StringLike": { "aws:RequestTag/purpose": "*" }
    ```

- aws:TagKeys：強制執行請求中使用的標籤金鑰。
  - 搭配 ForAllValues 修飾詞使用，若請求內提供特定標籤金鑰，將強制加以執行 (若請求內指定標籤，則僅允許特定標籤金鑰，不允許其他標籤)。例如，允許標籤金鑰 environment 或 cost-center：

    ```
    "ForAllValues:StringEquals": { "aws:TagKeys": ["environment","cost-center"] }
    ```

  - 搭配 ForAnyValue 修飾詞使用，以強制要求請求內至少具有一個指定的標籤金鑰。例如，請求內必須出現至少下列標籤金鑰 environment 或 webserver 之一：

    ```
    "ForAnyValue:StringEquals": { "aws:TagKeys": ["environment","webserver"] }
    ```

這些條件金鑰可套用於支援標記的資源建立動作，以及 ecs:TagResource 動作。若要了解 Amazon ECS API 動作是否支援標記，請參閱 Amazon ECS 的動作、資源和條件金鑰。

若要強制使用者在建立資源時指定標籤，您必須在資源建立動作內，搭配 aws:RequestTag 修飾詞使用 aws:TagKeys 條件金鑰或 ForAnyValue 條件金鑰。若使用者未針對資源建立動作指定標籤，則不會評估 ecs:TagResource 動作。

以條件而言，條件金鑰不區分大小寫，而條件值會區分大小寫。因此，欲強制標籤鍵區分大小寫，請使用 aws:TagKeys 條件索引鍵，其中標籤鍵指定為條件值。

如需多值條件的詳細資訊，請參閱《IAM 使用者指南》中的具有多個內容索引鍵或值的條件。

使用資源標籤控制對 Amazon ECS 資源的存取

當您建立授予使用者使用 Amazon ECS 資源之許可的 IAM 政策時，您可以在政策的 Condition 元素中包含標籤資訊，以根據標籤控制存取。這稱為以屬性型存取控制 (ABAC)。ABAC 對於使用者可以修改、使用或刪除哪些資源提供了更佳的控制。如需詳細資訊，請參閱 AWS的 ABAC 是什麼？

例如：您可以建立一個政策，允許使用者刪除叢集，但如果叢集具有標籤 environment=production，則拒絕該動作。若要這樣做，您可以使用 aws:ResourceTag 條件金鑰，根據連接至資源的標籤允許或拒絕存取資源。

```
"StringEquals": { "aws:ResourceTag/environment": "production" }
```

若要了解 Amazon ECS API 動作是否支援使用 aws:ResourceTag 條件金鑰控制存取，請參閱 Amazon ECS 的動作、資源和條件金鑰。由於 Describe 動作不支援資源層級許可，您必須在不同的陳述式中指定它們，無需條件。

如需 IAM 政策的範例，請參閱Amazon ECS 範例政策 。

如果您允許或拒絕使用者根據標籤存取資源，請務必考慮明確拒絕使用者將這些標籤新增至相同資源或從中移除的能力。否則，使用者可能透過修改標籤來避開您的限制，並取得資源的存取。

Amazon ECS 範例政策

您可以使用 IAM 政策來授予使用者許可，以檢視和使用 Amazon ECS 主控台中的特定資源。您可以使用上一節中的範例政策；不過，這些政策是專為使用 AWS CLI 或 AWS SDK 提出的請求而設計。

範例：允許使用者根據標籤刪除 Amazon ECS 叢集

當標籤具有「目的/測試」的金鑰/值對時，下列政策允許使用者刪除叢集。

```
{
  "Version": "2012-10-17",
  "Statement": [
       {
      "Action": [
        "ecs:DeleteCluster"
      ],
      "Effect": "Allow",
      "Resource": "arn:aws:ecs:region:account-id:cluster/*",
      "Condition": {
        "StringEquals": {
          "aws:ResourceTag/Purpose": "Testing"
        }
      }
    }
  ]
}
```

對 Amazon Elastic Container Service 身分和存取進行故障診斷

請使用以下資訊來協助您診斷和修正使用 Amazon ECS 和 IAM 時可能遇到的常見問題。

主題

- 我未獲授權，不得在 Amazon ECS 中執行動作
- 我未獲得執行 iam:PassRole 的授權
- 我想要允許 以外的人員 AWS 帳戶 存取我的 Amazon ECS 資源
- 其他疑難排解資源

我未獲授權，不得在 Amazon ECS 中執行動作

如果您收到錯誤，告知您未獲授權執行動作，您的政策必須更新，允許您執行動作。

下列範例錯誤會在mateojackson IAM 使用者嘗試使用主控台檢視一個虛構 *my-example-widget* 資源的詳細資訊，但卻無虛構 ecs:*GetWidget* 許可時發生。

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 ecs:GetWidget on resource: my-example-widget
```

在此情況下，必須更新 mateojackson 使用者的政策，允許使用 ecs:*GetWidget* 動作存取 *my-example-widget* 資源。

如果您需要協助，請聯絡您的 AWS 管理員。您的管理員提供您的簽署憑證。

我未獲得執行 iam:PassRole 的授權

如果您收到錯誤，告知您無權執行 iam:PassRole 動作，則必須更新您的政策，以允許您將角色傳遞至 Amazon ECS。

有些 AWS 服務 可讓您將現有角色傳遞給該服務，而不是建立新的服務角色或服務連結角色。如需執行此作業，您必須擁有將角色傳遞至該服務的許可。

當名為 marymajor 的 IAM 使用者嘗試使用主控台在 Amazon ECS 中執行動作時，發生下列範例錯誤。但是，動作請求服務具備服務角色授予的許可。Mary 沒有將角色傳遞至該服務的許可。

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

在這種情況下，Mary 的政策必須更新，允許她執行 iam:PassRole 動作。

如果您需要協助，請聯絡您的 AWS 管理員。您的管理員提供您的簽署憑證。

我想要允許 以外的人員 AWS 帳戶 存取我的 Amazon ECS 資源

您可以建立一個角色，讓其他帳戶中的使用者或您組織外部的人員存取您的資源。您可以指定要允許哪些信任物件取得該角色。針對支援基於資源的政策或存取控制清單 (ACL) 的服務，您可以使用那些政策來授予人員存取您的資源的許可。

如需進一步了解，請參閱以下內容：

- 若要了解 Amazon ECS 是否支援這些功能，請參閱 Amazon Elastic Container Service 如何與 IAM 搭配使用。
- 若要了解如何 AWS 帳戶 在您擁有的 資源間提供存取權，請參閱《IAM 使用者指南》中的在您擁有 AWS 帳戶 的另一個 IAM 使用者中提供存取權。
- 若要了解如何將資源的存取權提供給第三方 AWS 帳戶，請參閱《IAM 使用者指南》中的提供存取權給第三方 AWS 帳戶 擁有。
- 如需了解如何透過聯合身分提供存取權，請參閱 IAM 使用者指南中的將存取權提供給在外部進行身分驗證的使用者 (聯合身分)。

- 如需了解使用角色和資源型政策進行跨帳戶存取之間的差異，請參閱《IAM 使用者指南》中的 IAM 中的跨帳戶資源存取。

其他疑難排解資源

以下頁面提供有關錯誤代碼的資訊：

- Amazon ECS 已停止任務錯誤訊息
- 檢視 Amazon ECS 服務事件訊息

Amazon ECS 的 IAM 最佳實務

您可以使用 AWS Identity and Access Management (IAM) 透過規則型政策來管理和控制對 AWS 服務和資源的存取，以用於身分驗證和授權。更具體地說，透過此服務，您可以使用套用至使用者、群組或角色的政策來控制對 AWS 資源的存取。在這三個使用者中，使用者是可以對您的資源進行存取的帳戶。而且，IAM 角色是一組可以由經驗證身份擔任操作的許可，該身份與 IAM 以外的特定身份沒有關聯。如需詳細資訊，請參閱存取管理的 Amazon ECS 概觀：許可和政策。

遵循最低權限存取的政策

建立範圍為允許使用者執行其指定工作的政策。例如，如果開發人員需要定期停止任務，請建立僅允許該特定動作的政策。下列範例僅允許使用者停止屬於具特定 Amazon Resource Name (ARN) 叢集上特定 `task_family` 的任務。條件中參考 ARN 也是使用資源層級許可的範例。您可以使用資源層級許可來指定要套用動作的資源。

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

```
      ]
    }
  ]
}
```

讓叢集資源做為管理界限

範圍太窄的政策可能會造成角色擴散，並增加系統管理額外負荷。不要建立範圍僅限於特定任務或服務的角色，而是建立範圍為叢集的角色，並使用叢集做為主要管理界限。

建立自動化管道以隔離最終使用者與 API

您可以透過建立自動封裝應用程式並將應用程式部署到 Amazon ECS 叢集的管道來限制使用者可以使用的動作。這可以有效地將建立、更新和刪除任務的工作委派給管道。如需詳細資訊，請參閱《AWS CodePipeline 使用者指南》中的教學課程：具 CodePipeline 的 Amazon ECS 標準部署。

針對新增的安全層使用政策條件

當您需要新增的安全層時，請在政策中新增條件。如果您正在執行特權操作或需要限制可針對特定資源執行的一組動作時，這種做法非常有用。刪除叢集時，下列範例政策需要多重要素驗證。

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

套用至服務的標籤會傳播至屬於該服務一部分的所有任務。因此，您可以使用特定標籤建立範圍為 Amazon ECS 資源的角色。在下列政策中，IAM 主體會啟動和停止所有具有 標籤鍵`Department`和 標籤值的任務`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"}
            }
        }
    ]
}
```

定期稽核 APIs的存取

使用者可能會變更角色。變更角色之後，先前授予他們的許可可能不再適用。確保您稽核有權存取 Amazon ECS API 的人員，以及該存取權是否仍然被授權。考慮將 IAM 與使用者生命週期管理解決方案整合，該解決方案會在使用者離開組織時自動撤銷存取。如需詳細資訊，請參閱《 AWS Identity and Access Management 使用者指南》中的AWS 安全稽核指導方針。

在 Amazon Elastic Container Service 中記錄和監控

監控是維護 Amazon Elastic Container Service 和您的 AWS 解決方案可靠性、可用性和效能的重要部分。您應該從解決方案的所有部分 AWS 收集監控資料，以便在發生多點故障時更輕鬆地偵錯。 AWS 提供數種工具來監控 Amazon ECS 資源並回應潛在事件：

Amazon CloudWatch 警示

監看指定時段內的單一指標，並根據與多個時段內給定之閾值相對的指標值來執行一或多個動作。此動作是傳送到 Amazon Simple Notification Service (Amazon SNS) 主題或 Amazon EC2 Auto Scaling 政策的通知。CloudWatch 警示不會只因處於特定狀態就叫用動作，狀態必須已變更並已維持一段指定的時間。如需詳細資訊，請參閱使用 CloudWatch 監控 Amazon ECS。

如果服務的任務是使用 Fargate 啟動類型，您可以使用 CloudWatch 警示，以根據 CloudWatch 指標 (例如 CPU 和記憶體使用率) 來擴展和縮減服務中的任務。如需詳細資訊，請參閱自動擴展 Amazon ECS 服務。

如果叢集的任務或服務是使用 EC2 啟動類型，您可以使用 CloudWatch 警示，以根據 CloudWatch 指標 (例如叢集記憶體保留) 來縮減和擴展容器執行個體。

Amazon CloudWatch Logs

透過在任務定義中指定 `awslogs` 日誌驅動程式，可監控、存放及存取 Amazon ECS 任務中容器的日誌檔案。如需詳細資訊，請參閱使用 awslogs 驅動程式。

您也可以從 Amazon ECS 容器執行個體監控、存放及存取作業系統和 Amazon ECS 容器代理程式日誌檔案。這種存取日誌的方法可用於使用 EC2 啟動類型的容器。

Amazon CloudWatch Events

比對事件並將其路由至一或多個目標函數或串流，以進行變更、擷取狀態資訊，以及採取修正動作。如需詳細資訊，請參閱本指南使用 EventBridge 自動化對 Amazon ECS 錯誤的回應中的 和 EventBridge 是 Amazon EventBridge 使用者指南中的 Amazon CloudWatch Events 的演變。
EventBridge

AWS CloudTrail 日誌

CloudTrail 提供 Amazon ECS AWS 中使用者、角色或服務所採取動作的記錄。您可以利用 CloudTrail 所收集的資訊來判斷向 Amazon ECS 發出的請求，以及發出請求的 IP 地址、人員、時間和其他詳細資訊。如需詳細資訊，請參閱使用 記錄 Amazon ECS API 呼叫 AWS CloudTrail。

AWS Trusted Advisor

Trusted Advisor 利用從服務數十萬客戶中學到的 AWS 最佳實務。 Trusted Advisor 會檢查您的 AWS 環境，然後在有機會節省成本、改善系統可用性和效能，或協助解決安全漏洞時提出建議。所有 AWS 客戶都可以存取五個 Trusted Advisor 檢查。擁有商業或企業支援計劃的客戶可以檢視所有 Trusted Advisor 檢查。

如需詳細資訊，請參閱「支援 使用者指南」中的 AWS Trusted Advisor。

AWS Compute Optimizer

AWS Compute Optimizer 是一種 服務，可分析 AWS 資源的組態和使用率指標。這會報告您的資源是否已為最佳化，並產生最佳化建議，以降低成本並改善工作負載的效能。

如需詳細資訊，請參閱AWS Compute Optimizer Amazon ECS 的建議。

監控 Amazon ECS 的另一個重要部分是手動監控 CloudWatch 警示未涵蓋的項目。CloudWatch Trusted Advisor和其他 AWS 主控台儀表板提供 AWS 環境狀態的at-a-glance。我們建議您也檢查您容器執行個體上的日誌檔，以及您任務中的容器。

Amazon Elastic Container Service 的合規驗證

若要了解 是否 AWS 服務 在特定合規計劃的範圍內，請參閱AWS 服務 合規計劃範圍內然後選擇您感興趣的合規計劃。如需一般資訊，請參閱 AWS Compliance Programs。

您可以使用 下載第三方稽核報告 AWS Artifact。如需詳細資訊，請參閱在 中下載報告 AWS Artifact。

您在使用 時的合規責任 AWS 服務 取決於資料的敏感度、您公司的合規目標，以及適用的法律和法規。 AWS 提供下列資源以協助合規：

- 安全合規與治理 - 這些解決方案實作指南內容討論了架構考量，並提供部署安全與合規功能的步驟。
- Amazon Web Services 上的 HIPAA 安全與合規架構 – 本白皮書說明公司如何使用 AWS 來建立符合 HIPAA 資格的應用程式。

> **Note**
>
> 並非所有 AWS 服務 都符合 HIPAA 資格。如需詳細資訊，請參閱 HIPAA 資格服務參照。

- AWS 合規資源 – 此工作手冊和指南的集合可能適用於您的產業和位置。
- AWS 客戶合規指南 – 透過合規的角度了解共同的責任模型。本指南摘要說明保護 的最佳實務，AWS 服務 並將指南映射到跨多個架構的安全控制 （ 包括國家標準和技術研究所 (NIST)、支付卡產業安全標準委員會 (PCI) 和國際標準化組織 (ISO))。
- AWS Config 開發人員指南中的使用規則評估資源 – AWS Config 服務會評估資源組態符合內部實務、產業準則和法規的程度。
- AWS Security Hub – 這 AWS 服務 可讓您全面檢視其中的安全狀態 AWS。Security Hub 使用安全控制，可評估您的 AWS 資源並檢查您的法規遵循是否符合安全業界標準和最佳實務。如需支援的服務和控制清單，請參閱「Security Hub 控制參考」。
- Amazon GuardDuty – 這會監控您的環境是否有可疑和惡意活動，以 AWS 服務 偵測對您 AWS 帳戶、工作負載、容器和資料的潛在威脅。GuardDuty 可滿足特定合規架構所規定的入侵偵測需求，以協助您因應 PCI DSS 等各種不同的合規需求。
- AWS Audit Manager – 這 AWS 服務 可協助您持續稽核 AWS 用量，以簡化您管理風險的方式，以及符合法規和產業標準的方式。

Amazon ECS 的合規和安全性最佳實務

您使用 Amazon ECS 時的合規責任取決於資料的敏感度、您公司的合規目標及適用的法律和法規。

AWS 提供下列資源以協助合規：

- 安全與合規快速入門指南：這些部署指南討論架構考量，並提供在其中部署安全和以合規為中心之基準環境的步驟 AWS。
- HIPAA 安全與合規架構白皮書：此白皮書說明公司如何使用 AWS 來建立符合 HIPAA 規範的應用程式。
- 合規計劃範圍內的AWS 服務：此清單包含特定合規計劃範圍內的 AWS 服務。如需詳細資訊，請參閱 AWS 合規計劃。

支付卡產業資料安全標準 (PCI DSS)

在遵守 PCI DSS 時，請務必了解環境中持卡人資料 (CHD) 的完整流程。CHD 流程決定 PCI DSS 的適用性、定義持卡人資料環境 (CDE) 的界限和元件，以及 PCI DSS 評估的範圍。精確判斷 PCI DSS 範圍是定義安全狀態以及最終成功評估的關鍵。客戶必須具備範圍判定程序，以確保其完整性並偵測範圍的變更或偏離。

容器化應用程式的暫時性本質可在稽核組態時提供其他的複雜性。因此，客戶需要保持所有容器組態參數的感知，以確保在容器生命週期的所有階段都能滿足合規要求。

如需有關 Amazon ECS 上實現 PCI DSS 合規的其他資訊，請參閱下列白皮書。

- Amazon ECS 上 PCI DSS 合規性的架構
- AWS上 PCI DSS 範圍設定和分段的架構

HIPAA (美國健康保險流通與責任法案)

將 Amazon ECS 與處理受保護醫療資訊 (PHI) 的工作負載搭配使用，無需額外設定。Amazon ECS 充當協同運作服務，可協調在 Amazon EC2 上啟動容器的情況。其不會與正協同運作的工作負載中的資料一起運作，或對資料進行操作。符合 HIPAA 法規和 AWS 商業夥伴增補合約，在使用 Amazon ECS 啟動的容器存取時，PHI 應該在傳輸過程中和靜態加密。

每個 AWS 儲存選項都有各種加密靜態機制，例如 Amazon S3、Amazon EBS 和 AWS KMS。您可以部署覆蓋網路 (例如 VMS3 或 Weave Net)，以確保在容器之間傳輸的 PHI 完全加密，或提供備援加密層。也應啟用完整的記錄功能，並應將所有容器日誌導向至 Amazon CloudWatch。若要使用基礎設施安全的最佳實務設計您的 AWS 環境，請參閱 Security Pillar AWS Well-Architected Framework 中的基礎設施保護。

AWS Security Hub

使用 AWS Security Hub 來監控 Amazon ECS 的使用，因為它與安全最佳實務相關。Security Hub 會使用控制來評估資源組態和安全標準，協助您遵守各種合規架構。如需有關使用 Security Hub 評估 Amazon ECS 資源的詳細資訊，請參閱《AWS Security Hub 使用者指南》中的 Amazon ECS 控制項。

Amazon GuardDuty 搭配 Amazon ECS 執行期監控

Amazon GuardDuty 是一種威脅偵測服務，可協助保護您的帳戶、容器、工作負載和 AWS 環境中的資料。GuardDuty 使用機器學習 (ML) 模型，以及異常和威脅偵測功能，持續監控不同的日誌來源和執行時間活動，以識別環境中的潛在安全風險和惡意活動，並排定其優先順序。

在 GuardDuty 中使用執行期監控來識別惡意或未經授權的行為。執行期監控透過持續監控 AWS 日誌和聯網活動，以識別惡意或未經授權的行為，來保護 Fargate 和 EC2 上執行的工作負載。執行期監控使用輕量且全受管的 GuardDuty 安全代理程式，可分析主機上行為，例如檔案存取、程序執行和網路連線。這涵蓋的問題包括權限升級、使用公開的登入資料，或與惡意 IP 地址、網域的通訊，以及 Amazon EC2 執行個體和容器工作負載上存在惡意軟體。如需詳細資訊，請參閱《GuardDuty 使用者指南》中的 GuardDuty 執行期監控。 GuardDuty

合規建議

您應該儘早讓合規計劃擁有者參與您的企業，並使用 AWS 共同的責任模型來識別合規控制擁有權，以便在相關合規計劃中取得成功。

AWS Fargate 聯邦資訊處理標準 (FIPS-140)

美國聯邦資訊處理標準 (FIPS)。FIPS-140 是美國和加拿大政府標準，指定密碼模組的安全要求以保護敏感資訊。FIPS-140 定義了一組經驗證的密碼函數，可用於加密傳輸中的資料和靜態資料。

當您啟用 FIPS-140 合規時，您可以在 Fargate 上以符合 FIPS-140 的方式執行工作負載。如需 FIPS-140 合規的詳細資訊，請參閱聯邦資訊處理標準 (FIPS) 140-2。

AWS Fargate FIPS-140 考量事項

Fargate 上使用 FIPS-140 合規時，請考量下列事項：

- FIPS-140 合規僅適用於 AWS GovCloud (US) 區域。
- FIPS-140 合規預設為關閉。您必須將其開啟。
- 您的任務必須使用下列組態才能符合 FIPS-140 規範：

- operatingSystemFamily 必須是 LINUX。
- cpuArchitecture 必須是 X86_64。
- Fargate 平台版本必須是 1.4.0 以上的版本。

在 Fargate 上使用 FIPS

使用下列程序以在 Fargate 上使用 FIPS-140 合規。

1. 開啟 FIPS-140 合規的功能。如需詳細資訊，請參閱the section called “AWS Fargate 聯邦資訊處理標準 (FIPS-140) 合規”。
2. 您可以選擇性使用 ECS Exec 執行下列命令，以驗證叢集的 FIPS-140 合規狀態。

   使用您叢集的名稱取代 *my-cluster*。

   傳回值 "1" 表示您正在使用 FIPS。

   ```
   aws ecs execute-command --cluster cluster-name \
       --interactive \
       --command "cat /proc/sys/crypto/fips_enabled"
   ```

使用 CloudTrail 進行 Fargate FIPS-140 稽核

當您建立 AWS 帳戶時，會在您的帳戶中開啟 CloudTrail。當 API 和主控台活動在 Amazon ECS 中發生時，該活動會記錄在 CloudTrail 事件中，以及事件歷史記錄中的其他服務 AWS 事件。您可以在 AWS 帳戶中檢視、搜尋和下載最近的事件。如需詳細資訊，請參閱《使用 CloudTrail 事件歷史記錄檢視事件》https://docs.aws.amazon.com/awscloudtrail/latest/userguide/view-cloudtrail-events.html。

若要持續記錄您 AWS 帳戶中的事件，包括 Amazon ECS 的事件，請建立 CloudTrail 用來將日誌檔案交付至 Amazon S3 儲存貯體的追蹤。根據預設，當您在主控台建立權杖時，權杖會套用到所有區域。追蹤會記錄 AWS 分割區中所有 區域的事件，並將日誌檔案交付至您指定的 Amazon S3 儲存貯體。此外，您可以設定其他 AWS 服務，以進一步分析 CloudTrail 日誌中收集的事件資料並對其採取行動。如需詳細資訊，請參閱the section called “使用 記錄 Amazon ECS API 呼叫 AWS CloudTrail”。

以下範例顯示的是示範 PutAccountSettingDefault API 動作的 CloudTrail 日誌項目：

```
{
    "eventVersion": "1.08",
    "userIdentity": {
```

```
        "type": "IAMUser",
        "principalId": "AIDAIV5AJI5LXF5EXAMPLE",
         "arn": "arn:aws:iam::123456789012:user/jdoe",
        "accountId": "123456789012",
        "accessKeyId": "AKIAIPWIOFC3EXAMPLE",
    },
    "eventTime": "2023-03-01T21:45:18Z",
    "eventSource": "ecs.amazonaws.com",
    "eventName": "PutAccountSettingDefault",
    "awsRegion": "us-gov-east-1",
    "sourceIPAddress": "52.94.133.131",
    "userAgent": "aws-cli/2.9.8 Python/3.9.11 Windows/10 exe/AMD64 prompt/off command/
ecs.put-account-setting",
    "requestParameters": {
        "name": "fargateFIPSMode",
        "value": "enabled"
    },
    "responseElements": {
        "setting": {
            "name": "fargateFIPSMode",
            "value": "enabled",
            "principalArn": "arn:aws:iam::123456789012:user/jdoe"
        }
    },
    "requestID": "acdc731e-e506-447c-965d-f5f75EXAMPLE",
    "eventID": "6afced68-75cd-4d44-8076-0beEXAMPLE",
    "readOnly": false,
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "123456789012",
    "eventCategory": "Management",
    "tlsDetails": {
        "tlsVersion": "TLSv1.2",
        "cipherSuite": "ECDHE-RSA-AES128-GCM-SHA256",
        "clientProvidedHostHeader": "ecs-fips.us-gov-east-1.amazonaws.com"
    }
}
```

Amazon Elastic Container Service 的基礎設施安全性

Amazon Elastic Container Service 是受管服務，受到 AWS 全球網路安全的保護。如需 AWS 安全服務以及如何 AWS 保護基礎設施的相關資訊，請參閱AWS 雲端安全。若要使用基礎設施安全的最佳實務設計您的 AWS 環境，請參閱 Security Pillar AWS Well-Architected Framework 中的基礎設施保護。

您可以使用 AWS 已發佈的 API 呼叫，透過網路存取 Amazon ECS。使用者端必須支援下列專案：

- Transport Layer Security (TLS)。我們需要 TLS 1.2 並建議使用 TLS 1.3。
- 具備完美轉送私密(PFS)的密碼套件，例如 DHE (Ephemeral Diffie-Hellman)或 ECDHE (Elliptic Curve Ephemeral Diffie-Hellman)。現代系統(如 Java 7 和更新版本)大多會支援這些模式。

此外，請求必須使用存取金鑰 ID 和與 IAM 主體相關聯的私密存取金鑰來簽署。或者，您可以透過 AWS Security Token Service (AWS STS) 來產生暫時安全憑證來簽署請求。

您可以從任何網路位置呼叫這些 API 操作。Amazon ECS 支援資源型存取政策，這類政策可根據來源 IP 地址納入限制，因此請確認該政策已將網路位置的 IP 地址考慮在內。您也可以使用 Amazon ECS 政策，控制特定 Amazon Virtual Private Cloud 端點或特定 VPC 的存取權。實際上，這只會隔離網路中特定 VPC 對指定 Amazon ECS 資源 AWS 的網路存取。如需詳細資訊，請參閱Amazon ECS 介面 VPC 端點 (AWS PrivateLink)。

Amazon ECS 介面 VPC 端點 (AWS PrivateLink)

透過設定 Amazon ECS 以使用介面 VPC 端點，可提升 VPC 的安全狀態。介面端點採用一種技術 AWS PrivateLink，可讓您使用私有 IP 地址來私下存取 Amazon ECS APIs。 AWS PrivateLink 會限制 VPC 和 Amazon ECS 與 Amazon 網路之間的所有網路流量。您不需要網際網路閘道、NAT 裝置或虛擬私有閘道。

如需 AWS PrivateLink 和 VPC 端點的詳細資訊，請參閱《Amazon VPC 使用者指南》中的 VPC 端點。

考量事項

自 2023 年 12 月 23 日起， 區域中的端點考量事項

在您設定 Amazon ECS 的介面 VPC 端點之前，請注意以下幾點考量：

- 您必須具有下列區域特定的 VPC 端點：

> **Note**
>
> 如果您未設定所有端點，您的流量會經過公有端點，而不是 VPC 端點。

- `com.amazonaws.region.ecs-agent`
- `com.amazonaws.region.ecs-telemetry`
- `com.amazonaws.region.ecs`

例如，加拿大西部（卡加利）(ca-west-1) 區域需要下列 VPC 端點：

- `com.amazonaws.ca-west-1.ecs-agent`
- `com.amazonaws.ca-west-1.ecs-telemetry`
- `com.amazonaws.ca-west-1.ecs`

- 當您使用範本在新區域中建立 AWS 資源，且範本已從 2023 年 12 月 23 日之前引進的 區域複製時，取決於複製自 區域，請執行下列其中一個操作。

  例如，複製自 區域是美國東部（維吉尼亞北部）(us-east-1)。複製到 區域是加拿大西部（卡加利）(ca-west-1)。

組態	動作	
從 區域複製沒有任何 VPC 端點。	為新區域建立所有三個 VPC 端點（例如 `com.amazonaws.ca-west-1.ecs-agent`）。	
從 區域複製包含區域特定的 VPC 端點。	a. 為新區域建立所有三個 VPC 端點（例如 `com.amazonaws.ca-west-1.ecs-agent` ）。 b. 刪除從 區域複製的所有三個 VPC 端點（例如 `com.amazonaws.us-east-1.ecs-agent` ）。	

Fargate 啟動類型的 Amazon ECS VPC 端點的考量事項

當在部署 Fargate 任務的相同 VPC `ecr.api`中存在 `ecr.dkr`和 的 VPC 端點時，將使用 VPC 端點。如果沒有 VPC 端點，則會使用 Fargate 介面。

在您設定 Amazon ECS 的介面 VPC 端點之前，請注意以下幾點考量：

- 使用 Fargate 啟動類型的任務不需要 Amazon ECS 的介面 VPC 端點，但您可能需要以下要點中所述的 Amazon ECR、Secrets Manager 或 Amazon CloudWatch Logs 的介面 VPC 端點。
  - 若要讓任務從 Amazon ECR 中提取私有映像，請務必為 Amazon ECR 建立介面 VPC 端點。如需詳細資訊，請參閱 《Amazon Elastic Container Registry 使用者指南》中的介面 VPC 端點 (AWS PrivateLink)。

    > **Important**
    >
    > 如果您設定 Amazon ECR 以使用介面 VPC 端點，您就可以建立包含條件金鑰的任務執行角色，限制存取特定的 VPC 或 VPC 端點。如需詳細資訊，請參閱透過介面端點許可提取 Amazon ECR 映像的 Fargate 任務。

  - 若要讓任務從 Secrets Manager 中提取敏感資料，您必須建立 Secrets Manager 的介面 VPC 端點。如需詳細資訊，請參閱 《AWS Secrets Manager 使用者指南》中的搭配使用 Secrets Manager 與 VPC 端點。
  - 如果 VPC 沒有網際網路閘道且任務使用的是 `awslogs` 日誌驅動程式來將日誌資訊傳送至 CloudWatch Logs，您必須為 CloudWatch Logs 建立介面 VPC 端點。如需詳細資訊，請參閱 《Amazon CloudWatch Logs 使用者指南》中的使用 CloudWatch Events 搭配介面 VPC 端點。
- VPC 端點目前不支援跨區域請求。請確保在計劃對 Amazon ECS 發出 API 呼叫的相同區域中建立端點。例如，假設您想在美國東部 (維吉尼亞北部) 執行任務。那麼，您必須在美國東部 (維吉尼亞北部) 建立 Amazon ECS VPC 端點。在任何其他區域建立的 Amazon ECS VPC 端點無法在美國東部 (維吉尼亞北部) 執行任務。
- 透過 Amazon Route 53，VPC 端點僅支援 Amazon 提供的 DNS。如果您想要使用自己的 DNS，您可以使用條件式 DNS 轉送。如需詳細資訊，請參閱《Amazon VPC 使用者指南》中的 DHCP 選項集。
- 連接到 VPC 端點的安全群組必須允許從 VPC 的私有子網路，透過 TCP 連接埠 443 傳入的連線。
- Envoy 代理的 Service Connect (服務連接) 管理會使用 `com.amazonaws.region.ecs-agent` VPC 端點。當您未使用 VPC 端點時，Envoy 代理的 Service Connect (服務連接) 管理會使用該區域中的 `ecs-sc` 端點。如需每個區域中的 Amazon ECS 端點清單，請參閱 Amazon ECS 端點和配額。

EC2 啟動類型的 Amazon ECS VPC 端點的考量事項

在您設定 Amazon ECS 的介面 VPC 端點之前，請注意以下幾點考量：

- 使用 EC2 啟動類型的任務需要任務啟動所在的容器執行個體，才能執行 `1.25.1` 版或更新版本的 Amazon ECS 容器代理程式。如需詳細資訊，請參閱Amazon ECS Linux 容器執行個體管理。
- 若要讓任務從 Secrets Manager 中提取敏感資料，您必須建立 Secrets Manager 的介面 VPC 端點。如需詳細資訊，請參閱 《AWS Secrets Manager 使用者指南》中的搭配使用 Secrets Manager 與 VPC 端點。
- 如果 VPC 沒有網際網路閘道且任務使用的是 `awslogs` 日誌驅動程式來將日誌資訊傳送至 CloudWatch Logs，您必須為 CloudWatch Logs 建立介面 VPC 端點。如需詳細資訊，請參閱 《Amazon CloudWatch Logs 使用者指南》中的使用 CloudWatch Events 搭配介面 VPC 端點。
- VPC 端點目前不支援跨區域請求。請確保在計劃對 Amazon ECS 發出 API 呼叫的相同區域中建立端點。例如，假設您想在美國東部 (維吉尼亞北部) 執行任務。那麼，您必須在美國東部 (維吉尼亞北部) 建立 Amazon ECS VPC 端點。在任何其他區域中建立的 Amazon ECS VPC 端點無法在美國東部 （維吉尼亞北部） 執行任務。
- 透過 Amazon Route 53，VPC 端點僅支援 Amazon 提供的 DNS。如果您想要使用自己的 DNS，您可以使用條件式 DNS 轉送。如需詳細資訊，請參閱《Amazon VPC 使用者指南》中的 DHCP 選項集。
- 連接到 VPC 端點的安全群組必須允許從 VPC 的私有子網路，透過 TCP 連接埠 443 傳入的連線。

建立 Amazon ECS 的 VPC 端點

若要為 Amazon ECS 服務建立 VPC 端點，請使用 Amazon VPC 使用者指南中的使用界面 VPC 端點程序存取 AWS 服務來建立下列端點。如果您的 VPC 中有現有的容器執行個體，您應該依照其列出順序建立端點。如果您計劃在建立 VPC 端點後才建立您的容器執行個體，則其順序並不重要。

> **Note**
>
> 如果您未設定所有端點，您的流量會經過公有端點，而不是 VPC 端點。
>
> 當您建立端點時，Amazon ECS 也會為端點建立私有 DNS 名稱。例如，`ecs-a.region.amazonaws.com`ecs-agent 和 `ecs-t.region.amazonaws.com` ecs-telemetry。

- `com.amazonaws.region.ecs-agent`

- com.amazonaws.*region*.ecs-telemetry
- com.amazonaws.*region*.ecs

> **Note**
>
> *##* 表示 Amazon ECS 支援之 AWS 區域的區域識別符，例如 us-east-2 表示美國東部 (俄亥俄) 區域。

ecs-agent 端點使用 ecs:poll API，而ecs-telemetry端點使用 ecs:poll和 ecs:StartTelemetrySession API。

如果您擁有的現有任務是使用 EC2 啟動類型，在建立 VPC 端點後，每個容器執行個體需要挑選新的組態。若要這麼做，您必須擇一重新啟動每個容器執行個體或重新啟動每個容器執行個體上的 Amazon ECS 容器代理程式。若要重新開機容器代理程式，請執行下列動作：

若要重新啟動 Amazon ECS 容器代理程式

1. 透過 SSH 登入您的容器執行個體。
2. 停用容器代理程式。

```
sudo docker stop ecs-agent
```

3. 啟動容器代理程式。

```
sudo docker start ecs-agent
```

在建立 VPC 端點並重新啟動每個容器執行個體上的 Amazon ECS 容器代理程式後，所有新啟動的任務都會挑選新的組態。

為 Amazon ECS 建立 VPC 端點政策

您可以將端點政策連接至控制 Amazon ECS 存取權的 VPC 端點。此政策會指定下列資訊：

- 可執行動作的主體。
- 可執行的動作。
- 可供執行動作的資源。

如需詳細資訊，請參閱 Amazon VPC 使用者指南中的使用 VPC 端點控制對服務的存取。

範例：Amazon ECS 動作的 VPC 端點政策

以下是 Amazon ECS 端點政策的範例。在連接至端點後，此政策會授予存取許可，以允許建立並列出叢集。`CreateCluster` 和 `ListClusters` 動作不接受任何資源，因此資源定義已針對所有資源設為 *。

```
{
   "Statement":[
    {
      "Effect": "Allow",
      "Action": [
        "ecs:CreateCluster",
        "ecs:ListClusters"
      ],
      "Resource": [
        "*"
      ]
    }
  ]
}
```

AWS Amazon ECS 的共同責任模型

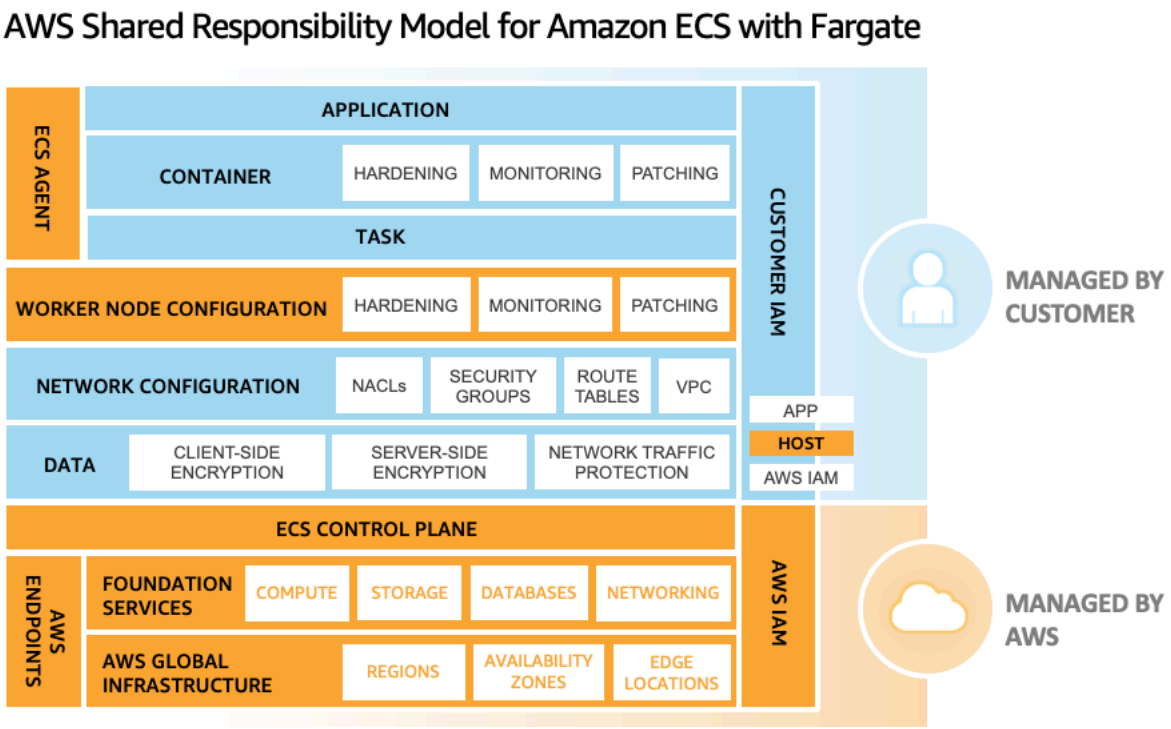
安全與合規是 AWS 和 客戶之間的共同責任。此共用模型有助於減輕客戶的營運負擔，因為 會 AWS 運作、管理和控制從主機作業系統和虛擬化層到服務運作所在設施實體安全性的元件。客戶負責和管理訪客作業系統（包括更新和安全修補程式）、其他相關聯的應用程式軟體，以及所提供安全群組防火牆的 AWS 組態。客戶應審慎思考所選的服務，因為使用的服務、服務與客戶 IT 環境的整合情形，以及適用的法律與法規不同，客戶應承擔的責任也會不同。此共同責任的性質也提供允許部署的彈性和客戶控制。

Fargate 啟動類型

下圖顯示 Fargate 啟動類型的共同責任模型。Fargate 會在隔離的硬體虛擬化環境中執行每個工作負載。因此，每個任務都會取得專用基礎設施容量。在 Fargate 上執行的容器化工作負載不會與其他任務共用作業系統、Linux 核心、網路介面、暫時性儲存、CPU 或記憶體。使用 Fargate 時，客戶不需負責保護執行其容器的運算基礎設施。Fargate 將佈建和修補客戶工作負載執行所在的基礎設施。如需詳細資訊，請參閱Amazon ECS AWS Fargate 的任務淘汰和維護 。

您負責管理下列資源：

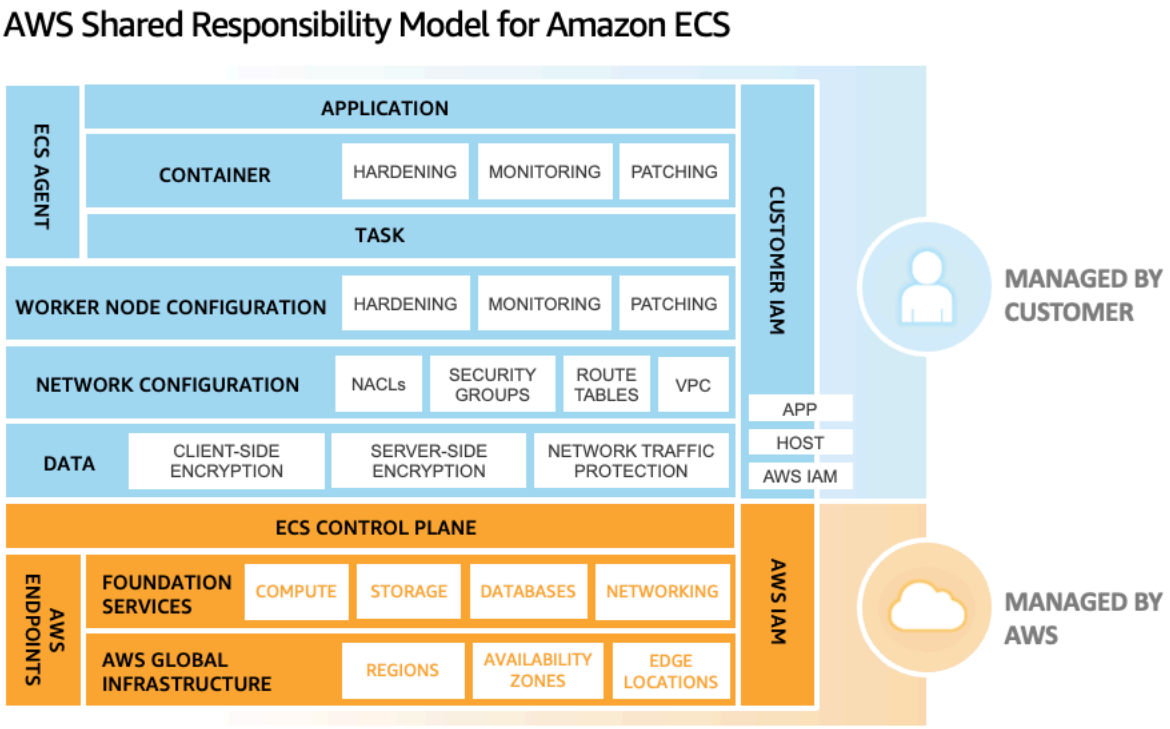
- 網路組態，包括 VPC、NACLs、安全群組和路由表
- 用戶端和服務儲存加密。如需詳細資訊，請參閱Amazon ECS 任務的儲存選項。
- 容器映像。如需詳細資訊，請參閱Amazon ECS 任務和容器安全最佳實務。
- 使用任務角色的應用程式 IAM 許可。如需詳細資訊，請參閱Amazon ECS 任務 IAM 角色。



EC2 啟動類型

下圖顯示 EC2 啟動類型的共同責任。當您在 EC2 執行個體上執行任務時，除了下列資源之外，您還需負責維護 EC2 執行個體：

- Amazon ECS 代理程式。
- • EC2 執行個體 AMI，包括修補和強化。
- 網路組態，包括 VPC、NACLs、安全群組和路由表。
- 用戶端和服務儲存加密。如需詳細資訊，請參閱Amazon ECS 任務的儲存選項。
- 容器映像。如需詳細資訊，請參閱Amazon ECS 任務和容器安全最佳實務。
- 使用任務角色的應用程式 IAM 許可。如需詳細資訊，請參閱Amazon ECS 任務 IAM 角色。



Amazon ECS 的網路安全最佳實務

網路安全性是包含數個子主題的廣泛主題。其中包括傳輸中加密、網路分隔和隔離、防火牆、流量路由和可觀測性。

傳輸中加密

加密網路流量可防止未經授權的使用者在經由網路傳輸資料時攔截和讀取資料。使用 Amazon ECS，網路加密可以透過以下任何方式來實作。

- 使用 Nitro 執行個體：

  根據預設，會自動加密下列 Nitro 執行個體類型之間的流量：C5n、G4、I3en、M5dn、M5n、P3dn、R5dn 和 R5n。當流量透過傳輸閘道、負載平衡器或類似媒介進行路由時，不會加密流量。

  - 傳輸中加密
  - 自 2019 年起的新公告
  - 這場演講來自 re:Inforce 2019
- 將伺服器名稱指示 (SNI) 與 Application Load Balancer 搭配使用：

應用 Application Load Balancer (ALB) 和 Network Load Balancer (NLB) 支援伺服器名稱指示 (SNI)。透過使用 SNI，您可以將多個安全應用程式置於單一接聽程式之後。為此，每個接聽程式都具有自己的 TLS 憑證。建議您使用 AWS Certificate Manager (ACM) 為負載平衡器佈建憑證，然後將其新增至接聽程式的憑證清單。 AWS 負載平衡器搭配 SNI 使用智慧型憑證選取演算法。如果用戶端提供的主機名稱符合憑證清單中的單一憑證，負載平衡器會選擇該憑證。如果用戶端提供的主機名稱符合清單中的多個憑證，負載平衡器會選取用戶端可支援的憑證。範例包括自我簽署憑證或透過 ACM 產生的憑證。

- 使用 Application Load Balancer 的 SNI
- 使用 Network Load Balancer 的 SNI

- 使用 TLS 憑證的端對端加密

  這涉及部署具有任務的 TLS 憑證。這可以是自我簽署的憑證，也可以是受信任憑證授權機構的憑證。您可以透過參考憑證的秘密來取得憑證。如果不是，您可以選擇執行向 ACM 發出憑證簽署請求 (CSR) 的容器，然後將產生的秘密掛載至共用磁碟區。

  - 使用具有 Amazon ECS 第 1 部分的 Network Load Balancer，一路維護傳輸層安全至您容器
  - 將 Transport Layer Security (TLS) 維護到容器第 2 部分：使用 AWS Private Certificate Authority

任務聯網

以下建議是考量 Amazon ECS 的運作方式。Amazon ECS 不使用覆蓋網路。反之，任務會設定為在不同的網路模式下運作。例如，設定為使用 `bridge` 模式的任務會從每部主機上執行的 Docker 網路取得不可路由的 IP 地址。設定為使用 `awsvpc` 網路模式的任務會從主機的子網路取得 IP 地址。設定為使用 `host` 網路的任務會使用主機的網路介面。`awsvpc` 是偏好的網路模式。這是因為此模式為唯一可用來為任務指派安全群組的模式。它也是 Amazon ECS 上可用於 AWS Fargate 任務的唯一模式。

任務的安全群組

建議您將任務設定為使用 `awsvpc` 網路模式。將任務設定為使用此模式後，Amazon ECS 代理程式會自動佈建彈性網絡介面 (ENI) 並將其附加至任務。佈建 ENI 時，任務會註冊在 AWS 安全群組中。安全群組做為虛擬防火牆，可用於控制傳入及傳出的流量。

AWS PrivateLink 和 Amazon ECS

AWS PrivateLink 是一種聯網技術，可讓您為不同的 AWS 服務建立私有端點，包括 Amazon ECS。在沒有網際網路閘道 (IGW) 連接至 Amazon VPC 且沒有其他路由到網際網路的沙盒環境中，則需要端點。使用 AWS PrivateLink 可確保對 Amazon ECS 服務的呼叫保留在 Amazon VPC 內，且不會周

遊網際網路。如需如何為 Amazon ECS 和其他相關服務建立 AWS PrivateLink 端點的說明，請參閱 Amazon ECS 介面 Amazon VPC 端點。

> **Important**
>
> AWS Fargate 任務不需要 Amazon ECS 的 AWS PrivateLink 端點。

Amazon ECR 和 Amazon ECS 都支援端點政策。這些政策可讓您完善服務的 API 存取。例如，您可以為 Amazon ECR 建立端點政策，該政策僅允許將映像推送到特定 AWS 帳戶的登錄檔。這類政策可用於防止透過容器映像洩漏資料，同時仍允許使用者推送至授權的 Amazon ECR 登錄檔。如需詳細資訊，請參閱使用 VPC 端點政策。

下列政策允許您的帳戶中的所有 AWS 主體僅針對 Amazon ECR 儲存庫執行所有動作：

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

您可以透過設定使用新 `PrincipalOrgID` 屬性的條件來進一步增強此功能。這可防止 IAM 主體推送和提取不屬於您一部分的影像 AWS Organizations。如需詳細資訊，請參閱 aws:PrincipalOrgID。

建議對 `com.amazonaws.region.ecr.dkr` 和 `com.amazonaws.region.ecr.api` 端點套用相同的政策。

容器代理程式設定

Amazon ECS 容器代理程式組態檔案包含數個與網路安全相關的環境變數。`ECS_AWSVPC_BLOCK_IMDS` 和 `ECS_ENABLE_TASK_IAM_ROLE_NETWORK_HOST` 用於封鎖任務對 Amazon EC2 中繼資料的存取。`HTTP_PROXY` 用於設定代理程式，以透過 HTTP Proxy 路由連線至網際網路。如需設定代理程式和 Docker 執行期以透過 Proxy 路由的指示，請參閱 HTTP Proxy 組態。

> **Important**
>
> 這些設定在您使用 AWS Fargate時無法使用。

網路安全建議

建議您在設定 Amazon VPC、負載平衡器和網路時執行下列動作。

在適用於 Amazon ECS 的情況下使用網路加密

您應該在適用的情況下使用網路加密。如果資料包含持卡人資料，某些合規計劃 (例如 PCI DSS) 會要求您加密傳輸中的資料。如果您的工作負載有類似的需求，請設定網路加密。

現代瀏覽器在連接至不安全的站點時警告使用者。如果面向大眾的負載平衡器前端是您的伺服器，請使用 TLS/SSL 加密從用戶端瀏覽器到負載平衡器的流量，並在授權時重新加密至後端。

使用**awsvpc**網路模式和安全群組來控制 Amazon ECS 中任務和其他資源之間的流量

當您需要控制任務之間以及任務與其他網路資源之間的流量時，您應該使用 awsvpc 網路模式和安全群組。如果您的服務位於 ALB 之後，請使用安全性群組，以僅允許來自其他使用與 ALB 相同安全群組之網路資源的輸入流量。如果您的應用程式位於 NLB 之後，請將任務的安全群組設定為僅允許來自 Amazon VPC CIDR 範圍以及指派給 NLB 的靜態 IP 地址的輸入流量。

安全群組也應該用來控制 Amazon VPC 內任務與其他資源 (例如 Amazon RDS 資料庫) 之間的流量。

需要嚴格隔離網路流量時，在不同的 Amazon VPCs 中建立 Amazon ECS 叢集

當網路流量需要嚴格隔離時，您應該在獨立的 Amazon VPC 中建立叢集。避免在叢集上執行具有嚴格安全要求的工作負載，而工作負載不必遵循這些要求。當強制執行嚴格的網路隔離時，請在獨立的 Amazon VPC 中建立叢集，並使用 Amazon VPC 端點將服務選擇性公開給其他 Amazon VPC。如需詳細資訊，請參閱 VPC 端點。

在 AWS PrivateLink Amazon ECS 需要時設定端點

您應該在必要時設定 AWS PrivateLink 端點端點。如果您的安全政策阻止您將網際網路閘道 (IGW) 連接至 Amazon VPCs，請為 Amazon ECS 和其他服務設定 AWS PrivateLink 端點，例如 Amazon ECR、 AWS Secrets Manager和 Amazon CloudWatch。

使用 Amazon VPC 流程日誌來分析 Amazon ECS 中長時間執行任務的往返流量

您應該使用 Amazon VPC 流程日誌分析流入及流出長時間執行任務的流量。使用 awsvpc 網路模式的任務會取得自己的 ENI。執行此操作，您可以使用 Amazon VPC 流程日誌監控進出個別任務的流量。Amazon VPC 流程日誌 (v3) 的近期更新使用流量中繼資料 (包括 VPC ID、子網路 ID 和執行個體 ID) 來豐富日誌。此中繼資料可用於協助縮小調查範圍。如需詳細資訊，請參閱 Amazon VPC 流程日誌。

> **Note**
>
> 由於容器的暫時性本質，流程日誌不一定是分析不同容器或容器以及其他網路資源之間流量模式的有效方法。

Amazon ECS 任務和容器安全最佳實務

您應該將容器映像視為對抗攻擊的第一道防線。不安全、構造不良的映像可允許攻擊者逸出容器的邊界並取得主機的存取。您應該執行以下操作以減輕發生這種情況的風險。

建議您在設定任務和容器時進行下列操作。

建立最小或使用 distroless 映像

首先從容器映像中刪除所有無關的二進位檔案。如果您使用 Amazon ECR Public Gallery 中不熟悉的映像，請檢查映像以參考每個容器層的內容。您可以使用 Dive 之類的應用程式來執行此操作。

或者，您可以使用僅包含應用程式及其執行期相依項的 distroless 映像。此映像不包含套件管理工具或 Shell。Distroless 映像可改善「掃描器的訊號雜訊，並減少您所需要確定來源的負擔」。如需詳細資訊，請參閱 distroless 上的 GitHub 文件。

Docker 具有一種機制，用於從稱為 scratch 的保留最小映像建立映像。如需詳細資訊，請參閱 Docker 文件中的使用 scratch 建立簡單的父映像。使用 Go 等語言，您可以建立一個靜態連結的二進位檔案並在 Dockerfile 中作為參考。以下範例展示如何完成這項操作。

```
############################
# 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.
```

上一個範例也是多階段建置的範例。從安全的角度來看，這些類型的建置很有吸引力，因為您可以使用這些建置來最小化推送至容器登錄檔的最終映像大小。沒有構建工具和其他無關二進位檔案的容器映像透過減少映像的受攻擊面來改善您的安全狀態。如需多階段建置的詳細資訊，請參閱 Docker 文件中的多階段建置。

掃描您的映像是否有漏洞

容器映像檔與虛擬機器對應部分類似，容器映像檔可以包含具有漏洞的二進位檔案和應用程式程式庫，或隨著時間的推移而產生漏洞。防止漏洞攻擊的最佳方法是使用映像掃描器定期掃描映像。

可以透過推送或隨需掃描 (每 24 小時掃描一次) 儲存在 Amazon ECR 中的映像。Amazon ECR 基本掃描使用 Clair，這是一種開放原始碼映像掃描解決方案。Amazon ECR 增強了使用 Amazon Inspector 的掃描。掃描映像後，結果會記錄到 Amazon EventBridge 中的 Amazon ECR 事件串流。您也可以從 Amazon ECR 主控台內或呼叫 DescribeImageScanFindings API 來查看掃描的結果。應刪除或重建具有 `HIGH` 或 `CRITICAL` 漏洞的映像。如果已部署的映像出現漏洞，則應盡快更換。

Docker Desktop Edge 2.3.6.0 以上版本可以掃描本機映像。掃描由 Snyk 提供支援，其為一種應用程式安全服務。當發現漏洞時，Snyk 會識別 Dockerfile 中具漏洞的圖層和相依項。也建議使用安全的替代方法，例如使用具有較少漏洞的更細微基礎映像，或將特定套件升級至較新版本。透過使用 Docker 掃描，開發人員可以在將映像推送至登錄檔之前解決潛在的安全問題。

- 使用 Amazon ECR 自動化映像合規，並 AWS Security Hub說明如何從 AWS Security Hub Amazon ECR 中顯示漏洞資訊，以及透過封鎖易受攻擊映像的存取來自動修補。

從映像移除特殊權限

存取權標記 `setuid` 並 `setgid` 允許執行具有擁有者許可或可執行檔群組的可執行檔。從映像中移除具有這些存取權的所有二進位檔案，因為這些二進位檔案可用來提升權限。考慮刪除所有可用於惡意目的的 Shell 和共用程序，如 `nc` 和 `curl` 。您可以使用以下命令找到具有 `setuid` 和 `setgid` 存取權的檔案。

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

若要從這些檔案中刪除這些特殊權限，請將以下指令新增至容器映像中。

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

建立一組精選映像

為組織中的不同應用程式堆疊建立一組經過審查的映像，而不是允許開發人員建立自己的映像。透過執行此動作，開發人員可以放棄學習如何編寫 Dockerfile 並專注於編寫程式碼。當變更合併至您的程式碼庫中時，CI/CD 管道可以自動編譯資產，然後將其儲存在成品儲存庫中。最後，將成品複製到適當的映像中，然後再將其推送至 Docker 登錄檔 (例如 Amazon ECR)。至少您應該建立一組基礎映像，開發人員可以從中建立自己的 Dockerfile。您應避免從 Docker Hub 推送映像。您並非永遠知道映像中的內容，而前 1000 個映像中大約有五分之一存在漏洞。這些映像及其漏洞的清單可以在 https://vulnerablecontainers.org/ 找到。

掃描應用程式套件和程式庫是否有漏洞

現在，使用開放原始碼程式庫很普遍。如同使用作業系統和 OS 套件，這些程式庫可能存在漏洞。作為開發生命週期的一部分，這些程式庫應在發現嚴重漏洞時進行掃描和更新。

Docker Desktop 使用 Snyk 執行本機掃描。也可以用來尋找開放原始碼程式庫中的漏洞和潛在的授權問題。可以直接整合至開發人員工作流程中，讓您能夠減輕開放原始碼程式庫所帶來的風險。如需詳細資訊，請參閱下列主題：

- 開放原始碼應用程式安全工具包括用於偵測應用程式漏洞的工具清單。

執行靜態程式碼分析

建置容器映像之前，您應該先執行靜態程式碼分析。此分析會針對您的原始程式碼執行，並用來識別惡意行為 (例如故障注入) 可能利用的編碼錯誤和程式碼。您可以使用 Amazon Inspector。如需詳細資訊，請參閱《Amazon Inspector 使用者指南》中的使用 Amazon Inspector 掃描 Amazon ECR 容器映像。 Amazon Inspector

以非根使用者身分執行容器

您應以非根使用者身分執行容器。在預設情況下，容器以 `root` 使用者身份執行，除非在 Dockerfile 中包含該 USER 指令。Docker 指派的預設 Linux 功能會限制可做為 `root` 執行的動作，但僅限於稍微限制。例如，做為 `root` 執行的容器仍不允許存取裝置。

作為 CI/CD 管道的一部分，您應該檢查 Docerfile 來查找 USER 指令，如果缺少該指令，則建置失敗。如需詳細資訊，請參閱下列主題：

- Dockerfile-lint 是來自 RedHat 的開放原始碼工具，可用於檢查檔案是否符合最佳實務。
- Hadolint 是建置符合最佳實務 Docker 映像檔的另一種工具。

使用唯讀的根檔案系統

您應該使用唯讀的根檔案系統。容器的根檔案系統預設為可寫入。當您使用 RO (唯讀) 根檔案系統設定容器時，會強制您明確定義資料可以保存的位置。這會減少您的受攻擊面，因為除非特別授予許可，否則無法寫入容器的檔案系統。

> **Note**
>
> 擁有唯讀根檔案系統可能會導致某些作業系統套件發生問題，這些套件預期能夠寫入檔案系統。如果您打算使用唯讀的根檔案系統，請事先徹底測試。

使用 CPU 和記憶體限制來設定任務 (Amazon EC2)

您應該使用 CPU 和記憶體限制來設定任務，以將下列風險降至最低。任務的資源限制會設定任務內所有容器可保留的 CPU 和記憶體數量上限。如果未設定限制，則任務可以存取主機的 CPU 和記憶體。這可能會導致部署在共用主機上的任務會剝奪其他任務系統資源的問題。

> **Note**
>
> AWS Fargate 任務上的 Amazon ECS 需要您指定 CPU 和記憶體限制，因為它會將這些值用於計費目的。對於 Amazon ECS Fargate 而言，一項佔用所有系統資源的任務並不是問題，因為每個任務都在自己的專用預留執行個體上執行。如果您未指定記憶體限制，Amazon ECS 會為每個容器配置至少 4MB。同樣，如果沒有為任務設定 CPU 限制，Amazon ECS 容器代理程式至少會為其指派 2 個 CPU。

不可變標籤與 Amazon ECR 一起使用

使用 Amazon ECR，您可以且應使用具有不可變標籤的設定映像。這樣可以防止將已更改或更新版本的映像推送至具相同標籤的映像儲存庫。這樣可以防止攻擊者將破解版本的映像推送至具相同標籤的映像。透過使用不可變標籤，您可以有效地強制自己為每次變更推送具有不同標籤的新映像。

避免以特權方式執行容器 (Amazon EC2)

您應該避免以特權方式執行容器。對於背景，在主機上使用擴展權限執行作為 `privileged` 執行的容器。這表示容器會繼承在主機上指派給 `root` 的所有 Linux 功能。應嚴格限制或禁止其使用。我們建議將 Amazon ECS 容器代理程式環境變數 `ECS_DISABLE_PRIVILEGED` 設定為 `true`，如不需要 `privileged` 時，以防止容器在特定主機上作為 `privileged` 執行。或者，您可以使用 AWS Lambda 掃描任務定義，以使用 `privileged` 參數。

> **Note**
>
> AWS Fargate上的Amazon ECS 不支援作為 `privileged` 執行的容器。

從容器中移除不必要的 Linux 功能

以下是指派給 Docker 容器的預設 Linux 功能清單。如需有關每項功能的詳細資訊，請參閱 Linux 功能概觀。

```
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
```

如果容器不需要上面列出的所有 Docker 核心功能，請考慮將其從容器中刪除。如需每個 Docker 核心功能的詳細資訊，請參閱 KernelCapabilities。透過執行以下操作，您可以了解正在使用哪些功能：

- 安裝作業系統套件 libcap-ng 並執行 `pscap` 公用程式，以列出每個程序正在使用的功能。
- 您也可以使用 capsh 來解讀程序正在使用哪些功能。

使用客戶自管金鑰 (CMK) 來加密推送至 Amazon ECR 的映像

您應該使用客戶自管金鑰 (CMK) 來加密推送至 Amazon ECR 的映像。推送至 Amazon ECR 的影像會使用 a AWS Key Management Service (AWS KMS) 受管金鑰自動進行靜態加密。如果您寧願使用自己的金鑰，Amazon ECR 現在支援使用客戶受管金鑰 (CMK) AWS KMS 加密。使用 CMK 啟用伺服器端加密之前，請檢閱靜態加密文件中列出的注意事項。

Amazon ECS 教學課程

以下教學課程說明如何使用 Amazon ECS 執行常見的任務。

您可以使用下列任何教學課程，使用 在 Amazon ECS 上部署任務 AWS CLI

教學課程概觀	進一步了解	
為 Fargate 啟動類型建立 Linux 任務。	使用 為 Fargate 啟動類型建立 Amazon ECS Linux 任務 AWS CLI	
為 Fargate 啟動類型建立 Windows 任務。	使用 為 Fargate 啟動類型建立 Amazon ECS Windows 任務 AWS CLI	
為 EC2 啟動類型建立 Linux 任務。	使用 為 EC2 啟動類型建立 Amazon ECS 任務 AWS CLI	

您可以使用下列任何教學課程，進一步了解監控和記錄。

教學課程概觀	進一步了解	
設定簡單的 Lambda 函數，以接聽任務事件並將其寫入 CloudWatch Logs 日誌串流。	設定 Amazon ECS 接聽 CloudWatch Events 事件	
設定 Amazon EventBridge 事件規則，該規則只會擷取任務因其中一個必要容器終止而停止執行的任務事件。	傳送 Amazon ECS 任務已停止事件的 Amazon Simple Notification Service 提醒	
串連原始屬於一個內容但分割為多個記錄或日誌行的日誌訊息。	串連多行或堆疊追蹤 Amazon ECS 日誌訊息	

教學課程概觀	進一步了解	
在 Amazon ECS 中執行的 Windows 執行個體上部署 Fluent Bit 容器，將 Windows 任務產生的日誌串流到 Amazon CloudWatch 以進行集中式記錄。	在 Amazon ECS Windows 容器上部署 Fluent Bit	

您可以使用下列任何教學課程，進一步了解如何在 Amazon ECS 上使用 Active Directory 身分驗證與群組受管服務帳戶。

教學課程概觀	進一步了解	
在 EC2 上使用群組受管服務帳戶搭配 Linux 容器。	在 Amazon ECS 上使用 gMSA for EC2 Linux容器 EC2	
在 EC2 上使用群組受管服務帳戶搭配 Windows 容器。	了解如何使用 gMSAs EC2 Windows 容器 for Amazon ECS	
在 Fargate 上使用群組受管服務帳戶搭配 Linux 容器。	在 Fargate 上使用 gMSA 做為 Linux容器	
建立執行 Windows 容器的任務，該容器具有使用無網域群組受管服務帳戶存取 Active Directory 的登入資料。	gMSA 使用 搭配無網域使用 Amazon ECS Windows 容器 AWS CLI	

使用 為 Fargate 啟動類型建立 Amazon ECS Linux 任務 AWS CLI

以下步驟會協助您使用 AWS CLI在 Amazon ECS 中設定叢集、註冊任務定義、執行 Linux 任務，以及執行其他常見案例。使用最新版本的 AWS CLI。如需如何升級至最新版本的詳細資訊，請參閱安裝或更新至最新版本的 AWS CLI。

主題

必要條件

本教學課程假設已完成下列先決條件。

- AWS CLI 已安裝並設定最新版本的 。如需安裝或升級 AWS CLI、安裝或更新至最新版本 AWS CLI的詳細資訊。
- 已完成「設定以使用 Amazon ECS。」中的步驟。
- 您的 AWS 使用者具有 IAM AmazonECS_FullAccess 政策範例中指定的必要許可。
- 您已建立 VPC 和安全群組。本教學課程使用託管於 Amazon ECR Public 的容器映像，因此任務必須有網際網路連線。若要將網際網路的路由提供給任務，請使用下列其中一個選項。
  - 透過具有彈性 IP 地址的 NAT 閘道使用私有子網路。
  - 使用公有子網路，然後將公有 IP 地址指派給任務。

  如需詳細資訊，請參閱the section called “建立 Virtual Private Cloud”。

  如需有關安全群組和規則的資訊，請參閱《Amazon Virtual Private Cloud 使用者指南》中的 VPC 的預設安全群組和規則範例。
- 如果您使用私有子網路遵循本教學課程，則可以使用 Amazon ECS Exec 直接與您的容器互動並測試部署。您必須建立任務 IAM 角色才能使用 ECS Exec。如需任務 IAM 角色和其他先決條件的詳細資訊，請參閱使用 Amazon ECS Exec 監控 Amazon ECS 容器。
- （選用）AWS CloudShell 是一種工具，可為客戶提供命令列，而不需要建立自己的 EC2 執行個體。如需詳細資訊，請參閱AWS CloudShell 《 使用者指南》中的什麼是 AWS CloudShell？。

步驟 1：建立叢集

您的帳戶預設會得到 default 叢集。

> **Note**
>
> 使用為您提供之 default 叢集的優點是，您不必在後續的命令中指定 --cluster *cluster_name* 選項。如果您建立的是自己的叢集，不是預設叢集，您必須在打算對該叢集使用的每個命令中指定 --cluster *cluster_name*。

使用下列命令以唯一的名稱建立您自己的叢集：

```
aws ecs create-cluster --cluster-name fargate-cluster
```

輸出：

```
{
    "cluster": {
        "status": "ACTIVE",
        "defaultCapacityProviderStrategy": [],
        "statistics": [],
        "capacityProviders": [],
        "tags": [],
        "clusterName": "fargate-cluster",
        "settings": [
            {
                "name": "containerInsights",
                "value": "disabled"
            }
        ],
        "registeredContainerInstancesCount": 0,
        "pendingTasksCount": 0,
        "runningTasksCount": 0,
        "activeServicesCount": 0,
        "clusterArn": "arn:aws:ecs:region:aws_account_id:cluster/fargate-cluster"
    }
}
```

步驟 2：註冊 Linux 任務定義

您必須先註冊任務定義，才能在您的 ECS 叢集中執行任務。任務定義是分在一組的容器清單。下列範例是使用託管於 Docker Hub 的 httpd 容器映像來建立 PHP Web 應用程式的簡單任務定義。如需可用之任務定義參數的詳細資訊，請參閱「Amazon ECS 任務定義」。在本教學課程中，僅當您在私有子網路中部署任務並希望測試部署時，才需要 taskRoleArn。如 必要條件 中所述，將 taskRoleArn 取代為您為使用 ECS Exec 建立的 IAM 任務角色。

```
{
       "family": "sample-fargate",
       "networkMode": "awsvpc",
       "taskRoleArn": "arn:aws:iam::aws_account_id:role/execCommandRole",
       "containerDefinitions": [
           {
               "name": "fargate-app",
               "image": "public.ecr.aws/docker/library/httpd:latest",
               "portMappings": [
                   {
                       "containerPort": 80,
                       "hostPort": 80,
                       "protocol": "tcp"
                   }
               ],
               "essential": true,
               "entryPoint": [
                   "sh",
                   "-c"
               ],
               "command": [
                   "/bin/sh -c \"echo '<html> <head> <title>Amazon ECS Sample
 App</title> <style>body {margin-top: 40px; background-color: #333;} </style> </
head><body> <div style=color:white;text-align:center> <h1>Amazon ECS Sample App</h1>
 <h2>Congratulations!</h2> <p>Your application is now running on a container in Amazon
 ECS.</p> </div></body></html>' >  /usr/local/apache2/htdocs/index.html && httpd-
foreground\""
               ]
           }
       ],
       "requiresCompatibilities": [
           "FARGATE"
       ],
       "cpu": "256",
       "memory": "512"
```

```
}
```

將任務定義 JSON 儲存為檔案，並使用 `--cli-input-json file://path_to_file.json` 選項進行傳遞。

將 JSON 檔案用於容器定義中：

```
aws ecs register-task-definition --cli-input-json file://$HOME/tasks/fargate-task.json
```

register-task-definition 命令會在完成註冊後傳回任務定義的描述。

步驟 3：列出任務定義

您可以使用 list-task-definitions 命令隨時列出您帳戶的任務定義。這個命令的輸出會顯示 `family` 和 `revision` 值，可在呼叫 run-task 或 start-task 時一起使用。

```
aws ecs list-task-definitions
```

輸出：

```
{
    "taskDefinitionArns": [
        "arn:aws:ecs:region:aws_account_id:task-definition/sample-fargate:1"
    ]
}
```

步驟 4：建立服務

為您的帳戶註冊任務之後，您就可以在您的叢集中建立已註冊任務的服務。在此範例中，您將透過在叢集中執行之 `sample-fargate:1` 任務定義的一個執行個體建立服務。該任務需要網際網路的路由，因此您可透過兩種方法達成。一種方法是，使用透過 NAT 閘道所設定的私有子網路，而該閘道在公有子網路中具有彈性 IP 地址。另一種方法是，使用公有子網路並將公有 IP 地址指派至任務。以下提供這兩種範例。

使用私有子網路的範例。需要 `enable-execute-command` 選項才能使用 Amazon ECS Exec。

```
aws ecs create-service --cluster fargate-cluster --service-name fargate-service --
task-definition sample-fargate:1 --desired-count 1 --launch-type "FARGATE" --network-
configuration "awsvpcConfiguration={subnets=[subnet-abcd1234],securityGroups=[sg-
abcd1234]}" --enable-execute-command
```

使用公有子網路的範例。

```
aws ecs create-service --cluster fargate-cluster --service-name fargate-service --
task-definition sample-fargate:1 --desired-count 1 --launch-type "FARGATE" --network-
configuration "awsvpcConfiguration={subnets=[subnet-abcd1234],securityGroups=[sg-
abcd1234],assignPublicIp=ENABLED}"
```

create-service 命令會在完成註冊後傳回任務定義的描述。

步驟 5：列出服務

列出您叢集的服務。您應該會看到您在先前一節所建立的服務。您可以記下這個命令傳回的服務名稱或完整的 ARN，稍後將用以描述服務。

```
aws ecs list-services --cluster fargate-cluster
```

輸出：

```
{
    "serviceArns": [
        "arn:aws:ecs:region:aws_account_id:service/fargate-cluster/fargate-service"
    ]
}
```

步驟 6：描述執行中的服務

使用之前擷取的服務名稱來描述服務，以取得任務的詳細資訊。

```
aws ecs describe-services --cluster fargate-cluster --services fargate-service
```

如果成功，這會傳回服務失敗和服務的描述。例如，在 `services` 區段中，您將可以找到部署的相關資訊，例如執行中或擱置中的任務狀態。您也可以找到任務定義、網路組態和時間戳記事件的相關資訊。在失敗區段中，您將可以找到與呼叫相關聯的失敗 (如果有的話) 的相關資訊。如需故障診斷，請參閱服務事件訊息。如需服務描述的詳細資訊，請參閱描述服務。

```
{
    "services": [
        {
            "networkConfiguration": {
                "awsvpcConfiguration": {
                    "subnets": [
```

```
                    "subnet-abcd1234"
                ],
                "securityGroups": [
                    "sg-abcd1234"
                ],
                "assignPublicIp": "ENABLED"
            }
        },
        "launchType": "FARGATE",
        "enableECSManagedTags": false,
        "loadBalancers": [],
        "deploymentController": {
            "type": "ECS"
        },
        "desiredCount": 1,
        "clusterArn": "arn:aws:ecs:region:aws_account_id:cluster/fargate-cluster",
        "serviceArn": "arn:aws:ecs:region:aws_account_id:service/fargate-service",
        "deploymentConfiguration": {
            "maximumPercent": 200,
            "minimumHealthyPercent": 100
        },
        "createdAt": 1692283199.771,
        "schedulingStrategy": "REPLICA",
        "placementConstraints": [],
        "deployments": [
            {
                "status": "PRIMARY",
                "networkConfiguration": {
                    "awsvpcConfiguration": {
                        "subnets": [
                            "subnet-abcd1234"
                        ],
                        "securityGroups": [
                            "sg-abcd1234"
                        ],
                        "assignPublicIp": "ENABLED"
                    }
                },
                "pendingCount": 0,
                "launchType": "FARGATE",
                "createdAt": 1692283199.771,
                "desiredCount": 1,
                "taskDefinition": "arn:aws:ecs:region:aws_account_id:task-
definition/sample-fargate:1",
```

```
                    "updatedAt": 1692283199.771,
                    "platformVersion": "1.4.0",
                    "id": "ecs-svc/9223370526043414679",
                    "runningCount": 0
                }
            ],
            "serviceName": "fargate-service",
            "events": [
                {
                    "message": "(service fargate-service) has started 2 tasks: (task
 53c0de40-ea3b-489f-a352-623bf1235f08) (task d0aec985-901b-488f-9fb4-61b991b332a3).",
                    "id": "92b8443e-67fb-4886-880c-07e73383ea83",
                    "createdAt": 1510811841.408
                },
                {
                    "message": "(service fargate-service) has started 2 tasks: (task
 b4911bee-7203-4113-99d4-e89ba457c626) (task cc5853e3-6e2d-4678-8312-74f8a7d76474).",
                    "id": "d85c6ec6-a693-43b3-904a-a997e1fc844d",
                    "createdAt": 1510811601.938
                },
                {
                    "message": "(service fargate-service) has started 2 tasks: (task
 cba86182-52bf-42d7-9df8-b744699e6cfc) (task f4c1ad74-a5c6-4620-90cf-2aff118df5fc).",
                    "id": "095703e1-0ca3-4379-a7c8-c0f1b8b95ace",
                    "createdAt": 1510811364.691
                }
            ],
            "runningCount": 0,
            "status": "ACTIVE",
            "serviceRegistries": [],
            "pendingCount": 0,
            "createdBy": "arn:aws:iam::aws_account_id:user/user_name",
            "platformVersion": "LATEST",
            "placementStrategy": [],
            "propagateTags": "NONE",
            "roleArn": "arn:aws:iam::aws_account_id:role/aws-service-role/
ecs.amazonaws.com/AWSServiceRoleForECS",
            "taskDefinition": "arn:aws:ecs:region:aws_account_id:task-definition/
sample-fargate:1"
        }
    ],
    "failures": []
}
```

步驟 7：測試

使用公有子網路部署的測試任務

描述服務中的任務，以便取得任務的彈性網路介面 (ENI)。

首先，要取得任務 ARN。

```
aws ecs list-tasks --cluster fargate-cluster --service fargate-service
```

輸出包含任務 ARN。

```
{
    "taskArns": [
        "arn:aws:ecs:us-east-1:123456789012:task/fargate-service/EXAMPLE
    ]
}
```

描述任務並找到 ENI ID。將任務 ARN 用於 tasks 參數。

```
aws ecs describe-tasks --cluster fargate-cluster --tasks arn:aws:ecs:us-
east-1:123456789012:task/service/EXAMPLE
```

連接資訊將在輸出中列出。

```
{
    "tasks": [
        {
            "attachments": [
                {
                    "id": "d9e7735a-16aa-4128-bc7a-b2d5115029e9",
                    "type": "ElasticNetworkInterface",
                    "status": "ATTACHED",
                    "details": [
                        {
                            "name": "subnetId",
                            "value": "subnetabcd1234"
                        },
                        {
                            "name": "networkInterfaceId",
                            "value": "eni-0fa40520aeEXAMPLE"
                        },
```

```
                    ]
                }
…
}
```

描述 ENI 以獲得公有 IP 地址。

```
aws ec2 describe-network-interfaces --network-interface-id  eni-0fa40520aeEXAMPLE
```

公有 IP 地址會在輸出中。

```
{
    "NetworkInterfaces": [
        {
            "Association": {
                "IpOwnerId": "amazon",
                "PublicDnsName": "ec2-34-229-42-222.compute-1.amazonaws.com",
                "PublicIp": "198.51.100.2"
            },
…
}
```

在 Web 瀏覽器中輸入公有 IP 地址，然後您應該會看到顯示 Amazon ECS 範例應用程式的網頁。

使用私有子網路部署的測試任務

描述任務並找出 managedAgents，以驗證 ExecuteCommandAgent 是否正在執行。記下 privateIPv4Address 以供稍後使用。

```
aws ecs describe-tasks --cluster fargate-cluster --tasks arn:aws:ecs:us-
east-1:123456789012:task/fargate-service/EXAMPLE
```

受管代理程式資訊將在輸出中列出。

```
{
     "tasks": [
        {
            "attachments": [
                {
                    "id": "d9e7735a-16aa-4128-bc7a-b2d5115029e9",
                    "type": "ElasticNetworkInterface",
                    "status": "ATTACHED",
```

```
                    "details": [
                        {
                            "name": "subnetId",
                            "value": "subnetabcd1234"
                        },
                        {
                            "name": "networkInterfaceId",
                            "value": "eni-0fa40520aeEXAMPLE"
                        },
                        {
                            "name": "privateIPv4Address",
                            "value": "10.0.143.156"
                        }
                    ]
                }
            ],
    ...
    "containers": [
        {
        ...
       "managedAgents": [
                        {
                            "lastStartedAt": "2023-08-01T16:10:13.002000+00:00",
                            "name": "ExecuteCommandAgent",
                            "lastStatus": "RUNNING"
                        }
                ],
        ...
    }
```

驗證 `ExecuteCommandAgent` 正在執行之後，您可以執行下列命令，以在任務中的容器上執行互動式 Shell。

```
aws ecs execute-command --cluster fargate-cluster \
    --task  arn:aws:ecs:us-east-1:123456789012:task/fargate-service/EXAMPLE  \
    --container  fargate-app \
    --interactive \
    --command "/bin/sh"
```

互動式 Shell 執行後，執行以下命令來安裝 cURL。

```
apt update
```

```
apt install curl
```

安裝 cURL 後，使用您先前獲得的私有 IP 地址執行以下命令。

```
 curl 10.0.143.156
```

您應該會看到相當於 Amazon ECS 範例應用程式網頁的 HTML。

```
<html>
    <head>
     <title>Amazon ECS Sample App</title>
     <style>body {margin-top: 40px; background-color: #333;} </style>
    </head>
      <body>
      <div style=color:white;text-align:center>
      <h1>Amazon ECS Sample App</h1>
      <h2>Congratulations!</h2> <p>Your application is now running on a container in
 Amazon ECS.</p>
      </div>
      </body>
</html>
```

步驟 8：清除

完成此教學課程時，建議您清除相關聯的資源，以免未使用的資源產生費用。

刪除服務。

```
aws ecs delete-service --cluster fargate-cluster --service fargate-service --force
```

刪除叢集。

```
aws ecs delete-cluster --cluster fargate-cluster
```

使用 為 Fargate 啟動類型建立 Amazon ECS Windows 任務 AWS CLI

以下步驟會協助您使用 AWS CLI在 Amazon ECS 中設定叢集、註冊任務定義、執行 Windows 任務，以及執行其他常見案例。請務必使用最新版本的 AWS CLI。如需如何升級至最新版本的詳細資訊，請參閱安裝或更新至最新版本的 AWS CLI。

主題

必要條件

本教學課程假設已完成下列先決條件。

- AWS CLI 已安裝並設定最新版本的 。如需安裝或升級 的詳細資訊 AWS CLI，請參閱安裝或更新至最新版本的 AWS CLI。
- 已完成「設定以使用 Amazon ECS。」中的步驟。
- 您的 AWS 使用者具有 IAM AmazonECS_FullAccess 政策範例中指定的必要許可。
- 您已建立 VPC 和安全群組。本教學課程使用託管於 Docker Hub 的容器映像，因此任務必須有網際網路連線。若要將網際網路的路由提供給任務，請使用下列其中一個選項。
  - 透過具有彈性 IP 地址的 NAT 閘道使用私有子網路。
  - 使用公有子網路，然後將公有 IP 地址指派給任務。

  如需詳細資訊，請參閱the section called “建立 Virtual Private Cloud”。

  如需有關安全群組和規則的資訊，請參閱《Amazon Virtual Private Cloud 使用者指南》中的 VPC 的預設安全群組和規則範例。

- （選用）AWS CloudShell 是一種工具，可為客戶提供命令列，而不需要建立自己的 EC2 執行個體。如需詳細資訊，請參閱AWS CloudShell 《 使用者指南》中的什麼是 AWS CloudShell？。

步驟 1：建立叢集

您的帳戶預設會得到 `default` 叢集。

> **Note**
>
> 使用為您提供之 `default` 叢集的優點是，您不必在後續的命令中指定 `--cluster` *`cluster_name`* 選項。如果您建立的是自己的叢集，不是預設叢集，您必須在打算對該叢集使用的每個命令中指定 `--cluster` *`cluster_name`*。

使用下列命令以唯一的名稱建立您自己的叢集：

```
aws ecs create-cluster --cluster-name fargate-cluster
```

輸出：

```
{
    "cluster": {
        "status": "ACTIVE",
        "statistics": [],
        "clusterName": "fargate-cluster",
        "registeredContainerInstancesCount": 0,
        "pendingTasksCount": 0,
        "runningTasksCount": 0,
        "activeServicesCount": 0,
        "clusterArn": "arn:aws:ecs:region:aws_account_id:cluster/fargate-cluster"
    }
}
```

步驟 2：註冊 Windows 任務定義

您必須先註冊任務定義，才能在您的 Amazon ECS 叢集中執行 Windows 任務。任務定義是分在一組的容器清單。以下範例是建立 Web 應用程式的簡單任務定義。如需可用之任務定義參數的詳細資訊，請參閱「Amazon ECS 任務定義」。

```
{
```

```
    "containerDefinitions": [
        {
            "command": ["New-Item -Path C:\\inetpub\\wwwroot\\index.html -Type file
 -Value '<html> <head> <title>Amazon ECS Sample App</title> <style>body {margin-top:
 40px; background-color: #333;} </style> </head><body> <div style=color:white;text-
align:center> <h1>Amazon ECS Sample App</h1> <h2>Congratulations!</h2> <p>Your
 application is now running on a container in Amazon ECS.</p>'; C:\\ServiceMonitor.exe
 w3svc"],
            "entryPoint": [
                "powershell",
                "-Command"
            ],
            "essential": true,
            "cpu": 2048,
            "memory": 4096,
            "image": "mcr.microsoft.com/windows/servercore/iis:windowsservercore-
ltsc2019",
            "name": "sample_windows_app",
            "portMappings": [
                {
                    "hostPort": 80,
                    "containerPort": 80,
                    "protocol": "tcp"
                }
            ]
        }
    ],
    "memory": "4096",
    "cpu": "2048",
    "networkMode": "awsvpc",
    "family": "windows-simple-iis-2019-core",
    "executionRoleArn": "arn:aws:iam::012345678910:role/ecsTaskExecutionRole",
    "runtimePlatform": {"operatingSystemFamily": "WINDOWS_SERVER_2019_CORE"},
    "requiresCompatibilities": ["FARGATE"]
}
```

上述範例 JSON 可以透過 AWS CLI 兩種方式傳遞至 ：您可以將任務定義 JSON 儲存為 檔案，並使用 `--cli-input-json file://`*`path_to_file.json`*選項傳遞。

將 JSON 檔案用於容器定義中：

```
aws ecs register-task-definition --cli-input-json file://$HOME/tasks/fargate-task.json
```

register-task-definition 命令會在完成註冊後傳回任務定義的描述。

步驟 3：列出任務定義

您可以使用 list-task-definitions 命令隨時列出您帳戶的任務定義。這個命令的輸出會顯示 `family` 和 `revision` 值，可在呼叫 run-task 或 start-task 時一起使用。

```
aws ecs list-task-definitions
```

輸出：

```
{
    "taskDefinitionArns": [
        "arn:aws:ecs:region:aws_account_id:task-definition/sample-fargate-windows:1"
    ]
}
```

步驟 4：建立服務

為您的帳戶註冊任務之後，您就可以在您的叢集中建立已註冊任務的服務。在此範例中，您將透過在叢集中執行之 `sample-fargate:1` 任務定義的一個執行個體建立服務。該任務需要網際網路的路由，因此您可透過兩種方法達成。一種方法是，使用透過 NAT 閘道所設定的私有子網路，而該閘道在公有子網路中具有彈性 IP 地址。另一種方法是，使用公有子網路並將公有 IP 地址指派至任務。以下提供這兩種範例。

使用私有子網路的範例。

```
aws ecs create-service --cluster fargate-cluster --service-name fargate-service
 --task-definition sample-fargate-windows:1 --desired-count 1 --launch-type
 "FARGATE" --network-configuration "awsvpcConfiguration={subnets=[subnet-
abcd1234],securityGroups=[sg-abcd1234]}"
```

使用公有子網路的範例。

```
aws ecs create-service --cluster fargate-cluster --service-name fargate-service
 --task-definition sample-fargate-windows:1 --desired-count 1 --launch-type
 "FARGATE" --network-configuration "awsvpcConfiguration={subnets=[subnet-
abcd1234],securityGroups=[sg-abcd1234],assignPublicIp=ENABLED}"
```

create-service 命令會在完成註冊後傳回任務定義的描述。

步驟 5：列出服務

列出您叢集的服務。您應該會看到您在先前一節所建立的服務。您可以記下這個命令傳回的服務名稱或完整的 ARN，稍後將用以描述服務。

```
aws ecs list-services --cluster fargate-cluster
```

輸出：

```
{
    "serviceArns": [
        "arn:aws:ecs:region:aws_account_id:service/fargate-service"
    ]
}
```

步驟 6：描述執行中的服務

使用之前擷取的服務名稱來描述服務，以取得任務的詳細資訊。

```
aws ecs describe-services --cluster fargate-cluster --services fargate-service
```

如果成功，這會傳回服務失敗和服務的描述。例如，在服務區段中，您將可以找到部署的相關資訊，例如執行中或擱置中的任務狀態。您也可以找到任務定義、網路組態和時間戳記事件的相關資訊。在失敗區段中，您將可以找到與呼叫相關聯的失敗 (如果有的話) 的相關資訊。如需故障診斷，請參閱服務事件訊息。如需服務描述的詳細資訊，請參閱描述服務。

```
{
    "services": [
        {
            "status": "ACTIVE",
            "taskDefinition": "arn:aws:ecs:region:aws_account_id:task-definition/
sample-fargate-windows:1",
            "pendingCount": 2,
            "launchType": "FARGATE",
            "loadBalancers": [],
            "roleArn": "arn:aws:iam::aws_account_id:role/aws-service-role/
ecs.amazonaws.com/AWSServiceRoleForECS",
            "placementConstraints": [],
            "createdAt": 1510811361.128,
            "desiredCount": 2,
            "networkConfiguration": {
```

```
                "awsvpcConfiguration": {
                    "subnets": [
                        "subnet-abcd1234"
                    ],
                    "securityGroups": [
                        "sg-abcd1234"
                    ],
                    "assignPublicIp": "DISABLED"
                }
            },
            "platformVersion": "LATEST",
            "serviceName": "fargate-service",
            "clusterArn": "arn:aws:ecs:region:aws_account_id:cluster/fargate-cluster",
            "serviceArn": "arn:aws:ecs:region:aws_account_id:service/fargate-service",
            "deploymentConfiguration": {
                "maximumPercent": 200,
                "minimumHealthyPercent": 100
            },
            "deployments": [
                {
                    "status": "PRIMARY",
                    "networkConfiguration": {
                        "awsvpcConfiguration": {
                            "subnets": [
                                "subnet-abcd1234"
                            ],
                            "securityGroups": [
                                "sg-abcd1234"
                            ],
                            "assignPublicIp": "DISABLED"
                        }
                    },
                    "pendingCount": 2,
                    "launchType": "FARGATE",
                    "createdAt": 1510811361.128,
                    "desiredCount": 2,
                    "taskDefinition": "arn:aws:ecs:region:aws_account_id:task-
definition/sample-fargate-windows:1",
                    "updatedAt": 1510811361.128,
                    "platformVersion": "0.0.1",
                    "id": "ecs-svc/9223370526043414679",
                    "runningCount": 0
                }
            ],
```

```
            "events": [
                {
                    "message": "(service fargate-service) has started 2 tasks: (task
 53c0de40-ea3b-489f-a352-623bf1235f08) (task d0aec985-901b-488f-9fb4-61b991b332a3).",
                    "id": "92b8443e-67fb-4886-880c-07e73383ea83",
                    "createdAt": 1510811841.408
                },
                {
                    "message": "(service fargate-service) has started 2 tasks: (task
 b4911bee-7203-4113-99d4-e89ba457c626) (task cc5853e3-6e2d-4678-8312-74f8a7d76474).",
                    "id": "d85c6ec6-a693-43b3-904a-a997e1fc844d",
                    "createdAt": 1510811601.938
                },
                {
                    "message": "(service fargate-service) has started 2 tasks: (task
 cba86182-52bf-42d7-9df8-b744699e6cfc) (task f4c1ad74-a5c6-4620-90cf-2aff118df5fc).",
                    "id": "095703e1-0ca3-4379-a7c8-c0f1b8b95ace",
                    "createdAt": 1510811364.691
                }
            ],
            "runningCount": 0,
            "placementStrategy": []
        }
    ],
    "failures": []
}
```

步驟 7：清除

完成此教學課程時，建議您清除相關聯的資源，以免未使用的資源產生費用。

刪除服務。

```
aws ecs delete-service --cluster fargate-cluster --service fargate-service --force
```

刪除叢集。

```
aws ecs delete-cluster --cluster fargate-cluster
```

使用 為 EC2 啟動類型建立 Amazon ECS 任務 AWS CLI

以下步驟會協助您使用 AWS CLI在 Amazon ECS 中設定叢集、註冊任務定義、執行任務，以及執行其他常見案例。使用最新版本的 AWS CLI。如需如何升級至最新版本的詳細資訊，請參閱安裝或更新至最新版本的 AWS CLI。

主題

必要條件

本教學課程假設已完成下列先決條件：

- AWS CLI 已安裝並設定最新版本的 。如需安裝或升級 的詳細資訊 AWS CLI，請參閱安裝或更新至最新版本的 AWS CLI。
- 已完成「設定以使用 Amazon ECS。」中的步驟。
- 您的 AWS 使用者具有 IAM AmazonECS_FullAccess 政策範例中指定的必要許可。
- 您已建立 VPC 和安全群組。如需詳細資訊，請參閱the section called “建立 Virtual Private Cloud”。
- （選用）AWS CloudShell 是一種工具，可為客戶提供命令列，而不需要建立自己的 EC2 執行個體。如需詳細資訊，請參閱AWS CloudShell 《使用者指南》中的什麼是 AWS CloudShell？。

步驟 1：建立叢集

當您啟動第一個容器執行個體時，您的帳戶預設會得到 `default` 叢集。

> **Note**
>
> 使用為您提供之 `default` 叢集的優點是，您不必在後續的命令中指定 `--cluster` *`cluster_name`* 選項。如果您建立的是自己的叢集，不是預設叢集，您必須在打算對該叢集使用的每個命令中指定 `--cluster` *`cluster_name`*。

使用下列命令以唯一的名稱建立您自己的叢集：

```
aws ecs create-cluster --cluster-name MyCluster
```

輸出：

```
{
    "cluster": {
        "clusterName": "MyCluster",
        "status": "ACTIVE",
        "clusterArn": "arn:aws:ecs:region:aws_account_id:cluster/MyCluster"
    }
}
```

步驟 2：使用 Amazon ECS AMI 啟動執行個體

您的叢集中必須要有 Amazon ECS 容器執行個體，才能在其中執行任務。如果您的叢集中沒有任何容器執行個體，請參閱 啟動 Amazon ECS Linux 容器執行個體 以取得詳細資訊。

步驟 3：列出容器執行個體

在您啟動容器執行個體後，Amazon ECS 代理會於幾分鐘內在您的預設叢集註冊執行個體。您可以執行下列命令列出叢集中的容器執行個體：

```
aws ecs list-container-instances --cluster default
```

輸出：

```
{
    "containerInstanceArns": [
        "arn:aws:ecs:us-east-1:aws_account_id:container-instance/container_instance_ID"
    ]
}
```

步驟 4：描述您的容器執行個體

在您取得容器執行個體的 ARN 或 ID 之後，您可以使用 describe-container-instances 命令取得執行個體的實用資訊，例如剩餘的和已註冊的 CPU 與記憶體資源。

```
aws ecs describe-container-instances --cluster default --container-
instances container_instance_ID
```

輸出：

```
{
    "failures": [],
    "containerInstances": [
        {
            "status": "ACTIVE",
            "registeredResources": [
                {
                    "integerValue": 1024,
                    "longValue": 0,
                    "type": "INTEGER",
                    "name": "CPU",
                    "doubleValue": 0.0
                },
                {
                    "integerValue": 995,
                    "longValue": 0,
                    "type": "INTEGER",
                    "name": "MEMORY",
                    "doubleValue": 0.0
                },
                {
                    "name": "PORTS",
                    "longValue": 0,
                    "doubleValue": 0.0,
                    "stringSetValue": [
                        "22",
                        "2376",
                        "2375",
                        "51678"
                    ],
                    "type": "STRINGSET",
                    "integerValue": 0
                },
```

```
                {
                    "name": "PORTS_UDP",
                    "longValue": 0,
                    "doubleValue": 0.0,
                    "stringSetValue": [],
                    "type": "STRINGSET",
                    "integerValue": 0
                }
            ],
            "ec2InstanceId": "instance_id",
            "agentConnected": true,
            "containerInstanceArn": "arn:aws:ecs:us-west-2:aws_account_id:container-
instance/container_instance_ID",
            "pendingTasksCount": 0,
            "remainingResources": [
                {
                    "integerValue": 1024,
                    "longValue": 0,
                    "type": "INTEGER",
                    "name": "CPU",
                    "doubleValue": 0.0
                },
                {
                    "integerValue": 995,
                    "longValue": 0,
                    "type": "INTEGER",
                    "name": "MEMORY",
                    "doubleValue": 0.0
                },
                {
                    "name": "PORTS",
                    "longValue": 0,
                    "doubleValue": 0.0,
                    "stringSetValue": [
                        "22",
                        "2376",
                        "2375",
                        "51678"
                    ],
                    "type": "STRINGSET",
                    "integerValue": 0
                },
                {
                    "name": "PORTS_UDP",
```

```
                    "longValue": 0,
                    "doubleValue": 0.0,
                    "stringSetValue": [],
                    "type": "STRINGSET",
                    "integerValue": 0
                }
            ],
            "runningTasksCount": 0,
            "attributes": [
                {
                    "name": "com.amazonaws.ecs.capability.privileged-container"
                },
                {
                    "name": "com.amazonaws.ecs.capability.docker-remote-api.1.17"
                },
                {
                    "name": "com.amazonaws.ecs.capability.docker-remote-api.1.18"
                },
                {
                    "name": "com.amazonaws.ecs.capability.docker-remote-api.1.19"
                },
                {
                    "name": "com.amazonaws.ecs.capability.logging-driver.json-file"
                },
                {
                    "name": "com.amazonaws.ecs.capability.logging-driver.syslog"
                }
            ],
            "versionInfo": {
                "agentVersion": "1.5.0",
                "agentHash": "b197edd",
                "dockerVersion": "DockerVersion: 1.7.1"
            }
        }
    ]
}
```

您也可以找到 Amazon EC2 執行個體 ID，用來在 Amazon EC2 主控台中監控執行個體，或搭配 aws ec2 describe-instances --instance-id ***instance_id*** 命令使用。

步驟 5：註冊任務定義

您必須先註冊任務定義，才能在您的 ECS 叢集中執行任務。任務定義是分在一組的容器清單。以下範例是一種簡單的任務定義，使用 Docker Hub 的 busybox 映像，只睡眠 360 秒。如需可用之任務定義參數的詳細資訊，請參閱「Amazon ECS 任務定義」。

```
{
    "containerDefinitions": [
        {
            "name": "sleep",
            "image": "busybox",
            "cpu": 10,
            "command": [
                "sleep",
                "360"
            ],
            "memory": 10,
            "essential": true
        }
    ],
    "family": "sleep360"
}
```

上述範例 JSON 可以透過 AWS CLI 兩種方式傳遞至 ：您可以將任務定義 JSON 儲存為 檔案，並使用 `--cli-input-json file://path_to_file.json`選項傳遞。或者，您可以逸出 JSON 中的引號，然後在命令行上傳遞 JSON 容器定義，如下範例所示。如果您選擇用命令列傳送容器定義，您的命令需要使用額外的 `--family` 參數，以保留多個彼此相關聯的任務定義版本。

將 JSON 檔案用於容器定義中：

```
aws ecs register-task-definition --cli-input-json file://$HOME/tasks/sleep360.json
```

將 JSON 字串用於容器定義中：

```
aws ecs register-task-definition --family sleep360 --container-definitions "[{\"name\":\"sleep\",\"image\":\"busybox\",\"cpu\":10,\"command\":[\"sleep\",\"360\"],\"memory\":10,\"essential\":true}]"
```

register-task-definition 會在完成註冊後，傳回任務定義的描述。

```
{
```

```
    "taskDefinition": {
        "volumes": [],
        "taskDefinitionArn": "arn:aws:ec2:us-east-1:aws_account_id:task-definition/
sleep360:1",
        "containerDefinitions": [
            {
                "environment": [],
                "name": "sleep",
                "mountPoints": [],
                "image": "busybox",
                "cpu": 10,
                "portMappings": [],
                "command": [
                    "sleep",
                    "360"
                ],
                "memory": 10,
                "essential": true,
                "volumesFrom": []
            }
        ],
        "family": "sleep360",
        "revision": 1
    }
}
```

步驟 6：列出任務定義

您可以使用 list-task-definitions 命令隨時列出您帳戶的任務定義。這個命令的輸出會顯示 `family` 和 `revision` 值，可在呼叫 run-task 或 start-task 時一起使用。

```
aws ecs list-task-definitions
```

輸出：

```
{
    "taskDefinitionArns": [
        "arn:aws:ec2:us-east-1:aws_account_id:task-definition/sleep300:1",
        "arn:aws:ec2:us-east-1:aws_account_id:task-definition/sleep300:2",
        "arn:aws:ec2:us-east-1:aws_account_id:task-definition/sleep360:1",
        "arn:aws:ec2:us-east-1:aws_account_id:task-definition/wordpress:3",
        "arn:aws:ec2:us-east-1:aws_account_id:task-definition/wordpress:4",
```

```
        "arn:aws:ec2:us-east-1:aws_account_id:task-definition/wordpress:5",
        "arn:aws:ec2:us-east-1:aws_account_id:task-definition/wordpress:6"
    ]
}
```

步驟 7：執行任務

在您註冊帳戶任務，並啟動已向叢集註冊的容器執行個體之後，您就可以在叢集中執行註冊的任務。在此範例中，您會在您的預設叢集中放置 sleep360:1 任務定義的單一執行個體。

```
aws ecs run-task --cluster default --task-definition sleep360:1 --count 1
```

輸出：

```
{
    "tasks": [
        {
            "taskArn": "arn:aws:ecs:us-east-1:aws_account_id:task/task_ID",
            "overrides": {
                "containerOverrides": [
                    {
                        "name": "sleep"
                    }
                ]
            },
            "lastStatus": "PENDING",
            "containerInstanceArn": "arn:aws:ecs:us-east-1:aws_account_id:container-
instance/container_instance_ID",
            "clusterArn": "arn:aws:ecs:us-east-1:aws_account_id:cluster/default",
            "desiredStatus": "RUNNING",
            "taskDefinitionArn": "arn:aws:ecs:us-east-1:aws_account_id:task-definition/
sleep360:1",
            "containers": [
                {
                    "containerArn": "arn:aws:ecs:us-
east-1:aws_account_id:container/container_ID",
                    "taskArn": "arn:aws:ecs:us-east-1:aws_account_id:task/task_ID",
                    "lastStatus": "PENDING",
                    "name": "sleep"
                }
            ]
        }
```

```
    ]
}
```

步驟 8：列出任務

列出您叢集的任務。您應該會看到您在先前一節所執行的任務。您可以記下這個命令傳回的任務 ID 或完整的 ARN，稍後用以描述任務。

```
aws ecs list-tasks --cluster default
```

輸出：

```
{
    "taskArns": [
        "arn:aws:ecs:us-east-1:aws_account_id:task/task_ID"
    ]
}
```

步驟 9：描述執行中的任務

使用之前擷取的任務 ID 來描述任務，以取得任務的詳細資訊。

```
aws ecs describe-tasks --cluster default --task task_ID
```

輸出：

```
{
    "failures": [],
    "tasks": [
        {
            "taskArn": "arn:aws:ecs:us-east-1:aws_account_id:task/task_ID",
            "overrides": {
                "containerOverrides": [
                    {
                        "name": "sleep"
                    }
                ]
            },
            "lastStatus": "RUNNING",
```

```
            "containerInstanceArn": "arn:aws:ecs:us-east-1:aws_account_id:container-
instance/container_instance_ID",
            "clusterArn": "arn:aws:ecs:us-east-1:aws_account_id:cluster/default",
            "desiredStatus": "RUNNING",
            "taskDefinitionArn": "arn:aws:ecs:us-east-1:aws_account_id:task-definition/
sleep360:1",
            "containers": [
                {
                    "containerArn": "arn:aws:ecs:us-
east-1:aws_account_id:container/container_ID",
                    "taskArn": "arn:aws:ecs:us-east-1:aws_account_id:task/task_ID",
                    "lastStatus": "RUNNING",
                    "name": "sleep",
                    "networkBindings": []
                }
            ]
        }
    ]
}
```

設定 Amazon ECS 接聽 CloudWatch Events 事件

了解如何設定簡單的 Lambda 函數，以接聽任務事件並將其寫入 CloudWatch Logs 日誌串流。

先決條件：設定測試叢集

如果您沒有可供擷取事件的執行中叢集，請遵循「the section called “為 Fargate 啟動類型建立叢集”」中的步驟建立叢集。在此教學課程的最後，您在此叢集上執行任務，以測試您的 Lambda 函數設定是否正確。

步驟 1：建立 Lambda 函數

在此程序中，您要建立一個簡單的 Lambda 函數，做為 Amazon ECS 事件資料流訊息的目標。

1. 在 https：//https://console.aws.amazon.com/lambda/ 開啟 AWS Lambda 主控台。
2. 選擇 Create function (建立函數)。
3. 在 Author from scratch (從頭開始撰寫) 畫面上，執行下列操作：
    a. 對於 Name (名稱)，輸入值。
    b. 針對 Runtime (執行時間)，選擇您的 Python 版本，例如：Python 3.9。

c. 針對 Role (角色)，選擇 Create a new role with basic Lambda permissions (建立具備基本 Lambda 許可的新角色)。

4. 選擇建立函數。
5. 在 Function code (函數程式碼) 區段中，編輯範本程式碼以符合下列範例：

```
import json

def lambda_handler(event, context):
    if event["source"] != "aws.ecs":
       raise ValueError("Function only supports input from events with a source
 type of: aws.ecs")

    print('Here is the event:')
    print(json.dumps(event))
```

這是一個簡單的 Python 3.9 函數，此函數列印 Amazon ECS 所傳送的事件。如果一切設定正確，則在此教學課程的最後，您會看到事件詳細資訊出現在與此 Lambda 函數建立關聯的 CloudWatch Logs 日誌串流中。

6. 選擇 Save (儲存)。

步驟 2：註冊事件規則

接著，您會建立一個 CloudWatch Events 事件規則，擷取來自 Amazon ECS 叢集的任務事件。此規則會擷取來自所定義的帳戶內所有叢集的所有事件。任務訊息本身包含事件來源的資訊 (包含其所在的叢集)，從而能以程式設計方式篩選和排序事件。

> **Note**
>
> 當您使用 AWS Management Console 建立事件規則時，主控台會自動新增必要的 IAM 許可，以授予 CloudWatch Events 呼叫 Lambda 函數的許可。如果您使用 建立事件規則 AWS CLI，則需要明確授予此許可。如需詳細資訊，請參閱《Amazon EventBridge 使用者指南》中的 Amazon EventBridge 和 Amazon EventBridge 事件模式中的事件。 EventBridge

將事件路由至 Lambda 函數

1. 透過 https://console.aws.amazon.com/cloudwatch/ 開啟 CloudWatch 主控台。
2. 在導覽窗格上，選擇 Events (事件)、Rules (規則)、Create rule (建立規則)。

3. 針對 Event Source (事件來源)，選擇 ECS 做為事件來源。根據預設，此規則套用至所有 Amazon ECS 群組的 Amazon ECS 事件。或者，您可以選取特定事件或特定 Amazon ECS 群組。
4. 對於 Targets (目標)，選擇 Add target (新增目標)，並對於 Target type (目標類型)，選擇 Lambda function (Lambda 函數)，然後選取您的 Lambda 函數。
5. 選擇設定詳細資訊。
6. 針對 Rule definition (規則定義)，輸入您規則的名稱和描述，然後選擇 Create rule (建立規則)。

步驟 3：建立任務定義

建立任務定義。

1. 開啟主控台，網址為 https://console.aws.amazon.com/ecs/v2。
2. 在導覽窗格中，選擇 Task Definitions (任務定義)。
3. 選擇 Create new Task Definition (建立新任務定義)，以及 Create new revision with JSON (使用 JSON 建立新修訂版)。
4. 複製下列任務定義範例並貼到方塊中，然後選擇 Save (儲存)。

```
{
   "containerDefinitions": [
      {
         "command": [
            "/bin/sh -c \"echo '<html> <head> <title>Amazon ECS Sample App</
title> <style>body {margin-top: 40px; background-color: #333;} </style> </
head><body> <div style=color:white;text-align:center> <h1>Amazon ECS Sample App</
h1> <h2>Congratulations!</h2> <p>Your application is now running on a container in
 Amazon ECS.</p> </div></body></html>' >  /usr/local/apache2/htdocs/index.html &&
 httpd-foreground\""
         ],
         "entryPoint": [
            "sh",
            "-c"
         ],
         "essential": true,
         "image": "httpd:2.4",
         "logConfiguration": {
            "logDriver": "awslogs",
            "options": {
               "awslogs-group" : "/ecs/fargate-task-definition",
               "awslogs-region": "us-east-1",
```

```
                "awslogs-stream-prefix": "ecs"
            }
         },
         "name": "sample-fargate-app",
         "portMappings": [
            {
               "containerPort": 80,
               "hostPort": 80,
               "protocol": "tcp"
            }
         ]
      }
   ],
   "cpu": "256",
   "executionRoleArn": "arn:aws:iam::012345678910:role/ecsTaskExecutionRole",
   "family": "fargate-task-definition",
   "memory": "512",
   "networkMode": "awsvpc",
   "runtimePlatform": {
        "operatingSystemFamily": "LINUX"
    },
   "requiresCompatibilities": [
       "FARGATE"
    ]
}
```

5. 選擇 Create (建立)。

步驟 4：測試您的規則

最後，您會建立一個 CloudWatch Events 事件規則，擷取來自 Amazon ECS 叢集的任務事件。此規則會擷取來自所定義的帳戶內所有叢集的所有事件。任務訊息本身包含事件來源的資訊 (包含其所在的叢集)，從而能以程式設計方式篩選和排序事件。

測試規則

1. 開啟主控台，網址為 https://console.aws.amazon.com/ecs/v2。
2. 選擇 Task definitions (任務定義)。
3. 選擇 console-sample-app-static，然後選擇 Deploy (部署)、Run new task (執行新任務)。
4. 在 Cluster (叢集) 欄位中選擇預設，然後選擇 Deploy (部署)。
5. 透過 https://console.aws.amazon.com/cloudwatch/ 開啟 CloudWatch 主控台。

6. 在導覽窗格上，選擇 Logs (日誌)，然後選取 Lambda 函數的日誌群組 (例如，/aws/lambda/*my-function*)。
7. 選取日誌串流，以檢視事件資料。

傳送 Amazon ECS 任務已停止事件的 Amazon Simple Notification Service 提醒

設定 Amazon EventBridge 事件規則，該規則只會擷取任務因其中一個必要容器終止而停止執行的任務事件。事件只會將具有特定 `stoppedReason` 屬性的任務事件傳送至指定的 Amazon SNS 主題。

先決條件：設定測試叢集

如果您沒有可供擷取事件的執行中叢集，請遵循使用 AWS Fargate上的 Linux 容器來開始使用主控台中的步驟建立叢集。在此教學課程的最後，您會在此叢集上執行任務，以測試您的 Amazon SNS 主題和 EventBridge 規則設定是否正確。

先決條件：設定 Amazon SNS 的許可

若要允許 EventBridge 發佈到 Amazon SNS 主題，請使用 aws sns get-topic-attributes 和 aws sns set-topic-attributes 命令。

如需有關如何建立許可的詳細資訊，請參閱《Amazon Simple Notification Service 開發人員指南》中的 Amazon SNS 許可。

新增下列許可：

```
{
  "Sid": "PublishEventsToMyTopic",
  "Effect": "Allow",
  "Principal": {
     "Service": "events.amazonaws.com"
  },
  "Action": "sns: Publish",
  "Resource": "arn:aws:sns:region:account-id:TaskStoppedAlert",
}
```

步驟 1：建立並訂閱 Amazon SNS 主題

在此教學課程中，您會設定 Amazon SNS 主題，做為新事件規則的事件目標。

如需有關如何建立和訂閱 Amazon SNS 主題的資訊，請參閱《Amazon Simple Notification Service 開發人員指南》中的 Amazon SNS 入門，並使用下表來決定要選取的選項。

選項	Value
Type	標準
名稱	TaskStoppedAlert
通訊協定	電子郵件
端點	您目前能存取的電子郵件地址

步驟 2：註冊事件規則

接下來，您會登錄事件規則，只擷取具有停止容器之任務的任務停止事件。

如需有關如何建立和訂閱 Amazon SNS 主題的資訊，請參閱《Amazon EventBridge 使用者指南》中的在 Amazon EventBridge 中建立規則，並使用下表來決定要選取的選項。

選項	Value
規則類型	具有事件模式的規則
事件來源	AWS 事件或 EventBridge 合作夥伴事件
事件模式	自訂模式 (JSON 編輯器)
事件模式	<pre>{ "source":["aws.ecs"], "detail-type":["ECS Task State Change"], "detail":{ "lastStatus":["STOPPED"</pre>

選項	Value
	<pre>], "stoppedReason":["Essentia l container in task exited"] } }</pre>
Target type (目標類型)	AWS 服務
目標	SNS 主題
主題	TaskStoppedAlert (您在步驟 1 中建立的主題)

步驟 3：測試您的規則

藉由執行在啟動後立即結束的任務，確認規則是否正常運作。如果您的事件規則設定正確，您應該會在幾分鐘內收到一封包含事件文字的電子郵件訊息。如果您有可以滿足規則需求的現有任務定義，請使用它來執行任務。如果沒有，以下步驟將引導您註冊 Fargate 任務定義並執行它。

1. 開啟主控台，網址為 https://console.aws.amazon.com/ecs/v2。
2. 在導覽窗格中，選擇 Task Definitions (任務定義)。
3. 選擇 Create new task definitio (建立新任務定義)、Create new task definition with JSON (使用 JSON 建立新的任務定義)。
4. 在 JSON 編輯工具方塊中，編輯您的 JSON 檔案，將下列內容複製至編輯器。

```
{
   "containerDefinitions":[
      {
         "command":[
            "sh",
            "-c",
            "sleep 5"
         ],
         "essential":true,
```

```
            "image":"amazonlinux:2",
            "name":"test-sleep"
         }
      ],
      "cpu":"256",
      "executionRoleArn":"arn:aws:iam::012345678910:role/ecsTaskExecutionRole",
      "family":"fargate-task-definition",
      "memory":"512",
      "networkMode":"awsvpc",
      "requiresCompatibilities":[
         "FARGATE"
      ]
}
```

5. 選擇 Create (建立)。

從主控台執行任務

1. 開啟主控台，網址為 https://console.aws.amazon.com/ecs/v2。
2. 在叢集頁面，選擇您要在先決條件中建立的叢集。
3. 在 Tasks (任務) 標籤中，選擇 Run new task (執行新任務)。
4. 針對 Application type (應用程式類型)，選擇 Task (任務)。
5. 在任務定義中，選擇 fargate-task-definition。
6. 針對 Desired tasks (所需任務)，輸入要啟動的任務數目。
7. 選擇 Create (建立)。

串連多行或堆疊追蹤 Amazon ECS 日誌訊息

從 AWS 適用於 Fluent Bit 2.22.0 版的 開始，包含多行篩選條件。多行篩選條件有助於串連最初屬於一筆內容但分割到多筆記錄或日誌明細行的日誌訊息。如需有關多行篩選條件的詳細資訊，請參閱《Fluent Bit 說明文件》。

分割日誌訊息的常見範例有：

- 堆疊追蹤。
- 在多行上列印日誌的應用程式。
- 因為日誌訊息長於指定的執行時間緩衝大小上限而分割的日誌訊息。您可以按照 GitHub 上的範例來串連由容器執行時間分割的日誌訊息：FireLens 範例：串聯部分/分割容器日誌。

所需的 IAM 許可

請務必先確定您擁有必要的 IAM 許可，讓容器代理程式可從 Amazon ECR 中提取容器映像，也讓容器可將日誌路由到 CloudWatch Logs。

對於這些許可，您必須具有下列角色：

- 任務 IAM 角色。
- 任務執行 IAM 角色。

若要使用 JSON 政策編輯器來建立政策

1. 登入 AWS Management Console ，並在 https：//https://console.aws.amazon.com/iam/ 開啟 IAM 主控台。
2. 在左側的導覽窗格中，選擇 Policies (政策)。

   如果這是您第一次選擇 Policies (政策)，將會顯示 Welcome to Managed Policies (歡迎使用受管政策) 頁面。選擇 Get Started (開始使用)。
3. 在頁面頂端，選擇 Create policy (建立政策)。
4. 在政策編輯器中，選擇 JSON 選項。
5. 輸入下列 JSON 政策文件：

```
{
 "Version": "2012-10-17",
 "Statement": [{
  "Effect": "Allow",
  "Action": [
   "logs:CreateLogStream",
   "logs:CreateLogGroup",
   "logs:PutLogEvents"
  ],
  "Resource": "*"
 }]
}
```

6. 選擇 Next (下一步)。

> **Note**
>
> 您可以隨時切換視覺化與 JSON 編輯器選項。不過，如果您進行變更或在視覺化編輯器中選擇下一步，IAM 就可能會調整您的政策結構，以便針對視覺化編輯器進行最佳化。如需詳細資訊，請參閱 IAM 使用者指南中的調整政策結構。

7. 在檢視與建立頁面上，為您正在建立的政策輸入政策名稱與描述 (選用)。檢視此政策中定義的許可，來查看您的政策所授予的許可。
8. 選擇 Create policy (建立政策) 儲存您的新政策。

確定何時使用多行日誌設定

以下是您在 CloudWatch Logs 主控台中看到的範例日誌片段，其中包含預設日誌設定。您可以查看以 `log` 開頭的行，以確定是否需要多行篩選條件。當內容相同時，您可以使用多行日誌設定，在此範例中，內容為「com.myproject.model.MyProject」。

```
2022-09-20T15:47:56:595-05-00                              {"container_id":
 "82ba37cada1d44d389b03e78caf74faa-EXAMPLE", "container_name": "example-
app", "source=": "stdout", "log": ": "     at com.myproject.modele.
(MyProject.badMethod.java:22)",
    {
      "container_id":  "82ba37cada1d44d389b03e78caf74faa-EXAMPLE",
      "container_name: ": "example-app",
      "source": "stdout",
      "log": ": "     at com.myproject.model.MyProject.badMethod(MyProject.java:22)",
      "ecs_cluster": "default",
      "ecs_task_arn": "arn:aws:region:123456789012:task/default/
b23c940d29ed4714971cba72cEXAMPLE",
      "ecs_task_definition": "firelense-example-multiline:3"
     }
```

```
2022-09-20T15:47:56:595-05-00                              {"container_id":
 "82ba37cada1d44d389b03e78caf74faa-EXAMPLE", "container_name": "example-app", "stdout",
 "log": ": "     at com.myproject.modele.(MyProject.oneMoreMethod.java:18)",
    {
      "container_id":  "82ba37cada1d44d389b03e78caf74faa-EXAMPLE",
      "container_name: ": "example-app",
      "source": "stdout",
```

```
      "log": ": "     at
 com.myproject.model.MyProject.oneMoreMethod(MyProject.java:18)",
      "ecs_cluster": "default",
      "ecs_task_arn": "arn:aws:region:123456789012:task/default/
b23c940d29ed4714971cba72cEXAMPLE,
      "ecs_task_definition": "firelense-example-multiline:3"
     }
```

使用多行日誌設定後，輸出看起來類似下列範例。

```
2022-09-20T15:47:56:595-05-00                              {"container_id":
 "82ba37cada1d44d389b03e78caf74faa-EXAMPLE", "container_name": "example-app",
 "stdout",...
    {
      "container_id":  "82ba37cada1d44d389b03e78caf74faa-EXAMPLE",
      "container_name: ": "example-app",
      "source": "stdout",
      "log:    "September 20, 2022 06:41:48 Exception in thread \"main\"
 java.lang.RuntimeException: Something has gone wrong, aborting!\n
    at com.myproject.module.MyProject.badMethod(MyProject.java:22)\n    at
    at com.myproject.model.MyProject.oneMoreMethod(MyProject.java:18)
 com.myproject.module.MyProject.main(MyProject.java:6)",
      "ecs_cluster": "default",
      "ecs_task_arn": "arn:aws:region:123456789012:task/default/
b23c940d29ed4714971cba72cEXAMPLE",
      "ecs_task_definition": "firelense-example-multiline:2"
     }
```

剖析並串連選項

若要剖析日誌並串連因新行而分割的行數，您可以使用這兩個選項中的任一選項。

- 使用您自己的剖析器檔案 (包含要剖析的規則) 並串連屬於相同訊息的明細行。
- 使用 Fluent Bit 內建剖析器。如需 Fluent Bit 內建剖析器所支援的語言清單，請參閱 Fluent Bit 說明文件。

下列教學會引導您逐步演練每個使用案例的步驟。這些步驟向您展示如何串聯多行並將日誌傳送到 Amazon CloudWatch。您可以為您的日誌指定其他目的地。

範例：使用您建立的剖析器

在本範例中，您會完成下列步驟：

1. 建置並上傳 Fluent Bit 容器的映像。
2. 建置並上傳可執行、失敗並產生多行堆疊追蹤之試用版多行應用程式的映像。
3. 建立任務定義並執行任務。
4. 檢視日誌以確認跨多行的訊息是否顯示為串連。

建置並上傳 Fluent Bit 容器的映像

此映像將包含您指定常規運算式的剖析器檔案和參照剖析器檔案的組態文件。

1. 建立名稱為 `FluentBitDockerImage` 的資料夾。
2. 在資料夾內，建立剖析器檔案 (包含要剖析日誌的規則) 並串連屬於相同訊息的明細行。

   a. 將以下內容貼到該剖析器檔案：

```
[MULTILINE_PARSER]
    name          multiline-regex-test
    type          regex
    flush_timeout 1000
    #
    # Regex rules for multiline parsing
    # ---------------------------------
    #
    # configuration hints:
    #
    #  - first state always has the name: start_state
    #  - every field in the rule must be inside double quotes
    #
    # rules |   state name  | regex pattern                  | next state
    # ------|---------------|--------------------------------------------
    rule      "start_state"   "/(Dec \d+ \d+\:\d+\:\d+)(.*)/"  "cont"
    rule      "cont"          "/^\s+at.*/"                     "cont"
```

   當您自訂 Regex 運算式模式時，我們建議您使用常規運算式編輯器來測試運算式。

   b. 儲存檔案為 `parsers_multiline.conf`。

3. 在 FluentBitDockerImage 資料夾內，建立參照了您在前一個步驟中建立的剖析器檔案之自訂組態檔案。

   如需有關自訂組態檔案的詳細資訊，請參閱《Amazon Elastic Container Service 開發人員指南》中的指定自訂組態檔案

   a. 將以下內容貼到該檔案：

   ```
   [SERVICE]
       flush                 1
       log_level             info
       parsers_file          /parsers_multiline.conf

   [FILTER]
       name                  multiline
       match                 *
       multiline.key_content log
       multiline.parser      multiline-regex-test
   ```

   > **Note**
   >
   > 您必須使用剖析器的絕對路徑。

   b. 儲存檔案為 extra.conf。

4. 在 FluentBitDockerImage 資料夾內，建立具有 Fluent Bit 映像與剖析器和您建立的組態檔案之 Dockerfile。

   a. 將以下內容貼到該檔案：

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

   ADD parsers_multiline.conf /parsers_multiline.conf
   ADD extra.conf /extra.conf
   ```

   b. 儲存檔案為 Dockerfile。

5. 使用 Dockerfile，建置內含剖析器與自訂組態檔案的自訂 Fluent Bit 映像。

> **Note**
>
> 您可以將剖析器檔案和組態檔案放置在 Docker 影像中除了 `/fluent-bit/etc/fluent-bit.conf` 以外的任何位置，因為該檔案路徑會由 FireLens 使用。

a. 建置映像：`docker build -t fluent-bit-multiline-image .`

   其中：`fluent-bit-multiline-image` 是此範例中映像的名稱。

b. 驗證映像已正確建立：`docker images –filter reference=fluent-bit-multiline-image`

   如果成功，輸出會顯示映像和 `latest` 標籤。

6. 將自訂 Fluent Bit 映像上傳至 Amazon Elastic 容器登錄檔。

   a. 建立 Amazon ECR 儲存庫，以便存放映像：`aws ecr create-repository --repository-name fluent-bit-multiline-repo --region us-east-1`

      其中：`fluent-bit-multiline-repo` 是此範例中儲存庫的名稱，`us-east-1` 是此範例中的區域。

      輸出為您提供了新儲存庫的詳細資訊。

   b. 使用來自先前輸出的 `repositoryUri` 值，標記您的映像：`docker tag *fluent-bit-multiline-image repositoryUri*`

      範例：`docker tag fluent-bit-multiline-image xxxxxxxxxxxx.dkr.ecr.us-east-1.amazonaws.com/fluent-bit-multiline-repo`

   c. 執行 Docker 影像以確認是否能正確執行：`docker images –filter reference=*repositoryUri*`

      在輸出中，儲存庫名稱從 fluent-bit-multiline-repo 變更為 `repositoryUri`。

   d. 透過執行 `aws ecr get-login-password` 命令並指定您要驗證的登錄 ID 來驗證 Amazon ECR：`aws ecr get-login-password | docker login --username AWS --password-stdin *registry ID*.dkr.ecr.*region*.amazonaws.com`

      範例：`ecr get-login-password | docker login --username AWS --password-stdin xxxxxxxxxxxx.dkr.ecr.us-east-1.amazonaws.com`

畫面上會顯示成功登入訊息。

e. 將映像推送至 Amazon ECR：docker push *registry ID*.dkr.ecr.*region*.amazonaws.com/*repository name*

範例：docker push xxxxxxxxxxxx.dkr.ecr.us-east-1.amazonaws.com/fluent-bit-multiline-repo

建置並上傳試用版多行應用程式的映像

此映像將包含執行應用程式的 Python 指令碼檔案和樣本日誌檔案。

當您執行任務時，應用程式會模擬執行，然後失敗並建立堆疊追蹤。

1. 建立名為 multiline-app 的資料夾：mkdir multiline-app
2. 建立 Python 指令碼檔案。
   a. 在 multiline-app 資料夾中，建立檔案，並將它命名為 main.py。
   b. 將以下內容貼到該檔案：

```
import os
import time
file1 = open('/test.log', 'r')
Lines = file1.readlines()

count = 0

for i in range(10):
    print("app running normally...")
    time.sleep(1)

# Strips the newline character
for line in Lines:
    count += 1
    print(line.rstrip())
print(count)
print("app terminated.")
```

   c. 儲存 main.py 檔案。
3. 建立範例日誌檔案。

a. 在 multiline-app 資料夾中，建立檔案，並將它命名為 test.log。

b. 將以下內容貼到該檔案：

```
single line...
Dec 14 06:41:08 Exception in thread "main" java.lang.RuntimeException:
 Something has gone wrong, aborting!
    at com.myproject.module.MyProject.badMethod(MyProject.java:22)
    at com.myproject.module.MyProject.oneMoreMethod(MyProject.java:18)
    at com.myproject.module.MyProject.anotherMethod(MyProject.java:14)
    at com.myproject.module.MyProject.someMethod(MyProject.java:10)
    at com.myproject.module.MyProject.main(MyProject.java:6)
another line...
```

c. 儲存 test.log 檔案。

4. 在 multiline-app 資料夾內，建立 Dockerfile。

a. 將以下內容貼到該檔案：

```
FROM public.ecr.aws/amazonlinux/amazonlinux:latest
ADD test.log /test.log

RUN yum upgrade -y && yum install -y python3

WORKDIR /usr/local/bin

COPY main.py .

CMD ["python3", "main.py"]
```

b. 儲存 Dockerfile 檔案。

5. 使用 Dockerfile，建置映像。

a. 建置映像：docker build -t multiline-app-image .

其中：multiline-app-image 是此範例中映像的名稱。

b. 驗證映像已正確建立：docker images –filter reference=multiline-app-image

如果成功，輸出會顯示映像和 latest 標籤。

6. 將映像上傳至 Amazon Elastic 容器登錄檔。

a. 建立 Amazon ECR 儲存庫，以便存放映像：`aws ecr create-repository --repository-name multiline-app-repo --region us-east-1`

   其中：`multiline-app-repo` 是此範例中儲存庫的名稱，`us-east-1` 是此範例中的區域。

   輸出為您提供了新儲存庫的詳細資訊。請記下 `repositoryUri` 值，因為您將在後續步驟用到它。

b. 使用來自先前輸出的 `repositoryUri` 值，標記您的映像：`docker tag multiline-app-image repositoryUri`

   範例：`docker tag multiline-app-image xxxxxxxxxxxx.dkr.ecr.us-east-1.amazonaws.com/multiline-app-repo`

c. 執行 Docker 影像以確認是否能正確執行：`docker images –filter reference=repositoryUri`

   在輸出中，儲存庫名稱會從 `multiline-app-repo` 變更為 `repositoryUri` 值。

d. 將映像推送至 Amazon ECR：`docker push aws_account_id.dkr.ecr.region.amazonaws.com/repository name`

   範例：`docker push xxxxxxxxxxxx.dkr.ecr.us-east-1.amazonaws.com/multiline-app-repo`

建立任務定義並執行任務

1. 建立檔案名稱為 `multiline-task-definition.json` 的任務定義檔案。
2. 將以下內容貼到該 `multiline-task-definition.json` 檔案：

```
{
    "family": "firelens-example-multiline",
    "taskRoleArn": "task role ARN",
    "executionRoleArn": "execution role ARN",
    "containerDefinitions": [
        {
            "essential": true,
            "image": "aws_account_id.dkr.ecr.us-east-1.amazonaws.com/fluent-bit-multiline-image:latest",
            "name": "log_router",
            "firelensConfiguration": {
                "type": "fluentbit",
```

```
                "options": {
                    "config-file-type": "file",
                    "config-file-value": "/extra.conf"
                }
            },
            "memoryReservation": 50
        },
        {
            "essential": true,
            "image": "aws_account_id.dkr.ecr.us-east-1.amazonaws.com/multiline-app-
image:latest",
            "name": "app",
            "logConfiguration": {
                "logDriver": "awsfirelens",
                "options": {
                    "Name": "cloudwatch_logs",
                    "region": "us-east-1",
                    "log_group_name": "multiline-test/application",
                    "auto_create_group": "true",
                    "log_stream_prefix": "multiline-"
                }
            },
            "memoryReservation": 100
        }
    ],
    "requiresCompatibilities": ["FARGATE"],
    "networkMode": "awsvpc",
    "cpu": "256",
    "memory": "512"
}
```

取代下列在 `multiline-task-definition.json` 任務定義中的項目：

a. *`task role ARN`*

若要尋找任務角色 ARN，請前往 IAM 主控台。選擇 Roles (角色) 並尋找您建立的 `ecs-task-role-for-firelens` 任務角色。選擇角色並複製顯示於 Summary (摘要) 區段的 ARN。

b. *`execution role ARN`*

若要尋找執行角色 ARN，請前往 IAM 主控台。選擇 Roles (角色) 並尋找 `ecsTaskExecutionRole` 角色。選擇角色並複製顯示於 Summary (摘要) 區段的 ARN。

c. *aws_account_id*

若要尋找您的 aws_account_id，請登入 AWS Management Console。選擇右上角的使用者名稱並複製您的帳戶 ID。

d. *us-east-1*

如有必要，請取代該區域。

3. 註冊任務定義檔案：`aws ecs register-task-definition --cli-input-json file://multiline-task-definition.json --region region`
4. 開啟主控台，網址為 https://console.aws.amazon.com/ecs/v2。
5. 在導覽窗格中，選擇 Task Definitions (任務定義)，然後選擇 firelens-example-multiline 系列，因為我們已在上述任務定義的第一行中將任務定義註冊到此系列。
6. 選擇最新版本。
7. 選擇部署、執行任務。
8. 在執行任務頁面上，針對叢集選擇叢集，然後在聯網下，針對子網路選擇任務的可用子網路。
9. 選擇建立。

確認 Amazon CloudWatch 中的多行日誌訊息是否顯示為串連

1. 透過 https://console.aws.amazon.com/cloudwatch/ 開啟 CloudWatch 主控台。
2. 在導覽窗格中，展開 Logs (日誌) 並選擇 Log groups (日誌群組)。
3. 選擇 multiline-test/applicatio 日誌群組。
4. 選擇日誌。檢視訊息。系統會串連與剖析器檔案中的規則相符之明細行，並顯示為單則訊息。

下列日誌程式碼片段顯示了串連到單一 Java 堆疊追蹤事件的明細行：

```
{
    "container_id": "xxxxxx",
    "container_name": "app",
    "source": "stdout",
    "log": "Dec 14 06:41:08 Exception in thread \"main\"
 java.lang.RuntimeException: Something has gone wrong, aborting!\n
 at com.myproject.module.MyProject.badMethod(MyProject.java:22)\n    at
 com.myproject.module.MyProject.oneMoreMethod(MyProject.java:18)\n
 at com.myproject.module.MyProject.anotherMethod(MyProject.java:14)\n
 at com.myproject.module.MyProject.someMethod(MyProject.java:10)\n    at
 com.myproject.module.MyProject.main(MyProject.java:6)",
```

```
    "ecs_cluster": "default",
    "ecs_task_arn": "arn:aws:ecs:us-east-1:xxxxxxxxxxxx:task/default/xxxxxx",
    "ecs_task_definition": "firelens-example-multiline:2"
}
```

下列日誌程式碼片段顯示了相同訊息僅以單筆明細行顯示的方式 (如果您執行了未串連多行日誌訊息的 Amazon ECS 容器)。

```
{
    "log": "Dec 14 06:41:08 Exception in thread \"main\"
 java.lang.RuntimeException: Something has gone wrong, aborting!",
    "container_id": "xxxxxx-xxxxxx",
    "container_name": "app",
    "source": "stdout",
    "ecs_cluster": "default",
    "ecs_task_arn": "arn:aws:ecs:us-east-1:xxxxxxxxxxxx:task/default/xxxxxx",
    "ecs_task_definition": "firelens-example-multiline:3"
}
```

範例：使用 Fluent Bit 內建剖析器

在本範例中，您會完成下列步驟：

1. 建置並上傳 Fluent Bit 容器的映像。
2. 建置並上傳可執行、失敗並產生多行堆疊追蹤之試用版多行應用程式的映像。
3. 建立任務定義並執行任務。
4. 檢視日誌以確認跨多行的訊息是否顯示為串連。

建置並上傳 Fluent Bit 容器的映像

此映像將包含參照了 Fluent Bit 剖析器的組態檔案。

1. 建立名稱為 `FluentBitDockerImage` 的資料夾。
2. 在 `FluentBitDockerImage` 資料夾內，建立參照了 Fluent Bit 內建剖析器檔案的自訂組態檔案。

   如需有關自訂組態檔案的詳細資訊，請參閱《Amazon Elastic Container Service 開發人員指南》中的指定自訂組態檔案

a. 將以下內容貼到該檔案：

```
[FILTER]
    name                  multiline
    match                 *
    multiline.key_content log
    multiline.parser      go
```

b. 儲存檔案為 `extra.conf`。

3. 在 `FluentBitDockerImage` 資料夾內，建立具有 Fluent Bit 映像與剖析器和您建立的組態檔案之 Dockerfile。

a. 將以下內容貼到該檔案：

```
FROM public.ecr.aws/aws-observability/aws-for-fluent-bit:latest
ADD extra.conf /extra.conf
```

b. 儲存檔案為 `Dockerfile`。

4. 使用 Dockerfile，建置內含自訂組態檔案的自訂 Fluent Bit 映像。

> **Note**
>
> 您可以將組態檔案放置在 Docker 影像中除了 `/fluent-bit/etc/fluent-bit.conf` 以外的任何位置，因為該檔案路徑會由 FireLens 使用。

a. 建置映像：`docker build -t fluent-bit-multiline-image .`

其中：`fluent-bit-multiline-image` 是此範例中映像的名稱。

b. 驗證映像已正確建立：`docker images –filter reference=fluent-bit-multiline-image`

如果成功，輸出會顯示映像和 `latest` 標籤。

5. 將自訂 Fluent Bit 映像上傳至 Amazon Elastic 容器登錄檔。

a. 建立 Amazon ECR 儲存庫，以便存放映像：`aws ecr create-repository --repository-name fluent-bit-multiline-repo --region us-east-1`

其中：fluent-bit-multiline-repo 是此範例中儲存庫的名稱，us-east-1 是此範例中的區域。

輸出為您提供了新儲存庫的詳細資訊。

b. 使用來自先前輸出的 repositoryUri 值，標記您的映像：docker tag *fluent-bit-multiline-image repositoryUri*

範例：docker tag fluent-bit-multiline-image xxxxxxxxxxxx.dkr.ecr.us-east-1.amazonaws.com/fluent-bit-multiline-repo

c. 執行 Docker 影像以確認是否能正確執行：docker images –filter reference=*repositoryUri*

在輸出中，儲存庫名稱從 fluent-bit-multiline-repo 變更為 repositoryUri。

d. 透過執行 aws ecr get-login-password 命令並指定您要驗證的登錄 ID 來驗證 Amazon ECR：aws ecr get-login-password | docker login --username AWS --password-stdin *registry ID*.dkr.ecr.*region*.amazonaws.com

範例：ecr get-login-password | docker login --username AWS --password-stdin xxxxxxxxxxxx.dkr.ecr.us-east-1.amazonaws.com

畫面上會顯示成功登入訊息。

e. 將映像推送至 Amazon ECR：docker push *registry ID*.dkr.ecr.*region*.amazonaws.com/*repository name*

範例：docker push xxxxxxxxxxxx.dkr.ecr.us-east-1.amazonaws.com/fluent-bit-multiline-repo

建置並上傳試用版多行應用程式的映像

此映像將包含執行應用程式的 Python 指令碼檔案和樣本日誌檔案。

1. 建立名為 multiline-app 的資料夾：mkdir multiline-app
2. 建立 Python 指令碼檔案。
   a. 在 multiline-app 資料夾中，建立檔案，並將它命名為 main.py。
   b. 將以下內容貼到該檔案：

```
import os
import time
file1 = open('/test.log', 'r')
Lines = file1.readlines()

count = 0

for i in range(10):
    print("app running normally...")
    time.sleep(1)

# Strips the newline character
for line in Lines:
    count += 1
    print(line.rstrip())
print(count)
print("app terminated.")
```

c. 儲存 main.py 檔案。

3. 建立範例日誌檔案。

a. 在 multiline-app 資料夾中，建立檔案，並將它命名為 test.log。

b. 將以下內容貼到該檔案：

```
panic: my panic

goroutine 4 [running]:
panic(0x45cb40, 0x47ad70)
  /usr/local/go/src/runtime/panic.go:542 +0x46c fp=0xc42003f7b8 sp=0xc42003f710
 pc=0x422f7c
main.main.func1(0xc420024120)
  foo.go:6 +0x39 fp=0xc42003f7d8 sp=0xc42003f7b8 pc=0x451339
runtime.goexit()
  /usr/local/go/src/runtime/asm_amd64.s:2337 +0x1 fp=0xc42003f7e0
 sp=0xc42003f7d8 pc=0x44b4d1
created by main.main
  foo.go:5 +0x58

goroutine 1 [chan receive]:
runtime.gopark(0x4739b8, 0xc420024178, 0x46fcd7, 0xc, 0xc420028e17, 0x3)
```

```
  /usr/local/go/src/runtime/proc.go:280 +0x12c fp=0xc420053e30 sp=0xc420053e00
 pc=0x42503c
runtime.goparkunlock(0xc420024178, 0x46fcd7, 0xc, 0x1000f010040c217, 0x3)
  /usr/local/go/src/runtime/proc.go:286 +0x5e fp=0xc420053e70 sp=0xc420053e30
 pc=0x42512e
runtime.chanrecv(0xc420024120, 0x0, 0xc420053f01, 0x4512d8)
  /usr/local/go/src/runtime/chan.go:506 +0x304 fp=0xc420053f20 sp=0xc420053e70
 pc=0x4046b4
runtime.chanrecv1(0xc420024120, 0x0)
  /usr/local/go/src/runtime/chan.go:388 +0x2b fp=0xc420053f50 sp=0xc420053f20
 pc=0x40439b
main.main()
  foo.go:9 +0x6f fp=0xc420053f80 sp=0xc420053f50 pc=0x4512ef
runtime.main()
  /usr/local/go/src/runtime/proc.go:185 +0x20d fp=0xc420053fe0 sp=0xc420053f80
 pc=0x424bad
runtime.goexit()
  /usr/local/go/src/runtime/asm_amd64.s:2337 +0x1 fp=0xc420053fe8
 sp=0xc420053fe0 pc=0x44b4d1

goroutine 2 [force gc (idle)]:
runtime.gopark(0x4739b8, 0x4ad720, 0x47001e, 0xf, 0x14, 0x1)
  /usr/local/go/src/runtime/proc.go:280 +0x12c fp=0xc42003e768 sp=0xc42003e738
 pc=0x42503c
runtime.goparkunlock(0x4ad720, 0x47001e, 0xf, 0xc420000114, 0x1)
  /usr/local/go/src/runtime/proc.go:286 +0x5e fp=0xc42003e7a8 sp=0xc42003e768
 pc=0x42512e
runtime.forcegchelper()
  /usr/local/go/src/runtime/proc.go:238 +0xcc fp=0xc42003e7e0 sp=0xc42003e7a8
 pc=0x424e5c
runtime.goexit()
  /usr/local/go/src/runtime/asm_amd64.s:2337 +0x1 fp=0xc42003e7e8
 sp=0xc42003e7e0 pc=0x44b4d1
created by runtime.init.4
  /usr/local/go/src/runtime/proc.go:227 +0x35

goroutine 3 [GC sweep wait]:
runtime.gopark(0x4739b8, 0x4ad7e0, 0x46fdd2, 0xd, 0x419914, 0x1)
  /usr/local/go/src/runtime/proc.go:280 +0x12c fp=0xc42003ef60 sp=0xc42003ef30
 pc=0x42503c
runtime.goparkunlock(0x4ad7e0, 0x46fdd2, 0xd, 0x14, 0x1)
  /usr/local/go/src/runtime/proc.go:286 +0x5e fp=0xc42003efa0 sp=0xc42003ef60
 pc=0x42512e
runtime.bgsweep(0xc42001e150)
```

```
  /usr/local/go/src/runtime/mgcsweep.go:52 +0xa3 fp=0xc42003efd8
 sp=0xc42003efa0 pc=0x419973
runtime.goexit()
  /usr/local/go/src/runtime/asm_amd64.s:2337 +0x1 fp=0xc42003efe0
 sp=0xc42003efd8 pc=0x44b4d1
created by runtime.gcenable
  /usr/local/go/src/runtime/mgc.go:216 +0x58
one more line, no multiline
```

c. 儲存 `test.log` 檔案。

4. 在 `multiline-app` 資料夾內，建立 Dockerfile。

a. 將以下內容貼到該檔案：

```
FROM public.ecr.aws/amazonlinux/amazonlinux:latest
ADD test.log /test.log

RUN yum upgrade -y && yum install -y python3

WORKDIR /usr/local/bin

COPY main.py .

CMD ["python3", "main.py"]
```

b. 儲存 `Dockerfile` 檔案。

5. 使用 Dockerfile，建置映像。

a. 建置映像：`docker build -t multiline-app-image .`

其中：`multiline-app-image` 是此範例中映像的名稱。

b. 驗證映像已正確建立：`docker images –filter reference=multiline-app-image`

如果成功，輸出會顯示映像和 `latest` 標籤。

6. 將映像上傳至 Amazon Elastic 容器登錄檔。

a. 建立 Amazon ECR 儲存庫，以便存放映像：`aws ecr create-repository --repository-name multiline-app-repo --region us-east-1`

其中：`multiline-app-repo` 是此範例中儲存庫的名稱，`us-east-1` 是此範例中的區域。

輸出為您提供了新儲存庫的詳細資訊。請記下 repositoryUri 值，因為您將在後續步驟用到它。

b. 使用來自先前輸出的 repositoryUri 值，標記您的映像：docker tag *multiline-app-image repositoryUri*

範例：docker tag multiline-app-image xxxxxxxxxxxx.dkr.ecr.us-east-1.amazonaws.com/multiline-app-repo

c. 執行 Docker 影像以確認是否能正確執行：docker images –filter reference=*repositoryUri*

在輸出中，儲存庫名稱會從 multiline-app-repo 變更為 repositoryUri 值。

d. 將映像推送至 Amazon ECR：docker push *aws_account_id*.dkr.ecr.*region*.amazonaws.com/*repository name*

範例：docker push *xxxxxxxxxxxx*.dkr.ecr.*us-east-1*.amazonaws.com/multiline-app-repo

建立任務定義並執行任務

1. 建立檔案名稱為 multiline-task-definition.json 的任務定義檔案。
2. 將以下內容貼到該 multiline-task-definition.json 檔案：

```
{
    "family": "firelens-example-multiline",
    "taskRoleArn": "task role ARN",
    "executionRoleArn": "execution role ARN",
    "containerDefinitions": [
        {
            "essential": true,
            "image": "aws_account_id.dkr.ecr.us-east-1.amazonaws.com/fluent-bit-multiline-image:latest",
            "name": "log_router",
            "firelensConfiguration": {
                "type": "fluentbit",
                "options": {
                    "config-file-type": "file",
                    "config-file-value": "/extra.conf"
                }
            },
```

```
            "memoryReservation": 50
        },
        {
            "essential": true,
            "image": "aws_account_id.dkr.ecr.us-east-1.amazonaws.com/multiline-app-
image:latest",
            "name": "app",
            "logConfiguration": {
                "logDriver": "awsfirelens",
                "options": {
                    "Name": "cloudwatch_logs",
                    "region": "us-east-1",
                    "log_group_name": "multiline-test/application",
                    "auto_create_group": "true",
                    "log_stream_prefix": "multiline-"
                }
            },
            "memoryReservation": 100
        }
    ],
    "requiresCompatibilities": ["FARGATE"],
    "networkMode": "awsvpc",
    "cpu": "256",
    "memory": "512"
}
```

取代下列在 multiline-task-definition.json 任務定義中的項目：

a. *task role ARN*

   若要尋找任務角色 ARN，請前往 IAM 主控台。選擇 Roles (角色) 並尋找您建立的 ecs-task-role-for-firelens 任務角色。選擇角色並複製顯示於 Summary (摘要) 區段的 ARN。

b. *execution role ARN*

   若要尋找執行角色 ARN，請前往 IAM 主控台。選擇 Roles (角色) 並尋找 ecsTaskExecutionRole 角色。選擇角色並複製顯示於 Summary (摘要) 區段的 ARN。

c. *aws_account_id*

   若要尋找您的 aws_account_id，請登入 AWS Management Console。選擇右上角的使用者名稱並複製您的帳戶 ID。

d. *us-east-1*

如有必要，請取代該區域。

3. 註冊任務定義檔案：`aws ecs register-task-definition --cli-input-json file://multiline-task-definition.json --region us-east-1`
4. 開啟主控台，網址為 https://console.aws.amazon.com/ecs/v2。
5. 在導覽窗格中，選擇 Task Definitions (任務定義)，然後選擇 `firelens-example-multiline` 系列，因為我們已在上述任務定義的第一行中將任務定義註冊到此系列。
6. 選擇最新版本。
7. 選擇部署、執行任務。
8. 在執行任務頁面上，針對叢集選擇叢集，然後在聯網下，針對子網路選擇任務的可用子網路。
9. 選擇建立。

確認 Amazon CloudWatch 中的多行日誌訊息是否顯示為串連

1. 透過 https://console.aws.amazon.com/cloudwatch/ 開啟 CloudWatch 主控台。
2. 在導覽窗格中，展開 Logs (日誌) 並選擇 Log groups (日誌群組)。
3. 選擇 `multiline-test/applicatio` 日誌群組。
4. 選擇日誌並檢視訊息。系統會串連與剖析器檔案中的規則相符之明細行，並顯示為單則訊息。

下列日誌程式碼片段顯示了串連到單一事件的 Go 堆疊追蹤：

```
{
    "log": "panic: my panic\n\ngoroutine 4 [running]:\npanic(0x45cb40,
 0x47ad70)\n  /usr/local/go/src/runtime/panic.go:542 +0x46c fp=0xc42003f7b8
 sp=0xc42003f710 pc=0x422f7c\nmain.main.func1(0xc420024120)\n  foo.go:6
 +0x39 fp=0xc42003f7d8 sp=0xc42003f7b8 pc=0x451339\nruntime.goexit()\n  /usr/
local/go/src/runtime/asm_amd64.s:2337 +0x1 fp=0xc42003f7e0 sp=0xc42003f7d8
 pc=0x44b4d1\ncreated by main.main\n  foo.go:5 +0x58\n\ngoroutine 1 [chan receive]:
\nruntime.gopark(0x4739b8, 0xc420024178, 0x46fcd7, 0xc, 0xc420028e17, 0x3)\n  /usr/
local/go/src/runtime/proc.go:280 +0x12c fp=0xc420053e30 sp=0xc420053e00 pc=0x42503c
\nruntime.goparkunlock(0xc420024178, 0x46fcd7, 0xc, 0x1000f010040c217, 0x3)\n
  /usr/local/go/src/runtime/proc.go:286 +0x5e fp=0xc420053e70 sp=0xc420053e30
 pc=0x42512e\nruntime.chanrecv(0xc420024120, 0x0, 0xc420053f01, 0x4512d8)\n
  /usr/local/go/src/runtime/chan.go:506 +0x304 fp=0xc420053f20 sp=0xc420053e70
 pc=0x4046b4\nruntime.chanrecv1(0xc420024120, 0x0)\n  /usr/local/go/src/runtime/
chan.go:388 +0x2b fp=0xc420053f50 sp=0xc420053f20 pc=0x40439b\nmain.main()\n
 foo.go:9 +0x6f fp=0xc420053f80 sp=0xc420053f50 pc=0x4512ef\nruntime.main()\n
```

```
  /usr/local/go/src/runtime/proc.go:185 +0x20d fp=0xc420053fe0 sp=0xc420053f80
 pc=0x424bad\nruntime.goexit()\n  /usr/local/go/src/runtime/asm_amd64.s:2337
 +0x1 fp=0xc420053fe8 sp=0xc420053fe0 pc=0x44b4d1\n\ngoroutine 2 [force gc
 (idle)]:\nruntime.gopark(0x4739b8, 0x4ad720, 0x47001e, 0xf, 0x14, 0x1)\n  /
usr/local/go/src/runtime/proc.go:280 +0x12c fp=0xc42003e768 sp=0xc42003e738
 pc=0x42503c\nruntime.goparkunlock(0x4ad720, 0x47001e, 0xf, 0xc420000114, 0x1)\n
  /usr/local/go/src/runtime/proc.go:286 +0x5e fp=0xc42003e7a8 sp=0xc42003e768
 pc=0x42512e\nruntime.forcegchelper()\n  /usr/local/go/src/runtime/proc.go:238
 +0xcc fp=0xc42003e7e0 sp=0xc42003e7a8 pc=0x424e5c\nruntime.goexit()\n  /usr/
local/go/src/runtime/asm_amd64.s:2337 +0x1 fp=0xc42003e7e8 sp=0xc42003e7e0
 pc=0x44b4d1\ncreated by runtime.init.4\n  /usr/local/go/src/runtime/proc.go:227
 +0x35\n\ngoroutine 3 [GC sweep wait]:\nruntime.gopark(0x4739b8, 0x4ad7e0,
 0x46fdd2, 0xd, 0x419914, 0x1)\n  /usr/local/go/src/runtime/proc.go:280 +0x12c
 fp=0xc42003ef60 sp=0xc42003ef30 pc=0x42503c\nruntime.goparkunlock(0x4ad7e0,
 0x46fdd2, 0xd, 0x14, 0x1)\n  /usr/local/go/src/runtime/proc.go:286 +0x5e
 fp=0xc42003efa0 sp=0xc42003ef60 pc=0x42512e\nruntime.bgsweep(0xc42001e150)\n
  /usr/local/go/src/runtime/mgcsweep.go:52 +0xa3 fp=0xc42003efd8 sp=0xc42003efa0
 pc=0x419973\nruntime.goexit()\n  /usr/local/go/src/runtime/asm_amd64.s:2337 +0x1
 fp=0xc42003efe0 sp=0xc42003efd8 pc=0x44b4d1\ncreated by runtime.gcenable\n  /usr/
local/go/src/runtime/mgc.go:216 +0x58",
    "container_id": "xxxxxx-xxxxxx",
    "container_name": "app",
    "source": "stdout",
    "ecs_cluster": "default",
    "ecs_task_arn": "arn:aws:ecs:us-east-1:xxxxxxxxxxxx:task/default/xxxxxx",
    "ecs_task_definition": "firelens-example-multiline:2"
}
```

下列日誌程式碼片段顯示了相同事件顯示的方式 (如果您執行了未串連多行日誌訊息的 ECS 容器)。日誌欄位包含單筆明細行。

```
{
    "log": "panic: my panic",
    "container_id": "xxxxxx-xxxxxx",
    "container_name": "app",
    "source": "stdout",
    "ecs_cluster": "default",
    "ecs_task_arn": "arn:aws:ecs:us-east-1:xxxxxxxxxxxx:task/default/xxxxxx",
    "ecs_task_definition": "firelens-example-multiline:3"
```

> **Note**
>
> 如果您的日誌轉到日誌檔案而不是標準輸出，我們建議您在結尾輸入外掛程式 (而非篩選條件) 中指定 `multiline.parser` 和 `multiline.key_content` 組態參數。

在 Amazon ECS Windows 容器上部署 Fluent Bit

Fluent Bit 是一種快速靈活的日誌處理器和路由器，支援各種作業系統。它可用來將日誌路由到各種 AWS 目的地，例如 Amazon CloudWatch Logs、Firehose Amazon S3 和 Amazon OpenSearch Service。Fluent Bit 支援常見的合作夥伴解決方案，例如 Datadog、Splunk 和自訂 HTTP 伺服器。如需有關 Fluent Bit 的詳細資訊，請參閱 Fluent Bit 網站。

AWS for Fluent Bit 映像在大多數區域的 Amazon ECR Public Gallery 和 Amazon ECR 儲存庫的 Amazon ECR 上可用以提供高可用性。如需詳細資訊，請參閱 GitHub 網站上的 aws-for-fluent-bit 。

本教學課程會逐步引導您如何在 Amazon ECS 中執行的 Windows 執行個體上部署 Fluent Bit 容器，從而將 Windows 任務產生的日誌串流至 Amazon CloudWatch 進行集中式記錄。

本教學課程使用下列作法：

- Fluent Bit 可作為具有常駐程式排程策略的服務執行。此策略可確保 Fluent Bit 的單一執行個體一律在叢集中的容器執行個體上執行。
  - 使用轉寄輸入外掛程式接聽連接埠 24224。
  - 向主機公開連接埠 24224，以便 Docker 執行時間可使用此公開連接埠將日誌傳送到 Fluent Bit。
  - 具有允許 Fluent Bit 將日誌記錄傳送到指定目的地的組態。
- 使用 Fluentd 日誌記錄驅動程式啟動所有其他 Amazon ECS 任務容器。如需詳細資訊，請參閱 Docker 文件網站上的 Fluentd logging driver (Fluentd 日誌記錄驅動程式)。
  - Docker 連線到主機命名空間內 localhost 上的 TCP 通訊端 24224。
  - Amazon ECS 代理程式會將標籤新增至容器，其中包括叢集名稱、任務定義系列名稱、任務定義修訂版編號、任務 ARN 和容器名稱。使用 Fluentd Docker 日誌記錄驅動程式的標籤選項將相同的資訊新增到日誌記錄中。如需詳細資訊，請參閱 Docker 文件網站上的 labels、labels-regex、env 和 env-regex。
  - 由於 Fluentd 日誌記錄驅動程式的 `async` 選項設定為 `true`，當 Fluent Bit 容器重新啟動時，Docker 會緩衝日誌，直到 Fluent Bit 容器重新啟動為止。您可以透過設定 fluentd-buffer-limit 選項來增加緩衝限制。如需詳細資訊，請參閱 Docker 文件網站上的 fluentd-buffer-limit。

工作流程如下所示：

- Fluent Bit 容器會啟動並接聽公開至主機的連接埠 24224。
- Fluent Bit 會使用任務定義中指定的任務 IAM 角色憑證。
- 在同一個執行個體上啟動的其他任務，會使用 Fluentd Docker 日誌記錄驅動程式連線到連接埠 24224 上的 Fluent Bit 容器。
- 當應用程式容器生成日誌時，Docker 執行時間會標記這些記錄，新增標籤中指定的其他中繼資料，然後在主機命名空間的連接埠 24224 上轉寄這些中繼資料。
- Fluent Bit 會接收連接埠 24224 上的日誌記錄，因為其公開到主機命名空間。
- Fluent Bit 會執行其內部處理，並依照指定的方式路由日誌。

本教學課程使用預設的 CloudWatch Fluent Bit 組態，可執行下列作業：

- 為每個叢集和任務定義系列建立新的日誌群組。
- 每當啟動新任務時，為上述產生的日誌群組中的每個任務容器建立新的日誌串流。每個串流將標記容器所屬的任務 ID。
- 在每個日誌項目中新增其他中繼資料，包括叢集名稱、任務 ARN、任務容器名稱、任務定義系列以及任務定義修訂版編號。

  例如，如果您的 `task_1` 帶有 `container_1` 和 `container_2`，而 `ask_2` 帶有 `container_3`，則下列項目為 CloudWatch 日誌串流：

  - `/aws/ecs/windows.ecs_task_1`

    `task-out.`*`TASK_ID`*`.container_1`

    `task-out.`*`TASK_ID`*`.container_2`
  - `/aws/ecs/windows.ecs_task_2`

    `task-out.`*`TASK_ID`*`.container_3`

步驟

- 必要條件
- 步驟 1：建立 IAM 存取角色
- 步驟 2：建立 Amazon ECS Windows 容器執行個體
- 步驟 3：設定 Fluent Bit

必要條件

本教學課程假設已完成下列先決條件：

- AWS CLI 已安裝並設定最新版本的 。如需詳細資訊，請參閱安裝或更新至最新版本的 AWS CLI。
- `aws-for-fluent-bit` 容器映像可用於下列 Windows 作業系統：
  - Windows Server 2019 Core
  - Windows Server 2019 Full
  - Windows Server 2022 Core
  - Windows Server 2022 Full
- 已完成「設定以使用 Amazon ECS。」中的步驟。
- 您有一個叢集。在本教學課程中，叢集名稱是 FluentBit-cluster。
- 您有一個帶有公共子網路的 VPC，EC2 執行個體將在其中啟動。您可以使用預設 VPC。您也可以使用允許 Amazon CloudWatch 端點連上子網路的私有子網路。如需有關 Amazon CloudWatch 端點的詳細資訊，請參閱《AWS 一般參考》中的 Amazon CloudWatch 端點和配額。如需使用 Amazon VPC 精靈建立 VPC 的相關資訊，請參閱the section called “建立 Virtual Private Cloud”。

步驟 1：建立 IAM 存取角色

建立 Amazon ECS IAM 角色。

1. 建立名為「ecsInstanceRole」的 Amazon ECS 容器執行個體角色。如需詳細資訊，請參閱 Amazon ECS 容器執行個體 IAM 角色。
2. 為命名為 `fluentTaskRole` 的 Fluent Bit 任務建立 IAM 角色。如需詳細資訊，請參閱the section called “任務 IAM 角色”。

在此 IAM 角色中授予的 IAM 許可是由任務容器承繼。為了允許 Fluent Bit 傳送日誌到 CloudWatch，您需要將下列許可附加到任務 IAM 角色。

```
{
    "Version": "2012-10-17",
    "Statement": [
    {
        "Effect": "Allow",
        "Action": [
            "logs:CreateLogStream",
            "logs:CreateLogGroup",
            "logs:DescribeLogStreams",
            "logs:PutLogEvents"
        ],
        "Resource": "*"
    }
    ]
}
```

3. 將政策連接到角色。

    a. 將上述內容儲存在名為 `fluent-bit-policy.json` 的檔案中。

    b. 執行下列命令，將內嵌政策附加到 `fluentTaskRole` IAM 角色。

    ```
    aws iam put-role-policy --role-name fluentTaskRole --policy-name
     fluentTaskPolicy --policy-document file://fluent-bit-policy.json
    ```

步驟 2：建立 Amazon ECS Windows 容器執行個體

建立 Amazon ECS Windows 容器執行個體。

建立 Amazon ECS 執行個體

1. 使用 `aws ssm get-parameters` 命令擷取託管 VPC 的區域 AMI ID。如需詳細資訊，請參閱擷取 Amazon ECS 最佳化 AMI 中繼資料。
2. 使用 Amazon EC2 主控台來啟動執行個體。

    a. 在 https://console.aws.amazon.com/ec2/ 開啟 Amazon EC2 主控台。

    b. 從導覽列中選取要使用的「區域」。

c. 在 EC2 儀表板中，選擇 Launch instance (啟動執行個體)。

d. 在 Name (名稱) 輸入唯一的名稱。

e. 在 Application and OS Images (Amazon Machine Image) (應用程式和作業系統映像 (Amazon Machine Image)) 欄位中，選擇您在步驟 1 中擷取的 AMI。

f. 對於 Instance type (執行個體類型)，選擇 `t3.xlarge`。

g. 在 Key pair (login) (金鑰對 (登入)) 欄位中，選擇一個金鑰對。

h. 在 Network settings (網路設定) 下的 Security group (安全群組) 欄位中，選擇一個現有的安全群組，或建立一個新的安全群組。

i. 在 Network settings (網路設定) 下的 Auto-assign Public IP (自動指派公有 IP) 欄位中，選取 Enable (啟用)。

j. 在 Advanced details (進階詳細資料) 下的 IAM instance profile (IAM 執行個體設定檔) 欄位中，選擇 ecsInstanceRole。

k. 利用下列使用者資料設定您的 Amazon ECS 容器執行個體。在 Advanced details (進階詳細資料) 下的 User data (使用者資料) 欄位中，貼入下列指令碼，以您的叢集名稱取代 *`cluster_name`*。

```
<powershell>
Import-Module ECSTools
Initialize-ECSAgent -Cluster cluster-name -EnableTaskENI -EnableTaskIAMRole -
LoggingDrivers '["awslogs","fluentd"]'
</powershell>
```

l. 準備就緒後，請選取 acknowledgment (確認) 欄位，再選擇 Launch Instances (啟動執行個體)。

m. 會有確認頁面讓您知道您的執行個體正在啟動。選擇 View Instances (檢視執行個體) 關閉確認頁面並返回主控台。

步驟 3：設定 Fluent Bit

您可以使用 提供的下列預設組態 AWS 來快速開始使用：

- Amazon CloudWatch，即基於《Fluent Bit 官方手冊》中 Amazon CloudWatch 的 Fluent Bit 外掛程式。

或者，您可以使用 提供的其他預設組態 AWS。如需詳細資訊，請參閱 Github 網站上有關 aws-for-fluent-bit 的 Overriding the entrypoint for the Windows image (覆寫 Windows 映像的進入點)。

預設的 Amazon CloudWatch Fluent Bit 組態如下所示。

取代下列變數：

- 將 *region* 取代為您要將 Amazon CloudWatch Logs 傳送到的區域。

```
[SERVICE]
    Flush               5
    Log_Level           info
    Daemon              off

[INPUT]
    Name                forward
    Listen              0.0.0.0
    Port                24224
    Buffer_Chunk_Size   1M
    Buffer_Max_Size     6M
    Tag_Prefix          ecs.

# Amazon ECS agent adds the following log keys as labels to the docker container.
# We would use fluentd logging driver to add these to log record while sending it to
 Fluent Bit.
[FILTER]
    Name                modify
    Match               ecs.*
    Rename              com.amazonaws.ecs.cluster ecs_cluster
    Rename              com.amazonaws.ecs.container-name ecs_container_name
    Rename              com.amazonaws.ecs.task-arn ecs_task_arn
    Rename              com.amazonaws.ecs.task-definition-family
 ecs_task_definition_family
    Rename              com.amazonaws.ecs.task-definition-version
 ecs_task_definition_version

[FILTER]
    Name                rewrite_tag
    Match               ecs.*
    Rule                $ecs_task_arn ^([a-z-:0-9]+)/([a-zA-Z0-9-_]+)/([a-z0-9]+)$
 out.$3.$ecs_container_name false
    Emitter_Name        re_emitted
```

```
[OUTPUT]
    Name                cloudwatch_logs
    Match               out.*
    region              region
    log_group_name      fallback-group
    log_group_template  /aws/ecs/$ecs_cluster.$ecs_task_definition_family
    log_stream_prefix   task-
    auto_create_group   On
```

每個進入 Fluent Bit 的日誌都有一個您指定的標籤，或者在您不提供標籤時自動生成。這些標籤可用來將不同的日誌路由到不同的目的地。如需詳細資訊，請參閱《Fluent Bit 官方手冊》中的標籤。

上述 Fluent Bit 組態具有下列屬性：

- 轉寄輸入外掛程式會接聽 TCP 連接埠 24224 上的傳入流量。
- 在該連接埠上收到的每個日誌項目都有一個標籤，轉寄輸入外掛程式會修改該標籤，以在記錄前面加上 `ecs.` 字串。
- Fluent Bit 內部管道會路由日誌項目，以使用符合規則運算式修改篩選條件。此篩選條件將日誌記錄 JSON 中的金鑰替換為 Fluent Bit 可以使用的格式。
- 然後，rewrite_tag 篩選條件會使用修改後的日誌項目。此篩選條件會將日誌記錄的標籤變更為格式輸出 *TASK_ID.CONTAINER_NAME*。
- 新標籤將路由至輸出 cloudwatch_logs 外掛程式，該外掛程式會使用 CloudWatch 輸出外掛程式的 `log_group_template` 和 `log_stream_prefix` 選項來建立前文所述的日誌群組和串流。如需其他資訊，請參閱《Fluent Bit 官方手冊》中的組態參數。

步驟 4：註冊會將日誌路由到 CloudWatch 的 Windows Fluent Bit 任務定義

註冊會將日誌路由到 CloudWatch 的 Windows Fluent Bit 任務定義。

> **Note**
>
> 此任務定義會將 Fluent Bit 容器連接埠 24224 公開至主機連接埠 24224。確認此連接埠未在 EC2 執行個體安全群組中開啟，防止從外部存取。

註冊任務定義

1. 使用下列內容建立名為 `fluent-bit.json` 的檔案。

取代下列變數：

- 將 *task-iam-role* 替代為任務 IAM 角色的 Amazon Resource Name (ARN)
- 將 *region* 替代為任務執行所在的區域

```
{
  "family": "ecs-windows-fluent-bit",
  "taskRoleArn": "task-iam-role",
  "containerDefinitions": [
    {
      "name": "fluent-bit",
      "image": "public.ecr.aws/aws-observability/aws-for-fluent-
bit:windowsservercore-latest",
      "cpu": 512,
      "portMappings": [
        {
          "hostPort": 24224,
          "containerPort": 24224,
          "protocol": "tcp"
        }
      ],
      "entryPoint": [
        "Powershell",
        "-Command"
      ],
      "command": [
        "C:\\entrypoint.ps1 -ConfigFile C:\\ecs_windows_forward_daemon\
\cloudwatch.conf"
      ],
      "environment": [
        {
          "name": "AWS_REGION",
          "value": "region"
        }
      ],
      "memory": 512,
      "essential": true,
      "logConfiguration": {
        "logDriver": "awslogs",
        "options": {
          "awslogs-group": "/ecs/fluent-bit-logs",
```

```
            "awslogs-region": "region",
            "awslogs-stream-prefix": "flb",
            "awslogs-create-group": "true"
          }
        }
      }
  ],
  "memory": "512",
  "cpu": "512"
}
```

2. 執行下列命令來註冊任務定義。

```
aws ecs register-task-definition --cli-input-json file://fluent-bit.json --
region region
```

您可以執行 `list-task-definitions` 命令列出您帳戶的任務定義。輸出顯示可與 `run-task` 或 `start-task` 搭配使用的系列和修訂版值。

步驟 5：使用常駐程式排程策略以 Amazon ECS 服務形式執行 **ecs-windows-fluent-bit** 任務定義

註冊帳戶的任務定義後，您就可以在叢集中執行任務。在本教學課程中，您會在 `FluentBit-cluster` 叢集中執行 `ecs-windows-fluent-bit:1` 任務定義的一個執行個體。在使用常駐程式排程策略的服務中執行任務，用以確保 Fluent Bit 的單一執行個體一律在每個容器執行個體上執行。

執行任務

1. 執行下列命令，以服務形式啟動 `ecs-windows-fluent-bit:1` 任務定義 (在上一個步驟中註冊)。

> **Note**
>
> 此任務定義使用 `awslogs` 日誌記錄驅動程式，您的容器執行個體需要具有必要的許可。

取代下列變數：

- 將 *region* 替代為服務執行所在的區域

```
aws ecs create-service \
    --cluster FluentBit-cluster \
    --service-name FluentBitForwardDaemonService \
    --task-definition ecs-windows-fluent-bit:1 \
    --launch-type EC2 \
    --scheduling-strategy DAEMON \
    --region region
```

2. 執行下列命令以列出您的任務。

   取代下列變數：

   - 將 *region* 替代為服務任務執行所在的區域

```
aws ecs list-tasks --cluster FluentBit-cluster --region region
```

步驟 6：註冊會產生日誌的 Windows 任務定義

註冊會產生日誌的任務定義。此任務定義會部署 Windows 容器映像，該映像會每秒向 `stdout` 寫入一個遞增數字。

任務定義會使用連線到 Fluent Bit 外掛程式接聽連接埠 24224 的 Fluentd 日誌記錄驅動程式。Amazon ECS 代理程式會為每個 Amazon ECS 容器新增標籤，包括叢集名稱、任務 ARN、任務定義系列名稱、任務定義修訂版編號和任務容器名稱。這些鍵/值對標籤會傳遞至 Fluent Bit。

> **Note**
>
> 此任務會使用 `default` 網路模式。不過，您也可以在任務中使用 `awsvpc` 網路模式。

註冊任務定義

1. 使用下列內容建立名為 `windows-app-task.json` 的檔案。

```
{
  "family": "windows-app-task",
  "containerDefinitions": [
```

```
    {
      "name": "sample-container",
      "image": "mcr.microsoft.com/windows/servercore:ltsc2019",
      "cpu": 512,
      "memory": 512,
      "essential": true,
      "entryPoint": [
        "Powershell",
        "-Command"
      ],
      "command": [
        "$count=1;while(1) { Write-Host $count; sleep 1; $count=$count+1;}"
      ],
      "logConfiguration": {
        "logDriver": "fluentd",
        "options": {
          "fluentd-address": "localhost:24224",
          "tag": "{{ index .ContainerLabels \"com.amazonaws.ecs.task-definition-
family\" }}",
          "fluentd-async": "true",
          "labels": "com.amazonaws.ecs.cluster,com.amazonaws.ecs.container-
name,com.amazonaws.ecs.task-arn,com.amazonaws.ecs.task-definition-
family,com.amazonaws.ecs.task-definition-version"
        }
      }
    }
  ],
  "memory": "512",
  "cpu": "512"
}
```

2. 執行下列命令來註冊任務定義。

   取代下列變數：

   - 將 *region* 替代為任務執行所在的區域

```
aws ecs register-task-definition --cli-input-json file://windows-app-task.json --
region region
```

   您可以執行 `list-task-definitions` 命令列出您帳戶的任務定義。輸出顯示可與 `run-task` 或 `start-task` 搭配使用的系列和修訂版值。

步驟 7：執行 windows-app-task 任務定義

註冊 windows-app-task 任務定義之後，請在 FluentBit-cluster 叢集中執行該定義。

執行任務

1. 執行您在上一個步驟中註冊的 windows-app-task:1 任務定義。

   取代下列變數：

   - 將 *region* 替代為任務執行所在的區域

   ```
   aws ecs run-task --cluster FluentBit-cluster --task-definition windows-app-task:1
    --count 2 --region region
   ```

2. 執行下列命令以列出您的任務。

   ```
   aws ecs list-tasks --cluster FluentBit-cluster
   ```

步驟 8：驗證 CloudWatch 上的日誌

為了驗證您的 Fluent Bit 設定，請在 CloudWatch 主控台中檢查下列日誌群組：

- /ecs/fluent-bit-logs – 這是日誌群組，對應在容器執行個體上執行的 Fluent Bit 常駐程式容器。
- /aws/ecs/FluentBit-cluster.windows-app-task - 這是日誌群組，對應在 FluentBit-cluster 叢集內為 windows-app-task 任務定義系列啟動的所有任務。

  task-out.*FIRST_TASK_ID*.sample-container – 此日誌串流包含 sample-container 任務容器中由任務的第一個執行個體產生的所有日誌。

  task-out.*SECOND_TASK_ID*.sample-container – 此日誌串流包含 sample-container 任務容器中由任務的第二個執行個體產生的所有日誌。

task-out.*TASK_ID*.sample-container 日誌串流的欄位與下列類似：

```
{
    "source": "stdout",
```

```
    "ecs_task_arn": "arn:aws:ecs:region:0123456789012:task/FluentBit-
cluster/13EXAMPLE",
    "container_name": "/ecs-windows-app-task-1-sample-container-cEXAMPLE",
    "ecs_cluster": "FluentBit-cluster",
    "ecs_container_name": "sample-container",
    "ecs_task_definition_version": "1",
    "container_id": "61f5e6EXAMPLE",
    "log": "10",
    "ecs_task_definition_family": "windows-app-task"
}
```

驗證 Fluent Bit 設定

1. 透過 https://console.aws.amazon.com/cloudwatch/ 開啟 CloudWatch 主控台。
2. 在導覽窗格中，選擇 Log groups (日誌群組)。請確定您位於將 Fluent Bit 部署到容器的區域。

   在 中的日誌群組清單中 AWS 區域，您應該會看到以下內容：

   - `/ecs/fluent-bit-logs`
   - `/aws/ecs/FluentBit-cluster.windows-app-task`

   如果您看到這些日誌群組，表示 Fluent Bit 設定已通過驗證。

步驟 9：清除

完成此教學課程時，清除與其相關的資源，以免未使用的資源產生費用。

清除教學課程資源

1. 停止 `windows-simple-task` 任務和 `ecs-fluent-bit` 任務。如需詳細資訊，請參閱the section called “停止任務”。
2. 執行下列命令來刪除 `/ecs/fluent-bit-logs` 日誌群組。如需有關刪除日誌群組的詳細資訊，請參閱《AWS Command Line Interface 參考》中的 delete-log-group。

   ```
   aws logs delete-log-group --log-group-name /ecs/fluent-bit-logs
   aws logs delete-log-group --log-group-name /aws/ecs/FluentBit-cluster.windows-app-
   task
   ```

3. 執行下列命令以終止執行個體。

```
aws ec2 terminate-instances --instance-ids instance-id
```

4. 執行下列命令以刪除 IAM 角色。

```
aws iam delete-role --role-name ecsInstanceRole
aws iam delete-role --role-name fluentTaskRole
```

5. 執行下列命令以刪除 Amazon ECS 叢集。

```
aws ecs delete-cluster --cluster FluentBit-cluster
```

在 Amazon ECS 上使用 gMSA for EC2 Linux容器 EC2

Amazon ECS 透過稱為群組受管服務帳戶 () 的特殊服務帳戶，支援 EC2 上 Linux 容器的 Active Directory 身分驗證gMSA。

.NET Core 應用程式等以 Linux 為基礎的網路應用程式，可使用 Active Directory 來促進使用者和服務之間的身分驗證和授權管理。您可以透過設計與 Active Directory 整合並在加入網域的伺服器上執行的應用程式來使用此功能。但是，由於 Linux 容器無法加入網域，因此您需要設定 Linux 容器來執行 gMSA。

使用 gMSA 執行的 Linux 容器依賴在容器主機 Amazon EC2 執行個體上執行的 `credentials-fetcher` 常駐程式。也就是說，常駐程式會從 Active Directory 網域控制站擷取 gMSA 憑證，然後將這些憑證傳輸到容器執行個體。如需有關服務帳戶的詳細資訊，請參閱 Microsoft Learn 網站上的針對 Windows 容器建立 gMSAs。

考量事項

針對 Linux 容器使用 gMSA 前，請考慮以下事項：

- 如果您的容器在 EC2 上執行，則可以針對 Windows 容器和 Linux 容器使用 gMSA。如需如何在 Fargate 上使用 gMSA Linux 容器的詳細資訊，請參閱 在 Fargate 上使用 gMSA 做為Linux容器。
- 您可能需要加入網域的 Windows 電腦才能完成先決條件。例如，您可能需要加入網域的 Windows 電腦，才能在使用 PowerShell 的 Active Directory 中建立 gMSA。RSATActive Director PowerShell 工具僅適用於 Windows。如需詳細資訊，請參閱安裝 Active Directory 管理工具。

- 您可以選擇無網域 gMSA 或將每個執行個體加入單一網域。使用無網域 gMSA 時，容器執行個體不會加入網域，執行個體上的其他應用程式無法使用憑證來存取網域，而加入不同網域的任務則可在相同的執行個體上執行。

  然後，選擇 CredSpec 的資料儲存，也可以選擇無網域 gMSA 的 Active Directory 使用者憑證的資料儲存。

  Amazon ECS 使用 Active Directory 憑證規格檔案 (CredSpec)。此檔案包含用於將 gMSA 帳戶內容傳播至容器的 gMSA 中繼資料。您會產生 CredSpec 檔案，然後將其儲存在下表中特定於容器執行個體的作業系統的 CredSpec 儲存選項之一。若要使用無網域方法，CredSpec 檔案中的選擇性區段可以在下表中的 domainless user credentials 儲存選項之一中指定憑證 (特定於容器執行個體的作業系統)。

儲存位置	Linux	Windows
Amazon Simple Storage Service	CredSpec	CredSpec
AWS Secrets Manager	無網域使用者憑證	無網域使用者憑證
Amazon EC2 Systems Manager 參數存放區	CredSpec	CredSpec，無網域使用者憑證
本機檔案	N/A	CredSpec

必要條件

在搭配 Amazon ECS 使用 Linux 容器功能適用的 gMSA 之前，請先完成下列各項：

- 您可以使用您希望容器存取的資源來設定 Active Directory 網域。Amazon ECS 支援下列設定：
  - AWS Directory Service Active Directory. AWS Directory Service 是託管在 Amazon EC2 上的 AWS 受管 Active Directory。如需詳細資訊，請參閱 AWS Directory Service 管理指南中的 AWS Managed Microsoft AD 入門。
  - 內部部署 Active Directory。您必須確認 Amazon ECS Linux 容器執行個體可以加入網域。如需詳細資訊，請參閱AWS Direct Connect。
- 您在 Active Directory 中擁有現有的 gMSA 帳戶。如需詳細資訊，請參閱在 Amazon ECS 上使用 gMSA for EC2 Linux容器 EC2 。

- 您已在 Amazon ECS Linux 容器執行個體上安裝並執行 `credentials-fetcher` 常駐程式。您也將一組初始的憑證新增至 `credentials-fetcher` 常駐程式，以便透過 Active Directory 進行身分驗證。

  > **Note**
  >
  > `credentials-fetcher` 常駐程式僅適用於 Amazon Linux 2023 和 Fedora 37 及更新版本。該常駐程式不適用於 Amazon Linux 2。如需詳細資訊，請參閱 GitHub 上的 aws/credentials-fetcher。

- 您可以設定 `credentials-fetcher` 常駐程式的憑證，以便使用 Active Directory 進行身分驗證。憑證必須是具有 gMSA 帳戶存取權的 Active Directory 安全群組的成員。決定是否要將執行個體加入網域，或是要使用無網域 gMSA。 中有多個選項。
- 您已新增必要的 IAM 許可 所需的許可取決於您為初始憑證選擇的方法以及儲存憑證規格的方法：
  - 如果您使用無網域gMSA做為初始登入資料， AWS Secrets Manager 則任務執行角色需要 的 IAM 許可。
  - 如果您將憑證規格存放在 SSM 參數存放區中，則任務執行角色需要 Amazon EC2 Systems Manager 參數存放區的 IAM 許可。
  - 如果您將憑證規格存放在 Amazon S3 中，則任務執行角色需要 Amazon Simple Storage Service 的 IAM 許可。

在 Amazon ECS 上設定能夠使用 gMSA 功能的 Linux 容器

準備基礎設施

下列步驟是執行一次的考量和設定。完成這些步驟後，您可以自動建立容器執行個體以重複使用此組態。

決定如何提供初始憑證，並在可重複使用的 EC2 啟動範本中設定 EC2 使用者資料以安裝 `credentials-fetcher` 常駐程式。

1. 決定是否要將執行個體加入網域，或是要使用無網域 gMSA。
   - 將 EC2 執行個體加入 Active Directory 網域

- 透過使用者資料加入執行個體

  在 EC2 啟動範本中新增將 Active Directory 網域加入 EC2 使用者資料的步驟。多個 Amazon EC2 Auto Scaling 群組可以使用相同的啟動範本。

  您可以在 Fedora 文件中使用這些步驟加入 Active Directory 或 FreeIPA 網域。

- 建立無網域 gMSA 的 Active Directory 使用者

  `credentials-fetcher` 常駐程式具有稱為無網域 gMSA 的功能。此功能需要網域，但 EC2 執行個體不需要加入網域。使用無網域 gMSA 時，容器執行個體不會加入網域，執行個體上的其他應用程式無法使用憑證來存取網域，而加入不同網域的任務則可在相同的執行個體上執行。相反地，您會在 CredSpec 檔案中提供 AWS Secrets Manager 祕密的名稱。祕密必須包含使用者名稱、密碼和要登入的網域。

  此功能受到支援，可與 Linux 和 Windows 容器搭配使用。

  此功能與 gMSA support for non-domain-joined container hosts 功能類似。如需有關 Windows 功能的詳細資訊，請參閱 Microsoft Learn 網站上的 gMSA 架構和改進。

  a. 在 Active Directory 網域中建立使用者。Active Directory 中的使用者必須具有存取您在任務中使用之 gMSA 服務帳戶的權限。

  b. 在 Active Directory 中建立使用者之後 AWS Secrets Manager，在 中建立秘密。如需詳細資訊，請參閱建立 AWS Secrets Manager 秘密。

  c. 將使用者的使用者名稱、密碼和網域分別輸入到名為 `username`、`password` 和 `domainName` 的 JSON 鍵值組中。

  ```
  {"username":"username","password":"passw0rd", "domainName":"example.com"}
  ```

  d. 將組態新增至服務帳戶的 CredSpec 檔案。其他的 `HostAccountConfig` 包含 Secrets Manager 祕密的 Amazon Resource Name (ARN)。

  在 Windows 上，`PluginGUID` 必須符合下列範例程式碼片段中的 GUID。在 Linux 上，會忽略 `PluginGUID`。將 `MySecret` 替換為含有祕密的 Amazon Resource Name (ARN) 的範例。

  ```
  "ActiveDirectoryConfig": {
      "HostAccountConfig": {
          "PortableCcgVersion": "1",
          "PluginGUID": "{859E1386-BDB4-49E8-85C7-3070B13920E1}",
  ```

```
            "PluginInput": {
                "CredentialArn": "arn:aws:secretsmanager:aws-
region:111122223333:secret:MySecret"
            }
        }
```

e. 無網域 gMSA 功能需要任務執行角色中的其他許可。請遵循步驟 (選用) 無網域 gMSA 祕密。

2. 設定執行個體並安裝 **credentials-fetcher** 常駐程式

您可以在 EC2 啟動範本中使用使用者資料指令碼來安裝 credentials-fetcher 常駐程式。以下範例示範兩種類型的使用者資料、cloud-config YAML 或 bash 指令碼。這些範例適用於 Amazon Linux 2023 (AL2023)。將 MyCluster 替換為您想要這些執行個體加入的 Amazon ECS 叢集的名稱。

- **cloud-config** YAML

```
Content-Type: text/cloud-config
package_reboot_if_required: true
packages:
  # prerequisites
  - dotnet
  - realmd
  - oddjob
  - oddjob-mkhomedir
  - sssd
  - adcli
  - krb5-workstation
  - samba-common-tools
  # https://github.com/aws/credentials-fetcher gMSA credentials management for
 containers
  - credentials-fetcher
write_files:
# configure the ECS Agent to join your cluster.
# replace MyCluster with the name of your cluster.
- path: /etc/ecs/ecs.config
  owner: root:root
  permissions: '0644'
  content: |
    ECS_CLUSTER=MyCluster
    ECS_GMSA_SUPPORTED=true
runcmd:
```

```
# start the credentials-fetcher daemon and if it succeeded, make it start after
 every reboot
- "systemctl start credentials-fetcher"
- "systemctl is-active credentials-fetcher && systemctl enable credentials-
fetcher"
```

- bash 指令碼

  如果您比較習慣 bash 指令碼，並且有多個要寫入 /etc/ecs/ecs.config 的變數，請使用以下 heredoc 格式。此格式會將開頭為 cat 和 EOF 之行間的所有項目寫入組態檔案。

```
#!/usr/bin/env bash
set -euxo pipefail

# prerequisites
timeout 30 dnf install -y dotnet realmd oddjob oddjob-mkhomedir sssd adcli
 krb5-workstation samba-common-tools
# install https://github.com/aws/credentials-fetcher gMSA credentials
 management for containers
timeout 30 dnf install -y credentials-fetcher

# start credentials-fetcher
systemctl start credentials-fetcher
systemctl is-active credentials-fetcher && systemctl enable credentials-fetcher

cat <<'EOF' >> /etc/ecs/ecs.config
ECS_CLUSTER=MyCluster
ECS_GMSA_SUPPORTED=true
EOF
```

選用組態變數適用於 credentials-fetcher 常駐程式，您可以在 /etc/ecs/ecs.config 中設定該變數。我們建議您在 YAML 區塊或與先前範例類似的 heredoc 內的使用者資料中設定變數。這樣做可防止多次編輯檔案時可能會發生的部分組態問題。如需有關 ECS 代理程式組態的詳細資訊，請參閱 GitHub 上的 Amazon ECS 容器代理程式。

- 或者，如果您變更 credentials-fetcher 常駐程式組態，將通訊端移至其他位置，您也可以使用變數 CREDENTIALS_FETCHER_HOST。

設定許可和祕密

針對每個應用程式和每個任務定義執行下列步驟一次。我們建議您使用授予最低權限的最佳實務，並縮減政策中使用的許可。這樣一來，每個任務都只能讀取所需的祕密。

1. (選用) 無網域 gMSA 祕密

   如果您使用執行個體未加入網域的無網域方法，請依照此步驟執行。

   您必須將以下許可作為內嵌政策新增至任務執行 IAM 角色。這樣做可讓 `credentials-fetcher` 常駐程式存取 Secrets Manager 的祕密。將 `MySecret` 替換為在 `Resource` 清單中含有祕密的 Amazon Resource Name (ARN) 的範例。

   ```
   {
       "Version": "2012-10-17",
       "Statement": [
           {
               "Effect": "Allow",
               "Action": [
                   "secretsmanager:GetSecretValue"
               ],
               "Resource": [
                   "arn:aws:ssm:aws-region:111122223333:secret:MySecret"
               ]
           }
       ]
   }
   ```

   > **Note**
   >
   > 如果您使用自己的 KMS 金鑰來加密秘密，則必須將必要的許可新增至此角色，並將此角色新增至 AWS KMS 金鑰政策。

2. 決定您要使用 SSM 參數存放區或 S3 來存放 CredSpec

   Amazon ECS 支援以下方法，在任務定義的 `credentialSpecs` 欄位中參考檔案路徑。

   如果您將執行個體加入單一網域，請在字串中在 ARN 開頭使用 `credentialspec:` 作為字首。如果您使用無網域 gMSA，請使用 `credentialspecdomainless:`。

   如需 CredSpec 的詳細資訊，請參閱 憑證規格檔案。

- Amazon S3 儲存貯體

  將憑證規格新增至 Amazon S3 儲存貯體。然後，在任務定義的 `credentialSpecs` 欄位中參考 Amazon S3 儲存貯體的 Amazon Resource Name (ARN)。

```
{
    "family": "",
    "executionRoleArn": "",
    "containerDefinitions": [
        {
            "name": "",
            ...
            "credentialSpecs": [
                "credentialspecdomainless:arn:aws:s3:::${BucketName}/
${ObjectName}"
            ],
            ...
        }
    ],
    ...
}
```

  若要讓您的任務能夠存取 S3 儲存貯體，請新增下列許可做為 Amazon ECS 任務執行 IAM 角色的內嵌政策。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor",
            "Effect": "Allow",
            "Action": [
                "s3:Get*",
                "s3:List*"
            ],
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-bucket",
                "arn:aws:s3:::amzn-s3-demo-bucket/{object}"
            ]
        }
    ]
```

```
}
```

- SSM 參數存放區參數

  將憑證規格新增至 SSM 參數存放區參數。然後，在任務定義的 `credentialSpecs` 欄位中參考 SSM 參數存放區參數的 Amazon Resource Name (ARN)。

```
{
    "family": "",
    "executionRoleArn": "",
    "containerDefinitions": [
        {
            "name": "",
            ...
            "credentialSpecs": [
                "credentialspecdomainless:arn:aws:ssm:aws-
region:111122223333:parameter/parameter_name"
            ],
            ...
        }
    ],
    ...
}
```

  若要讓您的任務能夠存取 SSM 參數存放區參數，請新增下列許可做為 Amazon ECS 任務執行 IAM 角色的內嵌政策。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ssm:GetParameters"
            ],
            "Resource": [
                "arn:aws:ssm:aws-region:111122223333:parameter/parameter_name"
            ]
        }
    ]
}
```

憑證規格檔案

Amazon ECS 使用 Active Directory 憑證規格檔案 (CredSpec)。此檔案包含用於將 gMSA 帳戶內容傳播至容器 Linux 的 gMSA 中繼資料。您會產生 CredSpec，並在任務定義的 `credentialSpecs` 欄位中參考該檔案。CredSpec 檔案不包含任何祕密。

以下是範例 CredSpec 檔案。

```
{
    "CmsPlugins": [
        "ActiveDirectory"
    ],
    "DomainJoinConfig": {
        "Sid": "S-1-5-21-2554468230-2647958158-2204241789",
        "MachineAccountName": "WebApp01",
        "Guid": "8665abd4-e947-4dd0-9a51-f8254943c90b",
        "DnsTreeName": "example.com",
        "DnsName": "example.com",
        "NetBiosName": "example"
    },
    "ActiveDirectoryConfig": {
        "GroupManagedServiceAccounts": [
            {
                "Name": "WebApp01",
                "Scope": "example.com"
            }
        ],
        "HostAccountConfig": {
            "PortableCcgVersion": "1",
            "PluginGUID": "{859E1386-BDB4-49E8-85C7-3070B13920E1}",
            "PluginInput": {
                "CredentialArn": "arn:aws:secretsmanager:aws-
region:111122223333:secret:MySecret"
            }
        }
    }
}
```

建立CredSpec

您可以在加入網域的 Windows 電腦上使用 CredSpec PowerShell 模組來建立 CredSpec。請依照 Microsoft Learn 網站上建立憑證規格中的步驟進行。

在 Fargate 上使用 gMSA 做為Linux容器

Amazon ECS 透過稱為群組受管服務帳戶 () 的特殊服務帳戶，支援 Fargate 上 Linux 容器的 Active Directory 身分驗證gMSA。

.NET Core 應用程式等以 Linux 為基礎的網路應用程式，可使用 Active Directory 來促進使用者和服務之間的身分驗證和授權管理。您可以透過設計與 Active Directory 整合並在加入網域的伺服器上執行的應用程式來使用此功能。但是，由於 Linux 容器無法加入網域，因此您需要設定 Linux 容器來執行 gMSA。

考量事項

在 Fargate 上使用 Linux gMSA容器之前，請考慮下列事項：

- 您必須執行平台 1.4 版或更新版本。
- 您可能需要加入網域的 Windows 電腦才能完成先決條件。例如，您可能需要加入網域的 Windows 電腦，才能在使用 PowerShell 的 Active Directory 中建立 gMSA。RSATActive Director PowerShell 工具僅適用於 Windows。如需詳細資訊，請參閱安裝 Active Directory 管理工具。
- 您必須使用無網域 gMSA。

  Amazon ECS 使用 Active Directory 憑證規格檔案 (CredSpec)。此檔案包含用於將 gMSA 帳戶內容傳播至容器的 gMSA 中繼資料。您可以產生 CredSpec 檔案，然後將其存放在 Amazon S3 儲存貯體中。
- 任務只能支援一個 Active Directory。

必要條件

在搭配 Amazon ECS 使用 Linux 容器功能適用的 gMSA 之前，請先完成下列各項：

- 您可以使用您希望容器存取的資源來設定 Active Directory 網域。Amazon ECS 支援下列設定：
  - AWS Directory Service Active Directory. AWS Directory Service 是託管在 Amazon EC2 上的 AWS 受管 Active Directory。如需詳細資訊，請參閱 AWS Directory Service 管理指南中的 AWS Managed Microsoft AD 入門。
  - 內部部署 Active Directory。您必須確認 Amazon ECS Linux 容器執行個體可以加入網域。如需詳細資訊，請參閱AWS Direct Connect。
- 您在 Active Directory 中有現有的gMSA帳戶，以及具有存取gMSA服務帳戶許可的使用者。如需詳細資訊，請參閱建立無網域 gMSA 的 Active Directory 使用者。

- 您有 Amazon S3 儲存貯體。如需詳細資訊，請參閱《Amazon S3 使用者指南》中的建立儲存貯體。

在 Amazon ECS 上設定能夠使用 gMSA 功能的 Linux 容器

準備基礎設施

下列步驟是執行一次的考量和設定。

- 建立無網域 gMSA 的 Active Directory 使用者

  當您使用無網域 時gMSA，容器不會加入網域。在容器上執行的其他應用程式無法使用 登入資料來存取網域。使用不同網域的任務可以在相同的容器上執行。您可以在 CredSpec 檔案中提供 AWS Secrets Manager 中的秘密名稱。祕密必須包含使用者名稱、密碼和要登入的網域。

  此功能與 gMSA support for non-domain-joined container hosts 功能類似。如需有關 Windows 功能的詳細資訊，請參閱 Microsoft Learn 網站上的 gMSA 架構和改進。

  a. 在 Active Directory 網域中設定使用者。Active Directory 中的使用者必須具有存取您在任務中使用的gMSA服務帳戶的許可。

  b. 您擁有可以解析 Active Directory 網域的 VPC 和子網路。使用指向 Active Directory 服務名稱的網域名稱，使用 DHCP 選項設定 VPC。如需有關如何設定 VPC DHCP 選項的詳細資訊，請參閱《Amazon Virtual Private Cloud 使用者指南》中的使用 DHCP 選項集。

  c. 在 中建立秘密 AWS Secrets Manager。

  d. 建立登入資料規格檔案。

設定許可和祕密

針對每個應用程式和每個任務定義執行下列步驟一次。我們建議您使用授予最低權限的最佳實務，並縮減政策中使用的許可。這樣一來，每個任務都只能讀取所需的祕密。

1. 在 Active Directory 網域中建立使用者。Active Directory 中的使用者必須具有存取您在任務中使用之 gMSA 服務帳戶的權限。
2. 在您成為 Active Directory 使用者之後，請在 中建立秘密 AWS Secrets Manager。如需詳細資訊，請參閱建立 AWS Secrets Manager 祕密。
3. 將使用者的使用者名稱、密碼和網域分別輸入到名為 `username`、`password` 和 `domainName` 的 JSON 鍵值組中。

```
{"username":"username","password":"passw0rd", "domainName":"example.com"}
```

4. 您必須將以下許可作為內嵌政策新增至任務執行 IAM 角色。這樣做可讓 `credentials-fetcher` 常駐程式存取 Secrets Manager 的祕密。將 `MySecret` 替換為在 `Resource` 清單中含有祕密的 Amazon Resource Name (ARN) 的範例。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "secretsmanager:GetSecretValue"
      ],
      "Resource": [
        "arn:aws:secretsmanager:aws-region:111122223333:secret:MySecret"
      ]
    }
  ]
}
```

> **Note**
>
> 如果您使用自己的 KMS 金鑰來加密秘密，則必須將必要的許可新增至此角色，並將此角色新增至 AWS KMS 金鑰政策。

5. 將憑證規格新增至 Amazon S3 儲存貯體。然後，在任務定義的 `credentialSpecs` 欄位中參考 Amazon S3 儲存貯體的 Amazon Resource Name (ARN)。

```
{
    "family": "",
    "executionRoleArn": "",
    "containerDefinitions": [
        {
            "name": "",
            ...
            "credentialSpecs": [
                "credentialspecdomainless:arn:aws:s3:::${BucketName}/${ObjectName}"
            ],
            ...
```

```
        }
    ],
    ...
}
```

若要讓您的任務能夠存取 S3 儲存貯體，請新增下列許可做為 Amazon ECS 任務執行 IAM 角色的內嵌政策。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor",
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:ListObject"
            ],
            "Resource": [
                "arn:aws:s3:::{bucket_name}",
                "arn:aws:s3:::{bucket_name}/{object}"
            ]
        }
    ]
}
```

憑證規格檔案

Amazon ECS 使用 Active Directory 憑證規格檔案 (CredSpec)。此檔案包含用於將 gMSA 帳戶內容傳播至容器 Linux 的 gMSA 中繼資料。您會產生 CredSpec，並在任務定義的 `credentialSpecs` 欄位中參考該檔案。CredSpec 檔案不包含任何祕密。

以下是範例 CredSpec 檔案。

```
{
    "CmsPlugins": [
        "ActiveDirectory"
    ],
    "DomainJoinConfig": {
        "Sid": "S-1-5-21-2554468230-2647958158-2204241789",
        "MachineAccountName": "WebApp01",
```

```
        "Guid": "8665abd4-e947-4dd0-9a51-f8254943c90b",
        "DnsTreeName": "example.com",
        "DnsName": "example.com",
        "NetBiosName": "example"
    },
    "ActiveDirectoryConfig": {
        "GroupManagedServiceAccounts": [
            {
                "Name": "WebApp01",
                "Scope": "example.com"
            }
        ],
        "HostAccountConfig": {
            "PortableCcgVersion": "1",
            "PluginGUID": "{859E1386-BDB4-49E8-85C7-3070B13920E1}",
            "PluginInput": {
                "CredentialArn": "arn:aws:secretsmanager:aws-
region:111122223333:secret:MySecret"
            }
        }
    }
}
```

建立 CredSpec 並將其上傳至 Amazon S3

您可以在加入網域的 Windows 電腦上使用 CredSpec PowerShell 模組來建立 CredSpec。請依照 Microsoft Learn 網站上建立憑證規格中的步驟進行。

建立登入資料規格檔案後，請將其上傳至 Amazon S3 儲存貯體。將 CredSpec 檔案複製至正在執行 AWS CLI 命令的電腦或環境中。

執行下列 AWS CLI 命令，將 CredSpec 上傳至 Amazon S3。將 *amzn-s3-demo-bucket* 取代為您的 Amazon S3 儲存貯體的名稱。您可以將檔案作為物件儲存在任何儲存貯體和位置，但必須在附加至任務執行角色的政策中允許存取該儲存貯體和位置。

對於 PowerShell，請使用下列命令：

```
$ Write-S3Object -BucketName "amzn-s3-demo-bucket" -Key "ecs-domainless-gmsa-credspec"
 -File "gmsa-cred-spec.json"
```

下列 AWS CLI 命令使用 sh和相容 Shell 所使用的反斜線接續字元。

```
$ aws s3 cp gmsa-cred-spec.json \
```

```
s3://amzn-s3-demo-bucket/ecs-domainless-gmsa-credspec
```

gMSA 使用 搭配無網域使用 Amazon ECS Windows 容器 AWS CLI

以下教學課程示範如何建立一個執行 Windows 容器的 Amazon ECS 任務，此容器具有透過 AWS CLI 存取 Active Directory 的憑證。使用無網域 gMSA 時，容器執行個體不會加入網域，執行個體上的其他應用程式無法使用憑證來存取網域，而加入不同網域的任務則可在相同的執行個體上執行。

主題

必要條件

本教學課程假設已完成下列先決條件：

- 已完成「設定以使用 Amazon ECS。」中的步驟。
- 您的 AWS 使用者具有 IAM AmazonECS_FullAccess 政策範例中指定的必要許可。

- AWS CLI 已安裝並設定最新版本的 。如需安裝或升級 的詳細資訊 AWS CLI，請參閱安裝 AWS Command Line Interface。
- 您可以使用您希望容器存取的資源來設定 Active Directory 網域。Amazon ECS 支援下列設定：
  - AWS Directory Service Active Directory. AWS Directory Service 是託管在 Amazon EC2 上的 AWS 受管 Active Directory。如需詳細資訊，請參閱 AWS Directory Service 管理指南中的 AWS Managed Microsoft AD 入門。
  - 內部部署 Active Directory。您必須確認 Amazon ECS Linux 容器執行個體可以加入網域。如需詳細資訊，請參閱AWS Direct Connect。
- 您擁有可以解析 Active Directory 網域的 VPC 和子網路。
- 您可以選擇無網域 gMSA 或將每個執行個體加入單一網域。使用無網域 gMSA 時，容器執行個體不會加入網域，執行個體上的其他應用程式無法使用憑證來存取網域，而加入不同網域的任務則可在相同的執行個體上執行。

  然後，選擇 CredSpec 的資料儲存，也可以選擇無網域 gMSA 的 Active Directory 使用者憑證的資料儲存。

  Amazon ECS 使用 Active Directory 憑證規格檔案 (CredSpec)。此檔案包含用於將 gMSA 帳戶內容傳播至容器的 gMSA 中繼資料。您會產生 CredSpec 檔案，然後將其儲存在下表中特定於容器執行個體的作業系統的 CredSpec 儲存選項之一。若要使用無網域方法，CredSpec 檔案中的選擇性區段可以在下表中的 domainless user credentials 儲存選項之一中指定憑證 (特定於容器執行個體的作業系統)。

儲存位置	Linux	Windows
Amazon Simple Storage Service	CredSpec	CredSpec
AWS Secrets Manager	無網域使用者憑證	無網域使用者憑證
Amazon EC2 Systems Manager 參數存放區	CredSpec	CredSpec，無網域使用者憑證
本機檔案	N/A	CredSpec

- （選用）AWS CloudShell 是一種工具，可為客戶提供命令列，而不需要建立自己的 EC2 執行個體。如需詳細資訊，請參閱AWS CloudShell 《使用者指南》中的什麼是 AWS CloudShell？。

步驟 1：在 Active Directory Domain Services (AD DS) 上建立和設定 gMSA 帳戶

在 Active Directory 網域上建立並設定 gMSA 帳戶。

1. 產生金鑰分佈服務根金鑰

   > **Note**
   >
   > 如果您使用的是 AWS Directory Service，則可以略過此步驟。

   KDS 根金鑰和 gMSA 許可是使用您 AWS 受管的 Microsoft AD 來設定。

   如果您尚未在網域中建立 gMSA 服務帳戶，則必須先產生金鑰分佈服務 (KDS) 根金鑰。KDS 負責建立、輪換和向授權主機發佈 gMSA 密碼。當 `ccg.exe` 需要擷取 gMSA 憑證時，將會聯絡 KDS 以擷取目前密碼。

   若要檢查是否已建立 KDS 根金鑰，請使用 `ActiveDirectory` PowerShell 模組，在網域控制器上以網域管理員權限執行下列 PowerShell cmdlet。如需有關此模組的詳細資訊，請參閱 Microsoft Learn 網站上的 ActiveDirectory 模組。

   ```
   PS C:\> Get-KdsRootKey
   ```

   如果命令傳回金鑰 ID，您可以略過此步驟的其餘部分。否則，執行下列命令以建立 KDS 根金鑰：

   ```
   PS C:\> Add-KdsRootKey -EffectiveImmediately
   ```

   儘管命令的引數 `EffectiveImmediately` 意味著金鑰會立即生效，但您需要等待 10 小時才能複製 KDS 根金鑰並可在所有網域控制器上使用。

2. 建立 gMSA 帳戶

   若要建立 gMSA 帳戶並允許 `ccg.exe` 擷取 gMSA 密碼，請從有權存取網域的 Windows 伺服器或用戶端執行下列 PowerShell 命令。將 `ExampleAccount` 替換為您想要的 gMSA 帳戶名稱。

   a.
   ```
   PS C:\> Install-WindowsFeature RSAT-AD-PowerShell
   ```

b.
```
PS C:\> New-ADGroup -Name "ExampleAccount Authorized Hosts" -SamAccountName
 "ExampleAccountHosts" -GroupScope DomainLocal
```

c.
```
PS C:\> New-ADServiceAccount -Name "ExampleAccount" -DnsHostName
 "contoso" -ServicePrincipalNames "host/ExampleAccount", "host/contoso" -
PrincipalsAllowedToRetrieveManagedPassword "ExampleAccountHosts"
```

d. 建立一個具有不會過期的永久密碼的使用者。這些登入資料會存放在 中， AWS Secrets Manager 並由每個任務用來加入網域。

```
PS C:\> New-ADUser -Name "ExampleAccount" -AccountPassword (ConvertTo-
SecureString -AsPlainText "Test123" -Force) -Enabled 1 -PasswordNeverExpires 1
```

e.
```
PS C:\> Add-ADGroupMember -Identity "ExampleAccountHosts" -Members
 "ExampleAccount"
```

f. 安裝用於在 Active Directory 中建立 CredSpec 物件的 PowerShell 模組並輸出 CredSpec JSON。

```
PS C:\> Install-PackageProvider -Name NuGet -Force
```

```
PS C:\> Install-Module CredentialSpec
```

g.
```
PS C:\> New-CredentialSpec -AccountName ExampleAccount
```

3. 將上一個命令的 JSON 輸出複製到名為 `gmsa-cred-spec.json` 的檔案中。這是 CredSpec 檔案。其在步驟 3 步驟 3：修改您的 CredSpec JSON 以包含無網域 gMSA 資訊 中使用。

步驟 2：將憑證上傳至 Secrets Manager

將 Active Directory 憑證複製至安全的憑證儲存系統中，以便每個任務都能擷取此憑證。這是無網域 gMSA 方法。使用無網域 gMSA 時，容器執行個體不會加入網域，執行個體上的其他應用程式無法使用憑證來存取網域，而加入不同網域的任務則可在相同的執行個體上執行。

此步驟使用 AWS CLI。您可以在預設 Shell (即 `bash`) 的 AWS CloudShell 中執行這些命令。

- 執行下列 AWS CLI 命令並取代使用者名稱、密碼和網域名稱，以符合您的環境。保留密碼的 ARN 以在下一個步驟中使用，步驟 3：修改您的 CredSpec JSON 以包含無網域 gMSA 資訊

以下命令使用由 sh 和相容 shell 所使用的反斜線接續字元。此命令與 PowerShell 不相容。您必須修改命令才能將其與 PowerShell 搭配使用。

```
$ aws secretsmanager create-secret \
--name gmsa-plugin-input \
--description "Amazon ECS - gMSA Portable Identity." \
--secret-string "{\"username\":\"ExampleAccount\",\"password\":\"Test123\",
\"domainName\":\"contoso.com\"}"
```

步驟 3：修改您的 CredSpec JSON 以包含無網域 gMSA 資訊

在將 CredSpec 上傳至其中一個儲存選項之前，請使用上一個步驟中 Secrets Manager 中密碼的 ARN 將資訊新增至 CredSpec。如需詳細資訊，請參閱 Microsoft Learn 網站上的未加入網域的容器主機使用案例的其他憑證規格組態。

1. 將下列資訊新增至 `ActiveDirectoryConfig` 內的 CredSpec 檔案。將 ARN 替換為上一個步驟 Secrets Manager 中的密碼。

   請注意，`PluginGUID` 值必須與下列範例程式碼片段中的 GUID 相符，且為必要值。

   ```
   "HostAccountConfig": {
         "PortableCcgVersion": "1",
         "PluginGUID": "{859E1386-BDB4-49E8-85C7-3070B13920E1}",
         "PluginInput": "{\"credentialArn\": \"arn:aws:secretsmanager:aws-
   region:111122223333:secret:gmsa-plugin-input\"}"
       }
   ```

   您也可以透過使用以下格式的 ARN，在 SSM 參數存放區中使用祕密：`\"arn:aws:ssm:aws-region:111122223333:parameter/gmsa-plugin-input\"`。

2. 修改 CredSpec 檔案之後，檔案應類似於以下範例：

   ```
   {
     "CmsPlugins": [
       "ActiveDirectory"
     ],
     "DomainJoinConfig": {
       "Sid": "S-1-5-21-4066351383-705263209-1606769140",
       "MachineAccountName": "ExampleAccount",
       "Guid": "ac822f13-583e-49f7-aa7b-284f9a8c97b6",
   ```

```
    "DnsTreeName": "contoso",
    "DnsName": "contoso",
    "NetBiosName": "contoso"
  },
  "ActiveDirectoryConfig": {
    "GroupManagedServiceAccounts": [
      {
        "Name": "ExampleAccount",
        "Scope": "contoso"
      },
      {
        "Name": "ExampleAccount",
        "Scope": "contoso"
      }
    ],
    "HostAccountConfig": {
      "PortableCcgVersion": "1",
      "PluginGUID": "{859E1386-BDB4-49E8-85C7-3070B13920E1}",
      "PluginInput": "{\"credentialArn\": \"arn:aws:secretsmanager:aws-
region:111122223333:secret:gmsa-plugin-input\"}"
    }
  }
}
```

步驟 4：將 CredSpec 上傳至 Amazon S3

此步驟使用 AWS CLI。您可以在預設 Shell (即 bash) 的 AWS CloudShell 中執行這些命令。

1. 將 CredSpec 檔案複製至正在執行 AWS CLI 命令的電腦或環境中。
2. 執行下列 AWS CLI 命令，將 CredSpec 上傳至 Amazon S3。將 MyBucket 取代為您的 Amazon S3 儲存貯體的名稱。您可以將檔案作為物件儲存在任何儲存貯體和位置，但必須在附加至任務執行角色的政策中允許存取該儲存貯體和位置。

   以下命令使用由 sh 和相容 shell 所使用的反斜線接續字元。此命令與 PowerShell 不相容。您必須修改命令才能將其與 PowerShell 搭配使用。

   ```
   $ aws s3 cp gmsa-cred-spec.json \
   s3://MyBucket/ecs-domainless-gmsa-credspec
   ```

步驟 5：(選用) 建立 Amazon ECS 叢集

預設情況下，您的帳戶會有一個名為 `default` 的 Amazon ECS 叢集。根據預設 AWS CLI，此叢集用於 、 SDKs和 AWS CloudFormation。您可以使用其他叢集來分組和組織任務和基礎結構，並為某些組態指派預設值。

您可以從 AWS Management Console、 AWS CLI SDKs或 建立叢集 AWS CloudFormation。叢集中的設定和組態不會影響 gMSA。

此步驟使用 AWS CLI。您可以在預設 Shell (即 `bash`) 的 AWS CloudShell 中執行這些命令。

```
$ aws ecs create-cluster --cluster-name windows-domainless-gmsa-cluster
```

> **Important**
>
> 如果您選擇建立自己的叢集，您必須在打算對該叢集使用的每個命令中指定 `--cluster clusterName`。

步驟 6：針對容器執行個體建立 IAM 角色

容器執行個體是在 ECS 任務中執行容器的主機電腦，例如 Amazon EC2 執行個體。每個容器執行個體都會註冊至 Amazon ECS 叢集。在您啟動 Amazon EC2 執行個體並將其註冊到叢集之前，您必須先為要使用的容器執行個體建立 IAM 角色。

若要建立容器執行個體角色，請參閱 Amazon ECS 容器執行個體 IAM 角色。預設值 `ecsInstanceRole` 具有足夠的許可來完成此教學課程。

步驟 7：建立自訂任務執行角色

Amazon ECS 可針對啟動每個任務所需的許可使用不同的 IAM 角色，而不是容器執行個體角色。此角色是任務執行角色。我們建議您建立僅具有 ECS 執行任務所需許可的任務執行角色，也稱為最低權限許可。如需有關最低權限原則的詳細資訊，請參閱《AWS Well-Architected 的架構》中的 SEC03-BP02 授予最低權限存取權。

1. 若要建立任務執行角色，請參閱 建立任務執行 角色。預設許可允許容器執行個體從 Amazon Elastic Container Registry 中提取容器映像，並從應用程式中提取 `stdout` 和 `stderr` 並記錄到 Amazon CloudWatch Logs 中。

由於角色需要本教學課程的自訂許可，因此您可以為該角色指定與 ecsTaskExecutionRole 不同的名稱。本教學課程將在後續步驟中使用 ecsTaskExecutionRole。

2. 建立自訂原則 (僅存在於此角色的內嵌政策或可重複使用的政策)，以新增下列許可。將第一個陳述式中 Resource 的 ARN 替換為 Amazon S3 儲存貯體和位置，並將第二個 Resource 替換為 Secrets Manager 中密碼的 ARN。

   如果您使用自訂金鑰在 Secrets Manager 中加密密碼，則也必須允許金鑰使用 kms:Decrypt。

   如果您使用 SSM 參數存放區而不是 Secrets Manager，您必須允許參數使用 ssm:GetParameter，而不是 secretsmanager:GetSecretValue。

```
{
"Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject"
      ],
      "Resource": "arn:aws:s3:::MyBucket/ecs-domainless-gmsa-credspec/gmsa-cred-spec.json"
    },
    {
      "Effect": "Allow",
      "Action": [
        "secretsmanager:GetSecretValue"
      ],
      "Resource": "arn:aws:secretsmanager:aws-region:111122223333:secret:gmsa-plugin-input"
    }
  ]
}
```

步驟 8：為 Amazon ECS Exec 建立任務角色

本教學課程使用 Amazon ECS Exec，透過在執行中的任務內執行命令來驗證功能。若要使用 ECS Exec，服務或任務必須開啟 ECS Exec，且任務角色 (但不是任務執行角色) 必須具有 ssmmessages 許可。如需了解所需的 IAM 政策，請參閱ECS Exec 許可。

此步驟使用 AWS CLI。您可以在預設 Shell (即 bash) 的 AWS CloudShell 中執行這些命令。

若要使用 建立任務角色 AWS CLI，請遵循下列步驟。

1. 建立稱為 ecs-tasks-trust-policy.json 的檔案，其中具有以下內容：

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

2. 建立 IAM 角色。您可以替換名稱 ecs-exec-demo-task-role，但保留以下步驟的名稱。

   以下命令使用由 sh 和相容 shell 所使用的反斜線接續字元。此命令與 PowerShell 不相容。您必須修改命令才能將其與 PowerShell 搭配使用。

```
$ aws iam create-role --role-name ecs-exec-demo-task-role \
--assume-role-policy-document file://ecs-tasks-trust-policy.json
```

   您可以刪除檔案 ecs-tasks-trust-policy.json。

3. 建立稱為 ecs-exec-demo-task-role-policy.json 的檔案，其中具有以下內容：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ssmmessages:CreateControlChannel",
                "ssmmessages:CreateDataChannel",
                "ssmmessages:OpenControlChannel",
                "ssmmessages:OpenDataChannel"
            ],
            "Resource": "*"
        }
    ]
```

```
}
```

4. 建立 IAM 政策並將其附加至上一個步驟中的角色。

   以下命令使用由 sh 和相容 shell 所使用的反斜線接續字元。此命令與 PowerShell 不相容。您必須修改命令才能將其與 PowerShell 搭配使用。

```
$ aws iam put-role-policy \
    --role-name ecs-exec-demo-task-role \
    --policy-name ecs-exec-demo-task-role-policy \
    --policy-document file://ecs-exec-demo-task-role-policy.json
```

   您可以刪除檔案 ecs-exec-demo-task-role-policy.json。

步驟 9：註冊使用無網域 gMSA 的任務定義

此步驟使用 AWS CLI。您可以在預設 Shell (即 bash) 的 AWS CloudShell 中執行這些命令。

1. 建立稱為 windows-gmsa-domainless-task-def.json 的檔案，其中具有以下內容：

```
{
  "family": "windows-gmsa-domainless-task",
  "containerDefinitions": [
    {
      "name": "windows_sample_app",
      "image": "mcr.microsoft.com/windows/servercore/iis",
      "cpu": 1024,
      "memory": 1024,
      "essential": true,
      "credentialSpecs": [
                "credentialspecdomainless:arn:aws:s3:::ecs-domainless-gmsa-
credspec/gmsa-cred-spec.json"
      ],
      "entryPoint": [
        "powershell",
        "-Command"
      ],
      "command": [
        "New-Item -Path C:\\inetpub\\wwwroot\\index.html -ItemType file -Value
 '<html> <head> <title>Amazon ECS Sample App</title> <style>body {margin-top:
 40px; background-color: #333;} </style> </head><body> <div style=color:white;text-
```

```
align:center> <h1>Amazon ECS Sample App</h1> <h2>Congratulations!</h2> <p>Your
 application is now running on a container in Amazon ECS.</p>' -Force ; C:\
\ServiceMonitor.exe w3svc"
      ],
      "portMappings": [
        {
          "protocol": "tcp",
          "containerPort": 80,
          "hostPort": 8080
        }
      ]
    }
  ],
  "taskRoleArn": "arn:aws:iam::111122223333:role/ecs-exec-demo-task-role",
  "executionRoleArn": "arn:aws:iam::111122223333:role/ecsTaskExecutionRole"
}
```

2. 執行下列命令來註冊任務定義：

   以下命令使用由 sh 和相容 shell 所使用的反斜線接續字元。此命令與 PowerShell 不相容。您必須修改命令才能將其與 PowerShell 搭配使用。

```
$ aws ecs register-task-definition \
--cli-input-json file://windows-gmsa-domainless-task-def.json
```

步驟 10：將 Windows 容器執行個體註冊至叢集

啟動 Amazon EC2 Windows 執行個體並執行 ECS 容器代理程式，以將其註冊為叢集中的容器執行個體。ECS 會在已註冊至叢集 (任務在其中啟動) 的容器執行個體上執行任務。

1. 若要啟動在 中為 Amazon ECS 設定的 Amazon EC2 Windows 執行個體 AWS Management Console，請參閱 啟動 Amazon ECS Windows 容器執行個體。在使用者資料的步驟停止。
2. 若為 gMSA，使用者資料必須先設定環境變數 ECS_GMSA_SUPPORTED，才能啟動 ECS 容器代理程式。

   對於 ECS Exec，代理程式必須以引數 -EnableTaskIAMRole 開頭。

   若要防止任務到達 EC2 IMDS Web 服務以擷取角色憑證，以保護執行個體 IAM 角色，請新增引數 -AwsvpcBlockIMDS。這只適用於使用 awsvpc 網路模式的任務。

```
<powershell>
[Environment]::SetEnvironmentVariable("ECS_GMSA_SUPPORTED", $TRUE, "Machine")
Import-Module ECSTools
Initialize-ECSAgent -Cluster windows-domainless-gmsa-cluster -EnableTaskIAMRole -
AwsvpcBlockIMDS
</powershell>
```

3. 檢閱 Summary (摘要) 面板中執行個體組態的摘要，並在準備就緒時選擇 Launch instance (啟動執行個體)。

步驟 11：驗證容器執行個體

您可以使用 AWS Management Console驗證叢集中是否存在容器執行個體。但是，gMSA 需要指示為屬性的其他功能。這些屬性不會顯示在 中 AWS Management Console，因此本教學課程會使用 AWS CLI。

此步驟使用 AWS CLI。您可以在預設 Shell (即 bash) 的 AWS CloudShell 中執行這些命令。

1. 列出叢集中的容器執行個體。容器執行個體的 ID 與 EC2 執行個體的 ID 不同。

```
$ aws ecs list-container-instances
```

輸出：

```
{
    "containerInstanceArns": [
        "arn:aws:ecs:aws-region:111122223333:container-
instance/default/MyContainerInstanceID"
    ]
}
```

例如，526bd5d0ced448a788768334e79010fd 是有效的容器執行個體 ID。

2. 使用上一個步驟中的容器執行個體 ID 來取得容器執行個體的詳細資訊。使用 ID 取代 MyContainerInstanceID。

以下命令使用由 sh 和相容 shell 所使用的反斜線接續字元。此命令與 PowerShell 不相容。您必須修改命令才能將其與 PowerShell 搭配使用。

```
$ aws ecs describe-container-instances \
     ----container-instances MyContainerInstanceID
```

請注意，輸出內容非常長。

3. 驗證 attributes 清單中是否存在一個索引鍵為 name 且值為 ecs.capability.gmsa-domainless 的物件。下列為物件的範例。

   輸出：

```
{
    "name": "ecs.capability.gmsa-domainless"
}
```

步驟 12：執行 Windows 任務

執行 Amazon ECS 任務。如果叢集中只有 1 個容器執行個體，您可以使用 run-task。如果有許多不同的容器執行個體，則相較於將置放條件限制新增至任務定義，以控制要執行此任務的容器執行個體類型，使用 start-task 並指定要在其上執行任務的容器執行個體 ID 可能會更為容易。

此步驟使用 AWS CLI。您可以在預設 Shell (即 bash) 的 AWS CloudShell 中執行這些命令。

1. 以下命令使用由 sh 和相容 shell 所使用的反斜線接續字元。此命令與 PowerShell 不相容。您必須修改命令才能將其與 PowerShell 搭配使用。

```
aws ecs run-task --task-definition windows-gmsa-domainless-task \
    --enable-execute-command --cluster windows-domainless-gmsa-cluster
```

   請注意，任務 ID 是由命令傳回。

2. 執行下列命令，以驗證任務已開始。這個命令會等待，直到任務開始時才傳回 Shell 提示字元。使用前一步驟中的任務 ID，取代 MyTaskID。

```
$ aws ecs wait tasks-running --task MyTaskID
```

步驟 13：驗證容器具有 gMSA 憑證

驗證任務中的容器具有 Kerberos 權杖。gMSA

此步驟使用 AWS CLI。您可以在預設 Shell (即 bash) 的 AWS CloudShell 中執行這些命令。

1. 以下命令使用由 sh 和相容 shell 所使用的反斜線接續字元。此命令與 PowerShell 不相容。您必須修改命令才能將其與 PowerShell 搭配使用。

```
$ aws ecs execute-command \
--task MyTaskID \
--container windows_sample_app \
--interactive \
--command powershell.exe
```

   輸出將是 PowerShell 提示字元。

2. 在容器內的 PowerShell 終端中執行以下命令。

```
PS C:\> klist get ExampleAccount$
```

   在輸出中，請注意 Principal 是您先前所建立的項目。

步驟 14：清除

完成此教學課程時，建議您清除相關聯的資源，以免未使用的資源產生費用。

此步驟使用 AWS CLI。您可以在預設 Shell (即 bash) 的 AWS CloudShell 中執行這些命令。

1. 停止任務。使用步驟 12 中的任務 ID 步驟 12：執行 Windows 任務 取代 MyTaskID。

```
$ aws ecs stop-task --task MyTaskID
```

2. 終止 Amazon EC2 執行個體。之後，叢集中的容器執行個體將會在一小時後自動刪除。

   您可以使用 Amazon EC2 主控台來尋找和終止執行個體。或者，您可以執行下列命令。若要執行命令，請在步驟 1 步驟 11：驗證容器執行個體 的 aws ecs describe-container-instances 命令輸出中尋找 EC2 執行個體識別碼。i-10a64379 是 EC2 執行個體 ID 的範例。

```
$ aws ec2 terminate-instances --instance-ids MyInstanceID
```

3. 刪除 Amazon S3 中的 CredSpec 檔案。將 MyBucket 取代為您的 Amazon S3 儲存貯體的名稱。

```
$ aws s3api delete-object --bucket MyBucket --key ecs-domainless-gmsa-credspec/
gmsa-cred-spec.json
```

4. 從 Secrets Manager 中刪除密碼。如果您改用 SSM 參數存放區，請刪除參數。

   以下命令使用由 sh 和相容 shell 所使用的反斜線接續字元。此命令與 PowerShell 不相容。您必須修改命令才能將其與 PowerShell 搭配使用。

```
$ aws secretsmanager delete-secret --secret-id gmsa-plugin-input \
    --force-delete-without-recovery
```

5. 取消註冊並刪除任務定義。透過取消註冊任務定義，您可以將其標記為非作用中狀態，以便無法用於啟動新任務。然後，您可以刪除任務定義。

   a. 透過指定版本取消註冊任務定義。ECS 會自動建立任務定義的版本，編號從 1 開始。您引用的版本與容器映像上的標籤格式相同，例如 :1。

```
$ aws ecs deregister-task-definition --task-definition windows-gmsa-domainless-
task:1
```

   b. 刪除任務定義。

```
$ aws ecs delete-task-definitions --task-definition windows-gmsa-domainless-
task:1
```

6. (選用) 如果您已建立叢集，請刪除 ECS 叢集。

```
$ aws ecs delete-cluster --cluster windows-domainless-gmsa-cluster
```

偵錯 Windows 容器的 Amazon ECS 無網域 gMSA

Amazon ECS 任務狀態

ECS 會嘗試僅啟動一次任務。任何有問題的任務均會停止，並設定為狀態 STOPPED。任務有兩種常見的問題類型。首先是無法啟動的任務。其次是應用程式已在容器之一內停止的任務。在

中 AWS Management Console，查看任務的已停止原因欄位，了解任務停止的原因。在 AWS CLI中，描述任務並查看 `stoppedReason`。如需 AWS Management Console 和 中的步驟 AWS CLI，請參閱 檢視 Amazon ECS 已停止的任務錯誤。

Windows 事件

容器中 gMSA 的 Windows 事件記錄在 `Microsoft-Windows-Containers-CCG` 日誌中，您可以在 `Logs\Microsoft\Windows\Containers-CCG\Admin`「應用程式和服務」區段的「事件檢視器」中找到。如需詳細的偵錯提示，請參閱 Microsoft Learn 網站上的 Windows 容器 gMSA 故障排除。

ECS 代理程式 gMSA 外掛程式

Windows 容器執行個體上 ECS 代理程式的 gMSA 外掛程式記錄位於下列目錄 `C:/ProgramData/Amazon/gmsa-plugin/` 中。查看此日誌，查看是否已從儲存位置 (例如 Secrets Manager) 下載無網域使用者憑證，以及是否正確讀取憑證格式。

了解如何使用 gMSAs EC2 Windows 容器 for Amazon ECS

Amazon ECS 透過名為群組受管服務帳戶 (gMSA) 的特殊類型的服務帳戶，支援對 Windows 容器的 Active Directory 身分驗證。

.NET 應用程式之類的以 Windows 為基礎的網路應用程式通常會使用 Active Directory 來促進使用者和服務之間的身分驗證和授權管理。開發人員通常會針對此目的設計其應用程式，以與 Active Directory 整合，並在已加入網域的伺服器上執行。由於以 Windows 為基礎的容器無法加入網域，因此您必須將 Windows 容器設定為與 gMSA 搭配執行。

使用 gMSA 執行的 Windows 容器依賴其主機 Amazon EC2 執行個體，以從 Active Directory 網域控制站擷取 gMSA 登入資料，並將這些資料提供給容器執行個體。如需詳細資訊，請參閱 為 Windows 容器建立 gMSA。

> **Note**
>
> Fargate 上的 Windows 容器不支援此功能。

主題

- 考量事項
- 必要條件

- 在 Amazon ECS 上設定 Windows 容器的 gMSA

考量事項

針對 Windows 容器使用 gMSA 時，應考慮下列事項：

- 針對您的容器執行個體使用 Amazon ECS 最佳化 Windows Server 2016 Full AMI 時，容器主機名稱必須與登入資料規格檔案中定義的 gMSA 帳戶名稱相同。若要指定容器的主機名稱，請使用 `hostname` 容器定義參數。如需詳細資訊，請參閱Network settings (網路設定)。
- 您可以選擇無網域 gMSA 或將每個執行個體加入單一網域。使用無網域 gMSA 時，容器執行個體不會加入網域，執行個體上的其他應用程式無法使用憑證來存取網域，而加入不同網域的任務則可在相同的執行個體上執行。

  然後，選擇 CredSpec 的資料儲存，也可以選擇無網域 gMSA 的 Active Directory 使用者憑證的資料儲存。

  Amazon ECS 使用 Active Directory 憑證規格檔案 (CredSpec)。此檔案包含用於將 gMSA 帳戶內容傳播至容器的 gMSA 中繼資料。您會產生 CredSpec 檔案，然後將其儲存在下表中特定於容器執行個體的作業系統的 CredSpec 儲存選項之一。若要使用無網域方法，CredSpec 檔案中的選擇性區段可以在下表中的 domainless user credentials 儲存選項之一中指定憑證 (特定於容器執行個體的作業系統)。

儲存位置	Linux	Windows
Amazon Simple Storage Service	CredSpec	CredSpec
AWS Secrets Manager	無網域使用者憑證	無網域使用者憑證
Amazon EC2 Systems Manager 參數存放區	CredSpec	CredSpec，無網域使用者憑證
本機檔案	N/A	CredSpec

必要條件

在搭配 Amazon ECS 使用 Windows 容器功能適用的 gMSA 之前，請先完成下列各項：

- 您可以使用您希望容器存取的資源來設定 Active Directory 網域。Amazon ECS 支援下列設定：
  - AWS Directory Service Active Directory. AWS Directory Service 是託管在 Amazon EC2 上的 AWS 受管 Active Directory。如需詳細資訊，請參閱 AWS Directory Service 管理指南中的 AWS Managed Microsoft AD 入門。
  - 內部部署 Active Directory。您必須確認 Amazon ECS Linux 容器執行個體可以加入網域。如需詳細資訊，請參閱AWS Direct Connect。
- 您在 Active Directory 中擁有現有的 gMSA 帳戶。如需詳細資訊，請參閱 為 Windows 容器建立 gMSA。
- 您已選擇使用無網域 gMSA 或託管 Amazon ECS 任務的 Amazon ECS Windows 容器執行個體必須是加入 Active Directory 的網域，並且是具有 gMSA 帳戶存取權的 Active Directory 安全群組成員。

  使用無網域 gMSA 時，容器執行個體不會加入網域，執行個體上的其他應用程式無法使用憑證來存取網域，而加入不同網域的任務則可在相同的執行個體上執行。
- 您已新增必要的 IAM 許可 所需的許可取決於您為初始憑證選擇的方法以及儲存憑證規格的方法：
  - 如果您使用無網域gMSA做為初始登入資料，Amazon EC2 執行個體角色 AWS Secrets Manager 需要 的 IAM 許可。
  - 如果您將憑證規格存放在 SSM 參數存放區中，則任務執行角色需要 Amazon EC2 Systems Manager 參數存放區的 IAM 許可。
  - 如果您將憑證規格存放在 Amazon S3 中，則任務執行角色需要 Amazon Simple Storage Service 的 IAM 許可。

在 Amazon ECS 上設定 Windows 容器的 gMSA

若要在 Amazon ECS 上設定 Windows 容器的 gMSA，您可以遵循包含設定先決條件 gMSA 使用 搭配無網域使用 Amazon ECS Windows 容器 AWS CLI 的完整教學課程。

以下章節詳細介紹了 CredSpec 組態。

主題

- 範例 CredSpec
- 無網域 gMSA 設定
- 在任務定義中參考登入資料規格檔案

範例 CredSpec

Amazon ECS 使用憑證規格檔案，其中包含將 gMSA 帳戶內容傳播到 Windows 容器所用的 gMSA 中繼資料。您可以產生登入資料規格檔案，並在任務定義的 `credentialSpec` 欄位中參考該檔案。登入資料規格檔案不包含任何秘密。

以下是範例登入資料規格檔案：

```
{
    "CmsPlugins": [
        "ActiveDirectory"
    ],
    "DomainJoinConfig": {
        "Sid": "S-1-5-21-2554468230-2647958158-2204241789",
        "MachineAccountName": "WebApp01",
        "Guid": "8665abd4-e947-4dd0-9a51-f8254943c90b",
        "DnsTreeName": "contoso.com",
        "DnsName": "contoso.com",
        "NetBiosName": "contoso"
    },
    "ActiveDirectoryConfig": {
        "GroupManagedServiceAccounts": [
            {
                "Name": "WebApp01",
                "Scope": "contoso.com"
            }
        ]
    }
}
```

無網域 gMSA 設定

我們建議使用無網域 gMSA，而不是將容器執行個體加入至單一網域。使用無網域 gMSA 時，容器執行個體不會加入網域，執行個體上的其他應用程式無法使用憑證來存取網域，而加入不同網域的任務則可在相同的執行個體上執行。

1. 在將 CredSpec 上傳至其中一個儲存選項之前，請使用 Secrets Manager 或 SSM 參數存放區中祕密的 ARN 將資訊新增 CredSpec。如需詳細資訊，請參閱 Microsoft Learn 網站上的未加入網域的容器主機使用案例的其他憑證規格組態。

   無網域 gMSA 憑證格式

以下是您 Active Directory 的無網域 gMSA 憑證的 JSON 格式。將憑證儲存在 Secrets Manager 或 SSM 參數存放區中。

```
{
    "username":"WebApp01",
    "password":"Test123!",
    "domainName":"contoso.com"
}
```

2. 將下列資訊新增至 ActiveDirectoryConfig 內的 CredSpec 檔案。將 ARN 替代為 Secrets Manager 或 SSM 參數存放區中的祕密。

   請注意，PluginGUID 值必須與下列範例程式碼片段中的 GUID 相符，且為必要值。

```
    "HostAccountConfig": {
        "PortableCcgVersion": "1",
        "PluginGUID": "{859E1386-BDB4-49E8-85C7-3070B13920E1}",
        "PluginInput": "{\"credentialArn\": \"arn:aws:secretsmanager:aws-
region:111122223333:secret:gmsa-plugin-input\"}"
    }
```

   您也可以透過使用以下格式的 ARN，在 SSM 參數存放區中使用祕密：\"arn:aws:ssm:*aws-region*:111122223333:parameter/*gmsa-plugin-input*\"。

3. 修改 CredSpec 檔案之後，檔案應類似於以下範例：

```
{
  "CmsPlugins": [
    "ActiveDirectory"
  ],
  "DomainJoinConfig": {
    "Sid": "S-1-5-21-4066351383-705263209-1606769140",
    "MachineAccountName": "WebApp01",
    "Guid": "ac822f13-583e-49f7-aa7b-284f9a8c97b6",
    "DnsTreeName": "contoso",
    "DnsName": "contoso",
    "NetBiosName": "contoso"
  },
  "ActiveDirectoryConfig": {
    "GroupManagedServiceAccounts": [
      {
```

```
        "Name": "WebApp01",
        "Scope": "contoso"
      },
      {
        "Name": "WebApp01",
        "Scope": "contoso"
      }
    ],
    "HostAccountConfig": {
      "PortableCcgVersion": "1",
      "PluginGUID": "{859E1386-BDB4-49E8-85C7-3070B13920E1}",
      "PluginInput": "{\"credentialArn\": \"arn:aws:secretsmanager:aws-
region:111122223333:secret:gmsa-plugin-input\"}"
    }
  }
}
```

在任務定義中參考登入資料規格檔案

Amazon ECS 支援以下方法，在任務定義的 `credentialSpecs` 欄位中參考檔案路徑。根據您是將容器執行個體加入單一網域，還是分別使用無網域 gMSA，您都可以針對這些選項提供 `credentialspec:` 或 `domainlesscredentialspec:`。

Amazon S3 儲存貯體

將憑證規格新增到 Amazon S3 儲存貯體，然後在任務定義的 `credentialSpecs` 欄位中參考 Amazon S3 儲存貯體的 Amazon Resource Name (ARN)。

```
{
    "family": "",
    "executionRoleArn": "",
    "containerDefinitions": [
        {
            "name": "",
            ...
            "credentialSpecs": [
                "credentialspecdomainless:arn:aws:s3:::${BucketName}/${ObjectName}"
            ],
            ...
        }
    ],
    ...
```

```
}
```

您也必須新增下列許可做為 Amazon ECS 任務執行 IAM 角色的內嵌政策，讓任務能夠存取 Amazon S3 儲存貯體。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor",
            "Effect": "Allow",
            "Action": [
                "s3:Get*",
                "s3:List*"
            ],
            "Resource": [
                "arn:aws:s3:::{bucket_name}",
                "arn:aws:s3:::{bucket_name}/{object}"
            ]
        }
    ]
}
```

SSM 參數存放區參數

將憑證規格新增至 SSM 參數存放區參數，然後在任務定義的 `credentialSpecs` 欄位中參考 SSM 參數存放區參數的 Amazon Resource Name (ARN)。

```
{
    "family": "",
    "executionRoleArn": "",
    "containerDefinitions": [
        {
            "name": "",
            ...
            "credentialSpecs": [

 "credentialspecdomainless:arn:aws:ssm:region:111122223333:parameter/parameter_name"
            ],
            ...
        }
    ],
```

```
    ...
}
```

您也必須新增下列許可做為 Amazon ECS 任務執行 IAM 角色的內嵌政策，讓任務能夠存取 SSM 參數存放區參數。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ssm:GetParameters"
            ],
            "Resource": [
                "arn:aws:ssm:region:111122223333:parameter/parameter_name"
            ]
        }
    ]
}
```

本機檔案

使用本機檔案中的登入資料規格詳細資訊，在任務定義的 `credentialSpecs` 欄位中參考檔案路徑。參照的檔案路徑必須相對於 `C:\ProgramData\Docker\CredentialSpecs` 目錄，並使用反斜線 (「\」) 作為檔案路徑分隔符號。

```
{
    "family": "",
    "executionRoleArn": "",
    "containerDefinitions": [
        {
            "name": "",
            ...
            "credentialSpecs": [
                "credentialspec:file://CredentialSpecDir\CredentialSpecFile.json"
            ],
            ...
        }
    ],
    ...
}
```

使用 EC2 Image Builder 建置自訂的 Amazon ECS 最佳化 AMIs

AWS 建議您使用 Amazon ECS 最佳化 AMIs因為這些 AMI 已預先設定了執行容器工作負載的需求和建議。有時候，您可能需要自訂 AMI 來新增其他軟體。您可以使用 EC2 Image Builder 來建立、管理和部署伺服器映像。您保留帳戶中建立的自訂映像的所有權。您可以使用 EC2 Image Builder 管道來自動化映像的更新和系統修補，或使用獨立命令來建立具有已定義組態資源的映像。

您可以為映像建立配方。配方包含父系映像，以及任何其他元件。您也可以建立分發自訂 AMI 的管道。

您可以為映像建立配方。Image Builder 映像配方是定義基礎映像和套用至基礎映像的元件的文件，以產生輸出 AMI 映像所需的組態。您也可以建立分發自訂 AMI 的管道。如需詳細資訊，請參閱 EC2 Image Builder 使用者指南中的 EC2 Image Builder 運作方式。 EC2

我們建議您使用下列其中一個 Amazon ECS 最佳化 AMIs 做為 EC2 Image Builder 中的「父映像」：

- Linux
  - Amazon ECS 最佳化 AL2023 x86
  - Amazon ECS 最佳化 Amazon Linux 2023 (arm64) AMI
  - Amazon ECS 最佳化 Amazon Linux 2 核心 5 AMI
  - Amazon ECS 最佳化 Amazon Linux 2 x86 AMI
- Windows
  - Amazon ECS 最佳化 Windows 2022 完整 x86
  - Amazon ECS 最佳化 Windows 2022 Core x86
  - Amazon ECS 最佳化 Windows 2019 完整 x86
  - Amazon ECS 最佳化 Windows 2019 Core x86
  - Amazon ECS 最佳化 Windows 2016 完整 x86

我們也建議您選取「使用最新的可用作業系統版本」。管道將為父系映像使用語意版本控制，這有助於偵測自動排程任務中的相依性更新。如需詳細資訊，請參閱 EC2 Image Builder 使用者指南中的語意版本控制。

AWS 會使用安全修補程式和新的容器代理程式版本定期更新 Amazon ECS 最佳化 AMI 映像。當您在映像配方中使用 AMI ID 做為父系映像時，您需要定期檢查父系映像的更新。如果有更新，您必須使用更新的 AMI 建立新的配方版本。這可確保您的自訂映像包含最新版本的父映像。如需如何建立工作

流程以使用新建立AMIs 自動更新 Amazon ECS 叢集中的 EC2 執行個體的詳細資訊，請參閱如何建立 AMI 強化管道並自動更新 ECS 執行個體機群。

您也可以指定透過受管 EC2 Image Builder 管道發佈的父系映像的 Amazon Resource Name (ARN)。Amazon 會透過受管管道定期發佈 Amazon ECS 最佳化 AMI 映像。這些映像可公開存取。您必須擁有正確的許可才能存取映像。當您在映像建置器配方中使用映像 ARN 而非 AMI 時，您的管道會在每次執行時自動使用最新版本的父映像。此方法不需要為每個更新手動建立新的配方版本。

使用映像 ARN 搭配基礎設施做為程式碼 (IaC)

您可以使用 EC2 Image Builder 主控台或基礎設施做為程式碼 （例如 AWS CloudFormation) 或 AWS SDK 來設定配方。當您在配方中指定父系映像時，您可以指定 EC2 AMI ID、映像建置器映像 ARN、AWS Marketplace 產品 ID 或容器映像。 AWS 會公開發佈 Amazon ECS 最佳化 AMIs 的 AMI IDs 和映像建置器映像 ARNs。以下是映像的 ARN 格式：

```
arn:${Partition}:imagebuilder:${Region}:${Account}:image/${ImageName}/${ImageVersion}
```

`ImageVersion` 具有下列格式。將`##`、`##`和`####`取代為最新的值。

```
<major>.<minor>.<patch>
```

您可以將 `major`、 `minor`和 取代`patch`為特定值，也可以使用映像的無版本 ARN，讓您的管道保持 up-to-date 最新版本的父映像。無版本 ARN 使用萬用字元格式 'x.x.x' 來表示映像版本。此方法可讓 Image Builder 服務自動解析為映像的最新版本。使用無版本 ARN 可確保您的參考一律指向可用的最新映像，簡化在部署中維護最新映像的程序。當您使用主控台建立配方時，EC2 Image Builder 會自動識別父系映像的 ARN。當您使用 IaC 建立配方時，您必須識別 ARN 並選取所需的映像版本，或使用無版本 arn 來指示最新的可用映像。我們建議您建立自動化指令碼來篩選，並僅顯示與您的條件相符的影像。下列 Python 指令碼說明如何擷取 Amazon ECS 最佳化 AMIs的清單。

指令碼接受兩個選用引數： `owner`和 `platform`，預設值分別為「Amazon」和「Windows」。擁有者引數的有效值為：`Self`、`Amazon`、 `Shared`和 `ThirdParty`。平台引數的有效值為 `Windows`和 `Linux`。例如，如果您執行的指令碼引`owner`數設為 `Amazon`，而 引數`platform`設為 `Linux`，則指令碼會產生 Amazon 發佈的影像清單，包括 Amazon ECS 最佳化的影像。

```
import boto3
import argparse

def list_images(owner, platform):
    # Create a Boto3 session
    session = boto3.Session()
```

```
    # Create an EC2 Image Builder client
    client = session.client('imagebuilder')

    # Define the initial request parameters
    request_params = {
        'owner': owner,
        'filters': [
            {
                'name': 'platform',
                'values': [platform]
            }
        ]
    }

    # Initialize the results list
    all_images = []

    # Get the initial response with the first page of results
    response = client.list_images(**request_params)

    # Extract images from the response
    all_images.extend(response['imageVersionList'])

    # While 'nextToken' is present, continue paginating
    while 'nextToken' in response and response['nextToken']:
        # Update the token for the next request
        request_params['nextToken'] = response['nextToken']

        # Get the next page of results
        response = client.list_images(**request_params)

        # Extract images from the response
        all_images.extend(response['imageVersionList'])

    return all_images

def main():
    # Initialize the parser
    parser = argparse.ArgumentParser(description="List AWS images based on owner and
 platform")

    # Add the parameters/arguments
```

```
    parser.add_argument("--owner", default="Amazon", help="The owner of the images.
 Default is 'Amazon'.")
    parser.add_argument("--platform", default="Windows", help="The platform type of the
 images. Default is 'Windows'.")

    # Parse the arguments
    args = parser.parse_args()

    # Retrieve all images based on the provided owner and platform
    images = list_images(args.owner, args.platform)

    # Print the details of the images
    for image in images:
        print(f"Name: {image['name']}, Version: {image['version']}, ARN:
 {image['arn']}")

if __name__ == "__main__":
    main()
```

搭配 使用映像 ARN AWS CloudFormation

映像建置器映像配方是一種藍圖，指定實現輸出映像的預期組態所需的父映像和元件。您可以使用 AWS::ImageBuilder::ImageRecipe 資源。將 ParentImage值設定為映像 ARN。使用所需映像的無版本 ARN，確保您的管道一律使用最新版本的映像。例如：arn:aws:imagebuilder:us-east-1:aws:image/amazon-linux-2023-ecs-optimized-x86/x.x.x。下列AWS::ImageBuilder::ImageRecipe資源定義使用 Amazon 受管映像 ARN：

```
ECSRecipe:
    Type: AWS::ImageBuilder::ImageRecipe
    Properties:
      Name: MyRecipe
      Version: '1.0.0'
      Components:
        - ComponentArn: [<The component arns of the image recipe>]
      ParentImage: "arn:aws:imagebuilder:us-east-1:aws:image/amazon-linux-2023-ecs-
optimized-x86/x.x.x"
```

如需 AWS::ImageBuilder::ImageRecipe 資源的詳細資訊，請參閱AWS CloudFormation 《使用者指南》中的 。

您可以設定 AWS::ImageBuilder::ImagePipeline 資源的 Schedule 屬性，在管道中自動建立新映像。排程包含開始條件和 Cron 表達式。如需詳細資訊，請參閱「AWS CloudFormation 使用者指南」中的 AWS::ImageBuilder::ImagePipeline。

以下AWS::ImageBuilder::ImagePipeline範例讓管道在每天 10：00AM 國際標準時間 (UTC) 執行組建。設定下列Schedule值：

- 將 PipelineExecutionStartCondition 設定為 EXPRESSION_MATCH_AND_DEPENDENCY_UPDATES_AVAILABLE。只有當父系映像或元件等相依資源在其語意版本中使用萬用字元 'x' 時，組建才會啟動。這可確保建置包含這些資源的最新更新。
- 將 ScheduleExpression 設定為 Cron 表達式 (0 10 * * ? *)。

```
ECSPipeline:
    Type: AWS::ImageBuilder::ImagePipeline
    Properties:
      Name: my-pipeline
      ImageRecipeArn: <arn of the recipe you created in previous step>
      InfrastructureConfigurationArn: <ARN of the infrastructure configuration
 associated with this image pipeline>
      Schedule:
        PipelineExecutionStartCondition:
 EXPRESSION_MATCH_AND_DEPENDENCY_UPDATES_AVAILABLE
        ScheduleExpression: 'cron(0 10 * * ? *)'
```

搭配 Terraform 使用映像 ARN

在 Terraform 中指定管道的父系映像和排程的方法與其中的方法一致 AWS CloudFormation。您可以使用 aws_imagebuilder_image_recipe 資源。將 parent_image值設定為映像 ARN。使用所需映像的無版本 ARN，確保您的管道一律使用最新版本的映像。如需詳細資訊，請參閱 Terraform 文件aws_imagebuilder_image_recipe中的 。

在 的排程組態區塊中aws_imagebuilder_image_pipeline resource，將schedule_expression引數值設定為您選擇的 Cron 表達式，以指定管道執行的頻率，並將 pipeline_execution_start_condition 設定為 EXPRESSION_MATCH_AND_DEPENDENCY_UPDATES_AVAILABLE。如需詳細資訊，請參閱 Terraform 文件aws_imagebuilder_image_pipeline中的 。

在 Amazon ECS 上使用 AWS 深度學習容器

AWS 深度學習容器提供一組 Docker 映像，用於在 TensorFlow 中訓練和提供模型，以及在 Amazon ECS 上提供 Apache MXNet （培養）。Deep Learning Containers 實現最佳化環境與 TensorFlow、Nvidia CUDA (適用於 GPU 執行個體) 和 Intel MKL (適用於 CPU 執行個體) 程式庫。Amazon ECR 中提供適用於 Deep Learning Containers 的容器映像，以便在 Amazon ECS 任務定義中進行參考。您可以使用 Deep Learning Containers 和 Amazon Elastic Inference 來降低推論成本。

若要在 Amazon ECS 上開始使用沒有彈性推論的深度學習容器，請參閱《 AWS 深度學習 AMIs 開發人員指南》中的 Amazon ECS 設定。

Amazon ECS 服務配額

下表提供 AWS 帳戶的 Amazon ECS 預設服務配額，也稱為限制。如需有關您可以與 Amazon ECS 搭配使用的其他 AWS 服務 (如 Elastic Load Balancing 和 Auto Scaling) 之服務配額的詳細資訊，請參閱 Amazon Web Services 一般參考 中的 AWS 服務配額。如需有關 Amazon ECS API 中 API 限流的資訊，請參閱請求 Amazon ECS API 限流。

Amazon ECS 服務配額

以下是 Amazon ECS 服務配額。

新 AWS 帳戶的初始配額可能較低，可能會隨著時間增加。Amazon ECS 會持續監控每個區域內的帳戶使用情況，再根據您的用量自動增加配額。針對顯示為可調整的值，您可以申請提高配額，請參閱《Service Quotas 使用指南》中的請求提高配額。

名稱	預設	可調整	描述
每個叢集的容量提供者	每個受支援的區域：20	否	能與叢集相關聯的容量提供者數量上限。
每個服務的 Classic Load Balancer	每個受支援的區域：1	否	每個服務的 Classic Load Balancer 數量上限
每個帳戶的叢集數量	每個受支援的區域：10,000	是	每個帳戶的叢集數目
每個叢集的容器執行個體數量	每個受支援的區域：5,000	否	每個叢集的容器執行個體數量
每個 start-task 的容器執行個體	每個受支援的區域：10	否	StartTask API 動作中指定的容器執行個體數量上限。
每個任務定義的容器數量	每個受支援的區域：10	否	任務定義內的容器定義數量上限。

名稱	預設	可調整	描述
ECS Exec 工作階段	每個支援的區域：1,000	否	每個容器的 ECS Exec 工作階段數量上限。
服務在 AWS Fargate 上啟動的任務速率	每個受支援的區域：500	否	Amazon ECS 服務排程器在 Fargate 上每分鐘可為每項服務佈建的任務數量上限。
由 Amazon EC2 或外部執行個體上的服務所啟動之任務比率	每個受支援的區域：500	否	Amazon ECS 服務排程器在 Amazon EC2 或外部執行個體上每分鐘可為每項服務佈建的最大任務數量。
每個任務定義系列的修訂數目	每個受支援的區域：1,000,000	否	每個任務定義系列的修訂數目上限。取消註冊或刪除任務定義修訂不會將其排除在此限制中。
每個 awsvpcConfiguration 的安全群組數量	每個受支援的區域：5	否	awsvpcConfiguration 內指定的安全群組數量上限。
每個叢集的服務數量	每個受支援的區域：5,000	否	每個叢集的服務數量上限
每個命名空間的服務	每個受支援的區域：100	是	可在命名空間內執行的服務數目上限。
每個 awsvpcConfiguration 的子網路數量	每個受支援的區域：16	否	awsvpcConfiguration 內指定的子網路數量上限。
每個資源的標籤	每個受支援的區域：50	否	每個資源的標籤數量上限。這適用於任務定義、叢集、任務和服務。

名稱	預設	可調整	描述
每個服務的目標群組	每個受支援的區域：5	否	每個服務的目標群組數量上限 (如果使用 Application Load Balancer 或 Network Load Balancer)。
任務定義大小	每個受支援的區域：64 KB	否	任務定義的大小上限 (KiB)。
每個叢集於「佈建」狀態下的任務數量	每個受支援的區域：500	否	每個叢集於「佈建」狀態下等候的任務的上限數量。此配額僅適用於使用 EC2 Auto Scaling 群組容量提供者啟動的任務。
每個 run-task 啟動的任務數量	每個受支援的區域：10	否	每個 RunTask API 動作可啟動的任務數量上限。
每個服務的任務	每個受支援的區域：5,000	否	每個服務的任務數量上限 (所需的計數)。

> **Note**
>
> 預設值是 設定的初始配額 AWS，與實際套用的配額值和最大可能的服務配額不同。如需詳細資訊，請參閱《Service Quotas 使用者指南》中的 Service Quotas 術語。

> **Note**
>
> 被設定使用 Amazon ECS 服務探索的服務有每個服務 1,000 項任務的限制。這是因為每個服務有執行個體數目的 AWS Cloud Map 服務配額。如需詳細資訊，請參閱《AWS Cloud Map》中的 Amazon Web Services 一般參考服務配額。

> **Note**
>
> 實際上，任務啟動速率還取決於其他注意事項，例如要下載和解壓縮的容器映像、運作狀態檢查以及啟用的其他整合 (例如向負載平衡器註冊任務)。與此處呈現的配額相比，您可能會看到任務啟動率的變化。這些變化是由您用於 服務的功能所造成。如需詳細資訊，請參閱Amazon ECS 服務參數的最佳實務 。

> **Note**
>
> 設定使用 Amazon ECS Service Connect 的服務有每個服務 1,000 項任務的限制。這是因為每個 AWS Cloud Map 服務的執行個體數量的服務配額所致。如需詳細資訊，請參閱《AWS Cloud Map 》中的 Amazon Web Services 一般參考服務配額。

AWS Fargate 服務配額

以下是 AWS Fargate 服務配額上的 Amazon ECS，並列在 Service Quotas 主控台AWS Fargate的服務下。

新 AWS 帳戶的初始配額可能較低，可能會隨著時間增加。Fargate 會持續監控每個區域內的帳戶使用情況，再根據您的用量自動增加配額。您也可以申請增加配額，以獲取顯示為可調整的值，請參閱《Service Quotas 使用指南》中的請求提高配額。

名稱	預設	可調整	描述
Fargate 隨需爆量啟動率	每個受支援的區域：100	是	在目前區域中，在此帳戶中以單一爆量 （爆量率） 啟動的隨需 Amazon ECS 任務或 Amazon EKS Pod 數量上限。
Fargate 隨需持續啟動率	每個受支援的區域：20	是	目前區域中此帳戶中每秒啟動的隨需 Amazon ECS

名稱	預設	可調整	描述
			任務或 Amazon EKS Pod 數量上限（持續速率）。
Fargate 隨需 vCPU 資源計數	每個受支援的區域：6	是	在當前區域中，此帳戶中以 Fargate 隨需同時執行的 Fargate vCPU 數量。
Fargate Spot 爆量啟動率	每個受支援的區域：100	是	目前區域中，在此帳戶中單一爆量（爆量率）中啟動的 Spot Amazon ECS 任務數量上限。
Fargate Spot 持續啟動率	每個受支援的區域：20	是	目前區域中此帳戶中每秒啟動的 Spot Amazon ECS 任務數量上限（持續速率）。
Fargate Spot vCPU 資源計數	每個受支援的區域：6	是	在當前區域中，此帳戶中以 Fargate Spot 同時執行的 Fargate vCPUs 數量。

Note

預設值是 設定的初始配額 AWS，與實際套用的配額值和最大可能的服務配額不同。如需詳細資訊，請參閱《Service Quotas 使用者指南》中的 Service Quotas 術語。

Note

Fargate 還強制執行 Amazon ECS 任務和 Amazon EKS Pod 啟動速率限制。如需詳細資訊，請參閱AWS Fargate 調節配額。

在 中管理您的 Amazon ECS AWS Fargate 和服務配額 AWS Management Console

Amazon ECS 已與 Service Quotas 整合，這項 AWS 服務可讓您從中央位置檢視和管理配額。如需詳細資訊，請參閱《Service Quotas 使用者指南》中的「什麼是 Service Quotas？」。

Service Quotas 可讓您輕鬆查詢 Amazon ECS 服務配額的值。

AWS Management Console

使用 AWS Management Console檢視 Amazon ECS 和 Fargate 服務配額

1. 開啟 Service Quotas 主控台，網址為 https://console.aws.amazon.com/servicequotas/。
2. 在導覽窗格中，選擇 AWS services (AWS 服務)。
3. 從 AWS services (服務) 清單中，搜尋並選取 Amazon Elastic Container Service (Amazon ECS) 或 AWS Fargate。

   在 Service quotas (服務配額) 清單中，您可以看到服務配額名稱、套用的值 (如果有的話)、AWS 預設配額，以及配額值是否可調整。
4. 若要檢視服務配額的其他資訊 (例如說明)，請選擇配額名稱。
5. (選用) 若要請求增加配額，請選取您要增加的配額、選取 Request quota increase (請求增加配額)、輸入或選取必要資訊，然後選取 Request (請求)。

若要使用 來使用服務配額， AWS Management Console 請參閱Service Quotas使用者指南。若要請求提高配額，請參閱《Service Quotas 使用者指南》https://docs.aws.amazon.com/servicequotas/latest/userguide/request-quota-increase.html中的請求提高配額。

AWS CLI

使用 AWS CLI檢視 Amazon ECS 和 Fargate 服務配額

執行下列命令，以檢視預設 Amazon ECS 配額。

```
aws service-quotas list-aws-default-service-quotas \
    --query 'Quotas[*].
{Adjustable:Adjustable,Name:QuotaName,Value:Value,Code:QuotaCode}' \
    --service-code ecs \
    --output table
```

執行下列命令，以檢視預設 Fargate 配額。

```
aws service-quotas list-aws-default-service-quotas \
    --query 'Quotas[*].
{Adjustable:Adjustable,Name:QuotaName,Value:Value,Code:QuotaCode}' \
    --service-code fargate \
    --output table
```

執行下列命令，以檢視您的套用 Fargate 配額。

```
aws service-quotas list-service-quotas \
    --service-code fargate
```

> **Note**
>
> Amazon ECS 不支援套用的配額。

如需使用 使用服務配額的詳細資訊 AWS CLI，請參閱 Service Quotas AWS CLI 命令參考。若要請求提升配額，請參閱 AWS CLI 命令參考中的 `request-service-quota-increase` 命令。

處理 Amazon ECS 服務配額和 API 限流限制

Amazon ECS 與數個 整合 AWS 服務，包括 Elastic Load Balancing AWS Cloud Map和 Amazon EC2。透過這種緊密整合，Amazon ECS 包含多種功能，例如服務負載平衡、Service Connect、任務聯網和叢集自動擴展。Amazon ECS 及其與所有維護服務配額和 API 速率限制整合 AWS 服務 的另一個 ECS，以確保一致的效能和使用率。這些服務配額也會防止意外佈建超過所需的資源，並防止可能增加帳單的惡意動作。

透過熟悉您的服務配額和 AWS API 速率限制，您可以規劃擴展工作負載，而不必擔心意外的效能降低。如需詳細資訊，請參閱請求 Amazon ECS API 的限流。

在 Amazon ECS 上擴展工作負載時，我們建議您考慮下列服務配額。

- AWS Fargate 具有配額，可限制每個 中並行執行中的任務數量 AWS 區域。Amazon ECS 上的隨需和 Fargate Spot 任務都有配額。每個服務配額也包含您在 Fargate 上執行的任何 Amazon EKS Pod。
- 對於在 Amazon EC2 執行個體上執行的任務，您可以為每個叢集註冊的 Amazon EC2 執行個體數量上限為 5，000。如果您搭配 Auto Scaling 群組容量提供者使用 Amazon ECS 叢集自動擴展，或自

行管理叢集的 Amazon EC2 執行個體，則此配額可能會成為部署瓶頸。如果您需要更多容量，您可以建立更多叢集或請求提高服務配額。

- 如果您搭配 Auto Scaling 群組容量提供者使用 Amazon ECS `Tasks in the PROVISIONING state per cluster` 叢集自動擴展，則在擴展服務時，請考慮配額。此配額是每個叢集PROVISIONING的狀態任務數量上限，容量提供者可以增加容量。當您同時啟動大量任務時，您可以輕鬆滿足此配額。其中一個範例是，如果您同時部署數十個 服務，每個服務都有數百個任務。發生這種情況時，容量提供者需要啟動新的容器執行個體，以在叢集容量不足時放置任務。當容量提供者啟動其他 Amazon EC2 執行個體時，Amazon ECS 服務排程器可能會繼續平行啟動任務。不過，此活動可能會因為叢集容量不足而受到調節。Amazon ECS 服務排程器實作退避和指數限流策略，以便在啟動新容器執行個體時重試任務置放。因此，您可能會遇到較慢的部署或橫向擴展時間。若要避免這種情況，您可以在下列其中一項中規劃服務部署。部署大量任務不需要增加叢集容量，或保留備用叢集容量以進行新的任務啟動。

除了在擴展工作負載時考慮 Amazon ECS 服務配額之外，也請考慮與 Amazon ECS AWS 服務 整合之其他 的服務配額。

Elastic Load Balancing

您可以設定 Amazon ECS 服務使用 Elastic Load Balancing，將流量平均分散到任務中。如需如何選擇負載平衡器的詳細資訊和建議的最佳實務，請參閱 使用負載平衡來分發 Amazon ECS 服務流量。

Elastic Load Balancing 服務配額

當您擴展工作負載時，請考慮下列 Elastic Load Balancing 服務配額。大多數 Elastic Load Balancing 服務配額都是可調整的，您可以在 Service Quotas 主控台中請求增加。

Application Load Balancer

當您使用 Application Load Balancer 時，視您的使用案例而定，您可能需要請求增加配額：

- `Targets per Application Load Balancer` 配額，即 Application Load Balancer 後方的目標數量。
- `Targets per Target Group per Region` 配額，即目標群組後方的目標數量。

如需詳細資訊，請參閱《Application Load Balancer 使用者指南》中的 Application Load Balancer 配額。

Network Load Balancer

您可以向 Network Load Balancer 註冊的目標數量有更嚴格的限制。使用 Network Load Balancer 時，您通常會想要啟用跨區域支援，這對於每個 Network Load Balancer 每個可用區域的目標數量`Targets per Availability Zone Per Network Load Balancer`上限有額外的擴展限制。如需詳細資訊，請參閱 Network Load Balancer 使用者指南中的 Network Load Balancer 配額。

Elastic Load Balancing API 限流

當您將 Amazon ECS 服務設定為使用負載平衡器時，目標群組運作狀態檢查必須先通過，服務才會被視為正常運作。為了執行這些運作狀態檢查，Amazon ECS 會代表您叫用 Elastic Load Balancing API 操作。如果您帳戶中有大量使用負載平衡器設定的服務，則可能會因為特別針對 `RegisterTarget`、`DeregisterTarget`和 `DescribeTargetHealth` Elastic Load Balancing API 操作的潛在限流，而使服務部署速度變慢。發生調節時，Amazon ECS 服務事件訊息中會發生調節錯誤。

如果您遇到 AWS Cloud Map API 限流，您可以聯絡 支援 以取得有關如何提高 AWS Cloud Map API 限流限制的指導。如需監控和故障診斷這類調節錯誤的詳細資訊，請參閱處理 Amazon ECS 限流問題。

彈性網路介面

隨著您的任務使用 `awsvpc` 網路模式，Amazon ECS 會為每個任務佈建唯一的彈性網路介面 (ENI)。當您的 Amazon ECS 服務使用 Elastic Load Balancing 負載平衡器時，這些網路介面也會註冊為服務中定義之適當目標群組的目標。

彈性網路介面服務配額

當您執行使用`awsvpc`網路模式的任務時，每個任務都會連接一個唯一的彈性網路介面。如果必須透過網際網路到達這些任務，請將公有 IP 地址指派給這些任務的彈性網路介面。當您擴展 Amazon ECS 工作負載時，請考慮這兩個重要的配額：

- `Network interfaces per Region` 配額，這是 中 AWS 區域 您帳戶的最大網路介面數量。
- `Elastic IP addresses per Region` 配額，這是 中彈性 IP 地址的數量上限 AWS 區域。

這兩種服務配額都是可調整的，您可以從 Service Quotas 主控台請求提高這些配額。如需詳細資訊，請參閱《Amazon Virtual Private Cloud 使用者指南》中的 Amazon VPC 服務配額。 Amazon Virtual Private Cloud

對於託管在 Amazon EC2 執行個體上的 Amazon ECS 工作負載，執行使用`awsvpc`網路模式的任務時，請考慮`Maximum network interfaces`服務配額，即每個 Amazon EC2 執行個體的網路執

行個體數量上限。此配額會限制您可以在執行個體上放置的任務數量。您無法調整配額，且無法在 Service Quotas 主控台中使用。如需詳細資訊，請參閱《Amazon EC2 使用者指南》中的每個執行個體類型的每個網路介面 IP 地址。

雖然您無法變更可連接到 Amazon EC2 執行個體的網路介面數量，但您可以使用彈性網路介面中繼功能來增加可用的網路介面數量。例如，根據預設，c5.large執行個體最多可以有三個網路介面。執行個體的主要網路介面視為一個配額。因此，您可以將額外的兩個網路介面連接到執行個體。由於使用 awsvpc 網路模式的每個任務都需要網路介面，因此您通常只能在此執行個體類型上執行兩個此類任務。這可能會導致叢集容量的使用率不足。如果您啟用彈性網路介面中繼，您可以提高網路介面密度，以在每個執行個體上放置更多任務。開啟中繼後，c5.large執行個體最多可以有 12 個網路介面。執行個體具有主要網路介面，Amazon ECS 會建立 "trunk" 網路介面並將其連接至執行個體。因此，使用此組態，您可以在執行個體上執行 10 個任務，而不是預設的兩個任務。如需詳細資訊，請參閱增加 Amazon ECS Linux 容器執行個體網路介面。

彈性網路介面 API 限流

當您執行使用awsvpc網路模式的任務時，Amazon ECS 依賴下列 Amazon EC2 APIs。每個 APIs都有不同的 API 節流。如需詳細資訊，請參閱《Amazon EC2 API 參考》中的請求 Amazon EC2 API 的限流。 Amazon EC2

- CreateNetworkInterface
- AttachNetworkInterface
- DetachNetworkInterface
- DeleteNetworkInterface
- DescribeNetworkInterfaces
- DescribeVpcs
- DescribeSubnets
- DescribeSecurityGroups
- DescribeInstances

如果在彈性網路介面佈建工作流程期間調節 Amazon EC2 API 呼叫，Amazon ECS 服務排程器會自動以指數退避重試。這些自動淘汰有時可能會導致啟動任務延遲，這會導致部署速度變慢。發生 API 限流時，您會在服務事件訊息Operations are being throttled. Will try again later.上看到訊息。如果您持續符合 Amazon EC2 API 限流，您可以聯絡 支援 以取得有關如何提高 API 限流限制的指導。如需監控和疑難排解調節錯誤的詳細資訊，請參閱處理調節問題。

AWS Cloud Map

Amazon ECS 服務探索和 Service Connect 使用 AWS Cloud Map APIs 來管理 Amazon ECS 服務的命名空間。如果您的服務有大量任務，請考慮下列建議。

AWS Cloud Map 服務配額

當 Amazon ECS `Tasks per service` 服務設定為使用服務探索或 Service Connect 時，服務任務數量上限的配額會受到 AWS Cloud Map `Instances per service`服務配額的影響，而該服務執行個體數量上限會受到影響。特別是， AWS Cloud Map 服務配額會減少您最多可以執行 1，0000 個服務任務的任務量。您無法變更 AWS Cloud Map 配額。如需詳細資訊，請參閱 AWS Cloud Map 服務配額。

AWS Cloud Map API 限流

Amazon ECS 會代表您呼叫 `ListInstances`、`RegisterInstance`、`GetInstancesHealthStatus`和 `DeregisterInstance` AWS Cloud Map APIs。當您啟動任務時，它們有助於探索服務並執行運作狀態檢查。當同時使用服務探索與大量任務的多個服務同時部署時，這可能會導致超過 AWS Cloud Map API 限流限制。發生這種情況時，您可能會在 Amazon ECS 服務事件訊息`Operations are being throttled. Will try again later`中看到以下訊息：部署速度較慢和任務啟動速度較慢。 AWS Cloud Map 不會記錄這些 APIs 的限流限制。如果您遇到這些調節，您可以聯絡 支援 ，以取得提高 API 調節限制的指導。如需有關監控和疑難排解此類調節錯誤的詳細資訊，請參閱 處理 Amazon ECS 限流問題。

Amazon ECS API 參考

除了 AWS Management Console 和 AWS Command Line Interface (AWS CLI)，Amazon ECS 還提供了一個 API。您可以使用 API，將管理 Amazon ECS 資源的任務自動化。

- 如需 Amazon ECS 資源的 API 操作清單，請參閱 Amazon ECS 資源執行的動作。
- 如需依字母排序的 API 操作清單，請參閱動作。
- 如需依字母排序的資料類型清單，請參閱資料類型。
- 如需常用查詢參數的清單，請參閱常用參數。
- 如需錯誤碼的說明，請參閱常見錯誤。

如需有關的詳細資訊 AWS CLI，請AWS Command Line Interface 參閱 Amazon ECS 的參考資料。

文件歷史紀錄

下表說明 Amazon Elastic Container Service 開發人員指南的主要更新及和新功能。我們也會經常更新文件，以處理您傳送給我們的意見回饋。

變更	描述	日期
在 Amazon ECS和 Fargate 上支援三種類型的故障注入。	您現在可以搭配 Amazon 和 Fargate 使用網路黑洞連接埠、網路延遲ECS和網路封包遺失故障注入。如需詳細資訊，請參閱使用 AWS Fault Injection Service 測試 Amazon ECS工作負載。	2024 年 12 月 19 日
支援具有增強可觀測性的 CloudWatch Container Insights。	您可以使用具有增強可觀測性的 CloudWatch Container Insights。如需詳細資訊，請參閱使用ECS具有增強可觀測性的 Container Insights 監控 Amazon 容器。	2024 年 12 月 2 日
支援預測擴展。	您可以使用預測擴展來預測未來需求，並視需要主動增加服務中的任務。如需詳細資訊，請參閱使用歷史模式透過預測擴展來擴展 CloudWatch 服務。	2024 年 11 月 21 日
支援服務重新平衡。	您現在可以讓 Amazon ECS自動平衡跨可用區域的服務任務。如需詳細資訊，請參閱在可用區域之間平衡 Amazon ECS 服務。	2024 年 11 月 19 日
支援將容器映像標籤的解析度設定為映像摘要。	您現在可以設定容器映像標籤的解析度，以摘要屬於服務之任務中的每個容器。如需詳細資訊，請參閱 容器映像解析度 和 `versionConsistency` 。	2024 年 11 月 19 日
新增`AmazonECSInfrastructureRolePolicyForVpcLattice` IAM政策	此`AmazonECSInfrastructureRolePolicyForVpcLattice` 政策可讓您存取代表您管理 Amazon ECS工作負載中的 VPC Lattice 功能所需的其他 AWS 服務資源。	2024 年 11 月 18 日
VPC Lattice 現在可用於 Amazon ECS服務。	您現在可以使用 VPC Lattice 標準化您的 service-to-service連線能力、安全性和可觀測性。如需詳細資訊，	2024 年 11 月 18 日

變更	描述	日期
	請參閱使用 Amazon VPC Lattice 連線、觀察和保護您的 Amazon ECS服務。	
支援使用服務部署和服務修訂的服務歷史記錄。	您可以使用服務部署和服務修訂來檢視部署歷史記錄。如需詳細資訊，請參閱使用 Amazon 服務部署和 Amazon 服務修訂檢視ECS服務歷史記錄。 ECS	2024 年 10 月 30 日
支援使用 Windows 作業系統啟動EC2類型的 Amazon EBS磁碟區。	Amazon ECS支援連接至 Windows 作業系統EC2執行個體上執行之任務的 Amazon EBS磁碟區。如需詳細資訊，請參閱搭配 Amazon 使用 Amazon EBS磁碟區 ECS。	2024 年 10 月 24 日
將許可新增至 AmazonECSInfrastructureRolePolicyForVolumes。	`AmazonECSInfrastructureRolePolicyForVolumes` 政策已更新，讓客戶能夠從快照建立 Amazon EBS磁碟區。如需詳細資訊，請參閱 AmazonECSInfrastructureRolePolicyForVolumes。	2024 年 10 月 10 日
Fargate Spot 適用於 Linux ARM。	Amazon ARM已新增對 Linux 上的 Fargate Spot 的支援ECS。	2024 年 9 月 6 日
Amazon ECS任務中個別容器的容器重新啟動政策。	您可以為任務定義中定義的必要和非必要容器啟用重新啟動政策，以更快地克服暫時性故障並維持任務可用性。如需詳細資訊，請參閱使用容器重新啟動政策重新啟動 Amazon ECS 任務中的個別容器。	2024 年 8 月 15 日
新增許可至 the section called “AmazonECS_FullAccess”	`AmazonECS_FullAccess` 已更新政策，以新增名為 IAM之角色的`iam:PassRole` 許可`ecsInfrastructureRole` 。這是 建立的預設IAM角色 AWS Management Console ，旨在用作ECS基礎設施角色，允許 Amazon ECS管理連接到ECS任務的 Amazon EBS磁碟區。	2024 年 8 月 13 日
針對 ECS Anywhere 新增 Amazon Linux 2023、CentOS Stream 9、Fedora 40 和 Ubuntu 24 支援	Amazon Linux 2023、CentOS Stream 9、Fedora 40 和 Ubuntu 24 作業系統的支援已新增至 ECS Anywhere。如需詳細資訊，請參閱支援的作業系統和系統架構。	2024 年 7 月 23 日

變更	描述	日期
使用滾動更新 (ECS) 部署類型的服務的容器映像解析。	為了確保使用滾動更新部署控制器的服務中的所有任務都使用相同的容器映像，Amazon 會將任務定義中指定的容器映像名稱和任何映像標籤ECS解析為容器映像摘要。如需詳細資訊，請參閱容器映像解析度。	2024 年 7 月 10 日
針對 ECS Anywhere 新增 Debian 11 和 Debian 12 支援	已將 Debian 11 和 Debian 12 作業系統的支援新增至 ECS Anywhere。如需詳細資訊，請參閱支援的作業系統和系統架構。	2024 年 3 月 28 日
gMSA for Linux Containers on Fargate 支援	Amazon 透過稱為群組受管服務帳戶 (g) 的特殊服務帳戶，ECS支援 Fargate 上 Linux 容器的 Active Directory 身分驗證MSA。如需詳細資訊，請參閱使用 gMSA for Linux Fargate 上的 容器。	2024 年 3 月 5 日
CloudWatch 針對連接至任務的 Amazon EBS磁碟區新增的指標	Amazon ECS 現在會針對已連接 Amazon EBS 磁碟區的任務，發佈 Amazon EBS儲存使用率的 CloudWatch 指標。如需詳細資訊，請參閱 Amazon ECS CloudWatch 指標。	2024 年 2 月 8 日
Service Connect TLS	您現在可以TLS搭配 Service Connect 使用 。	2024 年 1 月 22 日
Service Connect TLS受管政策	已新增 A mazonECSInfrastructureRolePolicyForServiceConnectTransportLayerSecurity政策。	2024 年 1 月 22 日
Service Connect 逾時組態更新	Service Connect 逾時組態現在可以更新，並包含兩個選用參數 - `idleTimeout` 和 `perRequestTimeout` 。	2024 年 1 月 22 日
Amazon ECS受管執行個體耗盡	您可以使用 Amazon ECS受管執行個體耗盡，以促進 Amazon ECS執行個體的正常終止。	2024 年 1 月 19 日
針對 ECS Anywhere 新增 Ubuntu 22 支援	對 Ubuntu 22 作業系統的支援已新增至 ECS Anywhere。如需詳細資訊，請參閱支援的作業系統和系統架構。	2024 年 1 月 16 日

變更	描述	日期
新增AmazonECS Infrastructure RolePolicyFor Volumes IAM政策	已新增 AmazonECSInfrastructureolePolicyForVolumes 。此政策會授予 Amazon ECS 呼叫 AWS API 以管理與 Amazon ECS工作負載相關聯的 Amazon EBS磁碟區所需的許可。	2024 年 1 月 11 日
Amazon ECS任務的 Amazon EBS資料磁碟區	您可以在部署期間為每個任務設定 1 個 Amazon EBS資料磁碟區，以連接至 ECS服務管理的獨立 Amazon ECS任務或任務。在部署時設定磁碟區可讓您建立可重複使用的任務定義，而不受特定磁碟區類型或設定的限制。Amazon EBS磁碟區為資料密集型容器化工作負載提供高可用性、經濟實惠、耐用、高效能的區塊儲存。	2024 年 1 月 11 日
Amazon ECS Classic 主控台已達到生命週期結束	Amazon ECS主控台已達到生命週期結束。	2023 年 12 月 4 日
更新的政策	受管IAM政策已更新為新的events許可，以及額外的 autoscaling 和 mazonECSServiceRolePolicyautoscaling-plans 許可。	2023 年 12 月 4 日
執行期監控支援	您可以使用執行期監控來監控 Amazon ECS工作負載，以識別惡意或未經授權的行為。如需詳細資訊，請參閱執行期監控。	2023 年 11 月 26 日
更新的政策	受AmazonECSServiceRolePolicy 管IAM政策已更新，以允許存取 AWS Cloud Map DiscoverInstancesRevision API。	2023 年 10 月 4 日
AWS Fargate 任務淘汰組態	您可以設定淘汰 Fargate 任務之前的等待期。如需詳細資訊，請參閱AWS Fargate 任務維護。	2023 年 9 月 5 日
中的其他任務定義參數 AWS Fargate	AWS Fargate 在 Linux 平台版本 systemControls 中新增對 pidMode和 的支援1.4.0。如需詳細資訊，請參閱任務定義。	2023 年 8 月 9 日

變更	描述	日期
Amazon ECS主控台任務定義頁面重新設計	Amazon ECS主控台中的任務定義頁面已重新設計，並包含其他選項。如需詳細資訊，請參閱使用主控台建立任務定義。	2023 年 7 月 26 日
Fargate 支援使用 Seekable OCI索引的延遲載入	AWS Fargate 正在引入 Seekable OCI(SOCI) 索引。使用 SOCI時，容器只會在映像提取上花幾秒鐘的時間，然後才能啟動，在背景下載映像時，提供環境設定和應用程式實例化的時間。如需詳細資訊，請參閱使用 Seekable OCI(SOCI) 延遲載入容器映像。	2023 年 7 月 17 日
改善對 Linux 和 Windows 上的 gMSA 的支援	任務定義具有適用於 Linux 和 Windows 的 gMSA 新`credentialSpecs` 欄位。已新增 Windows 上無網域 gMSA 的全新完整教學課程，請參閱教學課程：使用搭配無網域 gMSA 使用 Windows 容器 AWS CLI。如需詳細資訊，請參閱使用 gMSAs for Linux Containers 和使用 gMSAs for Windows Containers。	2023 年 7 月 14 日
改善ECS客服人員版本文件	已更新 Amazon ECS Agent 版本的文件。我們建議您使用 `v20.10.13` 或更新版本的 Docker 搭配最新版本的 Amazon ECS容器代理程式。代理程式的發行版本和變更可在 上使用 GitHub。如需詳細資訊，請參閱 Amazon ECS Linux 容器代理程式版本。	2023 年 6 月 20 日
更新 Fargate ARM64支援的 區域可用性	Fargate ARM64支援的 區域可用性已更新。如需詳細資訊，請參閱考量事項。	2023 年 6 月 19 日
改善叢集自動擴展文件	Amazon EC2 Auto Scaling 的 Amazon ECS擴展文件在準確性和可讀性上有顯著改善。如需詳細資訊，請參閱 Amazon ECS叢集自動擴展。	2023 年 5 月 4 日
資源建立的標記授權。	使用者必須擁有建立資源的動作許可，例如 `ecsCreateCluster` 。當您建立資源並指定該資源的標籤時， AWS 會執行額外的授權，以驗證是否有建立標籤的許可。如需詳細資訊，請參閱標記授權和建立時授予標記資源的許可。	2023 年 4 月 18 日

變更	描述	日期
支援 上的 Linux 容器 gMSA EC2	您可以使用 gMSA 驗證 上適用於 Linux 容器的 Active DirectoryEC2。如需詳細資訊，請參閱使用 gMSAs for Linux Containers。	2023 年 4 月 14 日
支援 AWS Fargate上 Windows 容器的暫時性儲存。	您可以使用 AWS Fargate上 Windows 容器的暫時性儲存。如需詳細資訊，請參閱 Fargate 任務儲存。	2023 年 4 月 14 日
AWS Cost Management 支援任務層級CUR資料	您可以在「成本與用量報告」中開啟任務層級的成本和資源用量。這會為在 AWS Fargate 和 上執行的任務新增分割成本分配資料EC2。如需詳細資訊，請參閱任務層級成本和用量報告。	2023 年 4 月 12 日
Amazon Linux 2023 Amazon ECS最佳化 AMI	您可以在 Amazon Linux 2023 Amazon ECS最佳化 上部署工作負載AMI。如需詳細資訊，請參閱Amazon ECS 最佳化 Linux AMIs。	2023 年 4 月 10 日
AWS Fargate 聯邦資訊處理標準 (FIPS) 140	您可以在 Amazon ECS 上 AWS Fargate 以符合聯邦資訊處理標準 (FIPS) 140 的方式部署工作負載。如需詳細資訊，請參閱AWS Fargate 聯邦資訊處理標準 (FIPS-140)。	2023 年 4 月 10 日
任務定義刪除	您可以使用 Amazon ECS主控台、 SDK和 刪除任務定義 AWS CLI。如需詳細資訊，請參閱使用主控台刪除任務定義修訂版和任務定義。	2023 年 2 月 24 日
AWS Fargate Compute Optimizer 中的服務建議	AWS Compute Optimizer 會根據在 AWS Fargate 上的Amazon ECS服務中執行任務的使用率，產生任務和容器大小建議。如需詳細資訊，請參閱在 Fargate 上檢視Amazon ECS服務的建議。	2023 年 1 月 27 日
Amazon ECS主控台	新的 Amazon ECS主控台現在是預設主控台。如需詳細資訊，請參閱新增 Amazon ECS主控台。	2023 年 1 月 19 日
已更新AmazonECS_FullAccess IAM政策	AmazonECS_FullAccess IAM 政策已更新，以包含在建立期間新增標籤至負載平衡器的許可。如需詳細資訊，請參閱AmazonECS_FullAccess。	2023 年 1 月 4 日

變更	描述	日期
使用 CloudWatch 警示來偵測 Amazon ECS服務部署失敗	您可以設定 Amazon 在偵測到指定的 CloudWatch 警示已進入 ALARM 狀態時，ECS將部署設定為失敗。如需詳細資訊，請參閱the section called “滾動更新部署”。	2022 年 12 月 19 日
支援容器連接埠映射	您可以在綁定到動態映射主機連接埠範圍的容器上設定連接埠號碼範圍。如需詳細資訊，請參閱the section called “連接埠映射”。	2022 年 12 月 15 日
Amazon ECS Service Connect 的一般可用性	此功能新增由 Amazon 服務部署所控制的 ECS 服務探索和服務網格。如需詳細資訊，請參閱the section called “ Service Connect”。	2022 年 11 月 27 日
已更新任務定義的新 Amazon ECS主控台體驗	新的 Amazon ECS主控台體驗現在包含任務定義JSON編輯器。如需詳細資訊，請參閱the section called “使用主控台建立任務定義”。	2022 年 10 月 27 日
已更新任務定義的新 Amazon ECS主控台體驗	新的 Amazon ECS主控台體驗現在包含任務定義JSON編輯器。如需詳細資訊，請參閱the section called “使用主控台建立任務定義”。	2022 年 10 月 27 日
新的 Amazon ECS主控台體驗已更新	新的 Amazon ECS主控台體驗已更新為額外的服務和任務參數。如需詳細資訊，請參閱 the section called “建立服務” 和 the section called “將應用程式執行為任務”。	2022 年 10 月 7 日
任務中繼資料端點版本 4 中的新資訊	任務中繼資料端點第 4 版現在包含 VPC ID 和服務名稱。如需詳細資訊，請參閱the section called “任務中繼資料端點第 4 版”。	2022 年 10 月 7 日
新的任務定義大小	Amazon ECS on Fargate 現在支援 8 vCPU 和 16 vCPU 任務大小。如需詳細資訊，請參閱the section called “任務大小”。	2022 年 9 月 16 日
ECS CLI 已封存的頁面	ECS CLI 文件已封存。我們建議您使用 AWS Copilot 滿足您的命令列工具需求。如需詳細資訊，請參閱使用 AWS Copilot 命令列介面建立 Amazon ECS 資源。	2022 年 9 月 15 日

變更	描述	日期
新的 Fargate 配額	Fargate 正在從任務計數型配額轉換為 v CPU型配額。如需詳細資訊，請參閱the section called “AWS Fargate 服務配額”。	2022 年 9 月 8 日
支援 Amazon EC2 Auto Scaling 暖集區。	您現在可以使用 Amazon EC2 Auto Scaling 暖集區更快地擴展應用程式並節省成本。如需詳細資訊，請參閱為您的 Amazon ECS Auto Scaling 群組設定預先初始化的執行個體。	2022 年 3 月 23 日
支援 ECS Anywhere 中的 Windows 執行個體。	ECS Anywhere 現在支援 Windows 執行個體。如需詳細資訊，請參閱外部啟動類型的 Amazon ECS 叢集。	2022 年 3 月 3 日
新增外部執行個體的 ECS Exec 支援	ECS 外部執行個體現在支援 Exec。如需詳細資訊，請參閱使用 ECS Exec 監控 Amazon ECS 容器。	2022 年 1 月 24 日
已更新新的 Amazon ECS 主控台體驗	新的 Amazon ECS主控台體驗支援建立和刪除叢集、更新任務定義，以及取消註冊任務定義。如需詳細資訊，請參閱 為 Fargate 啟動類型建立 Amazon ECS 叢集、刪除 Amazon ECS 叢集、使用主控台更新 Amazon ECS 任務定義 和 使用主控台取消註冊 Amazon ECS 任務定義修訂。	2021 年 12 月 8 日
已更新新的 Amazon ECS 主控台體驗	新的 Amazon ECS主控台體驗支援建立任務定義。如需詳細資訊，請參閱使用主控台建立 Amazon ECS 任務定義。	2021 年 11 月 23 日
Amazon ECS支援 Linux 的 64 位元ARM架構。	Amazon ECS支援 Linux 作業系統的 64 位元ARMCPU架構。如需詳細資訊，請參閱the section called “64 位元 ARM 工作負載的任務定義”。	2021 年 11 月 23 日
Amazon 對 fluentd log-driver-buffer-limit 選項的 ECS支援	Amazon ECS支援流利`log-driver-buffer-limit` 選項。如需詳細資訊，請參閱將 Amazon ECS 日誌傳送至 AWS 服務或 AWS Partner。	2021 年 11 月 22 日

變更	描述	日期
Amazon ECS最佳化 Linux AMI建置指令碼	Amazon ECS已開放原始碼建置指令碼，用於建置 Amazon ECS最佳化 的 Linux 變體AMI。如需詳細資訊，請參閱Amazon ECS 最佳化 Linux AMI 建置指令碼。	2021 年 11 月 19 日
容器執行個體運作狀態	Amazon ECS 新增對容器執行個體運作狀態監控的支援。如需詳細資訊，請參閱監控 Amazon ECS 容器執行個體運作狀態。	2021 年 11 月 10 日
支援 Windows Amazon ECS Exec	Amazon ECS Exec 支援 Windows。如需詳細資訊，請參閱使用 ECS Exec 監控 Amazon ECS 容器。	2021 年 11 月 1 日
支援 Fargate 上的 Windows 容器。	Amazon ECS支援 Fargate 上的 Windows 容器。如需詳細資訊，請參閱Fargate Windows 平台版本變更日誌。	2021 年 10 月 28 日
GPU 支援 Amazon ECS Anywhere 上的外部執行個體	Amazon ECS支援在任務定義中指定在外部執行個體上執行的任務GPU需求。如需詳細資訊，請參閱 GPU 工作負載的 Amazon ECS 任務定義 和 將外部執行個體註冊至 Amazon ECS 叢集。	2021 年 10 月 8 日
支援 Windows 上的 awsvpc 網路模式	Amazon ECS支援 Windows 上的awsvpc網路模式。如需詳細資訊，請參閱為 Amazon ECS 任務配置網路介面。	2021 年 7 月 15 日
Bottlerocket 正式發行	Amazon ECS支援 Bottlerocket 作業系統的 Amazon ECS最佳化AMI變體以 的形式提供AMI。如需詳細資訊，請參閱Amazon ECS 最佳化 Bottlerocket AMI。	2021 年 6 月 30 日
Amazon ECS排程任務更新	建立觸發 Amazon ECS排程任務的規則時，Amazon 新增對其他參數的 EventBridge 支援。	2021 年 6 月 25 日
AWS Amazon 的 受管政策 ECS	Amazon ECS 新增了服務連結角色的 AWS 受管政策文件。如需詳細資訊，請參閱AWS Amazon Elastic Container Service 的 受管政策。	2021 年 6 月 8 日

變更	描述	日期
入門 AWS CDK	新增了將 AWS CDK 與 Amazon 搭配使用的入門指南 ECS。如需詳細資訊，請參閱使用 建立 Amazon ECS 資源 AWS CDK。	2021 年 5 月 27 日
Amazon ECS Anywhere	Amazon ECS已新增向叢集註冊現場部署伺服器或虛擬機器 (VM) 的支援。如需詳細資訊，請參閱外部啟動類型的 Amazon ECS 叢集。	2021 年 5 月 25 日
Amazon ECS最佳化 Windows Server 20H2 核心 AMI	Amazon ECS已新增對適用於 Windows Server 20H2 Core 的新 Windows Amazon ECS最佳化AMI變體的支援。如需詳細資訊，請參閱Amazon ECS 最佳化 Linux AMIs。	2021 年 4 月 19 日
Amazon ECS Exec	Amazon ECS 發行了名為 ECS Exec 的新除錯工具。如需詳細資訊，請參閱使用 ECS Exec 監控 Amazon ECS 容器。	2021 年 3 月 15 日
VPC 端點政策支援	Amazon ECS 現在支援VPC端點政策。如需詳細資訊，請參閱為 Amazon ECS 建立 VPC 端點政策。	2021 年 1 月 11 日
全新主控台體驗	Amazon ECS發行了新的主控台體驗，支援建立或更新服務或執行獨立任務。如需詳細資訊，請參閱 使用 主控台建立 Amazon ECS 服務 和 以 Amazon ECS 任務執行應用程式。	2020 年 12 月 28 日
容量提供者更新	Amazon ECS 新增了更新現有 Auto Scaling 群組容量提供者的支援。	2020 年 11 月 23 日
ECS 現在支援 Amazon FSx for Windows File Server for Windows 任務	Amazon ECS 新增了在 Windows 任務定義中指定 Amazon FSx for Windows File Server 磁碟區的支援。如需詳細資訊，請參閱搭配 Amazon ECS 使用 FSx for Windows File Server 磁碟區。	2020 年 11 月 11 日
VPC 已新增雙堆疊模式支援	Amazon ECS 新增了對使用 awsvpc 網路模式的VPC 雙堆疊模式下使用 與任務的支援，該模式可支援 IPv6 地址。如需詳細資訊，請參閱在雙堆疊模式下使用 VPC。	2020 年 11 月 5 日

變更	描述	日期
任務中繼資料端點 v4 更新	Amazon 已將ECS其他中繼資料新增至任務中繼資料端點 v4 輸出。如需詳細資訊，請參閱Amazon ECS 任務中繼資料端點第 4 版。	2020 年 11 月 5 日
支援 Local Zones 和 Wavelength 區域	Amazon ECS 新增支援 Local Zones 和 Wavelength Zones 中的工作負載。如需詳細資訊，請參閱共用子網路、本機區域和 Wavelength 區域中的 Amazon ECS 應用程式。	2020 年 9 月 4 日
Bottlerocket 的 Amazon ECS變體 AMI	Bottlerocket 是以 Linux 為基礎的開放原始碼作業系統，由 專為執行容器 AWS 而建置。Amazon ECS最佳化的 Bottlerocket 作業系統AMI變體作為 提供AMI，您可以在啟動 Amazon ECS容器執行個體時使用。如需詳細資訊，請參閱Amazon ECS 最佳化 Bottlerocket AMI。	2020 年 8 月 31 日
任務中繼資料端點第 4 版已更新網路速度統計資訊	任務中繼資料端點第 4 版已更新，為使用至少執行容器代理程式版本之 Amazon EC2執行個體上託管的 `awsvpc`或 `bridge` 網路模式`1.43.0`的 Amazon ECS任務提供網路速率統計資料。如需詳細資訊，請參閱Amazon ECS 任務中繼資料端點第 4 版。	2020 年 8 月 10 日
Fargate 用量指標	AWS Fargate 提供 CloudWatch 用量指標，可讓您查看 Fargate 隨需和 Fargate Spot 資源的帳戶用量。如需詳細資訊，請參閱用量指標。	2020 年 8 月 3 日
AWS Copilot 0.1.0 版	新的 AWS Copilot CLI已推出，提供高階命令，以簡化ECS從本機開發環境在 Amazon 上建立、建立、發行和管理容器化應用程式。如需詳細資訊，請參閱使用 AWS Copilot 命令列介面建立 Amazon ECS 資源。	2020 年 7 月 9 日
AWS Fargate 平台版本棄用排程	已新增 Fargate 平台版本取代排程。如需詳細資訊，請參閱AWS Fargate Linux 平台版本棄用。	2020 年 7 月 8 日
AWS Fargate 區域擴展	Amazon ECS on AWS Fargate 已擴展至歐洲（米蘭）區域。	2020 年 6 月 25 日

變更	描述	日期
Amazon ECS最佳化 Amazon Linux 2 (Neuron) AMI已發行	Amazon ECS發行了適用於推論工作負載AMI的 Amazon ECS最佳化 Amazon Linux 2 (Neuron)。 如需詳細資訊，請參閱Amazon ECS 最佳化 Linux AMIs。	2020 年 6 月 24 日
新增對刪除容量提供者的支援	Amazon ECS 新增了刪除 Auto Scaling 群組容量提供者的支援。	2020 年 6 月 11 日
AWS Fargate 平台 1.4.0 版更新	自 2020 年 5 月 28 日起，任何使用 平台 1.4.0 版啟動的新 Fargate 任務，都會使用 AWS Fargate受管加密金鑰，以 AES-256 加密演算法加密其 20 GB 暫時性儲存。如需詳細資訊，請參閱Amazon ECS 的 Fargate 任務暫時性儲存。	2020 年 5 月 28 日
環境變數檔案支援	新增在工作定義中指定環境變數檔案的支援，可讓您大批新增環境變數至容器。如需詳細資訊，請參閱將個別環境變數傳遞至 Amazon ECS 容器。	2020 年 5 月 18 日
AWS Fargate 區域擴展	AWS Amazon 的 Fargate ECS 已擴展到非洲（開普敦）區域。	2020 年 5 月 11 日
服務配額已更新	下列服務配額已更新： • 每個帳戶的叢集已從 2,000 提高到 10,000。 如需詳細資訊，請參閱Amazon ECS 服務配額。	2020 年 4 月 17 日

變更	描述	日期
AWS Fargate 平台 1.4.0 版	AWS Fargate 平台 1.4.0 版已發行，其中包含下列功能： • 新增了對使用 Amazon EFS 檔案系統磁碟區進行持久性任務儲存的支援。如需詳細資訊，請參閱搭配 Amazon ECS 使用 Amazon EFS 磁碟區。 • 暫時性任務儲存體已增加至 20 GB。如需詳細資訊，請參閱 Amazon ECS 的 Fargate 任務暫時性儲存。 • 進出任務的網路流量行為已更新。從 平台 1.4 版開始，所有 Fargate 任務都會收到單一彈性網路介面（稱為任務 ENI），而所有網路流量都會流經 ENI中的該介面VPC，而且您可以透過VPC流程日誌查看。如需詳細資訊，請參閱 Fargate 啟動類型的 Amazon ECS任務聯網選項。 • 任務ENIs新增巨型訊框的支援。網路介面會設定最大傳輸單位 (MTU)，這是單一影格中最大承載的大小。越大MTU，應用程式承載可以容納在單一影格內越多，可減少每個影格的額外負荷並提高效率。當您的任務與目的地之間的網路路徑支援巨型訊框時，支援巨型訊框可降低額外負荷，例如保留在 中的所有流量VPC。 • CloudWatch Container Insights 將包含 Fargate 任務的網路效能指標。如需詳細資訊，請參閱使用具有增強可觀測性的 Container Insights 監控 Amazon ECS 容器。 • 已新增任務中繼資料端點 v4 的支援，該端點為您的 Fargate 任務提供其他資訊，包括任務的網路統計資料，以及執行中任務所在的可用區域。如需詳細資訊，請參閱Amazon ECS 任務中繼資料端點第 4 版。 •	2020 年 4 月 8 日

變更	描述	日期
	在容器定義中，新增對 SYS_PTRACE Linux 參數的支援。如需詳細資訊，請參閱Linux 參數。 • Fargate 容器代理程式會取代所有 Fargate 任務的 Amazon ECS容器代理程式使用。這項變更應該不會對您的任務執行方式產生影響。 • 容器執行時間目前使用 Containerd，而非 Docker。這項變更應該不會對您的任務執行方式產生影響。您會注意到，容器執行時間產生的一些錯誤訊息將從提及 Docker 變更為較普通的錯誤。 如需詳細資訊，請參閱適用於 Amazon ECS 的 Fargate 平台版本。	
Amazon EFS 檔案系統支援任務磁碟區	Amazon EFS 檔案系統可以用作 Amazon ECS和 Fargate 任務的資料磁碟區。如需詳細資訊，請參閱搭配 Amazon ECS 使用 Amazon EFS 磁碟區。	2020 年 4 月 8 日
Amazon ECS 任務中繼資料端點第 4 版	從 Amazon ECS容器代理程式 1.39.0 版和 Fargate 平台 1.4.0 版開始，名為 的環境變數ECS_CONTAINER_METADATA_URI_V4 會注入任務中的每個容器。當您查詢任務中繼資料第 4 版端點時，有各種任務中繼資料和 Docker 統計資訊可供任務使用。如需詳細資訊，請參閱Amazon ECS 任務中繼資料端點第 4 版。	2020 年 4 月 8 日
支援特定版本的 Secrets Manager 秘密作為環境變數插入。	新增使用特定 Secrets Manager 秘密版本指定敏感資料的支援。如需詳細資訊，請參閱將敏感資料傳遞至 Amazon ECS 容器。	2020 年 2 月 24 日
新增藍/綠 CodeDeploy 部署的其他部署組態選項	CodeDeploy 服務為 Amazon 部署類型新增了新的 Canary 和線性ECS部署組態。也可以定義自訂部署組態。如需詳細資訊，請參閱在部署之前驗證 Amazon ECS 服務的狀態 。	2020 年 2 月 6 日

變更	描述	日期
新增 efsVolumeConfiguration 任務定義參數	`efsVolumeConfiguration` 任務定義參數處於公有預覽，可讓您更輕鬆地將 Amazon EFS 檔案系統與Amazon ECS任務搭配使用。如需詳細資訊，請參閱搭配 Amazon ECS 使用 Amazon EFS 磁碟區。	2020 年 1 月 17 日
已更新 Amazon ECS容器代理程式記錄行為	Amazon ECS容器代理程式記錄位置和輪換行為已更新。如需詳細資訊，請參閱Amazon ECS 容器代理程式日誌組態參數。	2020 年 1 月 13 日
Fargate Spot	Amazon ECS 新增了使用 Fargate Spot 執行任務的支援。如需詳細資訊，請參閱Fargate 啟動類型的 Amazon ECS 叢集 。	2019 年 12 月 3 日
叢集 Auto Scaling	Amazon ECS叢集自動擴展可讓您更掌控在叢集中擴展任務的方式。如需詳細資訊，請參閱使用叢集自動擴展自動管理 Amazon ECS 容量。	2019 年 12 月 3 日
叢集容量提供者	Amazon ECS叢集容量提供者會決定要用於任務的基礎設施。如需詳細資訊，請參閱Amazon ECS 叢集。	2019 年 12 月 3 日
在 上建立叢集 AWS Outposts	Amazon ECS 現在支援在 上建立叢集 AWS Outposts。如需詳細資訊，請參閱the section called “上的 Amazon Elastic Container Service AWS Outposts”。	2019 年 12 月 3 日
服務動作事件	Amazon ECS 現在會在發生特定服務動作 EventBridge 時傳送事件至 Amazon。如需詳細資訊，請參閱Amazon ECS 服務動作事件。	2019 年 11 月 25 日
Amazon ECS GPU最佳化 AMI支援 G4 執行個體	使用 Amazon GPU最佳化 時，Amazon 新增對 g4 ECS執行個體類型系列的ECS支援AMI。如需詳細資訊，請參閱GPU 工作負載的 Amazon ECS 任務定義。	2019 年 10 月 8 日
FireLens 適用於 Amazon ECS	FireLens for Amazon ECS 為一般可用性。 FireLens for Amazon ECS可讓您使用任務定義參數，將日誌路由至服務或合作夥伴目的地， AWS 以進行日誌儲存和分析。如需詳細資訊，請參閱將 Amazon ECS 日誌傳送至 AWS 服務或 AWS Partner。	2019 年 9 月 30 日

變更	描述	日期
AWS Fargate 區域擴展	AWS Fargate 與 Amazon ECS 已擴展至歐洲（巴黎）、歐洲（斯德哥爾摩）和中東（巴林）區域。	2019 年 9 月 30 日
在 Amazon 上使用彈性推論的深度學習容器 ECS	Amazon ECS支援將 Amazon Elastic Inference 加速器連接至您的容器，讓執行深度學習推論工作負載更有效率。	2019 年 9 月 3 日
FireLens 適用於 Amazon ECS	FireLens for Amazon ECS 處於公有預覽。 FireLens for Amazon ECS可讓您使用任務定義參數，將日誌路由到 AWS 服務或合作夥伴目的地，以進行日誌儲存和分析。如需詳細資訊，請參閱將 Amazon ECS 日誌傳送至 AWS 服務或 AWS Partner。	2019 年 8 月 30 日
CloudWatch 容器洞見	CloudWatch Container Insights 現在已全面推出。它可讓您從容器化應用程式和微型服務收集、彙總和總結指標和日誌。如需詳細資訊，請參閱使用具有增強可觀測性的 Container Insights 監控 Amazon ECS 容器。	2019 年 8 月 30 日
容器層級交換組態	Amazon ECS 新增了在容器層級控制 Linux 容器執行個體上交換記憶體空間用量的支援。使用每個容器交換組態，任務定義內的每個容器都可以啟用或停用交換，並且對於已啟用交換的容器，可以限制所使用的交換空間數量上限。如需詳細資訊，請參閱管理 Amazon ECS 上的容器交換記憶體空間。	2019 年 8 月 16 日
AWS Fargate 區域擴展	AWS Fargate 與 Amazon ECS 已擴展至亞太區域（香港）區域。	2019 年 8 月 6 日
彈性網路介面中繼	新增了用於ENI中繼功能的其他支援 Amazon EC2執行個體類型。如需詳細資訊，請參閱支援增加 Amazon ECS 容器網路介面的執行個體。	2019 年 8 月 1 日
透過服務註冊多個目標群組	已新增在服務定義中指定多個目標群組的支援。如需詳細資訊，請參閱使用 Amazon ECS 服務註冊多個目標群組。	2019 年 7 月 30 日

變更	描述	日期
使用 Secrets Manager 秘密指定敏感資料	新增了使用 Secrets Manager 秘密指定敏感資料的教學課程。如需詳細資訊，請參閱在 Amazon ECS 中使用 Secrets Manager 秘密指定敏感資料。	2019 年 7 月 20 日
CloudWatch 容器洞見	Amazon ECS已新增對 CloudWatch Container Insights 的支援。如需詳細資訊，請參閱使用具有增強可觀測性的 Container Insights 監控 Amazon ECS 容器。	2019 年 7 月 9 日
Amazon ECS服務和任務集的資源層級許可	Amazon ECS已擴展 Amazon ECS服務和任務的資源層級許可支援。如需詳細資訊，請參閱Amazon Elastic Container Service 如何與 IAM 搭配使用。	2019 年 6 月 27 日
針對 2019 AWS年 5 月推出全新 ECSAmazon 最佳化AMI修補	Amazon ECS已更新 Amazon ECS最佳化，AMI以解決 AWS-2019-005 中所述的漏洞。	2019 年 6 月 17 日
彈性網路介面中繼	Amazon ECS推出使用具有更高彈性網路界面 (ENI) 密度的支援 Amazon 執行個體類型來啟動容器EC2 執行個體的支援。使用這些執行個體類型並選擇加入`awsvpcTrunking` 帳戶設定，可增加新啟動容器執行個體的ENI密度，讓您在每個容器執行個體上放置更多任務。如需詳細資訊，請參閱增加 Amazon ECS Linux 容器執行個體網路介面。	2019 年 6 月 6 日
AWS Fargate 平台 1.3.0 版更新	從 2019 年 5 月 1 日開始，啟動的任何新 Fargate 任務除了支援 `awslogs` 日誌驅動程式外，還支援 `splunk` 日誌驅動程式。如需詳細資訊，請參閱儲存與記錄。	2019 年 5 月 1 日
AWS Fargate 平台 1.3.0 版更新	從 2019 年 5 月 1 日開始，啟動的任何新 Fargate 任務支援使用 `secretOptions` 容器定義參數，以參考容器日誌組態中的敏感資料。如需詳細資訊，請參閱將敏感資料傳遞至 Amazon ECS 容器。	2019 年 5 月 1 日

變更	描述	日期
AWS Fargate 平台 1.3.0 版更新	從 2019 年 4 月 2 日起，啟動的任何新 Fargate 任務都支援將敏感資料注入您的容器，方法是將敏感資料存放在 AWS Secrets Manager 秘密或 AWS Systems Manager 參數存放區參數中，然後在容器定義中參考它們。如需詳細資訊，請參閱將敏感資料傳遞至 Amazon ECS 容器。	2019 年 4 月 2 日
AWS Fargate 平台 1.3.0 版更新	從 2019 年 3 月 27 日開始，啟動的任何新的 Fargate 任務可以使用額外的任務定義參數，讓您定義代理組態、容器啟動和關閉的相依性，以及每一容器的啟動和停止逾時值。如需詳細資訊，請參閱代理組態、容器相依性及容器逾時。	2019 年 3 月 27 日
Amazon ECS推出外部部署類型	外部部署類型可讓您使用任何第三方部署控制器來完全控制 Amazon ECS服務的部署程序。如需詳細資訊，請參閱使用第三方控制器部署 Amazon ECS 服務。	2019 年 3 月 27 日
AWS Amazon 上的深度學習容器 ECS	AWS 深度學習容器是一組 Docker 影像，用於在 Amazon Elastic Container Service (Amazon) TensorFlow 上訓練和提供模型ECS。深度學習容器提供最佳化環境 TensorFlow，包含 Nvidia CUDA（適用於GPU執行個體） 和 Intel MKL（適用於CPU執行個體） 程式庫，可在 Amazon 中使用ECR。如需詳細資訊，請參閱在 Amazon ECS 上使用 AWS 深度學習容器。	2019 年 3 月 27 日
Amazon ECS推出增強型容器相依性管理	Amazon ECS推出其他任務定義參數，可讓您定義容器啟動和關閉的相依性，以及每個容器的啟動和停止逾時值。如需詳細資訊，請參閱容器相依性。	2019 年 3 月 7 日

變更	描述	日期
Amazon ECS推出 `PutAccountSettingDefault` API	Amazon ECS推出 `PutAccountSettingDefault` API，允許使用者設定帳戶上所有使用者和角色的預設 ARN/ID 格式選擇加入狀態。在過去，必須具有帳戶擁有者身分才能設定帳戶的預設選擇狀態。 如需詳細資訊，請參閱Amazon Resource Names (ARNs) 和 IDs。	2019 年 2 月 8 日
Amazon ECS支援GPU工作負載	Amazon 推出對GPU工作負載的ECS支援，可讓您使用 GPU啟用 的容器執行個體建立叢集。在任務定義中，您可以指定所需的數量，GPUs客服人員ECS會將實體鎖定GPUs到容器。 如需詳細資訊，請參閱GPU 工作負載的 Amazon ECS 任務定義。	2019 年 2 月 4 日
Amazon ECS擴充秘密支援	Amazon ECS擴充了對在您的任務定義中直接使用 AWS Secrets Manager 秘密將敏感資料注入容器的支援。 如需詳細資訊，請參閱將敏感資料傳遞至 Amazon ECS 容器。	2019 年 1 月 21 日
介面VPC端點 (AWS PrivateLink)	已新增設定 支援的介面VPC端點的支援 AWS PrivateLink。這可讓您在 VPC和 Amazon 之間建立私有連線，ECS而不需要透過網際網路、NAT執行個體、VPN連線或 存取 AWS Direct Connect。 如需詳細資訊，請參閱界面VPC端點 (AWS PrivateLink)。	2018 年 12 月 26 日

變更	描述	日期
AWS Fargate 平台 1.3.0 版	已發行新的 AWS Fargate 平台版本，其中包含： • 新增使用 AWS Systems Manager 參數存放區參數將敏感資料注入容器的支援。 如需詳細資訊，請參閱將敏感資料傳遞至 Amazon ECS 容器。 • 新增 Fargate 任務的任務回收，這是重新整理屬於 Amazon ECS服務一部分的任務的程序。 如需詳細資訊，請參閱 Amazon AWS Fargate 上的任務淘汰和維護ECS。 如需詳細資訊，請參閱適用於 Amazon ECS 的 Fargate 平台版本。	2018 年 12 月 17 日
更新服務限制	以下服務限制已更新： • 每個帳戶每一區域的叢集數目已從 1000 提高到 2000。 • 每個叢集的容器執行個體數目已從 1000 提高到 2000。 • 每個叢集的服務數目已從 500 提高到 1000。 如需詳細資訊，請參閱Amazon ECS 服務配額。	2018 年 12 月 14 日
AWS Fargate 區域擴展	AWS Fargate 與 Amazon ECS 已擴展至亞太區域（孟買）和加拿大（中部）區域。 如需詳細資訊，請參閱支援 AWS Fargate 上 Amazon ECS 的區域。	2018 年 12 月 7 日

變更	描述	日期
Amazon ECS藍/綠部署	Amazon ECS 新增了對使用 的藍/綠部署的支援 CodeDeploy。此部署類型可在傳送生產流量到服務的新部署之前，讓您先驗證新部署。 如需詳細資訊，請參閱在部署之前驗證 Amazon ECS 服務的狀態 。	2018 年 11 月 27 日
Amazon ECS最佳化 Amazon Linux 2 (arm64) AMI已發行	Amazon ECS發行適用於 arm64 架構AMIs的 Amazon ECS最佳化 Amazon Linux 2。 如需詳細資訊，請參閱Amazon ECS 最佳化 Linux AMIs。	2018 年 11 月 26 日
在任務定義中增加支援其他 Docker 旗標	Amazon 在任務定義中ECS推出對下列 Docker 旗標的支援： • IPC 模式 • PID 模式	2018 年 11 月 16 日
Amazon ECS秘密支援	Amazon ECS 新增了使用 AWS Systems Manager 參數存放區參數將敏感資料注入容器的支援。 如需詳細資訊，請參閱將敏感資料傳遞至 Amazon ECS 容器。	2018 年 11 月 15 日
資源標記	Amazon ECS已新增支援將中繼資料標籤新增至您的服務、任務定義、任務、叢集和容器執行個體。 如需詳細資訊，請參閱標記 Amazon ECS 資源。	2018 年 11 月 15 日
AWS Fargate 區域擴展	AWS Fargate 與 Amazon ECS 已擴展至美國西部（加利佛尼亞北部）和亞太區域（首爾）區域。 如需詳細資訊，請參閱AWS Fargate 適用於 Amazon ECS。	2018 年 11 月 7 日

變更	描述	日期
更新服務限制	以下服務限制已更新： • 每個帳戶每個區域使用 Fargate 啟動類型的任務數量從 20 增加到 50。 • 使用 Fargate 啟動類型之任務的公有 IP 地址數量從 20 增加到 50。 如需詳細資訊，請參閱Amazon ECS 服務配額。	2018 年 10 月 31 日
AWS Fargate 區域擴展	AWS Fargate 與 Amazon ECS 已擴展至歐洲（倫敦）區域。 如需詳細資訊，請參閱AWS Fargate 適用於 Amazon ECS。	2018 年 10 月 26 日
Amazon ECS最佳化 Amazon Linux 2 已AMI發行	Amazon ECS vends Linux AMIs 已針對兩種變體中的服務進行最佳化。最新和建議的版本以 x; 為基礎。Amazon ECS也以 AMIs為基礎的 vend，但我們建議您將工作負載遷移至 Amazon Linux 2 變體，因為 Amazon Linux 的支援AMI將於 2020 年 6 月 30 日之前結束。 如需詳細資訊，請參閱Amazon ECS 最佳化 Linux AMIs。	2018 年 10 月 18 日
Amazon ECS 任務中繼資料端點第 3 版	從 Amazon ECS容器代理程式 1.21.0 版開始，代理程式會將稱為 的環境變數注入任務中的ECS_CONTAINER_METADATA_URI 每個容器。當您查詢任務中繼資料第 3 版端點時，在 Amazon ECS容器代理程式提供的HTTP端點上使用awsvpc網路模式的任務，可以使用各種任務中繼資料和 Docker 統計資料。如需詳細資訊，請參閱使用 Amazon ECS 中繼資料監控工作負載。	2018 年 10 月 18 日

變更	描述	日期
Amazon ECS服務探索區域擴展	Amazon ECS服務探索已將支援擴展至加拿大（中部）、南美洲（聖保羅）、亞太區域（首爾）、亞太區域（孟買）和歐洲（巴黎）區域。 如需詳細資訊，請參閱使用服務探索將 Amazon ECS 服務與 DNS 名稱連線。	2018 年 9 月 27 日
新增對容器定義中額外 Docker 標記的支援	Amazon 在容器定義中ECS推出對下列 Docker 旗標的支援： • 系統控制 • 互動性 • 虛擬終端機	2018 年 9 月 17 日
ECS 使用 AWS Fargate 任務支援 Amazon 的私有登錄檔身分驗證	Amazon ECS使用 推出對 Fargate 任務的支援 AWS Secrets Manager。此功能可讓您安全地存放您的登入資料，然後在您的容器定義中加以參考，如此可讓您的任務使用私有映像。 如需詳細資訊，請參閱在 Amazon ECS 中使用非AWS 容器映像。	2018 年 9 月 10 日
Amazon ECS服務探索區域擴展	Amazon ECS服務探索已將支援擴展至亞太區域（新加坡）、亞太區域（雪梨）、亞太區域（東京）、歐洲（法蘭克福）和歐洲（倫敦）區域。 如需詳細資訊，請參閱使用服務探索將 Amazon ECS 服務與 DNS 名稱連線。	2018 年 8 月 30 日
支援 Fargate 任務的排程任務	Amazon ECS推出 Fargate 啟動類型的排程任務支援。	2018 年 8 月 28 日

變更	描述	日期
使用 AWS Secrets Manager 支援的私有登錄檔身分驗證	Amazon ECS推出使用 的私有登錄檔身分驗證支援 AWS Secrets Manager。此功能可讓您安全地存放您的登入資料，然後在您的容器定義中加以參考，如此可讓您的任務使用私有映像。 如需詳細資訊，請參閱在 Amazon ECS 中使用非AWS 容器映像。	2018 年 8 月 16 日
新增 Docker 磁碟區支援	Amazon ECS推出對 Docker 磁碟區的支援。 如需詳細資訊，請參閱Amazon ECS 任務的儲存選項。	2018 年 8 月 9 日
AWS Fargate 區域擴展	AWS Amazon 的 Fargate ECS 已擴展至歐洲 （法蘭克福）、亞太區域 （新加坡） 和亞太區域 （雪梨）。 如需詳細資訊，請參閱AWS Fargate 適用於 Amazon ECS。	2018 年 7 月 19 日

變更	描述	日期
已新增 Amazon ECS服務排程器策略	Amazon ECS引進了服務排程器策略的概念。 現已推出兩個服務排程器策略概念： • REPLICA - 複本排程策略會置放並在整個叢集中維持所需的任務數量。根據預設，服務排程器會將任務分散至各個可用區域。您可以使用任務置放策略和限制條件，來自訂任務置放決策。如需詳細資訊，請參閱複本策略。 • DAEMON - 常駐程式排程策略會在每個符合您於叢集中所指定所有任務置放限制條件的作用中容器執行個體上，準確地部署一個任務。使用這項策略時，不需指定所需的任務數量、任務置放策略或使用 Service Auto Scaling 政策。如需詳細資訊，請參閱協助程式策略。 Note Fargate 任務不支援 DAEMON 排程策略。 .	2018 年 6 月 12 日
Amazon ECS容器代理程式 1.18.0 版	新發行的 Amazon ECS容器代理程式版本，新增了下列功能： • 新增使用 ECS_IMAGE_PULL_BEHAVIOR 參數自訂容器代理程式映像提取行為的支援。如需詳細資訊，請參閱Amazon ECS 容器代理程式組態。 如需詳細資訊，請參閱 amazon-ecs-agent github。	2018 年 5 月 24 日

變更	描述	日期
在設定服務探索時，新增對 `bridge` 和 `host` 網路模式的支援	新增使用指定 `bridge`或 `host` 網路模式的任務定義來設定 Amazon ECS服務的服務探索的支援。如需詳細資訊，請參閱使用服務探索將 Amazon ECS 服務與 DNS 名稱連線。	2018 年 5 月 22 日
新增對其他 Amazon ECS 最佳化AMI中繼資料參數的支援	新增子參數，可讓您以程式設計方式擷取 Amazon ECS 最佳化 AMI ID、映像名稱、作業系統、容器代理程式版本和執行時間版本。使用 Systems Manager 參數存放區 查詢中繼資料API。如需詳細資訊，請參閱擷取 Amazon ECS 最佳化 Linux AMI 中繼資料。	2018 年 5 月 9 日
AWS Fargate 區域擴展	AWS Fargate 與 Amazon ECS 已擴展至美國東部（俄亥俄）、美國西部（奧勒岡）和歐洲西部（愛爾蘭）區域。 如需詳細資訊，請參閱AWS Fargate 適用於 Amazon ECS。	2018 年 4 月 26 日
Amazon ECS最佳化AMI中繼資料擷取	新增使用 Systems Manager 參數存放區 以程式設計方式擷取 Amazon ECS最佳化AMI中繼資料的功能API。如需詳細資訊，請參閱擷取 Amazon ECS 最佳化 Linux AMI 中繼資料。	2018 年 4 月 10 日
AWS Fargate 平台版本	已發行新的 AWS Fargate 平台版本，其中包含： • 新增了對 使用 Amazon ECS 中繼資料監控工作負載 的支援。 • 新增了對 運作狀態檢查 的支援。 • 新增對 使用服務探索將 Amazon ECS 服務與 DNS 名稱連線 的支援 如需詳細資訊，請參閱適用於 Amazon ECS 的 Fargate 平台版本。	2018 年 3 月 26 日

變更	描述	日期
Amazon ECS Service Discovery	新增與 Route 53 的整合，以支援 Amazon ECS服務探索。如需詳細資訊，請參閱使用服務探索將 Amazon ECS 服務與 DNS 名稱連線。	2018 年 3 月 22 日
支援 Docker shm-size 和 tmpfs	新增對 Amazon ECS任務定義中 Docker shm-size 和 tmpfs 參數的支援。 如需更新ECSCLI語法的詳細資訊，請參閱 Linux 參數。	2018 年 3 月 20 日
容器運作狀態檢查	在容器定義中，新增對 Docker 運作狀態檢查的支援。如需詳細資訊，請參閱運作狀態檢查。	2018 年 3 月 8 日
AWS Fargate	新增了 Amazon ECS with AWS Fargate 的概觀。如需詳細資訊，請參閱AWS Fargate 適用於 Amazon ECS。	2018 年 2 月 22 日
Amazon ECS任務中繼資料端點	從 Amazon ECS容器代理程式 1.17.0 版開始，在 Amazon ECS容器代理程式提供的HTTP端點上使用awsvpc網路模式的任務，可以使用各種任務中繼資料和 Docker 統計資料。如需詳細資訊，請參閱使用 Amazon ECS 中繼資料監控工作負載。	2018 年 2 月 8 日
使用目標追蹤政策的 Amazon ECS Service Auto Scaling	在 Amazon ECS主控台中使用目標追蹤政策新增對 ECS Service Auto Scaling 的支援。如需詳細資訊，請參閱使用目標指標來擴展 Amazon ECS 服務。 移除ECS第一個執行精靈中步驟擴展的先前教學課程。並用新的目標追蹤教學課程取而代之。	2018 年 2 月 8 日
Docker 17.09 支援	新增對 Docker 17.09 的支援。如需詳細資訊，請參閱Amazon ECS 最佳化 Linux AMIs。	2018 年 1 月 18 日
新的服務排程器行為	更新了無法啟動之服務任務的行為資訊。記錄了當服務任務發生連續故障時觸發的新服務事件訊息。。	2018 年 1 月 11 日

變更	描述	日期
Elastic Load Balancing 運作狀態檢查初始化等待期間	新增了指定運作狀態檢查等待期間的能力。	2017 年 12 月 27 日
任務層級CPU和記憶體	在任務定義中新增在任務層級指定 CPU和 記憶體的支援。如需詳細資訊，請參閱TaskDefinition。	2017 年 12 月 12 日
任務執行角色	Amazon ECS容器代理程式會代表您呼叫 Amazon ECSAPI動作，因此需要 IAM 政策和角色，服務才能知道代理程式屬於您。任務執行角色涵蓋以下動作： • 呼叫 Amazon ECR 以提取容器映像 • 呼叫 CloudWatch 以存放容器應用程式日誌 如需詳細資訊，請參閱Amazon ECS 任務執行 IAM 角色。	2017 年 12 月 7 日
Windows 容器支援 GA	新增了 Windows Server 2016 容器的支援。如需詳細資訊，請參閱Amazon ECS 最佳化 AMI 變體。	2017 年 12 月 5 日
AWS Fargate GA	新增使用 Fargate 啟動類型啟動 Amazon ECS服務的支援。如需詳細資訊，請參閱Amazon ECS啟動類型。	2017 年 11 月 29 日
Amazon ECS名稱變更	Amazon Elastic Container Service 已重新命名（先前為 Amazon EC2 Container Service)。	2017 年 11 月 21 日
任務聯網	awsvpc 網路模式提供的任務聯網功能為 Amazon ECS任務提供與 Amazon EC2執行個體相同的聯網屬性。當您在任務定義中使用 awsvpc 網路模式時，從該任務定義啟動的每個任務都會取得自己的彈性網路界面、主要私有 IP 地址和內部DNS主機名稱。任務聯網功能可簡化容器聯網，並讓您更掌控容器化應用程式如何與中的彼此和其他服務通訊VPCs。如需詳細資訊，請參閱EC2 啟動類型的 Amazon ECS 任務聯網選項。	2017 年 11 月 14 日

變更	描述	日期
Amazon ECS容器中繼資料	Amazon ECS容器現在可以存取中繼資料，例如其 Docker 容器或映像 ID、聯網組態或 Amazon ARNs。如需詳細資訊，請參閱Amazon ECS 容器中繼資料檔案。	2017 年 11 月 2 日
Docker 17.06 支援	新增對 Docker 17.06 的支援。如需詳細資訊，請參閱Amazon ECS 最佳化 Linux AMIs。	2017 年 11 月 2 日
Docker 旗標 device 和 init 的支援	新增了在任務定義中使用 `LinuxParameters` 參數 (`devices` 和 `initProcessEnabled`) 之 Docker device 和 init 功能的支援。如需詳細資訊，請參閱LinuxParameters。	2017 年 11 月 2 日
Docker 旗標 cap-add 和 cap-drop 支援	新增了在任務定義中使用 `LinuxParameters` 參數 (`capabilities`) 之 Docker cap-add 和 cap-drop 功能的支援。如需詳細資訊，請參閱LinuxParameters。	2017 年 9 月 22 日
Network Load Balancer 支援	Amazon 已在 Amazon ECS主控台中ECS新增對 Network Load Balancer 的支援。	2017 年 9 月 7 日
RunTask 覆寫	新增了執行任務時的任務定義覆寫支援。這可讓您在變更任務定義時執行任務，而不需要建立新的任務定義修訂。如需詳細資訊，請參閱以 Amazon ECS 任務執行應用程式。	2017 年 6 月 27 日
Amazon ECS排程任務	新增了使用 cron 排程任務的支援。	2017 年 6 月 7 日
Amazon ECS主控台中的 Spot 執行個體	新增在 Amazon ECS主控台中建立 Spot Fleet 容器執行個體的支援。如需詳細資訊，請參閱啟動 Amazon ECS Linux 容器執行個體。	2017 年 6 月 6 日
新的 Amazon ECS最佳化 AMI版本的 Amazon SNS 通知	新增訂閱新 Amazon ECS最佳化AMI版本SNS通知的功能。	2017 年 3 月 23 日

變更	描述	日期
微服務和批次任務	已新增兩個 Amazon 常見使用案例的文件ECS：微服務和批次任務。如需詳細資訊，請參閱Amazon ECS 相關資訊。	2017 年 2 月 日
容器執行個體耗盡	新增了容器執行個體耗盡的支援，其提供從叢集移除容器執行個體的方法。如需詳細資訊，請參閱耗盡 Amazon ECS 容器執行個體。	2017 年 1 月 24 日
Docker 1.12 支援	新增對 Docker 1.12 的支援。如需詳細資訊，請參閱Amazon ECS 最佳化 Linux AMIs。	2017 年 1 月 24 日
新的任務置放策略	新增了任務置放策略的支援：屬性型置放、分箱封裝、可用區域分散，以及每個主機一個的置放。如需詳細資訊，請參閱使用策略來定義 Amazon ECS 任務置放。	2016 年 12 月 29 日
Beta 版 Windows 容器支援	新增了 Windows Server 2016 容器 (Beta) 的支援。如需詳細資訊，請參閱Amazon ECS 最佳化 AMI 變體。	2016 年 12 月 20 日
Blox OSS支援	新增對 Blox 的支援OSS，允許自訂任務排程器。如需詳細資訊，請參閱在 Amazon ECS 上排程您的容器。	2016 年 12 月 1 日
適用於 ECS 事件的 Amazon CloudWatch 事件串流	Amazon ECS 現在會將容器執行個體和任務狀態變更傳送至 CloudWatch Events。如需詳細資訊，請參閱使用 EventBridge 自動化對 Amazon ECS 錯誤的回應。	2016 年 11 月 21 日
Amazon ECS容器記錄至 CloudWatch 日誌	新增支援 awslogs 驅動程式將容器日誌串流傳送至 CloudWatch Logs。如需詳細資訊，請參閱將 Amazon ECS 日誌傳送至 CloudWatch 。	2016 年 9 月 12 日
支援動態連接埠的 Elastic Load Balancing Amazon ECS服務	新增了負載平衡器的支援，以為每個接聽程式支援多個 instance:port 組合，這可增加容器的彈性。現在，您可以讓 Docker 動態定義容器的主機連接埠，並且ECS排程器向負載平衡器註冊 instance：port。如需詳細資訊，請參閱使用負載平衡來分發 Amazon ECS 服務流量。	2016 年 8 月 11 日

變更	描述	日期
IAM Amazon ECS任務的角色	新增了將 IAM 角色與任務建立關聯的支援。相對於整個容器執行個體的單一角色，這可為容器提供更精細定義的許可。如需詳細資訊，請參閱Amazon ECS 任務 IAM 角色。	2016 年 7 月 13 日
Docker 1.11 支援	新增對 Docker 1.11 的支援。如需詳細資訊，請參閱Amazon ECS 最佳化 Linux AMIs。	2016 年 5 月 31 日
任務自動調整規模	Amazon ECS 新增了對 服務自動擴展任務的支援。如需詳細資訊，請參閱自動擴展 Amazon ECS 服務。	2016 年 5 月 18 日
依任務系列篩選的任務定義	新增了根據任務定義系列篩選任務定義清單的支援。如需詳細資訊，請參閱ListTaskDefinitions。	2016 年 5 月 17 日
Docker 容器和 Amazon ECS代理程式記錄	Amazon ECS 新增了將ECS代理程式和 Docker 容器日誌從容器執行個體傳送到 CloudWatch 日誌的功能，以簡化故障診斷問題。	2016 年 5 月 5 日
ECS最佳化AMI現在支援 Amazon Linux 2016.03。	已ECS最佳化的 AMI 已新增對 Amazon Linux 2016.03 的支援。如需詳細資訊，請參閱Amazon ECS 最佳化 Linux AMIs。	2016 年 4 月 5 日
Docker 1.9 支援	新增對 Docker 1.9 的支援。如需詳細資訊，請參閱Amazon ECS 最佳化 Linux AMIs。	2015 年 12 月 22 日
CloudWatch 叢集CPU和記憶體保留的指標	Amazon ECS 新增了 CPU和 記憶體保留的自訂 CloudWatch 指標。	2015 年 12 月 22 日
新的 Amazon ECS初次執行體驗	Amazon ECS主控台初次執行體驗新增了零點擊角色建立。	2015 年 23 月 11 日
跨可用區域的任務置放	Amazon ECS服務排程器新增了跨可用區域放置任務的支援。	2015 年 10 月 8 日

變更	描述	日期
CloudWatch Amazon ECS叢集和服務的 指標	Amazon 新增叢集中每個容器執行個體、服務和任務定義系列的ECS自訂 CloudWatch 指標和CPU記憶體使用率。這些新指標可用於使用 Auto Scaling 群組擴展叢集中的容器執行個體，或建立自訂 CloudWatch 警示。	2015 年 8 月 17 日
UDP 連接埠支援	新增對任務定義中UDP連接埠的支援。	2015 年 7 月 7 日
環境變數覆寫	新增對 deregisterTaskDefinition 和 環境變數覆寫的支援runTask。	2015 年 6 月 18 日
自動化 Amazon ECS代理程式更新	新增查看容器執行個體上執行之ECS代理程式版本的功能。也可以從 AWS Management Console AWS CLI、和 更新ECS代理程式SDK。	2015 年 6 月 11 日
Amazon ECS服務排程器和 Elastic Load Balancing 整合	新增了定義服務並將該服務與 Elastic Load Balancing 負載平衡器建立關聯的能力。	2015 年 4 月 9 日
Amazon ECS GA	美國東部（維吉尼亞北部）、美國西部（奧勒岡）、亞太區域（東京）和歐洲（愛爾蘭）區域的 Amazon ECS一般可用性。	2015 年 4 月 9 日

本文為英文版的機器翻譯版本，如內容有任何歧義或不一致之處，概以英文版為準。