aws

開發人員指南

截止日期雲端

截止日期雲端: 開發人員指南

Table of Contents

什麼是 AWS 截止日期雲端？

AWS Deadline Cloud 是一種全受管 AWS 服務，可讓您在幾分鐘內啟動和執行可擴展的處理陣列。它提供管理主控台，用於管理使用者、陣列、排程任務的佇列，以及執行處理作業的工作者機群。

此開發人員指南適用於各種使用案例的管道、工具和應用程式開發人員，包括下列項目：

- 管道開發人員和技術主管可以將截止日期雲端 APIs和功能整合到其自訂生產管道中。
- 獨立軟體廠商可以將 Deadline Cloud 整合到其應用程式中，讓數位內容建立藝術家和使用者能夠從工作站無縫提交 Deadline Cloud 轉譯任務。
- Web 和雲端服務開發人員可以將 Deadline Cloud 轉譯整合至其平台，讓客戶能夠以虛擬方式提供資產來檢視產品。

我們提供的工具可讓您直接使用管道的任何步驟：

- 您可以直接使用或從指令碼使用的命令列界面。
- 適用於 11 種熱門程式設計語言的 AWS SDK。
- 您可以從應用程式呼叫的 REST 型 Web 界面。

您也可以 AWS 服務 在自訂應用程式中使用其他 。例如，您可以使用：

- AWS CloudFormation 自動化建立和移除陣列、佇列和機群。
- Amazon CloudWatch 收集任務的指標。
- Amazon Simple Storage Service 可存放和管理數位資產和任務輸出。
- AWS IAM Identity Center 來管理陣列的使用者和群組。

開啟任務描述

Deadline Cloud 使用 Open Job Description (OpenJD) 規格來指定任務的詳細資訊。OpenJD 旨在定義解決方案之間可攜的任務。您可以使用它來定義任務，該任務是在工作者主機上執行的一組命令。

您可以使用 Deadline Cloud 提供的提交者來建立 OpenJD 任務範本，也可以使用您要建立範本的任何工具。建立範本後，您會將其傳送至截止日期雲端。如果您使用提交者，它會負責傳送範本。如果您以其他方式建立範本，您可以呼叫截止日期雲端命令列動作，或者您可以使用其中一個 AWS SDKs來傳送任務。無論哪種方式，Deadline Cloud 都會將任務新增至指定的佇列，並排程工作。

截止日期雲端的概念和術語

為了協助您開始使用 AWS 截止日期雲端，本主題說明其一些重要概念和術語。

陣列資源

此圖表顯示截止日期雲端陣列資源如何一起運作。

伺服器陣列

陣列包含與提交和執行任務相關的所有其他資源。陣列彼此獨立，因此有助於分隔生產環境。

佇列

佇列會保留在相關聯機群上排程的任務。使用者可以將任務提交至佇列，並在佇列中管理其優先順序和狀態。佇列必須與具有佇列機群關聯的機群相關聯，才能執行其任務，而且佇列可以與多個機群相關聯。

機群

機群包含執行中任務的運算容量。機群可以是服務受管或客戶受管。服務受管機群會在截止日期雲端中執行，並包含內建功能，例如自動擴展、授權和軟體存取。客戶受管機群會在您自己的運算資源上執行，例如 Amazon EC2 執行個體或內部部署伺服器。

Budget

預算會設定任務活動的花費閾值，並可讓您在達到閾值時採取動作，例如停止任務排程。

佇列環境

佇列環境會定義在每個工作者上執行的指令碼，以設定或銷毀工作負載環境。它們適用於設定環境變數、安裝軟體和設定資產儲存。

儲存設定檔

儲存設定檔是一組主機和工作站的組態，可告知資料位於檔案系統上的位置。截止日期 雲端使用儲存描述檔，在不同設定的主機上執行任務時映射路徑，例如從 提交Windows並在 上執行的任務Linux。

限制

限制可讓您追蹤共用資源的使用情況，例如浮動授權，並控制任務之間的配置方式。限制與具有佇列限制關聯的佇列相關聯。

監控

監視器會設定截止期限雲端監視器 Web 應用程式的 URL，讓使用者能夠監控和管理任務。它可以在瀏覽器中或透過截止日期雲端監控桌面應用程式存取。

任務執行資源

此圖表顯示截止日期雲端任務資源如何一起運作。

任務

任務是一組工作，使用者提交至 Deadline Cloud 以排程並在可用的工作者上執行。任務可能會轉譯 3D 場景或執行模擬。任務是從可重複使用的任務範本建立的，這些範本會定義執行期環境和程序，以及任務特定的參數。任務包含定義要執行之工作的步驟和任務，而且可以設定優先順序、工作者計數上限和重試設定。

任務優先順序

任務優先順序是截止日期雲端在佇列中處理任務的大致順序。您可以設定 1 到 100 之間的任務優先順序，優先順序較高的任務通常會先處理。具有相同優先順序的任務會依收到的順序處理。

任務屬性

任務屬性是您在提交轉譯任務時定義的設定。一些範例包括影格範圍、輸出路徑、任務附件、可轉譯攝影機等。屬性會根據提交轉譯的 DCC 而有所不同。

步驟

步驟是任務的一部分，提供執行許多相同任務的範本，但任務參數值除外。步驟可以對其他步驟具有相依性，可讓您建立具有循序或平行執行路徑的複雜工作流程。在轉譯任務中，步驟通常會定義轉譯影格的命令，並使用影格編號做為任務參數。

任務

任務是截止日期雲端中最小的工作單位。任務是步驟的一部分，由工作者執行，代表需要在任務中執行的個別操作。任務可以使用特定參數設定，並根據其功能和可用性指派給工作者。在轉譯任務中，任務通常會轉譯單一影格。

工作程序

工作者是機群的一部分，並從任務執行任務。工作者可以設定特定功能，例如 GPU 加速器、CPU 架構和作業系統。在服務受管機群中，隨著機群向外和向內擴展，會自動建立工作者。

執行個體

機群使用 CPU 資源的執行個體。執行個體是 Amazon EC2 效能執行個體。Deadline Cloud 使用隨需執行個體和 Spot 執行個體。

隨需執行個體

隨需執行個體會以秒計價，沒有長期承諾，而且不會中斷。

Spot 執行個體

Spot 執行個體是以折扣價格使用的未預留容量，但可能會因為隨需請求而中斷。

等待並儲存

「等待和儲存」功能以較低的成本提供延遲的任務排程，並且可以被隨需和 Spot 請求中斷。Wait and Save 僅適用於截止日期雲端服務受管機群。

Wait and Save 是在 AWS 截止日期雲端中管理視覺運算工作負載的執行。如需詳細資訊，請參閱 AWS 服務條款。

Session (工作階段)

工作階段代表工作者在任務上的工作順序。在單一工作階段期間，工作者可能會被指派多個任務，其會逐一執行。工作階段通常具有設定動作，可在執行任務動作之前設定環境和載入資產。

工作階段動作

工作階段動作代表工作階段期間執行的特定操作，例如設定環境、執行任務和同步資產。

其他重要概念和術語

用量總管

用量總管是截止日期雲端監視器的一項功能。它提供成本和用量的預估值。

預算管理員

Budget Manager 是截止日期雲端監視器的一部分。使用預算管理員來建立和管理預算。您也可以使用它來限制活動以保持在預算內。

截止日期雲端用戶端程式庫

開放原始碼用戶端程式庫包含用於管理截止日期雲端的命令列界面和程式庫。功能包括根據開放任務描述規格將任務套件提交至截止日期雲端、下載任務連接輸出，以及使用命令列界面 (CLI) 監控您的陣列。

數位內容建立應用程式 (DCC)

數位內容建立應用程式 DCCs) 是您建立數位內容的第三方產品。Deadline Cloud 內建與許多 DCCs 的整合，例如 Autodesk Maya、Blender 和 Maxon Cinema 4D，可讓您從 DCC 內提交任務，並在具有預先設定軟體和授權的服務受管機群上進行轉譯。

任務附件

任務連接是一種截止日期雲端功能，您可以在紋理、3D 模型和照明裝備等任務中上傳和下載資產。任務附件存放在 Amazon S3 中，並避免共用網路儲存的需求。

任務範本

任務範本會定義執行時間環境，以及做為截止日期雲端任務一部分執行的所有程序。

截止日期雲端提交者

截止日期雲端提交者是 DCC 的外掛程式，可讓使用者輕鬆地從 DCC 內提交任務。

授權端點

授權端點可讓 Deadline Cloud 在 VPC 內提供第三方產品的用量型授權。此模型會依您的進度付費，而且會向您收取使用時數和分鐘數的費用。授權端點未連線到陣列，可以獨立使用。

Tags (標籤)

標籤是您可以指派給 AWS 資源的標籤。每個標籤都包含您定義的索引鍵和選用值。您可以使用標籤，以不同的方式分類 AWS 資源，例如依用途、擁有者或環境。

以用量為基礎的授權 (UBL)

以用量為基礎的授權 (UBL) 是一種隨需授權模式，可供特定第三方產品使用。此模型是按您的用量付費，而且您需要支付使用時數和分鐘數的費用。

截止日期雲端架構指引

本主題提供指引和最佳實務，以使用截止日期雲端為您的工作負載設計和建置可靠、安全、有效率且符合成本效益的渲染陣列。使用此指南可協助您建置穩定且有效率的工作負載，讓您專注於創新、降低成本並改善客戶體驗。

此內容是供首席技術長 (CTO)、架構師、開發人員和營運團隊成員使用。

end-to-end轉譯工作流程需要多個程序層級的解決方案，例如任務產生、資產存取和任務監控。Deadline Cloud 為轉譯程序的每一層提供多個解決方案。透過在每個 layer 中選擇 Deadline Cloud 的選項，您可以設計符合您使用案例的工作流程。

對於每個 layer，您將需要決定哪種方法最適合您的使用案例。這些並非嚴格的案例定義，也不是使用截止日期雲端的唯一方法。反之，這些是一組高階概念，可協助您了解 Deadline Cloud 如何能夠適應您的業務或工作流程。您可以將截止日期雲端工作負載分成下列層級：任務來源、任務提交、應用程式管理、應用程式授權、資產存取、輸出管理和工作者基礎設施管理。

一般而言，您可以將某個 layer 中的任何案例與另一個 layer 中的任何其他案例mix-and-match，但以下指定的特定組合除外。


Job Source
Interactive Workflow
Automated Workflow
Job Submission
Integrated Submitter
Custom Job Definition
Application Management
Conda Application Management
Custom Application Management
Application Licensing
Deadline Cloud UBL
Custom Licensing
Asset Access
Job Attachments
Custom Storage
Job Monitoring
Deadline Cloud monitor
Custom Monitor Application
Automated Monitoring Solution
Worker Infrastructure

任務來源

任務來源是新任務將進入系統以供截止日期雲端轉譯的存取點。從高階來看，任務有兩個主要來源：人類互動性和自動化電腦系統。

互動式工作流程

在此案例中，藝術家或其他創意角色是要在截止日期雲端陣列中處理的主要工作產生器。這些任務的輸出通常是大型專案或團隊的主要成品。他們使用產業標準數位內容建立 (DCC) 工具等軟體來執行工作。它們會手動將任務提交至截止日期雲端陣列，並在之後檢視輸出以供檢閱。工作站本身不受 管理 AWS。

在大多數情況下，這些藝術家會在工作負載應用程式和監控層中使用截止日期雲端整合提交者和截止日期雲端監控。

自動化工作流程

在此案例中，客戶擁有的程式設計系統是截止日期雲端陣列中任務的主要產生器。這可以是零售管道中的資產產生，例如從 3D 模型或掃描產生的轉盤影片。這可能是為運動自動合成廣播圖形和玩家卡。此案例的主題是，個人不會手動將每個任務提交至截止日期雲端，而是將任務產生為較大系統的一部分。

使用自動化任務時，Deadline Cloud 整合提交者和 Deadline Cloud 監視器使用較少見。任務定義通常是由您撰寫的自訂應用程式開發，而任務輸出會自動流入數位資產管理 (DAM) 系統或媒體資產管理 (MAM) 系統以進行核准和分發。

任務提交

任務會使用 OpenJobDescription 範本提交至截止日期雲端。OpenJobDescription 是一種靈活的開放規格，用於定義在不同排程系統部署之間可攜式的批次處理任務。任務定義檔案說明任務的參數、任務的步驟、步驟如何根據任務輸入進行參數化，以及在工作者上執行的實際指令碼來執行處理。工作負載提交的概念是如何建立這些任務定義、誰建立它們以及如何提交它們。

整合提交者與 DCC

Deadline Cloud 整合提交者是將 Deadline Cloud 與業界標準 DCC 或軟體套件連結在一起的一種軟體。整合式提交者決定如何將轉譯、複合或其他工作負載的資料和組態轉換為任務範本，這是截止日期雲端可以理解的內容。許多整合的提交者是由截止日期的雲端團隊或軟體套件的建立者建立和維護，但

如果所需的應用程式尚不存在，您可以建立和維護自己的提交者。Deadline Cloud 團隊支援的一組有限 DCCs。

互動式工作流程通常涉及整合提交者，但並非一律如此。對於範本化的自動化工作流程，常見的工作流程是讓藝術家在其 DCC 中設定範本任務，並執行任務套件的一次性匯出。此任務套件定義了如何以參數化方式在截止日期雲端上執行該特定類型的任務。此任務套件可以整合到自動化工作流程案例，以用於自動化目的。

自訂任務定義

對於自訂應用程式和工作流程，您可以完全控制如何建立這些任務定義並提交至截止日期雲端。例如，電子商務網站可能會要求賣方上傳其銷售物件的 3D 模型。在此上傳之後，電子商務平台可以動態產生任務定義，以提交至截止日期雲端，使用通用照明在通用背景上自動產生轉盤動畫，以符合網站上可用的其他 3D 物件。在開發電子商務平台期間，軟體開發人員會建立任務定義、使用賣方最終提供的參數將其嵌入電子商務平台，並在平台產品上傳工作流程期間編寫程式碼以提交此任務。

Deadline Cloud 在 github 上的範例儲存庫中提供多個範例任務定義。

應用程式管理

將任務提交至截止日期雲端並指派給工作者之後，任務定義的指令碼會在工作者上執行。在大多數情況下，此指令碼會叫用應用程式來執行實際處理，例如轉譯器、複合、編碼、篩選或任何其他運算密集型任務。應用程式管理是確保必要版本的必要軟體可供工作者使用的概念。

您可以使用任何您喜歡的套件管理系統來管理應用程式，但 Deadline Cloud 提供多種工具，可輕鬆啟用 conda 套件。Conda 是一種開放原始碼、跨平台、語言無關的套件管理員和環境管理系統。

服務受管機群 (SMF) 的截止日期雲端受管 Conda 通道

使用服務受管機群時，系統會自動設定並設定截止日期的雲端受管 Conda 頻道，以供您的任務使用。Deadline Cloud 服務在此 conda 頻道中提供許多合作夥伴 DCC 應用程式和轉譯。如需詳細資訊，請參閱截止日期雲端使用者指南中的建立佇列環境。這些套件由截止日期雲端服務自動保持最新狀態，不需要您進行維護。此 conda 頻道僅在使用服務受管機群時可用，在使用客戶受管機群時無法使用。

自我管理 conda 頻道

如果您無法使用截止日期雲端管理的 conda 頻道，您必須決定如何在截止日期雲端機群上安裝、修補和以其他方式管理應用程式。其中一個選項是建立您設定和維護的 conda 頻道。這將與截止日期雲端受管 conda 通道最密切地相互操作。例如，您可以從截止日期雲端管理的 conda 頻道使用 DCC，但攜

帶自己的套件，其中包含特定的 DCC 外掛程式。如需此程序的詳細資訊，請參閱使用 S3 建立 conda 頻道。

自訂應用程式管理

對於應用程式管理，截止日期雲端的要求是在工作者上執行任務指令碼時，應用程式可在 PATH 中使用。

如果您已建置和維護 Rez 套件，您可以使用佇列環境從 Rez 儲存庫安裝應用程式。您可以在AWS 截止日期雲端 GitHub 組織找到佇列環境範例。

如果您已在具有長期工作者的客戶受管機群上或系統映像中管理應用程式，則應用程式管理不需要佇列環境。確保應用程式出現在任務使用者的路徑上，並提交任務。

應用程式授權

許多工作負載通常在 Deadline Cloud 上執行，需要軟體廠商的軟體授權。這些應用程式通常獲得每個座位、每個 CPU 或每個主機的授權。您有責任確保在 Deadline Cloud 上使用第三方軟體時遵守第三方授權合約。如果您使用開放原始碼軟體、自訂軟體或其他免授權軟體，則不需要設定此層。請記住，截止日期雲端僅支援轉譯授權，不支援工作站授權。

服務受管機群和用量型授權

使用截止日期雲端服務受管機群時，會自動為支援的軟體設定用量型授權 (UBL)。在服務受管機群上執行的任務會自動為支援的應用程式設定環境變數，以指示他們使用截止日期雲端授權伺服器。使用截止日期雲端 UBL 時，您只需支付使用授權應用程式的時數費用。

客戶受管機群和用量型授權

不使用服務受管機群時，也會提供截止日期雲端用量型授權 (UBL)。在此案例中，您將設定截止日期雲端授權端點，該端點會在您選取的 VPC 子網路中提供 IP 地址，以便存取截止日期雲端授權伺服器。在工作者上設定適當的軟體特定環境變數，並設定從工作者到這些授權端點 IP 地址的網路連線之後，工作者就可以簽出和簽入支援軟體的授權。在服務受管機群中使用 UBL 時，您每小時需支付與 相同的授權費用。

自訂授權

您可以使用截止日期雲端 UBL 不支援的應用程式，或者您可能已有仍然有效的預先存在授權。在此案例中，您必須負責設定從工作者 （客戶管理或服務管理） 到授權伺服器的網路路徑。如需自訂授權的詳細資訊，請參閱 將服務受管機群連接至自訂授權伺服器。

資產存取

將任務提交至工作者並設定應用程式後，工作者必須設定為存取任務所需的資產資料。這可以是 3D 資料、紋理資料、動畫資料、影片影格或任務中使用的任何其他類型資料。

從考慮資料目前存放的位置開始。這可能是在工作站硬碟、使用者協作工具、來源控制、內部部署或雲端中的共用檔案系統、Amazon S3 或任何數量的其他位置。

接著，請考慮工作者存取此資料的必要條件。此資料是否僅在您的公司網路上提供？ 存取資料需要哪些身分或登入資料？ 資料來源是否經過擴展，以使用您預期處理任務的工作者數量來支援任務？

任務附件

從資產存取機制開始的最簡單方法是截止日期雲端任務附件。使用任務附件提交任務時，任務所需的資料會上傳到 Amazon S3 儲存貯體，以及指定任務所需檔案的資訊清單檔案。使用任務附件時，不需要複雜的聯網或共用儲存設定。檔案只會上傳一次，因此後續上傳會更快完成。工作者完成處理任務後，輸出資料會上傳至 Amazon S3，以便藝術家或其他用戶端下載。任務連接可為任何大小的機群擴展規模，並可輕鬆快速地加入和使用。

任務連接並非適用於所有情況的最佳工具。如果您的資料已開啟 AWS，則任務附件會新增資料的額外副本，包括相關的傳輸時間和儲存成本。任務附件要求任務可以完整指定提交時所需的資料，以便上傳資料。

若要使用任務附件，您的截止日期雲端佇列必須具有相關聯的任務附件儲存貯體，而且佇列角色必須用來提供該儲存貯體的存取權。根據預設，截止日期雲端整合提交者都支援任務附件。如果您未使用截止日期雲端整合提交者，您可以透過整合截止日期雲端 Python 程式庫，將任務附件與自訂軟體搭配使用。

自訂儲存存取

如果您不使用任務附件，您必須負責確保工作者可以存取任務所需的資料。Deadline Cloud 提供多種工具來支援這項功能，並讓任務保持可攜式。當您已有藝術家和工作者的共用網路儲存體時，您可能想要使用自訂儲存解決方案，您偏好使用 LucidLink 等外部服務或其他原因。

使用儲存描述檔來建立工作站和工作者主機上的檔案系統模型。每個儲存設定檔都會描述其中一個系統組態的作業系統和檔案系統配置。使用儲存設定檔，當使用 Windows 工作站的藝術家提交由Linux工作者處理的任務時，截止日期雲端會確保路徑映射發生，以便工作者可以存取您設定的資料儲存體。

使用 Deadline Cloud 服務受管機群時，主機組態指令碼和 VPC 資源端點可讓工作者直接掛載和存取 VPC 中可用的共用儲存或其他服務。

任務監控和輸出管理

成功完成提交至截止日期雲端的任務後，人員或程序將下載任務輸出，以在截止日期雲端之外的業務工作流程中使用。任務失敗後，任務日誌和監控資訊有助於診斷問題。

截止日期雲端監視器

Deadline Cloud Monitor 應用程式可在 Web 和桌面上使用。此解決方案最適合在各種 DCCs 中使用互動式工作流程的工作室，並使用任務附件進行儲存。監視器僅在使用 IAM Identity Center 時支援您。IAM Identity Center 是人力身分產品，而不是消費者身分 (B2C) 解決方案，因此不適用於許多 B2C 案例。

自訂監控應用程式

如果您想要自訂使用者的監控體驗，您要建置 B2C 產品，或使用您選擇建立自訂監控應用程式的截止日期雲端建置高度專業化的系統。您可以使用AWS 截止日期雲端 API 來建立此自訂應用程式，將整體工作流程的內容與截止日期雲端概念結合在一起。例如，您的 B2C 產品可能有自己的專案概念，使用者可以設定這些概念，而您的應用程式可以在相同的界面中巢狀確定期限雲端任務。

自動化監控解決方案

在某些情況下，截止日期雲端不需要專用監控應用程式。此案例在自動化工作流程中很常見，其中 Deadline Cloud 用於自動轉譯管道中的資產，例如運動或新聞的廣播圖形。在此案例中，截止日期雲端 API 和 EventBridge 事件用於與外部媒體資產管理系統整合，以進行核准，並將資料移至程序的下一個步驟。

工作者基礎設施管理

截止日期雲端機群是一組伺服器（工作者），能夠處理提交至截止日期雲端佇列的任務，並且是任何截止日期雲端陣列的核心基礎設施。

服務受管機群

在服務受管機群中，Deadline Cloud 會負責工作者主機、作業系統、聯網、修補、自動擴展和執行轉譯陣列的其他因素。您可以指定所需的工作者數量下限和上限，以及應用程式所需的系統規格，而 Deadline Cloud 會執行其餘操作。服務受管機群是唯一可以使用截止雲端受管 Conda 通道來輕鬆管理產業 DCC 應用程式的機群選項。此外，截止日期雲端 UBL 會自動設定服務受管機群。等待並儲存機群以降低成本、延遲容錯的工作負載，只能使用服務受管機群。

客戶管理的機群

當您需要更多控制工作者主機及其環境時，您可以使用客戶受管機群。客戶管理的機群最適合在內部部署使用截止日期雲端。如需詳細資訊，請參閱 建立和使用截止日期 雲端客戶管理的機群。

範例架構

傳統生產工作室

傳統生產工作室需要大量的運算、儲存和聯網基礎設施，這些基礎設施可以跨越多個實體位置，以便為渲染工作負載提供服務。每個個別軟體套件和廠商都有唯一的硬體、軟體、聯網和授權需求，在解決版本控制、相容性和資源衝突時必須符合這些需求。

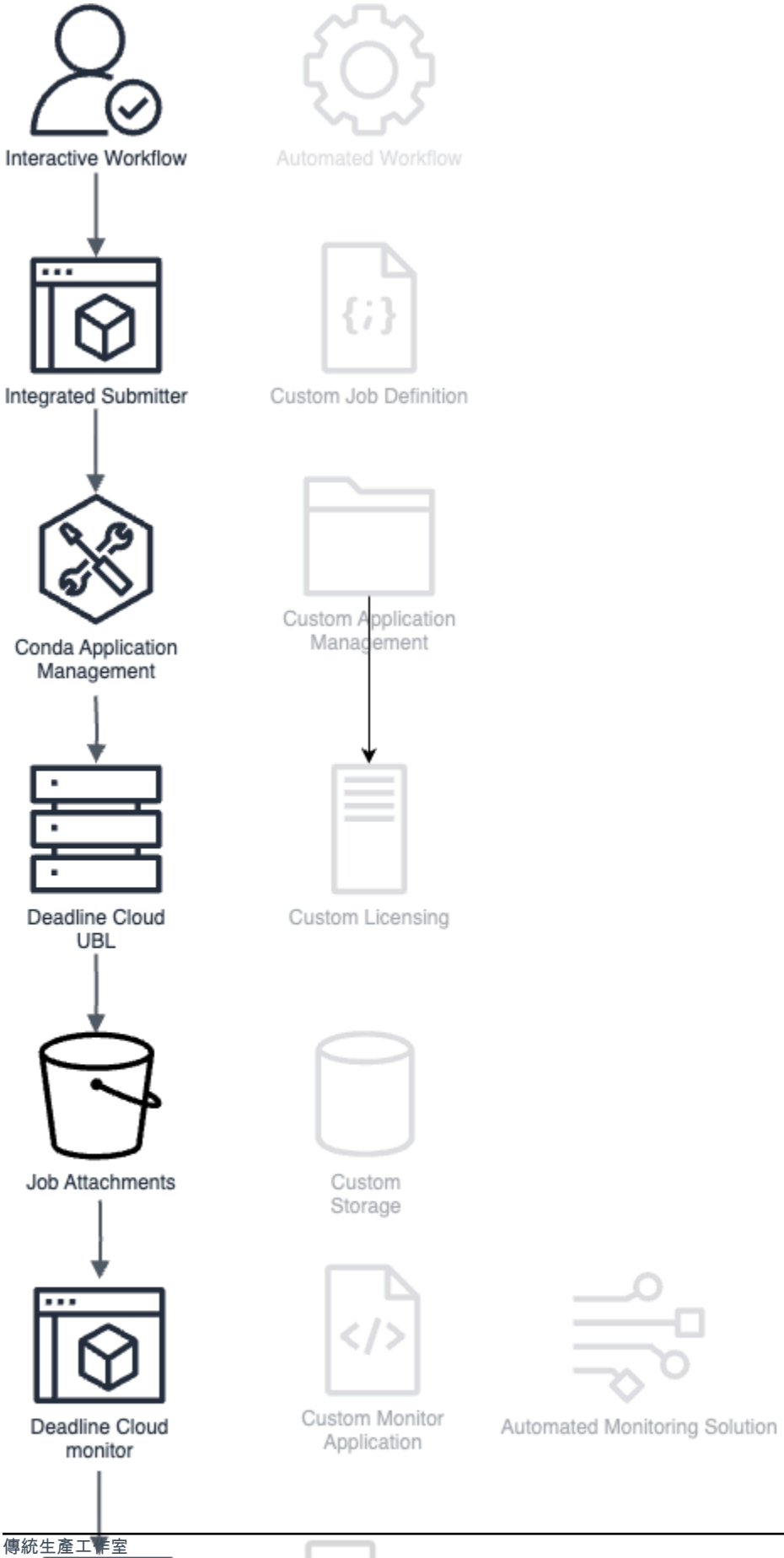
對於藝術家工作站、轉譯節點、網路儲存體、授權伺服器、任務佇列系統、監控工具和資產管理，有個別的基礎設施需求是很常見的。Studio 通常需要維護多個版本的 DCC 工具、轉譯器、外掛程式和自訂工具，同時管理整個轉譯陣列的複雜授權安排。當您考慮開發、品質保證和生產環境時，您的 Studio 基礎設施會變得更加複雜。

使用服務受管選項的典型截止日期雲端部署可解決或減少下列許多挑戰：

- 透過整合式 DCC 提交者提交互動式工作流程任務
- 透過截止日期雲端受管 Conda 通道進行應用程式管理
- 針對支援的軟體自動設定以用量為基礎的授權
- 透過任務附件進行資產管理
- 透過截止日期雲端監控應用程式進行監控
- 透過服務受管機群進行基礎設施管理

透過這種方法，藝術家可以直接從熟悉的 DCC 工具提交任務到可擴展的雲端轉譯陣列，而無需管理複雜的基礎設施。服務會自動處理軟體部署、授權、資料傳輸和基礎設施擴展。藝術家可以透過 Web 界面或桌面應用程式監控其任務，而輸出會自動存放在 Amazon S3 中，以便於存取。

透過此組態，工作室可以在幾分鐘內建立開發和生產環境，只需支付其使用的運算和授權費用，並專注於創意工作，而不是基礎設施管理。服務受管方法提供採用雲端轉譯的最快路徑，同時為藝術家維持熟悉的工作流程。


Interactive Workflow
Automated Workflow
Integrated Submitter
Custom Job Definition
Conda Application Management
Custom Application Management
Deadline Cloud UBL
Custom Licensing
Job Attachments
Custom Storage
Deadline Cloud monitor
Custom Monitor Application
Automated Monitoring Solution

雲端中的 Studio

現代視覺效果和動畫工作室逐漸將整個管道移至雲端，包括藝術家工作站。這種方法消除了現場部署基礎設施的需求，實現了全球協作，並為互動式工作和渲染提供無縫擴展。不過，它還在管理雲端資源、確保低延遲存取資料，以及整合雲端工作站與轉譯陣列方面帶來新的挑戰。

典型的雲端原生工作室需要統一方法來管理所有這些元件的雲端工作站、共用儲存、轉譯基礎設施和軟體部署。傳統方法通常會導致複雜的手動受管系統，難以平衡效能、成本和彈性。

雲端原生工作室的截止日期雲端部署可以使用下列方式實作：

- 透過雲端工作站上的整合式 DCC 提交者提交互動式工作流程任務
- 透過截止日期雲端受管 Conda 通道轉譯節點進行應用程式管理
- 針對支援的軟體自動設定以用量為基礎的授權
- 使用 FSx for Windows File Server 進行共用專案資料的自訂儲存存取
- 透過截止日期雲端監控應用程式進行監控
- 使用服務受管機群的基礎設施管理

這種方法可讓藝術家在可直接存取高效能共用儲存的雲端型工作站上工作，並將任務無縫提交至截止日期雲端陣列。Studio 可以使用相同的 conda 通道管理跨工作站和轉譯節點的軟體部署，以確保一致性並減少維護開銷。

此組態的主要優點包括：

- 與能夠從任何地方存取工作站的藝術家進行全球協作
- 跨工作站和轉譯節點的一致軟體環境
- 工作站和轉譯節點皆可存取的高效能共用儲存
- 彈性擴展互動式和批次運算資源
- 集中管理雲端中的所有 Studio 基礎設施

在此案例中的儲存組態通常涉及：

- 專案資料的 FSx for Windows File Server，可供雲端工作站和截止日期雲端工作者存取
- 截止日期雲端中的儲存描述檔，用於管理工作站和轉譯節點之間的路徑映射
- 使用 VPC 資源端點和主機組態指令碼，在截止日期的雲端工作者上直接掛載 FSx 共用

這種雲端原生方法可讓工作室消除內部部署基礎設施，為任何規模的專案實現快速擴展，同時保持熟悉的藝術家工作流程。它提供了靈活地使用服務受管和客戶受管資源的組合，並針對易於管理和特定效能需求進行最佳化。

透過利用雲端工作站與截止日期雲端，工作室可以實現完全整合、全球可存取的生產管道，從小型團隊無縫擴展到大型生產。

ECommerce自動化

現代電子商務平台需要大規模自動化資產產生，才能在數百萬個項目中提供豐富的產品視覺化。傳統方法需要大量的基礎設施投資，才能將大量 3D 模型處理為標準化產品媒體，這通常會導致系統佈建不足，進而建立處理待處理項目，或系統過度佈建且容量閒置。

典型的自動化電子商務工作流程需要處理產品上傳處理、3D 模型驗證、轉譯陣列管理、輸出處理，以及與產品資訊系統的整合。管理這些工作流程傳統上需要協調多個轉譯應用程式、運算資源和資料處理管道，同時確保一致的品質並大規模維持成本效益。

電子商務自動化的截止日期雲端部署可以使用下列方式實作：

- 透過現有電子商務擷取應用程式中的自訂 API 整合提交自動化工作流程任務
- 針對標準化產品視覺化量身打造的自訂任務定義
- 透過截止日期雲端受管 Conda 通道進行應用程式管理
- 針對支援的軟體自動設定以用量為基礎的授權
- 資產管理的直接 Amazon S3 整合
- 與現有產品管理系統整合的自訂監控應用程式
- 用於彈性擴展的服務受管機群

這種方法可以每天處理數千種產品，自動產生標準化產品視覺化，例如轉盤動畫。服務受管基礎設施會自動擴展以滿足可變需求，同時透過工作者重複使用和最佳化應用程式部署來維持成本效益。


eCommerce
Interactive Workflow
Automated Workflow
Integrated Submitter
Custom Job Definition
Conda Application Management
Custom Application Management
Deadline Cloud UBL
Custom Licensing
Job Attachments
Custom Storage
Deadline Cloud monitor
Custom Monitor Application
Automated Monitoring Solution

Whitelabel/OEM/B2C 客戶

傳統數位內容建立 (DCC) 軟體通常要求使用者在其工作站本機維護自己的轉譯基礎設施或程序轉譯，導致大量硬體投資或長時間等待，進而中斷創意工作流程。對於軟體供應商，提供雲端轉譯功能，傳統上需要建立和維護複雜的基礎設施和計費系統。

整合到 B2C 軟體的截止日期雲端部署，可直接在使用者熟悉的界面中實現無縫雲端轉譯。此整合結合了：

- 內嵌在 DCC 應用程式中的互動式工作流程任務提交
- 轉譯應用程式部署的截止日期雲端管理 conda 通道
- 自動設定以用量為基礎的授權
- 透過具有廠商受管儲存的任務附件進行資產管理
- 直接整合在 DCC 界面中的自訂監控
- 跨使用者共用的服務受管機群

此方法可讓最終使用者從軟體中按一下，將轉譯提交至雲端，而無需管理帳戶、基礎設施或複雜的設定。軟體廠商維護多租戶環境，其中：

- 使用者透過現有的軟體登入資料進行身分驗證
- 任務會自動路由至每個使用者的專用佇列
- 使用 IAM 控制的儲存字首安全地隔離資產
- 帳單是透過廠商的現有系統處理
- 任務狀態和輸出會直接串流回使用者的應用程式

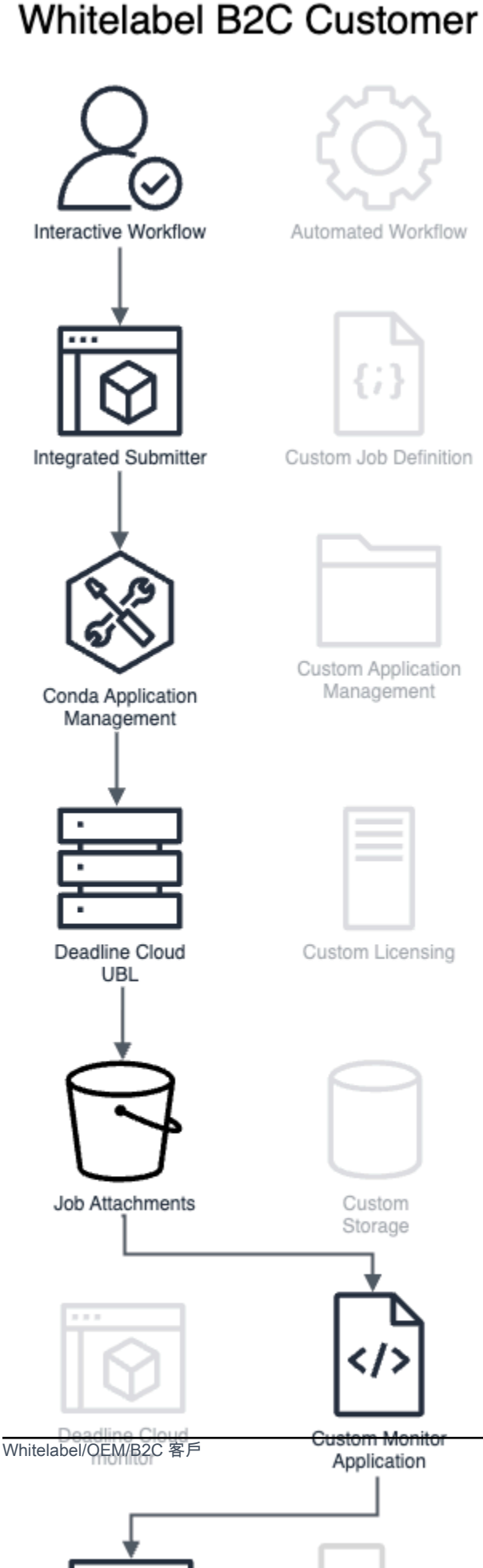
共用機群方法透過維護工作者的暖集區、將啟動時間降至最低，同時將整個使用者群的資源使用率最大化，來確保最佳效能。此組態可讓軟體廠商提供雲端轉譯作為無縫的產品功能，而不是需要額外設定或帳戶的獨立服務。

最終使用者受益於：

- 從熟悉的界面一鍵提交
- 無基礎設施管理Pay-as-you-go定價
- 透過共用基礎設施加快任務啟動時間
- 已完成轉譯的自動下載和組織

- 跨所有平台的一致體驗

此整合模式可讓軟體廠商為其整個使用者群提供企業級渲染功能，同時維持簡單、易於取用的體驗，讓他們的應用程式原生體驗。


Whitelabel B2C Customer
Interactive Workflow
Automated Workflow
Integrated Submitter
Custom Job Definition
Conda Application Management
Custom Application Management
Deadline Cloud UBL
Custom Licensing
Job Attachments
Custom Storage
Deadline Cloud monitor
Custom Monitor Application
Automated Monitoring Solution

什麼是截止日期雲端工作負載

使用 AWS 截止日期雲端，您可以提交任務以在雲端中執行應用程式，並處理資料，以生產對業務至關重要的內容或洞見。Deadline Cloud 使用 Open Job Description (OpenJD) 做為任務範本的語法，此規格專為視覺化運算管道的需求而設計，但適用於許多其他使用案例。一些範例工作負載包括電腦圖形渲染、物理模擬和光素測量。

工作負載從使用者使用 CLI 或自動產生的 GUI 提交到佇列的簡單任務套件，擴展到為應用程式定義工作負載動態產生任務套件的整合提交者外掛程式。

工作負載如何從生產環境中產生

若要了解生產環境中的工作負載，以及如何透過截止日期雲端支援工作負載，請考慮它們的呈現方式。生產可能涉及建立視覺效果、動畫、遊戲、產品目錄影像、用於建置資訊建模 (BIM) 的 3D 重建等。此內容通常由執行各種軟體應用程式和自訂指令碼的藝術或技術專家團隊建立。團隊成員使用生產管道在彼此之間傳遞資料。管道執行的許多任務都涉及密集運算，如果在使用者的工作站上執行，可能需要幾天的時間。

這些生產管道中的一些任務範例包括：

- 使用光素測量應用程式處理電影集拍攝的相片，以重建紋理數位網格。
- 在 3D 場景中執行粒子模擬，為電視節目的爆炸視覺效果新增多層細節。
- 將遊戲關卡的資料製作成外部發行和套用最佳化和壓縮設定所需的格式。
- 轉譯產品目錄的一組影像，包括顏色、背景和照明的變化。
- 在 3D 模型上執行自訂開發的指令碼，以套用由電影導演自訂建置和核准的外觀。

這些任務涉及許多參數來調整，以取得藝術結果或微調輸出品質。通常有一個 GUI 可以使用按鈕或功能表選取這些參數值，以在應用程式中本機執行程序。當使用者執行程序時，應用程式和主機電腦本身可能無法用於執行其他操作，因為它使用記憶體中的應用程式狀態，並且可能會消耗主機電腦的所有 CPU 和記憶體資源。

在許多情況下，程序非常快速。在生產過程中，當品質和複雜性的需求上升時，程序的速度會變慢。在開發期間花費 30 秒的字元測試，在套用到最終生產角色時，可以輕鬆變成 3 小時。透過此進展，在 GUI 內開始生命週期的工作負載可能會變得太大而無法適應。將其移植到截止日期雲端可以提高執行這些程序之使用者的生產力，因為他們可以恢復對工作站的完全控制，並且可以從截止日期雲端監視器追蹤更多反覆運算。

在截止日期雲端開發工作負載支援時，需要兩個層級的支援：

- 將工作負載從使用者工作站卸載至沒有平行處理或加速的截止日期雲端陣列。這可能未充分利用陣列中可用的運算資源，但能夠將長時間操作轉移到批次處理系統，讓使用者能夠使用自己的工作站完成更多工作。
- 最佳化工作負載的平行處理，以便利用截止日期雲端陣列的水平擴展快速完成。

有時候，讓工作負載平行執行很明顯。例如，電腦圖形轉譯的每個影格都可以獨立完成。不過，請務必不要卡在這種平行處理上。相反地，請了解，將長時間執行的工作負載卸載至 Deadline Cloud 可提供顯著的好處，即使沒有明顯的方式來分割工作負載。

工作負載的成分

若要指定截止日期雲端工作負載，請使用截止日期雲端 CLI 實作使用者提交至佇列的任務套件。建立任務套件的大部分工作是撰寫任務範本，但還有更多因素，例如如何提供工作負載所需的應用程式。以下是定義截止日期雲端工作負載時需要考慮的重要事項：

- 要執行的應用程式。任務必須能夠啟動應用程式程序，因此需要安裝可用的應用程式，以及應用程式使用的任何授權，例如存取浮動授權伺服器。這通常是陣列組態的一部分，而不是內嵌在任務套件本身。
  - 使用佇列環境設定任務
  - 將客戶受管機群連接至授權端點
- 任務參數定義。提交任務的使用者體驗會受到其提供的參數大幅影響。範例參數包括資料檔案、目錄和應用程式組態。
  - 任務套件的參數值元素
- 檔案資料流程。當任務執行時，它會從使用者提供的檔案讀取輸入，然後將其輸出寫入為新檔案。若要使用任務附件和路徑映射功能，任務必須為這些輸入和輸出指定目錄或特定檔案的路徑。
  - 在任務中使用檔案
- 步驟指令碼。步驟指令碼會使用正確的命令列選項執行應用程式二進位檔，以套用提供的任務參數。如果工作負載資料檔案包含絕對值而非相對路徑參考，它也會處理路徑映射等詳細資訊。
  - 任務套件的任務範本元素

工作負載可攜性

工作負載可在多個不同系統中執行，而不必在每次提交任務時變更，則為可攜式工作負載。例如，它可能會在已掛載不同共用檔案系統的不同轉譯陣列上執行，或在 Linux或 等不同作業系統上執行 Windows。當您實作可攜式任務套件時，使用者可以更輕鬆地在自己的特定陣列上執行任務，或針對其他使用案例進行調整。

以下是您可以讓任務套件成為可攜式的一些方法。

- 使用PATH任務套件中的任務參數和資產參考，完整指定工作負載所需的輸入資料檔案。這種方法讓任務可以根據共用檔案系統移植到陣列，以及製作輸入資料副本的陣列，例如截止日期雲端任務連接功能。
- 讓任務輸入檔案的檔案路徑參考可在不同的作業系統上重新定位和使用。例如，當使用者從Windows工作站提交任務以在Linux機群上執行時。
  - 使用相對檔案路徑參考，因此如果包含它們的目錄移至不同的位置，參考仍會解析。有些應用程式，例如 Blender，支援相對路徑和絕對路徑之間的選擇。
  - 如果您無法使用相對路徑， 支援 OpenJD 路徑映射中繼資料，並根據 Deadline Cloud 將檔案提供給任務的方式轉譯絕對路徑。
- 使用可攜式指令碼在任務中實作命令。Python 和 bash 是指令碼語言的兩個範例，可以用這種方式使用。您應該考慮在機群的所有工作者主機上提供兩者。
  - 使用指令碼解譯器二進位檔，例如 `python`或 `bash`，並將指令碼檔案名稱作為引數。相較於在 上使用指令碼檔案搭配其執行位元設定，此方法適用於所有作業系統Windows，包括 Linux。
  - 套用這些實務來撰寫可攜式 bash 指令碼：
    - 以單引號展開範本路徑參數，以處理具有空格和路徑分隔符號的Windows路徑。
    - 在 上執行時Windows，請注意與 MinGW 自動路徑轉譯相關的問題。例如，它會`aws logs tail /aws/deadline/...`將 之類的 AWS CLI 命令轉換為類似 的命令，`aws logs tail "C:/Program Files/Git/aws/deadline/..."`並且無法正確地追蹤日誌。設定 變數`MSYS_NO_PATHCONV=1`以關閉此行為。
    - 在大多數情況下，相同的程式碼適用於所有作業系統。當程式碼需要不同時，請使用 `if/else` 建構來處理案例。

```
if [[ "$(uname)" == MINGW* || "$(uname -s)" == MSYS_NT* ]]; then
    # Code for Windows
elif [[ "$(uname)" == Darwin ]]; then
    # Code for MacOS
else
    # Code for Linux and other operating systems
fi
```

  - 您可以使用 編寫可攜式 Python 指令碼`pathlib`，以處理檔案系統路徑差異，並避免操作特定的功能。Python 文件包含對此的註釋，例如在訊號程式庫文件中。 Linux特定的功能支援標記為「可用性：Linux」。
- 使用任務參數來指定應用程式需求。使用陣列管理員可在佇列環境中套用的一致慣例。

- 例如，您可以在任務中使用 CondaPackages和/或 RezPackages 參數，搭配預設參數值，列出任務所需的應用程式套件名稱和版本。然後，您可以使用其中一個範例 conda 或 Rez 佇列環境，為任務提供虛擬環境。

截止日期雲端資源入門

若要開始為 AWS 截止日期雲端建立自訂解決方案，您必須設定 資源。其中包括一個陣列、至少一個陣列的佇列，以及至少一個為佇列提供服務的工作者機群。您可以使用截止日期雲端主控台建立資源，也可以使用 AWS Command Line Interface。

在本教學課程中，您將使用 AWS CloudShell 建立簡單的開發人員陣列並執行工作者代理程式。然後，您可以提交和執行具有參數和附件的簡單任務、新增服務受管機群，並在完成時清除您的陣列資源。

以下各節將介紹 Deadline Cloud 的不同功能，以及它們如何一起運作和工作。遵循這些步驟有助於開發和測試新的工作負載和自訂。

如需使用 主控台設定陣列的說明，請參閱《截止日期雲端使用者指南》中的入門。

主題

- 建立截止日期雲端陣列
- 執行截止日期雲端工作者代理程式
- 使用截止日期雲端提交
- 在截止日期雲端中使用任務附件提交任務
- 在截止日期雲端中將服務受管機群新增至您的開發人員陣列
- 在截止日期雲端中清除您的陣列資源

建立截止日期雲端陣列

若要在 AWS 截止日期雲端中建立開發人員陣列和佇列資源，請使用 AWS Command Line Interface (AWS CLI)，如下列程序所示。您也將建立 AWS Identity and Access Management (IAM) 角色和客戶受管機群 (CMF)，並將機群與您的佇列建立關聯。然後，您可以設定 ， AWS CLI 並確認陣列已設定並依指定運作。

您可以使用此陣列來探索截止日期雲端的功能，然後開發和測試新的工作負載、自訂和管道整合。

建立陣列

1. 開啟工作階段 AWS CloudShell。您將使用 CloudShell 視窗輸入 AWS Command Line Interface (AWS CLI) 命令來執行本教學課程中的範例。繼續進行時，請保持 CloudShell 視窗開啟。

2. 為您的陣列建立名稱，並將該陣列名稱新增至 ~/.bashrc。這將使其可用於其他終端機工作階段。

```
echo "DEV_FARM_NAME=DeveloperFarm" >> ~/.bashrc
source ~/.bashrc
```

3. 建立陣列資源，並將其陣列 ID 新增至 ~/.bashrc。

```
aws deadline create-farm \
    --display-name "$DEV_FARM_NAME"

echo "DEV_FARM_ID=\$(aws deadline list-farms \
        --query \"farms[?displayName=='\$DEV_FARM_NAME'].farmId \
        | [0]\" --output text)" >> ~/.bashrc
source ~/.bashrc
```

4. 建立佇列資源，並將其佇列 ID 新增至 ~/.bashrc.

```
aws deadline create-queue \
    --farm-id $DEV_FARM_ID \
    --display-name "$DEV_FARM_NAME Queue" \
    --job-run-as-user '{"posix": {"user": "job-user", "group": "job-group"},
 "runAs":"QUEUE_CONFIGURED_USER"}'

echo "DEV_QUEUE_ID=\$(aws deadline list-queues \
        --farm-id \$DEV_FARM_ID \
        --query \"queues[?displayName=='\$DEV_FARM_NAME Queue'].queueId \
        | [0]\" --output text)" >> ~/.bashrc
source ~/.bashrc
```

5. 為機群建立 IAM 角色。此角色為機群中的工作者主機提供從佇列執行任務所需的安全登入資料。

```
aws iam create-role \
    --role-name "${DEV_FARM_NAME}FleetRole" \
    --assume-role-policy-document \
        '{
            "Version": "2012-10-17",
            "Statement": [
                {
                    "Effect": "Allow",
                    "Principal": {
                        "Service": "credentials.deadline.amazonaws.com"
                    },
```

```
                    "Action": "sts:AssumeRole"
                }
            ]
        }'
aws iam put-role-policy \
    --role-name "${DEV_FARM_NAME}FleetRole" \
    --policy-name WorkerPermissions \
    --policy-document \
        '{
            "Version": "2012-10-17",
            "Statement": [
                {
                    "Effect": "Allow",
                    "Action": [
                        "deadline:AssumeFleetRoleForWorker",
                        "deadline:UpdateWorker",
                        "deadline:DeleteWorker",
                        "deadline:UpdateWorkerSchedule",
                        "deadline:BatchGetJobEntity",
                        "deadline:AssumeQueueRoleForWorker"
                    ],
                    "Resource": "*",
                    "Condition": {
                        "StringEquals": {
                            "aws:PrincipalAccount": "${aws:ResourceAccount}"
                        }
                    }
                },
                {
                    "Effect": "Allow",
                    "Action": [
                        "logs:CreateLogStream"
                    ],
                    "Resource": "arn:aws:logs:*:*:*:/aws/deadline/*",
                    "Condition": {
                        "StringEquals": {
                            "aws:PrincipalAccount": "${aws:ResourceAccount}"
                        }
                    }
                },
                {
                    "Effect": "Allow",
                    "Action": [
                        "logs:PutLogEvents",
```

```
                        "logs:GetLogEvents"
                    ],
                    "Resource": "arn:aws:logs:*:*:*:/aws/deadline/*",
                    "Condition": {
                        "StringEquals": {
                            "aws:PrincipalAccount": "${aws:ResourceAccount}"
                        }
                    }
                }
            ]
        }'
```

6. 建立客戶受管機群 (CMF)，並將其機群 ID 新增至 ~/.bashrc。

```
FLEET_ROLE_ARN="arn:aws:iam::$(aws sts get-caller-identity \
        --query "Account" --output text):role/${DEV_FARM_NAME}FleetRole"
aws deadline create-fleet \
    --farm-id $DEV_FARM_ID \
    --display-name "$DEV_FARM_NAME CMF" \
    --role-arn $FLEET_ROLE_ARN \
    --max-worker-count 5 \
    --configuration \
        '{
            "customerManaged": {
                "mode": "NO_SCALING",
                "workerCapabilities": {
                    "vCpuCount": {"min": 1},
                    "memoryMiB": {"min": 512},
                    "osFamily": "linux",
                    "cpuArchitectureType": "x86_64"
                }
            }
        }'

echo "DEV_CMF_ID=\$(aws deadline list-fleets \
        --farm-id \$DEV_FARM_ID \
        --query \"fleets[?displayName=='\$DEV_FARM_NAME CMF'].fleetId \
        | [0]\" --output text)" >> ~/.bashrc
source ~/.bashrc
```

7. 將 CMF 與您的佇列建立關聯。

```
aws deadline create-queue-fleet-association \
```

```
    --farm-id $DEV_FARM_ID \
    --queue-id $DEV_QUEUE_ID \
    --fleet-id $DEV_CMF_ID
```

8. 安裝截止日期雲端命令列界面。

```
pip install deadline
```

9. 若要將預設陣列設定為陣列 ID，並將佇列設定為您先前建立的佇列 ID，請使用下列命令。

```
deadline config set defaults.farm_id $DEV_FARM_ID
deadline config set defaults.queue_id $DEV_QUEUE_ID
```

10. （選用）若要確認您的陣列已根據您的規格設定，請使用下列命令：

    - 列出所有陣列 – **`deadline farm list`**
    - 列出預設陣列中的所有佇列 – **`deadline queue list`**
    - 列出預設陣列中的所有機群 – **`deadline fleet list`**
    - 取得預設陣列 – **`deadline farm get`**
    - 取得預設佇列 – **`deadline queue get`**
    - 取得與預設佇列相關聯的所有機群 – **`deadline fleet get`**

後續步驟

建立陣列後，您可以在機群中的主機上執行截止日期雲端工作者代理程式，以處理任務。請參閱執行截止日期雲端工作者代理程式。

執行截止日期雲端工作者代理程式

您必須先在工作者主機上以開發人員模式執行 AWS 截止日期雲端工作者代理程式，才能執行您提交至開發人員陣列上的佇列的任務。

在本教學課程的其餘部分中，您將使用兩個 AWS CloudShell 索引標籤在開發人員陣列上執行 AWS CLI 操作。在第一個索引標籤中，您可以提交任務。在第二個索引標籤中，您可以執行工作者代理程式。

> **Note**
>
> 如果您讓 CloudShell 工作階段閒置超過 20 分鐘，將會逾時並停止工作者代理程式。若要重新啟動工作者代理程式，請遵循下列程序的指示。

您必須先設定截止日期雲端陣列、佇列和機群，才能啟動工作者代理程式。請參閱 建立截止日期雲端陣列。

在開發人員模式下執行工作者代理程式

1. 在第一個 CloudShell 索引標籤中仍開啟您的陣列時，開啟第二個 CloudShell 索引標籤，然後建立 `demoenv-logs`和 `demoenv-persist`目錄。

```
mkdir ~/demoenv-logs
mkdir ~/demoenv-persist
```

2. 從 PyPI 下載並安裝截止日期雲端工作者代理程式套件：

> **Note**
>
> 在 上Windows，代理程式檔案必須安裝在 Python 的全域網站套件目錄中。目前不支援 Python 虛擬環境。

```
python -m pip install deadline-cloud-worker-agent
```

3. 若要允許工作者代理程式為執行中的任務建立臨時目錄，請建立目錄：

```
sudo mkdir /sessions
sudo chmod 750 /sessions
sudo chown cloudshell-user /sessions
```

4. 使用DEV_FARM_IDDEV_CMF_ID您新增至 的變數，在開發人員模式下執行截止日期雲端工作者代理程式~/.bashrc。

```
deadline-worker-agent \
    --farm-id $DEV_FARM_ID \
    --fleet-id $DEV_CMF_ID \
    --run-jobs-as-agent-user \
```

```
    --logs-dir ~/demoenv-logs \
    --persistence-dir ~/demoenv-persist
```

當工作者代理程式初始化然後輪詢 UpdateWorkerSchedule API 操作時，會顯示下列輸出：

```
INFO    Worker Agent starting
[2024-03-27 15:51:01,292][INFO    ] # Worker Agent starting
[2024-03-27 15:51:01,292][INFO    ] AgentInfo
Python Interpreter: /usr/bin/python3
Python Version: 3.9.16 (main, Sep  8 2023, 00:00:00)  - [GCC 11.4.1 20230605 (Red
 Hat 11.4.1-2)]
Platform: linux
...
[2024-03-27 15:51:02,528][INFO    ] # API.Resp # [deadline:UpdateWorkerSchedule]
(200) params={'assignedSessions': {}, 'cancelSessionActions': {},
 'updateIntervalSeconds': 15} ...
[2024-03-27 15:51:17,635][INFO    ] # API.Resp # [deadline:UpdateWorkerSchedule]
(200) params=(Duplicate removed, see previous response) ...
[2024-03-27 15:51:32,756][INFO    ] # API.Resp # [deadline:UpdateWorkerSchedule]
(200) params=(Duplicate removed, see previous response) ...
...
```

5. 選取您的第一個 CloudShell 標籤，然後列出機群中的工作者。

```
deadline worker list --fleet-id $DEV_CMF_ID
```

會顯示如下所示的輸出：

```
Displaying 1 of 1 workers starting at 0

- workerId: worker-8c9af877c8734e89914047111f
  status: STARTED
  createdAt: 2023-12-13 20:43:06+00:00
```

在生產組態中，截止日期雲端工作者代理程式需要將多個使用者和組態目錄設定為主機上的管理使用者。您可以覆寫這些設定，因為您在自己的開發陣列中執行任務，只有您可以存取。

後續步驟

現在，工作者代理程式正在您的工作者主機上執行，您可以將任務傳送給工作者。您可以：

- 使用截止日期雲端提交 使用簡單的 OpenJD 任務套件。
- 在截止日期雲端中使用任務附件提交任務 使用不同的作業系統在工作站之間共用檔案。

使用截止日期雲端提交

若要在工作者主機上執行截止日期雲端任務，您可以建立並使用開放任務描述 (OpenJD) 任務套件來設定任務。套件會設定任務，例如指定任務的輸入檔案，以及寫入任務輸出的位置。本主題包含您可以設定任務套件的方法範例。

您必須先完成下列操作，才能遵循本節中的程序：

- 建立截止日期雲端陣列
- 執行截止日期雲端工作者代理程式

若要使用 Deadline Cloud AWS 執行任務，請使用下列程序。使用第一個 AWS CloudShell 索引標籤將任務提交到您的開發人員陣列。使用第二個 CloudShell 索引標籤來檢視工作者代理程式輸出。

主題

提交simple_job範例

建立陣列並執行工作者代理程式後，您可以將simple_job範例提交至截止日期雲端。

將simple_job範例提交至截止日期雲端

1. 選擇您的第一個 CloudShell 標籤。
2. 從 GitHub 下載範例。

```
cd ~
git clone https://github.com/aws-deadline/deadline-cloud-samples.git
```

3. 導覽至任務套件範例目錄。

```
cd ~/deadline-cloud-samples/job_bundles/
```

4. 提交simple_job範例。

```
deadline bundle submit simple_job
```

5. 選擇您的第二個 CloudShell 索引標籤，以檢視有關呼叫 BatchGetJobEntities、取得工作階段和執行工作階段動作的記錄輸出。

```
...
[2024-03-27 16:00:21,846][INFO    ] # Session.Starting
 # [session-053d77cef82648fe2] Starting new Session.
 [queue-3ba4ff683ff54db09b851a2ed8327d7b/job-d34cc98a6e234b6f82577940ab4f76c6]
[2024-03-27 16:00:21,853][INFO    ] # API.Req # [deadline:BatchGetJobEntity]
 resource={'farm-id': 'farm-3e24cfc9bbcd423e9c1b6754bc1',
 'fleet-id': 'fleet-246ee60f46d44559b6cce010d05', 'worker-id':
 'worker-75e0fce9c3c344a69bff57fcd83'} params={'identifiers': [{'jobDetails':
 {'jobId': 'job-d34cc98a6e234b6f82577940ab4'}}]} request_url=https://
scheduling.deadline.us-west-2.amazonaws.com/2023-10-12/farms/
farm-3e24cfc9bbcd423e /fleets/fleet-246ee60f46d44559b1 /workers/worker-
 75e0fce9c3c344a69b /batchGetJobEntity
[2024-03-27 16:00:22,013][INFO    ] # API.Resp # [deadline:BatchGetJobEntity](200)
 params={'entities': [{'jobDetails': {'jobId': 'job-d34cc98a6e234b6f82577940ab6',
 'jobRunAsUser': {'posix': {'user': 'job-user', 'group': 'job-group'},
 'runAs': 'QUEUE_CONFIGURED_USER'}, 'logGroupName': '/aws/deadline/
farm-3e24cfc9bbcd423e9c1b6754bc1/queue-3ba4ff683ff54db09b851a2ed83', 'parameters':
 '*REDACTED*', 'schemaVersion': 'jobtemplate-2023-09'}}], 'errors': []}
 request_id=a3f55914-6470-439e-89e5-313f0c6
[2024-03-27 16:00:22,013][INFO    ] # Session.Add #
 [session-053d77cef82648fea9c69827182] Appended new SessionActions.
 (ActionIds: ['sessionaction-053d77cef82648fea9c69827182-0'])
 [queue-3ba4ff683ff54db09b851a2ed8b/job-d34cc98a6e234b6f82577940ab6]
[2024-03-27 16:00:22,014][WARNING ] # Session.User #
 [session-053d77cef82648fea9c69827182] Running as the Worker Agent's
 user. (User: cloudshell-user) [queue-3ba4ff683ff54db09b851a2ed8b/job-
d34cc98a6e234b6f82577940ac6]
[2024-03-27 16:00:22,015][WARNING ] # Session.AWSCreds #
 [session-053d77cef82648fea9c69827182] AWS Credentials are not available: Queue has
 no IAM Role. [queue-3ba4ff683ff54db09b851a2ed8b/job-d34cc98a6e234b6f82577940ab6]
[2024-03-27 16:00:22,026][INFO    ] # Session.Logs #
 [session-053d77cef82648fea9c69827182] Logs streamed to: AWS CloudWatch
 Logs. (LogDestination: /aws/deadline/farm-3e24cfc9bbcd423e9c1b6754bc1/
```

```
queue-3ba4ff683ff54db09b851a2ed83/session-053d77cef82648fea9c69827181)
 [queue-3ba4ff683ff54db09b851a2ed83/job-d34cc98a6e234b6f82577940ab4]
[2024-03-27 16:00:22,026][INFO    ] # Session.Logs #
 [session-053d77cef82648fea9c69827182] Logs streamed to: local
 file. (LogDestination: /home/cloudshell-user/demoenv-logs/
queue-3ba4ff683ff54db09b851a2ed8b/session-053d77cef82648fea9c69827182.log)
 [queue-3ba4ff683ff54db09b851a2ed83/job-d34cc98a6e234b6f82577940ab4]
...
```

> **Note**
>
> 只會顯示工作者代理程式的記錄輸出。執行任務的工作階段有單獨的日誌。

6. 選擇您的第一個索引標籤，然後檢查工作者代理程式寫入的日誌檔案。

   a. 導覽至工作者代理程式日誌目錄並檢視其內容。

   ```
   cd ~/demoenv-logs
   ls
   ```

   b. 列印工作者代理程式建立的第一個日誌檔案。

   ```
   cat worker-agent-bootstrap.log
   ```

   此檔案包含工作者代理程式輸出，其如何稱為截止日期雲端 API，以在機群中建立工作者資源，然後擔任機群角色。

   c. 工作者代理程式加入機群時列印日誌檔案輸出。

   ```
   cat worker-agent.log
   ```

   此日誌包含工作者代理程式採取之所有動作的輸出，但不包含其執行任務之佇列的輸出，但這些資源IDs 除外。

   d. 在名為與佇列資源 ID 相同的目錄中列印每個工作階段的日誌檔案。

   ```
   cat $DEV_QUEUE_ID/session-*.log
   ```

   如果任務成功，日誌檔案輸出將類似於以下內容：

   ```
   cat $DEV_QUEUE_ID/$(ls -t $DEV_QUEUE_ID | head -1)
   ```

```
2024-03-27 16:00:22,026 WARNING Session running with no AWS Credentials.
2024-03-27 16:00:22,404 INFO
2024-03-27 16:00:22,405 INFO ================================================
2024-03-27 16:00:22,405 INFO --------- Running Task
2024-03-27 16:00:22,405 INFO ================================================
2024-03-27 16:00:22,406 INFO ------------------------------------------------
2024-03-27 16:00:22,406 INFO Phase: Setup
2024-03-27 16:00:22,406 INFO ------------------------------------------------
2024-03-27 16:00:22,406 INFO Writing embedded files for Task to disk.
2024-03-27 16:00:22,406 INFO Mapping: Task.File.runScript -> /sessions/
session-053d77cef82648fea9c698271812a/embedded_fileswa_gj55_/tmp2u9yqtsz
2024-03-27 16:00:22,406 INFO Wrote: runScript -> /sessions/
session-053d77cef82648fea9c698271812a/embedded_fileswa_gj55_/tmp2u9yqtsz
2024-03-27 16:00:22,407 INFO ------------------------------------------------
2024-03-27 16:00:22,407 INFO Phase: Running action
2024-03-27 16:00:22,407 INFO ------------------------------------------------
2024-03-27 16:00:22,407 INFO Running command /sessions/
session-053d77cef82648fea9c698271812a/tmpzuzxpslm.sh
2024-03-27 16:00:22,414 INFO Command started as pid: 471
2024-03-27 16:00:22,415 INFO Output:
2024-03-27 16:00:22,420 INFO Welcome to AWS Deadline Cloud!
2024-03-27 16:00:22,571 INFO
2024-03-27 16:00:22,572 INFO ================================================
2024-03-27 16:00:22,572 INFO --------- Session Cleanup
2024-03-27 16:00:22,572 INFO ================================================
2024-03-27 16:00:22,572 INFO Deleting working directory: /sessions/
session-053d77cef82648fea9c698271812a
```

7. 列印任務的相關資訊。

```
deadline job get
```

當您提交任務時，系統會將其儲存為預設值，因此您不需要輸入任務 ID。

simple_job 使用 參數提交

您可以使用參數提交任務。在下列程序中，您可以編輯simple_job範本以包含自訂訊息、提交simple_job，然後列印工作階段日誌檔案以檢視訊息。

使用 參數提交simple_job範例

1. 選取您的第一個 CloudShell 索引標籤，然後導覽至任務套件範例目錄。

```
cd ~/deadline-cloud-samples/job_bundles/
```

2. 列印simple_job範本的內容。

```
cat simple_job/template.yaml
```

具有 Message 參數的 parameterDefinitions區段應如下所示：

```
parameterDefinitions:
- name: Message
  type: STRING
  default: Welcome to AWS Deadline Cloud!
```

3. 使用參數值提交simple_job範例，然後等待任務完成執行。

```
deadline bundle submit simple_job \
    -p "Message=Greetings from the developer getting started guide."
```

4. 若要查看自訂訊息，請檢視最新的工作階段日誌檔案。

```
cd ~/demoenv-logs
cat $DEV_QUEUE_ID/$(ls -t $DEV_QUEUE_ID | head -1)
```

使用檔案 I/O 建立 simple_file_job 任務套件

轉譯任務需要讀取場景定義、從中轉譯影像，然後將該影像儲存至輸出檔案。您可以模擬此動作，方法是讓任務運算輸入雜湊，而不是轉譯影像。

使用檔案 I/O 建立 simple_file_job 任務套件

1. 選取您的第一個 CloudShell 索引標籤，然後導覽至任務套件範例目錄。

```
cd ~/deadline-cloud-samples/job_bundles/
```

2. 使用simple_job新名稱 複製 simple_file_job。

```
cp -r simple_job simple_file_job
```

3. 編輯任務範本，如下所示：

> **Note**
>
> 建議您將 nano用於這些步驟。如果您偏好使用 Vim，則必須使用 設定貼圖模式:set paste。

a. 在文字編輯器中開啟範本。

```
nano simple_file_job/template.yaml
```

b. 新增下列 type、 objectType和 dataFlow parameterDefinitions。

```
- name: InFile
  type: PATH
  objectType: FILE
  dataFlow: IN
- name: OutFile
  type: PATH
  objectType: FILE
  dataFlow: OUT
```

c. 將下列bash指令碼命令新增至從輸入檔案讀取並寫入輸出檔案的檔案結尾。

```
        # hash the input file, and write that to the output
        sha256sum "{{Param.InFile}}" > "{{Param.OutFile}}"
```

更新的 template.yaml應完全符合下列項目：

```
specificationVersion: 'jobtemplate-2023-09'
name: Simple File Job Bundle Example
parameterDefinitions:
- name: Message
  type: STRING
  default: Welcome to AWS Deadline Cloud!
- name: InFile
  type: PATH
```

```
  objectType: FILE
  dataFlow: IN
- name: OutFile
  type: PATH
  objectType: FILE
  dataFlow: OUT
steps:
- name: WelcomeToDeadlineCloud
  script:
    actions:
      onRun:
        command: '{{Task.File.Run}}'
    embeddedFiles:
    - name: Run
      type: TEXT
      runnable: true
      data: |
        #!/usr/bin/env bash
        echo "{{Param.Message}}"

        # hash the input file, and write that to the output
        sha256sum "{{Param.InFile}}" > "{{Param.OutFile}}"
```

> **Note**
>
> 如果您想要調整 中的間距`template.yaml`，請務必使用空格而非縮排。

d. 儲存檔案，然後結束文字編輯器。

4. 提供輸入和輸出檔案的參數值，以提交 simple_file_job。

```
deadline bundle submit simple_file_job \
    -p "InFile=simple_job/template.yaml" \
    -p "OutFile=hash.txt"
```

5. 列印任務的相關資訊。

```
deadline job get
```

- 您將看到如下所示的輸出：

```
parameters:
```

```
  Message:
    string: Welcome to AWS Deadline Cloud!
  InFile:
    path: /local/home/cloudshell-user/BundleFiles/JobBundle-Examples/simple_job/
template.yaml
  OutFile:
    path: /local/home/cloudshell-user/BundleFiles/JobBundle-Examples/hash.txt
```

- 雖然您只提供相對路徑，但參數會設定完整的路徑。當路徑類型為 時， 會將目前的工作目錄 AWS CLI 聯結至做為參數提供的任何路徑PATH。
- 在其他終端機視窗中執行的工作者代理程式會挑選並執行任務。此動作會建立 檔案，您可以使用下列命令檢視該hash.txt檔案。

```
cat hash.txt
```

此命令會列印類似以下的輸出。

```
eaa2df5d34b54be5ac34c56a24a8c237b8487231a607eaf530a04d76b89c9cd3 /local/home/
cloudshell-user/BundleFiles/JobBundle-Examples/simple_job/template.yaml
```

後續步驟

了解如何使用截止日期雲端 CLI 提交簡單任務之後，您可以探索：

- 在截止日期雲端中使用任務附件提交任務 了解如何在執行不同作業系統的主機上執行任務。
- 在截止日期雲端中將服務受管機群新增至您的開發人員陣列 在 Deadline Cloud 管理的主機上執行您的任務。
- 在截止日期雲端中清除您的陣列資源 關閉您用於本教學課程的資源。

在截止日期雲端中使用任務附件提交任務

許多陣列會使用共用檔案系統，在提交任務的主機與執行任務的主機之間共用檔案。例如，在先前的simple_file_job範例中，本機檔案系統是在 AWS CloudShell 終端機視窗之間共用，其會在您提交任務的索引標籤 1 中執行，並在您執行工作者代理程式的索引標籤 2 中執行。

當提交者工作站和工作者主機位於相同的區域網路上時，共用檔案系統是有利的。如果您將資料存放在靠近存取它的工作站的現場部署，則使用雲端型陣列表示您必須透過高延遲 VPN 共用檔案系統，或在雲端同步檔案系統。這些選項都不容易設定或操作。

AWS Deadline Cloud 提供具有任務附件的簡單解決方案，類似於電子郵件附件。使用任務附件，您可以將資料連接到任務。然後，截止日期雲端會處理在 Amazon Simple Storage Service (Amazon S3) 儲存貯體中傳輸和儲存任務資料的詳細資訊。

內容建立工作流程通常是反覆的，這表示使用者提交具有一小部分修改檔案的任務。由於 Amazon S3 儲存貯體會將任務附件存放在內容可定址的儲存體中，因此每個物件的名稱都是以物件資料的雜湊為基礎，目錄樹狀目錄的內容會以附加至任務的資訊清單檔案格式儲存。

您必須先完成下列操作，才能遵循本節中的程序：

- 建立截止日期雲端陣列
- 執行截止日期雲端工作者代理程式

若要使用任務附件執行任務，請完成下列步驟。

主題

將任務附件組態新增至佇列

若要在佇列中啟用任務附件，請將任務附件組態新增至帳戶中的佇列資源。

將任務附件組態新增至佇列

1. 選擇您的第一個 CloudShell 索引標籤，然後輸入下列其中一個命令，以使用 Amazon S3 儲存貯體做為任務附件。
   - 如果您沒有現有的私有 Amazon S3 儲存貯體，您可以建立和使用新的 S3 儲存貯體。

```
DEV_FARM_BUCKET=$(echo $DEV_FARM_NAME \
    | tr '[:upper:]' '[:lower:]')-$(xxd -l 16 -p /dev/urandom)
if [ "$AWS_REGION" == "us-east-1" ]; then LOCATION_CONSTRAINT=
```

```
else LOCATION_CONSTRAINT="--create-bucket-configuration \
    LocationConstraint=${AWS_REGION}"
fi
aws s3api create-bucket \
    $LOCATION_CONSTRAINT \
    --acl private \
    --bucket ${DEV_FARM_BUCKET}
```

- 如果您已有私有 Amazon S3 儲存貯體，您可以使用它，方法是將 *MY_BUCKET_NAME*取代為儲存貯體的名稱。

```
DEV_FARM_BUCKET=MY_BUCKET_NAME
```

2. 建立或選擇 Amazon S3 儲存貯體之後，請將儲存貯體名稱新增至 `~/.bashrc`，讓儲存貯體可供其他終端機工作階段使用。

```
echo "DEV_FARM_BUCKET=$DEV_FARM_BUCKET" >> ~/.bashrc
 source ~/.bashrc
```

3. 為佇列建立 AWS Identity and Access Management (IAM) 角色。

```
aws iam create-role --role-name "${DEV_FARM_NAME}QueueRole" \
    --assume-role-policy-document \
        '{
            "Version": "2012-10-17",
            "Statement": [
                {
                    "Effect": "Allow",
                    "Principal": {
                        "Service": "credentials.deadline.amazonaws.com"
                    },
                    "Action": "sts:AssumeRole"
                }
            ]
        }'
aws iam put-role-policy \
    --role-name "${DEV_FARM_NAME}QueueRole" \
    --policy-name S3BucketsAccess \
    --policy-document \
            '{
                "Version": "2012-10-17",
                "Statement": [
                {
```

```
                "Action": [
                    "s3:GetObject*",
                    "s3:GetBucket*",
                    "s3:List*",
                    "s3:DeleteObject*",
                    "s3:PutObject",
                    "s3:PutObjectLegalHold",
                    "s3:PutObjectRetention",
                    "s3:PutObjectTagging",
                    "s3:PutObjectVersionTagging",
                    "s3:Abort*"
                ],
                "Resource": [
                    "arn:aws:s3:::'$DEV_FARM_BUCKET'",
                    "arn:aws:s3:::'$DEV_FARM_BUCKET'/*"
                ],
                "Effect": "Allow"
            }
        ]
        }'
```

4. 更新您的佇列以包含任務附件設定和 IAM 角色。

```
QUEUE_ROLE_ARN="arn:aws:iam::$(aws sts get-caller-identity \
        --query "Account" --output text):role/${DEV_FARM_NAME}QueueRole"
aws deadline update-queue \
    --farm-id $DEV_FARM_ID \
    --queue-id $DEV_QUEUE_ID \
    --role-arn $QUEUE_ROLE_ARN \
    --job-attachment-settings \
        '{
            "s3BucketName": "'$DEV_FARM_BUCKET'",
            "rootPrefix": "JobAttachments"
        }'
```

5. 確認您已更新佇列。

```
deadline queue get
```

如下所示的輸出：

```
...
jobAttachmentSettings:
```

```
  s3BucketName: DEV_FARM_BUCKET
  rootPrefix: JobAttachments
roleArn: arn:aws:iam::ACCOUNT_NUMBER:role/DeveloperFarmQueueRole
...
```

simple_file_job 使用任務附件提交

當您使用任務附件時，任務套件必須提供 Deadline Cloud 足夠的資訊來判斷任務的資料流程，例如使用PATH參數。在 的情況下simple_file_job，您編輯了 `template.yaml` 檔案，以告知 Deadline Cloud 資料流程位於輸入檔案和輸出檔案中。

將任務附件組態新增至佇列後，您可以提交具有任務附件的 simple_file_job 範例。執行此操作後，您可以檢視記錄和任務輸出，以確認simple_file_job具有任務附件的 正常運作。

使用任務附件提交 simple_file_job 任務套件

1. 選擇您的第一個 CloudShell 標籤，然後開啟`JobBundle-Samples`目錄。
2.
```
cd ~/deadline-cloud-samples/job_bundles/
```
3. 將 simple_file_job 提交至佇列。出現確認上傳的提示時，請輸入 **y**。

```
deadline bundle submit simple_file_job \
    -p InFile=simple_job/template.yaml \
    -p OutFile=hash-jobattachments.txt
```
4. 若要檢視任務連接資料傳輸工作階段日誌輸出，請執行下列命令。

```
JOB_ID=$(deadline config get defaults.job_id)
SESSION_ID=$(aws deadline list-sessions \
        --farm-id $DEV_FARM_ID \
        --queue-id $DEV_QUEUE_ID \
        --job-id $JOB_ID \
        --query "sessions[0].sessionId" \
        --output text)
cat ~/demoenv-logs/$DEV_QUEUE_ID/$SESSION_ID.log
```
5. 列出在工作階段中執行的工作階段動作。

```
aws deadline list-session-actions \
    --farm-id $DEV_FARM_ID \
```

```
    --queue-id $DEV_QUEUE_ID \
    --job-id $JOB_ID \
    --session-id $SESSION_ID
```

如下所示的輸出：

```
{
    "sessionactions": [
        {
            "sessionActionId": "sessionaction-123-0",
            "status": "SUCCEEDED",
            "startedAt": "<timestamp>",
            "endedAt": "<timestamp>",
            "progressPercent": 100.0,
            "definition": {
                "syncInputJobAttachments": {}
            }
        },
        {
            "sessionActionId": "sessionaction-123-1",
            "status": "SUCCEEDED",
            "startedAt": "<timestamp>",
            "endedAt": "<timestamp>",
            "progressPercent": 100.0,
            "definition": {
                "taskRun": {
                    "taskId": "task-abc-0",
                    "stepId": "step-def"
                }
            }
        }
    ]
}
```

第一個工作階段動作會下載輸入任務附件，而第二個動作會像先前步驟一樣執行任務，然後上傳輸出任務附件。

6. 列出輸出目錄。

```
ls *.txt
```

目錄中hash.txt存在 等輸出，但hash-jobattachments.txt由於任務的輸出檔案尚未下載，因此不存在。

7. 從最近的任務下載輸出。

```
deadline job download-output
```

8. 檢視下載檔案的輸出。

```
cat hash-jobattachments.txt
```

如下所示的輸出：

```
eaa2df5d34b54be5ac34c56a24a8c237b8487231a607eaf530a04d76b89c9cd3  /tmp/openjd/
session-123/assetroot-abc/simple_job/template.yaml
```

了解任務附件如何儲存在 Amazon S3 中

您可以使用 AWS Command Line Interface (AWS CLI) 上傳或下載存放在 Amazon S3 儲存貯體的任務附件資料。了解 Deadline Cloud 如何在 Amazon S3 上存放任務附件，有助於您開發工作負載和管道整合。

檢查截止日期雲端任務附件在 Amazon S3 中的存放方式

1. 選擇您的第一個 CloudShell 索引標籤，然後開啟任務套件範例目錄。

```
cd ~/deadline-cloud-samples/job_bundles/
```

2. 檢查任務屬性。

```
deadline job get
```

如下所示的輸出：

```
parameters:
  Message:
    string: Welcome to AWS Deadline Cloud!
  InFile:
```

```
    path: /home/cloudshell-user/deadline-cloud-samples/job_bundles/simple_job/
template.yaml
  OutFile:
    path: /home/cloudshell-user/deadline-cloud-samples/job_bundles/hash-
jobattachments.txt
attachments:
  manifests:
  - rootPath: /home/cloudshell-user/deadline-cloud-samples/job_bundles/
    rootPathFormat: posix
    outputRelativeDirectories:
    - .
    inputManifestPath: farm-3040c59a5b9943d58052c29d907a645d/queue-
cde9977c9f4d4018a1d85f3e6c1a4e6e/Inputs/
f46af01ca8904cd8b514586671c79303/0d69cd94523ba617c731f29c019d16e8_input.xxh128
    inputManifestHash: f95ef91b5dab1fc1341b75637fe987ee
  fileSystem: COPIED
```

附件欄位包含資訊清單結構清單，描述任務執行時使用的輸入和輸出資料路徑。查看 `rootPath` 以查看提交任務之電腦上的本機目錄路徑。若要檢視包含資訊清單檔案的 Amazon S3 物件尾碼，請檢閱 `inputManifestFile`。資訊清單檔案包含任務輸入資料的目錄樹狀目錄快照的中繼資料。

3. Pretty 列印 Amazon S3 資訊清單物件，以查看任務的輸入目錄結構。

```
MANIFEST_SUFFIX=$(aws deadline get-job \
     --farm-id $DEV_FARM_ID \
     --queue-id $DEV_QUEUE_ID \
     --job-id $JOB_ID \
     --query "attachments.manifests[0].inputManifestPath" \
     --output text)
 aws s3 cp s3://$DEV_FARM_BUCKET/JobAttachments/Manifests/$MANIFEST_SUFFIX - | jq .
```

如下所示的輸出：

```
{
     "hashAlg": "xxh128",
     "manifestVersion": "2023-03-03",
     "paths": [
     {
         "hash": "2ec297b04c59c4741ed97ac8fb83080c",
         "mtime": 1698186190000000,
         "path": "simple_job/template.yaml",
```

```
            "size": 445
        }
        ],
        "totalSize": 445
    }
```

4. 建構 Amazon S3 字首，其中包含輸出任務附件的資訊清單，並在其中列出物件。

```
SESSION_ACTION=$(aws deadline list-session-actions \
    --farm-id $DEV_FARM_ID \
    --queue-id $DEV_QUEUE_ID \
    --job-id $JOB_ID \
    --session-id $SESSION_ID \
    --query "sessionActions[?definition.taskRun != null] | [0]")
STEP_ID=$(echo $SESSION_ACTION | jq -r .definition.taskRun.stepId)
TASK_ID=$(echo $SESSION_ACTION | jq -r .definition.taskRun.taskId)
TASK_OUTPUT_PREFIX=JobAttachments/Manifests/$DEV_FARM_ID/$DEV_QUEUE_ID/$JOB_ID/
$STEP_ID/$TASK_ID/
aws s3api list-objects-v2 --bucket $DEV_FARM_BUCKET --prefix $TASK_OUTPUT_PREFIX
```

輸出任務附件不會直接從任務資源中參考，而是根據陣列資源 IDs 放置在 Amazon S3 儲存貯體中。

5. 取得特定工作階段動作 ID 的最新資訊清單物件金鑰，然後漂亮地列印資訊清單物件。

```
SESSION_ACTION_ID=$(echo $SESSION_ACTION | jq -r .sessionActionId)
 MANIFEST_KEY=$(aws s3api list-objects-v2 \
     --bucket $DEV_FARM_BUCKET \
     --prefix $TASK_OUTPUT_PREFIX \
     --query "Contents[*].Key" --output text \
     | grep $SESSION_ACTION_ID \
     | sort | tail -1)
 MANIFEST_OBJECT=$(aws s3 cp s3://$DEV_FARM_BUCKET/$MANIFEST_KEY -)
 echo $MANIFEST_OBJECT | jq .
```

您會在輸出hash-jobattachments.txt中看到 檔案的屬性，如下所示：

```
{
     "hashAlg": "xxh128",
     "manifestVersion": "2023-03-03",
     "paths": [
     {
```

```
            "hash": "f60b8e7d0fabf7214ba0b6822e82e08b",
            "mtime": 1698785252554950,
            "path": "hash-jobattachments.txt",
            "size": 182
        }
        ],
        "totalSize": 182
    }
```

您的任務每個任務執行只會有單一資訊清單物件，但一般而言，每個任務執行可能會有更多物件。

6. 在Data字首下檢視內容可定址的 Amazon S3 儲存輸出。

```
FILE_HASH=$(echo $MANIFEST_OBJECT | jq -r .paths[0].hash)
FILE_PATH=$(echo $MANIFEST_OBJECT | jq -r .paths[0].path)
aws s3 cp s3://$DEV_FARM_BUCKET/JobAttachments/Data/$FILE_HASH -
```

如下所示的輸出：

```
eaa2df5d34b54be5ac34c56a24a8c237b8487231a607eaf530a04d76b89c9cd3  /tmp/openjd/
session-123/assetroot-abc/simple_job/template.yaml
```

後續步驟

了解如何使用截止日期雲端 CLI 提交具有附件的任務之後，您可以探索：

- 使用截止日期雲端提交 了解如何在工作者主機上使用 OpenJD 套件執行任務。
- 在截止日期雲端中將服務受管機群新增至您的開發人員陣列 在 Deadline Cloud 管理的主機上執行您的任務。
- 在截止日期雲端中清除您的陣列資源 關閉您用於本教學課程的資源。

在截止日期雲端中將服務受管機群新增至您的開發人員陣列

AWS CloudShell 無法提供足夠的運算容量來測試更大的工作負載。它也不會設定為使用在多個工作者主機上分配任務的任務。

您可以新增 Auto Scaling 服務受管機群 (SMF) 到您的開發人員陣列，而不是使用 CloudShell。SMF 可為較大的工作負載提供足夠的運算容量，並可處理需要將任務分配到多個工作者主機的任務。

在新增 SMF 之前，您必須設定截止日期雲端陣列、佇列和機群。請參閱 建立截止日期雲端陣列。

將服務受管機群新增至您的開發人員陣列

1. 選擇您的第一個 AWS CloudShell 索引標籤，然後建立服務受管機群，並將其機群 ID 新增至 `.bashrc`。此動作可供其他終端機工作階段使用。

```
FLEET_ROLE_ARN="arn:aws:iam::$(aws sts get-caller-identity \
        --query "Account" --output text):role/${DEV_FARM_NAME}FleetRole"
 aws deadline create-fleet \
     --farm-id $DEV_FARM_ID \
     --display-name "$DEV_FARM_NAME SMF" \
     --role-arn $FLEET_ROLE_ARN \
     --max-worker-count 5 \
     --configuration \
         '{
             "serviceManagedEc2": {
                 "instanceCapabilities": {
                     "vCpuCount": {
                         "min": 2,
                         "max": 4
                     },
                     "memoryMiB": {
                         "min": 512
                     },
                     "osFamily": "linux",
                     "cpuArchitectureType": "x86_64"
                 },
                 "instanceMarketOptions": {
                     "type": "spot"
                 }
             }
         }'

 echo "DEV_SMF_ID=$(aws deadline list-fleets \
         --farm-id $DEV_FARM_ID \
         --query "fleets[?displayName=='$DEV_FARM_NAME SMF'].fleetId \
         | [0]" --output text)" >> ~/.bashrc
 source ~/.bashrc
```

2. 將 SMF 與您的佇列建立關聯。

```
aws deadline create-queue-fleet-association \
     --farm-id $DEV_FARM_ID \
     --queue-id $DEV_QUEUE_ID \
     --fleet-id $DEV_SMF_ID
```

3. 提交simple_file_job至佇列。出現確認上傳的提示時，請輸入 **y**。

```
deadline bundle submit simple_file_job \
    -p InFile=simple_job/template.yaml \
    -p OutFile=hash-jobattachments.txt
```

4. 確認 SMF 正常運作。

```
deadline fleet get
```

   - 工作者可能需要幾分鐘的時間才能開始。重複 `deadline fleet get`命令，直到您可以看到機群正在執行。
   - `queueFleetAssociationsStatus` 適用於服務受管機群的 將是 `ACTIVE`。
   - SMF `autoScalingStatus`將從 變更為 `GROWING` `STEADY`。

   您的狀態看起來會類似以下內容：

```
fleetId: fleet-2cc78e0dd3f04d1db427e7dc1d51ea44
farmId: farm-63ee8d77cdab4a578b685be8c5561c4a
displayName: DeveloperFarm SMF
description: ''
status: ACTIVE
autoScalingStatus: STEADY
targetWorkerCount: 0
workerCount: 0
minWorkerCount: 0
maxWorkerCount: 5
```

5. 檢視您提交之任務的日誌。此日誌存放在 Amazon CloudWatch Logs 的日誌中，而不是 CloudShell 檔案系統。

```
 JOB_ID=$(deadline config get defaults.job_id)
 SESSION_ID=$(aws deadline list-sessions \
```

```
        --farm-id $DEV_FARM_ID \
        --queue-id $DEV_QUEUE_ID \
        --job-id $JOB_ID \
        --query "sessions[0].sessionId" \
        --output text)
aws logs tail /aws/deadline/$DEV_FARM_ID/$DEV_QUEUE_ID \
    --log-stream-names $SESSION_ID
```

後續步驟

在建立和測試服務受管機群之後，您應該移除您建立的資源，以避免不必要的費用。

- 在截止日期雲端中清除您的陣列資源 關閉您用於本教學課程的資源。

在截止日期雲端中清除您的陣列資源

若要開發和測試新的工作負載和管道整合，您可以繼續使用您在本教學課程中建立的截止日期雲端開發人員陣列。如果您不再需要開發人員陣列，可以在 Amazon CloudWatch Logs 中刪除其資源，包括陣列、機群、佇列、 AWS Identity and Access Management (IAM) 角色和日誌。刪除這些資源後，您需要再次開始教學課程才能使用這些資源。如需詳細資訊，請參閱截止日期雲端資源入門。

清除開發人員陣列資源

1. 選擇您的第一個 CloudShell 標籤，然後停止佇列的所有佇列機群關聯。

```
FLEETS=$(aws deadline list-queue-fleet-associations \
        --farm-id $DEV_FARM_ID \
        --queue-id $DEV_QUEUE_ID \
        --query "queueFleetAssociations[].fleetId" \
        --output text)
for FLEET_ID in $FLEETS; do
    aws deadline update-queue-fleet-association \
        --farm-id $DEV_FARM_ID \
        --queue-id $DEV_QUEUE_ID \
        --fleet-id $FLEET_ID \
        --status STOP_SCHEDULING_AND_CANCEL_TASKS
done
```

2. 列出佇列機群關聯。

```
aws deadline list-queue-fleet-associations \
     --farm-id $DEV_FARM_ID \
     --queue-id $DEV_QUEUE_ID
```

您可能需要重新執行命令，直到輸出報告 "status": "STOPPED"，然後您可以繼續下一個步驟。此程序可能需要幾分鐘的時間才能完成。

```
{
    "queueFleetAssociations": [
        {
            "queueId": "queue-abcdefgh01234567890123456789012id",
            "fleetId": "fleet-abcdefgh01234567890123456789012id",
            "status": "STOPPED",
            "createdAt": "2023-11-21T20:49:19+00:00",
            "createdBy": "arn:aws:sts::123456789012:assumed-role/RoleToBeAssumed/
MySessionName",
            "updatedAt": "2023-11-21T20:49:38+00:00",
            "updatedBy": "arn:aws:sts::123456789012:assumed-role/RoleToBeAssumed/
MySessionName"
        },
        {
            "queueId": "queue-abcdefgh01234567890123456789012id",
            "fleetId": "fleet-abcdefgh01234567890123456789012id",
            "status": "STOPPED",
            "createdAt": "2023-11-21T20:32:06+00:00",
            "createdBy": "arn:aws:sts::123456789012:assumed-role/RoleToBeAssumed/
MySessionName",
            "updatedAt": "2023-11-21T20:49:39+00:00",
            "updatedBy": "arn:aws:sts::123456789012:assumed-role/RoleToBeAssumed/
MySessionName"
        }
    ]
}
```

3. 刪除佇列的所有佇列機群關聯。

```
for FLEET_ID in $FLEETS; do
     aws deadline delete-queue-fleet-association \
         --farm-id $DEV_FARM_ID \
         --queue-id $DEV_QUEUE_ID \
         --fleet-id $FLEET_ID
```

```
 done
```

4. 刪除與佇列相關聯的所有機群。

```
for FLEET_ID in $FLEETS; do
     aws deadline delete-fleet \
         --farm-id $DEV_FARM_ID \
         --fleet-id $FLEET_ID
 done
```

5. 刪除佇列。

```
aws deadline delete-queue \
     --farm-id $DEV_FARM_ID \
     --queue-id $DEV_QUEUE_ID
```

6. 刪除陣列。

```
aws deadline delete-farm \
     --farm-id $DEV_FARM_ID
```

7. 刪除陣列的其他 AWS 資源。

   a. 刪除 fleet AWS Identity and Access Management (IAM) 角色。

```
aws iam delete-role-policy \
     --role-name "${DEV_FARM_NAME}FleetRole" \
     --policy-name WorkerPermissions
aws iam delete-role \
     --role-name "${DEV_FARM_NAME}FleetRole"
```

   b. 刪除佇列 IAM 角色。

```
aws iam delete-role-policy \
     --role-name "${DEV_FARM_NAME}QueueRole" \
     --policy-name S3BucketsAccess
aws iam delete-role \
     --role-name "${DEV_FARM_NAME}QueueRole"
```

   c. 刪除 Amazon CloudWatch Logs 日誌群組。每個佇列和機群都有自己的日誌群組。

```
aws logs delete-log-group \
     --log-group-name "/aws/deadline/$DEV_FARM_ID/$DEV_QUEUE_ID"
```

```
aws logs delete-log-group \
     --log-group-name "/aws/deadline/$DEV_FARM_ID/$DEV_CMF_ID"
aws logs delete-log-group \
     --log-group-name "/aws/deadline/$DEV_FARM_ID/$DEV_SMF_ID"
```

建置任務以提交至截止日期雲端

您可以使用任務套件將任務提交至截止日期雲端。任務套件是檔案的集合，包括 Open Job Description (OpenJD) 任務範本，以及轉譯任務所需的任何資產檔案。

任務範本說明工作者如何處理和存取資產，並提供工作者執行的指令碼。任務套件可讓藝術家、技術主管和管道開發人員輕鬆從本機工作站或內部部署轉譯陣列將複雜的任務提交至 Deadline Cloud。任務套件特別適用於處理大規模視覺效果、動畫或其他需要可擴展隨需運算資源之媒體轉譯專案的團隊。

您可以使用本機檔案系統來儲存檔案，並使用文字編輯器來建立任務範本，進而建立任務套件。建立套件後，使用截止日期雲端 CLI 或截止日期雲端提交工具等工具，將任務提交至截止日期雲端

您可以將資產存放在工作者之間共用的檔案系統中，也可以使用截止日期雲端任務附件，自動將資產移至工作者可以存取它們的 S3 儲存貯體。任務連接也有助於將輸出從任務移回工作站。

下列各節提供建立任務套件並將其提交至截止日期雲端的詳細說明。

主題

- 截止日期雲端的開啟任務描述 (OpenJD) 範本
- 在任務中使用檔案
- 使用任務附件來共用檔案
- 建立任務的資源限制
- 如何將任務提交至截止日期雲端
- 在截止日期雲端中排程任務
- 在截止日期雲端中修改任務

截止日期雲端的開啟任務描述 (OpenJD) 範本

任務套件是您用來定義 AWS 截止日期雲端任務的工具之一。它們將開放任務描述 (OpenJD) 範本分組，其中包含其他資訊，例如任務與任務附件搭配使用的檔案和目錄。您可以使用截止日期雲端命令列界面 (CLI) 來使用任務套件提交任務，以便佇列執行。

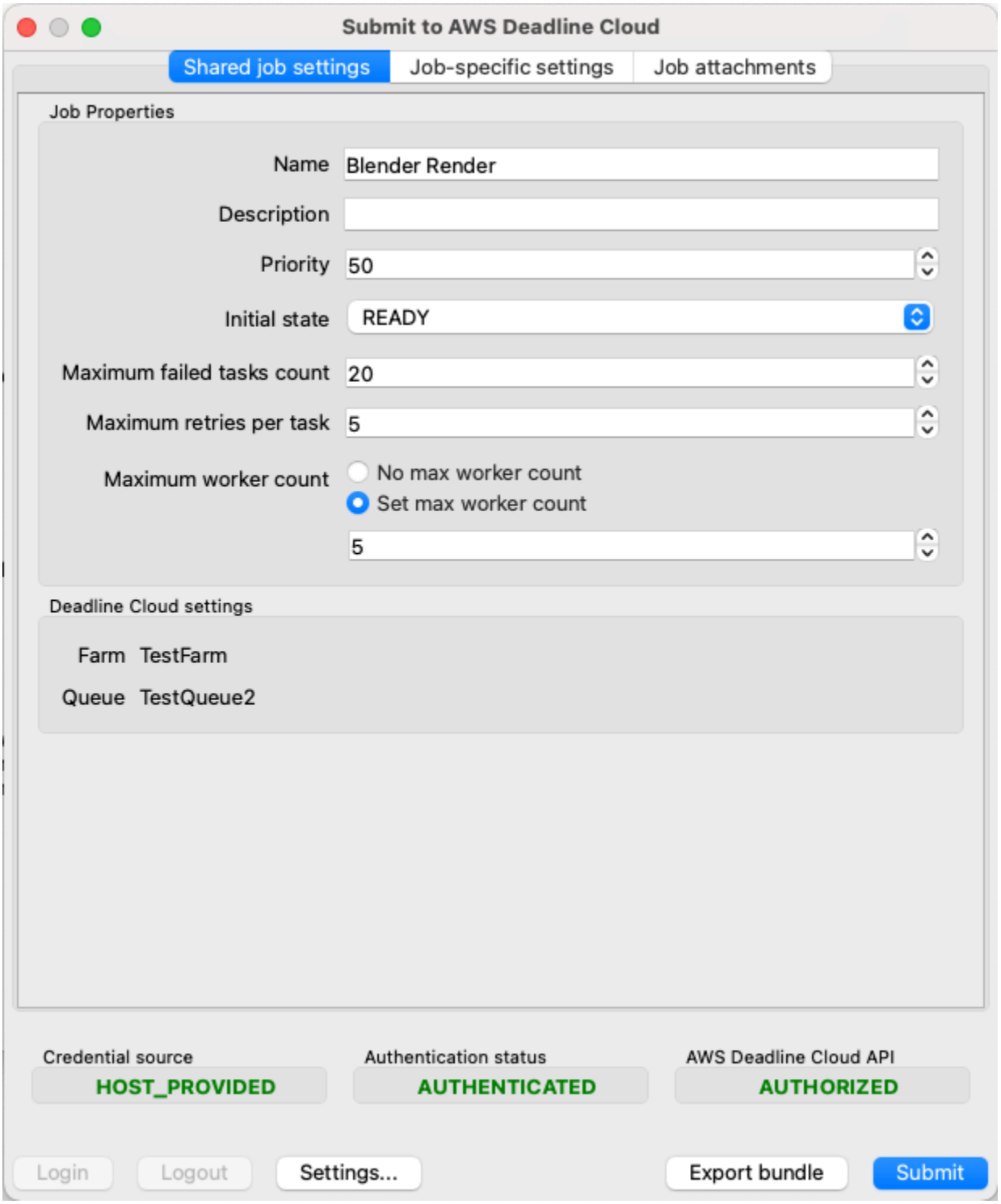
任務套件是一種目錄結構，其中包含 OpenJD 任務範本、定義任務的其他檔案，以及做為任務輸入所需的任務特定檔案。您可以指定將任務定義為 YAML 或 JSON 檔案的檔案。

唯一的必要檔案是 `template.yaml`或 `template.json`。您也可以包含下列檔案：

```
/template.yaml (or template.json)
/asset_references.yaml (or asset_references.json)
/parameter_values.yaml (or parameter_values.json)
/other job-specific files and directories
```

使用任務套件搭配截止日期雲端 CLI 和任務附件進行自訂任務提交，或者您可以使用圖形提交界面。例如，以下是來自 GitHub 的 Blender 範例。若要在 Blender 範例目錄中使用下列命令執行範例：

```
deadline bundle gui-submit blender_render
```



任務特定的設定面板是從任務範本中定義之任務參數的`userInterface`屬性產生。

若要使用命令列提交任務，您可以使用類似以下的命令

```
deadline bundle submit \
```

```
    --yes \
    --name Demo \
     -p BlenderSceneFile=location of scene file \
     -p OutputDir=file pathe for job output \
      blender_render/
```

或者，您可以在 deadline Python 套件中使用 deadline.client.api.create_job_from_job_bundle函數。

隨附於 Deadline Cloud 的所有任務提交者外掛程式，例如 Autodesk Maya 外掛程式，為您的提交產生任務套件，然後使用 Deadline Cloud Python 套件將您的任務提交至 Deadline Cloud。您可以在工作站的任務歷史記錄目錄中或使用提交者來查看提交的任務套件。您可以使用下列命令來尋找任務歷史記錄目錄：

```
deadline config get settings.job_history_dir
```

當您的任務在截止日期雲端工作者上執行時，它可以存取提供任務相關資訊的環境變數。環境變數為：

變數名稱	Available
DEADLINE_FARM_ID	所有動作
DEADLINE_FLEET_ID	所有動作
DEADLINE_WORKER_ID	所有動作
DEADLINE_QUEUE_ID	所有動作
DEADLINE_JOB_ID	所有動作
DEADLINE_STEP_ID	任務動作
DEADLINE_SESSION_ID	所有動作
DEADLINE_TASK_ID	任務動作
DEADLINE_SESSIONACTION_ID	所有動作

主題

- 任務套件的任務範本元素

- 任務範本的任務區塊
- 任務套件的參數值元素
- 任務套件的資產參考元素

任務套件的任務範本元素

任務範本會定義執行時間環境，以及做為截止日期雲端任務一部分執行的程序。您可以在範本中建立參數，以便用來建立只在輸入值中不同的任務，就像程式設計語言中的函數一樣。

當您將任務提交至截止日期雲端時，該任務會在套用至佇列的任何佇列環境中執行。佇列環境是使用 Open Job Description (OpenJD) 外部環境規格建置。如需詳細資訊，請參閱 OpenJD GitHub 儲存庫中的環境範本。

如需使用 OpenJD 任務範本建立任務的簡介，請參閱在 OpenJD GitHub 儲存庫中建立任務的簡介。如需其他資訊，請參閱執行任務的方式。OpenJD GitHub 儲存庫`samples`目錄中的 中有任務範本範例。

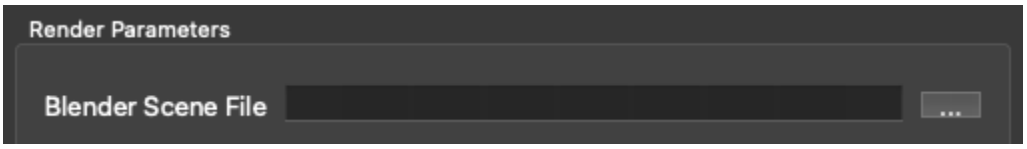
您可以 YAML 格式 (`template.yaml`) 或 JSON 格式 () 定義任務範本`template.json`。本節中的範例會以 YAML 格式顯示。

例如，`blender_render`範例的任務範本將輸入參數定義為`BlenderSceneFile`檔案路徑：

```
- name: BlenderSceneFile
  type: PATH
  objectType: FILE
  dataFlow: IN
  userInterface:
    control: CHOOSE_INPUT_FILE
    label: Blender Scene File
    groupLabel: Render Parameters
    fileFilters:
    - label: Blender Scene Files
      patterns: ["*.blend"]
    - label: All Files
      patterns: ["*"]
  description: >
    Choose the Blender scene file to render. Use the 'Job Attachments' tab
    to add textures and other files that the job needs.
```

`userInterface` 屬性會使用 命令在 Autodesk Maya 等應用程式的任務提交外掛程式中，定義`deadline bundle gui-submit`命令列自動產生之使用者介面的行為。

在此範例中，用於輸入 BlenderSceneFile 參數值的 UI 小工具是僅顯示.blend檔案的檔案選擇對話方塊。



如需使用 userInteface元素的更多範例，請參閱 GitHub 上 deadline-cloud-samples 儲存庫中的 gui_control_showcase 範例。

當您從任務套件提交任務時， objectType和 dataFlow 屬性會控制任務附件的行為。在此情況下， objectType: FILE 值dataFlow:IN表示 BlenderSceneFile 是任務附件的輸入檔案。

相反地， OutputDir 參數的定義具有 objectType: DIRECTORY和 dataFlow: OUT：

```
- name: OutputDir
  type: PATH
  objectType: DIRECTORY
  dataFlow: OUT
  userInterface:
    control: CHOOSE_DIRECTORY
    label: Output Directory
    groupLabel: Render Parameters
  default: "./output"
  description: Choose the render output directory.
```

任務附件會使用 OutputDir 參數的值做為任務寫入輸出檔案的目錄。

如需 objectType和 dataFlow 屬性的詳細資訊，請參閱 Open Job Description 規格中的 JobPathParameterDefinition

blender_render 任務範本範例的其餘部分會將任務的工作流程定義為單一步驟，動畫中的每個影格都會轉譯為個別任務：

```
steps:
- name: RenderBlender
  parameterSpace:
    taskParameterDefinitions:
    - name: Frame
      type: INT
      range: "{{Param.Frames}}"
  script:
```

```
    actions:
      onRun:
        command: bash
        # Note: {{Task.File.Run}} is a variable that expands to the filename on the
  worker host's
        # disk where the contents of the 'Run' embedded file, below, is written.
        args: ['{{Task.File.Run}}']
    embeddedFiles:
      - name: Run
        type: TEXT
        data: |
          # Configure the task to fail if any individual command fails.
          set -xeuo pipefail

          mkdir -p '{{Param.OutputDir}}'

          blender --background '{{Param.BlenderSceneFile}}' \
                  --render-output '{{Param.OutputDir}}/{{Param.OutputPattern}}' \
                  --render-format {{Param.Format}} \
                  --use-extension 1 \
                  --render-frame {{Task.Param.Frame}}
```

例如，如果 Frames 參數的值為 1-10，則會定義 10 個任務。每個 都有不同的 Frame 參數值。若要執行任務：

1. 內嵌檔案 data 屬性中的所有變數參考都會展開，例如 --render-frame 1。
2. data 屬性的內容會寫入磁碟上工作階段工作目錄中的檔案。
3. 任務的onRun命令會解析為 bash *location of embedded file*，然後執行。

如需內嵌檔案、工作階段和路徑映射位置的詳細資訊，請參閱 Open Job Description 規格中的任務執行方式。

在 deadline-cloud-samples/job_bundles 儲存庫中有更多任務範本的範例，以及 Open Job Descriptions 規格提供的範本範例。

任務範本的任務區塊

任務區塊可讓您將多個任務分組為稱為區塊的單一工作單位。例如，在轉譯任務中，這表示截止日期雲端可以將多個影格一起分派，而不是每個命令叫用一個影格。這可降低每個任務啟動應用程式的負荷，並縮短總任務執行時間。如需詳細資訊，請參閱 OpenJD Wiki 中的一次執行多個影格。

OpenJD 支援擴充功能，可將選用功能新增至任務範本。新增 TASK_CHUNKING延伸來啟用任務區塊。若要使用區塊，請將延伸模組新增至您的任務範本，並使用CHUNK[INT]任務參數類型。使用相同的deadline bundle submit命令提交區塊任務。例如，下列任務範本會以 10 的區塊轉譯影格：

```
specificationVersion: 'jobtemplate-2023-09'
extensions:
  - TASK_CHUNKING
name: Blender Render with Contiguous Chunking
parameterDefinitions:
  - name: BlenderSceneFile
    type: PATH
    objectType: FILE
    dataFlow: IN
  - name: Frames
    type: STRING
    default: "1-100"
  - name: OutputDir
    type: PATH
    objectType: DIRECTORY
    dataFlow: OUT
    default: "./output"
steps:
  - name: RenderBlender
    parameterSpace:
      taskParameterDefinitions:
        - name: Frame
          type: CHUNK[INT]
          range: "{{Param.Frames}}"
          chunks:
            defaultTaskCount: 10
            rangeConstraint: CONTIGUOUS
    script:
      actions:
        onRun:
          command: bash
          args: ["{{Task.File.Run}}"]
      embeddedFiles:
        - name: Run
          type: TEXT
          data: |
            set -xeuo pipefail

            mkdir -p '{{Param.OutputDir}}'
```

```
        # Parse the chunk range (e.g., "1-10") into start and end frames
        START_FRAME="$(echo '{{Task.Param.Frame}}' | cut -d- -f1)"
        END_FRAME="$(echo '{{Task.Param.Frame}}' | cut -d- -f2)"

        blender --background '{{Param.BlenderSceneFile}}' \
                --render-output '{{Param.OutputDir}}/output_####' \
                --render-format PNG \
                --use-extension 1 \
                -s "$START_FRAME" \
                -e "$END_FRAME" \
                --render-anim
```

在此範例中，Deadline Cloud 會將 100 個影格分成 1-10、 11-20等區塊。{{Task.Param.Frame}} 變數會展開至範圍表達式，例如 1-10。因為 rangeConstraint 設為 CONTIGUOUS，所以範圍一律為 start-end 格式。指令碼剖析此範圍，並使用 -s和 -e選項將開始和結束影格傳遞給 Blender--render-anim。

chunks 屬性支援下列欄位：

- defaultTaskCount – （必要） 要合併為單一區塊的任務數量。最大值為 150。
- rangeConstraint – （必要） 如果 CONTIGUOUS，區塊一律為連續範圍，例如 1-10。如果為 NONCONTIGUOUS，區塊可以是任意集，如 1,3,7-10。
- targetRuntimeSeconds – （選用） 每個區塊的目標執行時間，以秒為單位。截止日期 雲端可以動態調整區塊大小，在某些區塊完成後接近此目標。

如需更多任務區塊範例，包括具有連續和非連續區塊的基本和 Blender 範例，請參閱 GitHub 上截止日期雲端範例儲存庫中的任務區塊範例。

> **客戶受管機群需求**
>
> 任務區塊需要相容的工作者代理程式版本。如果您使用客戶管理的機群，請確保在提交具有區塊的工作之前更新您的工作者代理程式。服務受管機群一律使用相容的工作者代理程式版本。

> **下載區塊任務的輸出**
>
> 當您下載區塊任務中單一任務的輸出時，Deadline Cloud 會下載整個區塊的輸出。例如，如果同時處理影格 1-10，則下載影格 3 的輸出會包含所有影格 1-10。此功能需要 0deadline-cloud.53.3 版或更新版本。

任務套件的參數值元素

您可以使用 參數檔案來設定任務範本中某些任務參數的值，或在任務套件中設定 CreateJob 操作請求引數的值，以便在提交任務時不需要設定值。任務提交的 UI 可讓您修改這些值。

您可以 YAML 格式 (`parameter_values.yaml`) 或 JSON 格式 () 定義任務範本`parameter_values.json`。本節中的範例會以 YAML 格式顯示。

在 YAML 中，檔案格式為：

```
parameterValues:
- name: <string>
  value: <integer>, <float>, or <string>
- name: <string>
  value: <integer>, <float>, or <string>ab
... repeating as necessary
```

`parameterValues` 清單的每個元素都必須是下列其中一項：

- 任務範本中定義的任務參數。
- 在您提交任務之佇列的佇列環境中定義的任務參數。
- 建立任務時傳遞至 `CreateJob`操作的特殊參數。
  - `deadline:priority` – 值必須是整數。它會做為優先順序參數傳遞給 `CreateJob`操作。
  - `deadline:targetTaskRunStatus` – 值必須是字串。它會做為 targetTaskRunStatus 參數傳遞至 `CreateJob`操作。
  - `deadline:maxFailedTasksCount` – 值必須是整數。它會做為 maxFailedTasksCount 參數傳遞至 `CreateJob`操作。
  - `deadline:maxRetriesPerTask` – 值必須是整數。它會做為 maxRetriesPerTask 參數傳遞至 `CreateJob`操作。
  - `deadline:maxWorkercount` – 值必須是整數。它會做為 maxWorkerCount 參數傳遞至 `CreateJob`操作。

任務範本一律是範本，而不是要執行的特定任務。參數值檔案可讓任務套件做為範本，如果某些參數沒有在此檔案中定義的值，則做為特定任務提交，如果所有參數都具有值，則做為特定任務提交。

例如， blender_render 範例沒有參數檔案，其任務範本會定義沒有預設值的參數。此範本必須用作建立任務的範本。使用此任務套件建立任務後，Deadline Cloud 會將新的任務套件寫入任務歷史記錄目錄。

例如，當您使用下列命令提交任務時：

```
deadline bundle gui-submit blender_render/
```

新的任務套件包含的檔案parameter_values.yaml包含指定的參數：

```
% cat ~/.deadline/job_history/\(default\)/2024-06/2024-06-20-01-JobBundle-Demo/
parameter_values.yaml
parameterValues:
- name: deadline:targetTaskRunStatus
  value: READY
- name: deadline:maxFailedTasksCount
  value: 10
- name: deadline:maxRetriesPerTask
  value: 5
- name: deadline:priority
  value: 75
- name: BlenderSceneFile
  value: /private/tmp/bundle_demo/bmw27_cpu.blend
- name: Frames
  value: 1-10
- name: OutputDir
  value: /private/tmp/bundle_demo/output
- name: OutputPattern
  value: output_####
- name: Format
  value: PNG
- name: CondaPackages
  value: blender
- name: RezPackages
  value: blender
```

您可以使用下列命令建立相同的任務：

```
deadline bundle submit ~/.deadline/job_history/\(default\)/2024-06/2024-06-20-01-
JobBundle-Demo/
```

> **Note**
>
> 您提交的任務套件會儲存至您的任務歷史記錄目錄。您可以使用下列命令找到該目錄的位置：
>
> ```
> deadline config get settings.job_history_dir
> ```

任務套件的資產參考元素

您可以使用 Deadline Cloud 任務附件，在工作站和 Deadline Cloud 之間來回傳輸檔案。資產參考檔案會列出輸入檔案和目錄，以及附件的輸出目錄。如果您未列出此檔案中的所有檔案和目錄，您可以在使用 `deadline bundle gui-submit`命令提交任務時選取它們。

如果您不使用任務附件，則此檔案沒有作用。

您可以 YAML 格式 (`asset_references.yaml`) 或 JSON 格式 () 定義任務範本`asset_references.json`。本節中的範例會以 YAML 格式顯示。

在 YAML 中，檔案格式為：

```
assetReferences:
    inputs:
        # Filenames on the submitting workstation whose file contents are needed as
        # inputs to run the job.
        filenames:
        - list of file paths
        # Directories on the submitting workstation whose contents are needed as inputs
        # to run the job.
        directories:
        - list of directory paths

    outputs:
        # Directories on the submitting workstation where the job writes output files
        # if running locally.
        directories:
        - list of directory paths
```

```
    # Paths referenced by the job, but not necessarily input or output.
    # Use this if your job uses the name of a path in some way, but does not explicitly
need
    # the contents of that path.
    referencedPaths:
    - list of directory paths
```

選取要上傳至 Amazon S3 的輸入或輸出檔案時，截止日期雲端會將檔案路徑與儲存設定檔中列出的路徑進行比較。儲存描述檔中的每個 SHARED類型檔案系統位置都會摘要掛載在工作站和工作者主機上的網路檔案共用。截止日期 雲端只會上傳不在其中一個檔案共享上的檔案。

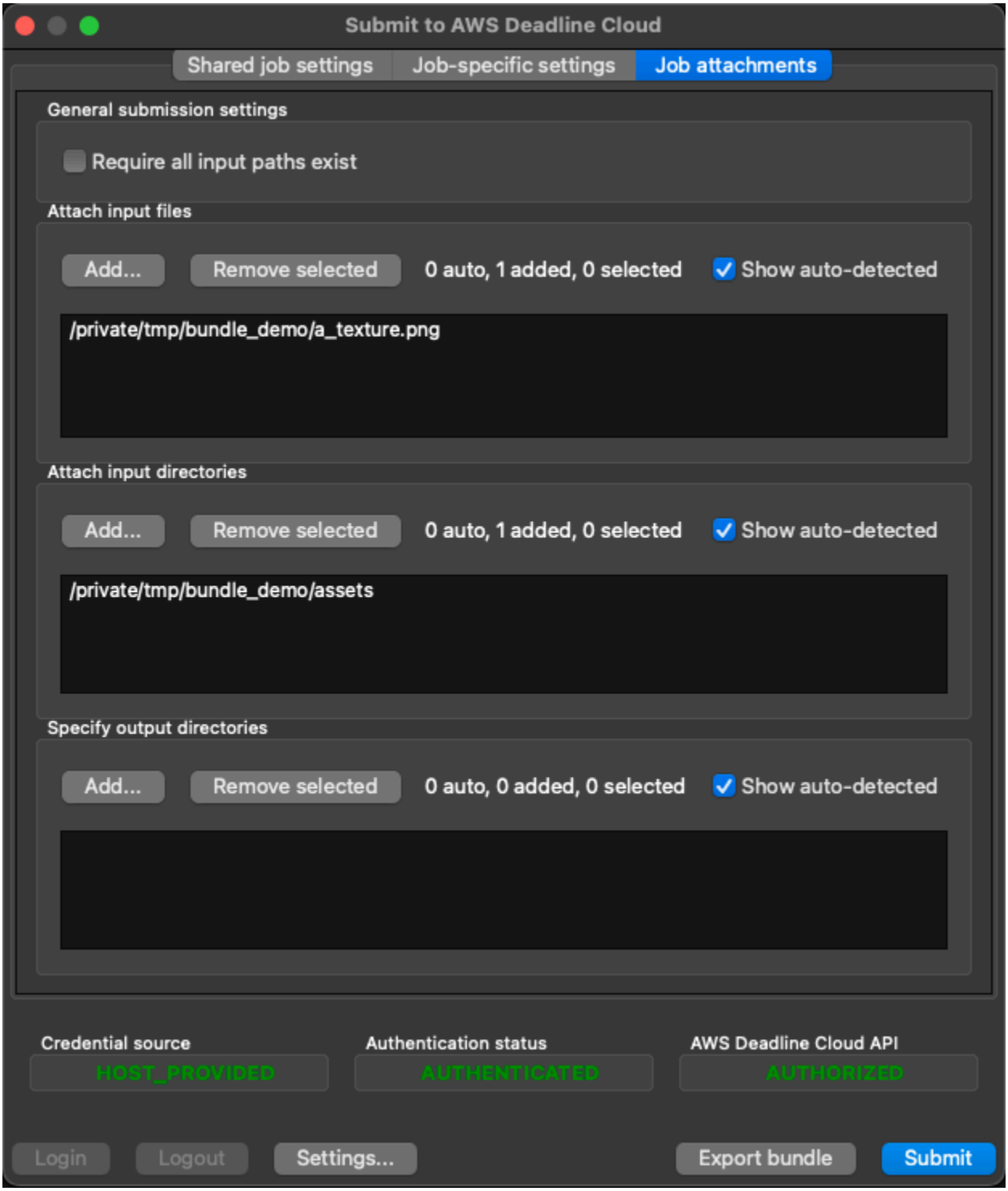
如需建立和使用儲存設定檔的詳細資訊，請參閱《 截止日期雲端使用者指南》中的在截止日期雲端中共用儲存。 AWS

Example- 截止日期雲端 GUI 建立的資產參考檔案

使用以下命令，使用 blender_render 範例提交任務。

```
deadline bundle gui-submit blender_render/
```

將一些額外的檔案新增至任務附件索引標籤上的任務：



提交任務後，您可以在任務歷史記錄目錄中查看任務套件中的 `asset_references.yaml` 檔案，以查看 YAML 檔案中的資產：

```
% cat ~/.deadline/job_history/\(default\)/2024-06/2024-06-20-01-JobBundle-Demo/
asset_references.yaml
```

```
assetReferences:
  inputs:
    filenames:
    - /private/tmp/bundle_demo/a_texture.png
    directories:
    - /private/tmp/bundle_demo/assets
  outputs:
    directories: []
  referencedPaths: []
```

在任務中使用檔案

您提交至 Deadline Cloud AWS 的許多任務都有輸入和輸出檔案。您的輸入檔案和輸出目錄可能位於共用檔案系統和本機磁碟機的組合上。任務需要在這些位置找到內容。Deadline Cloud 提供兩種功能：任務附件和儲存描述檔，可共同協助您的任務找到所需的檔案。

任務附件提供數種優點

- 使用 Amazon S3 在主機之間移動檔案
- 將檔案從工作站傳輸到工作者主機，反之亦然
- 適用於您啟用 功能的佇列中的任務
- 主要與服務受管機群搭配使用，但也與客戶受管機群相容。

使用儲存描述檔來映射工作站和工作者主機上共用檔案系統位置的配置。此映射可協助您的任務在工作站和工作者主機的位置不同時尋找共用檔案和目錄，例如使用 Windows型工作站和 Linux型工作者主機進行跨平台設定。任務附件也會使用儲存設定檔的檔案系統組態映射，來識別透過 Amazon S3 在主機之間來回傳輸所需的檔案。

如果您不是使用任務附件，而且不需要在工作站和工作者主機之間重新對應檔案和目錄位置，則不需要使用儲存設定檔建立檔案共用的模型。

主題

- 範例專案基礎設施
- 儲存設定檔和路徑映射

範例專案基礎設施

若要示範如何使用任務附件和儲存描述檔，請使用兩個不同的專案來設定測試環境。您可以使用截止日期雲端主控台來建立測試資源。

1. 如果您尚未建立測試陣列，請建立測試陣列。若要建立陣列，請遵循建立陣列中的程序。
2. 為兩個專案中的每個任務建立兩個佇列。若要建立佇列，請遵循建立佇列中的程序。

   a. 建立第一個名為 的佇列**Q1**。使用下列組態，針對所有其他項目使用預設值。

      - 針對任務附件，選擇建立新的 Amazon S3 儲存貯體。
      - 選取啟用與客戶受管機群的關聯。
      - 對於以使用者身分執行的 ，請在 POSIX 使用者和群組**jobuser**中輸入 。
      - 針對佇列服務角色，建立名為 的新角色 **AssetDemoFarm-Q1-Role**
      - 清除預設 conda 佇列環境核取方塊。

   b. 建立第二個名為 的佇列**Q2**。使用下列組態，針對所有其他項目使用預設值。

      - 針對任務附件，選擇建立新的 Amazon S3 儲存貯體。
      - 選取啟用與客戶受管機群的關聯。
      - 對於以使用者身分執行的 ，請在 POSIX 使用者和群組**jobuser**中輸入 。
      - 針對佇列服務角色，建立名為 的新角色 **AssetDemoFarm-Q2-Role**
      - 清除預設 conda 佇列環境核取方塊。

3. 建立單一客戶受管機群，從兩個佇列執行任務。若要建立機群，請遵循建立客戶受管機群中的程序。使用下列組態：

   - 對於名稱，請使用 **DemoFleet**。
   - 對於機群類型，選擇客戶受管
   - 針對機群服務角色，建立名為 AssetDemoFarm-Fleet-Role 的新角色。
   - 請勿將機群與任何佇列建立關聯。

測試環境假設主機之間使用網路檔案共用共用三個檔案系統。在此範例中，位置具有下列名稱：

- FSCommon - 包含兩個專案通用的輸入任務資產。
- FS1 - 包含專案 1 的輸入和輸出任務資產。
- FS2 - 包含專案 2 的輸入和輸出任務資產。

測試環境也會假設有三個工作站，如下所示：

- WSAll - 開發人員用於所有專案的 Linux型工作站。共用檔案系統位置為：
  - FSCommon: /shared/common
  - FS1: /shared/projects/project1
  - FS2: /shared/projects/project2
- WS1 - 用於專案 1 Windows的 型工作站。共用檔案系統位置為：
  - FSCommon: S:\
  - FS1: Z:\
  - FS2：無法使用
- WS1 - 用於專案 2 的 macOS型工作站。共用檔案系統位置為：
  - FSCommon: /Volumes/common
  - FS1：無法使用
  - FS2: /Volumes/projects/project2

最後，定義機群中工作者的共用檔案系統位置。以下範例會將此組態稱為 WorkerConfig。共用位置為：

- FSCommon: /mnt/common
- FS1: /mnt/projects/project1
- FS2: /mnt/projects/project2

您不需要設定任何符合此組態的共用檔案系統、工作站或工作者。示範不需要有共用的位置。

儲存設定檔和路徑映射

使用儲存描述檔來建立工作站和工作者主機上的檔案系統模型。每個儲存設定檔都會描述其中一個系統組態的作業系統和檔案系統配置。本主題說明如何使用儲存描述檔來建立主機的檔案系統組態模型，讓 Deadline Cloud 可以為您的任務產生路徑映射規則，以及如何從您的儲存描述檔產生這些路徑映射規則。

當您將任務提交至截止日期雲端時，您可以為任務提供選用的儲存設定檔 ID。此儲存設定檔說明提交工作站的檔案系統。它描述任務範本中檔案路徑使用的原始檔案系統組態。

您也可以將儲存描述檔與機群建立關聯。儲存設定檔說明機群中所有工作者主機的檔案系統組態。如果您有具有不同檔案系統組態的工作者，則必須將這些工作者指派給您陣列中的不同機群。

路徑映射規則描述路徑應如何從任務中指定的路徑重新映射到工作者主機上路徑的實際位置。Deadline Cloud 會將任務儲存描述檔中所述的檔案系統組態與執行任務之機群的儲存描述檔進行比較，以衍生這些路徑映射規則。

主題

具有儲存設定檔的模型共用檔案系統位置

儲存設定檔會將其中一個主機組態的檔案系統組態建模。範例專案基礎設施中有四種不同的主機組態。在此範例中，您會為每個 建立個別的儲存設定檔。您可以使用下列任一項來建立儲存設定檔：

- CreateStorageProfile API
- AWS::Deadline::StorageProfile CloudFormation 資源
- AWS 主控台

儲存設定檔是由檔案系統位置清單組成，每個檔案系統位置清單都會將與從主機提交或在主機上執行之任務相關的檔案系統位置和類型告知 Deadline Cloud。儲存描述檔應該只建立與任務相關的位置模型。例如，共用FSCommon位置位於 WS1 的工作站上S:\，因此對應的檔案系統位置為：

```
{
    "name": "FSCommon",
    "path": "S:\\",
    "type": "SHARED"
}
```

使用下列命令，在 WorkerConfigAWS CLI中使用 建立工作站組態 WS1、 WS2和 的儲存設定檔，WS3以及工作者組態AWS CloudShell：

```
# Change the value of FARM_ID to your farm's identifier
FARM_ID=farm-00112233445566778899aabbccddeeff
```

```
aws deadline create-storage-profile --farm-id $FARM_ID \
  --display-name WSAll \
  --os-family LINUX \
  --file-system-locations \
  '[
      {"name": "FSCommon", "type":"SHARED", "path":"/shared/common"},
      {"name": "FS1", "type":"SHARED", "path":"/shared/projects/project1"},
      {"name": "FS2", "type":"SHARED", "path":"/shared/projects/project2"}
  ]'

aws deadline create-storage-profile --farm-id $FARM_ID \
  --display-name WS1 \
  --os-family WINDOWS \
  --file-system-locations \
  '[
      {"name": "FSCommon", "type":"SHARED", "path":"S:\\"},
      {"name": "FS1", "type":"SHARED", "path":"Z:\\"}
   ]'

aws deadline create-storage-profile --farm-id $FARM_ID \
  --display-name WS2 \
  --os-family MACOS \
  --file-system-locations \
  '[
      {"name": "FSCommon", "type":"SHARED", "path":"/Volumes/common"},
      {"name": "FS2", "type":"SHARED", "path":"/Volumes/projects/project2"}
  ]'

aws deadline create-storage-profile --farm-id $FARM_ID \
  --display-name WorkerCfg \
  --os-family LINUX \
  --file-system-locations \
  '[
      {"name": "FSCommon", "type":"SHARED", "path":"/mnt/common"},
      {"name": "FS1", "type":"SHARED", "path":"/mnt/projects/project1"},
      {"name": "FS2", "type":"SHARED", "path":"/mnt/projects/project2"}
  ]'
```

> **Note**
>
> 您必須針對陣列中所有儲存設定檔的 name 屬性，使用相同的值來參考儲存設定檔中的檔案系統位置。Deadline Cloud 會比較名稱，以在產生路徑映射規則時，判斷來自不同儲存設定檔的檔案系統位置是否參照相同的位置。

設定機群的儲存設定檔

您可以設定機群，以包含儲存設定檔，在機群中的所有工作者上建立檔案系統位置的模型。機群中所有工作者的主機檔案系統組態必須符合其機群的儲存設定檔。具有不同檔案系統組態的工作者必須位於不同的機群中。

若要設定機群的組態以使用 `WorkerConfig` 中的 儲存設定檔AWS CLIAWS CloudShell：

```
# Change the value of FARM_ID to your farm's identifier
FARM_ID=farm-00112233445566778899aabbccddeeff
# Change the value of FLEET_ID to your fleet's identifier
FLEET_ID=fleet-00112233445566778899aabbccddeeff
# Change the value of WORKER_CFG_ID to your storage profile named WorkerConfig
WORKER_CFG_ID=sp-00112233445566778899aabbccddeeff

FLEET_WORKER_MODE=$( \
  aws deadline get-fleet --farm-id $FARM_ID --fleet-id $FLEET_ID \
   --query '.configuration.customerManaged.mode' \
)
FLEET_WORKER_CAPABILITIES=$( \
  aws deadline get-fleet --farm-id $FARM_ID --fleet-id $FLEET_ID \
   --query '.configuration.customerManaged.workerCapabilities' \
)

aws deadline update-fleet --farm-id $FARM_ID --fleet-id $FLEET_ID \
  --configuration \
  "{
    \"customerManaged\": {
      \"storageProfileId\": \"$WORKER_CFG_ID\",
      \"mode\": $FLEET_WORKER_MODE,
      \"workerCapabilities\": $FLEET_WORKER_CAPABILITIES
    }
  }"
```

設定佇列的儲存設定檔

佇列的組態包含提交至佇列之任務需要存取之共用檔案系統位置的區分大小寫名稱清單。例如，提交至佇列的任務Q1需要檔案系統位置FSCommon和 FS1。提交至佇列的任務Q2需要檔案系統位置FSCommon和 FS2。

若要將佇列的組態設定為需要這些檔案系統位置，請使用下列指令碼：

```
# Change the value of FARM_ID to your farm's identifier
FARM_ID=farm-00112233445566778899aabbccddeeff
# Change the value of QUEUE1_ID to queue Q1's identifier
QUEUE1_ID=queue-00112233445566778899aabbccddeeff
# Change the value of QUEUE2_ID to queue Q2's identifier
QUEUE2_ID=queue-00112233445566778899aabbccddeeff

aws deadline update-queue --farm-id $FARM_ID --queue-id $QUEUE1_ID \
  --required-file-system-location-names-to-add FSComm FS1

aws deadline update-queue --farm-id $FARM_ID --queue-id $QUEUE2_ID \
  --required-file-system-location-names-to-add FSComm FS2
```

佇列的組態也包含允許儲存設定檔的清單，這些設定檔適用於提交到 的任務，以及與該佇列相關聯的機群。只有定義佇列所有必要檔案系統位置的檔案系統位置的儲存設定檔，才允許在佇列的允許儲存設定檔清單中。

如果您提交的儲存描述檔不在佇列允許的儲存描述檔清單中，則任務會失敗。您可以隨時將沒有儲存設定檔的任務提交至佇列。標記為 WSAll和 的工作站組態WS1都有佇列 所需的檔案系統位置 (FSCommon 和 FS1)Q1。他們需要被允許將任務提交到佇列。同樣地，工作站會設定 WSAll並WS2符合佇列 的要求Q2。必須允許他們提交任務至該佇列。更新兩個佇列組態，以允許使用下列指令碼搭配這些儲存設定檔提交任務：

```
# Change the value of WSALL_ID to the identifier of the WSAll storage profile
WSALL_ID=sp-00112233445566778899aabbccddeeff
# Change the value of WS1 to the identifier of the WS1 storage profile
WS1_ID=sp-00112233445566778899aabbccddeeff
# Change the value of WS2 to the identifier of the WS2 storage profile
WS2_ID=sp-00112233445566778899aabbccddeeff

aws deadline update-queue --farm-id $FARM_ID --queue-id $QUEUE1_ID \
  --allowed-storage-profile-ids-to-add $WSALL_ID $WS1_ID
```

```
aws deadline update-queue --farm-id $FARM_ID --queue-id $QUEUE2_ID \
  --allowed-storage-profile-ids-to-add $WSALL_ID $WS2_ID
```

如果您將WS2儲存描述檔新增至佇列允許的儲存描述檔清單Q1，則會失敗：

```
$ aws deadline update-queue --farm-id $FARM_ID --queue-id $QUEUE1_ID \
  --allowed-storage-profile-ids-to-add $WS2_ID

An error occurred (ValidationException) when calling the UpdateQueue operation: Storage
 profile id: sp-00112233445566778899aabbccddeeff does not have required file system
 location: FS1
```

這是因為WS2儲存描述檔不包含FS1佇列Q1所需檔案系統位置的定義。

將已設定的機群與不在佇列允許儲存設定檔清單中的儲存設定檔建立關聯也會失敗。例如：

```
$ aws deadline create-queue-fleet-association --farm-id $FARM_ID \
   --fleet-id $FLEET_ID \
   --queue-id $QUEUE1_ID

An error occurred (ValidationException) when calling the CreateQueueFleetAssociation
 operation: Mismatch between storage profile ids.
```

若要修正錯誤，請將名為 WorkerConfig 的儲存設定檔新增至佇列 Q1和佇列 的允許儲存設定檔清單Q2。然後，將機群與這些佇列建立關聯，以便機群中的工作者可以從兩個佇列執行任務。

```
# Change the value of FLEET_ID to your fleet's identifier
FLEET_ID=fleet-00112233445566778899aabbccddeeff
# Change the value of WORKER_CFG_ID to your storage profile named WorkerCfg
WORKER_CFG_ID=sp-00112233445566778899aabbccddeeff

aws deadline update-queue --farm-id $FARM_ID --queue-id $QUEUE1_ID \
  --allowed-storage-profile-ids-to-add $WORKER_CFG_ID

aws deadline update-queue --farm-id $FARM_ID --queue-id $QUEUE2_ID \
  --allowed-storage-profile-ids-to-add $WORKER_CFG_ID

aws deadline create-queue-fleet-association --farm-id $FARM_ID \
  --fleet-id $FLEET_ID \
  --queue-id $QUEUE1_ID
```

```
aws deadline create-queue-fleet-association --farm-id $FARM_ID \
  --fleet-id $FLEET_ID \
  --queue-id $QUEUE2_ID
```

從儲存體設定檔衍生路徑映射規則

路徑映射規則描述路徑應如何從任務重新映射到工作者主機上路徑的實際位置。當任務在工作者上執行時，任務的儲存設定檔會與工作者機群的儲存設定檔進行比較，以衍生任務的路徑映射規則。

截止日期 雲端會為佇列組態中的每個必要檔案系統位置建立映射規則。例如，使用WSAll儲存描述檔提交到佇列的任務Q1具有路徑映射規則：

- FSComm: /shared/common -> /mnt/common
- FS1: /shared/projects/project1 -> /mnt/projects/project1

Deadline Cloud 會建立 FSComm和 FS1 檔案系統位置的規則，但即使 WSAll和 WorkerConfig儲存設定檔都定義 ，也無法建立FS2檔案系統位置FS2。這是因為佇列 Q1的必要檔案系統位置清單為["FSComm", "FS1"]。

您可以提交列印 Open Job Description 路徑映射規則檔案的任務，然後在任務完成後讀取工作階段日誌，以確認使用特定儲存設定檔提交的任務可用的路徑映射規則：

```
# Change the value of FARM_ID to your farm's identifier
FARM_ID=farm-00112233445566778899aabbccddeeff
# Change the value of QUEUE1_ID to queue Q1's identifier
QUEUE1_ID=queue-00112233445566778899aabbccddeeff
# Change the value of WSALL_ID to the identifier of the WSALL storage profile
WSALL_ID=sp-00112233445566778899aabbccddeeff

aws deadline create-job --farm-id $FARM_ID --queue-id $QUEUE1_ID \
  --priority 50 \
  --storage-profile-id $WSALL_ID \
  --template-type JSON --template \
  '{
    "specificationVersion": "jobtemplate-2023-09",
    "name": "DemoPathMapping",
    "steps": [
      {
        "name": "ShowPathMappingRules",
        "script": {
          "actions": {
```

```
            "onRun": {
              "command": "/bin/cat",
              "args": [ "{{Session.PathMappingRulesFile}}" ]
            }
          }
        }
      }
    ]
  }'
```

如果您使用截止日期雲端 CLI 提交任務，其settings.storage_profile_id組態設定會設定使用 CLI 提交的任務將擁有的儲存設定檔。若要使用WSAll儲存描述檔提交任務，請設定：

```
deadline config set settings.storage_profile_id $WSALL_ID
```

若要像在範例基礎設施中執行一樣執行客戶受管工作者，請遵循在截止日期雲端使用者指南中執行工作者代理程式中的程序來執行工作者 AWS CloudShell。如果您之前遵循這些指示，請先刪除 ~/demoenv-logs和 ~/demoenv-persist目錄。此外，設定方向參考的 DEV_FARM_ID和 DEV_CMF_ID環境變數值，如下所示，然後再執行此操作：

```
DEV_FARM_ID=$FARM_ID
DEV_CMF_ID=$FLEET_ID
```

任務執行後，您可以在任務的日誌檔案中看到路徑映射規則：

```
cat demoenv-logs/${QUEUE1_ID}/*.log
...
JJSON log results (see below)
...
```

日誌包含 FS1和 FSComm 檔案系統的映射。重新格式化可讀性，日誌項目如下所示：

```
{
    "version": "pathmapping-1.0",
    "path_mapping_rules": [
        {
            "source_path_format": "POSIX",
            "source_path": "/shared/projects/project1",
            "destination_path": "/mnt/projects/project1"
        },
```

```
        {
            "source_path_format": "POSIX",
            "source_path": "/shared/common",
            "destination_path": "/mnt/common"
        }
    ]
```

您可以提交具有不同儲存設定檔的任務，以查看路徑映射規則如何變更。

使用任務附件來共用檔案

使用任務附件讓不在共用目錄中的檔案可供您的任務使用，如果輸出檔案未寫入共用目錄，則擷取輸出檔案。任務連接使用 Amazon S3 在主機之間傳輸檔案。檔案存放在 S3 儲存貯體中，如果檔案的內容未變更，則不需要上傳檔案。

在服務受管機群上執行任務時，您必須使用任務附件，因為主機不會共用檔案系統位置。當任務的輸入或輸出檔案存放在共用網路檔案系統上時，例如當您的任務套件包含 shell 或 Python 指令碼時，任務附件也適用於客戶受管機群。

當您使用截止日期雲端 CLI 或截止日期雲端提交者提交任務套件時，任務連接會使用任務的儲存描述檔和佇列的必要檔案系統位置來識別不在工作者主機上的輸入檔案，並且應該上傳到 Amazon S3 作為任務提交的一部分。這些儲存設定檔也有助於 Deadline Cloud 識別工作者主機位置中的輸出檔案，這些檔案必須上傳至 Amazon S3，以便可供您的工作站使用。

任務附件範例使用來自 和 的陣列、機群、佇列範例專案基礎設施和儲存設定檔組態儲存設定檔和路徑映射。您應該在此之前完成這些區段。

在下列範例中，您會使用範例任務套件做為起點，然後將其修改以探索任務連接的功能。任務套件是任務使用任務附件的最佳方式。它們將目錄中的開啟任務描述任務範本與使用任務套件列出任務所需檔案和目錄的其他檔案結合在一起。如需任務套件的詳細資訊，請參閱 截止日期雲端的開啟任務描述 (OpenJD) 範本。

使用任務提交檔案

使用截止日期雲端，您可以讓任務工作流程存取工作者主機上共用檔案系統位置中無法使用的輸入檔案。任務附件允許轉譯任務僅存取位於本機工作站磁碟機或服務受管機群環境的檔案。提交任務套件時，您可以包含任務所需的輸入檔案和目錄清單。截止日期 雲端會識別這些非共用檔案，從本機電腦上傳到 Amazon S3，並將它們下載到工作者主機。它簡化了將輸入資產傳輸到轉譯節點的程序，確保可存取所有必要檔案以進行分散式任務執行。

您可以直接在任務套件中指定任務的檔案、在您使用環境變數或指令碼提供的任務範本中使用參數，以及使用任務`assets_references`的檔案。您可以使用其中一種方法或這三種方法的組合。您可以為任務的套件指定儲存描述檔，以便只上傳已在本機工作站上變更的檔案。

本節使用來自 GitHub 的範例任務套件，示範 Deadline Cloud 如何識別任務中要上傳的檔案、這些檔案如何在 Amazon S3 中組織，以及如何提供給處理任務的工作者主機。

主題

- Deadline Cloud 如何將檔案上傳至 Amazon S3
- Deadline Cloud 如何選擇要上傳的檔案
- 任務如何尋找任務連接輸入檔案

Deadline Cloud 如何將檔案上傳至 Amazon S3

此範例顯示 Deadline Cloud 如何將檔案從工作站或工作者主機上傳至 Amazon S3，以便共用這些檔案。它使用來自 GitHub 和截止日期雲端 CLI 的範例任務套件來提交任務。

首先將截止期限雲端範例 GitHub 儲存庫複製到您的AWS CloudShell環境，然後將`job_attachments_devguide`任務套件複製到您的主目錄：

```
git clone https://github.com/aws-deadline/deadline-cloud-samples.git
cp -r deadline-cloud-samples/job_bundles/job_attachments_devguide ~/
```

安裝截止日期雲端 CLI 以提交任務套件：

```
pip install deadline --upgrade
```

`job_attachments_devguide` 任務套件具有單一步驟，其中包含執行 bash shell 指令碼的任務，其檔案系統位置會以任務參數傳遞。任務參數的定義為：

```
...
- name: ScriptFile
  type: PATH
  default: script.sh
  dataFlow: IN
  objectType: FILE
...
```

dataFlow 屬性IN的值會告知任務附件， ScriptFile 參數的值是任務的輸入。default 屬性的值是任務套件目錄的相對位置，但也可以是絕對路徑。此參數定義會將任務套件目錄中script.sh的檔案宣告為執行任務所需的輸入檔案。

接下來，請確定截止日期雲端 CLI 未設定儲存設定檔，然後將任務提交至佇列 Q1：

```
# Change the value of FARM_ID to your farm's identifier
FARM_ID=farm-00112233445566778899aabbccddeeff
# Change the value of QUEUE1_ID to queue Q1's identifier
QUEUE1_ID=queue-00112233445566778899aabbccddeeff

deadline config set settings.storage_profile_id ''

deadline bundle submit --farm-id $FARM_ID --queue-id $QUEUE1_ID
 job_attachments_devguide/
```

在此命令執行後，來自截止日期雲端 CLI 的輸出如下所示：

```
Submitting to Queue: Q1
...
Hashing Attachments  [####################################]  100%
Hashing Summary:
    Processed 1 file totaling 39.0 B.
    Skipped re-processing 0 files totaling 0.0 B.
    Total processing time of 0.0327 seconds at 1.19 KB/s.

Uploading Attachments  [####################################]  100%
Upload Summary:
    Processed 1 file totaling 39.0 B.
    Skipped re-processing 0 files totaling 0.0 B.
    Total processing time of 0.25639 seconds at 152.0 B/s.

Waiting for Job to be created...
Submitted job bundle:
   job_attachments_devguide/
Job creation completed successfully
job-74148c13342e4514b63c7a7518657005
```

當您提交任務時，截止日期雲端會先雜湊script.sh檔案，然後將其上傳至 Amazon S3。

Deadline Cloud 會將 S3 儲存貯體視為內容可定址儲存體。檔案會上傳至 S3 物件。物件名稱衍生自檔案內容的雜湊。如果兩個檔案具有相同的內容，則無論檔案位於何處或名稱為何，它們都具有相同的雜湊值。此內容可定址儲存可讓 Deadline Cloud 在檔案已可用時避免上傳檔案。

您可以使用 AWS CLI 來查看上傳至 Amazon S3 的物件：

```
# The name of queue `Q1`'s job attachments S3 bucket
Q1_S3_BUCKET=$(
  aws deadline get-queue --farm-id $FARM_ID --queue-id $QUEUE1_ID \
    --query 'jobAttachmentSettings.s3BucketName' | tr -d '"'
)

aws s3 ls s3://$Q1_S3_BUCKET --recursive
```

兩個物件已上傳至 S3：

- `DeadlineCloud/Data/87cb19095dd5d78fcaf56384ef0e6241.xxh128` – 的內容`script.sh`。物件金鑰`87cb19095dd5d78fcaf56384ef0e6241`中的值是檔案內容的雜湊，副檔`xxh128`名表示雜湊值計算為 128 位元 xxhash。
- `DeadlineCloud/Manifests/<farm-id>/<queue-id>/Inputs/<guid>/a1d221c7fd97b08175b3872a37428e8c_input` – 任務提交的資訊清單物件。值 `<farm-id>`、 `<queue-id>`和 `<guid>`是您的陣列識別符、佇列識別符和隨機十六進位值。`a1d221c7fd97b08175b3872a37428e8c` 此範例中的值是從字串 計算的雜湊值`/home/cloudshell-user/job_attachments_devguide`，即 `script.sh` 所在的目錄。

資訊清單物件包含作為任務提交的一部分上傳至 S3 之特定根路徑上的輸入檔案資訊。下載此資訊清單檔案 (`aws s3 cp s3://$Q1_S3_BUCKET/<objectname>`)。其內容類似：

```
{
    "hashAlg": "xxh128",
    "manifestVersion": "2023-03-03",
    "paths": [
        {
            "hash": "87cb19095dd5d78fcaf56384ef0e6241",
            "mtime": 1721147454416085,
            "path": "script.sh",
            "size": 39
        }
    ],
    "totalSize": 39
```

```
}
```

這表示檔案script.sh已上傳，且該檔案內容的雜湊為 87cb19095dd5d78fcaf56384ef0e6241。此雜湊值符合物件名稱 中的值DeadlineCloud/Data/87cb19095dd5d78fcaf56384ef0e6241.xxh128。Deadline Cloud 會使用它來知道要為此檔案的內容下載哪個物件。

此檔案的完整結構描述可在 GitHub 中取得。

當您使用 CreateJob 操作時，您可以設定資訊清單物件的位置。您可以使用 GetJob 操作來查看位置：

```
{
    "attachments": {
        "file system": "COPIED",
        "manifests": [
            {
                "inputManifestHash": "5b0db3d311805ea8de7787b64cbbe8b3",
                "inputManifestPath": "<farm-id>/<queue-id>/Inputs/<guid>/
a1d221c7fd97b08175b3872a37428e8c_input",
                "rootPath": "/home/cloudshell-user/job_attachments_devguide",
                "rootPathFormat": "posix"
            }
        ]
    },
    ...
}
```

Deadline Cloud 如何選擇要上傳的檔案

任務附件視為上傳到 Amazon S3 做為任務輸入的檔案和目錄如下：

- 任務套件任務範本中定義之所有 PATH類型任務參數的值，dataFlow其值為 IN或 INOUT。
- 列為任務套件資產參考檔案中輸入的檔案和目錄。

如果您提交的任務沒有儲存描述檔，則會上傳所有考慮上傳的檔案。如果您使用儲存描述檔提交任務，如果檔案位於儲存描述檔的 SHARED類型檔案系統位置，而這些位置也是佇列所需的檔案系統位置，則檔案不會上傳到 Amazon S3。這些位置預期可在執行任務的工作者主機上使用，因此不需要將其上傳至 S3。

在此範例中，您會在 AWS CloudShell 環境中的 WSAll中建立SHARED檔案系統位置，然後將檔案新增至這些檔案系統位置。使用下列命令：

```
# Change the value of WSALL_ID to the identifier of the WSAll storage profile
WSALL_ID=sp-00112233445566778899aabbccddeeff

sudo mkdir -p /shared/common /shared/projects/project1 /shared/projects/project2
sudo chown -R cloudshell-user:cloudshell-user /shared

for d in /shared/common /shared/projects/project1 /shared/projects/project2; do
  echo "File contents for $d" > ${d}/file.txt
done
```

接著，將資產參考檔案新增至任務套件，其中包含您建立做為任務輸入的所有檔案。使用下列命令：

```
cat > ${HOME}/job_attachments_devguide/asset_references.yaml << EOF
assetReferences:
  inputs:
    filenames:
    - /shared/common/file.txt
    directories:
    - /shared/projects/project1
    - /shared/projects/project2
EOF
```

接著，設定截止日期雲端 CLI 以使用WSAll儲存設定檔提交任務，然後提交任務套件：

```

# Change the value of FARM_ID to your farm's identifier
FARM_ID=farm-00112233445566778899aabbccddeeff
# Change the value of QUEUE1_ID to queue Q1's identifier
QUEUE1_ID=queue-00112233445566778899aabbccddeeff
# Change the value of WSALL_ID to the identifier of the WSAll storage profile
WSALL_ID=sp-00112233445566778899aabbccddeeff

deadline config set settings.storage_profile_id $WSALL_ID

deadline bundle submit --farm-id $FARM_ID --queue-id $QUEUE1_ID
 job_attachments_devguide/
```

當您提交任務時，截止日期 Cloud 會將兩個檔案上傳至 Amazon S3。您可以從 S3 下載任務的資訊清單物件，以查看上傳的檔案：

```
for manifest in $( \
```

```
  aws deadline get-job --farm-id $FARM_ID --queue-id $QUEUE1_ID --job-id $JOB_ID \
    --query 'attachments.manifests[].inputManifestPath' \
    | jq -r '.[]'
); do
  echo "Manifest object: $manifest"
  aws s3 cp --quiet s3://$Q1_S3_BUCKET/DeadlineCloud/Manifests/$manifest /dev/stdout |
 jq .
done
```

在此範例中，有一個資訊清單檔案，其中包含下列內容：

```
{
    "hashAlg": "xxh128",
    "manifestVersion": "2023-03-03",
    "paths": [
        {
            "hash": "87cb19095dd5d78fcaf56384ef0e6241",
            "mtime": 1721147454416085,
            "path": "home/cloudshell-user/job_attachments_devguide/script.sh",
            "size": 39
        },
        {
            "hash": "af5a605a3a4e86ce7be7ac5237b51b79",
            "mtime": 1721163773582362,
            "path": "shared/projects/project2/file.txt",
            "size": 44
        }
    ],
    "totalSize": 83
}
```

使用資訊清單的 GetJob 操作來查看 `rootPath`是 "/"。

```
aws deadline get-job --farm-id $FARM_ID --queue-id $QUEUE1_ID --job-id $JOB_ID --query
 'attachments.manifests[*]'
```

一組輸入檔案的根路徑永遠是這些檔案最長的常見子路徑。如果您的任務是從 提交，Windows並且因為位於不同的磁碟機上而沒有通用子路徑的輸入檔案，您會在每個磁碟機上看到個別的根路徑。資訊清單中的路徑一律與資訊清單的根路徑相對，因此上傳的輸入檔案為：

- `/home/cloudshell-user/job_attachments_devguide/script.sh` – 任務套件中的指令碼檔案。

- /shared/projects/project2/file.txt – WSAll儲存設定檔中SHARED檔案系統位置中的檔案，該檔案不在佇列 所需的檔案系統位置清單中Q1。

檔案系統位置 FSCommon(/shared/common/file.txt) 和 FS1(/shared/projects/project1/file.txt) 中的檔案不在清單中。這是因為這些檔案系統位置位於WSAll儲存設定檔SHARED中，而且它們都位於佇列 中所需檔案系統位置的清單中Q1。

您可以使用 GetStorageProfileForQueue SHARED 操作，查看使用特定儲存設定檔提交之任務所考慮的檔案系統位置。 GetStorageProfileForQueue 若要查詢佇列WSAll的儲存設定檔，Q1請使用下列命令：

```
aws deadline get-storage-profile --farm-id $FARM_ID --storage-profile-id $WSALL_ID

aws deadline get-storage-profile-for-queue --farm-id $FARM_ID --queue-id $QUEUE1_ID --
storage-profile-id $WSALL_ID
```

任務如何尋找任務連接輸入檔案

若要讓任務使用 Deadline Cloud 使用任務附件上傳至 Amazon S3 的檔案，您的任務需要透過工作者主機上的檔案系統提供這些檔案。當任務的工作階段在工作者主機上執行時，Deadline Cloud 會將任務的輸入檔案下載到工作者主機本機磁碟機的暫存目錄中，並將每個任務根路徑的路徑映射規則新增至本機磁碟機上的檔案系統位置。

在此範例中，在 AWS CloudShell 索引標籤中啟動截止日期雲端工作者代理程式。讓任何先前提交的任務完成執行，然後從日誌目錄中刪除任務日誌：

```
rm -rf ~/devdemo-logs/queue-*
```

下列指令碼會修改任務套件，以顯示工作階段臨時工作目錄中的所有檔案，以及路徑映射規則檔案的內容，然後提交具有修改後套件的任務：

```
# Change the value of FARM_ID to your farm's identifier
FARM_ID=farm-00112233445566778899aabbccddeeff
# Change the value of QUEUE1_ID to queue Q1's identifier
QUEUE1_ID=queue-00112233445566778899aabbccddeeff
# Change the value of WSALL_ID to the identifier of the WSAll storage profile
WSALL_ID=sp-00112233445566778899aabbccddeeff

deadline config set settings.storage_profile_id $WSALL_ID
```

```
cat > ~/job_attachments_devguide/script.sh << EOF
#!/bin/bash

echo "Session working directory is: \$(pwd)"
echo
echo "Contents:"
find . -type f
echo
echo "Path mapping rules file: \$1"
jq . \$1
EOF

cat > ~/job_attachments_devguide/template.yaml << EOF
specificationVersion: jobtemplate-2023-09
name: "Job Attachments Explorer"
parameterDefinitions:
- name: ScriptFile
  type: PATH
  default: script.sh
  dataFlow: IN
  objectType: FILE
steps:
- name: Step
  script:
    actions:
      onRun:
        command: /bin/bash
        args:
        - "{{Param.ScriptFile}}"
        - "{{Session.PathMappingRulesFile}}"
EOF

deadline bundle submit --farm-id $FARM_ID --queue-id $QUEUE1_ID
 job_attachments_devguide/
```

在 AWS CloudShell 環境中的工作者執行任務之後，您可以查看任務執行的日誌：

```
cat demoenv-logs/queue-*/session*.log
```

日誌顯示工作階段中發生的第一件事是任務的兩個輸入檔案下載到工作者：

```
2024-07-17 01:26:37,824 INFO ================================================
```

```
2024-07-17 01:26:37,825 INFO --------- Job Attachments Download for Job
2024-07-17 01:26:37,825 INFO ===============================================
2024-07-17 01:26:37,825 INFO Syncing inputs using Job Attachments
2024-07-17 01:26:38,116 INFO Downloaded 142.0 B / 186.0 B of 2 files (Transfer rate:
 0.0 B/s)
2024-07-17 01:26:38,174 INFO Downloaded 186.0 B / 186.0 B of 2 files (Transfer rate:
 733.0 B/s)
2024-07-17 01:26:38,176 INFO Summary Statistics for file downloads:
Processed 2 files totaling 186.0 B.
Skipped re-processing 0 files totaling 0.0 B.
Total processing time of 0.09752 seconds at 1.91 KB/s.
```

接下來是任務script.sh執行的輸出：

- 提交任務時上傳的輸入檔案位於目錄下，其名稱開頭為工作階段暫時目錄中的「assetroot」。
- 輸入檔案的路徑已相對於「assetroot」目錄重新定位，而不是相對於任務輸入資訊清單的根路徑 ("/")。
- 路徑映射規則檔案包含一個額外的規則，可對應"/"至「assetroot」目錄的絕對路徑。

例如：

```
2024-07-17 01:26:38,264 INFO Output:
2024-07-17 01:26:38,267 INFO Session working directory is: /sessions/session-5b33f
2024-07-17 01:26:38,267 INFO
2024-07-17 01:26:38,267 INFO Contents:
2024-07-17 01:26:38,269 INFO ./tmp_xdhbsdo.sh
2024-07-17 01:26:38,269 INFO ./tmpdi00052b.json
2024-07-17 01:26:38,269 INFO ./assetroot-assetroot-3751a/shared/projects/project2/
file.txt
2024-07-17 01:26:38,269 INFO ./assetroot-assetroot-3751a/home/cloudshell-user/
job_attachments_devguide/script.sh
2024-07-17 01:26:38,269 INFO
2024-07-17 01:26:38,270 INFO Path mapping rules file: /sessions/session-5b33f/
tmpdi00052b.json
2024-07-17 01:26:38,282 INFO {
2024-07-17 01:26:38,282 INFO   "version": "pathmapping-1.0",
2024-07-17 01:26:38,282 INFO   "path_mapping_rules": [
2024-07-17 01:26:38,282 INFO     {
2024-07-17 01:26:38,282 INFO       "source_path_format": "POSIX",
2024-07-17 01:26:38,282 INFO       "source_path": "/shared/projects/project1",
2024-07-17 01:26:38,283 INFO       "destination_path": "/mnt/projects/project1"
```

```
2024-07-17 01:26:38,283 INFO     },
2024-07-17 01:26:38,283 INFO     {
2024-07-17 01:26:38,283 INFO       "source_path_format": "POSIX",
2024-07-17 01:26:38,283 INFO       "source_path": "/shared/common",
2024-07-17 01:26:38,283 INFO       "destination_path": "/mnt/common"
2024-07-17 01:26:38,283 INFO     },
2024-07-17 01:26:38,283 INFO     {
2024-07-17 01:26:38,283 INFO       "source_path_format": "POSIX",
2024-07-17 01:26:38,283 INFO       "source_path": "/",
2024-07-17 01:26:38,283 INFO       "destination_path": "/sessions/session-5b33f/
assetroot-assetroot-3751a"
2024-07-17 01:26:38,283 INFO     }
2024-07-17 01:26:38,283 INFO   ]
2024-07-17 01:26:38,283 INFO }
```

> **Note**
>
> 如果您提交的任務具有多個具有不同根路徑的資訊清單，則每個根路徑都有不同的「assetroot」命名目錄。

如果您需要參考其中一個輸入檔案、目錄或檔案系統位置的重新定位檔案系統位置，您可以處理任務中的路徑映射規則檔案並自行執行重新映射，或將PATH類型任務參數新增至任務套件中的任務範本，並將重新映射所需的值傳遞為該參數的值。例如，下列範例會將任務套件修改為具有其中一個任務參數，然後以檔案系統位置/shared/projects/project2做為其值來提交任務：

```
cat > ~/job_attachments_devguide/template.yaml << EOF
specificationVersion: jobtemplate-2023-09
name: "Job Attachments Explorer"
parameterDefinitions:
- name: LocationToRemap
  type: PATH
steps:
- name: Step
  script:
    actions:
      onRun:
        command: /bin/echo
        args:
        - "The location of {{RawParam.LocationToRemap}} in the session is
 {{Param.LocationToRemap}}"
EOF
```

```
deadline bundle submit --farm-id $FARM_ID --queue-id $QUEUE1_ID
 job_attachments_devguide/ \
  -p LocationToRemap=/shared/projects/project2
```

此任務執行的日誌檔案包含其輸出：

```
2024-07-17 01:40:35,283 INFO Output:
2024-07-17 01:40:35,284 INFO The location of /shared/projects/project2 in the session
 is /sessions/session-5b33f/assetroot-assetroot-3751a
```

從任務取得輸出檔案

此範例顯示 Deadline Cloud 如何識別任務產生的輸出檔案、決定是否將這些檔案上傳至 Amazon S3，以及如何在工作站上取得這些輸出檔案。

此範例使用 `job_attachments_devguide_output` 任務套件，而非 `job_attachments_devguide`任務套件。首先，從您複製的截止日期雲端範例 GitHub 儲存庫，在您的 AWS CloudShell 環境中複製套件：

```
cp -r deadline-cloud-samples/job_bundles/job_attachments_devguide_output ~/
```

此任務套件和`job_attachments_devguide`任務套件之間的重要差異在於在任務範本中新增新的任務參數：

```
...
parameterDefinitions:
...
- name: OutputDir
  type: PATH
  objectType: DIRECTORY
  dataFlow: OUT
  default: ./output_dir
  description: This directory contains the output for all steps.
...
```

參數的 `dataFlow` 屬性具有值 `OUT`。Deadline Cloud 使用 值為 `OUT`或 `dataFlow`的任務參數值`INOUT`做為任務的輸出。如果傳遞為這類任務參數值的檔案系統位置重新映射至執行任務的工作者上的本機檔案系統位置，則 Deadline Cloud 會在該位置尋找新檔案，並將這些檔案上傳到 Amazon S3 做為任務輸出。

若要查看其運作方式，請先在 AWS CloudShell 索引標籤中啟動截止日期雲端工作者代理程式。讓任何先前提交的任務完成執行。然後從日誌目錄中刪除任務日誌：

```
rm -rf ~/devdemo-logs/queue-*
```

接著，使用此任務套件提交任務。在 CloudShell 中執行的工作者執行後，請查看日誌：

```
# Change the value of FARM_ID to your farm's identifier
FARM_ID=farm-00112233445566778899aabbccddeeff
# Change the value of QUEUE1_ID to queue Q1's identifier
QUEUE1_ID=queue-00112233445566778899aabbccddeeff
# Change the value of WSALL_ID to the identifier of the WSAll storage profile
WSALL_ID=sp-00112233445566778899aabbccddeeff

deadline config set settings.storage_profile_id $WSALL_ID

deadline bundle submit --farm-id $FARM_ID --queue-id $QUEUE1_ID ./
job_attachments_devguide_output
```

日誌顯示檔案已偵測到為輸出並上傳至 Amazon S3：

```
2024-07-17 02:13:10,873 INFO ------------------------------------------------
2024-07-17 02:13:10,873 INFO Uploading output files to Job Attachments
2024-07-17 02:13:10,873 INFO ------------------------------------------------
2024-07-17 02:13:10,873 INFO Started syncing outputs using Job Attachments
2024-07-17 02:13:10,955 INFO Found 1 file totaling 117.0 B in output directory: /
sessions/session-7efa/assetroot-assetroot-3751a/output_dir
2024-07-17 02:13:10,956 INFO Uploading output manifest to
 DeadlineCloud/Manifests/farm-0011/queue-2233/job-4455/step-6677/
task-6677-0/2024-07-17T02:13:10.835545Z_sessionaction-8899-1/
c6808439dfc59f86763aff5b07b9a76c_output
2024-07-17 02:13:10,988 INFO Uploading 1 output file to S3: s3BucketName/DeadlineCloud/
Data
2024-07-17 02:13:11,011 INFO Uploaded 117.0 B / 117.0 B of 1 file (Transfer rate: 0.0
 B/s)
2024-07-17 02:13:11,011 INFO Summary Statistics for file uploads:
Processed 1 file totaling 117.0 B.
Skipped re-processing 0 files totaling 0.0 B.
Total processing time of 0.02281 seconds at 5.13 KB/s.
```

日誌也顯示 Deadline Cloud 在 Amazon S3 儲存貯體中建立新的資訊清單物件，該儲存貯體設定為由佇列 上的任務附件使用Q1。資訊清單物件的名稱衍生自產生輸出之任務的陣列、佇列、任務、步驟、

任務、時間戳記和sessionaction識別符。下載此資訊清單檔案，以查看截止日期雲端放置此任務輸出檔案的位置：

```
# The name of queue `Q1`'s job attachments S3 bucket
Q1_S3_BUCKET=$(
  aws deadline get-queue --farm-id $FARM_ID --queue-id $QUEUE1_ID \
    --query 'jobAttachmentSettings.s3BucketName' | tr -d '"'
)

# Fill this in with the object name from your log
OBJECT_KEY="DeadlineCloud/Manifests/..."

aws s3 cp --quiet s3://$Q1_S3_BUCKET/$OBJECT_KEY /dev/stdout | jq .
```

資訊清單看起來像：

```
{
  "hashAlg": "xxh128",
  "manifestVersion": "2023-03-03",
  "paths": [
    {
      "hash": "34178940e1ef9956db8ea7f7c97ed842",
      "mtime": 1721182390859777,
      "path": "output_dir/output.txt",
      "size": 117
    }
  ],
  "totalSize": 117
}
```

這會顯示輸出檔案的內容會儲存到 Amazon S3，就像儲存任務輸入檔案一樣。與輸入檔案類似，輸出檔案存放在 S3 中，其物件名稱包含檔案的雜湊和字首 DeadlineCloud/Data。

```
$ aws s3 ls --recursive s3://$Q1_S3_BUCKET | grep 34178940e1ef9956db8ea7f7c97ed842
2024-07-17 02:13:11        117 DeadlineCloud/
Data/34178940e1ef9956db8ea7f7c97ed842.xxh128
```

您可以使用截止日期雲端監視器或截止日期雲端 CLI，將任務的輸出下載到您的工作站：

```
deadline job download-output --farm-id $FARM_ID --queue-id $QUEUE1_ID --job-id $JOB_ID
```

所提交OutputDir任務中的任務參數值為 ./output_dir，因此輸出會下載至任務套件目錄中名為 output_dir的目錄。如果您將絕對路徑或不同的相對位置指定為 的值OutputDir，則輸出檔案會改為下載至該位置。

```
$ deadline job download-output --farm-id $FARM_ID --queue-id $QUEUE1_ID --job-id
 $JOB_ID
Downloading output from Job 'Job Attachments Explorer: Output'

Summary of files to download:
    /home/cloudshell-user/job_attachments_devguide_output/output_dir/output.txt (1
 file)

You are about to download files which may come from multiple root directories. Here are
 a list of the current root directories:
[0] /home/cloudshell-user/job_attachments_devguide_output
> Please enter the index of root directory to edit, y to proceed without changes, or n
 to cancel the download (0, y, n) [y]:

Downloading Outputs  [####################################]  100%
Download Summary:
    Downloaded 1 files totaling 117.0 B.
    Total download time of 0.14189 seconds at 824.0 B/s.
    Download locations (total file counts):
        /home/cloudshell-user/job_attachments_devguide_output (1 file)
```

在相依步驟中使用步驟中的檔案

此範例顯示任務中的一個步驟如何從相同任務中依賴的步驟存取輸出。

為了讓一個步驟的輸出可供另一個步驟使用，Deadline Cloud 會將其他動作新增至工作階段，以便在工作階段中執行任務之前下載這些輸出。您可以將這些步驟宣告為需要使用輸出之步驟的相依性，藉此告知從 下載輸出的步驟。

使用此範例job_attachments_devguide_output的任務套件。首先，從您複製的截止日期雲端範例 GitHub 儲存庫，在您的 AWS CloudShell 環境中建立複本。修改它以新增相依步驟，該步驟只會在現有步驟之後執行，並使用該步驟的輸出：

```
cp -r deadline-cloud-samples/job_bundles/job_attachments_devguide_output ~/

cat >> job_attachments_devguide_output/template.yaml << EOF
```

```
- name: DependentStep
  dependencies:
  - dependsOn: Step
  script:
    actions:
      onRun:
        command: /bin/cat
        args:
        - "{{Param.OutputDir}}/output.txt"
EOF
```

使用此修改後的任務套件建立的任務會以兩個單獨的工作階段執行，一個用於步驟「步驟」中的任務，另一個用於步驟「DependentStep」中的任務。

首先在 CloudShell 索引標籤中啟動截止日期雲端工作者代理程式。讓任何先前提交的任務完成執行，然後從日誌目錄中刪除任務日誌：

```
rm -rf ~/devdemo-logs/queue-*
```

接著，使用修改後的任務套件提交job_attachments_devguide_output任務。等待它在 CloudShell 環境中的工作者上完成執行。查看兩個工作階段的日誌：

```
# Change the value of FARM_ID to your farm's identifier
FARM_ID=farm-00112233445566778899aabbccddeeff
# Change the value of QUEUE1_ID to queue Q1's identifier
QUEUE1_ID=queue-00112233445566778899aabbccddeeff
# Change the value of WSALL_ID to the identifier of the WSAll storage profile
WSALL_ID=sp-00112233445566778899aabbccddeeff

deadline config set settings.storage_profile_id $WSALL_ID

deadline bundle submit --farm-id $FARM_ID --queue-id $QUEUE1_ID ./
job_attachments_devguide_output

# Wait for the job to finish running, and then:

cat demoenv-logs/queue-*/session-*
```

在名為 之步驟中任務的工作階段日誌中DependentStep，會執行兩個不同的下載動作：

```
2024-07-17 02:52:05,666 INFO ================================================
2024-07-17 02:52:05,666 INFO --------- Job Attachments Download for Job
2024-07-17 02:52:05,667 INFO ================================================
2024-07-17 02:52:05,667 INFO Syncing inputs using Job Attachments
2024-07-17 02:52:05,928 INFO Downloaded 207.0 B / 207.0 B of 1 file (Transfer rate: 0.0
 B/s)
2024-07-17 02:52:05,929 INFO Summary Statistics for file downloads:
Processed 1 file totaling 207.0 B.
Skipped re-processing 0 files totaling 0.0 B.
Total processing time of 0.03954 seconds at 5.23 KB/s.

2024-07-17 02:52:05,979 INFO
2024-07-17 02:52:05,979 INFO ================================================
2024-07-17 02:52:05,979 INFO --------- Job Attachments Download for Step
2024-07-17 02:52:05,979 INFO ================================================
2024-07-17 02:52:05,980 INFO Syncing inputs using Job Attachments
2024-07-17 02:52:06,133 INFO Downloaded 117.0 B / 117.0 B of 1 file (Transfer rate: 0.0
 B/s)
2024-07-17 02:52:06,134 INFO Summary Statistics for file downloads:
Processed 1 file totaling 117.0 B.
Skipped re-processing 0 files totaling 0.0 B.
Total processing time of 0.03227 seconds at 3.62 KB/s.
```

第一個動作會下載名為「步驟」的步驟所使用的`script.sh`檔案。第二個動作會從該步驟下載輸出。Deadline Cloud 會使用該步驟產生的輸出資訊清單做為輸入資訊清單，來決定要下載哪些檔案。

在相同的日誌中，您可以查看名為 "DependentStep" 步驟的輸出：

```
2024-07-17 02:52:06,213 INFO Output:
2024-07-17 02:52:06,216 INFO Script location: /sessions/session-5b33f/
assetroot-assetroot-3751a/script.sh
```

建立任務的資源限制

提交至 Deadline Cloud 的任務可能取決於多個任務之間共用的資源。例如，相較於浮動特定資源的授權，一個陣列可能有更多工作者。或者，共用檔案伺服器可能只能同時將資料提供給有限數量的工作者。在某些情況下，一或多個任務可以宣告所有這些資源，導致新工作者啟動時資源無法使用而造成錯誤。

為了協助解決此問題，您可以針對這些受限的資源使用限制。截止日期 雲端會考慮限制資源的可用性，並使用該資訊來確保資源在新工作者啟動時可用，以便任務因資源無法使用而失敗的可能性較低。

系統會為整個陣列建立限制。提交至佇列的任務只能取得與佇列相關聯的限制。如果您為未與佇列相關聯的任務指定限制，則該任務不相容且無法執行。

若要使用限制，您可以

- 建立限制
- 關聯限制和佇列
- 提交需要限制的任務

> **Note**
>
> 如果您執行的任務在佇列中具有與限制無關的資源限制，則該任務可能會耗用所有資源。如果您有受限的資源，請確定佇列中使用該資源的任務中的所有步驟都與限制相關聯。

對於陣列中定義的限制、與佇列相關聯，以及在任務中指定的限制，可能會發生以下四種情況之一：

- 如果您建立限制、將其與佇列建立關聯，並在任務的範本中指定限制，任務會執行並僅使用限制中定義的資源。
- 如果您建立限制，請在任務範本中指定限制，但不要將限制與佇列建立關聯，任務會標記為不相容，且不會執行。
- 如果您建立限制，請勿將其與佇列建立關聯，也不要在任務的範本中指定限制，任務會執行，但不會使用限制。
- 如果您完全不使用限制，任務會執行。

如果您將限制與多個佇列建立關聯，佇列會共用限制限制的資源。例如，如果您建立限制為 100，且一個佇列使用 60 個資源，則其他佇列只能使用 40 個資源。釋出資源時，任務可以從任何佇列取得該資源。

Deadline Cloud 提供兩個AWS CloudFormation指標，協助您監控限制提供的資源。您可以監控目前使用中的資源數量，以及限制中可用的資源數量上限。如需詳細資訊，請參閱《截止日期雲端開發人員指南》中的資源限制指標。

您可以將限制套用至任務範本中的任務步驟。當您在`hostRequirements`步驟的 `amounts`區段中指定限制的數量需求名稱，且具有相同限制與任務的佇列`amountRequirementName`相關聯時，為此步驟排程的任務會受到資源限制的限制。

如果步驟需要受限於達到限制的資源，則該步驟中的任務將不會由其他工作者取得。

您可以將多個限制套用至任務步驟。例如，如果步驟使用兩個不同的軟體授權，您可以為每個授權套用個別的限制。如果步驟需要兩個限制，並達到其中一個資源的限制，則在資源可用之前，該步驟中的任務將不會被其他工作者挑選。

停止和刪除限制

當您停止或刪除佇列與限制之間的關聯時，使用限制的任務會從需要此限制的步驟停止排程任務，並封鎖為步驟建立新工作階段。

處於就緒狀態的任務會保持就緒狀態，且任務會自動繼續與佇列和限制之間的關聯，再次變為作用中。您不需要重新排入任何任務的佇列。

當您停止或刪除佇列與限制之間的關聯時，有兩種選擇可讓您停止執行中的任務：

- 停止和取消任務 – 具有取得限制之工作階段的工作者會取消所有任務。
- 停止並完成執行中的任務 – 具有取得限制之工作階段的工作者完成其任務。

當您使用主控台刪除限制時，工作者會先停止執行任務，或在任務完成時最終停止執行任務。刪除關聯時，會發生下列情況：

- 需要限制的步驟標示為不相容。
- 包含這些步驟的整個任務都會取消，包括不需要限制的步驟。
- 任務標示為不相容。

如果與限制相關聯的佇列具有與限制數量需求名稱相符的機群功能相關聯的機群，則該機群將繼續處理具有指定限制的任務。

建立限制

您可以使用截止日期雲端主控台或截止日期雲端 API 中的 CreateLimit 操作來建立限制。限制是針對陣列定義，但與佇列相關聯。建立限制之後，您可以將其與一或多個佇列建立關聯。

建立限制

1. 從截止日期雲端主控台 (截止日期雲端主控台) 儀表板中，選取您要為其建立佇列的陣列。
2. 選擇要新增限制的陣列，選擇限制索引標籤，然後選擇建立限制。

3. 提供限制的詳細資訊。金額需求名稱是任務範本中用來識別限制的名稱。它必須以字首開頭，**amount.**後面接著金額名稱。在與限制相關聯的佇列中，數量需求名稱必須是唯一的。
4. 如果您選擇設定最大數量，即此限制允許的資源總數。如果您選擇無最大數量，則資源用量不受限制。即使資源用量不受限制，也會發出 CurrentCount Amazon CloudWatch 指標，以便您可以追蹤用量。如需詳細資訊，請參閱《截止日期雲端開發人員指南》中的 CloudWatch 指標。
5. 如果您已經知道應使用限制的佇列，現在可以選擇它們。您不需要關聯佇列即可建立限制。
6. 選擇建立限制。

關聯限制和佇列

建立限制之後，您可以將一或多個佇列與限制建立關聯。只有與限制相關聯的佇列會使用限制中指定的值。

您可以使用截止日期雲端主控台或截止日期雲端 API 中的 CreateQueueLimitAssociation 操作來建立與佇列的關聯。

將佇列與限制建立關聯

1. 從截止日期雲端主控台 (截止日期雲端主控台) 儀表板中，選取您要將限制與佇列建立關聯的陣列。
2. 選擇限制索引標籤，選擇要與佇列建立關聯的限制，然後選擇編輯限制。
3. 在關聯佇列區段中，選擇要與限制建立關聯的佇列。
4. 選擇儲存變更。

提交需要限制的任務

您可以將限制指定為任務或任務步驟的主機需求，以套用限制。如果您未在步驟中指定限制，且該步驟使用相關聯的資源，則在排程任務時，該步驟的用量不會計入限制。

有些截止日期雲端提交者可讓您設定主機需求。您可以在提交者中指定限制的金額需求名稱，以套用限制。

如果您的提交者不支援新增主機需求，您也可以編輯任務的任務範本來套用限制。

將限制套用至任務套件中的任務步驟

1. 使用文字編輯器開啟任務的任務範本。任務範本位於任務的任務套件目錄中。如需詳細資訊，請參閱《截止日期雲端開發人員指南》中的任務套件。

2. 尋找要套用限制之步驟的步驟定義。
3. 將以下內容新增至步驟定義。將 *amount.name* 取代為您限制的數量需求名稱。對於一般用途，您應該將min值設定為 1。

YAML

```
  hostRequirements:
    amounts:
    - name: amount.name
      min: 1
```

JSON

```
"hostRequirements": {
    "amounts": [
        {
            "name": "amount.name",
            "min": "1"
        }
    }
}
```

您可以在任務步驟中新增多個限制，如下所示。將 *amount.name_1* 和 *amount.name_2* 取代為您限制的數量需求名稱。

YAML

```
  hostRequirements:
    amounts:
    - name: amount.name_1
      min: 1
    - name: amount.name_2
      min: 1
```

JSON

```
"hostRequirements": {
    "amounts": [
        {
            "name": "amount.name_1",
```

```
            "min": "1"
        },
        {
            "name": "amount.name_2",
            "min": "1"
        }
    }
}
```

4. 將變更儲存至任務範本。

如何將任務提交至截止日期雲端

有許多不同的方式可以將任務提交至 AWS 截止日期雲端。本節說明您可以使用 Deadline Cloud 提供的工具或為工作負載建立自訂工具來提交任務的一些方式。

- 從終端機 – 當您第一次開發任務套件時，或當使用者使用命令列提交任務時
- 從指令碼 – 用於自訂和自動化工作負載
- 從應用程式 – 當使用者的工作在應用程式中時，或當應用程式的內容很重要時。

下列範例使用 `deadline` Python 程式庫和`deadline`命令列工具。兩者皆可從 PyPi 取得，並在 GitHub 上託管。

主題

- 從終端機將任務提交至截止日期雲端
- 使用指令碼將任務提交至截止日期雲端
- 在應用程式中提交任務

從終端機將任務提交至截止日期雲端

僅使用任務套件和截止日期雲端 CLI，您或您的技術性更高的使用者可以快速迭代撰寫任務套件，以測試提交任務。使用下列命令來提交任務套件：

```
deadline bundle submit <path-to-job-bundle>
```

如果您使用套件中沒有預設值的參數提交任務套件，您可以使用 / `-p --parameter`選項指定它們。

```
deadline bundle submit <path-to-job-bundle> -p <parameter-name>=<parameter-value> -
p ...
```

如需可用選項的完整清單，請執行 help 命令：

```
deadline bundle submit --help
```

使用 GUI 將任務提交至截止日期雲端

截止日期雲端 CLI 也隨附圖形化使用者介面，可讓使用者查看在提交任務之前必須提供的參數。如果您的使用者不想與命令列互動，您可以撰寫桌面捷徑，開啟對話方塊以提交特定任務套件：

```
deadline bundle gui-submit <path-to-job-bundle>
```

使用 `--browse`選項可以讓使用者選取任務套件：

```
deadline bundle gui-submit --browse
```

如需可用選項的完整清單，請執行 help 命令：

```
deadline bundle gui-submit --help
```

使用指令碼將任務提交至截止日期雲端

若要自動將任務提交至截止日期雲端，您可以使用 bash、Powershell 和批次檔案等工具編寫任務指令碼。

您可以新增功能，例如從環境變數或其他應用程式填入任務參數。您也可以在資料列中提交多個任務，或編寫要提交的任務套件建立指令碼。

使用 Python 提交任務

Deadline Cloud 也有開放原始碼 Python 程式庫，可與服務互動。來源碼可在 GitHub 上取得。

此程式庫可透過 pip () 在 pypi 上提供`pip install deadline`。它與截止日期雲端 CLI 工具使用的程式庫相同：

```
from deadline.client import api

job_bundle_path = "/path/to/job/bundle"
```

```
job_parameters = [
    {
        "name": "parameter_name",
        "value": "parameter_value"
    },
]

job_id = api.create_job_from_job_bundle(
    job_bundle_path,
    job_parameters
)
print(job_id)
```

若要建立類似 deadline bundle gui-submit命令的對話方塊，您可以從 使用 show_job_bundle_submitter函數deadline.client.ui.job_bundle_submitter。

下列範例會啟動 Qt 應用程式，並顯示任務套件提交者：

```
# The GUI components must be installed with pip install "deadline[gui]"
import sys
from qtpy.QtWidgets import QApplication
from deadline.client.ui.job_bundle_submitter import show_job_bundle_submitter

app = QApplication(sys.argv)
submitter = show_job_bundle_submitter(browse=True)
submitter.show()
app.exec()
print(submitter.create_job_response)
```

若要建立您自己的對話方塊，您可以在 中使用 SubmitJobToDeadlineDialog類別deadline.client.ui.dialogs.submit_job_to_deadline_dialog。您可以傳入值、嵌入您自己的任務特定索引標籤，並判斷如何建立 （或傳入） 任務套件。

在應用程式中提交任務

若要讓使用者輕鬆提交任務，您可以使用 提供的指令碼執行時間或外掛程式系統。使用者擁有熟悉的界面，您可以建立強大的工具，協助使用者提交工作負載。

在應用程式中內嵌任務套件

此範例示範提交您在應用程式中提供的任務套件。

若要讓使用者存取這些任務套件，請建立內嵌在啟動截止日期雲端 CLI 的選單項目中的指令碼。

下列指令碼可讓使用者選取任務套件：

```
deadline bundle gui-submit --install-gui
```

若要改為在選單項目中使用特定任務套件，請使用下列項目：

```
deadline bundle gui-submit </path/to/job/bundle> --install-gui
```

這會開啟一個對話方塊，使用者可以修改任務參數、輸入和輸出，然後提交任務。您可以為使用者在應用程式中提交的不同任務套件擁有不同的功能表項目。

如果您使用任務套件提交的任務包含類似的參數和跨提交的資產參考，您可以在基礎任務套件中填寫預設值。

從應用程式取得資訊

若要從應用程式提取資訊，讓使用者不必手動將其新增至提交，您可以將 Deadline Cloud 與應用程式整合，讓使用者可以使用熟悉的界面提交任務，而無需結束應用程式或使用命令列工具。

如果您的應用程式具有支援 Python 和 pyside/pyqt 的指令碼執行期，您可以使用截止日期雲端用戶端程式庫中的 GUI 元件來建立 UI。如需範例，請參閱 GitHub 上 Maya 整合的截止日期雲端。

Deadline Cloud 用戶端程式庫提供下列操作，協助您提供強大的整合使用者體驗：

- 提取佇列環境參數、任務參數和資產參考表單環境變數，並呼叫應用程式 SDK。
- 設定任務套件中的參數。為了避免修改原始套件，您應該複製套件並提交副本。

如果您使用 `deadline bundle gui-submit`命令來提交任務套件，您必須以程式設計方式 `parameter_values.yaml`和 `asset_references.yaml` 檔案，以從應用程式傳遞資訊。如需這些檔案的詳細資訊，請參閱 截止日期雲端的開啟任務描述 (OpenJD) 範本。

如果您需要比 OpenJD 提供的控制項更複雜的控制項、需要從使用者抽象化任務，或想要使整合符合應用程式的視覺化樣式，您可以撰寫自己的對話方塊，呼叫截止日期雲端用戶端程式庫來提交任務。

在截止日期雲端中排程任務

建立任務之後， AWS 截止日期 雲端會將其排程在與佇列相關聯的一或多個機群上進行處理。根據排程組態、為機群設定的功能，以及特定步驟的主機需求來選擇處理特定任務的機群。

下列各節提供排程任務程序的詳細資訊。

排程組態

您可以透過在佇列上設定排程組態，來設定截止日期雲端如何排程佇列中的任務。排程組態會控制工作者在任務間的分佈方式。

您可以使用截止日期雲端主控台或呼叫 CreateQueue 或 UpdateQueue APIs來設定排程組態。

有三種可用的排程組態：

- 優先順序，first-in-first-out(`priorityFifo`) – 排定最高優先順序、最先提交的任務 （預設）。
- 優先順序，平衡 (`priorityBalanced`) – 以最高優先順序將工作者平均分配到各個任務。
- 加權、平衡 (`weightedBalanced`) – 使用加權公式來判斷工作者如何在任務之間分配。

在所有排程組態中，進行中的任務會在做出新的排程決策之前執行到完成。如果您在任務執行時變更排程組態，則變更僅適用於接下來指派工作者的情況。執行中的任務不會中斷或重新指派。

優先順序，first-in-first-out

優先順序，first-in-first-out(`priorityFifo`) 是新佇列的預設排程組態。Deadline Cloud 會先將工作者指派給最高優先順序的任務。當多個任務具有相同的優先順序時，最舊 （最早提交） 的任務會先收到所有可用的工作者。

當您想要嚴格排序任務時，請使用優先順序 FIFO。當任務應按提交順序一次完成一個時，此組態是適當的，例如循序管道階段或批次處理，每個任務都必須在下一個任務開始之前完成。

此組態沒有其他參數。

優先順序，平衡

優先順序、平衡 (`priorityBalanced`) 以最高優先順序將工作者平均分配到所有任務。當最高優先順序只有一個任務時，Deadline Cloud 會將所有工作者指派給該任務。當多個任務共用最高優先順序時，工作者會平均分配到其中。如果工作者無法平均分配，則額外的工作者會分佈在最高優先順序的任務中。

當多個藝術家或使用者以相同的優先順序提交任務，且每個使用者需要立即意見回饋時，請使用優先順序平衡。此組態可確保沒有任何單一任務會獨佔所有可用的工作者，因此會在提交後立即分配所有使用者。

如果任務的剩餘任務少於工作者的份額，剩餘工作者會重新分配到相同優先順序層級的其他任務。如果最高優先順序的所有任務都已完全配置，剩餘工作者會層疊到下一個最高優先順序層級的任務。

此組態具有下列參數：

renderingTaskBuffer

控制工作者黏性。只有在轉譯任務的差異超過 renderingTaskBuffer值時，工作者才會從目前的任務切換到相同優先順序的另一個任務。較高的值可讓工作者處理目前任務的時間更長，減少內容切換。預設值為 1。

加權、平衡

加權、平衡 (weightedBalanced) 使用公式來計算每個任務的權重。Deadline Cloud 會先將工作者指派給最高權重的任務。如果多個任務具有相同的權重，則工作者會分配到其中。

當您需要精細控制工作者如何分散在具有不同優先順序、錯誤率和提交時間的任務時，請使用加權平衡。此組態適用於複雜的轉譯陣列環境，其中您想要調整任務優先順序、任務存留期、錯誤處理和工作者黏性之間的平衡。

每個任務的權重計算方式如下：

```
weight = (job.Priority * priorityWeight) +
         (job.Errors * errorWeight) +
         ((currentTimeInSeconds - job.SubmissionTime) * submissionTimeWeight) +
         ((job.RenderingTasks - renderingTaskBuffer) * renderingTaskWeight)
```

只有在工作者目前正在處理任務時，才會套用renderingTaskBuffer元件。renderingTaskWeight 通常設定為負值，以便具有指派工作者的任務獲得較低的權重，將其他任務帶到佇列前面。通常errorWeight也是負數，因此具有錯誤的作業會取消優先順序。您可以使用排程覆寫來執行最低和最高優先順序任務。

此組態具有下列參數：

priorityWeight

套用至任務優先順序的權重。正值表示先排定優先順序較高的任務。預設值為 100.0。範圍： 0到10000。

errorWeight

套用至任務錯誤計數的權重。負值表示會先排程沒有錯誤的任務。預設值為 -10.0。範圍：-10000到 10000。

submissionTimeWeight

套用至任務提交時間的權重（以秒為單位）。正值表示先排程先前提交的任務。預設值為 3.0。範圍： 0到 10000。

renderingTaskWeight

套用至目前為任務轉譯的任務數量的權重。負值表示接下來排定工作者較少的任務。預設值為 -100.0。範圍： -10000到 10000。

renderingTaskBuffer

轉譯任務權重生效前的轉譯任務數量。正值可讓工作者處理目前的任務。預設值為 1。範圍： 0到 1000。

maxPriorityOverride

選用。設為 時alwaysScheduleFirst，無論加權公式為何，在最高優先順序 (100) 的任務一律會排定在其他任務之前。當多個任務具有最高優先順序時，會使用標準加權公式中斷關聯。缺少覆寫時，最高優先順序任務會使用沒有特殊處理的標準加權公式。

minPriorityOverride

選用。設為 時alwaysScheduleLast，無論加權公式為何，一律會在其他任務之後排定最低優先順序 (0) 的任務。當多個任務具有最低優先順序時，會使用標準加權公式中斷關聯。缺少覆寫時，最低優先順序任務會使用沒有特殊處理的標準加權公式。

判斷機群相容性

建立任務之後，截止日期雲端會根據與提交任務的佇列相關聯的機群功能，檢查任務中每個步驟的主機需求。如果機群符合主機需求，任務會進入 READY 狀態。

如果任務中的任何步驟具有與佇列相關聯的機群無法滿足的需求，則該步驟的狀態會設為 NOT_COMPATIBLE。此外，任務中的其餘步驟也會取消。

機群的功能是在機群層級設定。即使機群中的工作者符合任務的要求，如果其機群不符合任務的要求，則不會從任務指派任務。

下列任務範本有一個步驟，指定步驟的主機需求：

```
name: Sample Job With Host Requirements
specificationVersion: jobtemplate-2023-09
```

```
steps:
- name: Step 1
  script:
    actions:
      onRun:
        args:
        - '1'
        command: /usr/bin/sleep
  hostRequirements:
    amounts:
    # Capabilities starting with "amount." are amount capabilities. If they start with
 "amount.worker.",
    # they are defined by the OpenJD specification. Other names are free for custom
 usage.
    - name: amount.worker.vcpu
      min: 4
      max: 8
    attributes:
    - name: attr.worker.os.family
      anyOf:
      - linux
```

此任務可以排程到具有下列功能的機群：

```
{
    "vCpuCount": {"min": 4, "max": 8},
    "memoryMiB": {"min": 1024},
    "osFamily": "linux",
    "cpuArchitectureType": "x86_64"
}
```

此任務無法排程到具有下列任何功能的機群：

```
{
    "vCpuCount": {"min": 4},
    "memoryMiB": {"min": 1024},
    "osFamily": "linux",
    "cpuArchitectureType": "x86_64"
}
    The vCpuCount has no maximum, so it exceeds the maximum vCPU host requirement.

{
    "vCpuCount": {"max": 8},
```

```
    "memoryMiB": {"min": 1024},
    "osFamily": "linux",
    "cpuArchitectureType": "x86_64"
}
    The vCpuCount has no minimum, so it doesn't satisfy the minimum vCPU host
 requirement.

{
    "vCpuCount": {"min": 4, "max": 8},
    "memoryMiB": {"min": 1024},
    "osFamily": "windows",
    "cpuArchitectureType": "x86_64"
}
    The osFamily doesn't match.
```

機群擴展

當任務指派給相容的服務受管機群時，機群會自動擴展。機群中的工作者數量會根據機群可執行的任務數量而變更。

當任務指派給客戶受管機群時，工作者可能已存在，或者可以使用事件型自動擴展來建立。如需詳細資訊，請參閱《Amazon EC2 Auto Scaling 使用者指南》中的使用 EventBridge 來處理自動擴展事件。Amazon EC2 Auto Scaling

工作階段

任務中的任務分為一或多個工作階段。工作者會執行工作階段來設定環境、執行任務，然後銷毀環境。每個工作階段都由工作者必須採取的一或多個動作組成。

當工作者完成區段動作時，可以將其他工作階段動作傳送給工作者。工作者會在工作階段中重複使用現有的環境和任務附件，以更有效率的方式完成任務。

在服務受管機群工作者上，工作階段目錄會在工作階段結束後刪除，但在工作階段之間保留其他目錄。此行為可讓您針對可在多個工作階段間重複使用的資料實作快取策略。若要快取工作階段之間的資料，請將資料存放在執行任務之使用者的主目錄下。例如，Conda 套件會快取在Windows工作者和/home/job-user/.conda-pkgsLinux工作者C:\Users\job-user\.conda-pkgs上的 的任務使用者主目錄下。在工作者關閉之前，此資料仍然可用。

任務附件是由做為截止日期 Cloud CLI 任務套件一部分的提交者建立。您也可以使用 create-job AWS CLI 命令的 --attachments選項來建立任務附件。環境定義在兩個位置：連接到特定佇列的佇列環境，以及任務範本中定義的任務和步驟環境。

有四種工作階段動作類型：

- syncInputJobAttachments – 將輸入任務附件下載至工作者。
- envEnter – 執行環境onEnter的動作。
- taskRun – 執行任務onRun的動作。
- envExit – 執行環境onExit的動作。

下列任務範本具有步驟環境。它具有設定步驟環境onEnter的定義、定義要執行之任務onRun的定義，以及縮減步驟環境onExit的定義。為此任務建立的工作階段將包含 envEnter動作、一或多個taskRun動作，以及 envExit動作。

```
name: Sample Job with Maya Environment
specificationVersion: jobtemplate-2023-09
steps:
- name: Maya Step
  stepEnvironments:
  - name: Maya
    description: Runs Maya in the background.
    script:
      embeddedFiles:
      - name: initData
        filename: init-data.yaml
        type: TEXT
        data: |
          scene_file: MyAwesomeSceneFile
          renderer: arnold
          camera: persp
      actions:
        onEnter:
          command: MayaAdaptor
          args:
          - daemon
          - start
          - --init-data
          - file://{{Env.File.initData}}
        onExit:
          command: MayaAdaptor
          args:
          - daemon
          - stop
  parameterSpace:
```

```
    taskParameterDefinitions:
    - name: Frame
      range: 1-5
      type: INT
  script:
    embeddedFiles:
    - name: runData
      filename: run-data.yaml
      type: TEXT
      data: |
        frame: {{Task.Param.Frame}}
    actions:
      onRun:
        command: MayaAdaptor
        args:
        - daemon
        - run
        - --run-data
        - file://{{ Task.File.runData }}
```

工作階段動作管道

工作階段動作管道可讓排程器將多個工作階段動作預先指派給工作者。工作者接著可以依序執行這些動作，減少或消除任務之間的閒置時間。

若要建立初始指派，排程器會建立具有一個任務的工作階段、工作者完成任務，然後排程器會分析任務持續時間以判斷未來的指派。

為了讓排程器有效，有任務持續時間規則。對於一分鐘以內的任務，排程器會使用 2 的動力成長模式。例如，對於 1 秒的任務，排程器會指派 2 個新任務，然後 4 個新任務，然後 8 個新任務。對於超過一分鐘的任務，排程器只會指派一個新任務，且管道會保持停用狀態。

若要計算管道大小，排程器會執行下列動作：

- 使用已完成任務的平均任務持續時間
- 讓工作者保持忙碌一分鐘的目標是
- 僅考慮相同工作階段中的任務
- 不跨工作者共用持續時間資料

隨著工作階段動作繁複，工作者會立即開始新的任務，而且排程器請求之間沒有等待時間。它也為長時間執行的程序提供更高的工作者效率和更好的任務分佈。

此外，如果有新的較高優先順序任務可用，工作者將在目前的工作階段結束之前完成所有先前指派的工作，並指派來自較高優先順序任務的新工作階段。

步驟相依性

Deadline Cloud 支援在步驟之間定義相依性，讓一個步驟等待另一個步驟完成再開始。您可以為步驟定義多個相依性。在其所有相依性完成之前，不會排程具有相依性的步驟。

如果任務範本定義循環相依性，則會拒絕任務，並將任務狀態設定為 CREATE_FAILED。

下列任務範本會建立具有兩個步驟的任務。 StepB 取決於 StepA。 StepB只會在 StepA 成功完成後執行。

建立任務後， StepA 會處於 READY 狀態，並StepB處於 PENDING 狀態。StepA 完成後， StepB會移至 READY 狀態。如果 StepA 失敗，或StepA如果 已取消，則 StepB會移至 CANCELED 狀態。

您可以設定多個步驟的相依性。例如，如果 同時StepC取決於 StepA和 StepB，則在另外兩個步驟完成之前， StepC不會開始。

步驟相依性有下列限制：

- 每個步驟的相依性 – 步驟最多可依賴 128 個其他步驟。
- 每個步驟的取用者 – 最多可以有 32 個其他步驟取決於單一步驟。

```
name: Step-Step Dependency Test
specificationVersion: 'jobtemplate-2023-09'
steps:
- name: A
  script:
    actions:
      onRun:
        command: bash
        args: ['{{ Task.File.run }}']
    embeddedFiles:
      - name: run
        type: TEXT
        data: |
          #!/bin/env bash

          set -euo pipefail
```

```
          sleep 1
          echo Task A Done!
- name: B
  dependencies:
  - dependsOn: A # This means Step B depends on Step A
  script:
    actions:
      onRun:
        command: bash
        args: ['{{ Task.File.run }}']
    embeddedFiles:
      - name: run
        type: TEXT
        data: |
          #!/bin/env bash

          set -euo pipefail

          sleep 1
          echo Task B Done!
```

在截止日期雲端中修改任務

您可以使用下列 AWS Command Line Interface (AWS CLI) update命令來修改任務的組態，或設定任務、步驟或任務的目標狀態：

- aws deadline update-job
- aws deadline update-step
- aws deadline update-task

在下列 update命令範例中，將每個 取代*user input placeholder*為您自己的資訊。

Example– 將任務排入佇列

除非有步驟相依性，否則任務中的所有任務都會切換到 READY 狀態。具有相依性的步驟會在還原PENDING時切換到 READY或 。

```
aws deadline update-job \
--farm-id farmID \
--queue-id queueID \
```

```
--job-id jobID \
--target-task-run-status PENDING
```

Example– 取消任務

任務中沒有 狀態SUCCEEDED或FAILED標記為 的所有任務CANCELED。

```
aws deadline update-job \
--farm-id farmID \
--queue-id queueID \
--job-id jobID \
--target-task-run-status CANCELED
```

Example– 標記任務失敗

任務中狀態為 的所有任務SUCCEEDED都保持不變。所有其他任務都會標示為 FAILED。

```
aws deadline update-job \
--farm-id farmID \
--queue-id queueID \
--job-id jobID \
--target-task-run-status FAILED
```

Example– 成功標記任務

任務中的所有任務都會移至 SUCCEEDED 狀態。

```
aws deadline update-job \
--farm-id farmID \
--queue-id queueID \
--job-id jobID \
--target-task-run-status SUCCEEDED
```

Example– 暫停任務

SUCCEEDED、 CANCELED或 FAILED 狀態的任務不會變更。所有其他任務都會標示為 SUSPENDED。

```
aws deadline update-job \
--farm-id farmID \
--queue-id queueID \
--job-id jobID \
```

```
--target-task-run-status SUSPENDED
```

Example– 變更任務的優先順序

更新佇列中任務的優先順序，以變更其排程的順序。較高優先順序的任務通常會先排程。

```
aws deadline update-job \
--farm-id farmID \
--queue-id queueID \
--job-id jobID \
--priority 100
```

Example– 變更允許的失敗任務數量

在取消其餘任務之前，更新任務可擁有的失敗任務數量上限。

```
aws deadline update-job \
--farm-id farmID \
--queue-id queueID \
--job-id jobID \
--max-failed-tasks-count 200
```

Example– 變更允許的任務重試次數

在任務失敗之前，更新任務的重試次數上限。達到重試次數上限的任務無法重新排入佇列，直到此值增加為止。

```
aws deadline update-job \
--farm-id farmID \
--queue-id queueID \
--job-id jobID \
--max-retries-per-task 10
```

Example– 封存任務

將任務的生命週期狀態更新為 ARCHIVED。封存的任務無法排程或修改。您只能封存處於 FAILED、SUCCEEDED、 CANCELED或 SUSPENDED 狀態的任務。

```
aws deadline update-job \
--farm-id farmID \
--queue-id queueID \
```

```
--job-id jobID \
--lifecycle-status ARCHIVED
```

Example– 變更任務的名稱

更新任務的顯示名稱。任務名稱長度最多可達 128 個字元。

```
aws deadline update-job \
--farm-id farmID \
--queue-id queueID \
--job-id jobID \
--name "New Job Name"
```

Example– 變更任務的描述

更新任務的描述。描述長度最多可達 2048 個字元。若要移除現有的描述，請傳遞空字串。

```
aws deadline update-job \
--farm-id farmID \
--queue-id queueID \
--job-id jobID \
--description "New Job Description"
```

Example– 將步驟排入佇列

除非有步驟相依性，否則步驟中的所有任務都會切換到 READY 狀態。具有相依性的步驟中的任務會切換到 READY或 PENDING，並還原任務。

```
aws deadline update-step \
--farm-id farmID \
--queue-id queueID \
--job-id jobID \
--step-id stepID \
--target-task-run-status PENDING
```

Example– 取消步驟

步驟中沒有 狀態SUCCEEDED或FAILED標記為 的所有任務CANCELED。

```
aws deadline update-step \
--farm-id farmID \
```

```
--queue-id queueID \
--job-id jobID \
--step-id stepID \
--target-task-run-status CANCELED
```

Example– 標記步驟失敗

該步驟中狀態為 的所有任務SUCCEEDED都保持不變。所有其他任務都會標示為 FAILED。

```
aws deadline update-step \
--farm-id farmID \
--queue-id queueID \
--job-id jobID \
--step-id stepID \
--target-task-run-status FAILED
```

Example– 將步驟標記為成功

步驟中的所有任務都會標示為 SUCCEEDED。

```
aws deadline update-step \
--farm-id farmID \
--queue-id queueID \
--job-id jobID \
--step-id stepID \
--target-task-run-status SUCCEEDED
```

Example– 暫停步驟

SUCCEEDED、 CANCELED或 FAILED 狀態的步驟中的任務不會變更。所有其他任務都會標示為 SUSPENDED。

```
aws deadline update-step \
--farm-id farmID \
--queue-id queueID \
--job-id jobID \
--step-id stepID \
--target-task-run-status SUSPENDED
```

Example– 變更任務的狀態

當您使用update-task截止日期雲端 CLI 命令時，任務會切換到指定的狀態。

```
aws deadline update-task \
--farm-id farmID \
--queue-id queueID \
--job-id jobID \
--step-id stepID \
--task-id taskID \
--target-task-run-status SUCCEEDED | SUSPENDED | CANCELED | FAILED | PENDING
```

建立和使用截止日期 雲端客戶管理的機群

當您建立客戶受管機群 (CMF) 時，您可以完全控制處理管道。您可以定義每個工作者的網路和軟體環境。Deadline Cloud 可做為任務的儲存庫和排程器。

工作者可以是 Amazon Elastic Compute Cloud (Amazon EC2) 執行個體、主機代管設施中的工作者，或現場部署工作者。每個工作者都必須執行截止日期雲端工作者代理程式。所有工作者都必須能夠存取截止日期雲端服務端點。

下列主題說明如何使用 Amazon EC2 執行個體建立基本 CMF。

主題

建立客戶受管機群

若要建立客戶受管機群 (CMF)，請完成下列步驟。

Deadline Cloud console

使用截止日期雲端主控台建立客戶受管機群

1. 開啟截止日期雲端主控台。
2. 選取陣列。可用的陣列清單隨即顯示。
3. 選取您要使用的陣列名稱。
4. 選取機群索引標籤，然後選擇建立機群。

5. 輸入機群的名稱。
6. （選用）輸入機群的描述。
7. 針對機群類型選取客戶受管。
8. 選取機群的服務存取權。
   a. 我們建議為每個機群使用建立和使用新的服務角色選項，以獲得更精細的許可控制。預設會選取此選項。
   b. 您也可以選取選擇服務角色，以使用現有的服務角色。
9. 檢閱您的選擇，然後選擇下一步。
10. 為您的機群選取作業系統。機群的所有工作者都必須擁有通用的作業系統。
11. 選取主機 CPU 架構。
12. 選取最小和最大 vCPU 和記憶體 硬體功能，以滿足機群的工作負載需求。
13. 選取 Auto Scaling 類型。如需詳細資訊，請參閱使用 EventBridge 來處理 Auto Scaling 事件。
    - 無擴展：您正在建立內部部署機群，並希望退出截止 Cloud Auto Scaling。
    - 擴展建議：您正在建立 Amazon Elastic Compute Cloud (Amazon EC2) 機群。
14. （選用）選取箭頭以展開新增功能區段。
15. （選用）選取新增 GPU 功能 - 選用的核取方塊，然後輸入最小和最大 GPUs 和記憶體。
16. 檢閱您的選擇，然後選擇下一步。
17. （選用）定義自訂工作者功能，然後選擇下一步。
18. 使用下拉式清單，選取要與機群建立關聯的一或多個佇列。

> **Note**
>
> 我們建議僅將機群與位於相同信任界限的佇列建立關聯。此建議可確保相同工作者上執行任務之間的強大安全界限。

19. 檢閱佇列關聯，然後選擇下一步。
20. （選用）對於預設 Conda 佇列環境，我們將為您的佇列建立環境，以安裝任務請求的 conda 套件。

> **Note**
>
> conda 佇列環境用於安裝任務請求的 conda 套件。一般而言，您應該取消勾選與 CMFs相關聯的佇列上的 conda 佇列環境，因為 CMFs 預設不會安裝必要的 conda 命令。

21. （選用）將標籤新增至 CMF。如需詳細資訊，請參閱標記您的 AWS 資源。
22. 檢閱您的機群組態並進行任何變更，然後選擇建立機群。
23. 選取機群索引標籤，然後記下機群 ID。

AWS CLI

使用 AWS CLI 建立客戶受管機群

1. 開啟終端機。
2. 在新編輯器`fleet-trust-policy.json`中建立 。
   a. 新增下列 IAM 政策，將 *ITALICIZED* 文字取代為您的帳戶 AWS ID 和截止日期雲端陣列 ID。

      JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "credentials.deadline.amazonaws.com"
            },
            "Action": "sts:AssumeRole",
            "Condition": {
                "StringEquals": {
                    "aws:SourceAccount": "111122223333"
                },
                "ArnEquals": {
                    "aws:SourceArn":
 "arn:aws:deadline:*:111122223333:farm/FARM_ID"
                }
```

```
            }
        }
    ]
}
```

b. 儲存您的變更。

3. 建立 `fleet-policy.json`。

a. 新增下列 IAM 政策。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "deadline:AssumeFleetRoleForWorker",
                "deadline:UpdateWorker",
                "deadline:DeleteWorker",
                "deadline:UpdateWorkerSchedule",
                "deadline:BatchGetJobEntity",
                "deadline:AssumeQueueRoleForWorker"
            ],
            "Resource": "arn:aws:deadline:*:111122223333:*",
            "Condition": {
                "StringEquals": {
                    "aws:PrincipalAccount": "${aws:ResourceAccount}"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogStream"
            ],
            "Resource": "arn:aws:logs:*:*:*://deadline/*",
            "Condition": {
                "StringEquals": {
                    "aws:PrincipalAccount": "${aws:ResourceAccount}"
                }
```

```
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "logs:PutLogEvents",
                "logs:GetLogEvents"
            ],
            "Resource": "arn:aws:logs:*:*:*:/aws/deadline/*",
            "Condition": {
                "StringEquals": {
                    "aws:PrincipalAccount": "${aws:ResourceAccount}"
                }
            }
        }
    ]
}
```

b. 儲存您的變更。

4. 為機群中的工作者新增要使用的 IAM 角色。

```
aws iam create-role --role-name FleetWorkerRoleName --assume-role-policy-
document file://fleet-trust-policy.json
aws iam put-role-policy --role-name FleetWorkerRoleName --policy-name
 FleetWorkerPolicy --policy-document file://fleet-policy.json
```

5. 建立 `create-fleet-request.json`。

a. 新增下列 IAM 政策，將 ITALICIZED 文字取代為您的 CMF 值。

> **Note**
>
> 您可以在 中找到 *ROLE_ARN*`create-cmf-fleet.json`。
> 對於 *OS_FAMILY*，您必須選擇其中一個 `linux`、 `macos`或 `windows`。

```
{
    "farmId": "FARM_ID",
    "displayName": "FLEET_NAME",
    "description": "FLEET_DESCRIPTION",
    "roleArn": "ROLE_ARN",
    "minWorkerCount": 0,
```

```
    "maxWorkerCount": 10,
    "configuration": {
        "customerManaged": {
            "mode": "NO_SCALING",
            "workerCapabilities": {
                "vCpuCount": {
                    "min": 1,
                    "max": 4
                },
                "memoryMiB": {
                    "min": 1024,
                    "max": 4096
                },
                "osFamily": "OS_FAMILY",
                "cpuArchitectureType": "x86_64",
            },
        },
    }
}
```

b. 儲存您的變更。

6. 建立您的機群。

```
aws deadline create-fleet --cli-input-json file://create-fleet-request.json
```

工作者主機設定和組態

工作者主機是指執行截止日期雲端工作者的主機機器。本節說明如何設定工作者主機，並根據您的特定需求進行設定。每個工作者主機都會執行稱為工作者代理程式的程式。工作者代理程式負責：

- 管理工作者生命週期。
- 同步指派的工作、進度和結果。
- 監控執行中的工作。
- 將日誌轉送至設定的目的地。

我們建議您使用提供的截止日期雲端工作者代理程式。工作者代理程式是開放原始碼，我們鼓勵功能請求，但您也可以開發和自訂以滿足您的需求。

若要完成以下章節中的任務，您需要下列項目：

Linux

- Linux以 為基礎的 Amazon Elastic Compute Cloud (Amazon EC2) 執行個體。我們建議使用 Amazon Linux 2023。
- sudo 權限
- Python 3.9 或更新版本

Windows

- Windows以 為基礎的 Amazon Elastic Compute Cloud (Amazon EC2) 執行個體。我們建議使用 Windows Server 2022。
- 工作者主機的管理員存取權
- 為所有使用者安裝 Python 3.9 或更新版本

建立和設定 Python 虛擬環境

Linux 如果您已安裝 Python 3.9 或更新版本並將其放置在 中，您可以在 上建立 Python 虛擬環境。PATH

> **Note**
>
> 在 上Windows，代理程式檔案必須安裝在 Python 的全域 site-packages 目錄中。目前不支援 Python 虛擬環境。

建立和啟用 Python 虛擬環境

1. 以root使用者身分開啟終端機 （ 或使用 sudo / su)。
2. 建立和啟用 Python 虛擬環境。

```
python3 -m venv /opt/deadline/worker
source /opt/deadline/worker/bin/activate
pip install --upgrade pip
```

安裝截止日期雲端工作者代理程式

在您設定 Python 並在 上建立虛擬環境之後Linux，請安裝截止日期雲端工作者代理程式 Python 套件。

安裝工作者代理程式 Python 套件

Linux

1. 以root使用者身分開啟終端機 （ 或使用 sudo / su)。
2. 從 PyPI 下載並安裝截止日期雲端工作者代理程式套件：

```
/opt/deadline/worker/bin/python -m pip install deadline-cloud-worker-agent
```

Windows

1. 開啟管理員命令提示字元或 PowerShell 終端機。
2. 從 PyPI 下載並安裝截止日期雲端工作者代理程式套件：

```
python -m pip install deadline-cloud-worker-agent
```

當您的Windows工作者主機需要長路徑名稱 （ 大於 250 個字元） 時，您必須啟用長路徑名稱，如下所示：

為Windows工作者主機啟用長路徑

1. 請確定已啟用長路徑登錄機碼。如需詳細資訊，請參閱在 Microsoft 網站上啟用日誌路徑的登錄設定。
2. 安裝適用於桌面 C++ x86 應用程式的 Windows SDK。如需詳細資訊，請參閱Windows開發中心中的 Windows SDK。
3. 在安裝工作者代理程式的環境中開啟 Python 安裝位置。預設值為 C:\Program Files \Python311。有一個名為 的可執行檔pythonservice.exe。
4. 在pythonservice.exe.manifest相同位置建立名為 的新檔案。新增下列項目：

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<assembly xmlns="urn:schemas-microsoft-com:asm.v1" manifestVersion="1.0">
  <assemblyIdentity type="win32" name="pythonservice" processorArchitecture="x86"
 version="1.0.0.0"/>
```

```
  <application xmlns="urn:schemas-microsoft-com:asm.v3">
    <windowsSettings>
      <longPathAware xmlns="http://schemas.microsoft.com/SMI/2016/
WindowsSettings">true</longPathAware>
    </windowsSettings>
  </application>
</assembly>
```

5. 開啟命令提示字元，並在您建立的資訊清單檔案的位置執行下列命令：

```
"C:\Program Files (x86)\Windows Kits\10\bin\10.0.26100.0\x86\mt.exe" -manifest
 pythonservice.exe.manifest -outputresource:pythonservice.exe;#1
```

您應該會看到類似下列的輸出：

```
Microsoft (R) Manifest Tool
Copyright (c) Microsoft Corporation.
All rights reserved.
```

工作者現在可以存取長路徑。若要清除，請移除 pythonservice.exe.manifest 檔案並解除安裝 SDK。

設定截止日期雲端工作者代理程式

您可以透過三種方式設定截止日期雲端工作者代理程式設定。我們建議您透過執行 install-deadline-worker工具來使用作業系統設定。

工作者代理程式不支援在 Windows 上以網域使用者身分執行 。若要以網域使用者身分執行任務，您可以在設定執行任務的佇列使用者時指定網域使用者帳戶。如需詳細資訊，請參閱《截止日期雲端使用者指南》中的截止日期雲端佇列的步驟 7。 AWS

命令列引數 — 當您從命令列執行截止日期雲端工作者代理程式時，可以指定引數。某些組態設定無法透過命令列引數使用。若要查看所有可用的命令列引數，請輸入 deadline-worker-agent --help。

環境變數 — 您可以透過設定以 開頭的環境變數，來設定截止日期雲端工作者代理程式DEADLINE_WORKER_。例如，若要查看所有可用的命令列引數，您可以使用 將工作者代理程式的輸出export DEADLINE_WORKER_VERBOSE=true設定為詳細。如需更多範例和資訊，請參閱 /etc/amazon/deadline/worker.toml.example 上的 Linux或 C:\ProgramData\Amazon\Deadline\Config\worker.toml.example上的 Windows。

組態檔案 — 安裝工作者代理程式時，它會建立位於 /etc/amazon/deadline/worker.tomlLinux 或 C:\ProgramData\Amazon\Deadline\Config\worker.toml上的組態檔案Windows。工作者代理程式會在啟動時載入此組態檔案。您可以使用範例組態檔案 (/etc/amazon/deadline/worker.toml.example 上的 Linux或 C:\ProgramData\Amazon\Deadline\Config\worker.toml.example上的 Windows)，根據您的特定需求量身打造預設的工作者代理程式組態檔案。

最後，我們建議您在部署軟體並如預期運作之後，為工作者代理程式啟用自動關閉。這可讓工作者機群在需要時擴展，並在任務完成時關閉。自動擴展有助於確保您僅使用所需的資源。若要讓自動擴展群組啟動的執行個體關閉，您必須將 shutdown_on_stop=true新增至worker.toml組態檔案。

啟用自動關閉

身為 **root** 使用者：

- 使用參數 安裝工作者代理程式**--allow-shutdown**。

  Linux

  輸入：

  ```
  /opt/deadline/worker/bin/install-deadline-worker \
    --farm-id FARM_ID \
    --fleet-id FLEET_ID \
    --region REGION \
    --allow-shutdown
  ```

  Windows

  輸入：

  ```
  install-deadline-worker ^
    --farm-id FARM_ID ^
    --fleet-id FLEET_ID ^
    --region REGION ^
    --allow-shutdown
  ```

建立任務使用者和群組

本節說明客服人員使用者與佇列上jobRunAsUser定義的 之間所需的使用者和群組關係。

Deadline Cloud 工作者代理程式應以主機上的專用代理程式特定使用者身分執行。您應該設定截止日期雲端佇列的 jobRunAsUser 屬性，以便工作者以特定的作業系統使用者和群組身分執行佇列任務。此組態表示您可以控制任務擁有的共用檔案系統許可。它也為您的任務和工作者代理程式使用者之間提供重要的安全界限。

Linux 任務使用者和群組

若要設定本機工作者代理程式使用者 和 jobRunAsUser，請確定您符合下列要求。如果您使用的是 Linux 可插拔身分驗證模組 (PAM)，例如 Active Directory 或 LDAP，您的程序可能會有所不同。

工作者代理程式使用者和共用jobRunAsUser群組會在您安裝工作者代理程式時設定。預設值為 deadline-worker-agent和 deadline-job-users，但您可以在安裝工作者代理程式時變更這些預設值。

```
install-deadline-worker \
    --user AGENT_USER_NAME \
    --group JOB_USERS_GROUP
```

命令應以根使用者身分執行。

- 每個 jobRunAsUser都應有一個相符的主要群組。使用 adduser命令建立使用者通常會建立相符的主要群組。

  ```
  adduser -r -m jobRunAsUser
  ```

- 的主要群組jobRunAsUser是工作者客服人員使用者的次要群組。共用群組可讓工作者代理程式在任務執行時提供檔案。

  ```
  usermod -a -G jobRunAsUser deadline-worker-agent
  ```

- jobRunAsUser 必須是共用任務群組的成員。

  ```
  usermod -a -G deadline-job-users jobRunAsUser
  ```

- jobRunAsUser 不得屬於工作者代理程式使用者的主要群組。工作者代理程式寫入的敏感檔案由代理程式的主要群組擁有。如果 jobRunAsUser是此群組的一部分，則工作者代理程式檔案可供工作者上執行的任務存取。
- 預設值 AWS 區域 必須符合工作者所屬的陣列區域。這應該套用到工作者上的所有jobRunAsUser帳戶。

```
sudo -u jobRunAsUser aws configure set default.region aws-region
```

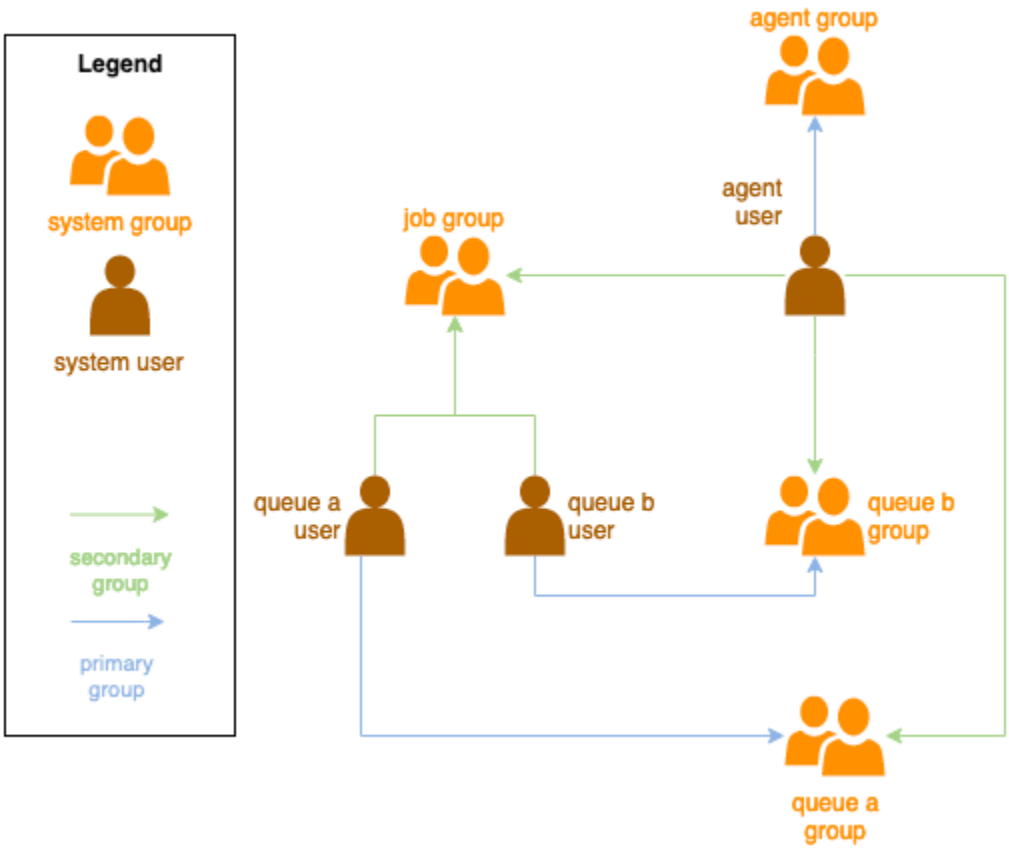
- 工作者代理程式使用者必須能夠以 身分執行sudo命令jobRunAsUser。執行下列命令以開啟編輯器來建立新的 sudoers 規則：

  ```
  visudo -f /etc/sudoers.d/deadline-worker-job-user
  ```

  將下列項目新增至 檔案：

  ```
  # Allows the Deadline Cloud worker agent OS user to run commands
   # as the queue OS user without requiring a password.
  deadline-worker-agent ALL=(jobRunAsUser) NOPASSWD:ALL
  ```

下圖說明代理程式使用者與與機群相關聯佇列jobRunAsUser的使用者和群組之間的關係。



Windows 使用者

若要使用Windows使用者做為 jobRunAsUser，必須符合下列要求：

- 所有佇列jobRunAsUser使用者都必須存在。
- 他們的密碼必須符合佇列JobRunAsUser欄位中指定的秘密值。如需說明，請參閱《截止日期雲端使用者指南》中的截止日期雲端佇列的步驟 7。 AWS
- 代理程式使用者必須能夠以這些使用者身分登入。

保護您的工作者主機

設定工作者主機時，請遵循安全最佳實務來保護敏感資訊，並維護適當的存取控制。

設定日誌資料夾許可

工作者代理程式會撰寫日誌檔案，其中可能包含來自主機組態指令碼和任務執行的敏感資訊。install-deadline-worker 命令會建立具有安全許可的日誌目錄。如果您需要在安裝之前手動建立目錄，請使用下列程序來比對服務受管機群使用的許可：

Linux

在 上設定日誌目錄許可 Linux

1. 建立日誌目錄：

```
sudo mkdir -p /var/log/amazon/deadline
```

2. 將擁有者和群組設定為工作者代理程式使用者：

```
sudo chown -R deadline-worker:deadline-worker /var/log/amazon/deadline
```

3. 將許可設定為 750：

```
sudo chmod -R 750 /var/log/amazon/deadline
```

這些許可可確保只有工作者代理程式使用者和群組可以存取日誌檔案，防止任務使用者和其他未經授權的使用者讀取潛在的敏感資訊。

Windows

在 上設定日誌目錄許可 Windows

1. 開啟管理員 PowerShell 終端機。

2. 建立日誌目錄：

```
New-Item -ItemType Directory -Force -Path "$env:PROGRAMDATA\Amazon\Deadline
\Logs"
```

3. 設定受限 ACLs以僅允許工作者代理程式使用者和管理員：

```
$acl = Get-Acl "$env:PROGRAMDATA\Amazon\Deadline\Logs"
$acl.SetAccessRuleProtection($true, $false)
$acl.Access | ForEach-Object { $acl.RemoveAccessRule($_) }
$agentRule = New-Object
 System.Security.AccessControl.FileSystemAccessRule("deadline-worker",
 "FullControl", "ContainerInherit,ObjectInherit", "None", "Allow")
$adminRule = New-Object
 System.Security.AccessControl.FileSystemAccessRule("Administrators",
 "FullControl", "ContainerInherit,ObjectInherit", "None", "Allow")
$acl.AddAccessRule($agentRule)
$acl.AddAccessRule($adminRule)
Set-Acl "$env:PROGRAMDATA\Amazon\Deadline\Logs" $acl
```

這些命令限制只有工作者代理程式使用者和管理員群組才能存取日誌目錄，防止任務使用者和其他未經授權的使用者讀取潛在的敏感資訊。

管理對Windows任務使用者秘密的存取

當您使用 Windows 設定佇列時`jobRunAsUser`，您必須指定 AWS Secrets Manager 秘密。此秘密的值預期為格式的 JSON 編碼物件：

```
{
    "password": "JOB_USER_PASSWORD"
}
```

若要讓工作者以佇列設定的 身分執行任務`jobRunAsUser`，機群的 IAM 角色必須具有取得秘密值的許可。如果秘密是使用客戶管理的 KMS 金鑰加密，則機群的 IAM 角色也必須具有使用 KMS 金鑰解密的許可。

強烈建議您遵循這些秘密的最低權限原則。這表示擷取佇列 → `jobRunAsUser` → `windows` 之秘密值的存取權`passwordArn`應為：

- 在機群和佇列之間建立佇列機群關聯時，授予機群角色

- 在機群和佇列之間刪除佇列機群關聯時，從機群角色撤銷

此外，在不再使用密碼時，應刪除包含jobRunAsUser密碼的 AWS Secrets Manager 秘密。

授予存取密碼秘密的權限

當佇列和機群相關聯時，截止日期雲端機群需要存取jobRunAsUser存放在佇列密碼秘密中的密碼。建議使用 AWS Secrets Manager 資源政策來授予機群角色的存取權。如果您嚴格遵守此準則，更容易判斷哪些機群角色可存取秘密。

授予對秘密的存取權

1. 開啟 AWS Secret Manager 主控台以存取秘密。
2. 在資源許可區段中，新增表單的政策陳述式：

   JSON

```
{
  "Version":"2012-10-17",
  "Statement" : [
    {
      "Effect" : "Allow",
      "Principal" : {
        "AWS" : "arn:aws:iam::111122223333:role/FleetRole"
      },
      "Action": [
        "secretsmanager:GetSecretValue"
      ],
      "Resource" : "arn:aws:secretsmanager:us-
east-1:111122223333:secret:YourSecretName-ABC123"
    }
  ]
}
```

撤銷對密碼秘密的存取

當機群不再需要存取佇列時，請移除對佇列 密碼秘密的存取jobRunAsUser。建議使用 AWS Secrets Manager 資源政策來授予機群角色的存取權。如果您嚴格遵守此準則，更容易判斷哪些機群角色可存取秘密。

撤銷對秘密的存取

1. 開啟 AWS Secret Manager 主控台以存取秘密。
2. 在資源許可區段中，移除表單的政策陳述式：

   JSON

```
{
  "Version":"2012-10-17",
  "Statement" : [
    {
      "Effect" : "Allow",
      "Principal" : {
        "AWS" : "arn:aws:iam::111122223333:role/FleetRole"
      },
      "Action" : [
        "secretsmanager:GetSecretValue"
      ],
      "Resource" : "arn:aws:secretsmanager:us-
east-1:111122223333:secret:YourSecretName-ABC123"
    }
  ]
}
```

安裝和設定任務所需的軟體

設定截止日期雲端工作者代理程式之後，您可以使用執行任務所需的任何軟體來準備工作者主機。

當您將任務提交至具有相關聯 的佇列時jobRunAsUser，任務會以該使用者身分執行。使用非絕對路徑的命令提交任務時，必須在該使用者的 PATH 中找到該命令。

在 Linux 上，您可以在下列其中一項中PATH為使用者指定 ：

- 他們的 ~/.bashrc或 ~/.bash_profile
- 系統組態檔案，例如 /etc/profile.d/*和 /etc/profile
- shell 啟動指令碼：/etc/bashrc。

在 Windows 上，您可以在下列其中一項中PATH為使用者指定 ：

- 其使用者特定的環境變數
- 全系統環境變數

安裝數位內容建立工具轉接器

Deadline Cloud 提供 OpenJobDescription 轉接器，以使用熱門的數位內容建立 (DCC) 應用程式。若要在客戶受管機群中使用這些轉接器，您必須安裝 DCC 軟體和應用程式轉接器。然後，確保軟體的可執行程式可在系統搜尋路徑（例如，在 PATH環境變數中）上使用。

在客戶受管機群上安裝 DCC 轉接器

1. 開啟終端機。
   a. 在 Linux 上，以root使用者身分開啟終端機（或使用 sudo / su)
   b. 在 上Windows，開啟管理員命令提示字元或 PowerShell 終端機。
2. 安裝截止日期雲端轉接器套件。

```
pip install deadline deadline-cloud-for-maya deadline-cloud-for-nuke deadline-
cloud-for-blender deadline-cloud-for-3ds-max
```

設定 AWS 登入資料

工作者生命週期的初始階段是引導。在此階段，工作者代理程式軟體會在您的機群中建立工作者，並從機群的角色取得 AWS 登入資料以供進一步操作。

AWS credentials for Amazon EC2

為具有截止日期雲端工作者主機許可的 Amazon EC2 建立 IAM 角色

1. 前往 https://console.aws.amazon.com/iam/ 開啟 IAM 主控台。
2. 在導覽窗格中，選擇導覽窗格中的角色，然後選擇建立角色。
3. 選取 AWS 服務。
4. 選取 EC2 做為服務或使用案例，然後選取下一步。
5. 若要授予必要的許可，請連接 AWSDeadlineCloud-WorkerHost AWS 受管政策。

On-premises AWS credentials

您的現場部署工作者會使用登入資料來存取截止日期雲端。為了獲得最安全的存取，我們建議您使用 IAM Roles Anywhere 來驗證您的工作者。如需詳細資訊，請參閱 IAM Roles Anywhere。

針對測試，您可以使用 IAM 使用者存取金鑰做為 AWS 登入資料。我們建議您透過包含限制性內嵌政策來設定 IAM 使用者的過期。

> **Important**
>
> 請注意下列警告：
>
> - 請勿使用您帳戶的根登入資料來存取 AWS 資源。這些登入資料可讓未管制的帳戶存取和很難撤銷這些帳戶。
> - 請勿在應用程式檔案中放置常值存取金鑰或憑證資訊。如果您不小心這麼做了，則會有暴露您登入資料的風險，例如，當您上傳專案到公有儲存庫時。
> - 請勿在您的專案區域中放入包含憑證的檔案。
> - 保護您的存取金鑰。請勿將您的存取金鑰提供給未經授權的當事方，即便是協助尋找您的帳戶識別符也不妥。如果這麼做，就可能會讓他人能夠永久存取您的帳戶。
> - 請注意，存放在共用 AWS 登入資料檔案中的任何登入資料都會以純文字儲存。

如需詳細資訊，請參閱《AWS 一般參考》中的管理 AWS 存取金鑰的最佳實務。

建立 IAM 使用者

1. 前往 https://console.aws.amazon.com/iam/ 開啟 IAM 主控台。
2. 在導覽窗格中，選取使用者，然後選取建立使用者。
3. 為使用者命名。清除提供使用者存取權 AWS 管理主控台的核取方塊，然後選擇下一步。
4. 選擇直接連接政策。
5. 從許可政策清單中，選擇 AWSDeadlineCloud-WorkerHost 政策，然後選擇下一步。
6. 檢閱使用者詳細資訊，然後選擇建立使用者。

限制使用者對有限時段的存取

您建立的任何 IAM 使用者存取金鑰都是長期憑證。為了確保這些憑證在處理不當時過期，您可以建立內嵌政策，指定金鑰不再有效的日期，讓這些憑證有時間限制。

1. 開啟您剛才建立的 IAM 使用者。在許可索引標籤中，選擇新增許可，然後選擇建立內嵌政策。
2. 在 JSON 編輯器中，指定下列許可。若要使用此政策，請以您自己的`aws:CurrentTime`時間和日期取代範例政策中的時間戳記值。

   JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Deny",
      "Action": "*",
      "Resource": "*",
      "Condition": {
        "DateGreaterThan": {
          "aws:CurrentTime": "2024-01-01T00:00:00Z"
        }
      }
    }
  ]
}
```

建立存取金鑰

1. 在使用者詳細資訊頁面上，選取安全登入資料索引標籤。在存取金鑰區段中，選擇建立存取金鑰。
2. 表示您想要使用其他金鑰，然後選擇下一步，然後選擇建立存取金鑰。
3. 在擷取存取金鑰頁面上，選擇顯示以顯示使用者私密存取金鑰的值。您可以複製憑證或下載 .csv 檔案。

存放使用者存取金鑰

- 將使用者存取金鑰存放在工作者主機系統的代理程式使用者的 AWS 登入資料檔案中：
  - 在 上Linux，檔案位於 `~/.aws/credentials`
  - 在 上Windows，檔案位於 `%USERPROFILE\.aws\credentials`

取代下列金鑰：

```
[default]
aws_access_key_id=ACCESS_KEY_ID
aws_secret_access_key=SECRET_ACCESS_KEY
```

> **Important**
>
> 當您不再需要此 IAM 使用者時，建議您將其移除，以符合AWS 安全最佳實務。我們建議您要求人類使用者在存取AWS IAM Identity Center時透過 使用臨時登入資料 AWS。

客戶受管機群的工作者託管資料流程

本主題說明 AWS 截止日期雲端 （ 截止日期雲端 ） 工作者在操作期間所建立的網路連線，包括已聯絡的端點、使用的通訊協定和傳輸的資料。此資訊適用於客戶受管機群 (CMF) 工作者，包括 Amazon Elastic Compute Cloud (Amazon EC2) 執行個體和內部部署工作者。使用此資訊來設定防火牆規則、建立 VPC 端點、執行安全稽核或規劃工作者主機的網路政策。如需服務受管機群聯網的相關資訊，請參閱 網際網路流量隱私權。

所有工作者通訊都是傳出通訊。工作者主機會啟動所有連線，您不需要允許任何傳入連線。所有連線都透過連接埠 443 使用 HTTPS (TLS 1.2 或更新版本）。

本主題包含下列章節：

- the section called “端點和通訊協定”
- the section called “工作者使用的 API 操作”
- the section called “傳輸的其他資料”
- the section called “私有連線選項”

端點和通訊協定

下表列出工作者主機在操作期間連線 AWS 的服務端點。如需每個服務的區域端點完整清單，請參閱《AWS 一般參考》中的服務端點和配額。

工作者主機端點參考

AWS 服務	Endpoint	連接埠/通訊協定	用途	必要
截止日期雲端（排程）	scheduling.deadline. *[Region]*.amazonaws.com	443 / HTTPS	工作者註冊、任務輪詢、狀態更新、登入資料交換、任務實體擷取。請參閱 the section called “工作者使用的 API 操作”。	一律
Amazon CloudWatch Logs (CloudWatch Logs)	logs.*[Region]*.amazonaws.com	443 / HTTPS	工作者代理程式和工作階段日誌交付。	一律
Amazon Simple Storage Service (Amazon S3)	s3.*[Region]*.amazonaws.com	443 / HTTPS	任務附件上傳和下載。	如果使用任務附件

如果您的任務使用其他 AWS 服務，您可能還需要允許對這些服務端點的傳出連線。

工作者使用的 API 操作

下列所有 API 操作都會使用 scheduling.deadline.*[Region]*.amazonaws.com端點。如需每個操作的完整請求和回應結構描述，請參閱截止日期雲端 API 參考。

引導階段

當工作者主機啟動時，工作者代理程式會向機群註冊。引導登入資料需要 AWSDeadlineCloud-WorkerHost AWS 受管政策中的許可，或同等的自訂許可。引導階段使用以下 API 操作：

- CreateWorker – 向機群註冊工作者。傳送主機名稱和 IP 地址。接收工作者 ID。
- AssumeFleetRoleForWorker – 取得機群角色登入資料。接收工作者代理程式用於後續操作的臨時 AWS 登入資料。

操作階段

引導後，工作者代理程式會輪詢工作和處理工作階段。機群角色需要AWSDeadlineCloud-FleetWorker AWS 受管政策中的許可或同等自訂許可，並使用下列 API 操作：

- UpdateWorker – 更新工作者狀態，例如在關機STOPPED期間將 更新為 。
- UpdateWorkerSchedule – 工作指派的輪詢。傳送工作階段動作狀態更新，包括完成狀態、進度百分比、進度訊息和輸出資訊清單雜湊。接收指派的工作階段（任務 ID、佇列 ID、工作階段動作、日誌組態）、取消請求、所需的工作者狀態，以及更新間隔。
- BatchGetJobEntity – 擷取指派工作的工作詳細資訊。傳送任務實體識別符。接收任務詳細資訊、環境詳細資訊和任務附件詳細資訊。
- AssumeFleetRoleForWorker – 定期重新整理機群角色登入資料。
- AssumeQueueRoleForWorker – 取得範圍限定於特定佇列的佇列角色登入資料。工作者使用這些登入資料來存取 Amazon S3 中的任務附件。

傳輸的其他資料

除了截止日期雲端排程 API 操作之外，工作者主機還會將下列資料傳輸至其他 AWS 服務：

日誌資料

工作者代理程式會使用 PutLogEvents API 操作，將工作者代理程式日誌和工作階段日誌（來自任務程序的 Stdout 和 stderr) 傳送至 CloudWatch Logs。

任務附件

工作者使用 和 PutObject API 操作透過 Amazon S3 傳輸輸入GetObject和輸出檔案。工作者會使用透過 取得AssumeQueueRoleForWorker的佇列角色登入資料進行此存取。

遙測（選用）

工作者代理程式會傳送操作指標，例如當機報告。您可以選擇不接收遙測集合。如需詳細資訊，請參閱選擇退出。

私有連線選項

您可以使用 AWS PrivateLink 在您的 VPC 中保留 CMF 工作者主機和截止日期雲端之間的流量，而無需周遊公有網際網路。對於現場部署工作者，您可以結合 AWS PrivateLink AWS Direct Connect

(Direct Connect) 或 VPN 連接。如需詳細資訊，請參閱AWS Deadline Cloud 使用界面端點存取 (AWS PrivateLink)。

測試工作者主機的組態

安裝工作者代理程式、安裝處理任務所需的軟體，並設定工作者代理程式的 AWS 登入資料後，您應該在AMI為機群建立 之前，先測試安裝是否可以處理您的任務。您應該測試下列項目：

- Deadline Cloud 工作者代理程式已正確設定為做為系統服務執行。
- 工作者輪詢相關聯的佇列以進行工作。
- 工作者成功處理傳送至與機群相關聯佇列的任務。

測試組態並成功處理代表性任務後，您可以使用設定的工作者為 AMI Amazon EC2 工作者建立 ，或作為現場部署工作者的模型。

> **Note**
>
> 如果您正在測試自動擴展機群的工作者主機組態，在下列情況下，您可能會難以測試工作者：
>
> - 如果佇列中沒有工作，Deadline Cloud 會在工作者啟動後立即停止工作者代理程式。
> - 如果工作者代理程式設定為在停止時關閉主機，則如果佇列中沒有工作，代理程式會關閉機器。
>
> 為了避免這些問題，請使用不會自動擴展的預備機群來設定和測試您的工作者。測試工作者主機之後，請務必先設定正確的機群 ID，再製作 AMI。

測試工作者主機組態

1. 啟動作業系統服務來執行工作者代理程式。

   Linux

   從根 shell 執行下列命令：

   ```
   systemctl start deadline-worker
   ```

Windows

從管理員命令提示字元或PowerShell終端機中，輸入下列命令：

```
sc.exe start DeadlineWorker
```

2. 監控工作者，以確保其啟動並輪詢工作。

Linux

從根 shell 執行下列命令：

```
systemctl status deadline-worker
```

命令應傳回如下的回應：

```
Active: active (running) since Wed 2023-06-14 14:44:27 UTC; 7min ago
```

如果回應看起來不像這樣，請使用下列命令檢查日誌檔案：

```
tail -n 25 /var/log/amazon/deadline/worker-agent.log
```

Windows

從管理員命令提示字元或PowerShell終端機中，輸入下列命令：

```
sc.exe query DeadlineWorker
```

命令應傳回如下的回應：

```
STATE   : 4 RUNNING
```

如果回應不包含 RUNNING，請檢查工作者日誌檔案。開啟 和管理員PowerShell提示，並執行下列命令：

```
Get-Content -Tail 25 -Path $env:PROGRAMDATA\Amazon\Deadline\Logs\worker-
agent.log
```

3. 將任務提交至與機群相關聯的佇列。任務應該代表機群處理的任務。

4. 使用截止日期雲端監視器或 CLI 監控任務進度。如果任務失敗，請檢查工作階段和工作者日誌。
5. 視需要更新工作者主機的組態，直到任務成功完成為止。
6. 當測試任務成功時，您可以停止工作者：

   Linux

   從根 shell 執行下列命令：

   ```
   systemctl stop deadline-worker
   ```

   Windows

   從管理員命令提示字元或PowerShell終端機中，輸入下列命令：

   ```
   sc.exe stop DeadlineWorker
   ```

建立 Amazon Machine Image

若要建立要在 Amazon Elastic Compute Cloud (Amazon EC2AMI) 客戶受管機群 Amazon Machine Image(CMF) 中使用的 ()，請完成本節中的任務。您必須建立 Amazon EC2 執行個體，才能繼續。如需詳細資訊，請參閱《Amazon EC2 Linux 執行個體使用者指南》中的啟動執行個體。 Amazon EC2

> **Important**
>
> 建立 會AMI建立 Amazon EC2 執行個體連接磁碟區的快照。執行個體上安裝的任何軟體都會持續存在，因此執行個體會在您從 啟動執行個體時重複使用AMI。建議您採用修補策略，並在套用到機群之前定期更新任何AMI具有更新軟體的新 。

準備 Amazon EC2 執行個體

在建置 之前AMI，您必須先刪除工作者狀態。工作者代理程式啟動之間會維持工作者狀態。如果此狀態持續存在 AMI，則從中啟動的所有執行個體都會共用相同的狀態。

我們也建議您刪除任何現有的日誌檔案。當您準備 AMI 時，日誌檔案可以保留在 Amazon EC2 執行個體上。在診斷使用 AMI 的工作者機群中可能發生的問題時，刪除這些檔案可將混淆降到最低。

您也應該啟用工作者代理程式系統服務，以便在 Amazon EC2 啟動時啟動截止日期雲端工作者代理程式。

最後，我們建議您啟用工作者代理程式自動關閉。這可讓工作者機群在需要時擴展，並在轉譯任務完成時關閉。此自動擴展有助於確保您僅在需要時使用資源。

準備 Amazon EC2 執行個體

1. 開啟 Amazon EC2 主控台。
2. 啟動 Amazon EC2 執行個體。如需詳細資訊，請參閱啟動執行個體。
3. 設定主機以連線至您的身分提供者 (IdP)，然後掛載其所需的任何共用檔案系統。
4. 遵循教學課程至 安裝截止日期雲端工作者代理程式，然後是 設定工作者代理程式和 建立任務使用者和群組。
5. 如果您要準備AMI以 Amazon Linux 2023 為基礎的 來執行與 VFX 參考平台相容的軟體，則需要更新數個需求。如需詳細資訊，請參閱AWS 《截止日期雲端使用者指南》中的 VFX 參考平台相容性。
6. 開啟終端機。
   a. 在 Linux 上，以`root`使用者身分開啟終端機 （ 或使用 `sudo` / `su`)
   b. 在 上Windows，開啟管理員命令提示字元或 PowerShell 終端機。
7. 確保工作者服務未執行，並設定為在開機時啟動：
   a. 在 Linux 上執行

```
systemctl stop deadline-worker
systemctl enable deadline-worker
```

   b. 在 Windows上執行

```
sc.exe stop DeadlineWorker
sc.exe config DeadlineWorker start= auto
```

8. 刪除工作者狀態。
   a. 在 Linux 上執行

```
rm -rf /var/lib/deadline/*
```

   b. 在 Windows上執行

```
del /Q /S %PROGRAMDATA%\Amazon\Deadline\Cache\*
```

9. 刪除日誌檔案。

   a. 在 Linux 上執行

   ```
   rm -rf /var/log/amazon/deadline/*
   ```

   b. 在 Windows上執行

   ```
   del /Q /S %PROGRAMDATA%\Amazon\Deadline\Logs\*
   ```

10. 在 上Windows，建議執行在開始選單中找到的 Amazon EC2Launch 設定應用程式，以完成執行個體的最終主機準備和關閉。

> **Note**
>
> 您必須選擇不使用 Sysprep 的關機，絕不選擇使用 Sysprep 的關機。使用 Sysprep 關閉會導致所有本機使用者無法使用。如需詳細資訊，請參閱 Windows 執行個體使用者指南中建立自訂 AMI 主題的開始之前一節。

建置 AMI

若要建置 AMI

1. 開啟 Amazon EC2 主控台。
2. 在導覽窗格中選取執行個體，然後選取您的執行個體。
3. 選擇執行個體狀態，然後選擇停止執行個體。
4. 執行個體停止後，選擇動作。
5. 選擇映像和範本，然後選擇建立映像。
6. 輸入映像名稱。
7. （選用）輸入影像的描述。
8. 選擇 Create image (建立映像)。

使用 Amazon EC2 Auto Scaling 群組建立機群基礎設施

本節說明如何建立 Amazon EC2 Auto Scaling 機群。

使用以下 CloudFormation YAML 範本建立 Amazon EC2 Auto Scaling (Auto Scaling) 群組、具有兩個子網路的 Amazon Virtual Private Cloud (Amazon VPC)、執行個體描述檔和執行個體存取角色。這些是在子網路中使用 Auto Scaling 啟動執行個體時需要。

您應該檢閱並更新執行個體類型的清單，以符合您的轉譯需求。

如需 CloudFormation YAML 範本中使用的資源和參數的完整說明，請參閱AWS CloudFormation 《使用者指南》中的截止日期雲端資源類型參考。

建立 Amazon EC2 Auto Scaling 機群

1. 使用下列範例來建立定義 `FarmID`、 `FleetID`和 `AMIId` 參數的 CloudFormation 範本。將範本儲存至本機電腦上`.YAML`的檔案。

```
AWSTemplateFormatVersion: 2010-09-09
Description: Amazon Deadline Cloud customer-managed fleet
Parameters:
  FarmId:
    Type: String
    Description: Farm ID
  FleetId:
    Type: String
    Description: Fleet ID
  AMIId:
    Type: String
    Description: AMI ID for launching workers
Resources:
  deadlineVPC:
    Type: 'AWS::EC2::VPC'
    Properties:
      CidrBlock: 100.100.0.0/16
  deadlineWorkerSecurityGroup:
    Type: 'AWS::EC2::SecurityGroup'
    Properties:
      GroupDescription: !Join
        - ' '
        - - Security group created for Deadline Cloud workers in the fleet
          - !Ref FleetId
      GroupName: !Join
```

```
        - ''
        - - deadlineWorkerSecurityGroup-
          - !Ref FleetId
      SecurityGroupEgress:
        - CidrIp: 0.0.0.0/0
          IpProtocol: '-1'
      SecurityGroupIngress: []
      VpcId: !Ref deadlineVPC
  deadlineIGW:
    Type: 'AWS::EC2::InternetGateway'
    Properties: {}
  deadlineVPCGatewayAttachment:
    Type: 'AWS::EC2::VPCGatewayAttachment'
    Properties:
      VpcId: !Ref deadlineVPC
      InternetGatewayId: !Ref deadlineIGW
  deadlinePublicRouteTable:
    Type: 'AWS::EC2::RouteTable'
    Properties:
      VpcId: !Ref deadlineVPC
  deadlinePublicRoute:
    Type: 'AWS::EC2::Route'
    Properties:
      RouteTableId: !Ref deadlinePublicRouteTable
      DestinationCidrBlock: 0.0.0.0/0
      GatewayId: !Ref deadlineIGW
    DependsOn:
      - deadlineIGW
      - deadlineVPCGatewayAttachment
  deadlinePublicSubnet0:
    Type: 'AWS::EC2::Subnet'
    Properties:
      VpcId: !Ref deadlineVPC
      CidrBlock: 100.100.16.0/22
      AvailabilityZone: !Join
        - ''
        - - !Ref 'AWS::Region'
          - a
  deadlineSubnetRouteTableAssociation0:
    Type: 'AWS::EC2::SubnetRouteTableAssociation'
    Properties:
      RouteTableId: !Ref deadlinePublicRouteTable
      SubnetId: !Ref deadlinePublicSubnet0
  deadlinePublicSubnet1:
```

```
    Type: 'AWS::EC2::Subnet'
    Properties:
      VpcId: !Ref deadlineVPC
      CidrBlock: 100.100.20.0/22
      AvailabilityZone: !Join
        - ''
        - - !Ref 'AWS::Region'
          - c
  deadlineSubnetRouteTableAssociation1:
    Type: 'AWS::EC2::SubnetRouteTableAssociation'
    Properties:
      RouteTableId: !Ref deadlinePublicRouteTable
      SubnetId: !Ref deadlinePublicSubnet1
  deadlineInstanceAccessAccessRole:
    Type: 'AWS::IAM::Role'
    Properties:
      RoleName: !Join
        - '-'
        - - deadline
          - InstanceAccess
          - !Ref FleetId
      AssumeRolePolicyDocument:
        Statement:
          - Effect: Allow
            Principal:
              Service: ec2.amazonaws.com
            Action:
              - 'sts:AssumeRole'
      Path: /
      ManagedPolicyArns:
        - 'arn:aws:iam::aws:policy/CloudWatchAgentServerPolicy'
        - 'arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore'
        - 'arn:aws:iam::aws:policy/AWSDeadlineCloud-WorkerHost'
  deadlineInstanceProfile:
    Type: 'AWS::IAM::InstanceProfile'
    Properties:
      Path: /
      Roles:
        - !Ref deadlineInstanceAccessAccessRole
  deadlineLaunchTemplate:
    Type: 'AWS::EC2::LaunchTemplate'
    Properties:
      LaunchTemplateName: !Join
        - ''
```

```
        - - deadline-LT-
          - !Ref FleetId
      LaunchTemplateData:
        NetworkInterfaces:
          - DeviceIndex: 0
            AssociatePublicIpAddress: true
            Groups:
              - !Ref deadlineWorkerSecurityGroup
            DeleteOnTermination: true
        ImageId: !Ref AMIId
        InstanceInitiatedShutdownBehavior: terminate
        IamInstanceProfile:
          Arn: !GetAtt
            - deadlineInstanceProfile
            - Arn
        MetadataOptions:
          HttpTokens: required
          HttpEndpoint: enabled

  deadlineAutoScalingGroup:
    Type: 'AWS::AutoScaling::AutoScalingGroup'
    Properties:
      AutoScalingGroupName: !Join
        - ''
        - - deadline-ASG-autoscalable-
          - !Ref FleetId
      MinSize: 0
      MaxSize: 10
      VPCZoneIdentifier:
        - !Ref deadlinePublicSubnet0
        - !Ref deadlinePublicSubnet1
      NewInstancesProtectedFromScaleIn: true
      MixedInstancesPolicy:
        InstancesDistribution:
          OnDemandBaseCapacity: 0
          OnDemandPercentageAboveBaseCapacity: 0
          SpotAllocationStrategy: capacity-optimized
          OnDemandAllocationStrategy: lowest-price
        LaunchTemplate:
          LaunchTemplateSpecification:
            LaunchTemplateId: !Ref deadlineLaunchTemplate
            Version: !GetAtt
              - deadlineLaunchTemplate
              - LatestVersionNumber
```

```
        Overrides:
          - InstanceType: m5.large
          - InstanceType: m5d.large
          - InstanceType: m5a.large
          - InstanceType: m5ad.large
          - InstanceType: m5n.large
          - InstanceType: m5dn.large
          - InstanceType: m4.large
          - InstanceType: m3.large
          - InstanceType: r5.large
          - InstanceType: r5d.large
          - InstanceType: r5a.large
          - InstanceType: r5ad.large
          - InstanceType: r5n.large
          - InstanceType: r5dn.large
          - InstanceType: r4.large
    MetricsCollection:
      - Granularity: 1Minute
        Metrics:
          - GroupMinSize
          - GroupMaxSize
          - GroupDesiredCapacity
          - GroupInServiceInstances
          - GroupTotalInstances
          - GroupInServiceCapacity
          - GroupTotalCapacity
```

2. 在 https：//https://console.aws.amazon.com/cloudformation 開啟 CloudFormation 主控台。

   使用 CloudFormation 主控台，使用上傳您建立之範本檔案的指示來建立堆疊。如需詳細資訊，請參閱AWS CloudFormation 《 使用者指南》中的在 CloudFormation 主控台上建立堆疊。

> **Note**
>
> - 連接至工作者 Amazon EC2 執行個體的 IAM 角色登入資料可供該工作者上執行的所有程序使用，其中包括任務。工作者應擁有最低操作權限： `deadline:CreateWorker`和 `deadline:AssumeFleetRoleForWorker.`
> - 工作者代理程式會取得佇列角色的登入資料，並將其設定為執行任務來使用。Amazon EC2 執行個體描述檔角色不應包含任務所需的許可。

使用截止日期雲端擴展建議功能自動擴展 Amazon EC2 機群

Deadline Cloud 利用 Amazon EC2 Auto Scaling (Auto Scaling) 群組自動擴展 Amazon EC2 客戶受管機群 (CMF)。您需要設定機群模式，並在帳戶中部署所需的基礎設施，讓機群自動擴展。您部署的基礎設施適用於所有機群，因此您只需要設定一次。

基本工作流程是：您可以將機群模式設定為自動擴展，然後當建議的機群大小變更 （ 一個事件包含機群 ID、建議的機群大小和其他中繼資料 ） 時，Deadline Cloud 會傳送該機群的 EventBridge 事件。您將擁有 EventBridge 規則來篩選相關事件，並擁有 Lambda 來使用它們。Lambda 將與 Amazon EC2 Auto Scaling 整合`AutoScalingGroup`，以自動擴展 Amazon EC2 機群。

將機群模式設定為 **EVENT_BASED_AUTO_SCALING**

將您的機群模式設定為 `EVENT_BASED_AUTO_SCALING`。您可以使用 主控台來執行此操作，或使用 AWS CLI 直接呼叫 `CreateFleet`或 `UpdateFleet` API。設定模式之後，當建議的機群大小變更時，Deadline Cloud 會開始傳送 EventBridge 事件。

- 範例`UpdateFleet`命令：

```
aws deadline update-fleet \
  --farm-id FARM_ID \
  --fleet-id FLEET_ID \
  --configuration file://configuration.json
```

- 範例`CreateFleet`命令：

```
aws deadline create-fleet \
  --farm-id FARM_ID \
  --display-name "Fleet name" \
  --max-worker-count 10 \
  --configuration file://configuration.json
```

以下是上述 CLI 命令`configuration.json`中使用的範例 (`--configuration file://configuration.json`)。

- 若要在機群上啟用 Auto Scaling，您應該將 模式設定為 `EVENT_BASED_AUTO_SCALING`。
- `workerCapabilities` 是建立 CMF 時指派給 CMF 的預設值。如果您需要增加 CMF 可用的資源，您可以變更這些值。

設定機群模式後，截止日期雲端會開始發出該機群的機群大小建議事件。

```
{
    "customerManaged": {
        "mode": "EVENT_BASED_AUTO_SCALING",
        "workerCapabilities": {
            "vCpuCount": {
                "min": 1,
                "max": 4
            },
            "memoryMiB": {
                "min": 1024,
                "max": 4096
            },
            "osFamily": "linux",
            "cpuArchitectureType": "x86_64"
        }
    }
}
```

使用 CloudFormation 範本部署 Auto Scaling 堆疊

您可以設定 EventBridge 規則來篩選事件、設定 Lambda 來取用事件並控制 Auto Scaling，以及設定 SQS 佇列來存放未處理的事件。使用下列 CloudFormation 範本來部署堆疊中的所有項目。成功部署資源後，您可以提交任務，機群會自動擴展。

```
Resources:
  AutoScalingLambda:
    Type: 'AWS::Lambda::Function'
    Properties:
      Code:
        ZipFile: |-
          """
          This lambda is configured to handle "Fleet Size Recommendation Change"
          messages. It will handle all such events, and requires
          that the ASG is named based on the fleet id. It will scale up/down the fleet
          based on the recommended fleet size in the message.

          Example EventBridge message:
          {
              "version": "0",
              "id": "6a7e8feb-b491-4cf7-a9f1-bf3703467718",
              "detail-type": "Fleet Size Recommendation Change",
```

```
            "source": "aws.deadline",
            "account": "111122223333",
            "time": "2017-12-22T18:43:48Z",
            "region": "us-west-1",
            "resources": [],
            "detail": {
                "farmId": "farm-12345678900000000000000000000000",
                "fleetId": "fleet-12345678900000000000000000000000",
                "oldFleetSize": 1,
                "newFleetSize": 5,
            }
        }
        """

        import json
        import boto3
        import logging

        logger = logging.getLogger()
        logger.setLevel(logging.INFO)

        auto_scaling_client = boto3.client("autoscaling")

        def lambda_handler(event, context):
            logger.info(event)
            event_detail = event["detail"]
            fleet_id = event_detail["fleetId"]
            desired_capacity = event_detail["newFleetSize"]

            asg_name = f"deadline-ASG-autoscalable-{fleet_id}"
            auto_scaling_client.set_desired_capacity(
                AutoScalingGroupName=asg_name,
                DesiredCapacity=desired_capacity,
                HonorCooldown=False,
            )

            return {
                'statusCode': 200,
                'body': json.dumps(f'Successfully set desired_capacity for {asg_name}
to {desired_capacity}')
            }
    Handler: index.lambda_handler
    Role: !GetAtt
      - AutoScalingLambdaServiceRole
```

```
        - Arn
      Runtime: python3.11
    DependsOn:
      - AutoScalingLambdaServiceRoleDefaultPolicy
      - AutoScalingLambdaServiceRole
  AutoScalingEventRule:
    Type: 'AWS::Events::Rule'
    Properties:
      EventPattern:
        source:
          - aws.deadline
        detail-type:
          - Fleet Size Recommendation Change
      State: ENABLED
      Targets:
        - Arn: !GetAtt
            - AutoScalingLambda
            - Arn
          DeadLetterConfig:
            Arn: !GetAtt
              - UnprocessedAutoScalingEventQueue
              - Arn
          Id: Target0
          RetryPolicy:
            MaximumRetryAttempts: 15
  AutoScalingEventRuleTargetPermission:
    Type: 'AWS::Lambda::Permission'
    Properties:
      Action: 'lambda:InvokeFunction'
      FunctionName: !GetAtt
        - AutoScalingLambda
        - Arn
      Principal: events.amazonaws.com
      SourceArn: !GetAtt
        - AutoScalingEventRule
        - Arn
  AutoScalingLambdaServiceRole:
    Type: 'AWS::IAM::Role'
    Properties:
      AssumeRolePolicyDocument:
        Statement:
          - Action: 'sts:AssumeRole'
            Effect: Allow
            Principal:
```

```
            Service: lambda.amazonaws.com
        Version: 2012-10-17
      ManagedPolicyArns:
        - !Join
          - ''
          - - 'arn:'
            - !Ref 'AWS::Partition'
            - ':iam::aws:policy/service-role/AWSLambdaBasicExecutionRole'
  AutoScalingLambdaServiceRoleDefaultPolicy:
    Type: 'AWS::IAM::Policy'
    Properties:
      PolicyDocument:
        Statement:
          - Action: 'autoscaling:SetDesiredCapacity'
            Effect: Allow
            Resource: '*'
        Version: 2012-10-17
      PolicyName: AutoScalingLambdaServiceRoleDefaultPolicy
      Roles:
        - !Ref AutoScalingLambdaServiceRole
  UnprocessedAutoScalingEventQueue:
    Type: 'AWS::SQS::Queue'
    Properties:
      QueueName: deadline-unprocessed-autoscaling-events
    UpdateReplacePolicy: Delete
    DeletionPolicy: Delete
  UnprocessedAutoScalingEventQueuePolicy:
    Type: 'AWS::SQS::QueuePolicy'
    Properties:
      PolicyDocument:
        Statement:
          - Action: 'sqs:SendMessage'
            Condition:
              ArnEquals:
                'aws:SourceArn': !GetAtt
                  - AutoScalingEventRule
                  - Arn
            Effect: Allow
            Principal:
              Service: events.amazonaws.com
            Resource: !GetAtt
              - UnprocessedAutoScalingEventQueue
              - Arn
        Version: 2012-10-17
```

```
      Queues:
        - !Ref UnprocessedAutoScalingEventQueue
```

執行機群運作狀態檢查

建立機群之後，您應該建立自訂運作狀態檢查，以確保您的機群保持運作狀態良好且沒有停滯的執行個體，以協助避免不必要的成本。請參閱在 GitHub 上部署截止日期雲端機群運作狀態檢查。 GitHub 運作狀態檢查可以降低意外變更 Amazon Machine Image、啟動範本或執行未偵測到的網路組態的風險。

設定和使用截止日期雲端服務受管機群

服務受管機群 (SMF) 是由 Deadline Cloud 管理的工作者集合。SMF 無需針對處理需求管理機群擴展，或在任務完成後減少機群大小。

當 SMF 與使用預設 conda 佇列環境的佇列相關聯時，Deadline Cloud 會使用適當的軟體套件來設定機群中的工作者。如需支援的合作夥伴應用程式，請參閱AWS 《截止日期雲端使用者指南》中的預設 conda 佇列環境。

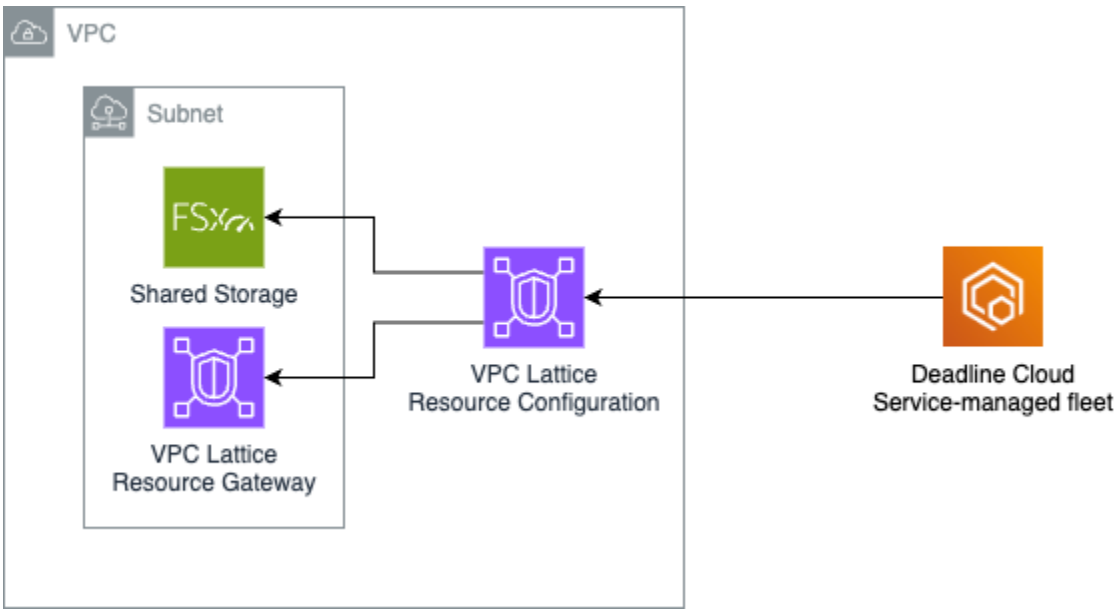
在大多數情況下，您不需要變更 SMF 來處理工作負載。不過，在某些情況下，您可能需要變更機群。

> **Note**
>
> 若要使用主機組態指令碼在工作者上安裝自訂軟體，請參閱 執行具有管理員權限的主機組態指令碼。

主題

使用 VPC 資源端點將 VPC 資源連接至您的 SMF

透過適用於 Deadline Cloud 服務受管機群 (SMF) 的 Amazon VPC 資源端點，您可以將網路檔案系統 (NFS)、授權伺服器和資料庫等 VPC 資源與您的 Deadline Cloud 工作者連線。此功能可讓您利用 Deadline Cloud 的全受管平台，同時與 VPC 中的現有基礎設施整合。



> **Tip**
>
> 如需設定 Amazon FSx 叢集並將其連接到服務受管機群的參考 CloudFormation 範本，請參閱 GitHub 上截止日期雲端範例儲存庫中的 smf_vpc_fsx。

VPC 資源端點的運作方式

VPC 資源端點使用 VPC Lattice 在您的最終雲端 SMF 工作者與 VPC 中的資源之間建立安全連線。連線是單向的，這表示工作者可以建立與 VPC 資源的連線，並來回傳輸資料，但 VPC 中的資源無法與工作者建立連線。

當您將 VPC 資源連線至截止日期雲端服務受管機群時，工作者可以使用私有網域名稱存取 VPC 中的資源。此外，流量會透過 VPC Lattice 從工作者流向 VPC 資源，而 VPC 中的資源也會看到來自 VPC Lattice 資源閘道的流量。

若要進一步了解，請參閱 VPC Lattice 使用者指南。

先決條件

將 VPC 資源連線至截止日期雲端服務受管機群之前，請確定您有下列項目：

- 具有 VPC AWS 的帳戶，其中包含您要連線的資源。
- 建立和管理 VPC Lattice 資源的 IAM 許可。
- 具有至少一個服務受管機群的截止日期雲端陣列。

- 您想要讓 存取的 VPC 資源 (FSx、NFS、授權伺服器等）。

設定 VPC 資源端點

若要設定 VPC 資源端點，您需要在 VPC Lattice 和 中建立資源AWS RAM，然後將這些資源連接到截止日期雲端中的機群。若要為您的 SMF 設定 VPC 資源端點，請完成下列步驟。

1. 若要在 VPC Lattice 中建立資源閘道，請參閱 VPC Lattice 使用者指南中的建立資源閘道。
2. 若要在 VPC Lattice 中建立資源組態，請參閱 VPC Lattice 使用者指南中的建立資源組態。
3. 若要與截止日期雲端機群共用資源，請在其中建立資源共用 AWS RAM。如需說明，請參閱建立資源共享。

   建立資源共享時，針對委託人，從下拉式清單中選取服務委託人，然後輸入 **`fleets.deadline.amazonaws.com`**。
4. 若要將資源組態與您的截止日期雲端機群連線，請完成下列步驟。

   a. 如果您還沒有，請開啟截止日期雲端主控台。
   b. 在導覽窗格中，選擇陣列，然後選取您的陣列。
   c. 選擇機群索引標籤，然後選取您的機群。
   d. 選擇 Configuration (組態) 標籤。
   e. 在 VPC 資源端點下，選擇編輯。
   f. 選取您建立的資源組態，然後選擇儲存變更。

存取您的 VPC 資源

將您的 VPC 資源連線至機群後，工作者可以使用以下格式的私有網域名稱來存取它：
`<resource_config_id>.resource-endpoints.deadline.<region>.amazonaws.com`

此網域是私有的，只有工作者才能存取（而不是從網際網路或工作站存取）。

若要在工作者上掛載或設定對 VPC 資源的存取，請使用主機組態指令碼。主機組態指令碼會在工作者開始時以管理員權限執行，可讓您掛載檔案系統、設定網路設定，或執行其他設定任務。

身分驗證和安全性

對於需要身分驗證的資源，請將登入資料安全地存放在 AWS Secrets Manager 中，從您的主機組態指令碼或任務指令碼存取秘密，並實作適當的檔案系統許可來控制存取。在多個機群之間共用資源時，請

考慮安全問題。例如，如果兩個機群連接到相同的共用儲存體，在一個機群上執行的任務可能可以存取從另一個機群建立的資產。

技術考量事項

使用 VPC 資源端點時，請考慮下列事項：

- 連線只能從工作者啟動至 VPC 資源，不能從 VPC 資源啟動至工作者。
- 建立後，即使資源組態中斷連線，連線仍會持續存在，直到重設為止。
- VPC Lattice 連線會自動處理可用區域之間的連線，無需額外費用。您的資源閘道必須與 VPC 資源共用可用區域，因此我們建議您設定資源閘道以跨所有可用區域。
- 經過 VPC 資源端點的流量使用網路位址轉譯 (NAT)，這與所有使用案例不相容。例如，Microsoft Active Directory (AD) 無法透過 NAT 連線。

如需 VPC Lattice 配額的詳細資訊，請參閱 VPC Lattice 的配額。

疑難排解

如果您遇到 VPC 資源端點的問題，請檢查下列項目。

- 如果您收到錯誤訊息，例如「mount.nfs：掛載時伺服器拒絕存取」，您可能需要更新 NFS 磁碟區的用戶端組態。
- 從 Amazon EC2 執行個體或在 VPC AWS CloudShell 中進行測試，以驗證您的資源組態設定。
- 使用簡單的 CLI 任務測試您的截止日期雲端連線。如需詳細資訊，請參閱 GitHub 上的截止日期雲端範例。
- 如果您遇到連線失敗，請檢查資源閘道安全群組上的設定。
- 啟用 VPC 存取日誌以監控連線。

搭配服務受管機群使用任務附件

任務附件會使用 Amazon Simple Storage Service (Amazon S3) 在您的工作站與截止日期雲端工作者之間傳輸檔案。您可以單獨使用任務附件，或與共用儲存搭配使用，將輔助資料連接到未與其他任務共用的任務，例如本機存放的任務指令碼、組態檔案或專案資產。

如需有關任務附件如何運作的詳細資訊，請參閱《截止日期雲端使用者指南》中的任務附件。如需在任務套件中指定輸入和輸出檔案的詳細資訊，請參閱 使用任務附件來共用檔案。

選擇檔案系統模式

使用附件提交任務時，您可以透過設定 `fileSystem` 屬性，選擇工作者從 Amazon S3 載入檔案的方式：

- COPIED （預設） – 在任務開始之前，將所有檔案下載至本機磁碟。當每個任務需要大部分的輸入檔案時最佳。
- VIRTUAL – 掛載可隨需下載檔案的虛擬檔案系統。當任務只需要輸入檔案的子集時最佳。僅適用於 Linux SMF 工作者。

> **Important**
>
> VIRTUAL 模式快取可能會增加記憶體使用量，而且未針對所有工作負載最佳化。建議您在執行生產任務之前測試工作負載。

如需設定檔案系統模式的詳細資訊，請參閱《截止日期雲端使用者指南》中的虛擬檔案系統。

最佳化傳輸效能

從 Amazon S3 同步檔案到 SMF 工作者的速度取決於機群的 Amazon Elastic Block Store (Amazon EBS) 磁碟區組態。根據預設，SMF 工作者會使用具有基準效能設定的 gp3 Amazon EBS 磁碟區。對於具有大型輸入檔案或許多小型檔案的工作負載，您可以透過增加 Amazon EBS 輸送量和 IOPS 設定來提高傳輸速度。您可以使用 AWS Command Line Interface () 更新這些設定AWS CLI。

輸送量 (MiB/s)

可從磁碟區讀取或寫入資料的速率。預設為 125 MiB/s，gp3 磁碟區上限為 1，000 MiB/s。針對大型循序檔案傳輸增加 。

IOPS

每秒的輸入/輸出操作。預設為 3，000 IOPS，gp3 磁碟區的上限為 16，000 IOPS。傳輸多個小型檔案時增加 。

> **Note**
>
> 提高 Amazon EBS 輸送量和 IOPS 會增加機群成本。如需定價資訊，請參閱截止日期雲端定價。

使用 更新現有機群上的 Amazon EBS 設定 AWS CLI

- 執行以下命令：

```
aws deadline update-fleet \
  --farm-id farm-0123456789abcdef0 \
  --fleet-id fleet-0123456789abcdef0 \
  --configuration '{
    "serviceManagedEc2": {
      "instanceCapabilities": {
        "vCpuCount": {"min": 4},
        "memoryMiB": {"min": 8192},
        "osFamily": "linux",
        "cpuArchitectureType": "x86_64",
        "rootEbsVolume": {
          "sizeGiB": 250,
          "iops": 6000,
          "throughputMiB": 500
        }
      },
      "instanceMarketOptions": {"type": "spot"}
    }
  }'
```

下載任務輸出

任務完成後，請使用截止日期雲端 CLI 或 AWS 截止日期雲端監視器 （ 截止日期雲端監視器 ） 下載輸出檔案：

```
deadline job download-output --job-id job-0123456789abcdef0
```

如需在任務完成時自動下載輸出，請參閱《截止日期雲端使用者指南》中的自動下載。

在工作者上部署和設定自訂軟體

AWS Deadline Cloud 提供多種方法來部署和設定工作者的自訂軟體、外掛程式和工具。您選擇的方法取決於您的需求，例如您是否需要管理員權限、軟體變更的頻率，以及軟體是否應可供所有任務或僅限特定任務使用。

選擇部署方法

使用下表為您的使用案例選擇正確的部署方法。

條件	佇列環境	主機組態指令碼	自訂 conda 套件
需要管理員權限	否	是	否
執行時	工作階段開始	工作者啟動	工作階段開始
Scope (範圍)	每個佇列或任務	機群中的所有工作者	每個佇列或任務
可以透過任務提交來控制	是	否	是
設定複雜性	低	中	高
最適合	簡單外掛程式、指令碼、環境變數	系統驅動程式、Docker、儲存體掛載	具有相依性的複雜應用程式

快速決策指南：

- 需要管理員或根權限？ 使用主機組態指令碼。
- 沒有管理員權限的簡單外掛程式或指令碼？ 使用佇列環境。
- 具有版本控制需求的複雜應用程式？ 建立自訂 conda 套件。

使用佇列環境設定任務

AWS Deadline Cloud 使用佇列環境在工作者上設定軟體。環境可讓您對工作階段中的所有任務執行一次耗時的任務，例如設定和捨棄。它定義了啟動或停止工作階段時要在工作者上執行的動作。您可以為佇列設定環境、在佇列中執行的任務，以及任務的個別步驟。

您可以將環境定義為佇列環境或任務環境。使用截止日期雲端主控台或截止日期：CreateQueueEnvironment 操作建立佇列環境，並在您提交之任務的任務範本中定義任務環境。它們遵循環境的開放任務描述 (OpenJD) 規格。如需詳細資訊，請參閱 GitHub 上 OpenJD 規格中的<Environment>。

除了 name和 之外description，每個環境還包含兩個在主機上定義環境的欄位。這些類別為：

- script – 在工作者上執行此環境時所採取的動作。
- variables – 一組在進入環境時設定的環境變數名稱/值對。

您必須設定至少一個 script或 variables。

您可以在任務範本中定義多個環境。每個環境都會按照範本中列出的順序套用。您可以使用此功能來協助管理環境的複雜性。

Deadline Cloud 的預設佇列環境使用 conda 套件管理員將軟體載入環境，但您可以使用其他套件管理員。預設環境會定義兩個參數，以指定應載入的軟體。這些變數是由 Deadline Cloud 提供的提交者所設定，不過您可以在自己的指令碼和使用預設環境的應用程式中進行設定。這些類別為：

- CondaPackages – 要為任務安裝之 conda 套件比對規格的空間分隔清單。例如，Blender 提交者會新增 blender=3.6以在 Blender 3.6 中轉譯影格。
- CondaChannels – 要從中安裝套件之 conda 通道的空間分隔清單。對於服務受管機群，套件會從deadline-cloud頻道安裝。您可以新增其他頻道。

使用 OpenJD 佇列環境控制任務環境

您可以使用佇列環境為轉譯任務定義自訂環境。佇列環境是一種範本，可控制在特定佇列中執行之任務的環境變數、檔案映射和其他設定。它可讓您為提交至佇列的任務量身打造執行環境，以符合工作負載的需求。 AWS Deadline Cloud 提供三個巢狀層級，您可以在其中套用開放任務描述 (OpenJD) 環境：佇列、任務和步驟。透過定義佇列環境，您可以確保不同類型的任務具有一致且最佳化的效能、簡化資源配置，以及簡化佇列管理。

佇列環境是您從 AWS 管理主控台或使用 連接到 AWS 帳戶中佇列的範本 AWS CLI。您可以為佇列建立一個環境，也可以建立套用的多個佇列環境，以建立執行環境。此方法可讓您在步驟中建立和測試環境，以協助確保其適用於您的任務。

任務和步驟環境會在您用來在佇列中建立任務的任務範本中定義。OpenJD 語法在這些不同的環境中是相同的。在本節中，我們會在任務範本中向他們顯示 。

主題

- 在佇列環境中設定環境變數
- 在佇列環境中設定路徑
- 從佇列環境執行背景協助程式程序

在佇列環境中設定環境變數

許多應用程式和架構使用環境變數來控制功能設定、記錄層級和顯示組態。您可以使用開放任務描述 (OpenJD) 環境來設定其範圍內每個任務命令繼承的環境變數。

環境變數範圍

AWS 截止日期 雲端會從您連接至佇列的佇列環境套用環境變數。在任務範本中，您也可以使用 OpenJD 環境在任務和步驟層級定義環境變數。在較窄範圍定義的變數會覆寫具有較廣泛範圍相同名稱的變數。

- 佇列環境 – 您附加至截止日期雲端中佇列的範本。變數會套用至提交至佇列的所有任務。您可以使用固定值的`variables`映射來設定變數，或使用動態值的指令碼。
- 任務環境 – 在任務範本`jobEnvironments`的 下定義。變數適用於任務中的所有步驟和任務。任務層級變數會以相同名稱覆寫佇列層級變數。
- 步驟環境 – 在任務範本`stepEnvironments`的 下定義。變數僅適用於該步驟中的任務。步驟層級變數會以相同名稱覆寫任務層級或佇列層級變數。

在佇列環境中設定變數

您可以使用固定值的`variables`映射，或在動態值`script`的 `onEnter` 動作中使用 ，在佇列環境中設定環境變數。

下列佇列環境範本使用`variables`映射將`QT_QPA_PLATFORM`變數設定為 `offscreen`，這可讓使用 Qt 架構的應用程式在沒有互動式顯示的工作者主機上執行。

```
specificationVersion: 'environment-2023-09'
environment:
  name: QtOffscreen
  variables:
    QT_QPA_PLATFORM: offscreen
```

對於動態值，例如修改PATH或啟用虛擬環境，請使用以 openjd_env: *VAR*=*value*stdout 格式列印行的指令碼。字openjd_env:首為必要。使用 echo、 export或其他不含字首的 Shell 機制，不會將變數傳播至任務。

下列佇列環境範本會使用指令碼設定QT_QPA_PLATFORM變數。

```
specificationVersion: 'environment-2023-09'
environment:
  name: QtOffscreen
  script:
    actions:
      onEnter:
        command: bash
        args:
        - "{{Env.File.Enter}}"
    embeddedFiles:
    - name: Enter
      type: TEXT
      data: |
        #!/bin/env bash
        set -euo pipefail
        echo "openjd_env: QT_QPA_PLATFORM=offscreen"
```

若要將佇列環境連接至佇列，請使用截止日期雲端主控台或 AWS CLI。如需詳細資訊，請參閱《 AWS 截止日期雲端使用者指南》中的建立佇列環境。下列 AWS CLI 命令會從範本檔案建立佇列環境。

```
aws deadline create-queue-environment \
    --farm-id FARM_ID \
    --queue-id QUEUE_ID \
    --priority 1 \
    --template-type YAML \
    --template file://my-queue-env.yaml
```

如需建立和啟用 conda 虛擬環境等更複雜的範例，請參閱 GitHub 上的截止日期雲端佇列環境範例。

在任務範本中設定變數

在任務範本中，將variables映射新增至 jobEnvironments或 stepEnvironments項目。每個項目都是索引鍵/值對，其中索引鍵是變數名稱，而值是變數值。

下列任務範本會將 QT_QPA_PLATFORM 環境變數設定為 offscreen，允許使用 Qt Framework 的應用程式在沒有互動式顯示的工作者主機上執行。

```
specificationVersion: 'jobtemplate-2023-09'
name: MyJob
jobEnvironments:
- name: JobEnv
  variables:
    QT_QPA_PLATFORM: offscreen
```

您可以在單一環境定義中設定多個變數。

```
jobEnvironments:
- name: JobEnv
  variables:
    JOB_VERBOSITY: MEDIUM
    JOB_PROJECT_ID: my-project-id
    JOB_ENDPOINT_URL: https://my-host-name/my/path
    QT_QPA_PLATFORM: offscreen
```

您可以使用 {{Param.*ParameterName*}} 語法，在變數值中參考任務參數。

```
jobEnvironments:
- name: JobEnv
  variables:
    JOB_EXAMPLE_PARAM: "{{Param.ExampleParam}}"
```

若要覆寫特定步驟的任務層級變數，請使用相同的變數名稱定義stepEnvironments項目。下列範例JOB_PROJECT_ID會在任務層級以值 定義 project-12，然後使用 覆寫步驟層級的值step-project-12。步驟中的任務使用步驟層級值。

```
specificationVersion: 'jobtemplate-2023-09'
name: MyJob
jobEnvironments:
- name: JobEnv
  variables:
    JOB_PROJECT_ID: project-12
steps:
- name: MyStep
  stepEnvironments:
  - name: StepEnv
    variables:
      JOB_PROJECT_ID: step-project-12
```

試用：執行環境變數範例

Deadline Cloud 範例儲存庫包含任務套件，示範設定和檢視環境變數。範例任務範本會在任務和步驟層級定義變數，然後執行列印合併結果的任務。使用下列程序執行範例並檢查結果。

先決條件

1. 如果您沒有具有佇列和相關聯 Linux 機群的截止日期雲端陣列，請遵循截止日期雲端主控台中的引導式加入體驗，以使用預設設定建立。
2. 如果您的工作站上沒有截止日期雲端 CLI AWS 和截止日期雲端監視器，請遵循設定截止日期雲端提交者中的步驟。
3. 使用 `git`複製截止日期雲端範例 GitHub 儲存庫。

```
git clone https://github.com/aws-deadline/deadline-cloud-samples.git
cd deadline-cloud-samples/job_bundles
```

執行範例

1. 使用截止日期雲端 CLI 提交`job_env_vars`範例。

```
deadline bundle submit job_env_vars
```

2. 在截止日期雲端監視器中，選取要監控其進度的新任務。與佇列相關聯的Linux機群有可用的工作者後，任務會在幾秒鐘內完成。選取任務，然後在任務面板的右上角選單中選擇檢視日誌。

比較工作階段動作及其定義

日誌檢視會顯示三個工作階段動作。在文字編輯器中開啟檔案 job_env_vars/template.yaml，將每個動作與其在任務範本中的定義進行比較。

1. 選取啟動 JobEnv 工作階段動作。日誌輸出會顯示正在設定的任務層級環境變數。

```
Setting: JOB_VERBOSITY=MEDIUM
Setting: JOB_EXAMPLE_PARAM=An example parameter value
Setting: JOB_PROJECT_ID=project-12
Setting: JOB_ENDPOINT_URL=https://internal-host-name/some/path
Setting: QT_QPA_PLATFORM=offscreen
```

   來自任務範本的下列行定義了此環境。

```
jobEnvironments:
- name: JobEnv
  variables:
    JOB_VERBOSITY: MEDIUM
    JOB_EXAMPLE_PARAM: "{{Param.ExampleParam}}"
    JOB_PROJECT_ID: project-12
    JOB_ENDPOINT_URL: https://internal-host-name/some/path
    QT_QPA_PLATFORM: offscreen
```

2. 選取啟動 StepEnv 工作階段動作。日誌輸出會顯示步驟層級變數，包括覆寫的 JOB_PROJECT_ID。

```
Setting: STEP_VERBOSITY=HIGH
Setting: JOB_PROJECT_ID=step-project-12
```

   來自任務範本的下列行定義了此環境。

```
stepEnvironments:
- name: StepEnv
  variables:
    STEP_VERBOSITY: HIGH
    JOB_PROJECT_ID: step-project-12
```

3. 選取任務執行工作階段動作。日誌輸出會顯示任務可用的合併環境變數。請注意，JOB_PROJECT_ID使用步驟層級值 step-project-12。

```
Environment variables starting with JOB_*:
JOB_ENDPOINT_URL=https://internal-host-name/some/path
JOB_EXAMPLE_PARAM='An example parameter value'
JOB_PROJECT_ID=step-project-12
JOB_VERBOSITY=MEDIUM

Environment variables starting with STEP_*:
STEP_VERBOSITY=HIGH
```

在佇列環境中設定路徑

使用 OpenJD 環境在 環境中提供新命令。首先建立包含指令碼檔案的目錄，然後將該目錄新增至PATH環境變數，以便指令碼中的可執行檔可以執行它們，而不需要每次指定目錄路徑。環境定義中

的變數清單不提供修改變數的方式，因此您可以改為執行指令碼來執行此操作。指令碼設定並修改 之後PATH，它會使用命令 將變數匯出至 OpenJD 執行期echo "openjd_env: PATH=$PATH"。

先決條件

執行下列步驟，從 Deadline Cloud 範例 github 儲存庫使用環境變數執行範例任務套件。

1. 如果您沒有具有佇列和相關聯 Linux 機群的截止日期雲端陣列，請遵循截止日期雲端主控台中的引導式加入體驗，以使用預設設定建立。
2. 如果您的工作站上沒有截止日期雲端 CLI 和截止日期雲端監視器，請遵循使用者指南中設定截止日期雲端提交者中的步驟。
3. 使用 git複製截止日期雲端範例 GitHub 儲存庫。

```
git clone https://github.com/aws-deadline/deadline-cloud-samples.git
 Cloning into 'deadline-cloud-samples'...
 ...
cd deadline-cloud-samples/job_bundles
```

執行路徑範例

1. 使用截止日期雲端 CLI 提交job_env_with_new_command範例。

```
 $ deadline bundle submit job_env_with_new_command
 Submitting to Queue: MySampleQueue
 ...
```

2. 在截止日期雲端監視器中，您會看到新任務，並且可以監控其進度。一旦與佇列相關聯的Linux機群有工作者可執行任務，任務會在幾秒鐘內完成。選取任務，然後選擇任務面板右上角選單中的檢視日誌選項。

   右側有兩個工作階段動作：啟動 RandomSleepCommand 和任務執行。視窗中央的日誌檢視器對應至右側的所選工作階段動作。

比較工作階段動作及其定義

在本節中，您會使用截止日期雲端監視器來比較工作階段動作與任務範本中定義它們的位置。它從上一節繼續。

在文字編輯器中開啟檔案 job_env_with_new_command/template.yaml。將工作階段動作與任務範本中定義它們的位置進行比較。

1. 在截止日期雲端監視器中選取啟動 RandomSleepCommand 工作階段動作。您將看到如下所示的日誌輸出。

```
 2024/07/16 17:25:32-07:00
 2024/07/16 17:25:32-07:00 ==============================================
 2024/07/16 17:25:32-07:00 --------- Entering Environment: RandomSleepCommand
 2024/07/16 17:25:32-07:00 ==============================================
 2024/07/16 17:25:32-07:00 ----------------------------------------------
 2024/07/16 17:25:32-07:00 Phase: Setup
 2024/07/16 17:25:32-07:00 ----------------------------------------------
 2024/07/16 17:25:32-07:00 Writing embedded files for Environment to disk.
 2024/07/16 17:25:32-07:00 Mapping: Env.File.Enter -> /sessions/session-
ab132a51b9b54d5da22cbe839dd946baaw1c8hk5/embedded_filesf3tq_1os/tmpbt8j_c3f
 2024/07/16 17:25:32-07:00 Mapping: Env.File.SleepScript -> /sessions/session-
ab132a51b9b54d5da22cbe839dd946baaw1c8hk5/embedded_filesf3tq_1os/tmperastlp4
 2024/07/16 17:25:32-07:00 Wrote: Enter -> /sessions/session-
ab132a51b9b54d5da22cbe839dd946baaw1c8hk5/embedded_filesf3tq_1os/tmpbt8j_c3f
 2024/07/16 17:25:32-07:00 Wrote: SleepScript -> /sessions/session-
ab132a51b9b54d5da22cbe839dd946baaw1c8hk5/embedded_filesf3tq_1os/tmperastlp4
 2024/07/16 17:25:32-07:00 ----------------------------------------------
 2024/07/16 17:25:32-07:00 Phase: Running action
 2024/07/16 17:25:32-07:00 ----------------------------------------------
 2024/07/16 17:25:32-07:00 Running command sudo -u job-user -i setsid -w /sessions/
session-ab132a51b9b54d5da22cbe839dd946baaw1c8hk5/tmpbwrquq5u.sh
 2024/07/16 17:25:32-07:00 Command started as pid: 2205
 2024/07/16 17:25:32-07:00 Output:
 2024/07/16 17:25:33-07:00 openjd_env: PATH=/sessions/session-
ab132a51b9b54d5da22cbe839dd946baaw1c8hk5/bin:/opt/conda/condabin:/home/job-
user/.local/bin:/home/job-user/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/
bin:/sbin:/bin:/var/lib/snapd/snap/bin
 No newer logs at this moment.
```

任務範本的下列幾行指定了此動作。

```
 jobEnvironments:
 - name: RandomSleepCommand
   description: Adds a command 'random-sleep' to the environment.
   script:
     actions:
       onEnter:
         command: bash
         args:
```

```
        - "{{Env.File.Enter}}"
    embeddedFiles:
    - name: Enter
      type: TEXT
      data: |
        #!/bin/env bash
        set -euo pipefail

        # Make a bin directory inside the session's working directory for providing
 new commands
        mkdir -p '{{Session.WorkingDirectory}}/bin'

        # If this bin directory is not already in the PATH, then add it
        if ! [[ ":$PATH:" == *':{{Session.WorkingDirectory}}/bin:'* ]]; then
          export "PATH={{Session.WorkingDirectory}}/bin:$PATH"

          # This message to Open Job Description exports the new PATH value to the
 environment
          echo "openjd_env: PATH=$PATH"
        fi

        # Copy the SleepScript embedded file into the bin directory
        cp '{{Env.File.SleepScript}}' '{{Session.WorkingDirectory}}/bin/random-
sleep'
        chmod u+x '{{Session.WorkingDirectory}}/bin/random-sleep'
    - name: SleepScript
      type: TEXT
      runnable: true
      data: |
        ...
```

2. 在截止日期雲端監視器中選取啟動 StepEnv 工作階段動作。您會看到如下所示的日誌輸出。

```
 2024/07/16 17:25:33-07:00
 2024/07/16 17:25:33-07:00 ================================================
 2024/07/16 17:25:33-07:00 --------- Running Task
 2024/07/16 17:25:33-07:00 ================================================
 2024/07/16 17:25:33-07:00 ------------------------------------------------
 2024/07/16 17:25:33-07:00 Phase: Setup
 2024/07/16 17:25:33-07:00 ------------------------------------------------
 2024/07/16 17:25:33-07:00 Writing embedded files for Task to disk.
 2024/07/16 17:25:33-07:00 Mapping: Task.File.Run -> /sessions/session-
ab132a51b9b54d5da22cbe839dd946baaw1c8hk5/embedded_filesf3tq_1os/tmpdrwuehjf
```

```
 2024/07/16 17:25:33-07:00 Wrote: Run -> /sessions/session-
ab132a51b9b54d5da22cbe839dd946baaw1c8hk5/embedded_filesf3tq_1os/tmpdrwuehjf
 2024/07/16 17:25:33-07:00 ------------------------------------------------
 2024/07/16 17:25:33-07:00 Phase: Running action
 2024/07/16 17:25:33-07:00 ------------------------------------------------
 2024/07/16 17:25:33-07:00 Running command sudo -u job-user -i setsid -w /sessions/
session-ab132a51b9b54d5da22cbe839dd946baaw1c8hk5/tmpz81iaqfw.sh
 2024/07/16 17:25:33-07:00 Command started as pid: 2256
 2024/07/16 17:25:33-07:00 Output:
 2024/07/16 17:25:34-07:00 + random-sleep 12.5 27.5
 2024/07/16 17:26:00-07:00 Sleeping for duration 26.90
 2024/07/16 17:26:00-07:00 ------------------------------------------------
 2024/07/16 17:26:00-07:00 Uploading output files to Job Attachments
 2024/07/16 17:26:00-07:00 ------------------------------------------------
```

3. 任務範本的下列幾行指定了此動作。

```
 steps:
 - name: EnvWithCommand
   script:
     actions:
       onRun:
         command: bash
         args:
         - '{{Task.File.Run}}'
     embeddedFiles:
     - name: Run
       type: TEXT
       data: |
         set -xeuo pipefail

         # Run the script installed into PATH by the job environment
         random-sleep 12.5 27.5
   hostRequirements:
     attributes:
     - name: attr.worker.os.family
       anyOf:
       - linux
```

從佇列環境執行背景協助程式程序

在許多轉譯使用案例中，載入應用程式和場景資料可能需要相當長的時間。如果任務為每個影格重新載入它們，它會花費大部分時間在額外負荷上。通常可以將應用程式載入一次做為背景協助程式程序，讓它載入場景資料，然後透過程序間通訊 (IPC) 傳送命令來執行轉譯。

許多開放原始碼截止日期雲端整合都使用此模式。開放任務描述專案在所有支援的作業系統上，提供具有強大 IPC 模式的轉接器執行期程式庫。

為了示範此模式，有一個獨立的範例任務套件，它使用 Python 和 bash 程式碼來實作背景協助程式和 IPC，讓任務與其通訊。協助程式在 Python 中實作，並監聽 POSIX SIGUSR1 訊號，了解何時處理任務。任務詳細資訊會傳遞至特定 JSON 檔案中的協助程式，而執行任務的結果會傳回為另一個 JSON 檔案。

先決條件

執行下列步驟，從 Deadline Cloud 範例 github 儲存庫使用協助程式程序執行範例任務套件。

1. 如果您沒有具有佇列和相關聯 Linux 機群的截止日期雲端陣列，請遵循截止日期雲端主控台中的引導式加入體驗，以使用預設設定建立。
2. 如果您的工作站上沒有截止日期雲端 CLI 和截止日期雲端監視器，請遵循使用者指南中設定截止日期雲端提交者中的步驟。
3. 使用 `git`複製截止日期雲端範例 GitHub 儲存庫。

```
git clone https://github.com/aws-deadline/deadline-cloud-samples.git
 Cloning into 'deadline-cloud-samples'...
 ...
cd deadline-cloud-samples/job_bundles
```

執行協助程式範例

1. 使用截止日期雲端 CLI 提交`job_env_daemon_process`範例。

```
 git clone https://github.com/aws-deadline/deadline-cloud-samples.git
Cloning into 'deadline-cloud-samples'...
 ...
cd deadline-cloud-samples/job_bundles
```

2. 在截止日期雲端監控應用程式中，您會看到新的任務，並且可以監控其進度。一旦與佇列相關聯的Linux機群有工作者可執行任務，大約會在一分鐘內完成。選取其中一個任務後，選擇任務面板右上角選單中的檢視日誌選項。

   右側有兩個工作階段動作：啟動 DaemonProcess 和任務執行。視窗中央的日誌檢視器對應至右側的所選工作階段動作。

   選取檢視所有任務的日誌選項。時間軸會顯示在工作階段中執行的其餘任務，以及結束環境Shut down DaemonProcess的動作。

檢視協助程式日誌

1. 在本節中，您會使用截止日期雲端監視器來比較工作階段動作與任務範本中定義它們的位置。它從上一節繼續。

   在文字編輯器中開啟檔案 job_env_daemon_process/template.yaml。將工作階段動作與任務範本中定義它們的位置進行比較。

2. 在截止日期雲端監視器中選取Launch DaemonProcess工作階段動作。您將看到如下所示的日誌輸出。

```
 2024/07/17 16:27:20-07:00
 2024/07/17 16:27:20-07:00 ================================================
 2024/07/17 16:27:20-07:00 --------- Entering Environment: DaemonProcess
 2024/07/17 16:27:20-07:00 ================================================
 2024/07/17 16:27:20-07:00 ------------------------------------------------
 2024/07/17 16:27:20-07:00 Phase: Setup
 2024/07/17 16:27:20-07:00 ------------------------------------------------
 2024/07/17 16:27:20-07:00 Writing embedded files for Environment to disk.
 2024/07/17 16:27:20-07:00 Mapping: Env.File.Enter -> /sessions/
session-972e21d98dde45e59c7153bd9258a64dohwg4yg1/embedded_fileswy00x5ra/enter-daemon-
process-env.sh
 2024/07/17 16:27:20-07:00 Mapping: Env.File.Exit -> /sessions/
session-972e21d98dde45e59c7153bd9258a64dohwg4yg1/embedded_fileswy00x5ra/exit-daemon-
process-env.sh
 2024/07/17 16:27:20-07:00 Mapping: Env.File.DaemonScript -> /sessions/
session-972e21d98dde45e59c7153bd9258a64dohwg4yg1/embedded_fileswy00x5ra/daemon-
script.py
 2024/07/17 16:27:20-07:00 Mapping: Env.File.DaemonHelperFunctions -> /sessions/
session-972e21d98dde45e59c7153bd9258a64dohwg4yg1/embedded_fileswy00x5ra/daemon-
helper-functions.sh
```

```
 2024/07/17 16:27:20-07:00 Wrote: Enter -> /sessions/
session-972e21d98dde45e59c7153bd9258a64dohwg4yg1/embedded_fileswy00x5ra/enter-daemon-
process-env.sh
 2024/07/17 16:27:20-07:00 Wrote: Exit -> /sessions/
session-972e21d98dde45e59c7153bd9258a64dohwg4yg1/embedded_fileswy00x5ra/exit-daemon-
process-env.sh
 2024/07/17 16:27:20-07:00 Wrote: DaemonScript -> /sessions/
session-972e21d98dde45e59c7153bd9258a64dohwg4yg1/embedded_fileswy00x5ra/daemon-
script.py
 2024/07/17 16:27:20-07:00 Wrote: DaemonHelperFunctions -> /sessions/
session-972e21d98dde45e59c7153bd9258a64dohwg4yg1/embedded_fileswy00x5ra/daemon-
helper-functions.sh
 2024/07/17 16:27:20-07:00 ----------------------------------------------
 2024/07/17 16:27:20-07:00 Phase: Running action
 2024/07/17 16:27:20-07:00 ----------------------------------------------
 2024/07/17 16:27:20-07:00 Running command sudo -u job-user -i setsid -w /sessions/
session-972e21d98dde45e59c7153bd9258a64dohwg4yg1/tmp_u8slys3.sh
 2024/07/17 16:27:20-07:00 Command started as pid: 2187
 2024/07/17 16:27:20-07:00 Output:
 2024/07/17 16:27:21-07:00 openjd_env: DAEMON_LOG=/sessions/
session-972e21d98dde45e59c7153bd9258a64dohwg4yg1/daemon.log
 2024/07/17 16:27:21-07:00 openjd_env: DAEMON_PID=2223
 2024/07/17 16:27:21-07:00 openjd_env: DAEMON_BASH_HELPER_SCRIPT=/sessions/
session-972e21d98dde45e59c7153bd9258a64dohwg4yg1/embedded_fileswy00x5ra/daemon-
helper-functions.sh
```

任務範本的下列幾行指定了此動作。

```
  stepEnvironments:
  - name: DaemonProcess
    description: Runs a daemon process for the step's tasks to share.
    script:
      actions:
        onEnter:
          command: bash
          args:
          - "{{Env.File.Enter}}"
        onExit:
          command: bash
          args:
          - "{{Env.File.Exit}}"
      embeddedFiles:
      - name: Enter
```

```
          filename: enter-daemon-process-env.sh
          type: TEXT
          data: |
            #!/bin/env bash
            set -euo pipefail

            DAEMON_LOG='{{Session.WorkingDirectory}}/daemon.log'
            echo "openjd_env: DAEMON_LOG=$DAEMON_LOG"
            nohup python {{Env.File.DaemonScript}} > $DAEMON_LOG 2>&1 &
            echo "openjd_env: DAEMON_PID=$!"
            echo "openjd_env:
 DAEMON_BASH_HELPER_SCRIPT={{Env.File.DaemonHelperFunctions}}"

            echo 0 > 'daemon_log_cursor.txt'
      ...
```

3. 在截止日期雲端監視器中選取其中一個任務執行：N 個工作階段動作。您將看到如下所示的日誌輸出。

```
2024/07/17 16:27:22-07:00
 2024/07/17 16:27:22-07:00 ============================================
 2024/07/17 16:27:22-07:00 --------- Running Task
 2024/07/17 16:27:22-07:00 ============================================
 2024/07/17 16:27:22-07:00 Parameter values:
 2024/07/17 16:27:22-07:00 Frame(INT) = 2
 2024/07/17 16:27:22-07:00 --------------------------------------------
 2024/07/17 16:27:22-07:00 Phase: Setup
 2024/07/17 16:27:22-07:00 --------------------------------------------
 2024/07/17 16:27:22-07:00 Writing embedded files for Task to disk.
 2024/07/17 16:27:22-07:00 Mapping: Task.File.Run -> /sessions/
session-972e21d98dde45e59c7153bd9258a64dohwg4yg1/embedded_fileswy00x5ra/run-task.sh
 2024/07/17 16:27:22-07:00 Wrote: Run -> /sessions/
session-972e21d98dde45e59c7153bd9258a64dohwg4yg1/embedded_fileswy00x5ra/run-task.sh
 2024/07/17 16:27:22-07:00 --------------------------------------------
 2024/07/17 16:27:22-07:00 Phase: Running action
 2024/07/17 16:27:22-07:00 --------------------------------------------
 2024/07/17 16:27:22-07:00 Running command sudo -u job-user -i setsid -w /sessions/
session-972e21d98dde45e59c7153bd9258a64dohwg4yg1/tmpv4obfkhn.sh
 2024/07/17 16:27:22-07:00 Command started as pid: 2301
 2024/07/17 16:27:22-07:00 Output:
 2024/07/17 16:27:23-07:00 Daemon PID is 2223
 2024/07/17 16:27:23-07:00 Daemon log file is /sessions/
session-972e21d98dde45e59c7153bd9258a64dohwg4yg1/daemon.log
```

```
2024/07/17 16:27:23-07:00
2024/07/17 16:27:23-07:00 === Previous output from daemon
2024/07/17 16:27:23-07:00 ===
2024/07/17 16:27:23-07:00
2024/07/17 16:27:23-07:00 Sending command to daemon
2024/07/17 16:27:23-07:00 Received task result:
2024/07/17 16:27:23-07:00 {
2024/07/17 16:27:23-07:00   "result": "SUCCESS",
2024/07/17 16:27:23-07:00   "processedTaskCount": 1,
2024/07/17 16:27:23-07:00   "randomValue": 0.2578537967668988,
2024/07/17 16:27:23-07:00   "failureRate": 0.1
2024/07/17 16:27:23-07:00 }
2024/07/17 16:27:23-07:00
2024/07/17 16:27:23-07:00 === Daemon log from running the task
2024/07/17 16:27:23-07:00 Loading the task details file
2024/07/17 16:27:23-07:00 Received task details:
2024/07/17 16:27:23-07:00 {
2024/07/17 16:27:23-07:00  "pid": 2329,
2024/07/17 16:27:23-07:00  "frame": 2
2024/07/17 16:27:23-07:00 }
2024/07/17 16:27:23-07:00 Processing frame number 2
2024/07/17 16:27:23-07:00 Writing result
2024/07/17 16:27:23-07:00 Waiting until a USR1 signal is sent...
2024/07/17 16:27:23-07:00 ===
2024/07/17 16:27:23-07:00
2024/07/17 16:27:23-07:00 ------------------------------------------------
2024/07/17 16:27:23-07:00 Uploading output files to Job Attachments
2024/07/17 16:27:23-07:00 ------------------------------------------------
```

下列來自任務範本的行會指定此動作。``` 步驟：

```
steps:
- name: EnvWithDaemonProcess
  parameterSpace:
    taskParameterDefinitions:
    - name: Frame
      type: INT
      range: "{{Param.Frames}}"

  stepEnvironments:
    ...

  script:
```

```
    actions:
      onRun:
        timeout: 60
        command: bash
        args:
        - '{{Task.File.Run}}'
    embeddedFiles:
    - name: Run
      filename: run-task.sh
      type: TEXT
      data: |
        # This bash script sends a task to the background daemon process,
        # then waits for it to respond with the output result.

        set -euo pipefail

        source "$DAEMON_BASH_HELPER_SCRIPT"

        echo "Daemon PID is $DAEMON_PID"
        echo "Daemon log file is $DAEMON_LOG"

        print_daemon_log "Previous output from daemon"

        send_task_to_daemon "{\"pid\": $$, \"frame\": {{Task.Param.Frame}} }"
        wait_for_daemon_task_result

        echo Received task result:
        echo "$TASK_RESULT" | jq .

        print_daemon_log "Daemon log from running the task"

  hostRequirements:
    attributes:
    - name: attr.worker.os.family
      anyOf:
      - linux
```

為您的任務提供應用程式

您可以使用佇列環境載入應用程式來處理任務。當您使用截止日期雲端主控台建立服務受管機群時，您可以選擇建立使用 conda 套件管理員載入應用程式的佇列環境。

如果您想要使用不同的套件管理員，您可以為該管理員建立佇列環境。如需使用 Rez 的範例，請參閱使用不同的套件管理員。

Deadline Cloud 提供 conda 頻道，將一系列的轉譯應用程式載入您的環境。它們支援 Deadline Cloud 為數位內容建立應用程式提供的提交者。

您也可以載入軟體，讓 conda-forge 用於您的任務。下列範例顯示使用 Deadline Cloud 提供的佇列環境在執行任務之前載入應用程式的任務範本。

主題

從 conda 頻道取得應用程式

您可以為安裝所選軟體的截止日期雲端工作者建立自訂佇列環境。此範例佇列環境的行為與主控台用於服務受管機群的環境相同。它會直接執行 conda 以建立環境。

環境會為每個在工作者上執行的截止日期雲端工作階段建立新的 conda 虛擬環境，然後在完成時刪除環境。

Conda 會快取下載的套件，使其不需要再次下載，但每個工作階段都必須將所有套件連結至環境。

環境定義了三種指令碼，當 Deadline Cloud 在工作者上啟動工作階段時執行。第一個指令碼會在呼叫`onEnter`動作時執行。它會呼叫其他兩個 來設定環境變數。當指令碼完成執行時，Conda 環境可用於所有指定的環境變數集。

如需範例的最新版本，請參閱 GitHub 上 deadline-cloud-samples 儲存庫中的 conda_queue_env_console_equivalent.yaml。

如果您想要使用 Conda 頻道中無法使用的應用程式，您可以在 Amazon S3 中建立 conda 頻道，然後為該應用程式建置自己的套件。如需進一步了解，請參閱使用 S3 建立 conda 頻道。

從 conda-forge 取得開放原始碼程式庫

本節說明如何從`conda-forge`頻道使用開放原始碼程式庫。下列範例是使用 `polars` Python 套件的任務範本。

任務會設定佇列環境中定義的 `CondaPackages`和 `CondaChannels` 參數，告知 Deadline Cloud 從何處取得套件。

設定參數的任務範本區段為：

```
- name: CondaPackages
  description: A list of conda packages to install. The job expects a Queue Environment
 to handle this.
  type: STRING
  default: polars
- name: CondaChannels
  description: A list of conda channels to get packages from. The job expects a Queue
 Environment to handle this.
  type: STRING
  default: conda-forge
```

如需完整範例任務範本的最新版本，請參閱 stage_1_self_contained_template/template.yaml。如需載入 conda 套件的最新版本佇列環境，請參閱 GitHub 上 deadline-cloud-samples 儲存庫中的 conda_queue_env_console_equivalent.yaml。

Blender 從截止日期雲端管道取得

下列範例顯示Blender從 `deadline-cloud` conda 頻道取得的任務範本。此頻道支援 Deadline Cloud 為數位內容建立軟體提供的提交者，但您可以使用相同的頻道來載入軟體供自己使用。

如需`deadline-cloud`頻道提供的軟體清單，請參閱AWS 《截止日期雲端使用者指南》中的預設佇列環境。

此任務會設定佇列環境中定義的 `CondaPackages` 參數，以指示截止日期雲端Blender載入環境。

設定 參數的任務範本區段為：

```
- name: CondaPackages
  type: STRING
  userInterface:
    control: LINE_EDIT
    label: Conda Packages
    groupLabel: Software Environment
  default: blender
  description: >
    Tells the queue environment to install Blender from the deadline-cloud conda
 channel.
```

如需完整範例任務範本的最新版本，請參閱 blender_render/template.yaml。如需載入 conda 套件的最新版本佇列環境，請參閱 上 deadline-cloud-samples 儲存庫中的 conda_queue_env_console_equivalent.yamlGitHub。

使用不同的套件管理員

截止日期雲端的預設套件管理員為 conda。如果您需要使用不同的套件管理員，例如 Rez，您可以建立自訂佇列環境，其中包含改用套件管理員的指令碼。

此範例佇列環境提供與主控台用於服務受管機群的環境相同的行為。它會以 取代 conda 套件管理員 Rez。

環境定義了三種指令碼，當 Deadline Cloud 在工作者上啟動工作階段時執行。第一個指令碼會在呼叫`onEnter`動作時執行。它會呼叫其他兩個 來設定環境變數。當指令碼完成執行時，Rez環境可使用所有指定的環境變數集。

此範例假設您有一個客戶受管機群，該機群使用 Rez 套件的共用檔案系統。

如需範例的最新版本，請參閱 上 deadline-cloud-samples 儲存庫中的 rez_queue_env.yaml。 GitHub

使用 S3 建立 conda 頻道

如果您的任務需要執行 `deadline-cloud`或 `conda-forge`頻道上無法使用的應用程式，您可以託管自訂 conda 頻道來提供您自己的套件。當您在截止日期雲端 AWS （ 截止日期雲端 ） 主控台中建立佇列時，主控台預設會新增 Conda 佇列環境。若要讓套件可供任務使用，請將自訂頻道新增至佇列環境。

conda 頻道是靜態託管內容，您可以透過各種方式託管，包括檔案系統或 Amazon Simple Storage Service (Amazon S3) 儲存貯體。如果您的截止日期雲端陣列使用資產的共用檔案系統，您可以使用其上的任何路徑做為頻道名稱。您可以在 Amazon S3 儲存貯體中託管頻道，以便使用 AWS Identity and Access Management (IAM) 許可進行更廣泛的存取。

您可以在本機建置和測試套件，然後將其發佈到頻道。在本機建置套件是開始在套件建置配方上迭代的簡單方法，無需基礎設施設定。您也可以使用截止日期雲端套件建置佇列來建置套件，並將其發佈至頻道。套件建置佇列可簡化多個作業系統和加速器組態的維護套件。您可以從任何地方更新版本並提交整組套件組建。

您可以透過多種方式為 Studio 和 Deadline Cloud Farm 設定頻道。您可以有一個 Amazon S3 頻道，並將所有工作站和陣列主機設定為使用它。您也可以擁有多個頻道，並使用 AWS DataSync

(DataSync) 設定鏡像。例如，您的 Deadline Cloud 套件建置佇列可以發佈到 Amazon S3 頻道，該頻道在現場部署針對工作站和現場部署陣列主機進行鏡像。

主題

在本機建置和測試套件

在將套件發佈至 Amazon S3 或在截止日期雲端陣列上設定 CI/CD 自動化之前，您可以使用本機檔案系統頻道在工作站上建置和測試 conda 套件。此方法可讓您在配方上於本機快速迭代並驗證套件。

`rattler-build publish` 命令會建置配方、將產生的套件複製到頻道，並在一個步驟中為頻道編製索引。當您以本機檔案系統目錄為目標時，如果目錄不存在， 會自動`rattler-build`建立和初始化頻道。

下列指示使用來自 上 Deadline Cloud 範例儲存庫的 Blender4.5 範例配方。 https://github.com/aws-deadline/deadline-cloud-samples GitHub您可以取代與範例儲存庫不同的配方，或使用您自己的配方。

先決條件

開始之前，請在工作站上安裝下列工具：

- pixi – 用於安裝`rattler-build`和測試套件的套件管理員。從 pixi.sh 安裝 pixi。
- rattler-build – Deadline Cloud conda 配方所使用的套件建置工具。安裝 pixi 之後，請執行下列命令來安裝 `rattler-build`。

```
pixi global install rattler-build
```

- git – 複製範例儲存庫時需要。在 上Windows，適用於 的 git Windows 也提供 bash shell，其中一些Windows範例配方需要。

建置套件並將其發佈至本機頻道

在此程序中，您會複製截止日期雲端範例儲存庫，並使用 `rattler-build publish`來建置和發佈套件至本機檔案系統頻道。

> **Note**
>
> 大型應用程式可能需要數十 GB 的可用磁碟空間，才能進行來源封存、解壓縮檔案和建置輸出。請確定您使用的磁碟有足夠的可用空間供套件建置輸出使用。

建置套件並將其發佈至本機頻道

1. 複製截止日期雲端範例儲存庫。

   ```
   git clone https://github.com/aws-deadline/deadline-cloud-samples.git
   ```

2. 切換至 `conda_recipes` 目錄。

   ```
   cd deadline-cloud-samples/conda_recipes
   ```

3. 執行下列命令來建置 Blender 4.5 配方，並將套件發佈至本機頻道目錄。

   在 Linux和 macOS上執行下列命令。

   ```
   rattler-build publish blender-4.5/recipe/recipe.yaml \
       --to file://$HOME/my-conda-channel \
       --build-number=+1
   ```

   在 Windows(cmd) 上執行下列命令。

   ```
   rattler-build publish blender-4.5/recipe/recipe.yaml ^
       --to file://%USERPROFILE%/my-conda-channel ^
       --build-number=+1
   ```

`rattler-build publish` 命令會執行下列動作：

- 從配方建置套件。
- 如果目錄不存在，則建立頻道目錄。
- 將套件檔案複製到頻道。
- 為頻道編製索引，讓套件管理員可以找到套件。

如果您的套件配方取決於來自特定頻道的套件，例如 conda-forge，請將 `-c conda-forge`新增至 命令。

> **關於組建編號**
>
> `--build-number=+1` 選項會根據目的地頻道中已存在的內容，自動挑選下一個建置編號。最佳實務是絕不覆寫頻道中的套件。如果套件具有相同的檔案名稱，則一律建置為新的組建編號。當您建置到生產頻道或反映生產的預備頻道時，使用 會`--build-number=+1`達成此目標。
> 如果您想要直接控制建置編號，您可以使用 等特定值來設定建置編號`--build-number=7`。
> 如果您省略 選項， `rattler-build`會使用 `recipe.yaml` 檔案中定義的組建編號。

如需 的詳細資訊`rattler-build publish`，請參閱 rattler-build 發佈文件。

偵錯組建

如果建置失敗， 會`rattler-build`保留建置目錄，以便您進行調查。執行下列命令，在建置環境中開啟互動式 Shell，並在建置期間設定所有環境變數。

```
rattler-build debug shell
```

從偵錯 shell，您可以修改檔案、執行個別建置命令，以及新增相依性來隔離問題。如需詳細資訊，請參閱 rattler-build 文件中的偵錯組建。

測試套件

在您建置和發佈套件之後，請建立臨時 pixi 專案。使用 專案從本機頻道安裝套件，並驗證其是否正常運作。

測試套件

1. 建立暫時測試目錄，並使用本機頻道初始化 pixi 專案。

   在 Linux和 macOS上執行下列命令。

   ```
   mkdir package-test-env
   cd package-test-env
   pixi init --channel file://$HOME/my-conda-channel
   ```

   在 Windows(cmd) 上執行下列命令。

   ```
   mkdir package-test-env
   cd package-test-env
   pixi init --channel file://%USERPROFILE%/my-conda-channel
   ```

2. 將套件新增至專案。

   ```
   pixi add blender=4.5
   ```

3. 驗證套件是否正常運作。

   ```
   pixi run blender --version
   ```

   `pixi run` 命令會啟用專案目錄的 conda 環境，並在其中執行指定的命令。環境會保留在專案目錄中，因此您可以從其他終端機使用相同的`pixi run`命令。

當您對套件感到滿意時，您可以將套件發佈到 Amazon S3 Conda 頻道，以便截止日期雲端工作者可以安裝套件。請參閱將套件發佈至 S3 conda 頻道。

從頻道移除套件

避免從用於生產的頻道中移除套件，因為 lockfiles 會依雜湊參考特定套件。移除套件可防止從這些 lockfile 重新建立環境。對於開發和測試頻道，您可以從頻道目錄中刪除`.conda`檔案，然後重新索引頻道，以移除特定套件。首先，安裝 `rattler-index`。

```
pixi global install rattler-index
```

然後刪除套件檔案並重新索引頻道。

在 Linux和 macOS上執行下列命令。

```
rm $HOME/my-conda-channel/linux-64/blender-4.5.0-hb0f4dca_1.conda
rattler-index fs $HOME/my-conda-channel
```

在 Windows(cmd) 上執行下列命令。

```
del %USERPROFILE%\my-conda-channel\win-64\blender-4.5.0-hb0f4dca_1.conda
rattler-index fs %USERPROFILE%\my-conda-channel
```

套件檔案存放在平台特定的子目錄中`linux-64`，例如 `win-64`、 或 `osx-arm64`。列出這些子目錄的內容，以尋找您要移除之套件的確切檔案名稱。

清除

測試之後，您可以移除測試專案和本機頻道。

清除測試資源

1. 移除測試專案目錄。

   在 Linux和 macOS上執行下列命令。

   ```
   rm -rf package-test-env
   ```

   在 Windows(cmd) 上執行下列命令。

   ```
   rmdir /s /q package-test-env
   ```

2. 移除本機 conda 頻道目錄。

   在 Linux和 macOS上執行下列命令。

   ```
   rm -rf $HOME/my-conda-channel
   ```

   在 Windows(cmd) 上執行下列命令。

   ```
   rmdir /s /q %USERPROFILE%\my-conda-channel
   ```

3. （選用）移除包含建置套件檔案的`rattler-build`輸出目錄。

在 Linux和 macOS上執行下列命令。

```
rm -rf deadline-cloud-samples/conda_recipes/output
```

在 Windows(cmd) 上執行下列命令。

```
rmdir /s /q deadline-cloud-samples\conda_recipes\output
```

將套件發佈至 Amazon S3 conda 頻道

您可以將 conda 套件發佈到 Amazon Simple Storage Service (Amazon S3) 儲存貯體，以便 AWS 截止日期雲端 （截止日期雲端） 工作者可以安裝它們來執行任務。此`rattler-build publish`命令使用 Amazon S3 的方式與使用本機檔案系統頻道的方式相同。命令可以建置配方並發佈結果，或發佈您已建置的套件檔案。在這兩種情況下， 命令都會將套件上傳至儲存貯體，並在單一步驟中為頻道編製索引。

`rattler-build publish` 命令 AWS 會使用標準登入資料鏈向 進行身分驗證，因此會像任何 AWS 工具一樣使用您的 AWS 組態。如需設定登入資料的詳細資訊，請參閱《AWS Command Line Interface (AWS CLI) 使用者指南》中的組態和登入資料檔案設定。

先決條件

將套件發佈至 Amazon S3 之前，請先完成下列先決條件：

- pixi 和 rattler-build – 從 pixi.sh 安裝 pixi，然後安裝 `rattler-build`。

  ```
  pixi global install rattler-build
  ```

- git – 複製範例儲存庫時需要。在 上Windows，適用於 的 git Windows 也提供 `bash` shell，其中一些Windows範例配方需要。
- Amazon S3 儲存貯體 – 用作 conda 頻道的 Amazon S3 儲存貯體。您可以從截止日期雲端陣列使用任務附件儲存貯體，或建立單獨的儲存貯體。
- AWS 登入資料 – 使用 `aws configure`命令或 `aws login`命令在工作站上設定登入資料。如需詳細資訊，請參閱 AWS Command Line Interface 使用者指南中的設定 AWS CLI。
- IAM 許可 – （選用） 若要減少憑證擁有的許可範圍，您可以使用 AWS Identity and Access Management (IAM) 政策，僅授予 Amazon S3 儲存貯體和您使用的頻道字首的下列許可 （例如 `/Conda/*`）：

- s3:GetObject
- s3:PutObject
- s3:DeleteObject
- s3:ListBucket
- s3:GetBucketLocation

將套件發佈至 Amazon S3 頻道

使用 rattler-build publish搭配 s3://目標，將套件發佈到您的 Amazon S3 Conda 頻道。如果頻道不存在於儲存貯體中， 會自動rattler-build初始化頻道。開始之前，請確定您已完成先決條件。

下列範例會從 上的截止日期雲端範例儲存庫發佈 Blender4.5 範例配方GitHub。 https://github.com/aws-deadline/deadline-cloud-samples您可以取代與範例儲存庫不同的配方，或使用您自己的配方。

> **Note**
>
> 大型應用程式可能需要數十 GB 的可用磁碟空間，才能進行來源封存、解壓縮檔案和建置輸出。請確定您使用的磁碟有足夠的可用空間供套件建置輸出使用。

將套件發佈至 Amazon S3 頻道

1. 複製截止日期雲端範例儲存庫。

```
git clone https://github.com/aws-deadline/deadline-cloud-samples.git
```

2. 切換至 conda_recipes 目錄。

```
cd deadline-cloud-samples/conda_recipes
```

3. 執行下列命令。將 *amzn-s3-demo-bucket* 取代為您的儲存貯體名稱。

```
rattler-build publish blender-4.5/recipe/recipe.yaml --to s3://amzn-s3-demo-bucket/
Conda/Default --build-number=+1
```

   /Conda/Default 字首會在儲存貯體中組織頻道。您可以使用不同的字首，但參考頻道的所有命令和佇列組態的字首必須一致。

> **關於組建編號**
>
> `--build-number=+1` 選項會根據目的地頻道中已存在的內容，自動挑選下一個建置編號。最佳實務是絕不覆寫頻道中的套件。如果套件具有相同的檔案名稱，則一律建置為新的組建編號。當您建置到生產頻道或反映生產的預備頻道時，使用 會`--build-number=+1`達成此目標。
>
> 如果您想要直接控制建置編號，您可以使用 等特定值來設定建置編號`--build-number=7`。
>
> 如果您省略 選項， `rattler-build`會使用 `recipe.yaml` 檔案中定義的組建編號。

如果您的套件配方取決於來自特定頻道的套件，例如 conda-forge，請將 `-c conda-forge`新增至 命令。

您也可以發佈已建置的套件檔案，例如來自本機建置`.conda`的檔案。將 *amzn-s3-demo-bucket* 取代為您的儲存貯體名稱。

```
rattler-build publish output/linux-64/blender-4.5.0-hb0f4dca_0.conda \
    --to s3://amzn-s3-demo-bucket/Conda/Default
```

初始化或重新索引頻道

當您使用 `rattler-build publish` 發佈套件時，如果頻道尚未存在，命令會自動初始化頻道。在大多數情況下，您不需要手動初始化或重新索引頻道。

在下列情況中，您可能需要手動初始化或重新索引頻道：

- 您想要在發佈任何套件之前建立空白頻道，例如，確認您的截止日期雲端佇列環境可以連線至頻道。
- 您已使用 Amazon S3 工具直接上傳或刪除`.conda`檔案，而不是使用 `rattler-build publish`，且頻道索引已過期。

初始化空白頻道

若要初始化空白頻道，請建立 `repodata.json` 檔案，並將其上傳至頻道字首的`noarch`子目錄。將 *amzn-s3-demo-bucket* 取代為您的儲存貯體名稱。

```
echo '{"info":{"subdir":"noarch"},"packages":{},"packages.conda":{},"removed":
[],"repodata_version":1}' > empty_channel_repodata.json
aws s3api put-object --body empty_channel_repodata.json --key Conda/Default/noarch/
repodata.json --bucket amzn-s3-demo-bucket
```

/Conda/Default 字首必須符合佇列環境使用的頻道字首。初始化頻道之後，您可以使用 將套件發佈至頻道rattler-build publish。

為頻道重新編製索引

如果頻道索引已過期，請使用 從頻道中的套件檔案rattler-index重建索引。首先，安裝 rattler-index。

```
pixi global install rattler-index
```

然後重新索引頻道。將 *amzn-s3-demo-bucket* 取代為您的儲存貯體名稱。

```
rattler-index s3 s3://amzn-s3-demo-bucket/Conda/Default
```

測試套件

發佈套件後，請建立臨時 pixi 專案，以驗證套件是否正常運作。專案會從 Amazon S3 頻道安裝套件。

測試套件

1. 建立暫時測試目錄，並使用 Amazon S3 頻道初始化 pixi 專案。將 *amzn-s3-demo-bucket* 取代為您的儲存貯體名稱。

   ```
   mkdir package-test-env
   cd package-test-env
   pixi init --channel s3://amzn-s3-demo-bucket/Conda/Default
   ```

2. 將套件新增至專案。

   ```
   pixi add blender=4.5
   ```

3. 驗證套件是否正常運作。

   ```
   pixi run blender --version
   ```

   pixi run 命令會啟用專案目錄的 conda 環境，並在其中執行指定的命令。環境會保留在專案目錄中，因此您可以從其他終端機使用相同的pixi run命令。

從頻道移除套件

避免從用於生產的頻道中移除套件，因為 lockfiles 會依雜湊參考特定套件。移除套件可防止從這些 lockfile 重新建立環境。對於開發和測試通道，您可以從儲存貯體中刪除.conda檔案，然後重新索引通道，以移除特定套件。

刪除套件檔案，然後重新索引頻道。將 *amzn-s3-demo-bucket* 取代為您的儲存貯體名稱。

```
aws s3 rm s3://amzn-s3-demo-bucket/Conda/Default/linux-64/blender-4.5.0-
hb0f4dca_1.conda
```

刪除檔案後，請重新索引頻道以更新頻道中繼資料。如需說明，請參閱為頻道重新編製索引。

套件檔案存放在平台特定的子目錄中linux-64，例如 win-64、 或 osx-arm64。若要列出子目錄中的套件，請執行下列命令。

```
aws s3 ls s3://amzn-s3-demo-bucket/Conda/Default/linux-64/
```

清除

測試後，移除測試專案目錄。

清除測試資源

- 移除測試專案目錄。

  在 Linux和 macOS上執行下列命令。

  ```
  rm -rf package-test-env
  ```

  在 Windows(cmd) 上執行下列命令。

  ```
  rmdir /s /q package-test-env
  ```

偵錯組建

如果建置失敗， 會rattler-build保留建置目錄，以便您進行調查。執行下列命令，在建置環境中開啟互動式 Shell，並在建置期間設定所有環境變數。

```
rattler-build debug shell
```

從偵錯 shell，您可以修改檔案、執行個別建置命令，以及新增相依性來隔離問題。如需詳細資訊，請參閱 rattler-build 文件中的偵錯組建。

為其他平台建置套件

`rattler-build publish` 命令會為執行命令之工作站的作業系統建置套件。如果您的截止日期雲端機群使用與工作站不同的作業系統，或者您的套件有其他主機需求，則您有下列選項：

- 在符合目標作業系統的主機`rattler-build publish`上執行 。例如，使用執行 的 Amazon Elastic Compute Cloud (Amazon EC2) 執行個體Linux來建置Linux機群的套件。
- 使用截止日期雲端套件建置佇列來自動化目標平台上的建置。請參閱建立套件建置佇列。
- （進階）使用跨編譯為工作站的不同平台建置套件。如需詳細資訊，請參閱 rattler-build 文件中的跨編譯。

後續步驟

將套件發佈至 Amazon S3 Conda 頻道後，請將截止日期雲端佇列設定為使用該頻道：

- 設定自訂 conda 套件的生產佇列許可 – 授予生產佇列對 Amazon S3 conda 頻道的唯讀存取權。
- 將 conda 頻道新增至佇列環境 – 設定佇列環境以從 Amazon S3 conda 頻道安裝套件。

設定自訂 conda 套件的生產佇列許可

您的生產佇列需要佇列 S3 儲存貯體中/Conda字首的唯讀許可。開啟與生產佇列相關聯之角色的 AWS Identity and Access Management (IAM) 頁面，並使用下列內容修改政策：

1. 開啟截止日期雲端主控台，並導覽至套件建置佇列的佇列詳細資訊頁面。
2. 選擇佇列服務角色，然後選擇編輯佇列。
3. 捲動至佇列服務角色區段，然後在 IAM 主控台中選擇檢視此角色。
4. 從許可政策清單中，為您的佇列選擇 AmazonDeadlineCloudQueuePolicy。
5. 從許可索引標籤中，選擇編輯。
6. 將新區段新增至佇列服務角色，如下所示。將 *amzn-s3-demo-bucket* 和 *111122223333* 取代為您自己的儲存貯體和帳戶。

```
{
   "Effect": "Allow",
   "Sid": "CustomCondaChannelReadOnly",
   "Action": [
    "s3:GetObject",
    "s3:ListBucket"
   ],
   "Resource": [
    "arn:aws:s3:::amzn-s3-demo-bucket",
    "arn:aws:s3:::amzn-s3-demo-bucket/Conda/*"
   ],
   "Condition": {
    "StringEquals": {
     "aws:ResourceAccount": "111122223333"
    }
   }
  },
```

將 conda 頻道新增至佇列環境

若要使用 S3 conda 頻道，您需要將`s3://amzn-s3-demo-bucket/Conda/Default`頻道位置新增至您提交至截止日期雲端的任務`CondaChannels`參數。Deadline Cloud 提供的提交者會提供欄位來指定自訂 conda 頻道和套件。

您可以編輯生產佇列的 conda 佇列環境，以避免修改每個任務。請執行下列步驟：

1. 開啟截止日期雲端主控台，並導覽至生產佇列的佇列詳細資訊頁面。
2. 選擇環境索引標籤。
3. 選取 Conda 佇列環境，然後選擇編輯。
4. 選擇 JSON 編輯器，然後在指令碼中尋找 的參數定義`CondaChannels`。
5. 編輯該行，`default: "deadline-cloud"`使其從新建立的 S3 conda 頻道開始：

```
default: "s3://amzn-s3-demo-bucket/Conda/Default deadline-cloud"
```

服務受管機群預設會為 conda 啟用彈性的頻道優先順序。對於請求 `blender=4.5` 4.5 Blender 同時位於新頻道和`deadline-cloud`頻道的任務，將從頻道清單中的第一個頻道提取套件。如果在第一個通道中找不到指定的套件版本，則會檢查後續通道的套件版本。

對於客戶管理的機群，您可以使用 Deadline Cloud 範例GitHub儲存庫中的其中一個 conda 佇列環境範例來啟用 conda 套件的使用。

為應用程式或外掛程式建立 conda 套件

conda 套件是以任何語言撰寫的軟體壓縮封存。Conda 支援各種作業系統和架構組合，因此您可以封裝完整的應用程式Blender，例如 Maya、 以及 Python 和其他語言的Nuke程式庫。如需 conda 套件的詳細資訊，請參閱 conda 文件中的套件。

若要使用 conda 套件，請將它安裝到虛擬環境中。conda 虛擬環境具有安裝套件的字首目錄。安裝套件會在支援時使用檔案的硬連結或重新連結，因此使用相同套件建立多個環境不會使用顯著的額外磁碟空間。若要使用虛擬環境，請啟用它來設定環境變數。啟用會執行套件提供的指令碼，讓每個套件都有機會修改 PATH 或其他環境變數。Conda 套件通常包含應用程式或程式庫，但彈性啟用表示也可以指向安裝在共用檔案系統上的應用程式。

建立自訂套件包含三個階段：配方包含建置指示、套件是建置成品 (`.conda` 或 `.tar.bz2` 檔案），以及頻道託管套件以供安裝。`rattler-build publish` 命令會處理這三個步驟：它可以將配方建置到套件中並發佈到頻道，也可以直接採用套件成品來發佈它。

conda-forge 社群會維護各種開放原始碼軟體的套件配方，並在`conda-forge`頻道中託管套件成品。您可以設定佇列以包含 `conda-forge` 做為套件來源，然後建置取決於要執行之 conda-forge 套件的自訂套件。針對 Linux，conda-forge 託管完整的編譯器工具鏈，包括 CUDA 支援，並選取一致的編譯和連結選項。您可以在自己的配方中使用 conda-forge 套件做為相依性，或在相同的環境中將其與自訂套件一起安裝。

您可以將整個應用程式結合到 conda 套件中，包括相依性。Deadline Cloud 在截止日期雲端通道中為服務受管機群提供的套件使用此二進位重新封裝方法。這會組織與 安裝相同的檔案，以符合 conda 虛擬環境。

> **Note**
>
> 大型應用程式可能需要數十 GB 的可用磁碟空間，才能進行來源封存、解壓縮檔案和建置輸出。請確定您使用的磁碟有足夠的可用空間供套件建置輸出使用。

封裝應用程式

為 conda 重新封裝應用程式時，有兩個目標：

- 應用程式的大多數檔案應與主要 conda 虛擬環境結構分開。然後，環境可以將應用程式與 Conda-forge 等其他來源的套件混合。
- 啟用 conda 虛擬環境時，應用程式應可從 PATH 環境變數取得。

為 conda 重新封裝應用程式

1. 將安裝應用程式的 conda 建置配方寫入 等子目錄中`$CONDA_PREFIX/opt/<application-name>`。這會將它與標準字首目錄分開，例如 `bin`和 `lib`。
2. 將符號連結或啟動指令碼新增至 `$CONDA_PREFIX/bin`以執行應用程式二進位檔。

   或者，建立 `conda activate`命令將執行的 activate.d 指令碼，將應用程式二進位目錄新增至 PATH。在 上Windows，在所有可以建立環境的地方都不支援符號連結，請改用應用程式啟動或 activate.d 指令碼。
3. 有些應用程式依賴於在截止日期雲端服務受管機群上預設未安裝的程式庫。例如，非互動式任務通常不需要 X11 視窗系統，但某些應用程式仍需要在沒有圖形界面的情況下執行。您必須在您建立的套件內提供這些相依性。
4. 如果應用程式支援外掛程式，請提供外掛程式套件應遵循的明確慣例，以在虛擬環境中與應用程式整合。例如，Maya2026 年範例配方會記錄此Maya外掛程式慣例。
5. 請務必遵循您所封裝應用程式的著作權和授權合約。建議您的 conda 頻道使用私有 Amazon S3 儲存貯體，以控制對陣列的分佈和限制套件存取。

`deadline-cloud` 頻道中套件的範例配方可在 上的截止日期雲端範例儲存庫中取得GitHub。

封裝外掛程式

應用程式外掛程式可以封裝為自己的 conda 套件。建立外掛程式套件時，請遵循下列準則：

- 在組建配方 中包含主機應用程式套件做為組建和執行相依性`recipe.yaml`。使用版本限制條件，以便建置配方僅與相容的套件一起安裝。
- 遵循主機應用程式套件慣例來註冊外掛程式。

轉接器套件

某些截止日期雲端應用程式整合使用擴展應用程式界面的轉接器，以簡化撰寫任務範本。轉接器是一種命令列界面，支援執行背景協助程式、報告狀態和套用路徑映射。如需詳細資訊，請參閱 上的開啟

任務描述轉接器執行期GitHub。例如， 上的 deadline-cloud-for-maya GitHub包含整合的任務提交 GUI 和Maya轉接器，可在服務受管機群上做為maya-openjd套件使用。

來自截止日期雲端提交者 GUIs 的任務提交包含CondaPackages參數值，指定要包含在虛擬環境中以執行任務的 conda 套件。的CondaPackages參數值Maya通常看起來類似maya=2026.* maya-openjd=0.15.* maya-mtoa，而且可能包含外掛程式套件的替代項目。當佇列環境設定用於執行任務的 conda 虛擬環境時，它會將這些套件名稱和版本限制解析為相容，並新增其需要執行的所有相依性套件。每個轉接器和外掛程式套件都會指定其相容的項目，包括 的哪些版本Maya、Python 的哪些版本，以及其他相依性。

若要使用我們的範例建置自己的轉接器套件，例如 上的 maya-openjd 配方GitHub，您可以建置 Python 套件和 conda-forge 提供的其他相依性。您可能需要先建立截止日期和 openjd-adaptor-runtime 配方，以滿足相依性。

建立 的 conda 組建配方 Blender

Blender 可免費使用且易於與 conda 封裝，這使得它成為學習如何為 AWS 截止日期雲端 （ 截止日期雲端 ） 建立 conda 套件的理想起點。Foundation 為多個作業系統Blender提供應用程式封存。Deadline Cloud 範例儲存庫中的 Blender 4.5 範例配方會將這些封存GitHub套件封裝為 conda 套件。

了解配方

recipe.yaml 檔案定義 rattler-build 範本語法中的套件中繼資料、來源 URLs 和建置選項。 https://rattler-build.prefix.dev/latest/reference/recipe_file/#spec-reference配方會指定版本編號一次，並根據作業系統提供不同的來源 URLs。

中的 build區段recipe.yaml會關閉二進位重新定位和動態共用物件 (DSO) 連結檢查。這些選項控制套件在任何目錄字首安裝到 conda 虛擬環境時的運作方式。build 區段中使用的預設值旨在分別封裝每個相依性程式庫，但當二進位重新封裝應用程式時，您需要對其進行變更。 Blender 不需要任何 RPATH 調整，因為應用程式封存是以可重新定位性為考量而建置。如需新增可重新定位性的範例，請參閱建立 Maya 的 conda 配方。

在套件建置期間，build.sh 或 build_win.sh 指令碼會執行 ，將檔案安裝到環境中。這些指令碼會將安裝檔案複製到 $PREFIX/opt/blender、從 $PREFIX/bin （ 在 上Linux) 建立符號連結，以及設定設定環境變數的啟用指令碼，例如 BLENDER_LOCATION。在 上Windows，啟用指令碼會將Blender目錄新增至 PATH，而不是建立符號連結。

Windows 建置指令碼使用 bash而非 cmd.exe .bat 檔案，以跨平台保持一致性。您可以為 安裝 git Windows 以提供bash套件建置。

配方也包含 deadline-cloud.yaml 檔案，指定 Conda 平台和中繼資料，用於將自動化套件建置任務提交至截止日期雲端。如需詳細資訊，請參閱提交套件建置任務。

建置Blender套件

使用 rattler-build publish建置 Blender 4.5 配方，並將套件發佈至頻道。您可以發佈至本機檔案系統頻道進行測試，或直接發佈至 Amazon S3 頻道以供生產使用。如果您在本機完成建置和測試套件中的設定，請從 conda_recipes目錄執行下列命令。

```
rattler-build publish blender-4.5/recipe/recipe.yaml \
    --to file://$HOME/my-conda-channel \
    --build-number=+1
```

對於其他發佈選項：

- 若要發佈至 Amazon S3 頻道，請參閱將套件發佈至 S3 conda 頻道。
- 若要使用截止日期雲端套件建置佇列自動化建置，請參閱使用截止日期雲端自動化套件建置。

使用Blender轉譯任務測試套件

建置 Blender 4.5 套件之後，您可以使用轉譯任務進行測試。如果您沒有Blender場景，請從Blender示範檔案頁面下載 Blender 3.5 - Cozy Kitchen 場景。Deadline Cloud 範例儲存庫包含blender_render任務套件和 Conda 佇列環境，可用於本機和雲端測試。

在本機測試

您可以使用 Open Job Description CLI 在工作站上執行任務範本。使用 安裝 CLIpip。

```
pip install openjd-cli
```

從範例儲存庫的 job_bundles目錄中，執行下列命令。將 */path/to/scene.blend* 取代為Blender場景檔案的路徑。

```
openjd run blender_render/template.yaml \
    --environment ../queue_environments/conda_queue_env_pyrattler.yaml \
    -p CondaPackages=blender=4.5 \
    -p CondaChannels=file://$HOME/my-conda-channel \
    -p BlenderSceneFile=/path/to/scene.blend \
```

```
    -p Frames=1
```

--environment 選項會套用 conda 佇列環境，這會使用 中指定的套件來建立 conda 虛擬環境CondaPackages。CondaChannels 參數會告知佇列環境在何處尋找套件。如果您發佈至 Amazon S3 頻道而非本機頻道，請將file://路徑取代為您的s3://頻道 URL。

在截止日期雲端進行測試

將生產佇列設定為使用 Amazon S3 conda 頻道後，您可以將轉譯任務提交至截止日期雲端。從範例儲存庫的 job_bundles目錄中，執行下列命令。

```
deadline bundle submit blender_render \
    -p CondaPackages=blender=4.5 \
    -p BlenderSceneFile=/path/to/scene.blend \
    -p Frames=1
```

使用截止日期雲端監視器來追蹤任務的進度。在監視器中，選取任務的任務，然後選擇檢視日誌。選取啟動 Conda 工作階段動作，以確認在 Amazon S3 頻道中找到套件。

建立 的 conda 組建配方 Autodesk Maya

相較於開放原始碼應用程式，例如 等商業應用程式Autodesk Maya引入了額外的封裝需求，例如 Blender。Blender 配方會在開放原始碼授權下封裝簡單的可重新定位封存。商業應用程式通常透過安裝程式分發，並且需要授權管理組態。

商業應用程式的考量事項

封裝商業應用程式時，適用下列考量事項。詳細資訊說明每個 如何套用至 Maya。

- 授權 – 了解應用程式的授權權利和限制。您可能需要設定授權管理系統。閱讀Autodesk有關 雲端權利的訂閱優勢常見問答集，以了解 的雲端權利Maya。 Autodesk產品依賴通常需要管理員存取權才能設定ProductInformation.pit的檔案。精簡型用戶端的產品功能提供可重新定位的替代方案。如需詳細資訊，請參閱 Maya 和 MotionBuilder 的精簡型用戶端授權。
- 系統程式庫相依性 – 有些應用程式依賴於未安裝在服務受管機群工作者主機上的程式庫。 Maya 依賴於程式庫，包括 freetype 和 fontconfig。當這些程式庫可在系統套件管理員中使用時，例如 dnf AL2023 的 ，您可以使用套件管理員做為來源。由於 RPM 套件並非建置為可重新定位，因此您需要使用 等工具patchelf來解析Maya安裝字首中的相依性。
- 安裝的管理員存取權 – 有些安裝程式需要管理員存取權。服務受管機群不提供管理員存取權，因此您需要在個別系統上安裝應用程式，並建立套件建置的檔案封存。的Windows安裝程式Maya需要此

方法。配方中的 README.md 使用新啟動的 Amazon Elastic Compute Cloud (Amazon EC2) 執行個體記錄可重複的程序。

- 外掛程式整合 – 範例Maya套件定義 MAYA_NO_HOME=1將應用程式與使用者層級組態隔離，並將模組搜尋路徑新增至 ，MAYA_MODULE_PATH以便外掛程式套件可以在虛擬環境中放置.mod檔案。如需完整的外掛程式整合慣例，請參閱 Maya 2026 年範例配方。

了解配方

recipe.yaml 檔案以 rattler-build 範本語法定義套件中繼資料。檢閱 檔案的下列區段：

- source – 參考安裝程式封存，包括 sha256 雜湊。在 上Linux，來源是Autodesk安裝程式封存。在 上Windows，來源同時包含安裝程式封存和來自 的cleanMayaForCloud.py指令碼Autodesk，Maya以準備雲端部署。當您變更來源檔案時更新雜湊，例如封裝新版本時。
- build – 關閉預設二進位重新定位和 DSO 連結檢查，因為自動機制無法對 Maya 使用的程式庫和二進位目錄正常運作。在 上Linux，配方包含 patchelf作為手動設定相對 RPATHs建置相依性。
- 關於 – 用於瀏覽或處理 conda 頻道內容之應用程式的中繼資料。

組建指令碼 (https：//build.sh for Linux、build_win.sh for Windows) 包含說明每個步驟的註解。指令碼會執行下列關鍵任務：

- 解壓縮安裝程式 – 將Maya安裝檔案解壓縮至 conda 字首。由於安裝程式格式， Linux和 Windows指令碼的處理方式不同。如需詳細資訊，請參閱建置指令碼。
- 安裝系統程式庫相依性 – 在 上Linux，指令碼會下載和擷取Maya需要但服務受管機群主機上不存在的系統程式庫。指令碼會將這些程式庫複製到 Mayalib目錄中，以便在 conda 環境中使用。
- 使用 patchelf 設定相對RPATHs：在 上Linux，指令碼會使用 patchelf --add-rpath將 $ORIGIN-相對路徑新增至共用程式庫。此方法遵循 conda 建議，絕不可在 conda LD_LIBRARY_PATH 環境中使用。指令碼會在多個目錄層級 (lib、lib/python*/site-packages、lib/python*/lib-dynload) 修補程式庫，讓每個程式庫都能找到其相對於自己位置的相依性。配方遵循的最佳實務是設定 DT_RUNPATH而非 DT_RPATH，這可讓 在需要時LD_LIBRARY_PATH覆寫搜尋路徑以進行偵錯。
- 設定精簡型用戶端授權 – 指令碼會如 所述設定精簡型用戶端授權Autodesk，讓ProductInformation.pit檔案可以位於 conda 環境中，而不需要系統層級的管理員存取權。
- 設定啟用指令碼 – 指令碼會建立啟用和停用設定環境變數的指令碼MAYA_NO_HOME，包括 MAYA_LOCATION、MAYA_VERSION、 和 MAYA_MODULE_PATH。在 上Windows，指令碼會產生 .sh和 .bat啟用檔案，因為截止日期雲端範例佇列環境會使用 bash 在 上啟用環境Windows。

建置Maya套件

建置Maya套件之前，請從Autodesk您的帳戶下載Maya安裝程式。對於 Linux，將封存直接放入 `conda_recipes/archive_files`目錄。對於 Windows，請遵循 https：//README.md 中的程序來建立封存。

使用 `rattler-build publish`建置和發佈套件。Maya 配方需要 `patchelf`做為 的建置相依性Linux，可從 conda-forge 取得。新增 `-c conda-forge`以在建置期間提供相依性。從 `conda_recipes`目錄中，執行下列命令。

```
rattler-build publish maya-2026/recipe/recipe.yaml \
    --to file://$HOME/my-conda-channel \
    --build-number=+1 \
    -c conda-forge
```

對於其他發佈選項：

- 若要發佈至 Amazon S3 頻道，請參閱將套件發佈至 S3 conda 頻道。
- 若要使用截止日期雲端套件建置佇列自動化建置，請參閱使用截止日期雲端自動化套件建置。若要同時建置 Linux和 Windows套件，請使用 `--all-platforms`選項搭配 `submit-package-job`指令碼。

若要使用 Maya和 轉譯轉盤範例Arnold，請同時建置MtoA外掛程式和Maya轉接器套件。發佈所有三個套件後，您可以使用 轉盤和 Maya/Arnold 任務套件，從截止日期雲端範例儲存庫提交測試轉譯任務。請參閱使用 Maya 轉譯任務測試套件。

建立Maya轉接器的 conda 組建配方

`maya-openjd` 套件提供Maya與 AWS 截止日期雲端 （ 截止日期雲端 ） 任務提交整合的轉接器。當您使用截止日期雲端提交器 GUI 提交Maya轉譯任務時， `CondaPackages` 參數會與`maya`套件`maya-openjd`一起包含 。轉接器會在任務工作階段期間處理啟動 Maya、通訊轉譯參數和管理應用程式生命週期。如需轉接器的詳細資訊，請參閱轉接器套件。

了解配方

Maya-openjd 範例配方會從發佈至 PyPI 的 deadline-cloud-for-maya 來源套件建置轉接器。recipe.yaml 會使用 安裝套件`pip`至 conda 環境。

配方取決於 Python 和來自 Deadline Cloud 範例儲存庫的其他兩個套件，您需要先建置這些套件：

- 截止日期 – 截止日期雲端用戶端程式庫。
- openjd-adaptor-runtime – Open Job Description 轉接器執行時間。

Python 和其他相依性可從 conda-forge 取得，因此當您建置轉接器套件時，請將 `-c conda-forge`新增至 `rattler-build publish`命令。

建置轉接器套件

`maya-openjd` 套件取決於來自截止日期雲端範例儲存庫的其他兩個套件。從 `conda_recipes`目錄依序建置這三個套件。每個命令上的`-c conda-forge`選項是滿足 Python 和 Python 程式庫的配方相依性。

建置 `deadline`套件。

```
rattler-build publish deadline/recipe/recipe.yaml \
    --to file://$HOME/my-conda-channel \
    --build-number=+1 \
    -c conda-forge
```

建置 `openjd-adaptor-runtime`套件。

```
rattler-build publish openjd-adaptor-runtime/recipe/recipe.yaml \
    --to file://$HOME/my-conda-channel \
    --build-number=+1 \
    -c conda-forge
```

建置 `maya-openjd`套件。

```
rattler-build publish maya-openjd/recipe/recipe.yaml \
    --to file://$HOME/my-conda-channel \
    --build-number=+1 \
    -c conda-forge
```

對於其他發佈選項：

- 若要發佈至 Amazon S3 頻道，請參閱將套件發佈至 S3 conda 頻道。
- 若要使用截止日期雲端套件建置佇列自動化建置，請參閱使用截止日期雲端自動化套件建置。

建立Autodesk Maya to Arnold (MtoA)外掛程式的 conda 組建配方

Maya to Arnold (MtoA) 外掛程式會將Arnold轉譯器新增為 中的選項Maya。MtoA 範例配方示範如何將外掛程式封裝為與主機應用程式套件整合的個別 conda 套件。

了解配方

recipe.yaml 會針對建置和執行需求，指定對maya套件的相依性。此相依性使用版本限制，因此外掛程式僅安裝相容的Maya版本。

配方使用與Maya配方相同的來源封存。建置指令碼會在Maya套件在 中設定的$PREFIX/usr/autodesk/maya$MAYA_VERSION/modules目錄中安裝Arnold和MtoA建立 mtoa.mod 檔案MAYA_MODULE_PATH。 Maya並使用相同的授權技術，因此Maya套件已包含 Arnold 所需的授權資訊。

建置MtoA套件

在建置Maya套件之前建置MtoA套件，因為 MtoA 取決於Maya建置時間。使用 rattler-build publish建置和發佈套件。從 conda_recipes目錄中，執行下列命令。

```
rattler-build publish maya-mtoa-2026/recipe/recipe.yaml \
    --to file://$HOME/my-conda-channel \
    --build-number=+1
```

rattler-build publish 命令使用目標頻道做為解析相依性時的最高優先順序頻道，因此您先前發佈的maya套件會自動可用。

對於其他發佈選項：

- 若要發佈至 Amazon S3 頻道，請參閱將套件發佈至 S3 conda 頻道。
- 若要使用截止日期雲端套件建置佇列自動化建置，請參閱使用截止日期雲端自動化套件建置。

使用Maya轉譯任務測試套件

建置 Maya、 MtoA和 maya-openjd套件之後，您可以使用轉譯任務來測試它們。Deadline Cloud 範例儲存庫包含具有 Maya/ 任務套件的轉盤Arnold，可使用 Maya和 轉譯動畫Arnold。任務套件也會使用 FFmpeg 來編碼可從conda-forge頻道取得的影片。

在本機測試

您可以使用 Open Job Description CLI 在工作站上執行任務範本。使用 安裝 CLIpip。

```
pip install openjd-cli
```

從範例儲存庫中的 job_bundles目錄，執行下列命令。ErrorOnArnoldLicenseFail=false 參數Arnold會指示 使用浮水印轉譯，而不是在沒有可用的授權時失敗。

```
openjd run turntable_with_maya_arnold/template.yaml \
    --environment ../queue_environments/conda_queue_env_pyrattler.yaml \
    -p CondaPackages="maya maya-mtoa maya-openjd ffmpeg" \
    -p CondaChannels="file://$HOME/my-conda-channel conda-forge" \
    -p ErrorOnArnoldLicenseFail=false \
    -p FrameRange=1-5
```

--environment 選項會套用 conda 佇列環境，這會使用 中指定的套件來建立 conda 虛擬環境CondaPackages。CondaChannels 參數包含自訂套件的本機頻道和 conda-forge的本機頻道ffmpeg。如果您發佈至 Amazon S3 頻道而非本機頻道，請將file://路徑取代為您的s3://頻道URL。

當任務完成時，轉譯的輸出會位於 turntable_with_maya_arnold/output/目錄中。

在截止日期雲端進行測試

將生產佇列設定為使用 Amazon S3 conda 頻道之後，請將轉譯任務提交至截止日期雲端。將conda-forge頻道新增至 conda 佇列環境中的 CondaChannels 參數，以提供轉接器所需的來源ffmpeg和 Python 相依性。從範例儲存庫中的 job_bundles目錄，執行下列命令。

```
deadline bundle submit turntable_with_maya_arnold
```

使用截止日期雲端監視器來追蹤任務的進度。在監視器中，選取任務的任務，然後選擇檢視日誌。選取啟動 Conda maya工作階段動作，以確認在 Amazon S3 頻道中找到 maya-mtoa、 和 maya-openjd套件。

使用截止日期雲端自動化套件建置

對於 CI/CD 工作流程或當您需要為多個作業系統建置套件時，您可以建立截止日期雲端套件建置佇列。佇列排程會在您的機群上建置任務，這會建置套件並將其發佈到您的 Amazon Simple Storage Service (Amazon S3) Conda 頻道。這可簡化維護所有所需組態中軟體版本的持續套件建置。

您可以使用 AWS CloudFormation (CloudFormation) 範本，或從截止日期雲端主控台手動建立套件建置佇列。 CloudFormation 範本會使用生產佇列和已設定的套件建置佇列來部署完整的陣列。從主控台建立佇列可讓您進一步控制個別設定。

使用 建立套件建置佇列 CloudFormation

您可以使用 CloudFormation 範本來建立包含套件建置佇列的截止日期雲端陣列。範本會使用私有 Amazon S3 Conda 頻道設定生產佇列和套件建置佇列。

部署範本之前，請建立 Amazon S3 儲存貯體以保留任務附件和您的 conda 頻道。您可以從 Amazon S3 主控台建立儲存貯體。部署範本時，您需要儲存貯體名稱。

部署 CloudFormation 範本

1. 從 上的 Deadline Cloud 範例儲存庫下載 deadline-cloud-starter-farm-template.yaml 範本GitHub。
2. 從CloudFormation 主控台中，選擇建立堆疊，然後選擇使用新資源 （標準）。
3. 選取 選項以上傳範本檔案，然後上傳`deadline-cloud-starter-farm-template.yaml`檔案。
4. 輸入堆疊的名稱，例如 **`StarterFarm`**，並提供任務附件的 Amazon S3 儲存貯體名稱和 conda 頻道。
5. 請依照 CloudFormation 主控台步驟完成堆疊建立。

如需範本參數和自訂選項的詳細資訊，請參閱 上 Deadline Cloud 範例儲存庫中的入門陣列 README GitHub。

從主控台建立套件建置佇列

遵循在截止日期雲端使用者指南中建立佇列中的指示。進行下列變更：

- 在步驟 5 中，選擇現有的 Amazon S3 儲存貯體。指定根資料夾名稱，例如 ，**`DeadlineCloudPackageBuild`**以便建置成品與您一般的截止日期雲端附件保持獨立。
- 在步驟 6 中，您可以將套件建置佇列與現有機群建立關聯，或者，如果目前的機群不適合，則可以建立全新的機群。
- 在步驟 9 中，為套件建置佇列建立新的服務角色。您將修改許可，為佇列提供上傳套件和重新索引 conda 頻道所需的許可。

設定套件建置佇列許可

若要允許套件建置佇列存取佇列 Amazon S3 儲存貯體中的`/Conda`字首，您必須修改佇列的角色，以授予其讀取/寫入存取權。角色需要下列許可，套件建置任務才能上傳新套件並重新索引頻道。

- s3:GetObject
- s3:PutObject
- s3:ListBucket
- s3:GetBucketLocation
- s3:DeleteObject

1. 開啟截止日期雲端主控台，並導覽至套件建置佇列的佇列詳細資訊頁面。
2. 選擇佇列服務角色，然後選擇編輯佇列。
3. 捲動至佇列服務角色區段，然後在 IAM 主控台中選擇檢視此角色。
4. 從許可政策清單中，為您的佇列選擇 AmazonDeadlineCloudQueuePolicy。
5. 從許可索引標籤中，選擇編輯。
6. 將新區段新增至佇列服務角色，如下所示。將 *amzn-s3-demo-bucket* 和 *111122223333* 取代為您自己的儲存貯體和帳戶。

```
{
   "Effect": "Allow",
   "Sid": "CustomCondaChannelReadWrite",
   "Action": [
    "s3:GetObject",
    "s3:PutObject",
    "s3:DeleteObject",
    "s3:ListBucket",
    "s3:GetBucketLocation"
   ],
   "Resource": [
    "arn:aws:s3:::amzn-s3-demo-bucket",
    "arn:aws:s3:::amzn-s3-demo-bucket/Conda/*"
   ],
   "Condition": {
    "StringEquals": {
     "aws:ResourceAccount": "111122223333"
    }
   }
  },
```

提交套件建置任務

建立套件建置佇列並設定佇列許可後，您可以提交任務來建置 conda 套件。上的 Deadline Cloud 範例儲存庫中的`submit-package-job`指令碼會GitHub提交 conda 配方的建置任務。

您需要下列項目：

- 工作站上安裝的截止日期雲端 CLI。
- 作用中AWS 的截止日期雲端監視器（截止日期雲端監視器）登入工作階段。
- 截止日期雲端範例儲存庫的複製。

提交套件建置任務

1. 開啟截止日期雲端組態 GUI，並將預設陣列和佇列設定為套件建置佇列。

   ```
   deadline config gui
   ```

2. 變更為範例儲存庫中的 `conda_recipes` 目錄。

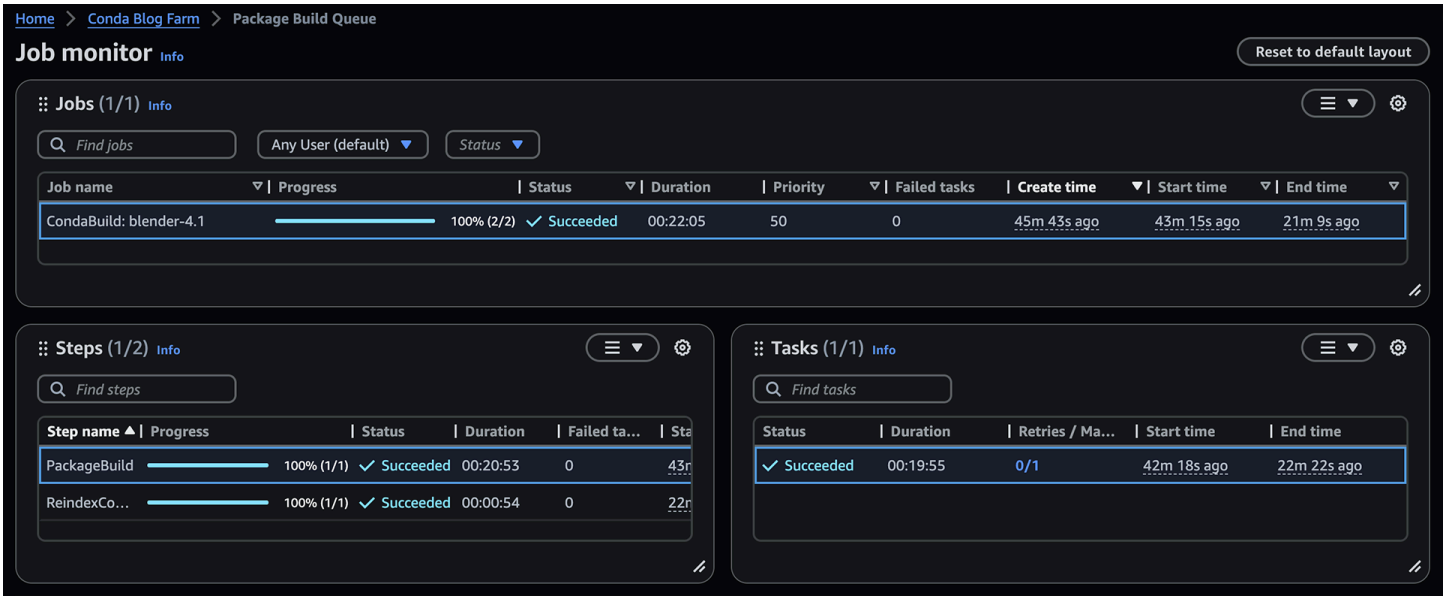
   ```
   cd deadline-cloud-samples/conda_recipes
   ```

3. 使用配方目錄執行`submit-package-job`指令碼。下列範例會建置 Blender 4.5 配方。

   ```
   ./submit-package-job blender-4.5/
   ```

   如果配方需要您尚未下載的來源封存，指令碼會提供下載指示。下載封存並再次執行指令碼。

提交任務後，請使用截止日期雲端監視器來檢視任務的進度和狀態。



監視器會顯示任務的兩個步驟：建置套件，然後重新索引 conda 頻道。當您在套件建置步驟的任務上按一下滑鼠右鍵並選擇檢視日誌時，監視器會顯示工作階段動作：

- 同步附件 – 複製輸入任務附件或掛載虛擬檔案系統。
- 啟動 Conda – 佇列環境動作。建置任務不會指定 conda 套件，因此此動作會快速完成。
- 啟動 CondaBuild Env – 使用建置 conda 套件和重新索引頻道所需的軟體建立 conda 虛擬環境。
- 任務執行 – 建置套件並將結果上傳至 Amazon S3。

當動作執行時，他們會將日誌傳送至 Amazon CloudWatch (CloudWatch)。當任務完成時，選取檢視所有任務的日誌，以查看有關環境設定和縮減的其他日誌。

執行具有管理員權限的主機組態指令碼

主機組態指令碼可讓您在服務受管機群工作者上執行管理任務，例如軟體安裝。這些指令碼會以更高的權限執行 (sudo在 上為 Linux，在 上為 管理員Windows)，讓您可以靈活地為系統設定工作者。

Deadline Cloud 會在工作者進入 `STARTING` 狀態之後，以及在執行任何任務之前執行指令碼。

> **Important**
>
> 指令碼以更高的許可執行。您有責任確保指令碼不會造成任何安全問題。
> 當您使用主機組態指令碼時，您必須負責監控機群的運作狀態。

主機組態指令碼的常見用途包括：

- 安裝需要管理員存取權的軟體
- 安裝Docker容器
- 安裝第三方雲端儲存解決方案，例如 LucidLink。如需逐步解說，請參閱適用於 M&E 的 部落格上的使用適用於截止日期雲端的服務受管機群指令碼設定 LucidLink。 AWS

您可以使用 主控台或使用 建立和更新主機組態指令碼 AWS CLI。

Console

1. 在機群詳細資訊頁面上，選擇組態索引標籤。
2. 在指令碼欄位中，輸入要以更高許可執行的指令碼。您可以選擇匯入以從工作站載入指令碼。
3. 設定執行指令碼的逾時期間，以秒為單位。預設值為 300 秒 (5 分鐘)。
4. 選擇儲存變更以儲存指令碼。

Create with CLI

使用下列 AWS CLI 命令來建立具有主機組態指令碼的機群。將####文字取代為您的資訊。

```
aws deadline create-fleet \
--farm-id farm-12345 \
--display-name "fleet-name" \
--max-worker-count 1 \
--configuration '{
"serviceManagedEc2": {
  "instanceCapabilities": {
    "vCpuCount": {"min": 2},
    "memoryMiB": {"min": 4096},
    "osFamily": "linux",
    "cpuArchitectureType": "x86_64"
  },
  "instanceMarketOptions": {"type":"spot"}
}
}' \
--role-arn arn:aws:iam::111122223333:role/role-name \
--host-configuration '{ "scriptBody": "script body", "scriptTimeoutSeconds": timeout
 value}'
```

Update with CLI

使用下列 AWS CLI 命令來更新機群的主機組態指令碼。將####文字取代為您的資訊。

```
aws deadline update-fleet \
--farm-id farm-12345 \
--fleet-id fleet-455678 \
--host-configuration '{ "scriptBody": "script body", "scriptTimeoutSeconds": timeout
 value}'
```

下列指令碼示範：

- 指令碼可用的環境變數
- 該 AWS 登入資料正在 shell 中運作
- 指令碼正在提升的 shell 中執行

Linux

使用下列指令碼來顯示指令碼正在執行並具有 `root` 權限：

```
# Print environment variables
set
# Check AWS Credentials
aws sts get-caller-identity
```

Windows

使用下列PowerShell指令碼來顯示指令碼正在以管理員權限執行：

```
Get-ChildItem env: | ForEach-Object { "$($_.Name)=$($_.Value)" }
aws sts get-caller-identity
function Test-AdminPrivileges {
  $currentUser = New-Object
 Security.Principal.WindowsPrincipal([Security.Principal.WindowsIdentity]::GetCurrent())
  $isAdmin =
 $currentUser.IsInRole([Security.Principal.WindowsBuiltInRole]::Administrator)

  return $isAdmin
}
```

```
if (Test-AdminPrivileges) {
  Write-Host "The current PowerShell session is elevated (running as
 Administrator)."
} else {
  Write-Host "The current PowerShell session is not elevated (not running as
 Administrator)."
}
exit 0
```

主機組態指令碼故障診斷

當您執行主機組態指令碼時：

- 成功時：工作者執行任務
- 失敗時（非零結束碼或當機）：
  - 工作者關閉

  機群會使用最新的主機組態指令碼自動啟動新的工作者

若要監控指令碼：

1. 在截止日期雲端主控台中開啟機群頁面。
2. 選擇檢視工作者以開啟截止日期雲端監視器。
3. 在監控頁面中檢視工作者狀態。

> **Tip**
>
> 測試主機組態指令碼時，請將機群的最大工作者計數設定為 1，以避免在指令碼上反覆運算時啟動多個工作者。

重要說明：

- 因錯誤而關閉的工作者無法在監視器中的工作者清單中使用。使用 CloudWatch Logs 在下列日誌群組中檢視工作者日誌：

```
/aws/deadline/farm-XXXXX/fleet-YYYYY
```

在該日誌群組中，尋找名為 的串流worker-*ZZZZZ*。

- CloudWatch Logs 會根據您設定的保留期間來保留工作者日誌。

監控主機組態指令碼執行

透過主機組態指令碼，您可以完全控制截止日期雲端工作者。您可以安裝任何軟體套件、重新設定作業系統參數，或掛載共用檔案系統。透過此進階功能和 Deadline Cloud 擴展到數千個工作者的能力，您可以監控組態指令碼何時成功執行或失敗。

我們建議您使用下列解決方案來監控主機組態指令碼執行。

CloudWatch Logs 監控

所有機群主機組態日誌都會串流到機群的 CloudWatch 日誌群組，特別是工作者的 CloudWatch 日誌串流。例如， `/aws/deadline/farm-123456789012/fleet-777788889999`是陣列 `123456789012`、機群 的日誌群組`777788889999`。

每個工作者都會佈建專用日誌串流，例如 `worker-123456789012`。主機組態日誌包含日誌橫幅，例如執行主機組態指令碼和完成執行主機組態指令碼，結束程式碼：0。指令碼的結束程式碼包含在完成的橫幅中，可以使用 CloudWatch 工具進行查詢。

CloudWatch Logs 洞察

CloudWatch Logs Insights 提供進階功能來分析日誌資訊。例如，下列 Log Insights 查詢剖析主機組態結束程式碼，依時間排序：

```
fields @timestamp, @message, @logStream, @log
| filter @message like /Finished running Host Configuration Script/
| parse @message /exit code: (?<exit_code>\d+)/
| display @timestamp, exit_code
| sort @timestamp desc
```

如需有關 CloudWatch Logs Insights 的詳細資訊，請參閱《Amazon CloudWatch Logs 使用者指南》中的使用 CloudWatch Logs Insights 分析日誌資料。

工作者客服人員結構化記錄

Deadline Cloud 的工作者代理程式會將結構化 JSON 日誌發佈至 CloudWatch。工作者代理程式提供各種結構化日誌，用於分析工作者運作狀態。如需詳細資訊，請參閱 GitHub 上的截止日期雲端工作者代理程式記錄。

結構化日誌的屬性會解壓縮至 Log Insights 中的欄位。您可以使用此 CloudWatch 功能來計算和分析主機組態啟動失敗。例如，計數和 bin 查詢可用來判斷失敗發生的頻率：

```
fields @timestamp, @message, @logStream, @log
| sort @timestamp desc
| filter message like /Worker Agent host configuration failed with exit code/
| stats count(*) by exit_code, bin(1h)
```

指標和警示的 CloudWatch 指標篩選條件

您可以設定 CloudWatch 指標篩選條件，從日誌產生 CloudWatch 指標。指標篩選條件可讓您建立警示和儀表板，以監控主機組態指令碼執行。

建立指標篩選條件

1. 開啟 CloudWatch 主控台。
2. 在導覽窗格中，選擇日誌，然後選擇日誌群組。
3. 選取機群的日誌群組。
4. 選擇 Create metric filter (建立指標篩選條件)。
5. 使用下列其中一項定義您的篩選條件模式：

   - 對於成功指標：

     ```
     {$.message = "*Worker Agent host configuration succeeded.*"}
     ```

   - 對於失敗指標：

     ```
     {$.exit_code != 0 && $.message = "*Worker Agent host configuration failed with
      exit code*"}
     ```

6. 選擇下一步以建立具有下列值的指標：

   - 指標命名空間：您的指標命名空間 （例如 **MyDeadlineFarm**)
   - 指標名稱：您請求的指標名稱 （例如 **host_config_failure**)
   - 指標值： **1** （每個執行個體的計數為 1)
   - 預設值：保留空白
   - 單位： **Count**

建立指標篩選條件後，您可以設定標準 CloudWatch 警示，以針對更高的主機組態失敗率採取動作，或將指標新增至 CloudWatch 儀表板以進行day-to-day操作和監控。

如需詳細資訊，請參閱《Amazon CloudWatch Logs 使用者指南》中的篩選和模式語法。

將軟體授權與截止日期雲端搭配使用

Deadline Cloud 提供兩種為您的任務提供軟體授權的方法：

- 以用量為基礎的授權 (UBL) – 根據機群使用處理任務的時數來追蹤和計費。沒有固定的授權數量，因此您的機群可以視需要擴展。UBL 是服務受管機群的標準。對於客戶管理的機群，您可以連接 UBL 的截止日期雲端授權端點。UBL 為您的截止日期雲端工作者提供授權以進行轉譯，它不會為您的 DCC 應用程式提供授權。
- 自備授權 (BYOL) – 可讓您將現有的軟體授權與服務或客戶管理的機群搭配使用。您可以使用 BYOL 連線到截止日期雲端用量型授權不支援之軟體的授權伺服器。您可以透過連線至自訂授權伺服器，將 BYOL 與服務受管機群搭配使用。

主題

結合 BYOL 和 UBL

您可以結合 BYOL 和 UBL，讓您的工作者先使用您現有的授權，並在您的 BYOL 授權用盡時自動回到截止日期雲端用量型授權。當您的現有授權數量有限，但在尖峰工作負載期間需要擴展超過該容量時，此方法非常有用。

合併授權的運作方式

當您設定合併授權時，佇列環境會設定授權環境變數，以便在 UBL 授權端點之前列出 BYOL 授權伺服器。大多數第三方應用程式會依照授權伺服器出現在環境變數中的順序進行檢查。當工作者請求授權時，應用程式會先聯絡您的 BYOL 授權伺服器。如果沒有可用的 BYOL 授權，應用程式會回到 UBL 授權端點。

中提供的 BYOL 佇列環境範本會自動將服務受管機群連接至自訂授權伺服器設定此備用行為。佇列環境中的 Python 指令碼會在您的 BYOL 授權伺服器地址前面加上現有的 UBL 授權環境變數。若要使用合併授權，請在您要備用行為的產品的佇列環境指令碼中保留 UBL 區段。

若要針對特定產品僅使用不含 UBL 備用的 BYOL，請從指令碼中移除該產品的 UBL 區段，並將授權環境變數直接新增至佇列環境的 variables區段。例如，若要僅針對電影 4D 使用 BYOL，請從指令碼中移除電影 4D 區段，並g_licenseServerRLM: 127.0.0.1:7057新增至 variables區段。

範例：搭配 UBL 備用使用 BYOL Cinema 4D 授權

考慮在其內部部署網路的授權伺服器上擁有現有 Cinema 4D 授權的工作室。Studio 想要將這些授權用於截止日期雲端轉譯，但也想要在所有 BYOL 授權都使用時返回 UBL，以擴展超出其授權計數的範圍。

若要設定此設定，請遵循中的步驟，將服務受管機群連接至自訂授權伺服器並對佇列環境範本進行下列變更：

使用 UBL 備用設定 BYOL Cinema 4D 授權

1. 將 LicenseInstanceId 參數設定為有權存取 Cinema 4D 授權伺服器的授權伺服器或代理的 Amazon Elastic Compute Cloud (Amazon EC2) 執行個體 ID。
2. 設定 LicensePorts 參數以包含連接埠 7057（電影 4D RLM 授權連接埠）。
3. 在 Python 指令碼中，保留將 BYOL 伺服器附加至 UBL 組態的 Cinema 4D 區段：

   ```
   # Cinema4D
   os.environ["g_licenseServerRLM"] = f"127.0.0.1:7057;
   {os.environ.get('g_licenseServerRLM', '')}"
   print(f"openjd_env: g_licenseServerRLM={os.environ['g_licenseServerRLM']}")
   ```

   此組態g_licenseServerRLM會設定為 127.0.0.1:7057;*UBL_endpoint*:7057。電影 4D 會127.0.0.1:7057先檢查 BYOL 伺服器。如果沒有可用的授權，則 Cinema 4D 會回到 UBL 端點。
4. 移除您未使用之產品的區段（例如 Arnold、Nuke 或 SideFX），以保持組態乾淨。

如果您也有其他產品只使用 BYOL 而不使用 UBL 備用，請將這些授權環境變數直接新增至佇列環境的 variables區段，並從 Python 指令碼中移除對應的 區段。

合併授權的考量事項

當您使用合併授權時，請記住下列考量事項：

- 有些應用程式不支援單一環境變數中的多個授權伺服器。例如，V-Ray 會改用 XML 組態檔案。佇列環境範本會分別處理 V-Ray 組態。如需詳細資訊，請參閱 中佇列環境範本中的 V-Ray 區段將服務受管機群連接至自訂授權伺服器。
- 環境變數中授權伺服器的順序決定優先順序。首先列出 BYOL 伺服器，以便在 UBL 授權之前使用您現有的授權。
- 在Windows工作者上，使用分號 (;) 分隔環境變數中的授權伺服器項目，而非冒號 ():。如需設定授權環境變數的詳細資訊，請參閱 將客戶受管機群連接至授權端點。
- 若要搭配客戶受管機群使用 UBL 備用，請設定授權端點。如需詳細資訊，請參閱將客戶受管機群連接至授權端點。

將服務受管機群連接至自訂授權伺服器

您可以攜帶自己的授權伺服器，以與截止日期雲端服務受管機群搭配使用。若要取得自己的授權，您可以使用陣列中的佇列環境來設定授權伺服器。若要設定授權伺服器，您應該已設定陣列和佇列。

連線至軟體授權伺服器的方式取決於機群的組態和軟體廠商的需求。一般而言，您可以透過下列兩種方式之一存取伺服器：

- 直接傳送至授權伺服器。您的工作者會使用網際網路從軟體廠商的授權伺服器取得授權。您的所有工作者都必須能夠連線到伺服器。
- 透過授權代理。您的工作者連線到本機網路中的代理伺服器。只有代理伺服器才能透過網際網路連線至廠商的授權伺服器。

使用下列指示，您可以使用 Amazon EC2 Systems Manager (SSM) 將連接埠從工作者執行個體轉送到您的授權伺服器或代理執行個體。在以下範例中，如果您的授權伺服器無法提供授權，則會使用 Deadline Cloud 的用量型授權。在耗盡授權之後，移除不適用於您不想使用用量型授權之管道或產品的區段。

主題

步驟 1：設定佇列環境

您可以在佇列中設定佇列環境來存取授權伺服器。首先，請確定您已使用下列其中一種方法來設定具有授權伺服器存取權的 AWS 執行個體：

- 授權伺服器 – 執行個體會直接託管授權伺服器。
- 授權代理 – 執行個體具有授權伺服器的網路存取權，並將授權伺服器連接埠轉送至授權伺服器。如需如何設定授權代理執行個體的詳細資訊，請參閱 步驟 2：（選用）授權代理執行個體設定。

如需設定授權環境變數的詳細資訊，請參閱 步驟 3：將轉譯應用程式連接到端點。對於自訂授權伺服器設定，授權伺服器地址會保持 localhost，而不是 Amazon VPC 端點。

將必要的許可新增至佇列角色

1. 從截止日期雲端主控台中，選擇前往儀表板。
2. 從儀表板中選取陣列，然後選取您要設定的佇列。
3. 從佇列詳細資訊 > 服務角色中，選取角色。
4. 選擇新增許可，然後選擇建立內嵌政策。
5. 選取 JSON 政策編輯器，然後將下列文字複製並貼到編輯器中。

   JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "",
            "Effect": "Allow",
            "Action": [
                "ssm:StartSession"
            ],
            "Resource": [
                "arn:aws:ssm:us-east-1::document/AWS-
StartPortForwardingSession",
                "arn:aws:ec2:us-east-1:111122223333:instance/instance_id"
            ]
        }
    ]
}
```

6. 儲存新政策之前，請在政策文字中取代下列值：

- region 將 取代為您陣列所在的 AWS 區域
- instance_id 將 取代為您正在使用的授權伺服器或代理執行個體的執行個體 ID
- account_id 以包含您陣列 AWS 的帳號取代

7. 選擇下一步。
8. 針對政策名稱，輸入 **LicenseForwarding**。
9. 選擇建立政策以儲存您的變更，並建立具有所需許可的政策。

將新的佇列環境新增至佇列

1. 如果您尚未前往儀表板，請從截止日期雲端主控台選擇前往儀表板。
2. 從儀表板中選取陣列，然後選取您要設定的佇列。
3. 選擇佇列環境 **>** 動作 **>** 使用 YAML 建立新的 。
4. 將下列文字複製並貼到 YAML 指令碼編輯器。

   Windows

```
specificationVersion: "environment-2023-09"
parameterDefinitions:
 - name: LicenseInstanceId
   type: STRING
   description: >
    The Instance ID of the license server/proxy instance
   default: ""
 - name: LicenseInstanceRegion
   type: STRING
   description: >
    The region containing this farm
   default: ""
 - name: LicensePorts
   type: STRING
   description: >
    Comma-separated list of ports to be forwarded to the license server/proxy
    instance. Example: "2701,2702,7075,2703,6101,1715,1716,1717,7054,7055,30304"
   default: "2701,2702,7075,2703,6101,1715,1716,1717,7054,7055,30304"
environment:
 name: BYOL License Forwarding
 variables:
```

```
  example_LICENSE: 2701@localhost
 script:
  actions:
   onEnter:
    command: bash
    args: [ "{{Env.File.Enter}}" ]
   onExit:
    command: bash
    args: [ "{{Env.File.Exit}}" ]
  embeddedFiles:
   - name: Enter
     type: TEXT
     runnable: True
     data: |
      curl "https://s3.amazonaws.com/session-manager-downloads/plugin/latest/
windows/SessionManagerPlugin.zip" -o "{{Session.WorkingDirectory}}/ssm-
plugin.zip"
      powershell -Command "Expand-Archive -Path '{{Session.WorkingDirectory}}/
ssm-plugin.zip' -DestinationPath '{{Session.WorkingDirectory}}/ssm-plugin'
 -Force; Expand-Archive -Path '{{Session.WorkingDirectory}}/ssm-plugin/
package.zip' -DestinationPath '{{Session.WorkingDirectory}}/ssm-plugin/package'
 -Force"
      conda activate
      python "{{Env.File.StartSession}}" "{{Session.WorkingDirectory}}/ssm-
plugin/package/bin/session-manager-plugin.exe"
   - name: Exit
     type: TEXT
     runnable: True
     data: |
      echo Killing SSM Manager Plugin PIDs: $BYOL_SSM_PIDS
      for pid in ${BYOL_SSM_PIDS//,/ }; do kill $pid; done
   - name: StartSession
     type: TEXT
     data: |
      import boto3
      import json
      import subprocess
      import sys
      import os
      import tempfile

      instance_id = "{{Param.LicenseInstanceId}}"
      region = "{{Param.LicenseInstanceRegion}}"
      license_ports_list = "{{Param.LicensePorts}}".split(",")
```

```
      ssm_client = boto3.client("ssm", region_name=region)
      pids = []

      for port in license_ports_list:
        session_response = ssm_client.start_session(
          Target=instance_id,
          DocumentName="AWS-StartPortForwardingSession",
          Parameters={"portNumber": [port], "localPortNumber": [port]}
        )

        cmd = [
          sys.argv[1],
          json.dumps(session_response),
          region,
          "StartSession",
          "",
          json.dumps({"Target": instance_id}),
          f"https://ssm.{region}.amazonaws.com"
        ]

        process = subprocess.Popen(cmd, stdout=subprocess.DEVNULL,
 stderr=subprocess.DEVNULL)
        pids.append(process.pid)
        print(f"SSM Port Forwarding Session started for port {port}")

      print(f"openjd_env: BYOL_SSM_PIDS={','.join(str(pid) for pid in pids)}")

      # Enabling UBL after the BYOL has run out requires prepending the BYOL
 configuration to the existing license setup
      # Remove the sections that do not apply to your pipeline, or you do not
 want to use UBL after exhausting the BYOL licenses.
      # The port numbers used may not match what your license server is serving.

      # Arnold
      os.environ["ADSKFLEX_LICENSE_FILE"] = f"2701@localhost;
{os.environ.get('ADSKFLEX_LICENSE_FILE', '')}"
      print(f"openjd_env:
 ADSKFLEX_LICENSE_FILE={os.environ['ADSKFLEX_LICENSE_FILE']}")

      # Cinema4D
      os.environ["g_licenseServerRLM"] = f"localhost:7057;
{os.environ.get('g_licenseServerRLM', '')}"
```

```
      print(f"openjd_env:
 g_licenseServerRLM={os.environ['g_licenseServerRLM']}")

      # Nuke
      os.environ["foundry_LICENSE"] = f"6101@localhost;
{os.environ.get('foundry_LICENSE', '')}"
      print(f"openjd_env: foundry_LICENSE={os.environ['foundry_LICENSE']}")

      # SideFX
      os.environ["SESI_LMHOST"] = f"localhost:1715;
{os.environ.get('SESI_LMHOST', '')}"
      print(f"openjd_env: SESI_LMHOST={os.environ['SESI_LMHOST']}")

      # Redshift and Red Giant
      os.environ["redshift_LICENSE"] = f"7054@localhost;7055@localhost;
{os.environ.get('redshift_LICENSE', '')}"
      print(f"openjd_env: redshift_LICENSE={os.environ['redshift_LICENSE']}")

      # V-Ray doesn't support multiple license servers in a single environment
 variable
      # See https://documentation.chaos.com/space/LIC5/125050770/Sharing+a
+License+Configuration+in+a+Network
      vray_license = os.environ.get('VRAY_AUTH_CLIENT_SETTINGS', '')
      xml_content = """<VRLClient>
        <LicServer>
          <Host>localhost</Host>
          <Port>30304</Port>"""

      if vray_license and vray_license.startswith('licset://'):
          server_parts = vray_license.removeprefix('licset://').split(':')
          if len(server_parts) >= 2:
              xml_content += f"""
          <Host1>{server_parts[0]}</Host1>
          <Port1>{server_parts[1]}</Port1>"""

      xml_content += """
          <User></User>
          <Pass></Pass>
        </LicServer>
      </VRLClient>"""

      temp_dir = tempfile.gettempdir()
      xml_path = os.path.join(temp_dir, 'vrlclient.xml')
```

```
      with open(xml_path, 'w') as f:
          f.write(xml_content)

      os.environ["VRAY_AUTH_CLIENT_FILE_PATH"] = temp_dir
      print(f"openjd_env:
 VRAY_AUTH_CLIENT_FILE_PATH={os.environ['VRAY_AUTH_CLIENT_FILE_PATH']}")

      # Clear the existing VRAY_AUTH_CLIENT_SETTINGS so only the vrlclient.xml
 file is used.
      os.environ["VRAY_AUTH_CLIENT_SETTINGS"] = ''
      print(f"openjd_env:
 VRAY_AUTH_CLIENT_SETTINGS={os.environ['VRAY_AUTH_CLIENT_SETTINGS']}")

      # Print out the created xml file's contents
      print(f"V-Ray configuration file: {xml_path}")
      with open(xml_path, 'r') as f:
          print(f"{f.read()}")
```

Linux

```
specificationVersion: "environment-2023-09"
parameterDefinitions:
 - name: LicenseInstanceId
   type: STRING
   description: >
    The Instance ID of the license server/proxy instance
   default: ""
 - name: LicenseInstanceRegion
   type: STRING
   description: >
    The region containing this farm
   default: ""
 - name: LicensePorts
   type: STRING
   description: >
    Comma-separated list of ports to be forwarded to the license server/proxy
    instance. Example: "2701,2702,7075,2703,6101,1715,1716,1717,7054,7055,30304"
   default: "2701,2702,7075,2703,6101,1715,1716,1717,7054,7055,30304"
environment:
 name: BYOL License Forwarding
 variables:
```

```
  example_LICENSE: 2701@localhost
 script:
  actions:
   onEnter:
    command: bash
    args: [ "{{Env.File.Enter}}" ]
   onExit:
    command: bash
    args: [ "{{Env.File.Exit}}" ]
  embeddedFiles:
   - name: Enter
     type: TEXT
     runnable: True
     data: |
      curl https://s3.amazonaws.com/session-manager-downloads/plugin/
latest/linux_64bit/session-manager-plugin.rpm -Ls | rpm2cpio - | cpio -iv
 --to-stdout ./usr/local/sessionmanagerplugin/bin/session-manager-plugin >
 {{Session.WorkingDirectory}}/session-manager-plugin
      chmod +x {{Session.WorkingDirectory}}/session-manager-plugin
      conda activate
      python {{Env.File.StartSession}} {{Session.WorkingDirectory}}/session-
manager-plugin
   - name: Exit
     type: TEXT
     runnable: True
     data: |
      echo Killing SSM Manager Plugin PIDs: $BYOL_SSM_PIDS
      for pid in ${BYOL_SSM_PIDS//,/ }; do kill $pid; done
   - name: StartSession
     type: TEXT
     data: |
      import boto3
      import json
      import subprocess
      import sys
      import os
      import tempfile

      instance_id = "{{Param.LicenseInstanceId}}"
      region = "{{Param.LicenseInstanceRegion}}"
      license_ports_list = "{{Param.LicensePorts}}".split(",")

      ssm_client = boto3.client("ssm", region_name=region)
      pids = []
```

```
      for port in license_ports_list:
        session_response = ssm_client.start_session(
          Target=instance_id,
          DocumentName="AWS-StartPortForwardingSession",
          Parameters={"portNumber": [port], "localPortNumber": [port]}
        )

        cmd = [
          sys.argv[1],
          json.dumps(session_response),
          region,
          "StartSession",
          "",
          json.dumps({"Target": instance_id}),
          f"https://ssm.{region}.amazonaws.com"
        ]

        process = subprocess.Popen(cmd, stdout=subprocess.DEVNULL,
 stderr=subprocess.DEVNULL)
        pids.append(process.pid)
        print(f"SSM Port Forwarding Session started for port {port}")

      print(f"openjd_env: BYOL_SSM_PIDS={','.join(str(pid) for pid in pids)}")

      # Enabling UBL after the BYOL has run out requires prepending the BYOL
 configuration to the existing license setup
      # Remove the sections that do not apply to your pipeline, or you do not
 want to use UBL after exhausting the BYOL licenses.
      # The port numbers used may not match what your license server is serving.

      # Arnold
      os.environ["ADSKFLEX_LICENSE_FILE"] = f"2701@localhost:
{os.environ.get('ADSKFLEX_LICENSE_FILE', '')}"
      print(f"openjd_env:
 ADSKFLEX_LICENSE_FILE={os.environ['ADSKFLEX_LICENSE_FILE']}")

      # Nuke
      os.environ["foundry_LICENSE"] = f"6101@localhost:
{os.environ.get('foundry_LICENSE', '')}"
      print(f"openjd_env: foundry_LICENSE={os.environ['foundry_LICENSE']}")

      # SideFX
```

```
      os.environ["SESI_LMHOST"] = f"localhost:1715;
{os.environ.get('SESI_LMHOST', '')}"
      print(f"openjd_env: SESI_LMHOST={os.environ['SESI_LMHOST']}")

      # Redshift and Red Giant
      os.environ["redshift_LICENSE"] = f"7054@localhost:7055@localhost:
{os.environ.get('redshift_LICENSE', '')}"
      print(f"openjd_env: redshift_LICENSE={os.environ['redshift_LICENSE']}")

      # V-Ray doesn't support multiple license servers in a single environment
 variable
      # See https://documentation.chaos.com/space/LIC5/125050770/Sharing+a
+License+Configuration+in+a+Network
      vray_license = os.environ.get('VRAY_AUTH_CLIENT_SETTINGS', '')
      xml_content = """<VRLClient>
        <LicServer>
          <Host>localhost</Host>
          <Port>30304</Port>"""

      if vray_license and vray_license.startswith('licset://'):
          server_parts = vray_license.removeprefix('licset://').split(':')
          if len(server_parts) >= 2:
              xml_content += f"""
          <Host1>{server_parts[0]}</Host1>
          <Port1>{server_parts[1]}</Port1>"""

      xml_content += """
          <User></User>
          <Pass></Pass>
        </LicServer>
      </VRLClient>"""

      temp_dir = tempfile.gettempdir()
      xml_path = os.path.join(temp_dir, 'vrlclient.xml')

      with open(xml_path, 'w') as f:
          f.write(xml_content)

      os.environ["VRAY_AUTH_CLIENT_FILE_PATH"] = temp_dir
      print(f"openjd_env:
 VRAY_AUTH_CLIENT_FILE_PATH={os.environ['VRAY_AUTH_CLIENT_FILE_PATH']}")

      # Clear the existing VRAY_AUTH_CLIENT_SETTINGS so only the vrlclient.xml
 file is used.
```

```
    os.environ["VRAY_AUTH_CLIENT_SETTINGS"] = ''
    print(f"openjd_env:
VRAY_AUTH_CLIENT_SETTINGS={os.environ['VRAY_AUTH_CLIENT_SETTINGS']}")

    # Print out the created xml file's contents
    print(f"V-Ray configuration file: {xml_path}")
    with open(xml_path, 'r') as f:
        print(f"{f.read()}")
```

5. 儲存佇列環境之前，請視需要對環境文字進行下列變更：

   - 更新下列參數的預設值，以反映您的環境：
     - LicenseInstanceID – 授權伺服器或代理執行個體的 Amazon EC2 執行個體 ID
     - LicenseInstanceRegion – 包含您陣列 AWS 的區域
     - LicensePorts – 以逗號分隔的連接埠清單，以轉送至授權伺服器或代理執行個體 （ 例如 2700，2701)
   - 如果您想要在用盡自有授權 (BYOL) 後使用以用量為基礎的授權 (UBL)，請確定連接埠適用於您的授權伺服器。如果您在 BYOL 用盡之後不想使用 UBL，請將任何必要的授權環境變數新增至變數區段。

     這些變數應將 DCCs導向授權伺服器連接埠上的 localhost。例如，如果您的 Foundry 授權伺服器正在接聽連接埠 6101，您可以將變數新增為 **`foundry_LICENSE: 6101@localhost`**。
6. （選用） 您可以將優先順序設定為 0，也可以將其變更為在多個佇列環境中以不同的方式排序優先順序。
7. 選擇建立佇列環境以儲存新環境。

   設定佇列環境後，提交至此佇列的任務會從已設定的授權伺服器擷取授權。

步驟 2：（選用） 授權代理執行個體設定

除了使用授權伺服器之外，您也可以使用授權代理。若要建立授權代理，請建立具有授權伺服器網路存取權的新 Amazon Linux 2023 執行個體。如有需要，您可以使用 VPN 連線設定此存取權。如需詳細資訊，請參閱《Amazon VPC 使用者指南》中的 VPN 連線。

若要為截止日期雲端設定授權代理執行個體，請遵循此程序中的步驟。在此新執行個體上執行下列組態步驟，以啟用將授權流量轉送到您的授權伺服器

1. 若要安裝 HAProxy 套件，請輸入

```
sudo yum install haproxy
```

2. 使用下列內容更新 /etc/haproxy/haproxy.cfg 組態檔案的接聽授權伺服器區段：

   a. 將 LicensePort1 和 LicensePort2 取代為要轉送至授權伺服器的連接埠號碼。新增或移除逗號分隔值，以容納所需的連接埠數量。

   b. 以授權伺服器的主機名稱或 IP 地址取代 LicenseServerHost。

```
lobal
    log         127.0.0.1 local2
    chroot      /var/lib/haproxy
    user        haproxy
    group       haproxy
    daemon

defaults
    timeout queue           1m
    timeout connect         10s
    timeout client          1m
    timeout server          1m
    timeout http-keep-alive 10s
    timeout check           10s

listen license-server
     bind *:LicensePort1,*:LicensePort2
     server license-server LicenseServerHost
```

3. 若要啟用和啟動 HAProxy 服務，請執行下列命令：

```
sudo systemctl enable haproxy
sudo service haproxy start
```

完成這些步驟後，從轉送佇列環境傳送至 localhost 的授權請求應轉送至指定的授權伺服器。

步驟 3： CloudFormation 範本設定

您可以使用 CloudFormation 範本來設定整個陣列，以使用您自己的授權。

1. 修改下一個步驟中提供的範本，將任何必要的授權環境變數新增至 BYOLQueueEnvironment 下的變數區段。
2. 使用下列 CloudFormation 範本。

```
AWSTemplateFormatVersion: 2010-09-09
Description: "Create &ADC; resources for BYOL"

Parameters:
  LicenseInstanceId:
    Type: AWS::EC2::Instance::Id
    Description: Instance ID for the license server/proxy instance
  LicensePorts:
    Type: String
    Description: Comma-separated list of ports to forward to the license instance

Resources:
  JobAttachmentBucket:
    Type: AWS::S3::Bucket
    Properties:
      BucketName: !Sub byol-example-ja-bucket-${AWS::AccountId}-${AWS::Region}
      BucketEncryption:
        ServerSideEncryptionConfiguration:
          - ServerSideEncryptionByDefault:
              SSEAlgorithm: AES256

  Farm:
    Type: AWS::Deadline::Farm
    Properties:
      DisplayName: BYOLFarm

  QueuePolicy:
    Type: AWS::IAM::ManagedPolicy
    Properties:
      ManagedPolicyName: BYOLQueuePolicy
      PolicyDocument:
        Version: 2012-10-17
        Statement:
          - Effect: Allow
            Action:
              - s3:GetObject
              - s3:PutObject
              - s3:ListBucket
```

```
              - s3:GetBucketLocation
            Resource:
              - !Sub ${JobAttachmentBucket.Arn}
              - !Sub ${JobAttachmentBucket.Arn}/job-attachments/*
            Condition:
              StringEquals:
                aws:ResourceAccount: !Sub ${AWS::AccountId}
          - Effect: Allow
            Action: logs:GetLogEvents
            Resource: !Sub arn:aws:logs:${AWS::Region}:${AWS::AccountId}:log-
group:/aws/deadline/${Farm.FarmId}/*
          - Effect: Allow
            Action:
              - s3:ListBucket
              - s3:GetObject
            Resource:
              - "*"
            Condition:
              ArnLike:
                s3:DataAccessPointArn:
                  - arn:aws:s3:*:*:accesspoint/deadline-software-*
              StringEquals:
                s3:AccessPointNetworkOrigin: VPC

  BYOLSSMPolicy:
    Type: AWS::IAM::ManagedPolicy
    Properties:
      ManagedPolicyName: BYOLSSMPolicy
      PolicyDocument:
        Version: 2012-10-17
        Statement:
          - Effect: Allow
            Action:
              - ssm:StartSession
            Resource:
              - !Sub arn:aws:ssm:${AWS::Region}::document/AWS-
StartPortForwardingSession
              - !Sub arn:aws:ec2:${AWS::Region}:${AWS::AccountId}:instance/
${LicenseInstanceId}


  WorkerPolicy:
    Type: AWS::IAM::ManagedPolicy
    Properties:
```

```
      ManagedPolicyName: BYOLWorkerPolicy
      PolicyDocument:
        Version: 2012-10-17
        Statement:
          - Effect: Allow
            Action:
              - logs:CreateLogStream
            Resource: !Sub arn:aws:logs:${AWS::Region}:${AWS::AccountId}:log-
group:/aws/deadline/${Farm.FarmId}/*
            Condition:
              ForAnyValue:StringEquals:
                aws:CalledVia:
                  - deadline.amazonaws.com
          - Effect: Allow
            Action:
              - logs:PutLogEvents
              - logs:GetLogEvents
            Resource: !Sub arn:aws:logs:${AWS::Region}:${AWS::AccountId}:log-
group:/aws/deadline/${Farm.FarmId}/*


  QueueRole:
    Type: AWS::IAM::Role
    Properties:
      RoleName: BYOLQueueRole
      ManagedPolicyArns:
        - !Ref QueuePolicy
        - !Ref BYOLSSMPolicy
      AssumeRolePolicyDocument:
        Version: 2012-10-17
        Statement:
          - Effect: Allow
            Action:
              - sts:AssumeRole
            Principal:
              Service:
                - credentials.deadline.amazonaws.com
                - deadline.amazonaws.com
            Condition:
              StringEquals:
                aws:SourceAccount: !Sub ${AWS::AccountId}
              ArnEquals:
                aws:SourceArn: !Ref Farm
```

```
  WorkerRole:
    Type: AWS::IAM::Role
    Properties:
      RoleName: BYOLWorkerRole
      ManagedPolicyArns:
        - arn:aws:iam::aws:policy/AWSDeadlineCloud-FleetWorker
        - !Ref WorkerPolicy
      AssumeRolePolicyDocument:
        Version: 2012-10-17
        Statement:
          - Effect: Allow
            Action:
              - sts:AssumeRole
            Principal:
              Service: credentials.deadline.amazonaws.com


  Queue:
    Type: AWS::Deadline::Queue
    Properties:
      DisplayName: BYOLQueue
      FarmId: !GetAtt Farm.FarmId
      RoleArn: !GetAtt QueueRole.Arn
      JobRunAsUser:
        Posix:
          Group: ""
          User: ""
        RunAs: WORKER_AGENT_USER
      JobAttachmentSettings:
        RootPrefix: job-attachments
        S3BucketName: !Ref JobAttachmentBucket

  Fleet:
    Type: AWS::Deadline::Fleet
    Properties:
      DisplayName: BYOLFleet
      FarmId: !GetAtt Farm.FarmId
      MinWorkerCount: 1
      MaxWorkerCount: 2
      Configuration:
        ServiceManagedEc2:
          InstanceCapabilities:
            VCpuCount:
              Min: 4
```

```
              Max: 16
            MemoryMiB:
              Min: 4096
              Max: 16384
            OsFamily: LINUX
            CpuArchitectureType: x86_64
          InstanceMarketOptions:
            Type: on-demand
      RoleArn: !GetAtt WorkerRole.Arn

  QFA:
    Type: AWS::Deadline::QueueFleetAssociation
    Properties:
      FarmId: !GetAtt Farm.FarmId
      FleetId: !GetAtt Fleet.FleetId
      QueueId: !GetAtt Queue.QueueId

  CondaQueueEnvironment:
    Type: AWS::Deadline::QueueEnvironment
    Properties:
      FarmId: !GetAtt Farm.FarmId
      Priority: 5
      QueueId: !GetAtt Queue.QueueId
      TemplateType: YAML
      Template: |
        specificationVersion: 'environment-2023-09'
        parameterDefinitions:
        - name: CondaPackages
          type: STRING
          description: >
            This is a space-separated list of conda package match specifications to
 install for the job.
            E.g. "blender=3.6" for a job that renders frames in Blender 3.6.

            See https://docs.conda.io/projects/conda/en/latest/user-guide/concepts/
pkg-specs.html#package-match-specifications
          default: ""
          userInterface:
            control: LINE_EDIT
            label: Conda Packages
        - name: CondaChannels
          type: STRING
          description: >
```

```
            This is a space-separated list of conda channels from which to install
 packages. &ADC; SMF packages are
            installed from the "deadline-cloud" channel that is configured by
 &ADC;.

            Add "conda-forge" to get packages from the https://conda-forge.org/
 community, and "defaults" to get packages
            from Anaconda Inc (make sure your usage complies with https://
www.anaconda.com/terms-of-use).
          default: "deadline-cloud"
          userInterface:
            control: LINE_EDIT
            label: Conda Channels
        environment:
          name: Conda
          script:
            actions:
              onEnter:
                command: "conda-queue-env-enter"
                args: ["{{Session.WorkingDirectory}}/.env", "--packages",
 "{{Param.CondaPackages}}", "--channels", "{{Param.CondaChannels}}"]
              onExit:
                command: "conda-queue-env-exit"

  BYOLQueueEnvironment:
    Type: AWS::Deadline::QueueEnvironment
    Properties:
      FarmId: !GetAtt Farm.FarmId
      Priority: 10
      QueueId: !GetAtt Queue.QueueId
      TemplateType: YAML
      Template: !Sub |
        specificationVersion: "environment-2023-09"
        parameterDefinitions:
        - name: LicenseInstanceId
          type: STRING
          description: >
            The Instance ID of the license server/proxy instance
          default: ""
        - name: LicenseInstanceRegion
          type: STRING
          description: >
            The region containing this farm
          default: ""
```

```
        - name: LicensePorts
          type: STRING
          description: >
            Comma-separated list of ports to be forwarded to the license server/
proxy
            instance. Example:
 "2701,2702,7075,2703,6101,1715,1716,1717,7054,7055,30304"
          default: "2701,2702,7075,2703,6101,1715,1716,1717,7054,7055,30304"
        environment:
        name: BYOL License Forwarding
        variables:
          example_LICENSE: 2701@localhost
        script:
          actions:
          onEnter:
            command: bash
            args: [ "{{Env.File.Enter}}" ]
          onExit:
            command: bash
            args: [ "{{Env.File.Exit}}" ]
          embeddedFiles:
          - name: Enter
            type: TEXT
            runnable: True
            data: |
              curl https://s3.amazonaws.com/session-manager-downloads/plugin/
latest/linux_64bit/session-manager-plugin.rpm -Ls | rpm2cpio - | cpio -iv
 --to-stdout ./usr/local/sessionmanagerplugin/bin/session-manager-plugin >
 {{Session.WorkingDirectory}}/session-manager-plugin
              chmod +x {{Session.WorkingDirectory}}/session-manager-plugin
              conda activate
              python {{Env.File.StartSession}} {{Session.WorkingDirectory}}/
session-manager-plugin
          - name: Exit
            type: TEXT
            runnable: True
            data: |
              echo Killing SSM Manager Plugin PIDs: $BYOL_SSM_PIDS
              for pid in ${BYOL_SSM_PIDS//,/ }; do kill $pid; done
          - name: StartSession
            type: TEXT
            data: |
              import boto3
              import json
```

```
              import subprocess
              import sys
              import os
              import tempfile

              instance_id = "{{Param.LicenseInstanceId}}"
              region = "{{Param.LicenseInstanceRegion}}"
              license_ports_list = "{{Param.LicensePorts}}".split(",")

              ssm_client = boto3.client("ssm", region_name=region)
              pids = []

              for port in license_ports_list:
                session_response = ssm_client.start_session(
                  Target=instance_id,
                  DocumentName="AWS-StartPortForwardingSession",
                  Parameters={"portNumber": [port], "localPortNumber": [port]}
                )

                cmd = [
                  sys.argv[1],
                  json.dumps(session_response),
                  region,
                  "StartSession",
                  "",
                  json.dumps({"Target": instance_id}),
                  f"https://ssm.{region}.amazonaws.com"
                ]

                process = subprocess.Popen(cmd, stdout=subprocess.DEVNULL,
stderr=subprocess.DEVNULL)
                pids.append(process.pid)
                print(f"SSM Port Forwarding Session started for port {port}")

              print(f"openjd_env: BYOL_SSM_PIDS={','.join(str(pid) for pid in
pids)}")

              # Enabling UBL after the "bring your own license" (BYOL) has run out
requires prepending the BYOL configuration to the existing license setup
              # Remove the sections that do not apply to your pipeline, or you do
not want to use UBL after exhausting the BYOL licenses.
              # The port numbers used may not match what your license server is
serving.
```

```
                # Arnold
                os.environ["ADSKFLEX_LICENSE_FILE"] = f"2701@localhost:
{os.environ.get('ADSKFLEX_LICENSE_FILE', '')}"
                print(f"openjd_env:
 ADSKFLEX_LICENSE_FILE={os.environ['ADSKFLEX_LICENSE_FILE']}")

                # Nuke
                os.environ["foundry_LICENSE"] = f"6101@localhost:
{os.environ.get('foundry_LICENSE', '')}"
                print(f"openjd_env: foundry_LICENSE={os.environ['foundry_LICENSE']}")

                # SideFX
                os.environ["SESI_LMHOST"] = f"localhost:1715;
{os.environ.get('SESI_LMHOST', '')}"
                print(f"openjd_env: SESI_LMHOST={os.environ['SESI_LMHOST']}")

                # Redshift and Red Giant
                os.environ["redshift_LICENSE"] = f"7054@localhost:7055@localhost:
{os.environ.get('redshift_LICENSE', '')}"
                print(f"openjd_env:
 redshift_LICENSE={os.environ['redshift_LICENSE']}")

                # V-Ray doesn't support multiple license servers in a single
 environment variable
                # See https://documentation.chaos.com/space/LIC5/125050770/Sharing+a
+License+Configuration+in+a+Network
                vray_license = os.environ.get('VRAY_AUTH_CLIENT_SETTINGS', '')
                xml_content = """<VRLClient>
                  <LicServer>
                    <Host>localhost</Host>
                    <Port>30304</Port>"""

                if vray_license and vray_license.startswith('licset://'):
                    server_parts = vray_license.removeprefix('licset://').split(':')
                    if len(server_parts) >= 2:
                        xml_content += f"""
                    <Host1>{server_parts[0]}</Host1>
                    <Port1>{server_parts[1]}</Port1>"""

                xml_content += """
                    <User></User>
                    <Pass></Pass>
                  </LicServer>
                </VRLClient>"""
```

```
            temp_dir = tempfile.gettempdir()
            xml_path = os.path.join(temp_dir, 'vrlclient.xml')

            with open(xml_path, 'w') as f:
                f.write(xml_content)

            os.environ["VRAY_AUTH_CLIENT_FILE_PATH"] = temp_dir
            print(f"openjd_env:
VRAY_AUTH_CLIENT_FILE_PATH={os.environ['VRAY_AUTH_CLIENT_FILE_PATH']}")

            # Clear the existing VRAY_AUTH_CLIENT_SETTINGS so only the
vrlclient.xml file is used.
            os.environ["VRAY_AUTH_CLIENT_SETTINGS"] = ''
            print(f"openjd_env:
VRAY_AUTH_CLIENT_SETTINGS={os.environ['VRAY_AUTH_CLIENT_SETTINGS']}")

            # Print out the created xml file's contents
            print(f"V-Ray configuration file: {xml_path}")
            with open(xml_path, 'r') as f:
                print(f"{f.read()}")
```

3. 部署 CloudFormation 範本時，請提供下列參數：

   - 使用授權伺服器或代理執行個體的 Amazon EC2 執行個體 ID 更新 LicenseInstanceID
   - 使用以逗號分隔的連接埠清單更新 LicensePorts，以轉送至授權伺服器或代理執行個體 （例如 2700，2701)
   - 在範本**example_LICENSE: 2700@localhost**中取代 以新增授權環境變數

4. 部署 範本以使用您自己的授權功能來設定您的陣列。

將客戶受管機群連接至授權端點

AWS Deadline Cloud 用量型授權伺服器為特定第三方產品提供隨需授權。使用以用量為基礎的授權，您可以隨需付費。您只需按使用時間付費。以用量為基礎的授權為您的截止日期雲端工作者提供授權以進行轉譯，它不會為您的 DCC 應用程式提供授權。

截止日期雲端用量型授權伺服器可與任何機群類型搭配使用，只要截止日期雲端工作者可以與授權伺服器通訊即可。授權伺服器會在服務受管機群中自動設定。只有客戶管理的機群才需要下列設定。

若要建立授權伺服器，您需要陣列 VPC 的安全群組，以允許第三方授權的流量。

主題

步驟 1：建立安全群組

使用 Amazon VPC 主控台為陣列的 VPC 建立安全群組。設定安全群組以允許下列傳入規則：

- Autodesk Maya 和 Arnold – 2701 - 2702、TCP、IPv4、IPv6
- 電影 4D – 7057、TCP、IPv4、IPv6
- Foundry Nuke – 6101、TCP、IPv4、IPv6
- 紅色巨型 – 7055、TCP、IPV4
- Redshift – 7054、TCP、IPv4、IPv6
- SideFX Houdini、Mantra 和 Karma – 1715 - 1717、TCP、IPv4、IPv6
- VRay – 30304、TCP、IPV4

每個傳入規則的來源都是機群的工作者安全群組。

如需建立安全群組的詳細資訊，請參閱《Amazon Virtual Private Cloud 使用者指南》中的建立安全群組。

步驟 2：設定授權端點

授權端點可讓您存取第三方產品的授權伺服器。授權請求會傳送至授權端點。端點會將它們路由到適當的授權伺服器。授權伺服器會追蹤用量限制和權利。在截止日期中建立授權端點 雲端會在您的 VPC 中佈建 AWS PrivateLink 介面端點。這些端點會根據標準 AWS PrivateLink 定價計費。如需詳細資訊，請參閱 AWS PrivateLink 定價。

使用適當的許可，您可以建立授權端點。如需建立授權端點所需的政策，請參閱允許建立授權端點的政策。

您可以從 Deadline Cloud 主控台的儀表板建立授權端點。

1. 從左側導覽窗格中，選擇授權端點，然後選擇建立授權端點。

2. 在建立授權端點頁面中，完成下列操作：

   - 選取 VPC。
   - 選取包含您截止日期雲端工作者的子網路。您最多可以選擇 10 個子網路。
   - 選取您在步驟 1 中建立的安全群組。針對更複雜的案例，您最多可以選取 10 個安全群組。
   - （選用）選擇新增標籤並新增一或多個標籤。您最多可以新增 50 個標籤。
3. 選擇建立授權端點。當授權端點建立時，它會顯示在授權端點頁面上。
4. 從計量產品區段中，選擇新增產品，然後選取您要新增至授權端點的產品。選擇新增。

若要從授權端點移除產品，請在計量產品區段中，選取產品，然後選擇移除。在確認中，再次選擇移除。

步驟 3：將轉譯應用程式連接到端點

設定授權端點後，應用程式會使用與使用第三方授權伺服器相同的授權端點。您通常會將環境變數或其他系統設定，例如 Microsoft Windows 登錄機碼，設定為授權伺服器連接埠和地址，藉此設定應用程式的授權伺服器。

若要取得授權端點 DNS 名稱，請在主控台中選取授權端點，然後在 DNS 名稱區段中選擇複製圖示。

組態範例

Example– Autodesk Maya 和 Arnold

> **Note**
>
> 您可以同時使用 Autodesk Maya 和 Arnold。使用適用於 Autodesk Maya 的連接埠 2702 和適用於 Arnold 的連接埠 2701。

對於 Autodesk Maya，將環境變數`ADSKFLEX_LICENSE_FILE`設定為：

```
2702@VPC_Endpoint_DNS_Name
```

針對 Arnold，將環境變數`ADSKFLEX_LICENSE_FILE`設定為：

```
2701@VPC_Endpoint_DNS_Name
```

對於 Autodesk Maya 和 Arnold，將環境變數設定為ADSKFLEX_LICENSE_FILE：

```
2702@VPC_Endpoint_DNS_Name:2701@VPC_Endpoint_DNS_Name
```

> **Note**
>
> 對於Windows工作者，請使用分號（；）而非冒號（：）來分隔端點。

Example– 電影 4D

將環境變數g_licenseServerRLM設定為：

```
VPC_Endpoint_DNS_Name:7057
```

建立環境變數之後，您應該能夠使用類似以下的命令列轉譯映像：

```
"C:\Program Files\Maxon Cinema 4D 2025\Commandline.exe" -render ^
    "C:\Users\User\MyC4DFileWithRedshift.c4d" -frame 0 ^
    -oimage "C:\Users\Administrator\User\MyOutputImage.png
```

Example– Foundry Nuke

將環境變數foundry_LICENSE設定為：

```
6101@VPC_Endpoint_DNS_Name
```

若要測試授權是否正常運作，您可以在終端機中執行 Nuke：

```
~/nuke/Nuke14.0v5/Nuke14.0 -x
```

Example– 紅色巨型

將環境變數redshift_LICENSE設定為：

```
7055@VPC_Endpoint_DNS_Name
```

請注意，Red Giant 和 Redshift 具有相同的redshift_LICENSE環境變數。如果您想要同時使用這兩個應用程式，您可以將環境變數設定為：

```
7054@VPC_Endpoint_DNS_Name:7055@VPC_Endpoint_DNS_Name
```

> **Note**
>
> 對於Windows工作者，請使用分號（；）而非冒號（：）來分隔端點。

若要測試授權是否正常運作，請確定已安裝 After Effects 和 Red Giant。然後，您可以使用類似以下的命令來轉譯專案：

```
C:\Program Files\Adobe\Adobe After Effects 2025\Support Files\aerender.exe -comp "Comp
 1" -project
         C:\Users\MyUser\myAfterEffectsProjectUsingRedGiant.aep -output
           C:\Users\MyUser\myMovieWithRedGiant.mp4
```

Example– Redshift

將環境變數`redshift_LICENSE`設定為：

```
7054@VPC_Endpoint_DNS_Name
```

建立環境變數之後，您應該能夠使用類似以下的命令列轉譯映像：

```
C:\ProgramData\redshift\bin\redshiftCmdLine.exe ^
    C:\demo\proxy\RS_Proxy_Demo.rs ^
    -oip C:\demo\proxy\images
```

Example– SideFX Houdini、Mantra 和 Karma

執行以下命令：

```
/opt/hfs19.5.640/bin/hserver -S
 "http://VPC_Endpoint_DNS_Name:1715;http://VPC_Endpoint_DNS_Name:1716;http://
VPC_Endpoint_DNS_Name:1717;"
```

若要測試授權是否正常運作，您可以透過此命令轉譯 Houdini 場景：

```
/opt/hfs19.5.640/bin/hython ~/forpentest.hip -c "hou.node('/out/mantra1').render()"
```

Example– VRay

將環境變數VRAY_AUTH_CLIENT_SETTINGS設定為：

```
licset://VPC_Endpoint_DNS_Name:30304
```

將環境變數VRAY_AUTH_CLIENT_FILE_PATH設定為：

```
/null
```

若要測試授權是否正常運作，您可以使用類似以下的命令在 VRay 中轉譯映像：

```
/usr/Chaos/V-Ray/bin/vray -sceneFile=/root/my_scene.vrscene -display=0
```

步驟 4：刪除授權端點

刪除客戶受管機群時，請記得刪除授權端點。如果您未刪除授權端點，則會繼續向您收取 AWS PrivateLink 固定成本

您可以從截止日期雲端主控台的儀表板刪除授權端點。

1. 從左側導覽窗格中，選擇授權端點。
2. 選取您要刪除的端點，然後選擇刪除，然後再次選擇刪除以確認。

搭配截止日期雲端使用 AI 代理器

使用 AI 代理器在截止日期雲端中撰寫任務套件、開發 conda 套件和疑難排解任務。本主題說明什麼是 AI 代理器、有效使用這些代理器的重點，以及協助代理器了解截止日期雲端的資源。

AI 代理器是一種軟體工具，使用大型語言模型 (LLM) 自動執行任務。AI 代理器可以讀取和寫入檔案、執行命令，並根據意見回饋根據解決方案反覆執行。範例包括命令列工具，例如 Kiro 和 IDE 整合助理。

使用 AI 代理器的重點

下列重點可協助您在將 AI 代理器與截止日期雲端搭配使用時獲得更好的結果。

- 提供基礎 – AI 代理器在能夠存取相關文件、規格和範例時表現最佳。您可以透過將代理程式指向特定文件頁面、將現有的範例程式碼做為參考共用、將相關的開放原始碼儲存庫複製到本機工作區，以及提供第三方應用程式的文件來提供基礎。
- 指定成功條件 – 定義客服人員的預期成果和技術需求。例如，當您要求代理程式開發任務套件時，請指定任務輸入、參數和預期的輸出。如果您不確定規格，請要求客服人員先提議選項，然後一起精簡需求。
- 啟用意見回饋迴圈 – 當 AI 代理器可以測試其解決方案並接收意見回饋時，它們會更有效地迭代。請不要在第一次嘗試時預期可行的解決方案，而是讓代理程式能夠執行其解決方案並檢閱結果。當代理程式可以存取狀態更新、日誌和驗證錯誤時，此方法運作良好。例如，當您開發任務套件時，允許代理程式提交任務並檢閱日誌。
- 預期反覆 – 即使內容良好，客服人員也可以偏離正軌或做出不符合您環境的假設。觀察客服人員如何接近任務，並在整個過程中提供指引。如果客服人員遇到困難，請新增缺少的內容，透過指向特定日誌檔案來協助尋找錯誤、在發現需求時縮小需求範圍，並新增負面需求，以明確說明客服人員應該避免的情況。

客服人員內容的資源

下列資源可協助 AI 代理器了解截止日期雲端概念，並產生準確的輸出。

- 截止日期雲端模型內容協定 (MCP) 伺服器 – 對於支援模型內容協定的客服人員，截止日期雲端儲存庫包含截止日期雲端用戶端，其中包含用於與任務互動的 MCP 伺服器。
- AWS文件 MCP 伺服器 – 對於支援 MCP 的代理程式，請設定AWS文件 MCP 伺服器，讓代理程式直接存取AWS文件，包括截止日期雲端使用者指南和開發人員指南。

- 開啟任務描述規格 – GitHub 上的開啟任務描述規格會定義任務範本的結構描述。當客服人員需要了解任務範本的結構和語法時，請參考此儲存庫。
- deadline-cloud-samples – deadline-cloud-samples 儲存庫包含常見應用程式和使用案例的範例任務套件、conda 配方和CloudFormation範本。
- aws-deadline GitHub 組織 – aws-deadline GitHub 組織包含許多第三方應用程式的參考外掛程式，您可以用作其他整合的範例。

範例提示：撰寫任務套件

下列範例提示示範如何使用 AI 代理器建立任務套件，以訓練 LoRA （低排名調整） 轉接器來產生 AI 映像。提示說明先前討論的要點：它透過指向相關儲存庫來提供基礎、定義任務套件輸出的成功條件，並概述反覆開發的意見回饋迴圈。

```
Write a pair of job bundles for Deadline Cloud that use the diffusers Python library to
 train a LoRA adapter on a set of images and then generate images from it.

Requirements:
- The training job takes a set of JPEG images as input, uses an image description,
 LoRA rank, learning rate, batch size, and number of training steps as parameters, and
 outputs a `.safetensors` file.
- The generation job takes the `.safetensors` file as input and the number of images
 to generate, then outputs JPEG images. The jobs use Stable Diffusion 1.5 as the base
 model.
- The jobs run `diffusers` as a Python script. Install the necessary packages using
 conda by setting the job parameters:
    - `CondaChannels`: `conda-forge`
    - `CondaPackages`: list of conda packages to install

For context, clone the following repositories to your workspace and review their
 documentation and code:
- OpenJobDescription specification: https://github.com/OpenJobDescription/openjd-
specifications/blob/mainline/wiki/2023-09-Template-Schemas.md
- Deadline Cloud sample job bundles: https://github.com/aws-deadline/deadline-cloud-
samples/tree/mainline/job_bundles
- diffusers library: https://github.com/huggingface/diffusers

Read through the provided context before you start. To develop a job bundle, iterate
 with the following steps until the submitted job succeeds. If a step fails, update the
 job bundle and restart the loop:
1. Create a job bundle.
2. Validate the job template syntax: `openjd check`
```

```
3. Submit the job to Deadline Cloud: `deadline bundle submit`
4. Wait for the job to complete: `deadline job wait`
5. View the job status and logs: `deadline job logs`
6. Download the job output: `deadline job download-output`

To verify the training and generation jobs work together, iterate with the following
 steps until the generation job produces images that resemble the dog in the training
 data:
1. Develop and submit a training job using the training images in `./exdog`
2. Wait for the job to succeed then download its output.
3. Develop and submit a generation job using the LoRA adapter from the training job.
4. Wait for the job to succeed then download its output.
5. Inspect the generated images. If they resemble the dog in the training data, you're
 done. Otherwise, review the job template, job parameters, and job logs to identify and
 fix the issue.
```

監控AWS截止日期雲端

監控是維護截止日期雲端 （ 截止日期雲端 ） AWS和您AWS解決方案的可靠性、可用性和效能的重要部分。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。開始監控截止日期雲端之前，您應該建立監控計畫，其中包含下列問題的答案：

- 監控目標是什麼？
- 監控哪些資源？
- 監控這些資源的頻率為何？
- 將使用哪些監控工具？
- 誰將執行監控任務？
- 發生問題時應該通知誰？

AWS和 Deadline Cloud 提供可用來監控資源和回應潛在事件的工具。其中有些工具會為您執行監控，有些工具需要手動介入。您應該盡可能自動化監控任務。

- Amazon CloudWatch AWS會即時監控您的 AWS資源和您在 上執行的應用程式。您可以收集和追蹤指標、建立自訂儀板表，以及設定警示，在特定指標達到您指定的閾值時通知您或採取動作。例如，您可以讓 CloudWatch 追蹤 CPU 使用量或其他 Amazon EC2 執行個體指標，並在需要時自動啟動新的執行個體。如需詳細資訊，請參閱 Amazon CloudWatch 使用者指南。

  Deadline Cloud 有三個 CloudWatch 指標。
- Amazon CloudWatch Logs 可讓您監控、存放和存取來自 Amazon EC2 執行個體、CloudTrail 及其他來源的日誌檔案。CloudWatch Logs 可監控日誌檔案中的資訊，並在達到特定閾值時通知您。您也可以將日誌資料存檔在高耐用性的儲存空間。如需詳細資訊，請參閱 Amazon CloudWatch Logs 使用者指南。
- Amazon EventBridge 可用來自動化您的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。來自 AWS服務的事件會以近乎即時的方式交付至 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。如需詳細資訊，請參閱 Amazon EventBridge 使用者指南。
- AWS CloudTrail 會擷取由AWS您的帳戶發出或代表您的帳戶發出的 API 呼叫和相關事件，並將日誌檔案交付至您指定的 Amazon S3 儲存貯體。您可以識別呼叫的使用者和帳戶AWS、進行呼叫的來源 IP 地址，以及呼叫的時間。如需詳細資訊，請參閱「AWS CloudTrail 使用者指南」。

主題

- 使用 記錄 Deadline Cloud API 呼叫 AWS CloudTrail
- 使用 CloudWatch 進行監控
- 使用 管理截止日期雲端事件 Amazon EventBridge

使用 記錄 Deadline Cloud API 呼叫 AWS CloudTrail

Deadline Cloud 已與 整合AWS CloudTrail，此服務提供使用者、角色或 所採取動作的記錄 AWS 服務。CloudTrail 會將 的所有 API 呼叫擷取 Deadline Cloud 為事件。擷取的呼叫包括來自 Deadline Cloud 主控台的呼叫，以及對 Deadline Cloud API 操作的程式碼呼叫。使用 CloudTrail 收集的資訊，您可以判斷提出的請求 Deadline Cloud、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 事件歷史記錄為 AWS 區域中過去 90 天記錄的管理事件，提供可檢視、可搜尋、可下載且不可變的記錄。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 管理主控台 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。

Deadline Cloud CloudTrail 中的資料事件

資料事件提供在資源上或在資源中執行的資源操作的相關資訊 (例如，讀取或寫入 Amazon S3 物件)。這些也稱為資料平面操作。資料事件通常是大量資料的活動。根據預設，CloudTrail 不會記錄資料事件。CloudTrail 事件歷史記錄不會記錄資料事件。

資料事件需支付額外的費用。如需 CloudTrail 定價的詳細資訊，請參閱 AWS CloudTrail 定價。

您可以使用 CloudTrail 主控台 AWS CLI或 CloudTrail API 操作來記錄 Deadline Cloud 資源類型的資料事件。如需如何記錄資料事件的詳細資訊，請參閱 AWS CloudTrail 使用者指南中的使用 AWS 管理主控台　記錄資料事件和使用 AWS Command Line Interface記錄資料事件。

下表列出您可以記錄資料事件 Deadline Cloud 的資源類型。資料事件類型 (主控台) 資料行會顯示從 CloudTrail 主控台上的資料事件類型清單中選擇的值。resources.type 值欄會顯示值，您會在使用 AWS CLI 或 CloudTrail APIs 設定進階事件選取器時指定該`resources.type`值。記錄到 CloudTrail 的資料 API 資料行會針對資源類型顯示記錄到 CloudTrail 的 API 呼叫。

資料事件類型 (主控台)	resources.type 值	記錄到 CloudTrail 的資料 API
截止日期機群	`AWS::Deadline::Fleet`	• SearchWorkers
截止日期佇列	`AWS::Deadline::Fleet`	• SearchJobs
截止日期工作者	`AWS::Deadline::Worker`	• GetWorker • ListSessionsForWorker • UpdateWorkerSchedule

資料事件類型 (主控台)	resources.type 值	記錄到 CloudTrail 的資料 API
		• BatchGetJobEntity • ListWorkers
截止日期任務	AWS::Deadline::Job	• ListStepConsumers • UpdateTask • ListJobs • GetStep • ListSteps • GetJob • GetTask • GetSession • ListSessions • CreateJob • ListSessionActions • ListTasks • CopyJobTemplate • UpdateSession • UpdateStep • UpdateJob • ListJobParameterDefinitions • GetSessionAction • ListStepDependencies • SearchTasks • SearchSteps

您可以設定進階事件選取器來篩選 `eventName`、`readOnly` 和 `resources.ARN` 欄位，以僅記錄對您重要的事件。如需這些欄位的詳細資訊，請參閱AWS CloudTrail API 參考中的AdvancedFieldSelector。

Deadline Cloud CloudTrail 中的管理事件

管理事件提供有關在 資源上執行的管理操作的資訊 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。

AWS Deadline Cloud 會將下列 Deadline Cloud 控制平面操作記錄到 CloudTrail 做為管理事件。

- associate-member-to-farm
- associate-member-to-fleet
- associate-member-to-job
- associate-member-to-queue
- assume-fleet-role-for-read
- assume-fleet-role-for-worker
- assume-queue-role-for-read
- assume-queue-role-for-user
- assume-queue-role-for-worker
- create-budget
- create-farm
- create-fleet
- create-license-endpoint
- create-limit
- create-monitor
- create-queue
- create-queue-environment
- create-queue-fleet-association
- create-queue-limit-association
- create-storage-profile
- create-worker
- delete-budget
- delete-farm
- delete-fleet

- delete-license-endpoint
- delete-limit
- delete-metered-product
- delete-monitor
- delete-queue
- delete-queue-environment
- delete-queue-fleet-association
- delete-queue-limit-association
- delete-storage-profile
- delete-worker
- disassociate-member-from-farm
- disassociate-member-from-fleet
- disassociate-member-from-job
- disassociate-member-from-queue
- get-application-version
- get-budget
- get-farm
- get-feature-map
- get-fleet
- get-license-endpoint
- get-limit
- get-monitor
- get-queue
- get-queue-environment
- get-queue-fleet-association
- get-queue-limit-association
- get-sessions-statistics-aggregation
- get-storage-profile
- get-storage-profile-for-queue
- list-available-metered-products

- list-budgets
- list-farm-members
- list-farms
- list-fleet-members
- list-fleets
- list-job-members
- list-license-endpoints
- list-limit
- list-metered-products
- list-monitor
- list-queue-environments
- list-queue-fleet-associations
- list-queue-limit-associations
- list-queue-members
- list-queues
- list-storage-profiles
- list-storage-profiles-for-queue
- list-tags-for-resource
- put-metered-product
- start-sessions-statistics-aggregation
- tag-resource
- untag-resource
- update-budget
- update-farm
- update-fleet
- update-limit
- update-monitor
- update-queue
- update-queue-environment
- update-queue-fleet-association

- update-queue-limit-association
- update-storage-profile
- update-worker

Deadline Cloud 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 日誌檔案不是公有 API 呼叫的已排序堆疊追蹤，因此事件不會以任何特定順序顯示。

以下範例顯示的 CloudTrail 事件會示範 `CreateFarm` 操作。

```
{
    "eventVersion": "0",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "EXAMPLE-PrincipalID:EXAMPLE-Session",
        "arn": "arn:aws:sts::111122223333:assumed-role/EXAMPLE-UserName/EXAMPLE-
Session",
        "accountId": "111122223333",
        "accessKeyId": "EXAMPLE-accessKeyId",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "EXAMPLE-PrincipalID",
                "arn": "arn:aws:iam::111122223333:role/EXAMPLE-UserName",
                "accountId": "111122223333",
                "userName": "EXAMPLE-UserName"
            },
            "webIdFederationData": {},
            "attributes": {
                "mfaAuthenticated": "false",
                "creationDate": "2021-03-08T23:25:49Z"
            }
        }
    },
    "eventTime": "2021-03-08T23:25:49Z",
    "eventSource": "deadline.amazonaws.com",
    "eventName": "CreateFarm",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "192.0.2.0",
```

```
    "userAgent": "EXAMPLE-userAgent",
    "requestParameters": {
        "displayName": "example-farm",
        "kmsKeyArn": "arn:aws:kms:us-west-2:111122223333:key/111122223333",
        "X-Amz-Client-Token": "12abc12a-1234-1abc-123a-1a11bc1111a",
        "description": "example-description",
        "tags": {
            "purpose_1": "e2e"
            "purpose_2": "tag_test"
        }
    },
    "responseElements": {
        "farmId": "EXAMPLE-farmID"
    },
    "requestID": "EXAMPLE-requestID",
    "eventID": "EXAMPLE-eventID",
    "readOnly": false,
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333"
    "eventCategory": "Management",
}
```


JSON 範例顯示可協助您識別事件的 、 AWS 區域 IP 地址和其他「`requestParameters`」，例如「`displayName`」和「`kmsKeyArn`」。

如需有關 CloudTrail 記錄內容的資訊，請參閱《AWS CloudTrail 使用者指南》中的 CloudTrail record contents。

使用 CloudWatch 進行監控

Amazon CloudWatch (CloudWatch) 會收集原始資料，並將其處理為可讀且幾近即時的指標。您可以在 https：//https://console.aws.amazon.com/cloudwatch/ 開啟 CloudWatch 主控台，以檢視和篩選截止日期雲端指標。

這些統計資料會保留 15 個月，因此您可以存取歷史資訊，以更清楚 Web 應用程式或服務的效能。您也可以設定留意特定閾值的警示，當滿足這些閾值時傳送通知或採取動作。如需詳細資訊，請參閱 Amazon CloudWatch 使用者指南。

Deadline Cloud 有兩種類型的日誌：任務日誌和工作者日誌。任務日誌是當您以指令碼或 DCC 執行的方式執行執行日誌時。任務日誌可能會顯示資產載入、圖磚轉譯或找不到紋理等事件。

工作者日誌會顯示工作者代理程式程序。這些可能包括工作者客服人員啟動、自行註冊、報告進度、載入組態或完成任務等事項。

這些日誌的命名空間為 `/aws/deadline/*`。

對於截止日期雲端，工作者會將這些日誌上傳至 CloudWatch Logs。根據預設，日誌永遠不會過期。如果任務輸出大量資料，可能會產生額外費用。如需詳細資訊，請參閱 Amazon CloudWatch 定價。

您可以調整每個日誌群組的保留政策。較短的保留會移除舊日誌，並有助於降低儲存成本。若要保留日誌，您可以在移除日誌之前將其封存至 Amazon Simple Storage Service。如需詳細資訊，請參閱《Amazon CloudWatch 使用者指南》中的使用主控台將日誌資料匯出至 Amazon S3。 Amazon CloudWatch

> **Note**
>
> CloudWatch 日誌讀取受 限制AWS。如果您計劃加入許多藝術家，我們建議您聯絡AWS客戶支援，並請求提高 CloudWatch `GetLogEvents` 中的配額。此外，我們建議您在未偵錯時關閉日誌結尾入口網站。

如需詳細資訊，請參閱《Amazon CloudWatch 使用者指南》中的 CloudWatch Logs 配額。 Amazon CloudWatch

CloudWatch 指標

截止日期 Cloud 會將指標傳送至 Amazon CloudWatch。您可以使用 AWS 管理主控台、 AWS CLI或 API 來列出截止日期雲端傳送至 CloudWatch 的指標。根據預設，每個資料點都會涵蓋活動開始時間之後的 1 分鐘。如需有關如何使用 AWS 管理主控台或 檢視可用指標的資訊AWS CLI，請參閱《Amazon CloudWatch 使用者指南》中的檢視可用指標。

客戶受管機群指標

`AWS/DeadlineCloud` 命名空間包含客戶受管機群的下列指標：

指標	Description	單位
`RecommendedFleetSize`	Deadline Cloud 建議您用來處理任務的工作者數量。您可以使用此指標來擴展或收縮機群中的工作者數量。	計數

指標	Description	單位
UnhealthyWorkerCount	指派給處理運作狀態不佳之任務的工作者數目。	計數

您可以使用下列維度來精簡客戶受管機群指標：

維度	Description
FarmId	此維度會篩選您向指定陣列請求的資料。
FleetId	此維度會篩選您向指定工作者機群請求的資料。

授權指標

AWS/DeadlineCloud 命名空間包含下列授權指標：

指標	Description	單位
LicensesInUse	使用中的授權工作階段數目。	計數

您可以使用下列維度來精簡授權指標：

維度	Description
FleetId	使用此維度來篩選指定服務受管機群的資料。對於客戶管理的機群，請改用 LicenseEndpointId 維度。
LicenseEndpointId	使用此維度來篩選指定授權端點的資料。
產品	使用此維度來篩選指定計量產品的資料。

資源限制指標

AWS/DeadlineCloud 命名空間包含下列資源限制指標：

指標	Description	單位
CurrentCount	使用此限制建立模型的資源數量。	計數
MaxCount	此限制建模的資源數量上限。如果您使用 API 將maxCount值設定為 -1，截止日期雲端不會發出MaxCount指標。	計數

您可以使用下列維度來精簡並行限制指標：

維度	Description
FarmId	此維度會篩選您向指定陣列請求的資料。
LimitId	此維度會篩選您請求的資料至指定的限制。

建議的警示

使用 CloudWatch，您可以建立警示來監看指標，並在超過閾值時向您傳送通知或執行另一個動作。如需設定 CloudWatch 警示的詳細資訊，請參閱 Amazon CloudWatch 使用者指南。

我們建議您為下列截止日期雲端指標設定警示：

LicensesInUse

維度：FleetId、 LicenseEndpointId

警示描述：此警示會偵測服務受管機群或授權端點的作用中授權工作階段何時接近您的帳戶配額。如果發生這種情況，您可以提高授權工作階段的帳戶配額。使用 Service Quotas 查看您目前的配額並請求增加。若要進一步了解，請參閱 Service Quotas 使用者指南。

目的：在達到配額限制之前監控用量，以防止授權簽出失敗。

統計資料：最大值

建議的閾值：授權工作階段配額的 90%

閾值對正：將閾值設定為配額的百分比，以便在達到限制之前採取動作。

Period：1 分鐘

警示資料點數目：1

評估期：1

比較運算子： GREATER_THAN_THRESHOLD

其他資源

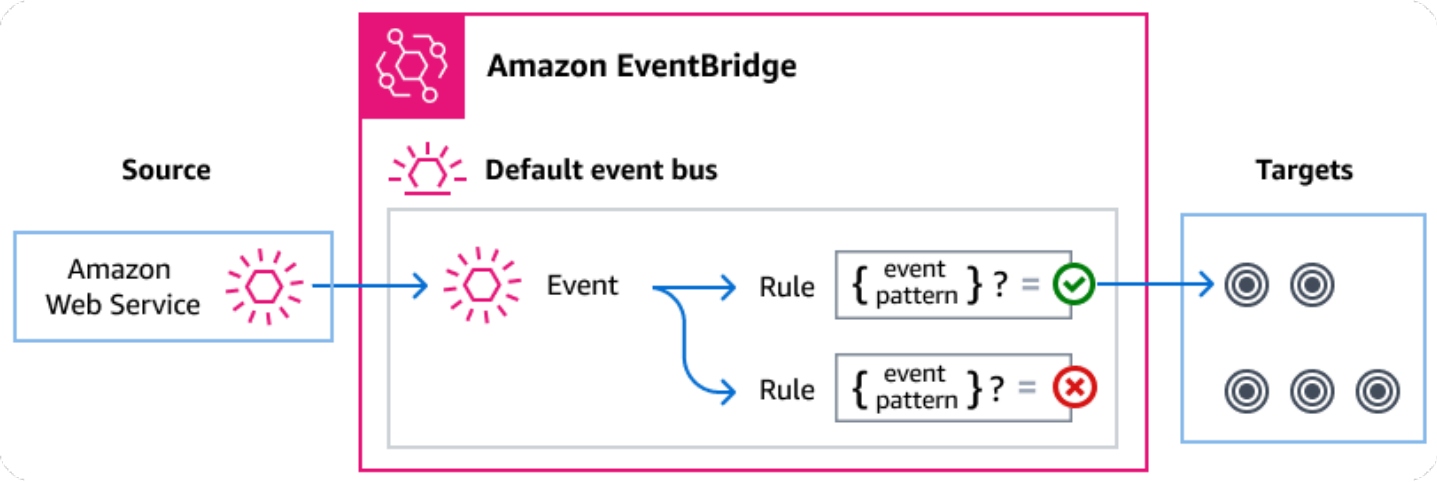
- 《Amazon CloudWatch 使用者指南》https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/
- Service Quotas 使用者指南

使用 管理截止日期雲端事件 Amazon EventBridge

Amazon EventBridge 是一種無伺服器服務，使用事件將應用程式元件連接在一起，讓您更輕鬆地建置可擴展的事件驅動型應用程式。事件驅動架構是一種建置鬆耦合軟體系統的方式，透過發出和回應事件來協作。事件代表資源或環境中的變更。

以下是其運作方式：

如同許多 AWS 服務，Deadline Cloud 會產生事件並將其傳送至 EventBridge 預設事件匯流排。（預設事件匯流排會自動在每個 AWS 帳戶中佈建。）事件匯流排是接收事件，並將事件傳遞至零個或多個目的地或目標的路由器。為事件匯流排指定的規則會在事件到達時評估事件。每項規則都會檢查事件是否與規則的事件模式相符。如果事件不相符，事件匯流排會將事件傳送至指定的目標。



主題

截止日期雲端事件

Deadline Cloud 會自動將下列事件傳送至預設 EventBridge 事件匯流排。符合規則事件模式的事件會盡力交付至指定的目標。事件可能無法按順序交付。

如需詳細資訊，請參閱《 使用者指南》中的EventBridge 事件。 Amazon EventBridge

事件詳細資訊類型	Description
已達到預算閾值	當佇列達到其指派預算的百分比時傳送。
任務生命週期狀態變更	當任務的生命週期狀態變更時傳送。
任務執行狀態變更	當任務中任務的整體狀態變更時傳送。
步驟生命週期狀態變更	當任務中步驟的生命週期狀態變更時傳送。
步驟執行狀態變更	當步驟中任務的整體狀態變更時傳送。
任務執行狀態變更	當任務狀態變更時傳送。

使用 EventBridge 規則交付截止日期雲端事件

若要讓 EventBridge 預設事件匯流排將截止日期雲端事件傳送至目標，您必須建立規則。每個規則都包含一個事件模式，與事件匯流排上收到的每個事件 EventBridge 相符。如果事件資料符合指定的事件模式， 會將該事件 EventBridge 傳遞至規則的目標 (s)。

如需建立事件匯流排規則的完整說明，請參閱EventBridge 《 使用者指南》中的建立對事件做出反應的規則。

建立符合截止日期雲端事件的事件模式

每個事件模式都是 JSON 物件，它包含：

- 識別傳送事件之服務的 source 屬性。對於截止日期雲端事件，來源為 aws.deadline。
- (選擇性)：包含要比對之事件類型陣列的 detail-type 屬性。
- (選擇性)：包含要比對的任何其他事件資料的 detail 屬性。

例如，下列事件模式會比對farmId為截止日期雲端指定的 的所有機群大小建議變更事件：

```
{
  "source": ["aws.deadline"],
  "detail-type": ["Fleet Size Recommendation Change"],
  "detail": {
     "farmId": "farm-12345678900000000000000000000000"
  }
}
```

如需撰寫事件模式的詳細資訊，請參閱EventBridge 《 使用者指南》中的事件模式。

截止日期雲端事件詳細資訊參考

AWS 服務中的所有事件都有一組通用的欄位，其中包含事件的中繼資料，例如事件來源 AWS 的服務、事件產生的時間、事件發生的帳戶和區域，以及其他。如需這些一般欄位的定義，請參閱Amazon EventBridge 《 使用者指南》中的事件結構參考。

此外，每個事件都有一個 detail 欄位，其中包含該特定事件的特定資料。以下參考定義了各種截止日期雲端事件的詳細資訊欄位。

使用 EventBridge 選取和管理截止日期雲端事件時，請謹記下列事項：

- 截止日期雲端中所有事件的 source 欄位設定為 aws.deadline。
- detail-type 欄位指定事件類型。

  例如 Fleet Size Recommendation Change。
- detail 欄位包含該特定事件的特定資料。

如需有關建構事件模式以讓規則符合截止日期雲端事件的資訊，請參閱Amazon EventBridge 《 使用者指南》中的事件模式。

如需事件及其 EventBridge 處理方式的詳細資訊，請參閱Amazon EventBridge 《 使用者指南》中的Amazon EventBridge 事件。

主題

已達到預算閾值事件

您可以使用 Budget Threshold Reached 事件來監控已使用的預算百分比。當使用的百分比超過下列閾值時，截止日期 Cloud 會傳送事件：

- 10、20、30、40、50、60、70、75、80、85、90、95、96、97、98、99、100

截止日期雲端傳送預算閾值達到事件的頻率會隨著預算接近其限制而增加。此頻率可讓您在預算接近其限制時密切監控預算，並採取動作來避免過度支出。您也可以設定自己的預算閾值。當用量超過您的自訂閾值時，截止日期 Cloud 會傳送事件。

如果您變更預算金額，則下次 Deadline Cloud 傳送預算閾值達到事件時，會根據已使用預算的目前百分比。例如，如果您將 50 USD 新增至已達到其限制的 100 USD 預算，下一個達到預算閾值事件會指出預算為 75%。

以下是`Budget Threshold Reached`事件的詳細資訊欄位。

以下包含 `source`和 `detail-type` 欄位，因為它們包含截止日期雲端事件的特定值。如需所有事件中包含的其他中繼資料欄位的定義，請參閱Amazon EventBridge 《 使用者指南》中的事件結構參考。

```
{
    "version": "0",
    "id": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
    "detail-type": "Budget Threshold Reached",
    "source": "aws.deadline",
    "account": "111122223333",
    "time": "2017-12-22T18:43:48Z",
    "region": "aa-example-1",
```

```
    "resources": [],
    "detail": {
        "farmId": "farm-12345678900000000000000000000000",
        "budgetId": "budget-12345678900000000000000000000000",
        "thresholdInPercent": 0
    }
}
```

`detail-type`

識別事件的類型。

對於此事件，此值為 `Budget Threshold Reached`。

`source`

識別產生事件的服務。對於截止日期雲端事件，此值為 `aws.deadline`。

`detail`

包含事件相關資訊的 JSON 物件。產生事件的服務會決定此欄位的內容。

對於此事件，此資料包含：

`farmId`

包含任務的陣列識別符。

`budgetId`

已達到閾值的預算識別符。

`thresholdInPercent`

已使用的預算百分比。

機群大小建議變更事件

當您將機群設定為使用事件型自動擴展時，Deadline Cloud 會傳送事件，供您用來管理機群。每個事件都包含有關機群目前大小和請求大小的資訊。如需使用 EventBridge 事件和 Lambda 函數處理事件的範例，請參閱使用截止日期雲端擴展建議功能自動擴展 Amazon EC2 機群。

當發生下列情況時，會傳送機群大小建議變更事件：

- 當建議的機群大小變更且 oldFleetSize 與 newFleetSize 不同時。
- 當服務偵測到實際機群大小與建議的機群大小不相符時。您可以在 GetFleet 操作回應中，從 workerCount 取得實際機群大小。當作用中的 Amazon EC2 執行個體無法註冊為截止日期雲端工作者時，可能會發生這種情況。

以下是Fleet Size Recommendation Change事件的詳細資訊欄位。

以下包含 source和 detail-type 欄位，因為它們包含截止日期雲端事件的特定值。如需所有事件中包含的其他中繼資料欄位的定義，請參閱Amazon EventBridge 《 使用者指南》中的事件結構參考。

```
{
    "version": "0",
    "id": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
    "detail-type": "Fleet Size Recommendation Change",
    "source": "aws.deadline",
    "account": "111122223333",
    "time": "2017-12-22T18:43:48Z",
    "region": "aa-example-1",
    "resources": [],
    "detail": {
        "farmId": "farm-12345678900000000000000000000000",
        "fleetId": "fleet-12345678900000000000000000000000",
        "oldFleetSize": 1,
        "newFleetSize": 5,
    }
}
```

detail-type

識別事件的類型。

對於此事件，此值為 Fleet Size Recommendation Change。

source

識別產生事件的服務。對於截止日期雲端事件，此值為 aws.deadline。

detail

包含事件相關資訊的 JSON 物件。產生事件的服務會決定此欄位的內容。

對於此事件，此資料包含：

`farmId`

包含任務的陣列識別符。

`fleetId`

需要變更大小的機群識別符。

`oldFleetSize`

機群的目前大小。

`newFleetSize`

建議的機群新大小。

任務生命週期狀態變更事件

當您建立或更新任務時，截止日期雲端會設定生命週期狀態，以顯示最近使用者啟動動作的狀態。

任務生命週期狀態變更事件會針對任何生命週期狀態變更傳送，包括建立任務的時間。

以下是`Job Lifecycle Status Change`事件的詳細資訊欄位。

以下包含 `source`和 `detail-type` 欄位，因為它們包含截止日期雲端事件的特定值。如需所有事件中包含的其他中繼資料欄位的定義，請參閱Amazon EventBridge 《 使用者指南》中的事件結構參考。

```
{
    "version": "0",
    "id": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
    "detail-type": "Job Lifecycle Status Change",
    "source": "aws.deadline",
    "account": "111122223333",
    "time": "2017-12-22T18:43:48Z",
    "region": "aa-example-1",
    "resources": [],
    "detail": {
        "farmId": "farm-12345678900000000000000000000000",
        "queueId": "queue-12345678900000000000000000000000",
        "jobId": "job-12345678900000000000000000000000",
        "previousLifecycleStatus": "UPDATE_IN_PROGRESS",
        "lifecycleStatus": "UPDATE_SUCCEEDED"
    }
```

```
}
```

`detail-type`

識別事件的類型。

對於此事件，此值為 `Job Lifecycle Status Change`。

`source`

識別產生事件的服務。對於截止日期雲端事件，此值為 `aws.deadline`。

`detail`

包含事件相關資訊的 JSON 物件。產生事件的服務會決定此欄位的內容。

對於此事件，此資料包含：

`farmId`

包含任務的陣列識別符。

`queueId`

包含任務之佇列的識別符。

`jobId`

任務的識別符。

`previousLifecycleStatus`

任務要離開的生命週期狀態。當您第一次提交任務時，不會包含此欄位。

`lifecycleStatus`

任務進入的生命週期狀態。

任務執行狀態變更事件

任務由許多任務組成。每個任務都有一個狀態。所有任務的狀態都會合併，以提供任務的整體狀態。如需詳細資訊，請參閱《截止日期雲端使用者指南》中的截止日期雲端中的任務狀態。 AWS

任務執行狀態變更事件會在下列情況下傳送：

- 合併taskRunStatus欄位會變更。
- 除非任務處於就緒狀態，否則任務會重新排入佇列。

下列情況下，不會傳送任務執行狀態變更事件：

- 任務會先建立。若要監控任務建立，請監控任務生命週期狀態變更事件的變更。
- 任務taskRunStatusCounts的欄位會變更，但合併任務執行狀態不會變更。

以下是Job Run Status Change事件的詳細資訊欄位。

以下包含 source和 detail-type 欄位，因為它們包含截止日期雲端事件的特定值。如需所有事件中包含的其他中繼資料欄位的定義，請參閱Amazon EventBridge 《 使用者指南》中的事件結構參考。


```
{
    "version": "0",
    "id": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
    "detail-type": "Job Run Status Change",
    "source": "aws.deadline",
    "account": "111122223333",
    "time": "2017-12-22T18:43:48Z",
    "region": "aa-example-1",
    "resources": [],
    "detail": {
        "farmId": "farm-12345678900000000000000000000000",
        "queueId": "queue-12345678900000000000000000000000",
        "jobId": "job-12345678900000000000000000000000",
        "previousTaskRunStatus": "RUNNING",
        "taskRunStatus": "SUCCEEDED",
        "taskRunStatusCounts": {
            "PENDING": 0,
            "READY": 0,
            "RUNNING": 0,
            "ASSIGNED": 0,
            "STARTING": 0,
            "SCHEDULED": 0,
            "INTERRUPTING": 0,
            "SUSPENDED": 0,
            "CANCELED": 0,
            "FAILED": 0,
            "SUCCEEDED": 20,
            "NOT_COMPATIBLE": 0
```

```
        }
    }
}
```

detail-type

識別事件的類型。

對於此事件，此值為 Job Run Status Change。

source

識別產生事件的服務。對於截止日期雲端事件，此值為 aws.deadline。

detail

包含事件相關資訊的 JSON 物件。產生事件的服務會決定此欄位的內容。

對於此事件，此資料包含：

farmId

包含任務的陣列識別符。

queueId

包含任務之佇列的識別符。

jobId

任務的識別符。

previousTaskRunStatus

任務要離開的任務執行狀態。

taskRunStatus

任務正在進入的任務執行狀態。

taskRunStatusCounts

每個狀態的任務數量。

步驟生命週期狀態變更事件

當您建立或更新事件時，截止日期雲端會設定任務的生命週期狀態，以描述最近使用者啟動動作的狀態。

步驟生命週期狀態變更事件會在下列情況下傳送：

- 步驟更新開始 (UPDATE_IN_PROGRESS)。
- 步驟更新已成功完成 (UPDATE_SUCCEEDED)。
- 步驟更新失敗 (UPDATE_FAILED)。

第一次建立步驟時，不會傳送事件。若要監控步驟建立，請監控任務生命週期狀態變更事件的變更。

以下是Step Lifecycle Status Change事件的詳細資訊欄位。

以下包含 source和 detail-type 欄位，因為它們包含截止日期雲端事件的特定值。如需所有事件中包含的其他中繼資料欄位的定義，請參閱Amazon EventBridge 《 使用者指南》中的事件結構參考。

```
{
    "version": "0",
    "id": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
    "detail-type": "Step Lifecycle Status Change",
    "source": "aws.deadline",
    "account": "111122223333",
    "time": "2017-12-22T18:43:48Z",
    "region": "aa-example-1",
    "resources": [],
    "detail": {
        "farmId": "farm-12345678900000000000000000000000",
        "queueId": "queue-12345678900000000000000000000000",
        "jobId": "job-12345678900000000000000000000000",
        "stepId": "step-12345678900000000000000000000000",
        "previousLifecycleStatus": "UPDATE_IN_PROGRESS",
        "lifecycleStatus": "UPDATE_SUCCEEDED"
    }
}
```

`detail-type`

識別事件的類型。

對於此事件，此值為 Step Lifecycle Status Change。

`source`

識別產生事件的服務。對於截止日期雲端事件，此值為 aws.deadline。

`detail`

包含事件相關資訊的 JSON 物件。產生事件的服務會決定此欄位的內容。

對於此事件，此資料包含：

`farmId`

包含任務的陣列識別符。

`queueId`

包含任務之佇列的識別符。

`jobId`

任務的識別符。

`stepId`

目前任務步驟的識別符。

`previousLifecycleStatus`

步驟要離開的生命週期狀態。

`lifecycleStatus`

步驟進入的生命週期狀態。

步驟執行狀態變更事件

任務中的每個步驟都由許多任務組成。每個任務都有一個狀態。任務狀態會合併，以提供步驟和任務的整體狀態。

步驟執行狀態變更事件會在下列情況下傳送：

- 合併taskRunStatus的變更。
- 步驟會重新排入佇列，除非該步驟處於就緒狀態。

發生下列情況時，不會傳送事件：

- 步驟會先建立。若要監控步驟建立，請監控任務生命週期狀態變更事件的變更。
- 步驟taskRunStatusCounts的變更，但合併的步驟任務執行狀態不會變更。

以下是Step Run Status Change事件的詳細資訊欄位。

以下包含 source和 detail-type 欄位，因為它們包含截止日期雲端事件的特定值。如需所有事件中包含的其他中繼資料欄位的定義，請參閱Amazon EventBridge 《 使用者指南》中的事件結構參考。

```
{
    "version": "0",
    "id": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
    "detail-type": "Step Run Status Change",
    "source": "aws.deadline",
    "account": "111122223333",
    "time": "2017-12-22T18:43:48Z",
    "region": "aa-example-1",
    "resources": [],
    "detail": {
        "farmId": "farm-12345678900000000000000000000000",
        "queueId": "queue-12345678900000000000000000000000",
        "jobId": "job-12345678900000000000000000000000",
        "stepId": "step-12345678900000000000000000000000",
        "previousTaskRunStatus": "RUNNING",
        "taskRunStatus": "SUCCEEDED",
        "taskRunStatusCounts": {
            "PENDING": 0,
            "READY": 0,
            "RUNNING": 0,
            "ASSIGNED": 0,
            "STARTING": 0,
            "SCHEDULED": 0,
            "INTERRUPTING": 0,
            "SUSPENDED": 0,
            "CANCELED": 0,
            "FAILED": 0,
            "SUCCEEDED": 20,
            "NOT_COMPATIBLE": 0
       }
    }
}
```

detail-type

識別事件的類型。

對於此事件，此值為 Step Run Status Change。

`source`

識別產生事件的服務。對於截止日期雲端事件，此值為 `aws.deadline`。

`detail`

包含事件相關資訊的 JSON 物件。產生事件的服務會決定此欄位的內容。

對於此事件，此資料包含：

`farmId`

包含任務的陣列識別符。

`queueId`

包含任務之佇列的識別符。

`jobId`

任務的識別符。

`stepId`

目前任務步驟的識別符。

`previousTaskRunStatus`

步驟要離開的執行狀態。

`taskRunStatus`

步驟進入的執行狀態。

`taskRunStatusCounts`

每個狀態中步驟的任務數量。

任務執行狀態變更事件

`runStatus` 欄位會在任務執行時更新。事件會在下列情況下傳送：

- 任務的執行狀態會變更。
- 除非任務處於 READY 狀態，否則任務會重新排入佇列。

發生下列情況時，不會傳送事件：

- 任務會先建立。若要監控任務建立，請監控任務生命週期狀態變更事件的變更。

以下是Task Run Status Change事件的詳細資訊欄位。

以下包含 source和 detail-type 欄位，因為它們包含截止日期雲端事件的特定值。如需所有事件中包含的其他中繼資料欄位的定義，請參閱Amazon EventBridge 《 使用者指南》中的事件結構參考。

```
{
    "version": "0",
    "id": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
    "detail-type": "Task Run Status Change",
    "source": "aws.aws.deadline",
    "account": "111122223333",
    "time": "2017-12-22T18:43:48Z",
    "region": "aa-example-1",
    "resources": [],
    "detail": {
        "farmId": "farm-12345678900000000000000000000000",
        "queueId": "queue-12345678900000000000000000000000",
        "jobId": "job-12345678900000000000000000000000",
        "stepId": "step-12345678900000000000000000000000",
        "taskId": "task-12345678900000000000000000000000-0",
        "previousRunStatus": "RUNNING",
        "runStatus": "SUCCEEDED"
    }
}
```

`detail-type`

識別事件的類型。

對於此事件，此值為 `Fleet Size Recommendation Change`。

`source`

識別產生事件的服務。對於截止日期雲端事件，此值為 `aws.deadline`。

`detail`

包含事件相關資訊的 JSON 物件。產生事件的服務會決定此欄位的內容。

對於此事件，此資料包含：

`farmId`

包含任務的陣列識別符。

queueId

包含任務之佇列的識別符。

jobId

任務的識別符。

stepId

目前任務步驟的識別符。

taskId

執行中任務的識別符。

previousRunStatus

任務要離開的執行狀態。

runStatus

任務進入的執行狀態。

使用 查詢工作階段統計資料彙總資料 AWS CLI

若要追蹤成本、分析資源用量，或識別哪些使用者耗用最多資源，您可以使用 AWS Command Line Interface (AWS CLI) 查詢 AWS 截止日期雲端 （ 截止日期雲端 ） 陣列的彙總工作階段統計資料。工作階段統計資料 API 提供成本、執行期和用量的資料，您可以依佇列、機群、執行個體類型或使用者等各種維度分組。

查詢工作階段統計資料是非同步程序。首先，您啟動彙總請求，然後使用彙總 ID 擷取結果。

啟動彙總請求

若要啟動彙總請求，請執行 `start-sessions-statistics-aggregation`命令。下列範例會依特定佇列的使用者 ID 將統計資料分組。將`####`文字取代為您的資訊。

```
aws deadline start-sessions-statistics-aggregation \
    --farm-id farm-id \
    --resource-ids '{"queueIds":["queue-id"]}' \
    --start-time 2025-11-24T10:00:00Z \
    --end-time 2025-11-25T18:00:00Z \
    --group-by '["USER_ID"]' \
    --period HOURLY \
    --statistics '["SUM"]' \
    --timezone UTC-08:00 \
    --region region-name
```

您可以依其他維度將統計資料分組，例如 `QUEUE_ID`、`INSTANCE_TYPE`、、 `FLEET_ID` `JOB_ID`或 `LICENSE_PRODUCT`。如需所有可用參數的詳細資訊，請參閱《 AWS CLI 命令參考》中的 start-sessions-statistics-aggregation。

回應包含彙總 ID：

```
{
    "aggregationId": "92b35143f2d04641979bc9b777232f38"
}
```

擷取結果

使用彙總 ID 執行 `get-sessions-statistics-aggregation`命令以擷取結果。將`####`文字取代為您的資訊。

```
aws deadline get-sessions-statistics-aggregation \
    --farm-id farm-id \
    --aggregation-id aggregation-id \
    --region region-name
```

下列範例顯示當您依使用者 ID 將統計資料分組時的回應。userId 欄位包含 UUID，您必須映射至使用者名稱以識別使用者：

```
{
    "statistics": [
        {
            "userId": "f9c1f3f0-1031-70dc-4d25-30d7225b04a0",
            "count": 1,
            "costInUsd": {
                "sum": 0.0
            },
            "runtimeInSeconds": {
                "sum": 53.773
            },
            "aggregationStartTime": "2025-11-24T22:00:00Z",
            "aggregationEndTime": "2025-11-24T23:00:00Z"
        }
    ],
    "status": "COMPLETED"
}
```

若要尋找與 相關聯的使用者名稱userId，請參閱 the section called “使用 userID 擷取使用者中繼資料”。

如需 API 的詳細資訊，請參閱截止日期雲端 API 參考。

主題

- the section called “使用 userID 擷取使用者中繼資料”

在身分存放區中使用 userID 擷取使用者中繼資料和屬性

> **Note**
>
> 此程序也適用於 SearchJobs API 傳回`createdBy`的欄位，該 API 使用相同的使用者 ID 格式。

工作階段統計資料中的 `userId` 欄位包含下列其中一個值：

- AWS Identity and Access Management (IAM) 角色 ARN，例如：`arn:aws:sts::123456789012:assumed-role/Admin/user-Isengard`。
- IAM Identity Center 使用者 ID (UUID)，例如：`f9c1f3f0-1031-70dc-4d25-30d7225b04a0`。

對於 IAM 角色 ARNs，使用者名稱會顯示在 ARN 本身中。對於 IAM Identity Center 使用者 IDs，您可以使用 IAM Identity Center Identity Store API 查詢使用者名稱。

若要識別與 IAM Identity Center 使用者 ID 相關聯的使用者名稱，請使用下列程序。開始之前，請從 IAM Identity Center 設定取得 Identity Store ID。如需詳細資訊，請參閱the section called “尋找您的身分存放區 ID”。

映射使用者 ID

1. 執行下列命令，將 *`IdentityStoreId`* 取代為 Identity Store ID，並將 *`userUUID`* 取代為工作階段統計資料回應`userId`中的 ：

```
aws identitystore describe-user \
    --identity-store-id IdentityStoreId \
    --user-id userUUID
```

2. 檢閱回應，其中包含使用者名稱：

```
{
    "UserName": "jdoe",
    "UserId": "f9c1f3f0-1031-70dc-4d25-30d7225b04a0",
    "Name": {
        "FamilyName": "Doe",
        "GivenName": "Jane"
    },
```

```
    "DisplayName": "Jane Doe",
    "Emails": [{
        "Value": "jdoe@example.com",
        "Type": "work",
        "Primary": true
    }],
    "IdentityStoreId": "d-xxxxxxxxxx"
}
```

尋找您的身分存放區 ID

若要將使用者 IDs映射至使用者名稱，您需要 Identity Store ID。您可以使用 IAM Identity Center 主控台或 尋找 Identity Store ID AWS CLI。

主控台

若要使用主控台尋找您的身分存放區 ID，請使用下列程序。

1. 登入 AWS 管理主控台並開啟 IAM Identity Center 主控台。
2. 在導覽窗格中，選擇設定。
3. 複製 IAM Identity Center Identity Store ID 值。格式是 d-xxxxxxxxxx。

AWS CLI

執行下列命令，將 *region-name* 取代為設定 IAM Identity Center 執行個體的區域：

```
aws sso-admin list-instances --region region-name
```

回應包含 IdentityStoreId：

```
{
    "Instances": [
        {
            "CreatedDate": "2025-11-19T15:45:55.160000-08:00",
            "IdentityStoreId": "d-xxxxxxxxxx",
            "InstanceArn": "arn:aws:sso:::instance/ssoins-xxxxxxxxxxxxxxxx",
            "OwnerAccountId": "123456789012",
            "Status": "ACTIVE"
        }
```

```
    ]
}
```

驗證使用者映射

將使用者 ID 映射至使用者名稱後，您可以在 IAM Identity Center 主控台中驗證使用者 ID 是否符合預期的使用者。若要驗證使用者映射，請使用下列程序。

1. 登入 AWS 管理主控台並開啟 IAM Identity Center 主控台。
2. 在導覽窗格中，選擇使用者 。
3. 從 AWS CLI 回應中選擇使用者名稱。
4. 在一般資訊區段中，確認使用者 ID 符合工作階段統計資料`userId`中的 。

其他資源

- IAM Identity Center 使用者指南
- IAM Identity Center Identity Store API 參考

中的安全性 Deadline Cloud

的雲端安全性 AWS 是最高優先順序。身為 AWS 客戶，您可以受益於資料中心和網路架構，這些架構專為滿足最安全敏感組織的需求而建置。

安全性是 AWS 與您之間的共同責任。共同責任模式將其描述為雲端的安全性，和雲端中的安全性：

- 雲端的安全性 – AWS 負責保護在 AWS 服務 中執行的基礎設施 AWS 雲端。 AWS 也為您提供可安全使用的服務。在AWS 合規計畫中，第三方稽核人員會定期測試和驗證我們安全的有效性。若要了解適用的合規計劃 AWS Deadline Cloud，請參閱AWS 服務 合規計劃範圍。
- 雲端的安全性 – 您的責任取決於您使用 AWS 服務 的 。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。

本文件可協助您了解如何在使用 時套用共同責任模型 Deadline Cloud。下列主題說明如何設定 Deadline Cloud 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 Deadline Cloud 資源。

主題

- 中的資料保護 Deadline Cloud
- 截止日期雲端中的身分和存取管理
- 的合規驗證 Deadline Cloud
- 中的彈性 Deadline Cloud
- 截止日期雲端中的基礎設施安全性
- 截止日期雲端中的組態和漏洞分析
- 預防跨服務混淆代理人
- AWS Deadline Cloud 使用界面端點存取 (AWS PrivateLink)
- 受限的網路環境
- 截止日期雲端的安全最佳實務

中的資料保護 Deadline Cloud

AWS 共同責任模型適用於 中的資料保護 AWS Deadline Cloud。如此模型所述， AWS 負責保護執行所有 的全域基礎設施 AWS 雲端。您負責維護在此基礎設施上託管內容的控制權。您也同時負責所使

用 AWS 服務 的安全組態和管理任務。如需資料隱私權的詳細資訊，請參閱資料隱私權常見問答集。如需歐洲資料保護的相關資訊，請參閱「一般資料保護法規 (GDPR) 中心」。

基於資料保護目的，我們建議您保護 AWS 帳戶 登入資料，並使用 AWS IAM Identity Center 或 AWS Identity and Access Management (IAM) 設定個別使用者。如此一來，每個使用者都只會獲得授與完成其任務所必須的許可。我們也建議您採用下列方式保護資料：

- 每個帳戶均要使用多重要素驗證 (MFA)。
- 使用 SSL/TLS 與 AWS 資源通訊。我們需要 TLS 1.2 並建議使用 TLS 1.3。
- 使用 設定 API 和使用者活動記錄 AWS CloudTrail。如需有關使用 CloudTrail 追蹤擷取 AWS 活動的資訊，請參閱AWS CloudTrail 《 使用者指南》中的使用 CloudTrail 追蹤。
- 使用 AWS 加密解決方案，以及其中的所有預設安全控制 AWS 服務。
- 使用進階的受管安全服務 (例如 Amazon Macie)，協助探索和保護儲存在 Amazon S3 的敏感資料。
- 如果您在 AWS 透過命令列界面或 API 存取 時需要 FIPS 140-3 驗證的密碼編譯模組，請使用 FIPS 端點。如需有關 FIPS 和 FIPS 端點的更多相關資訊，請參閱聯邦資訊處理標準 (FIPS) 140-3。

我們強烈建議您絕對不要將客戶的電子郵件地址等機密或敏感資訊，放在標籤或自由格式的文字欄位中，例如名稱欄位。這包括當您使用 Deadline Cloud 或使用主控台、API AWS CLI或其他 AWS 服務 AWS SDKs 時。您在標籤或自由格式文字欄位中輸入的任何資料都可能用於計費或診斷日誌。如果您提供外部伺服器的 URL，我們強烈建議請勿在驗證您對該伺服器請求的 URL 中包含憑證資訊。

輸入 Deadline Cloud 任務範本中名稱欄位的資料也可能包含在帳單或診斷日誌中，且不應包含機密或敏感資訊。

主題

靜態加密

AWS Deadline Cloud 使用存放在 AWS Key Management Service (AWS KMS) 中的加密金鑰，透過靜態加密來保護敏感資料。所有 都可以使用靜態加密 AWS 區域 Deadline Cloud 。

加密資料表示沒有有效金鑰的使用者或應用程式無法讀取儲存在磁碟上的敏感資料。只有具有有效受管金鑰的一方才能解密資料。

Deadline Cloud 當服務受管機群工作者執行個體終止時， 會刪除 Amazon Elastic Block Store 磁碟區。

如需 Deadline Cloud 如何使用 AWS KMS 加密靜態資料的資訊，請參閱 金鑰管理。

傳輸中加密

對於傳輸中的資料， AWS Deadline Cloud 會使用 Transport Layer Security (TLS) 1.2 或 1.3 來加密服務與工作者之間傳送的資料。我們需要 TLS 1.2 並建議使用 TLS 1.3。此外，如果您使用虛擬私有雲端 (VPC)，您可以使用 在 VPC 與 之間 AWS PrivateLink 建立私有連線 Deadline Cloud。

金鑰管理

建立新的陣列時，您可以選擇下列其中一個金鑰來加密您的陣列資料：

- AWS 擁有的 KMS 金鑰 – 如果您在建立陣列時未指定金鑰，則預設加密類型。KMS 金鑰由 擁有 AWS Deadline Cloud。您無法檢視、管理或使用 AWS 擁有的金鑰。不過，您不需要採取任何動作來保護加密資料的金鑰。如需詳細資訊，請參閱 AWS Key Management Service 開發人員指南中的 AWS 擁有的金鑰。
- 客戶受管 KMS 金鑰 – 您在建立陣列時指定客戶受管金鑰。陣列中的所有內容都會使用 KMS 金鑰加密。金鑰會存放在您的帳戶中，並由您建立、擁有和管理，並 AWS KMS 收取費用。您可以完全控制 KMS 金鑰。您可以執行下列任務：
  - 建立和維護關鍵政策
  - 建立和維護 IAM 政策和授予操作
  - 啟用和停用金鑰政策
  - 新增 標籤
  - 建立金鑰別名

  您無法手動輪換與 Deadline Cloud 陣列搭配使用的客戶擁有金鑰。支援自動輪換金鑰。

  如需詳細資訊，請參閱《 AWS Key Management Service 開發人員指南》中的客戶擁有的金鑰。

  若要建立客戶受管金鑰，請遵循《 AWS Key Management Service 開發人員指南》中的建立對稱客戶受管金鑰的步驟。

Deadline Cloud 如何使用 AWS KMS 授予

Deadline Cloud 需要授予才能使用您的客戶受管金鑰。當您建立使用客戶受管金鑰加密的陣列時， 會透過傳送CreateGrant請求至 AWS KMS 來代表您 Deadline Cloud 建立授予，以存取您指定的 KMS 金鑰。

Deadline Cloud 使用多個授予。每個授權都會由 Deadline Cloud 需要加密或解密資料的 不同部分使用。 Deadline Cloud 也會使用授權來允許存取 AWS 其他用來代表您存放資料的 服務，例如 Amazon Simple Storage Service、Amazon Elastic Block Store 或 OpenSearch。

准許 Deadline Cloud 管理服務受管機群中的機器， Deadline Cloud 包括 中的帳號和角色，GranteePrincipal而不是服務主體。雖然不是典型的，但這對於使用為陣列指定的客戶受管 KMS 金鑰來加密服務受管機群中工作者的 Amazon EBS 磁碟區是必要的。

客戶受管金鑰政策

金鑰政策會控制客戶受管金鑰的存取權限。每個金鑰都必須只有一個金鑰政策，其中包含可決定誰可以使用金鑰及其使用方式的陳述式。當您建立客戶受管金鑰時，您可以指定金鑰政策。如需詳細資訊，請參閱《AWS Key Management Service 開發人員指南》中的管理客戶受管金鑰的存取。

CreateFarm 的最低 IAM 政策

若要使用客戶受管金鑰來使用主控台或 CreateFarm API 操作建立陣列，必須允許下列 AWS KMS API 操作：

- kms:CreateGrant：新增客戶受管金鑰的授權。授予主控台對指定 AWS KMS 金鑰的存取權。如需詳細資訊，請參閱《 AWS Key Management Service 開發人員指南》中的使用授權。
- kms:Decrypt – 允許 Deadline Cloud 解密陣列中的資料。
- kms:DescribeKey – 提供客戶受管金鑰詳細資訊， Deadline Cloud 以允許 驗證金鑰。
- kms:GenerateDataKey – 允許 Deadline Cloud 使用唯一的資料金鑰加密資料。

下列政策陳述式會授予 CreateFarm操作的必要許可。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
```

```
        {
            "Sid": "DeadlineCreateGrants",
            "Effect": "Allow",
            "Action": [
                "kms:Decrypt",
                "kms:GenerateDataKey",
                "kms:CreateGrant",
                "kms:DescribeKey"
            ],
            "Resource": "arn:aws:kms:us-
west-2:111122223333:key/1234567890abcdef0",
            "Condition": {
                "StringEquals": {
                    "kms:ViaService": "deadline.us-west-2.amazonaws.com"
                }
            }
        }
    ]
}
```

唯讀操作的最低 IAM 政策

若要將客戶受管金鑰用於唯讀 Deadline Cloud 操作，例如取得有關陣列、佇列和機群的資訊。必須允許下列 AWS KMS API 操作：

- kms:Decrypt – 允許 Deadline Cloud 解密陣列中的資料。
- kms:DescribeKey – 提供客戶受管金鑰詳細資訊， Deadline Cloud 以允許 驗證金鑰。

下列政策陳述式會授予唯讀操作的必要許可。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "DeadlineReadOnly",
            "Effect": "Allow",
            "Action": [
                "kms:Decrypt",
```

```
                "kms:DescribeKey"
            ],
            "Resource": "arn:aws:kms:us-
west-2:111122223333:key/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
            "Condition": {
                "StringEquals": {
                    "kms:ViaService": "deadline.us-west-2.amazonaws.com"
                }
            }
        }
    ]
}
```

讀取寫入操作的最低 IAM 政策

若要將客戶受管金鑰用於讀寫 Deadline Cloud 操作，例如建立和更新陣列、佇列和機群。必須允許下列 AWS KMS API 操作：

- kms:Decrypt – 允許 Deadline Cloud 解密陣列中的資料。
- kms:DescribeKey – 提供客戶受管金鑰詳細資訊， Deadline Cloud 以允許 驗證金鑰。
- kms:GenerateDataKey – 允許 Deadline Cloud 使用唯一的資料金鑰加密資料。

下列政策陳述式會授予 CreateFarm操作的必要許可。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "DeadlineReadWrite",
            "Effect": "Allow",
            "Action": [
                "kms:Decrypt",
                "kms:DescribeKey",
                "kms:GenerateDataKey"
            ],
            "Resource": "arn:aws:kms:us-
west-2:111122223333:key/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
            "Condition": {
```

```
                "StringEquals": {
                    "kms:ViaService": "deadline.us-west-2.amazonaws.com"
                }
            }
        }
    ]
}
```

監控加密金鑰

當您搭配 Deadline Cloud 陣列使用 AWS KMS 客戶受管金鑰時，您可以使用 AWS CloudTrail或 Amazon CloudWatch Logs 來追蹤 Deadline Cloud 傳送至 的請求 AWS KMS。

授予的 CloudTrail 事件

建立授予時，通常會在呼叫 `CreateFarm`、 `CreateMonitor`或 `CreateFleet`操作時發生下列 CloudTrail 事件範例。

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "AROAIGDTESTANDEXAMPLE:SampleUser01",
        "arn": "arn:aws::sts::111122223333:assumed-role/Admin/SampleUser01",
        "accountId": "111122223333",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE3",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "AROAIGDTESTANDEXAMPLE",
                "arn": "arn:aws::iam::111122223333:role/Admin",
                "accountId": "111122223333",
                "userName": "Admin"
            },
            "webIdFederationData": {},
            "attributes": {
                "creationDate": "2024-04-23T02:05:26Z",
                "mfaAuthenticated": "false"
            }
        },
        "invokedBy": "deadline.amazonaws.com"
    },
```

```
    "eventTime": "2024-04-23T02:05:35Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "CreateGrant",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "deadline.amazonaws.com",
    "userAgent": "deadline.amazonaws.com",
    "requestParameters": {
        "operations": [
            "CreateGrant",
            "Decrypt",
            "DescribeKey",
            "Encrypt",
            "GenerateDataKey"
        ],
        "constraints": {
            "encryptionContextSubset": {
                "aws:deadline:farmId": "farm-abcdef1234567890098765432 1fedcba",
                "aws:deadline:accountId": "111122223333"
            }
        },
        "granteePrincipal": "deadline.amazonaws.com",
        "keyId": "arn:aws::kms:us-west-2:111122223333:key/a1b2c3d4-5678-90ab-cdef-
EXAMPLE11111",
        "retiringPrincipal": "deadline.amazonaws.com"
    },
    "responseElements": {
        "grantId": "6bbe819394822a400fe5e3a75d0e9ef16c1733143fff0c1fc00dc7ac282a18a0",
        "keyId": "arn:aws::kms:us-west-2:111122223333: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-2:111122223333:key/a1b2c3d4-5678-90ab-cdef-
EXAMPLE44444"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333",
```

```
    "eventCategory": "Management"
}
```

用於解密的 CloudTrail 事件

使用客戶受管 KMS 金鑰解密值時，會發生下列 CloudTrail 事件範例。

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "AROAIGDTESTANDEXAMPLE:SampleUser01",
        "arn": "arn:aws::sts::111122223333:assumed-role/SampleRole/SampleUser01",
        "accountId": "111122223333",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "AROAIGDTESTANDEXAMPLE",
                "arn": "arn:aws::iam::111122223333:role/SampleRole",
                "accountId": "111122223333",
                "userName": "SampleRole"
            },
            "webIdFederationData": {},
            "attributes": {
                "creationDate": "2024-04-23T18:46:51Z",
                "mfaAuthenticated": "false"
            }
        },
        "invokedBy": "deadline.amazonaws.com"
    },
    "eventTime": "2024-04-23T18:51:44Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "Decrypt",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "deadline.amazonaws.com",
    "userAgent": "deadline.amazonaws.com",
    "requestParameters": {
        "encryptionContext": {
            "aws:deadline:farmId": "farm-abcdef12345678900987654321fedcba",
            "aws:deadline:accountId": "111122223333",
            "aws-crypto-public-key": "AotL+SAMPLEVALUEiOMEXAMPLEaaqNOTREALaGTESTONLY
+p/5H+EuKd4Q=="
        },
```

```
        "encryptionAlgorithm": "SYMMETRIC_DEFAULT",
        "keyId": "arn:aws::kms:us-west-2:111122223333:key/a1b2c3d4-5678-90ab-cdef-
EXAMPLE11111"
    },
    "responseElements": null,
    "requestID": "aaaaaaaa-bbbb-cccc-dddd-eeeeeeffffff",
    "eventID": "ffffffff-eeee-dddd-cccc-bbbbbbaaaaaa",
    "readOnly": true,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws::kms:us-west-2:111122223333:key/a1b2c3d4-5678-90ab-cdef-
EXAMPLE11111"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333",
    "eventCategory": "Management"
}
```

用於加密的 CloudTrail 事件

使用客戶受管 KMS 金鑰加密值時，會發生下列 CloudTrail 事件範例。

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "AROAIGDTESTANDEXAMPLE:SampleUser01",
        "arn": "arn:aws::sts::111122223333:assumed-role/SampleRole/SampleUser01",
        "accountId": "111122223333",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "AROAIGDTESTANDEXAMPLE",
                "arn": "arn:aws::iam::111122223333:role/SampleRole",
                "accountId": "111122223333",
                "userName": "SampleRole"
            },
            "webIdFederationData": {},
            "attributes": {
```

```
                "creationDate": "2024-04-23T18:46:51Z",
                "mfaAuthenticated": "false"
            }
        },
        "invokedBy": "deadline.amazonaws.com"
    },
    "eventTime": "2024-04-23T18:52:40Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "GenerateDataKey",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "deadline.amazonaws.com",
    "userAgent": "deadline.amazonaws.com",
    "requestParameters": {
        "numberOfBytes": 32,
        "encryptionContext": {
            "aws:deadline:farmId": "farm-abcdef1234567890098765432 1fedcba",
            "aws:deadline:accountId": "111122223333",
            "aws-crypto-public-key": "AotL+SAMPLEVALUEiOMEXAMPLEaaqNOTREALaGTESTONLY
+p/5H+EuKd4Q=="
        },
        "keyId": "arn:aws::kms:us-
west-2:111122223333:key/abcdef12-3456-7890-0987-654321fedcba"
    },
    "responseElements": null,
    "requestID": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
    "eventID": "a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",
    "readOnly": true,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws::kms:us-west-2:111122223333:key/a1b2c3d4-5678-90ab-cdef-
EXAMPLE33333"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333",
    "eventCategory": "Management"
}
```

刪除客戶受管 KMS 金鑰

在 AWS Key Management Service (AWS KMS) 中刪除客戶受管 KMS 金鑰具有破壞性和潛在危險性。它會不可逆地刪除金鑰材料和與金鑰相關聯的所有中繼資料。刪除客戶受管 KMS 金鑰後，您無法再解密該金鑰加密的資料。刪除金鑰表示資料無法復原。

這就是為什麼在刪除 KMS 金鑰之前， AWS KMS 讓客戶有長達 30 天的等待期。預設等待期間為 30 天。

關於等待期

由於刪除客戶受管 KMS 金鑰具有破壞性和潛在危險性，因此我們要求您設定 7–30 天的等待期間。預設等待期間為 30 天。

不過，實際等待期間可能比您排定的期間長最多 24 小時。若要取得要刪除金鑰的實際日期和時間，請使用 DescribeKey操作。您也可以在 AWS KMS 主控台的金鑰詳細資訊頁面的一般組態區段中，查看金鑰的排程刪除日期。請注意時區。

在等待期間，客戶受管金鑰的狀態和金鑰狀態為待刪除。

- 待刪除的客戶受管 KMS 金鑰無法用於任何密碼編譯操作。
- AWS KMS 不會輪換待刪除之客戶受管 KMS 金鑰的備份金鑰。

如需刪除客戶受管 KMS 金鑰的詳細資訊，請參閱《 AWS Key Management Service 開發人員指南》中的刪除客戶主金鑰。

網際網路流量隱私權

AWS Deadline Cloud 支援 Amazon Virtual Private Cloud (Amazon VPC) 保護連線。Amazon VPC 提供您可以用來提高和監控虛擬私有雲端 (VPC) 安全性的功能。

您可以使用在 VPC 內執行的 Amazon Elastic Compute Cloud (Amazon EC2) 執行個體來設定客戶受管機群 (CMF)。透過部署要使用的 Amazon VPC 端點 AWS PrivateLink，CMF 中的工作者與 Deadline Cloud 端點之間的流量會保留在您的 VPC 內。此外，您可以設定 VPC 以限制執行個體的網際網路存取。

在服務受管機群中，工作者無法從網際網路連線，但他們確實可以存取網際網路並透過網際網路連線至 Deadline Cloud 服務。每個服務受管機群都會在自己的隔離網路中執行，而工作者執行個體仍專屬於個別客戶。

選擇退出

AWS Deadline Cloud 會收集特定操作資訊，以協助我們開發和改善 Deadline Cloud。收集的資料包含 AWS 您的帳戶 ID 和使用者 ID 等物件，因此如果您有 的問題，我們可以正確識別您的身分 Deadline Cloud。我們也會收集 Deadline Cloud 特定資訊，例如資源 IDs（適用時為 FarmID 或 QueueID)、產品名稱（例如 JobAttachments、WorkerAgent 等） 和產品版本。

您可以選擇使用應用程式組態選擇退出此資料收集。與用戶端工作站和機群工作者 Deadline Cloud互動的每個電腦都需要分別選擇退出。

Deadline Cloud Monitor - 桌面

Deadline Cloud monitor - 桌面會收集操作資訊，例如當當機和開啟應用程式時，以協助我們了解應用程式何時發生問題。若要選擇退出收集此操作資訊，請前往設定頁面並清除開啟資料收集，以測量截止日期雲端監視器的效能。

在您選擇退出後，桌面監視器不會再傳送操作資料。任何先前收集的資料都會保留，但仍可用於改善服務。如需更多資訊，請參閱 資料隱私權常見問答集。

AWS Deadline Cloud CLI 和工具

AWS Deadline Cloud CLI、提交者和工作者代理程式都會收集操作資訊，例如何時發生當機，以及何時提交任務，以協助我們了解您何時遇到這些應用程式的問題。若要選擇退出收集此操作資訊，請使用下列任一方法：

- 在終端機中，輸入 **`deadline config set telemetry.opt_out true`**。

  以目前使用者身分執行時，這會選擇退出 CLI、提交者和工作者代理程式。
- 安裝 Deadline Cloud 工作者代理程式時，請新增**`--telemetry-opt-out`**命令列引數。例如 **`./install.sh --farm-id $FARM_ID --fleet-id $FLEET_ID --telemetry-opt-out`**。
- 在執行工作者代理程式、CLI 或提交者之前，請設定環境變數：**`DEADLINE_CLOUD_TELEMETRY_OPT_OUT=true`**

在您選擇退出後， Deadline Cloud 工具不會再傳送操作資料。任何先前收集的資料都會保留，但仍可用於改善服務。如需更多資訊，請參閱 資料隱私權常見問答集。

截止日期雲端中的身分和存取管理

AWS Identity and Access Management (IAM) 是 AWS 服務 ，可協助管理員安全地控制對 AWS 資源的存取。IAM 管理員可控制誰可以進行身分驗證 （ 登入 ） 和授權 （ 具有許可 ），以使用截止日期雲端資源。IAM 是您可以免費使用 AWS 服務 的 。

主題

目標對象

使用方式 AWS Identity and Access Management (IAM) 會根據您的角色而有所不同：

- 服務使用者 — 若無法存取某些功能，請向管理員申請所需許可 (請參閱 對 AWS 截止日期雲端身分和存取進行故障診斷)
- 服務管理員 — 負責設定使用者存取權並提交相關許可請求 (請參閱 Deadline Cloud 如何與 IAM 搭配使用)
- IAM 管理員 — 撰寫政策以管理存取控制 (請參閱 截止日期雲端的身分型政策範例)

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、IAM 使用者或擔任 IAM 角色身分進行身分驗證。

您可以使用身分來源的登入資料，例如 AWS IAM Identity Center (IAM Identity Center)、單一登入身分驗證或 Google/Facebook 登入資料，以聯合身分的形式登入。如需有關登入的詳細資訊，請參閱《AWS 登入 使用者指南》中的如何登入您的 AWS 帳戶。

對於程式設計存取， AWS 提供 SDK 和 CLI 以密碼編譯方式簽署請求。如需詳細資訊，請參閱《IAM 使用者指南》中的 API 請求的AWS 第 4 版簽署程序。

AWS 帳戶 根使用者

當您建立 時 AWS 帳戶，您會從一個名為 AWS 帳戶 theroot 使用者的登入身分開始，該身分具有對所有 AWS 服務 和 資源的完整存取權。強烈建議不要使用根使用者來執行日常任務。有關需要根使用者憑證的任務，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使用聯合身分提供者，以 AWS 服務 使用臨時憑證存取 。

聯合身分是您企業目錄、Web 身分提供者的使用者，或使用來自身分來源的 AWS 服務 憑證存取 Directory Service 。聯合身分會擔任角色，而該角色會提供臨時憑證。

若需集中化管理存取權限，建議使用 AWS IAM Identity Center。如需詳細資訊，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https://docs.aws.amazon.com/IAM/latest/UserGuide/id_users.html是一種身分具備單人或應用程式的特定許可權。建議以臨時憑證取代具備長期憑證的 IAM 使用者。如需詳細資訊，請參閱《IAM 使用者指南》中的要求人類使用者使用聯合身分提供者來 AWS 使用臨時憑證存取 。

IAM 群組會指定 IAM 使用者集合，使管理大量使用者的許可權更加輕鬆。如需詳細資訊，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https://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles.html的身分具有特定許可權，其可以提供臨時憑證。您可以透過從使用者切換到 IAM 角色 （主控台） 或呼叫 AWS CLI 或 AWS API 操作來擔任角色。如需詳細資訊，請參閱《IAM 使用者指南》中的擔任角色的方法。

IAM 角色適用於聯合身分使用者存取、臨時 IAM 使用者許可、跨帳戶存取權與跨服務存取，以及在 Amazon EC2 執行的應用程式。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。

使用政策管理存取權

您可以透過建立政策並將其連接到身分或資源 AWS 來控制 AWS 中的存取。政策定義與身分或資源相關聯的許可。當委託人提出請求時 AWS ， 會評估這些政策。大多數政策會以 JSON 文件 AWS 的形式存放在 中。如需進一步了解 JSON 政策文件，請參閱《IAM 使用者指南》中的 JSON 政策概觀。

管理員會使用政策，透過定義哪些主體可在哪些條件下對哪些資源執行動作，以指定可存取的範圍。

預設情況下，使用者和角色沒有許可。IAM 管理員會建立 IAM 政策並將其新增至角色，供使用者後續擔任。IAM 政策定義動作的許可，無論採用何種方式執行。

身分型政策

身分型政策是附加至身分 (使用者、使用者群組或角色) 的 JSON 許可政策文件。這類政策控制身分可對哪些資源執行哪些動作，以及適用的條件。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。

身分型政策可分為內嵌政策 (直接內嵌於單一身分) 與受管政策 (可附加至多個身分的獨立政策)。如需了解如何在受管政策及內嵌政策之間做選擇，請參閱《IAM 使用者指南》中的在受管政策與內嵌政策之間選擇。

資源型政策

資源型政策是附加到資源的 JSON 政策文件。範例包括 IAM 角色信任政策與 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。您必須在資源型政策中指定主體。

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。

其他政策類型

AWS 支援其他政策類型，可設定更多常見政策類型授予的最大許可：

- 許可界限 — 設定身分型政策可授與 IAM 實體的最大許可。如需詳細資訊，請參閱《 IAM 使用者指南》中的 IAM 實體許可界限。
- 服務控制政策 (SCP) — 為 AWS Organizations中的組織或組織單位指定最大許可。如需詳細資訊，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) — 設定您帳戶中資源可用許可的上限。如需詳細資訊，請參閱《AWS Organizations 使用者指南》中的資源控制政策 (RCP)。
- 工作階段政策 — 在以程式設計方式為角色或聯合身分使用者建立臨時工作階段時，以參數形式傳遞的進階政策。如需詳細資訊，請參《IAM 使用者指南》中的工作階段政策。

多種政策類型

當多種類型的政策適用於請求時，產生的許可會更複雜而無法理解。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。

Deadline Cloud 如何與 IAM 搭配使用

在您使用 IAM 管理對截止日期雲端的存取之前，請先了解哪些 IAM 功能可與截止日期雲端搭配使用。

可與截止日期雲端搭配使用的 IAM AWS 功能

IAM 功能	截止日期雲端支援
身分型政策	是
資源型政策	否
政策動作	是
政策資源	是
政策條件索引鍵 (服務特定)	是
ACL	否
ABAC (政策中的標籤)	是
臨時憑證	是
轉送存取工作階段 (FAS)	是
服務角色	是
服務連結角色	否

若要全面了解 Deadline Cloud 和其他 如何與大多數 IAM 功能 AWS 服務 搭配使用，請參閱《IAM 使用者指南》中的與 AWS IAM 搭配使用的 服務。

截止日期雲端的身分型政策

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。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。

截止日期雲端的身分型政策範例

若要檢視截止日期雲端身分型政策的範例，請參閱 截止日期雲端的身分型政策範例。

截止日期雲端中的資源型政策

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。委託人可以包含帳戶、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，您可以在其他帳戶內指定所有帳戶或 IAM 實體作為資源型政策的主體。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。

截止日期雲端的政策動作

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，哪個主體在什麼條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。政策會使用動作來授予執行相關聯動作的許可。

若要查看截止日期雲端動作的清單，請參閱服務授權參考中的AWS 截止日期雲端定義的動作。

Deadline Cloud 中的政策動作在動作之前使用下列字首：

```
deadline
```

若要在單一陳述式中指定多個動作，請用逗號分隔。

```
"Action": [
```

```
      "deadline:action1",
      "deadline:action2"
         ]
```

若要檢視截止日期雲端身分型政策的範例，請參閱 截止日期雲端的身分型政策範例。

截止日期雲端的政策資源

支援政策資源：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。

Resource JSON 政策元素可指定要套用動作的物件。最佳實務是使用其 Amazon Resource Name (ARN) 來指定資源。若動作不支援資源層級許可，使用萬用字元 (*) 表示該陳述式適用於所有資源。

```
"Resource": "*"
```

若要查看截止日期雲端資源類型及其 ARNs的清單，請參閱服務授權參考中的AWS 截止日期雲端定義的資源。若要了解您可以使用哪些動作指定每個資源的 ARN，請參閱AWS 截止日期雲端定義的動作。

若要檢視截止日期雲端身分型政策的範例，請參閱 截止日期雲端的身分型政策範例。

截止日期雲端的政策條件索引鍵

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 元素會根據定義的條件，指定陳述式的執行時機。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。

若要查看截止日期雲端條件索引鍵的清單，請參閱《服務授權參考》中的AWS 截止日期雲端的條件索引鍵。若要了解您可以使用條件金鑰的動作和資源，請參閱AWS 截止日期雲端定義的動作。

若要檢視截止日期雲端身分型政策的範例，請參閱 截止日期雲端的身分型政策範例。

截止日期雲端中的 ACLs

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。

ABAC 與截止日期雲端

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，依據稱為標籤的屬性來定義許可。您可以將標籤連接至 IAM 實體 AWS 和資源，然後設計 ABAC 政策，以便在委託人的標籤符合資源上的標籤時允許操作。

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)。

將臨時登入資料與截止日期雲端搭配使用

支援臨時憑證：是

臨時登入資料提供 AWS 資源的短期存取權，當您使用聯合或切換角色時，會自動建立。 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 使用者指南》中的 IAM 中的臨時安全憑證與可與 IAM 搭配運作的AWS 服務。

轉送截止日期雲端的存取工作階段

支援轉寄存取工作階段 (FAS)：是

轉送存取工作階段 (FAS) 使用呼叫 的委託人許可 AWS 服務，結合 AWS 服務 請求向下游服務提出請求。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。

截止日期雲端的服務角色

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給 AWS 服務。

> **Warning**
>
> 變更服務角色的許可可能會中斷截止日期雲端功能。只有在截止日期雲端提供指引時，才能編輯服務角色。

截止日期雲端的服務連結角色

支援服務連結角色：否

服務連結角色是連結至 的一種服務角色 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。

如需建立或管理服務連結角色的詳細資訊，請參閱可搭配 IAM 運作的AWS 服務。在資料表中尋找服務，其中包含服務連結角色欄中的 Yes。選擇是連結，以檢視該服務的服務連結角色文件。

截止日期雲端的身分型政策範例

根據預設，使用者和角色沒有建立或修改截止日期雲端資源的許可。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。

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 Deadline Cloud 定義的動作和資源類型的詳細資訊，包括每種資源類型的 ARNs 格式，請參閱服務授權參考中的 AWS Deadline Cloud 的動作、資源和條件索引鍵。

主題

- 政策最佳實務
- 使用截止日期雲端主控台
- 存取主控台的政策
- 將任務提交至佇列的政策
- 允許建立授權端點的政策
- 允許監控特定陣列佇列的政策

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、存取或刪除截止日期雲端資源。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，請使用將許可授予許多常見使用案例的 AWS 受管政策。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 例如 使用服務動作 AWS 服務，您也可以使用條件來授予其存取權 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 以提高安全性。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》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。

使用截止日期雲端主控台

若要存取 AWS 截止日期雲端主控台，您必須擁有一組最低許可。這些許可必須允許您列出和檢視 中有關截止日期雲端資源的詳細資訊 AWS 帳戶。如果您建立比最基本必要許可更嚴格的身分型政策，則對於具有該政策的實體 (使用者或角色) 而言，主控台就無法如預期運作。

對於僅呼叫 AWS CLI 或 AWS API 的使用者，您不需要允許最低主控台許可。反之，只需允許存取符合他們嘗試執行之 API 操作的動作就可以了。

為了確保使用者和角色仍然可以使用截止日期雲端主控台，也請將截止日期雲端*ConsoleAccess*或*ReadOnly* AWS 受管政策連接到實體。如需詳細資訊，請參閱《IAM 使用者指南》中的新增許可到使用者。

存取主控台的政策

若要授予對 Deadline Cloud 主控台中所有功能的存取權，請將此身分政策連接至您想要擁有完整存取權的使用者或角色。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [{
        "Sid": "EC2InstanceTypeSelection",
        "Effect": "Allow",
        "Action": [
            "ec2:DescribeInstanceTypeOfferings",
            "ec2:DescribeInstanceTypes",
            "ec2:GetInstanceTypesFromInstanceRequirements",
            "pricing:GetProducts"
        ],
        "Resource": ["*"]
    },
    {
        "Sid": "VPCResourceSelection",
        "Effect": "Allow",
        "Action": [
            "ec2:DescribeVpcs",
            "ec2:DescribeSubnets",
            "ec2:DescribeSecurityGroups"
        ],
        "Resource": ["*"]
    },
    {
        "Sid": "ViewVpcLatticeResources",
        "Effect": "Allow",
        "Action": [
            "vpc-lattice:ListResourceConfigurations",
            "vpc-lattice:GetResourceConfiguration",
            "vpc-lattice:GetResourceGateway"
        ],
        "Resource": ["*"]
    },
    {
        "Sid": "ManageVpcEndpointsViaDeadline",
        "Effect": "Allow",
```

```
        "Action": [
            "ec2:CreateVpcEndpoint",
            "ec2:DescribeVpcEndpoints",
            "ec2:DeleteVpcEndpoints",
        "ec2:CreateTags"
        ],
        "Resource": ["*"],
        "Condition": {
        "StringEquals": { "aws:CalledViaFirst": "deadline.amazonaws.com" }
        }
    },
    {
        "Sid": "ChooseJobAttachmentsBucket",
        "Effect": "Allow",
        "Action": ["s3:GetBucketLocation", "s3:ListAllMyBuckets"],
        "Resource": "*"
    },
    {
        "Sid": "CreateDeadlineCloudLogGroups",
        "Effect": "Allow",
        "Action": ["logs:CreateLogGroup"],
        "Resource": "arn:aws:logs:*:*:log-group:/aws/deadline/*",
        "Condition": {
        "StringLike": { "aws:CalledViaFirst": "deadline.amazonaws.com" }
        }
    },
    {
        "Sid": "ValidateDependencies",
        "Effect": "Allow",
        "Action": ["s3:ListBucket"],
        "Resource": "*",
        "Condition": {
        "StringLike": { "aws:CalledViaFirst": "deadline.amazonaws.com" }
        }
    },
    {
        "Sid": "RoleSelection",
        "Effect": "Allow",
        "Action": ["iam:GetRole", "iam:ListRoles",
"iam:ListAttachedRolePolicies"],
        "Resource": "*"
    },
    {
        "Sid": "PassRoleToDeadlineCloud",
```

```
        "Effect": "Allow",
        "Action": ["iam:PassRole"],
        "Condition": {
        "StringLike": { "iam:PassedToService": "deadline.amazonaws.com" }
    },
        "Resource": "*"
    },
    {
        "Sid": "KMSKeySelection",
        "Effect": "Allow",
        "Action": ["kms:ListKeys", "kms:ListAliases"],
        "Resource": "*"
    },
    {
        "Sid": "IdentityStoreReadOnly",
        "Effect": "Allow",
        "Action": [
            "identitystore:DescribeUser",
            "identitystore:DescribeGroup",
            "identitystore:ListGroups",
            "identitystore:ListUsers",
            "identitystore:IsMemberInGroups",
            "identitystore:ListGroupMemberships",
            "identitystore:ListGroupMembershipsForMember",
            "identitystore:GetGroupMembershipId"
    ],
        "Resource": "*"
    },
    {
        "Sid": "OrganizationAndIdentityCenterIdentification",
        "Effect": "Allow",
        "Action": [
            "sso:ListDirectoryAssociations",
            "organizations:DescribeAccount",
            "organizations:DescribeOrganization",
            "sso:DescribeRegisteredRegions",
            "sso:GetManagedApplicationInstance",
            "sso:GetSharedSsoConfiguration",
            "sso:ListInstances",
            "sso:GetApplicationAssignmentConfiguration",
            "sso:GetSSOStatus",
            "sso:ListRegions",
            "sso:DescribeRegion"
    ],
```

```
        "Resource": "*"
    },
    {
        "Sid": "ManagedDeadlineCloudIDCApplication",
        "Effect": "Allow",
        "Action": [
            "sso:CreateApplication",
            "sso:PutApplicationAssignmentConfiguration",
            "sso:PutApplicationAuthenticationMethod",
            "sso:PutApplicationGrant",
            "sso:DeleteApplication",
            "sso:UpdateApplication"
    ],
        "Resource": "*",
        "Condition": {
        "StringLike": { "aws:CalledViaFirst": "deadline.amazonaws.com" }
        }
    },
    {
        "Sid": "ChooseSecret",
        "Effect": "Allow",
        "Action": ["secretsmanager:ListSecrets"],
        "Resource": "*"
    },
    {
        "Sid": "DeadlineMembershipActions",
        "Effect": "Allow",
        "Action": [
            "deadline:AssociateMemberToFarm",
            "deadline:AssociateMemberToFleet",
            "deadline:AssociateMemberToQueue",
            "deadline:AssociateMemberToJob",
            "deadline:DisassociateMemberFromFarm",
            "deadline:DisassociateMemberFromFleet",
            "deadline:DisassociateMemberFromQueue",
            "deadline:DisassociateMemberFromJob",
            "deadline:ListFarmMembers",
            "deadline:ListFleetMembers",
            "deadline:ListQueueMembers",
            "deadline:ListJobMembers"
    ],
        "Resource": ["*"]
    },
    {
```

```
        "Sid": "DeadlineControlPlaneActions",
        "Effect": "Allow",
        "Action": [
            "deadline:CreateMonitor",
            "deadline:GetMonitor",
            "deadline:UpdateMonitor",
            "deadline:DeleteMonitor",
            "deadline:ListMonitors",
            "deadline:CreateFarm",
            "deadline:GetFarm",
            "deadline:UpdateFarm",
            "deadline:DeleteFarm",
            "deadline:ListFarms",
            "deadline:CreateQueue",
            "deadline:GetQueue",
            "deadline:UpdateQueue",
            "deadline:DeleteQueue",
            "deadline:ListQueues",
            "deadline:CreateFleet",
            "deadline:GetFleet",
            "deadline:UpdateFleet",
            "deadline:DeleteFleet",
            "deadline:ListFleets",
            "deadline:ListWorkers",
            "deadline:CreateQueueFleetAssociation",
            "deadline:GetQueueFleetAssociation",
            "deadline:UpdateQueueFleetAssociation",
            "deadline:DeleteQueueFleetAssociation",
            "deadline:ListQueueFleetAssociations",
            "deadline:CreateQueueEnvironment",
            "deadline:GetQueueEnvironment",
            "deadline:UpdateQueueEnvironment",
            "deadline:DeleteQueueEnvironment",
            "deadline:ListQueueEnvironments",
            "deadline:CreateLimit",
            "deadline:GetLimit",
            "deadline:UpdateLimit",
            "deadline:DeleteLimit",
            "deadline:ListLimits",
            "deadline:CreateQueueLimitAssociation",
            "deadline:GetQueueLimitAssociation",
            "deadline:DeleteQueueLimitAssociation",
            "deadline:UpdateQueueLimitAssociation",
            "deadline:ListQueueLimitAssociations",
```

```
            "deadline:CreateStorageProfile",
            "deadline:GetStorageProfile",
            "deadline:UpdateStorageProfile",
            "deadline:DeleteStorageProfile",
            "deadline:ListStorageProfiles",
            "deadline:ListStorageProfilesForQueue",
            "deadline:ListBudgets",
            "deadline:TagResource",
            "deadline:UntagResource",
            "deadline:ListTagsForResource",
            "deadline:CreateLicenseEndpoint",
            "deadline:GetLicenseEndpoint",
            "deadline:DeleteLicenseEndpoint",
            "deadline:ListLicenseEndpoints",
            "deadline:ListAvailableMeteredProducts",
            "deadline:ListMeteredProducts",
            "deadline:PutMeteredProduct",
            "deadline:DeleteMeteredProduct",
            "deadline:GetMonitorSettings",
            "deadline:UpdateMonitorSettings"
        ],
        "Resource": ["*"]
      }]
}
```

將任務提交至佇列的政策

在此範例中，您會建立縮小範圍政策，授予在特定陣列中將任務提交至特定佇列的許可。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "SubmitJobsFarmAndQueue",
            "Effect": "Allow",
            "Action": "deadline:CreateJob",
            "Resource": "arn:aws:deadline:us-east-1:111122223333:farm/FARM_A/
queue/QUEUE_B/job/*"
        }
    ]
```

```
}
```

允許建立授權端點的政策

在此範例中，您會建立縮小範圍政策，授予建立和管理授權端點所需的許可。使用此政策為與陣列相關聯的 VPC 建立授權端點。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [{
        "Sid": "CreateLicenseEndpoint",
        "Effect": "Allow",
        "Action": [
            "deadline:CreateLicenseEndpoint",
            "deadline:DeleteLicenseEndpoint",
            "deadline:GetLicenseEndpoint",
            "deadline:ListLicenseEndpoints",
            "deadline:PutMeteredProduct",
            "deadline:DeleteMeteredProduct",
            "deadline:ListMeteredProducts",
            "deadline:ListAvailableMeteredProducts",
            "ec2:CreateVpcEndpoint",
            "ec2:DescribeVpcEndpoints",
            "ec2:DeleteVpcEndpoints"
        ],
        "Resource": [
            "arn:aws:deadline:*:111122223333:*",
            "arn:aws:ec2:*:111122223333:vpc-endpoint/*"
        ]
    }]
}
```

允許監控特定陣列佇列的政策

在此範例中，您會建立縮小範圍政策，授予許可來監控特定陣列特定佇列中的任務。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [{
        "Sid": "MonitorJobsFarmAndQueue",
        "Effect": "Allow",
        "Action": [
            "deadline:SearchJobs",
            "deadline:ListJobs",
            "deadline:GetJob",
            "deadline:SearchSteps",
            "deadline:ListSteps",
            "deadline:ListStepConsumers",
            "deadline:ListStepDependencies",
            "deadline:GetStep",
            "deadline:SearchTasks",
            "deadline:ListTasks",
            "deadline:GetTask",
            "deadline:ListSessions",
            "deadline:GetSession",
            "deadline:ListSessionActions",
            "deadline:GetSessionAction"
        ],
        "Resource": [
            "arn:aws:deadline:us-east-1:123456789012:farm/FARM_A/queue/QUEUE_B",
            "arn:aws:deadline:us-east-1:123456789012:farm/FARM_A/queue/QUEUE_B/*"
        ]
    }]
}
```

AWS 截止日期雲端的 受管政策

AWS 受管政策是由 AWS AWS 受管政策建立和管理的獨立政策旨在為許多常用案例提供許可，以便您可以開始將許可指派給使用者、群組和角色。

請記住， AWS 受管政策可能不會授予特定使用案例的最低權限許可，因為這些許可可供所有 AWS 客戶使用。我們建議您定義特定於使用案例的客戶管理政策，以便進一步減少許可。

您無法變更 AWS 受管政策中定義的許可。如果 AWS 更新受 AWS 管政策中定義的許可，則更新會影響政策連接的所有主體身分（使用者、群組和角色）。當新的 AWS 服務 啟動或新的 API 操作可用於現有服務時， AWS 最有可能更新 AWS 受管政策。

如需詳細資訊，請參閱 IAM 使用者指南中的 AWS 受管政策。

AWS 受管政策： AWSDeadlineCloud-FleetWorker

您可以將`AWSDeadlineCloud-FleetWorker`政策連接至 AWS Identity and Access Management (IAM) 身分。

此政策授予此機群中的工作者從服務連線和接收任務所需的許可。

許可詳細資訊

此政策包含以下許可：

- `deadline` – 允許主體管理機群中的工作者。

如需政策詳細資訊的 JSON 清單，請參閱 AWS 受管政策參考指南中的 AWSDeadlineCloud-FleetWorker。

AWS 受管政策： AWSDeadlineCloud-WorkerHost

您可將 `AWSDeadlineCloud-WorkerHost` 政策連接到 IAM 身分。

此政策會授予最初連線至 服務所需的許可。它可以用作 Amazon Elastic Compute Cloud (Amazon EC2) 執行個體描述檔。

許可詳細資訊

此政策包含以下許可：

- `deadline` – 允許使用者建立工作者、擔任工作者的機群角色，以及將標籤套用至工作者

如需政策詳細資訊的 JSON 清單，請參閱 AWS 受管政策參考指南中的 AWSDeadlineCloud-WorkerHost。

AWS 受管政策： AWSDeadlineCloud-UserAccessFarms

您可將 AWSDeadlineCloud-UserAccessFarms 政策連接到 IAM 身分。

此政策可讓使用者根據他們所屬的陣列及其成員層級來存取陣列資料。

許可詳細資訊

此政策包含以下許可：

- deadline – 允許使用者存取陣列資料。
- ec2 – 允許使用者查看 Amazon EC2 執行個體類型的詳細資訊。
- identitystore – 允許使用者查看使用者和群組名稱。
- kms – 允許使用者為其 AWS Key Management Service (IAM Identity Center AWS KMS) 執行個體設定 AWS IAM Identity Center () 客戶受管金鑰。

如需政策詳細資訊的 JSON 清單，請參閱《AWS 受管政策參考指南》中的 AWSDeadlineCloud-UserAccessFarms。

AWS 受管政策： AWSDeadlineCloud-UserAccessFleets

您可將 AWSDeadlineCloud-UserAccessFleets 政策連接到 IAM 身分。

此政策允許使用者根據他們所屬的陣列及其成員層級來存取機群資料。

許可詳細資訊

此政策包含以下許可：

- deadline – 允許使用者存取陣列資料。
- ec2 – 允許使用者查看 Amazon EC2 執行個體類型的詳細資訊。
- identitystore – 允許使用者查看使用者和群組名稱。

如需政策詳細資訊的 JSON 清單，請參閱《AWS 受管政策參考指南》中的 AWSDeadlineCloud-UserAccessFleets。

AWS 受管政策： AWSDeadlineCloud-UserAccessJobs

您可將 AWSDeadlineCloud-UserAccessJobs 政策連接到 IAM 身分。

此政策允許使用者根據他們所屬的陣列及其成員層級來存取任務資料。

許可詳細資訊

此政策包含以下許可：

- `deadline` – 允許使用者存取陣列資料。
- `ec2` – 允許使用者查看 Amazon EC2 執行個體類型的詳細資訊。
- `identitystore` – 允許使用者查看使用者和群組名稱。

如需政策詳細資訊的 JSON 清單，請參閱《AWS 受管政策參考指南》中的 AWSDeadlineCloud-UserAccessJobs。

AWS 受管政策： AWSDeadlineCloud-UserAccessQueues

您可將 `AWSDeadlineCloud-UserAccessQueues` 政策連接到 IAM 身分。

此政策允許使用者根據他們所屬的陣列及其成員層級來存取佇列資料。

許可詳細資訊

此政策包含以下許可：

- `deadline` – 允許使用者存取陣列資料。
- `ec2` – 允許使用者查看 Amazon EC2 執行個體類型的詳細資訊。
- `identitystore` – 允許使用者查看使用者和群組名稱。

如需政策詳細資訊的 JSON 清單，請參閱《AWS 受管政策參考指南》中的 AWSDeadlineCloud-UserAccessQueues。

AWS 受管政策的雲端更新截止日期

檢視自此服務開始追蹤這些變更以來，對 Deadline Cloud AWS 受管政策進行更新的詳細資訊。如需此頁面變更的自動提醒，請訂閱截止日期雲端文件歷史記錄頁面上的 RSS 摘要。

變更	描述	Date
AWSDeadlineCloud-UserAccessFarms – 變更	Deadline Cloud 新增了動作kms:Decrypt ，讓您可以將 AWS KMS 客戶管理的金鑰與 IAM Identity Center 執行個體搭配使用。	2025 年 12 月 22 日
AWSDeadlineCloud-WorkerHost – 變更	Deadline Cloud 新增了動作deadline:TagResource deadline:ListTagsForResource ，並允許您新增和檢視與機群中的工作者相關聯的標籤。	2025 年 5 月 30 日
AWSDeadlineCloud-UserAccessFarms – 變更 AWSDeadlineCloud-UserAccessJobs – 變更 AWSDeadlineCloud-UserAccessQueues – 變更	截止日期 雲端新增了 動作deadline:GetJobTemplate deadline:ListJobParameterDefinitions ，並允許您重新提交任務。	2024 年 10 月 7 日
截止日期 雲端開始追蹤變更	截止日期 雲端開始追蹤其 AWS 受管政策的變更。	2024 年 4 月 2 日

服務角色

Deadline Cloud 如何使用 IAM 服務角色

Deadline Cloud 會自動擔任 IAM 角色，並提供臨時登入資料給工作者、任務和 Deadline Cloud Monitor。這種方法消除了手動憑證管理，同時透過以角色為基礎的存取控制來維護安全性。

當您建立監視器、機群和佇列時，您可以指定 Deadline Cloud 代表您擔任的 IAM 角色。然後，工作者和截止日期雲端監視器會從這些角色接收臨時登入資料以供存取 AWS 服務。

機群角色

設定機群角色，為 Deadline Cloud 工作者提供接收工作所需的許可，並報告該工作的進度。

您通常不需要自行設定此角色。您可以在截止日期雲端主控台中為您建立此角色，以包含必要的許可。使用下列指南來了解此角色的故障診斷詳細資訊。

以程式設計方式建立或更新機群時，請使用 `CreateFleet`或 `UpdateFleet` API 操作指定機群角色 ARN。

機群角色的功能

機群角色為工作者提供以下許可：

- 接收新工作並向截止日期雲端服務報告進行中工作的進度
- 管理工作者生命週期和狀態
- 將工作者日誌的日誌事件記錄到 Amazon CloudWatch Logs

設定機群角色信任政策

您的機群角色必須信任截止日期雲端服務，並限定您的特定陣列範圍。

最佳實務是，信任政策應包含混淆代理人保護的安全條件。若要進一步了解混淆代理人保護，請參閱截止日期雲端使用者指南中的混淆代理人。

- `aws:SourceAccount` 確保只有來自相同 的資源 AWS 帳戶 才能擔任此角色。
- `aws:SourceArn` 將角色假設限制為特定的截止日期雲端陣列。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowDeadlineCredentialsService",
      "Effect": "Allow",
      "Action": "sts:AssumeRole",
      "Principal": {
        "Service": "credentials.deadline.amazonaws.com"
      },
      "Condition": {
        "StringEquals": {
          "aws:SourceAccount": "YOUR_ACCOUNT_ID"
```

```
        },
        "ArnEquals": {
          "aws:SourceArn": "arn:aws:deadline:REGION:YOUR_ACCOUNT_ID:farm/YOUR_FARM_ID"
        }
      }
    }
  ]
}
```

連接機群角色許可

將下列 AWS 受管政策連接至您的機群角色：

AWSDeadlineCloud-FleetWorker

此受管政策提供以下項目的許可：

- deadline:AssumeFleetRoleForWorker - 允許工作者重新整理其登入資料。
- deadline:UpdateWorker - 允許工作者更新其狀態（例如，退出時已停止）。
- deadline:UpdateWorkerSchedule - 用於取得工作和報告進度。
- deadline:BatchGetJobEntity - 用於擷取任務資訊。
- deadline:AssumeQueueRoleForWorker - 用於在任務執行期間存取佇列角色登入資料。

新增加密陣列的 KMS 許可

如果您的陣列是使用 KMS 金鑰建立的，請將這些許可新增至您的機群角色，以確保工作者可以存取陣列中的加密資料。

只有在您的陣列具有相關聯的 KMS 金鑰時，才需要 KMS 許可。kms:ViaService 條件必須使用 格式deadline.*{region}*.amazonaws.com。

建立機群時，會為該機群建立 CloudWatch Logs 日誌群組。Deadline Cloud 服務會使用工作者的許可，專門為該特定工作者建立日誌串流。工作者設定並執行後，工作者將使用這些許可將日誌事件直接傳送到 CloudWatch Logs。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "CreateLogStream",
      "Effect": "Allow",
```

```
      "Action": [
        "logs:CreateLogStream"
      ],
      "Resource": "arn:aws:logs:REGION:YOUR_ACCOUNT_ID:log-group:/aws/
deadline/YOUR_FARM_ID/*",
      "Condition": {
        "ForAnyValue:StringEquals": {
          "aws:CalledVia": [
            "deadline.REGION.amazonaws.com"
          ]
        }
      }
    },
    {
      "Sid": "ManageLogEvents",
      "Effect": "Allow",
      "Action": [
        "logs:PutLogEvents",
        "logs:GetLogEvents"
      ],
      "Resource": "arn:aws:logs:REGION:YOUR_ACCOUNT_ID:log-group:/aws/
deadline/YOUR_FARM_ID/*"
    },
    {
      "Sid": "ManageKmsKey",
      "Effect": "Allow",
      "Action": [
        "kms:Decrypt",
        "kms:DescribeKey",
        "kms:GenerateDataKey"
      ],
      "Resource": "YOUR_FARM_KMS_KEY_ARN",
      "Condition": {
        "StringEquals": {
          "kms:ViaService": "deadline.REGION.amazonaws.com"
        }
      }
    }
  ]
}
```

修改機群角色

機群角色的許可無法自訂。描述的許可始終是必要的，新增額外的許可沒有效果。

客戶管理的機群主機角色

如果您在 Amazon EC2 執行個體或內部部署主機上使用客戶管理的機群，請設定 WorkerHost 角色。

WorkerHost 角色的功能

WorkerHost 角色會在客戶受管機群主機上引導工作者。它提供主機所需的最低許可，以便：

- 在截止日期雲端中建立工作者
- 擔任機群角色以擷取操作登入資料
- 使用機群標籤標記工作者 （ 如果已啟用標籤傳播 ）

設定 WorkerHost 角色許可

將下列 AWS 受管政策連接至 WorkerHost 角色：

AWSDeadlineCloud-WorkerHost

此受管政策提供以下項目的許可：

- `deadline:CreateWorker` - 允許主機註冊新的工作者。
- `deadline:AssumeFleetRoleForWorker` - 允許主機擔任機群角色。
- `deadline:TagResource` - 允許在建立期間標記工作者 （ 如果啟用 ）。
- `deadline:ListTagsForResource` - 允許讀取機群標籤以進行傳播。

了解引導程序

WorkerHost 角色僅在初始工作者啟動期間使用：

1. 工作者代理程式使用 WorkerHost 登入資料在主機上啟動。
2. 它會叫用 `deadline:CreateWorker` 向截止日期雲端註冊。
3. 然後，它會叫用 `deadline:AssumeFleetRoleForWorker`來擷取機群角色登入資料。
4. 從現在開始，工作者只會對所有操作使用機群角色登入資料。

工作者開始執行後，不會使用 WorkerHost 角色。服務受管機群不需要此政策。在服務受管機群中，會自動執行引導。

佇列角色

處理任務時，工作者會擔任佇列角色。此角色提供完成任務所需的許可。

以程式設計方式建立或更新佇列時，請使用 CreateQueue或 UpdateQueue API 操作指定佇列角色 ARN。

設定佇列角色信任政策

您的佇列角色必須信任截止日期雲端服務。

最佳實務是，信任政策應包含混淆代理人保護的安全條件。若要進一步了解混淆代理人保護，請參閱截止日期雲端使用者指南中的混淆代理人。

- aws:SourceAccount 確保只有來自相同 的資源 AWS 帳戶 才能擔任此角色。
- aws:SourceArn 將角色假設限制為特定的截止日期雲端陣列。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": [
          "credentials.deadline.amazonaws.com",
          "deadline.amazonaws.com"
        ]
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "StringEquals": {
          "aws:SourceAccount": "YOUR_ACCOUNT_ID"
        },
        "ArnEquals": {
          "aws:SourceArn": "arn:aws:deadline:us-west-2:123456789012:farm/{farm-id}"
        }
      }
    }
  ]
}
```

了解佇列角色許可

佇列角色不使用單一受管政策。相反地，當您在主控台中設定佇列時，Deadline Cloud 會根據您的組態為您的佇列建立自訂政策。

此自動建立的政策可讓您存取：

任務附件

用於任務輸入和輸出檔案的指定 Amazon S3 儲存貯體讀取和寫入存取權：

```
{
  "Effect": "Allow",
  "Action": [
    "s3:GetObject",
    "s3:PutObject",
    "s3:ListBucket",
    "s3:GetBucketLocation"
  ],
  "Resource": [
    "arn:aws:s3:::YOUR_JOB_ATTACHMENTS_BUCKET",
    "arn:aws:s3:::YOUR_JOB_ATTACHMENTS_BUCKET/YOUR_PREFIX/*"
  ],
  "Condition": {
    "StringEquals": {
      "aws:ResourceAccount": "YOUR_ACCOUNT_ID"
    }
  }
}
```

任務日誌

此佇列中任務的 CloudWatch Logs 讀取存取權。每個佇列都有自己的日誌群組，每個工作階段都有自己的日誌串流：

```
{
  "Effect": "Allow",
  "Action": [
    "logs:GetLogEvents"
  ],
  "Resource": "arn:aws:logs:REGION:YOUR_ACCOUNT_ID:log-group:/aws/
deadline/YOUR_FARM_ID/*"
```

```
}
```

第三方軟體

存取以下載 Deadline Cloud 支援的第三方軟體（例如 Maya、Blender 等）：

```
{
  "Effect": "Allow",
  "Action": [
    "s3:ListBucket",
    "s3:GetObject"
  ],
  "Resource": "*",
  "Condition": {
    "ArnLike": {
      "s3:DataAccessPointArn": "arn:aws:s3:*:*:accesspoint/deadline-software-*"
    },
    "StringEquals": {
      "s3:AccessPointNetworkOrigin": "VPC"
    }
  }
}
```

新增任務的許可

為您的任務需要存取 AWS 服務 的佇列角色新增許可。撰寫 OpenJobDescription 步驟指令碼時，AWS CLI 和 SDK 會自動使用佇列角色的登入資料。使用此項目來存取完成任務所需的其他服務。

範例使用案例包括：

- 用於擷取自訂資料
- 通道到自訂授權伺服器的 SSM 許可
- 用於發出自訂指標的 CloudWatch
- 為動態工作流程建立新任務的截止日期雲端許可

如何使用佇列角色登入資料

Deadline Cloud 提供佇列角色登入資料給：

- 任務執行期間的工作者

- 透過截止日期雲端 CLI 的使用者，並在與任務附件和日誌互動時監控

Deadline Cloud 會為每個佇列建立個別的 CloudWatch Logs 日誌群組。任務使用佇列角色登入資料將日誌寫入其佇列的日誌群組。截止日期雲端 CLI 和監視器使用佇列角色（透過 `deadline:AssumeQueueRoleForRead`) 從佇列的日誌群組讀取任務日誌。截止日期雲端 CLI 和監視器使用佇列角色（透過 `deadline:AssumeQueueRoleForUser`) 上傳或下載任務附件資料。

監控角色

設定監控角色，讓截止日期雲端監控 Web 和桌面應用程式存取您的截止日期雲端資源。

以程式設計方式建立或更新監視器時，請使用 `CreateMonitor`或 `UpdateMonitor` API 操作指定監視器角色 ARN。

監控角色的功能

監控角色可讓 Deadline Cloud Monitor 提供最終使用者存取：

- 截止日期雲端整合提交者、CLI 和監控所需的基本功能
- 最終使用者的自訂功能

設定監控角色信任政策

您的監控角色必須信任截止日期雲端服務。

最佳實務是，信任政策應包含混淆代理人保護的安全條件。若要進一步了解混淆代理人保護，請參閱截止日期雲端使用者指南中的混淆代理人。

`aws:SourceAccount` 確保只有來自相同 的資源 AWS 帳戶 才能擔任此角色。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "credentials.deadline.amazonaws.com"
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "StringEquals": {
```

```
          "aws:SourceAccount": "YOUR_ACCOUNT_ID"
        }
      }
    }
  ]
}
```

連接監視器角色許可

將下列所有 AWS 受管政策連接至監視器角色以進行基本操作：

- AWSDeadlineCloud-UserAccessFarms
- AWSDeadlineCloud-UserAccessFleets
- AWSDeadlineCloud-UserAccessJobs
- AWSDeadlineCloud-UserAccessQueues

監控角色的運作方式

使用截止日期雲端監視器時，服務使用者會使用 AWS IAM Identity Center (IAM Identity Center) 登入，並擔任監視器角色。監控應用程式會使用擔任的角色登入資料來顯示監控 UI，包括陣列、機群、佇列和其他資訊的清單。

使用 Deadline Cloud Monitor 桌面應用程式時，這些登入資料會使用與最終使用者提供的設定檔名稱對應的具名 AWS 登入資料設定檔，在工作站上另外提供。請參閱 AWS SDK 和工具參考指南，進一步了解具名設定檔。

此具名設定檔是截止日期 CLI 和提交者存取截止日期雲端資源的方式。

自訂進階使用案例的監控角色

您可以自訂監控角色，以修改使用者在每個存取層級（檢視器、貢獻者、管理員、擁有者）可以執行的操作，或新增進階工作流程的許可。

自訂存取層級許可

連接到監控角色的四個 AWS 受管政策控制每個存取層級可以執行的操作。您可以將自訂政策新增至監控角色，以使用 `deadline:MembershipLevel`條件金鑰授予或限制特定存取層級的許可。

例如，若要允許貢獻者更新和取消任務（通常僅限於經理和擁有者），請新增如下所示的政策：

```
{
```

```
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "deadline:UpdateJob",
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "deadline:MembershipLevel": "CONTRIBUTOR"
        }
      }
    }
  ]
}
```

透過此政策，貢獻者除了提交任務之外，還可以更新和取消任務。

新增進階工作流程的許可

您可以將自訂 IAM 政策新增至監控角色，以授予其他許可給所有監控使用者。這適用於進階指令碼工作流程，其中使用者需要存取標準截止日期雲端功能 AWS 服務 以外的 。

修改您的監控角色時，請遵循下列準則：

- 請勿移除任何 受管政策。移除這些政策會中斷監視器功能。

Deadline Cloud Monitor 如何使用監控角色登入資料

期限 雲端監視器會在您驗證時自動取得監控角色登入資料。此功能可讓桌面應用程式提供增強型監控功能，超越標準 Web 瀏覽器中可用的功能。

當您使用截止日期雲端監視器登入時，它會自動建立您可以與 AWS CLI 或任何其他 AWS 工具搭配使用的設定檔。此設定檔使用監控角色登入資料， AWS 服務 可讓您根據監控角色中的許可，以程式設計方式存取 。

截止日期雲端提交者的運作方式相同 - 他們使用截止日期雲端監視器建立的設定檔，以 AWS 服務 適當的角色許可存取 。

期限雲端角色的進階自訂

您可以使用其他許可擴展截止日期雲端角色，以啟用基本轉譯工作流程以外的進階使用案例。此方法利用 Deadline Cloud 的存取管理系統， AWS 服務 根據佇列成員資格控制對其他 的存取。

與 的團隊協作 AWS CodeCommit

將 AWS CodeCommit 許可新增至佇列角色，以在專案儲存庫上啟用團隊協作。此方法將 Deadline Cloud 的存取管理系統用於其他使用案例 - 只有具有特定佇列存取權的使用者才會收到這些 AWS CodeCommit 許可，允許您透過 Deadline Cloud 佇列成員資格管理每個專案儲存庫的存取權。

這對於藝術家在轉譯工作流程中需要存取儲存在 AWS CodeCommit 儲存庫中的專案特定資產、指令碼或組態檔案的情況非常有用。

將 AWS CodeCommit 許可新增至佇列角色

將下列許可新增至佇列角色以啟用 AWS CodeCommit 存取：

```
{
  "Effect": "Allow",
  "Action": [
    "codecommit:GitPull",
    "codecommit:GitPush",
    "codecommit:GetRepository",
    "codecommit:ListRepositories"
  ],
  "Resource": "arn:aws:codecommit:REGION:YOUR_ACCOUNT_ID:PROJECT_REPOSITORY"
}
```

在藝術家工作站上設定登入資料提供者

將每個藝術家工作站設定為使用截止日期雲端佇列登入資料進行 AWS CodeCommit 存取。此設定會在每個工作站完成一次。

設定登入資料提供者

1. 將登入資料提供者設定檔新增至您的組態檔案 AWS (~/.aws/config)：

   ```
   [profile queue-codecommit]
   credential_process = deadline queue export-credentials --farm-id farm-
   XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX --queue-id queue-XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
   ```

2. 設定 Git 以將此設定檔用於 AWS CodeCommit 儲存庫：

   ```
   git config --global credential.https://git-codecommit.REGION.amazonaws.com.helper
    '!aws codecommit credential-helper --profile queue-codecommit $@'
   ```

```
git config --global credential.https://git-
codecommit.REGION.amazonaws.com.UseHttpPath true
```

將 *farm-XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX* 和 *queue-XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX* 取代為您實際的陣列和佇列 IDs。將 *REGION* 取代為您的 AWS 區域 （例如 us-west-2)。

使用 AWS CodeCommit 搭配佇列登入資料

設定完成後，Git 操作會在存取 AWS CodeCommit 儲存庫時自動使用佇列角色登入資料。deadline queue export-credentials 命令會傳回如下所示的臨時登入資料：

```
{
  "Version": 1,
  "AccessKeyId": "ASIA...",
  "SecretAccessKey": "...",
  "SessionToken": "...",
  "Expiration": "2025-11-10T23:02:23+00:00"
}
```

這些登入資料會視需要自動重新整理，Git 操作將順暢運作：

```
git clone https://git-codecommit.REGION.amazonaws.com/v1/repos/PROJECT_REPOSITORY
git pull
git push
```

藝術家現在可以使用其佇列許可存取專案儲存庫，而不需要個別的 AWS CodeCommit 登入資料。只有具有特定佇列存取權的使用者才能存取相關聯的儲存庫，透過截止日期雲端的佇列成員資格系統啟用精細存取控制。

對 AWS 截止日期雲端身分和存取進行故障診斷

使用以下資訊來協助您診斷和修正使用截止日期雲端和 IAM 時可能遇到的常見問題。

主題

- 我無權在截止日期雲端中執行動作
- 我未獲得執行 iam:PassRole 的授權
- 我想要允許 以外的人員 AWS 帳戶 存取我的截止日期雲端資源

我無權在截止日期雲端中執行動作

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 deadline:*GetWidget* 許可時發生。

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 deadline: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 deadline:*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，您的政策必須更新，以允許您將角色傳遞至截止日期雲端。

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 的 IAM marymajor 使用者嘗試使用主控台在截止日期雲端中執行動作時，會發生下列範例錯誤。但是，動作請求服務具備服務角色授予的許可。Mary 沒有將角色傳遞給服務的許可。

```
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 存取我的截止日期雲端資源

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：

- 若要了解截止日期雲端是否支援這些功能，請參閱 Deadline Cloud 如何與 IAM 搭配使用。

- 若要了解如何在您擁有 AWS 帳戶 的 資源之間提供存取權，請參閱《IAM 使用者指南》中的在您擁有 AWS 帳戶 的另一個 中提供存取權給 IAM 使用者。
- 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》中的將存取權提供給第三方 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。

的合規驗證 Deadline Cloud

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 合規計劃。

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。

您使用 時的合規責任 AWS 服務 取決於資料的機密性、您公司的合規目標，以及適用的法律和法規。如需使用 時合規責任的詳細資訊 AWS 服務，請參閱 AWS 安全文件。

中的彈性 Deadline Cloud

AWS 全球基礎設施是以 AWS 區域 和 可用區域為基礎建置。 AWS 區域 提供多個實體隔離和隔離的可用區域，這些可用區域與低延遲、高輸送量和高備援聯網連接。透過可用區域，您可以設計與操作的應用程式和資料庫，在可用區域之間自動容錯移轉而不會發生中斷。可用區域的可用性、容錯能力和擴展能力，均較單一或多個資料中心的傳統基礎設施還高。

如需 AWS 區域 和可用區域的詳細資訊，請參閱 AWS 全球基礎設施。

AWS Deadline Cloud 不會備份存放在任務附件 S3 儲存貯體中的資料。您可以使用任何標準 Amazon S3 備份機制來啟用任務連接資料的備份，例如 S3 版本控制或 AWS Backup。

截止日期雲端中的基礎設施安全性

做為受管服務， AWS Deadline Cloud 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來設計您的 AWS 環境，請參閱安全支柱 AWS Well-Architected Framework 中的基礎設施保護。

您可以使用 AWS 發佈的 API 呼叫，透過網路存取截止日期雲端。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。

Deadline Cloud 不支援使用 AWS PrivateLink 虛擬私有雲端 (VPC) 端點政策。它使用 AWS PrivateLink 預設政策，授予端點的完整存取權。如需詳細資訊，請參閱AWS PrivateLink 《 使用者指南》中的預設端點政策。

截止日期雲端中的組態和漏洞分析

AWS 處理基本安全任務，例如訪客作業系統 (OS) 和資料庫修補、防火牆組態和災難復原。這些程序已由適當的第三方進行檢閱並認證。如需詳細資訊，請參閱以下資源：

- 共同的責任模型
- Amazon Web Services：安全程序概觀 (白皮書)

AWS Deadline Cloud 會管理服務受管或客戶受管機群上的任務：

- 對於服務受管機群，Deadline Cloud 會管理訪客作業系統。
- 對於客戶管理的機群，您需負責管理作業系統。

如需有關 AWS 截止日期雲端組態和漏洞分析的其他資訊，請參閱

- 截止日期雲端的安全最佳實務

預防跨服務混淆代理人

混淆代理人問題屬於安全性問題，其中沒有執行動作許可的實體可以強制具有更多許可的實體執行該動作。在 中 AWS，跨服務模擬可能會導致混淆代理人問題。在某個服務 (呼叫服務) 呼叫另一個服務 (被呼叫服務) 時，可能會發生跨服務模擬。可以操縱呼叫服務來使用其許可，以其不應有存取許可的方式對其他客戶的資源採取動作。為了預防這種情況， AWS 提供的工具可協助您保護所有服務的資料，而這些服務主體已獲得您帳戶中資源的存取權。

我們建議在資源政策中使用 aws:SourceArn和 aws:SourceAccount全域條件內容索引鍵，以限制將另一個服務 AWS Deadline Cloud 提供給資源的許可。如果您想要僅允許一個資源與跨服務存取相關

聯，則請使用 aws:SourceArn。如果您想要允許該帳戶中的任何資源與跨服務使用相關聯，請使用 aws:SourceAccount。

防範混淆代理人問題的最有效方法是使用 aws:SourceArn 全域條件內容索引鍵，其中包含資源的完整 Amazon Resource Name (ARN)。如果不知道資源的完整 ARN，或者如果您指定了多個資源，請使用 aws:SourceArn 全域內容條件索引鍵搭配萬用字元 (*) 來表示 ARN 的未知部分。例如 arn:aws:deadline:*:*123456789012*:*。

如果 aws:SourceArn 值不包含帳戶 ID (例如 Amazon S3 儲存貯體 ARN)，您必須使用這兩個全域條件內容索引鍵來限制許可。

下列範例示範如何在 中使用 aws:SourceArn和 aws:SourceAccount全域條件內容索引鍵 Deadline Cloud ，以防止混淆代理人問題。

JSON

```
{
  "Version":"2012-10-17",
  "Statement": {
    "Sid": "ConfusedDeputyPreventionExamplePolicy",
    "Effect": "Allow",
    "Principal": {
      "Service": "deadline.amazonaws.com"
    },
    "Action": "deadline:CreateFarm",
    "Resource": [
      "*"
    ],
    "Condition": {
      "ArnLike": {
        "aws:SourceArn": "arn:aws:deadline:*:111122223333:*"
      },
      "StringEquals": {
        "aws:SourceAccount": "111122223333"
      }
    }
  }
}
```

AWS Deadline Cloud 使用界面端點存取 (AWS PrivateLink)

您可以使用 在 VPC 與 之間 AWS PrivateLink 建立私有連線 AWS Deadline Cloud。您可以 Deadline Cloud 像在 VPC 中一樣存取 ，無需使用網際網路閘道、NAT 裝置、VPN 連接或 Direct Connect 連線。VPC 中的執行個體不需要公有 IP 地址即可存取 Deadline Cloud。

您可以建立由 AWS PrivateLink提供支援的介面端點來建立此私有連線。我們會在您為介面端點啟用的每個子網中建立端點網路介面。這些是請求者管理的網路介面，可作為目的地為 Deadline Cloud之流量的進入點。

Deadline Cloud 也提供雙堆疊端點。雙堆疊端點支援透過 IPv6 和 IPv4 的請求。

如需詳細資訊，請參閱《AWS PrivateLink 指南》中的「透過 AWS PrivateLink存取 AWS 服務」。

的考量事項 Deadline Cloud

設定介面端點之前 Deadline Cloud，請參閱《 AWS PrivateLink 指南》中的使用介面 VPC 端點存取 AWS 服務。

Deadline Cloud 支援透過界面端點呼叫其所有 API 動作。

根據預設， Deadline Cloud 允許透過介面端點完整存取 。或者，您可以將安全群組與端點網路介面建立關聯，以控制 Deadline Cloud 透過介面端點傳入 的流量。

Deadline Cloud 也支援 VPC 端點政策。如需詳細資訊，請參閱《AWS PrivateLink 指南》中的使用端點政策控制 VPC 端點的存取。

Deadline Cloud 端點

Deadline Cloud 使用四個端點來存取服務 AWS PrivateLink - 兩個用於 IPv4，兩個用於 IPv6。

工作者使用 `scheduling.deadline.region.amazonaws.com`端點從佇列取得任務、向 報告進度 Deadline Cloud，以及將任務輸出傳回。如果您使用客戶受管機群，除非您使用管理操作，否則排程端點是唯一需要建立的端點。例如，如果任務建立更多任務，您需要啟用管理端點來呼叫 `CreateJob`操作。

Deadline Cloud 監視器使用 `management.deadline.region.amazonaws.com`來管理陣列中的資源，例如建立和修改佇列和機群，或取得任務、步驟和任務的清單。

AWS SDKs 和 CLI 會自動將 `management`和 字`scheduling`首新增至端點。如果您想要停用此行為，請參閱 AWS SDKs和工具參考指南中的主機字首注入一節。

Deadline Cloud 也需要下列 AWS 服務端點的端點：

- 如果您在沒有網際網路連線的子網路中設定客戶受管機群，則必須為 Amazon CloudWatch Logs 建立 VPC 端點，以便工作者可以寫入日誌。如需詳細資訊，請參閱使用 CloudWatch 監控。
- 如果您使用任務附件，則必須為 Amazon Simple Storage Service (Amazon S3) 建立 VPC 端點，以便工作者可以存取附件。如需詳細資訊，請參閱 中的任務附件 Deadline Cloud。

建立 的端點 Deadline Cloud

您可以使用 Amazon VPC Deadline Cloud 主控台或 AWS Command Line Interface () 為 建立介面端點 AWS CLI。如需詳細資訊，請參閱《AWS PrivateLink 指南》中的「建立介面端點」。

Deadline Cloud 使用下列服務名稱建立 的管理和排程端點。將##取代為您已部署 AWS 區域 的 Deadline Cloud。

```
com.amazonaws.region.deadline.management
```

```
com.amazonaws.region.deadline.scheduling
```

Deadline Cloud 支援雙堆疊端點。

如果您為介面端點啟用私有 DNS，您可以使用 Deadline Cloud 其預設的區域 DNS 名稱向 提出 API 請求。例如，`scheduling.deadline.us-east-1.amazonaws.com`用於工作者操作，或`management.deadline.us-east-1.amazonaws.com`用於所有其他操作。

如果您的客戶受管機群位於沒有網際網路連線的子網路上，您必須使用下列服務名稱建立 CloudWatch Logs 端點：

```
com.amazonaws.region.logs
```

如果您使用任務附件傳輸檔案，則必須使用以下服務名稱建立 Amazon S3 端點：

```
com.amazonaws.region.s3
```

受限的網路環境

Deadline Cloud 提供藝術家或其他使用者在其本機工作站上使用的工具。這些工具需要存取 AWS API 和 Web 端點才能執行其功能。如果您使用新一代防火牆 (NGFW) 或安全 Web Gateway (SWG) 等

Web 內容篩選解決方案來篩選特定 AWS 網域或 URL 端點的存取權，則必須將下列網域或 URL 端點新增至 Web 內容篩選解決方案允許清單。

AWS 要允許清單的 API 端點

截止日期雲端用戶端工具，例如 AWS 管理主控台、監視器、CLI 和整合式提交者，除了截止日期雲端之外，還需要存取 AWS APIs。這些端點僅支援 IPv4。

- scheduling.deadline.*[Region]*.amazonaws.com
- management.deadline.*[Region]*.amazonaws.com
- logs.*[Region]*.amazonaws.com
- ec2.*[Region]*.amazonaws.com
- s3.*[Region]*.amazonaws.com
- sts.*[Region]*.amazonaws.com
- identitystore.*[Region]*.amazonaws.com

要允許清單的 Web 網域

Deadline Cloud Monitor 需要存取下列網域才能操作。

如需允許列出 AWS 登入網域的詳細資訊，請參閱AWS 《 登入使用者指南》中的要新增至允許清單的網域。

- downloads.deadlinecloud.amazonaws.com
- d2ev1rdnjzhmnr.cloudfront.net
- prod.log.shortbread.aws.dev
- prod.tools.shortbread.aws.dev
- prod.log.shortbread.analytics.console.aws.a2z.com
- prod.tools.shortbread.analytics.console.aws.a2z.com
- global.help-panel.docs.aws.a2z.com
- *[Region]*.signin.aws
- *[Region]*.signin.aws.amazon.com
- sso.*[Region]*.amazonaws.com
- portal.sso.*[Region]*.amazonaws.com

- oidc.*[Region]*.amazonaws.com
- assets.sso-portal.*[Region]*.amazonaws.com

要允許清單的環境特定端點

這些網域會根據截止日期雲端的特定組態而有所不同。如果建立其他截止日期雲端監視器或佇列，則需要允許列出其他網域。

- *[Directory ID or alias]*.awsapps.com

  此網域繫結至 IAM Identity Center 設定，且此 中的所有設定應使用相同的 IAM Identity Center 執行個體。您可以在設定 → AWS 存取入口網站 URL 下的 IAM Identity Center 主控台中找到確切的值。

- *[Monitor alias]*.*[Region]*.deadlinecloud.amazonaws.com

  此網域適用於截止日期雲端中的監控設定。藝術家在其瀏覽器或截止日期雲端監控應用程式中輸入此連結。如果未來在其他帳戶或區域中設定截止日期雲端，此網域將會變更。您可以在儀表板 → 監控概觀 → 監控詳細資訊 → URL 的截止日期雲端主控台中找到此值。

- *[Bucket name]*.*[Region]*.s3.amazonaws.com

  這是截止日期雲端佇列所使用的任務附件儲存貯體網域。每個佇列都可以設定自己的任務附件儲存貯體。您可以在 佇列 → 佇列詳細資訊 → 任務附件下的截止日期雲端主控台中找到確切的儲存貯體名稱。如需任務附件的詳細資訊，請參閱佇列文件。

截止日期雲端的安全最佳實務

AWS Deadline Cloud (Deadline Cloud) 提供許多安全功能，供您在開發和實作自己的安全政策時考慮。以下最佳實務為一般準則，並不代表完整的安全解決方案。這些最佳實務可能不適用或無法滿足您的環境需求，因此請將其視為實用建議就好，而不要當作是指示。

> **Note**
>
> 如需許多安全主題重要性的詳細資訊，請參閱共同責任模型。

資料保護

基於資料保護目的，我們建議您保護 AWS 帳戶 登入資料，並使用 AWS Identity and Access Management (IAM) 設定個別帳戶。如此一來，每個使用者都只會獲得授與完成其任務所必須的許可。我們也建議您採用下列方式保護資料：

- 每個帳戶均要使用多重要素驗證 (MFA)。
- 使用 SSL/TLS 與 AWS 資源通訊。我們需要 TLS 1.2 並建議使用 TLS 1.3。
- 使用 設定 API 和使用者活動記錄 AWS CloudTrail。
- 使用 AWS 加密解決方案，以及 中的所有預設安全控制 AWS 服務。
- 使用進階受管安全服務，例如 Amazon Macie，可協助探索和保護存放在 Amazon Simple Storage Service (Amazon S3) 中的個人資料。
- 如果您在透過命令列介面或 API 存取 AWS 時，需要 FIPS 140-2 驗證的加密模組，請使用 FIPS 端點。如需 FIPS 和 FIPS 端點的詳細資訊，請參閱聯邦資訊處理標準 (FIPS) 140-2 概觀。

我們強烈建議您絕對不要將客戶帳戶號碼等敏感的識別資訊，放在自由格式的欄位中，例如名稱欄位。此建議包括當您使用 AWS Deadline Cloud 或使用主控台、API AWS CLI或其他 AWS 服務 時 AWS SDKs 。您在截止日期雲端或其他服務中輸入的任何資料都可能被挑選納入診斷日誌中。當您提供外部伺服器的 URL 時，請勿在驗證您對該伺服器請求的 URL 中包含登入資料資訊。

AWS Identity and Access Management 許可

使用使用者、 AWS Identity and Access Management (IAM) 角色，以及將最低權限授予使用者，來管理對 AWS 資源的存取。建立憑證管理政策和程序，以建立、分發、輪換和撤銷 AWS 存取憑證。如需詳細資訊，請參《IAM 使用者指南》中的 IAM 最佳實務。

以使用者和群組身分執行任務

在截止日期雲端中使用佇列功能時，最佳實務是指定作業系統 (OS) 使用者及其主要群組，以便作業系統使用者具有佇列任務的最低權限許可。

當您指定「以使用者身分執行」（和群組）時，提交至佇列之任務的任何程序都會使用該作業系統使用者執行，並繼承該使用者的相關聯作業系統許可。

機群和佇列組態結合以建立安全狀態。在佇列端，「以使用者身分執行任務」和 IAM 角色可指定為使用佇列任務的作業系統和 AWS 許可。機群定義基礎設施（工作者主機、網路、掛載的共用儲存體），當與特定佇列相關聯時，會在佇列中執行任務。工作者主機上可用的資料需要由一或多個相關聯

佇列中的任務存取。指定使用者或群組有助於保護任務中的資料免受其他佇列、其他已安裝的軟體或其他可存取工作者主機的使用者影響。當佇列沒有使用者時，它會以代理程式使用者身分執行，其可以模擬 (sudo) 任何佇列使用者。如此一來，沒有使用者的佇列就可以將權限提升到另一個佇列。

聯網

為了防止流量遭到攔截或重新導向，請務必保護網路流量的路由方式和位置。

建議您以下列方式保護您的聯網環境：

- 保護 Amazon Virtual Private Cloud (Amazon VPC) 子網路路由表，以控制 IP 層流量的路由方式。
- 如果您在陣列或工作站設定中使用 Amazon Route 53 (Route 53) 做為 DNS 供應商，請安全地存取 Route 53 API。
- 如果您 AWS 使用內部部署工作站或其他資料中心連線到 以外的 Deadline Cloud，請保護任何內部部署聯網基礎設施。這包括路由器、交換器和其他聯網裝置上的 DNS 伺服器和路由表。

任務和任務資料

截止日期 雲端任務會在工作者主機的工作階段中執行。每個工作階段都會在工作者主機上執行一或多個程序，這通常會要求您輸入資料才能產生輸出。

若要保護此資料，您可以使用佇列設定作業系統使用者。工作者代理程式會使用佇列作業系統使用者來執行工作階段子程序。這些子程序會繼承佇列作業系統使用者的許可。

我們建議您遵循最佳實務，以安全存取這些子程序存取的資料。如需詳細資訊，請參閱共同責任模式。

陣列結構

您可以透過多種方式安排截止日期雲端機群和佇列。不過，某些安排會有安全性影響。

一個陣列具有最安全的界限之一，因為它無法與其他陣列共用截止日期雲端資源，包括機群、佇列和儲存設定檔。不過，您可以在陣列內共用外部 AWS 資源，這會危及安全界限。

您也可以使用適當的組態，在相同陣列內的佇列之間建立安全界限。

請遵循下列最佳實務，在相同的陣列中建立安全佇列：

- 僅將機群與相同安全界限內的佇列建立關聯。注意下列事項：
  - 在工作者主機上執行任務後，資料可能會保留在後方，例如暫存目錄或佇列使用者的主目錄中。

  - 無論您提交任務的佇列為何，相同的作業系統使用者都會在服務擁有的機群工作者主機上執行所有任務。
  - 任務可能會讓程序在工作者主機上執行，讓來自其他佇列的任務能夠觀察其他執行中的程序。
- 確保只有相同安全界限內的佇列才能共用任務附件的 Amazon S3 儲存貯體。
- 確保只有相同安全界限內的佇列才能共用作業系統使用者。
- 保護與陣列整合的任何其他 AWS 資源到邊界。

任務連接佇列

任務附件與使用 Amazon S3 儲存貯體的佇列相關聯。

- 寫入 Amazon S3 儲存貯體中根字首並從中讀取的任務附件。您可以在 `CreateQueue` API 呼叫中指定此根字首。
- 儲存貯體具有對應的 `Queue Role`，指定授予佇列使用者存取儲存貯體和根字首的角色。建立佇列時，您可以指定 `Queue Role` Amazon Resource Name (ARN) 以及任務附件儲存貯體和根字首。
- 對 `AssumeQueueRoleForRead`、 `AssumeQueueRoleForUser`和 `AssumeQueueRoleForWorker` API 操作的授權呼叫會傳回一組 的臨時安全登入資料`Queue Role`。

如果您建立佇列並重複使用 Amazon S3 儲存貯體和根字首，則會有向未經授權方公開資訊的風險。例如，QueueA 和 QueueB 共用相同的儲存貯體和根字首。在安全工作流程中，ArtistA 可以存取 QueueA，但無法存取 QueueB。不過，當多個佇列共用儲存貯體時，ArtistA 可以存取 QueueB 資料中的資料，因為它使用與 QueueA 相同的儲存貯體和根字首。

主控台預設會設定安全的佇列。確保佇列具有 Amazon S3 儲存貯體和根字首的不同組合，除非它們是常見安全界限的一部分。

若要隔離佇列，您必須`Queue Role`將 設定為僅允許佇列存取儲存貯體和根字首。在下列範例中，將每個`####`取代為您的資源特定資訊。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
```

```
            "Action": [
                "s3:GetObject",
                "s3:PutObject",
                "s3:ListBucket",
                "s3:GetBucketLocation"
            ],
            "Effect": "Allow",
            "Resource": [
                "arn:aws:s3:::JOB_ATTACHMENTS_BUCKET_NAME",

 "arn:aws:s3:::JOB_ATTACHMENTS_BUCKET_NAME/JOB_ATTACHMENTS_ROOT_PREFIX/*"
            ],
            "Condition": {
                "StringEquals": {
                    "aws:ResourceAccount": "111122223333"
                }
            }
        },
        {
            "Action": [
                "logs:GetLogEvents"
            ],
            "Effect": "Allow",
            "Resource": "arn:aws:logs:us-east-1:111122223333:log-group:/aws/
deadline/FARM_ID/*"
        }
    ]
}
```

您也必須在角色上設定信任政策。在下列範例中，將####文字取代為您的資源特定資訊。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Action": [
                "sts:AssumeRole"
            ],
            "Effect": "Allow",
            "Principal": {
```

```
                "Service": "deadline.amazonaws.com"
            },
            "Condition": {
                "StringEquals": {
                    "aws:SourceAccount": "111122223333"
                },
                "ArnEquals": {
                    "aws:SourceArn": "arn:aws:deadline:us-
east-1:111122223333:farm/FARM_ID"
                }
            }
        },
        {
            "Action": [
                "sts:AssumeRole"
            ],
            "Effect": "Allow",
            "Principal": {
                "Service": "credentials.deadline.amazonaws.com"
            },
            "Condition": {
                "StringEquals": {
                    "aws:SourceAccount": "111122223333"
                },
                "ArnEquals": {
                    "aws:SourceArn": "arn:aws:deadline:us-
east-1:111122223333:farm/FARM_ID"
                }
            }
        }
    ]
}
```

自訂軟體 Amazon S3 儲存貯體

您可以將下列陳述式新增至 ，Queue Role以存取 Amazon S3 儲存貯體中的自訂軟體。在下列範例中，將 *Software_BUCKET_NAME* 取代為 S3 儲存貯體的名稱，並將 *BUCKET_ACCOUNT_OWNER* 取代為擁有儲存貯體的 AWS 帳戶 ID。

```
"Statement": [
    {
        "Action": [
```

```
            "s3:GetObject",
            "s3:ListBucket"
        ],
        "Effect": "Allow",
        "Resource": [
            "arn:aws:s3:::SOFTWARE_BUCKET_NAME",
            "arn:aws:s3:::SOFTWARE_BUCKET_NAME/*"
        ],
        "Condition": {
         "StringEquals": {
            "aws:ResourceAccount": "BUCKET_ACCOUNT_OWNER"
         }
      }
    }
]
```

如需 Amazon S3 安全最佳實務的詳細資訊，請參閱《Amazon Simple Storage Service 使用者指南》中的 Amazon [Amazon S3 安全最佳實務](#)。

工作者主機

保護工作者主機，以協助確保每個使用者只能對其指派的角色執行操作。

建議使用下列最佳實務來保護工作者主機：

- 使用主機組態指令碼可以變更工作者的安全性和操作。不正確的組態可能會導致工作者不穩定或停止運作。您需負責偵錯此類故障。
- 除非提交到這些佇列的任務位於相同的安全界限內，否則請勿對多個佇列使用相同的`jobRunAsUser`值。
- 請勿將佇列設定為工作者代理程式執行時所使用的`jobRunAsUser`作業系統使用者名稱。
- 授予佇列使用者預期佇列工作負載所需的最低權限作業系統許可。確保他們沒有工作代理程式檔案或其他共用軟體的檔案系統寫入許可。
- 確保只有 上的根使用者Linux和 `Administrator`擁有Windows自己的帳戶，並且可以修改工作者代理程式檔案。
- 在Linux工作者主機上，請考慮在 中設定umask覆寫`/etc/sudoers`，以允許工作者代理程式使用者以佇列使用者身分啟動程序。此組態有助於確保其他使用者無法存取寫入佇列的檔案。
- 授予受信任的個人對工作者主機的最低權限存取權。
- 將許可限制為本機 DNS 覆寫組態檔案 (`/etc/hosts` 上的 Linux和 `C:\Windows\system32\etc\hosts`上的 Windows)，以及在工作站和工作者主機作業系統上路由資料表。

- 限制對工作站和工作者主機作業系統上 DNS 組態的許可。
- 定期修補作業系統和所有已安裝的軟體。此方法包括專門與截止日期雲端搭配使用的軟體，例如提交者、轉接器、工作者代理程式、OpenJD套件等。
- 針對Windows佇列 使用強式密碼`jobRunAsUser`。
- 定期輪換佇列 的密碼`jobRunAsUser`。
- 確保最低權限存取Windows密碼秘密，並刪除未使用的秘密。
- 請勿將排程命令提供給佇列`jobRunAsUser`許可，以在未來執行：
  - 在 上Linux，拒絕這些帳戶存取 `cron`和 `at`。
  - 在 上Windows，拒絕這些帳戶存取Windows任務排程器。

> **Note**
>
> 如需定期修補作業系統和已安裝軟體之重要性的詳細資訊，請參閱共同責任模型。

主機組態指令碼

- 使用主機組態指令碼可以變更工作者的安全性和操作。不正確的組態可能會導致工作者不穩定或停止運作。您需負責偵錯此類故障。

工作站

請務必保護可存取截止日期雲端的工作站。此方法有助於確保您提交至 Deadline Cloud 的任何任務都無法執行向您的 計費的任意工作負載 AWS 帳戶。

我們建議採用下列最佳實務來保護藝術家工作站。如需詳細資訊，請參閱 共同責任模型。

- 保護任何提供存取權的持久憑證 AWS，包括截止日期雲端。如需詳細資訊，請參閱《 IAM 使用者指南》中的管理 IAM 使用者的存取金鑰。
- 僅安裝受信任且安全的軟體。
- 要求使用者與身分提供者聯合使用 AWS 臨時憑證存取 。
- 在截止日期雲端提交者程式檔案上使用安全許可，以防止竄改。
- 授予受信任的個人對藝術家工作站的最低權限存取權。
- 僅使用您透過截止日期雲端監視器取得的提交者和轉接器。

- 將許可限制為本機 DNS 覆寫組態檔案 (/etc/hosts Linux和 上的 macOS，以及 C:\Windows \system32\etc\hosts上的 Windows)，以及在工作站和工作者主機作業系統上路由資料表。
- 限制 對工作站和工作者主機作業系統/etc/resolve.conf的許可。
- 定期修補作業系統和所有已安裝的軟體。此方法包括專門與截止日期雲端搭配使用的軟體，例如提交者、轉接器、工作者代理程式、OpenJD套件等。

驗證下載軟體的真實性

在下載安裝程式後驗證軟體的真偽，以防止檔案遭到竄改。此程序適用於 Windows和 Linux系統。

Windows

若要驗證下載檔案的真實性，請完成下列步驟。

1. 在下列命令中，將 *file*取代為您要驗證的檔案。例如 ***C:\PATH\TO\MY \DeadlineCloudSubmitter-windows-x64-installer.exe*** 。此外，請將 取代*signtool-sdk-version*為已安裝的 SignTool SDK 版本。例如 **10.0.22000.0**。

   ```
   "C:\Program Files (x86)\Windows Kits\10\bin\signtool-sdk-version\x86\signtool.exe" verify /vfile
   ```

2. 例如，您可以執行下列命令來驗證截止日期雲端提交者安裝程式檔案：

   ```
   "C:\Program Files (x86)\Windows Kits\10\bin
   \10.0.22000.0\x86\signtool.exe" verify /v DeadlineCloudSubmitter-
   windows-x64-installer.exe
   ```

Linux

若要驗證下載檔案的真實性，請使用gpg命令列工具。

1. 執行下列命令來匯入OpenPGP金鑰：

   ```
    gpg --import --armor <<EOF
   -----BEGIN PGP PUBLIC KEY BLOCK-----

   mQINBGlANDUBEACg6zffjN43gqe5ryPhk+wQM10rEdvmItw4WPWaVsN+/at/OIJw
   MGCagSYXcgR+jKbsHQOQoEQdo5SrxxHjpKTEs3KQhGvf+ehrU1Ac7koXKIBWtes+
   BI9F0slRECz0nXTOy/cd/90RXjpF07mreTLIKNIbybULfad82nYykpITjFr5XRGj
   /shYkucxRQZdwkgkIYyV25pPICPd2RsX+Zua85jV8mCqVffDfRXvgcPe3+ofClj/
   ```

```
2CE8UfUIqO8Csua4YEkSqr3aeoTOEFT4kuQR5nFXVzorOEkQtO3gB35KNWKMlIOU
2vA+wyoL7nWSii4yfYtW3EZ+3gq6HxvnT9Zs8MC53uTOiOdamASXecYREwGmY/io
6n5XTEA/35LNbl4A756vSTZ7h4VFJAN5BpuqxstI1D7ou94skoSmcPoC/iniTvY9
kZylU5OCH/nifMAHM2a5jrQel80cW4oko9eyc8ENQpSy15JElFOKFf7D/4tcZJLF
F0VBTXbhfvq3dPfoq94IWt7p54Ovwj0S//CEu3jZYbNl2QC/3YiHE2H2XyGCQbq6
2MjcuxLnEapoRIqfbi8GPtCWVPzm28WGyKIDofWICczzeJFFJnvzrY3wRG64ibKJ
bR/uedwua1UuiC482V1FD5ffmzSSs8ktTp9hgj7RGDXlc9NTcF1jHxG9hwARAQAB
tCxBV1MgRGVhZGxpbmUgQ2xvdWQgPGF3cy1kZWFkbGluZUBhbWF6b24uY29tPokC
VwQTAQgAQRYhBJmXd7So2csyehiIYsg71N18bhtjBQJpQDQ1AhsvBQkDwmcABQsJ
CAcCAiICBhUKCQgLAgQWAgMBAh4HAheAAAoJEMg71N18bhtjk2UP/3h4KlEzZO/7
BxRmkbixuo1QuqOGvA6tXbSWaM8QH5jglcvL12PZLALklLT4v82uCsLR1lF8/Tch
cCl0SZEOFIS+XxAaw1Xfai6jlyLhabOwKF2ylq5eJlLcw1lh2nAArDRb4fLD0m1g
Dfqetq/XEpyXpOSkWxGRV4RlUdjQfytxrmcUnsT5/fk5f9VDdblu6K/lEmwfyYjB
lXv0uUCkqPot0SmbvOh3PY3Hi3n54ncy8NfTeV+TUvSe3C1s1zNl8aqHoTxJB/eU
kp+LFZ9m+igpSYnKeglKnytylH3KGCjTHglT/QXnI1wNTqmj1kFBVwtt/y1mtnA+
CPIUHP1CtbKsHaLtpp4llBm5TVtPN/Wqqicn5QLl4khg7R4K+V2aaA4ubY6p1tG9
0fFhN5tTnHDSKWMfmb83wfh5Zkcg85c3egjoit+wgGQRAQVqbznx7NqAHs9VoDIu
SPcAr+C329AOBzod4gyNGH7Ah5DkMITo4O4+axnAU9yhFOHcMJmTIask/fNg1Aum
OqYPMUwcgv1GZjLaTJyfGGC1xALsYR0KHnwIehD06MHR/Z98bGkcV8+Y0q8UPsd1
VN1fc1rjCJh/AT3w6owvG4DaEwspseSjzHv16mW4e2N6Uu23SPzgQsJ5qYN2g8D+
P7N9LGDfP8DaYc5JM9mlyFmYI2Q94ufL
=rY5l
-----END PGP PUBLIC KEY BLOCK-----
EOF
```

2. 決定是否信任0penPGP金鑰。決定是否信任上述金鑰時需要考慮的一些因素包括：

   - 您用來從此網站取得 GPG 金鑰的網際網路連線是安全的。
   - 您在 上存取此網站的裝置是安全的。
   - AWS 已採取措施來保護此網站上的0penPGP公有金鑰託管。

3. 如果您決定信任OpenPGP金鑰，請編輯金鑰以信任 ，gpg類似下列範例：

```
$ gpg --edit-key 0xB840C08C29A90796A071FAA5F6CD3CE6B76F3CEF

    gpg (GnuPG) 2.0.22; Copyright (C) 2013 Free Software Foundation, Inc.
    This is free software: you are free to change and redistribute it.
    There is NO WARRANTY, to the extent permitted by law.


    pub  4096R/4BF0B8D2  created: 2023-06-23  expires: 2025-06-22  usage: SCEA
                         trust: unknown       validity: unknown
    [ unknown] (1). AWS Deadline Cloud example@example.com
```

```
   gpg> trust
   pub  4096R/4BF0B8D2  created: 2023-06-23  expires: 2025-06-22  usage: SCEA
                        trust: unknown       validity: unknown
   [ unknown] (1). AWS Deadline Cloud aws-deadline@amazon.com

   Please decide how far you trust this user to correctly verify other users'
keys
   (by looking at passports, checking fingerprints from different sources,
etc.)

     1 = I don't know or won't say
     2 = I do NOT trust
     3 = I trust marginally
     4 = I trust fully
     5 = I trust ultimately
     m = back to the main menu

   Your decision? 5
   Do you really want to set this key to ultimate trust? (y/N) y

   pub  4096R/4BF0B8D2  created: 2023-06-23  expires: 2025-06-22  usage: SCEA
                        trust: ultimate      validity: unknown
   [ unknown] (1). AWS Deadline Cloud aws-deadline@amazon.com
   Please note that the shown key validity is not necessarily correct
   unless you restart the program.

   gpg> quit
```

4. 驗證截止日期雲端提交者安裝程式

   若要驗證截止日期雲端提交者安裝程式，請完成下列步驟：

   a. 下載截止日期雲端提交者安裝程式的簽章檔案。

      下載簽章檔案 (.sig)

   b. 執行下列動作來驗證截止日期雲端提交者安裝程式的簽章：

      ```
      gpg --verify ./DeadlineCloudSubmitter-linux-x64-installer.run.sig ./
      DeadlineCloudSubmitter-linux-x64-installer.run
      ```

5. 驗證截止日期雲端監視器

> **Note**
>
> 您可以使用簽章檔案或平台特定方法，驗證截止日期雲端監視器下載。如需平台特定的方法，請參閱 Linux (Debian)標籤、 Linux(RPM) 標籤，或根據您下載的檔案類型的 Linux (AppImage) 標籤。

若要使用簽章檔案驗證截止日期雲端監控桌面應用程式，請完成下列步驟：

a. 為您的截止日期雲端監視器安裝程式下載對應的簽章檔案：

- 下載 .deb 簽章檔案
- 下載 .rpm 簽章檔案
- 下載 .AppImage 簽章檔案

b. 驗證簽章：

對於 .deb：

```
gpg --verify ./deadline-cloud-monitor_amd64.deb.sig ./deadline-cloud-
monitor_amd64.deb
```

對於 .rpm：

```
gpg --verify ./deadline-cloud-monitor.x86_64.rpm.sig ./deadline-cloud-
monitor.x86_64.rpm
```

對於 .AppImage：

```
gpg --verify ./deadline-cloud-monitor_amd64.AppImage.sig ./deadline-cloud-
monitor_amd64.AppImage
```

c. 確認輸出看起來類似以下內容：

```
gpg: Signature made Mon Apr 1 21:10:14 2024 UTC

gpg: using RSA key B840C08C29A90796A071FAA5F6CD3CE6B7
```

如果輸出包含片語 Good signature from "AWS Deadline Cloud"，表示簽章已成功驗證，您可以執行截止日期雲端監視器安裝指令碼。

歷史金鑰

```
-----BEGIN PGP PUBLIC KEY BLOCK-----

mQINBGX6GQsBEADduUtJgqSXI+q76O6fsFwEYKmbnlyL0xKvlq32EZuyv0otZo5L
le4m5Gg52AzrvPvDiUTLooAlvYeozaYyirIGsK08Ydz0Ftdjroiuh/mw9JSJDJRI
rnRn5yKet1JFezkjopA3pjsTBP6lW/mb1bDBDEwwwtH0x9lV7A03FJ9T7Uzu/qSh
qO/UYdkafro3cPASvkqgDt2tCvURfBcUCAjZVFcLZcVD5iwXacxvKsxxS/e7kuVV
I1+VGT8Hj8XzWYhjCZxOLZk/fvpYPMyEEujN0fYUp6RtMIXve0C9awwMCy5nBG2J
eE2Ol5DsCpTaBd4Fdr3LWcSs8JFA/YfP9auL3NczOozPoVJt+fw8CBlVIXO0J7l5
hvHDjcC+5v0wxqAlMG6+f/SX7CT8FXK+L3iOJ5gBYUNXqHSxUdv8kt76/KVmQa1B
Akl+MPKpMq+lhw++S3G/lXqwWaDNQbRRw7dSZHymQVXvPp1nsqc3hV7KlOM+6s6g
1g4mvFY4lf6DhptwZLWyQXU8rBQpojvQfiSmDFrFPWFi5BexesuVnkGIolQoklKx
AVUSdJPVEJCteyy7td4FPhBaSqT5vW3+ANbr9b/uoRYWJvn17dN0cc9HuRh/Ai+I
nkfECo2WUDLZ0fEKGjGyFX+todWvJXjvc5kmE9Ty5vJp+M9Vvb8jd6t+mwARAQAB
tCxBV1MgRGVhZGxpbmUgQ2xvdWQgPGF3cy1kZWFkbGluZUBhbWF6b24uY29tPokC
VwQTAQgAQRYhBLhAwIwpqQeWoHH6pfbNPOa3bzzvBQJl+hkLAxsvBAUJA8JnAAUL
CQgHAgIiAgYVCgkICwIDFgIBAh4HAheAAAoJEPbNPOa3bzzvKswQAJXzKSAY8sY8
F6Eas2oYwIDDdDurs8FiEnFghjUEO6MTt9AykF/jw+CQg2UzFtEyObHBymhgmhXE
3buVeom96tgM3ZDfZu+sxi5pGX6oAQnZ6riztN+VpkpQmLgwtMGpSMLl3KLwnv2k
WK8mrR/fPMkfdaewB7A6RIUYiW33GAL4KfMIs8/vIwIJw99NxHpZQVoU6dFpuDtE
1OuxGcCqGJ7mAmo6H/YawSNp2Ns80gyqIKYo7o3LJ+WRroIRlQyctq8gnR9JvYXX
42ASqLq5+OXKo4qh81blXKYqtc176BbbSNFjWnzIQgKDgNiHFZCdcOVgqDhwO15r
NICbqqwwNLj/Fr2kecYx180Ktpl0jOOw5IOyh3bf3MVGWnYRdjvA1v+/CO+55N4g
z0kf50Lcdu5RtqV10XBCifn28pecqPaSdYcssYSRl5DLiFktGbNzTGcZZwITTKQc
af8PPdTGtnnb6P+cdbW3bt9MVtN5/dgSHLThnS8MPEuNCtkTnpXshuVuBGgwBMdb
qUC+HjqvhZzbwns8dr5WI+6HWNBFgGANn6ageYl58vVp0UkuNP8wcWjRARciHXZx
ku6W2jPTHDWGNrBQO2Fx7fd2QYJheIPPAShHcfJO+xgWCof45D0vAxAJ8gGg9Eq+
gFWhsx4NSHn2gh1gDZ41Ou/4exJ1lwPM
=uVaX
-----END PGP PUBLIC KEY BLOCK-----
EOF
```

Linux (AppImage)

若要驗證使用 Linux .AppImage 二進位檔的套件，請先完成Linux索引標籤中的步驟 1-3，然後完成下列步驟。

1. 從 GitHub 的 AppImageUpdate 頁面，下載 validate-x86_64.AppImage 檔案。
2. 下載檔案後，若要新增執行許可，請執行下列命令。

```
chmod a+x ./validate-x86_64.AppImage
```

3. 若要新增執行許可，請執行下列命令。

```
chmod a+x ./deadline-cloud-monitor_<APP_VERSION>_amd64.AppImage
```

4. 若要驗證截止日期雲端監視器簽章，請執行下列命令。

```
./validate-x86_64.AppImage ./deadline-cloud-monitor_<APP_VERSION>_amd64.AppImage
```

如果輸出包含片語 `Validation successful`，表示簽章已成功驗證，您可以安全地執行截止日期雲端監視器安裝指令碼。

Linux (Debian)

若要驗證使用 Linux .deb 二進位檔的套件，請先完成 Linux標籤中的步驟 1-3。

dpkg 是大多數debian以 為基礎的Linux分佈中的核心套件管理工具。您可以使用 工具驗證 .deb 檔案。

1. 下載截止日期雲端監視器 .deb 檔案：

   下載截止日期雲端監視器 (.deb)
2. 驗證 .deb 檔案：

```
dpkg-sig --verify deadline-cloud-monitor_amd64.deb
```

3. 輸出將類似於：

```
Processing deadline-cloud-monitor_amd64.deb...
GOODSIG _gpgbuilder B840C08C29A90796A071FAA5F6CD3C 171200
```

4. 若要驗證 .deb 檔案，請確認 GOODSIG存在於輸出中。

Linux (RPM)

若要驗證使用 Linux .rpm 二進位檔的套件，請先完成Linux標籤中的步驟 1-3。

1. 下載截止日期雲端監視器 .rpm 檔案：

   下載截止日期雲端監視器 (.rpm)

2. 驗證 .rpm 檔案：

```
gpg --export --armor "Deadline Cloud" > key.pub
sudo rpm --import key.pub
rpm -K deadline-cloud-monitor.x86_64.rpm
```

3. 輸出將類似於：

```
deadline-cloud-monitor.x86_64.rpm: digests signatures OK
```

4. 若要驗證 .rpm 檔案，請確認 `digests signatures OK` 位於輸出中。

文件歷史記錄

如需有關AWS截止日期雲端更新的資訊，請參閱截止日期雲端版本備註。

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。