



使用者指南

Amazon ECR



API 版本 2015-09-21

Copyright © 2026 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon ECR: 使用者指南

Copyright © 2026 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon 的商標和商業外觀不得用於任何非 Amazon 的產品或服務，也不能以任何可能造成客戶混淆、任何貶低或使 Amazon 名譽受損的方式使用 Amazon 的商標和商業外觀。所有其他非 Amazon 擁有的商標均為其各自擁有者的財產，這些擁有者可能附屬於 Amazon，或與 Amazon 有合作關係，亦或受到 Amazon 贊助。

Table of Contents

什麼是 Amazon ECR	1
概念和元件	1
常用案例	3
Amazon ECR 的功能	4
如何開始使用 Amazon ECR	5
Amazon ECR 定價	5
在影像的生命週期中移動影像	6
先決條件	6
安裝 AWS CLI	6
安裝 Docker	6
步驟 1：建立 Docker 映像	7
步驟 2：建立儲存庫	10
步驟 3：驗證您的預設登錄檔	10
步驟 4：將映像推送至 Amazon ECR	11
步驟 5：從 Amazon ECR 提取映像	12
步驟 6：刪除映像	13
步驟 7：刪除儲存庫	13
最佳化效能	14
提出要求	16
IPv6 入門	16
測試 IP 地址相容性	17
使用雙重堆疊端點提出請求	17
從 docker CLI 使用 Amazon ECR 端點	18
在 IAM 原則中使用 IPv6 地址	18
私有登錄檔	20
登錄檔概念	20
登錄檔身分驗證	20
使用 Amazon ECR 憑證協助程式	21
使用授權字符	21
使用 HTTP API 身分驗證	22
登錄檔設定	22
Blob 掛載	23
Blob 掛載概念	23
Blob 掛載組態	24

登錄檔許可	24
登錄檔政策範例	25
授予跨帳戶複寫的許可	28
授予提取快取的許可	30
私有儲存庫	31
儲存庫概念	31
建立儲存庫以存放映像	32
後續步驟	34
檢視儲存庫詳細資訊	34
刪除儲存庫	35
儲存庫政策	35
儲存庫政策與 IAM 政策的比較	36
儲存庫政策範例	37
設定儲存庫政策陳述式	43
標記儲存庫	44
標籤基本概念	44
標記您的資源以便計費	45
新增 標籤	45
刪除標籤	46
私有映像	48
推送映像	48
所需的 IAM 許可	49
推送 Docker 映像	50
推送多架構映像	52
推送 Helm Chart	54
刪除成品	56
檢視映像詳細資訊	58
提取映像	59
提取 Amazon Linux 容器映像	61
刪除映像	62
封存映像	63
什麼是 ECR 封存儲存類別？	63
封存映像	64
還原映像	66
重新標記映像	67
防止覆寫映像標籤	70

設定影像標籤可變性 (AWS 管理主控台)	70
設定影像標籤可變性 (AWS CLI)	71
容器映像資訊清單格式	72
Amazon ECR 映像資訊清單轉換	72
將 Amazon ECR 映像與 Amazon ECS 搭配使用	73
所需的 IAM 許可	74
在任務定義中指定 Amazon ECR 映像	75
將 Amazon ECR 映像與 Amazon EKS 搭配使用	76
所需的 IAM 許可	76
在 Amazon EKS 叢集上安裝 Helm Chart	77
簽署影像	79
選擇簽署方法	79
考量事項	79
受管簽署	79
先決條件	80
開始使用	81
考量事項	83
簽章驗證	84
使用 Amazon EKS 進行受管驗證	84
Amazon ECS 的 Lambda 許可控制器	84
使用標記法 CLI 手動驗證	84
設定 Notation 用戶端的身分驗證	84
手動簽署	85
先決條件	85
掃描映像是否有漏洞	86
儲存庫的篩選條件	87
篩選萬用字元	87
增強型掃描	87
增強型掃描的注意事項	88
變更增強型掃描持續時間	89
所需的 IAM 許可	89
設定增強型掃描	90
EventBridge 事件	92
擷取問題清單	97
基本型掃描	98
作業系統支援基本掃描	98

設定基本掃描	99
手動掃描映像	99
擷取問題清單	101
對映像掃描進行故障診斷	102
了解掃描狀態 SCAN_ELIGIBILITY_EXPIRED	103
同步上游登錄檔	104
儲存庫建立範本	104
使用提取快取規則的考量事項	105
所需的 IAM 許可	106
使用登錄檔許可	107
後續步驟	108
設定跨帳戶 ECR 到 ECR PTC 的許可	109
跨帳戶 ECR 到 ECR 提取快取所需的 IAM 政策	109
建立提取快取規則	111
先決條件	111
使用 AWS 管理主控台	112
使用 AWS CLI	119
後續步驟	122
驗證提取快取規則	123
使用提取快取規則提取映像	124
儲存您的上游儲存庫憑證	125
自訂儲存庫字首	133
疑難排解提取快取問題	134
複寫映像	136
複寫政策需求	136
政策組態概觀	136
目的地登錄檔政策需求	136
來源帳戶需求	137
常見的誤解	138
對複寫失敗進行故障診斷	138
私有映像複寫的考量	138
複寫範例	140
範例：將跨區域複寫設定為單一目的地區域	140
範例：使用儲存庫篩選條件設定跨區域複寫	140
範例：設定跨區域複寫至多個目的地區域	141
範例：設定跨帳戶複寫	141

範例：指定組態中的多個規則	142
範例：移除所有複寫設定	143
設定複寫	143
移除複寫	145
儲存庫建立範本	147
運作方式	147
建立儲存庫建立範本	150
建立儲存庫建立範本的 IAM 許可	150
建立自訂政策	151
建立 IAM 角色	152
建立儲存庫建立範本	153
更新儲存庫建立範本	157
刪除儲存庫建立範本	158
自動化映像的清除	160
生命週期政策如何運作	160
生命週期政策評估規則	161
建立生命週期政策預覽	162
建立生命週期政策	164
先決條件	164
生命週期政策範例	165
生命週期政策範本	166
篩選映像存在時間	166
篩選上次提取的時間	167
篩選封存轉換時間	168
篩選映像計數	168
篩選多個規則	169
篩選單一規則中的多個標籤	171
篩選所有映像	173
存檔範例	176
生命週期政策屬性	178
規則優先順序	179
Description	179
標籤狀態	179
標籤模式清單	180
標籤字首清單	180
儲存類別	180

計數類型	181
計數單位	181
Count (計數)	181
Action	182
提取時間更新排除	183
管理提取時間更新排除	183
提取時間更新排除的考量事項	186
安全	187
身分和存取權管理	187
目標對象	188
使用身分驗證	188
使用政策管理存取權	189
Amazon Elastic Container Registry 如何與 IAM 搭配使用	190
身分型政策範例	194
使用標籤型存取控制	198
AWS Amazon ECR 的 受管政策	200
使用服務連結角色	205
疑難排解	212
資料保護	214
靜態加密	215
法規遵循驗證	222
基礎設施安全性	222
介面 VPC 端點 (AWS PrivateLink)	222
預防跨服務混淆代理人	230
監控	233
視覺化您的 Service Quotas 和設定警報	234
用量指標	235
用量報告	237
儲存庫指標	237
啟用 CloudWatch 指標	237
可用的指標與維度	237
使用 CloudWatch 檢視指標	238
事件和 EventBridge	238
來自 Amazon ECR 的範例事件	239
使用 記錄 AWS CloudTrail 動作	245
CloudTrail 中的 Amazon ECR 資訊	246

了解 Amazon ECR 日誌檔案項目	247
使用 AWS SDKs	264
程式碼範例	265
基本概念	266
Hello Amazon ECR	266
了解基本概念	270
動作	326
案例	382
Amazon ECR 入門	382
Service Quotas	391
在中管理您的 Amazon ECR 服務配額 AWS 管理主控台	395
建立 CloudWatch 警示以監控 API 用量指標	395
疑難排解	396
Docker 疑難排解	396
Docker 日誌不包含預期的錯誤訊息	396
當從 Amazon ECR 儲存庫提取映像時，出現錯誤：「Filesystem Verification Failed」(檔案系統驗證失敗) 或「404: Image Not Found」(404：找不到映像)	396
當從 Amazon ECR 提取映像時出現錯誤：「Filesystem Layer Verification Failed」(檔案系統分層驗證失敗)	397
當推送至儲存庫時，出現 HTTP 403 錯誤或「無基本身分驗證憑證」錯誤	398
Amazon ECR 錯誤訊息故障診斷	398
HTTP 429: Too Many Requests or ThrottleException (過多請求或 ThrottleException)	399
HTTP 403: "User [arn] is not authorized to perform [operation]" (使用者 [arn] 未授權您執行此 [操作])	399
HTTP 404：「儲存庫不存在」錯誤	399
錯誤：無法從非 TTY 裝置執行互動式登入	400
搭配 Amazon ECR 使用 Podman	401
使用 Podman 向 Amazon ECR 驗證	401
搭配 Podman 使用 Amazon ECR 登入資料協助程式	401
使用 Podman 從 Amazon ECR 提取映像	402
使用執行 Amazon ECR 的容器 Podman	402
使用將映像推送至 Amazon ECR Podman	402
文件歷史紀錄	403
.....	cdix

什麼是 Amazon Elastic Container Registry ?

Amazon Elastic Container Registry (Amazon ECR) 是一種安全、可擴展且可靠的 AWS 受管容器映像登錄服務。Amazon ECR 使用 IAM 支援具有資源型許可 AWS 的私有儲存庫。如此可讓指定的使用者或 Amazon EC2 執行個體存取您的容器儲存庫及映像。您可以使用偏好的 CLI 來推送、提取和管理 Docker 映像、Open Container Initiative (OCI) 映像以及與 OCI 相容的成品。

Note

Amazon ECR 也支援公有容器映像存放庫。如需詳細資訊，請參閱《Amazon ECR Public 使用者指南》中的[什麼是 Amazon ECR Public](#)。

AWS 容器服務團隊會在 GitHub 上維護公有藍圖。它包含有關團隊正在進行哪些工作的資訊，並允許所有 AWS 客戶提供直接意見回饋。如需詳細資訊，請參閱[AWS 容器藍圖](#)。

Amazon ECR 的概念和元件

Amazon ECR 是由提供的全受管 Docker 容器登錄服務 AWS。它可讓您安全地可靠地存放、管理和部署 Docker 容器映像。這些概念和元件共同運作，在中提供安全、可擴展且可靠的 Docker 容器登錄服務 AWS，讓您能夠有效率地管理和部署容器化應用程式。

以下是 Amazon ECR 的一些重要概念和元件：

登錄

Amazon ECR 登錄檔是提供給每個 AWS 帳戶的私有儲存庫，您可以在其中建立一或多個儲存庫。這些儲存庫可讓您在 AWS 環境中存放和分發 Docker 映像、Open Container Initiative (OCI) 映像和其他與 OCI 相容的成品。如需詳細資訊，請參閱[Amazon ECR 私有登錄檔](#)。

授權字符

您的用戶端在能夠推送與提取映像前，需要驗證至 Amazon ECR 私有登錄檔以做為 AWS 使用者。如需詳細資訊，請參閱[Amazon ECR 中的私有登錄檔身分驗證](#)。

儲存庫

Amazon ECR 中的儲存庫是邏輯集合，您可以在其中存放 Docker 映像、Open Container Initiative (OCI) 映像和其他與 OCI 相容的成品。在單一 Amazon ECR 登錄檔中，您可以有多個儲存庫來組織容器映像。如需詳細資訊，請參閱[Amazon ECR 私有儲存庫](#)。

儲存庫政策

您可透過儲存庫政策來控制您儲存庫的存取以及其中的內容。如需詳細資訊，請參閱[Amazon ECR 中的私有儲存庫政策](#)。

映像

您可以推送與提取容器映像至您的儲存庫。您可在開發系統上以本機使用這些映像，或者您也可以可以在 Amazon ECS 任務定義和 Amazon EKS Pod 規格。如需詳細資訊，請參閱[將 Amazon ECR 映像與 Amazon ECS 搭配使用](#)及[將 Amazon ECR 映像與 Amazon EKS 搭配使用](#)。

生命週期政策

Amazon ECR 生命週期政策可讓您透過定義刪除和過期舊或未使用的映像的規則來管理映像的生命週期。如需詳細資訊，請參閱[在 Amazon ECR 中使用生命週期政策來自動化映像的清除](#)。

影像掃描

Amazon ECR 提供整合式映像掃描功能，可協助識別容器映像中的軟體漏洞。如需詳細資訊，請參閱[掃描映像是否有 Amazon ECR 中的軟體漏洞](#)。

存取控制

Amazon ECR 使用 IAM 來控制對儲存庫的存取。您可以建立具有特定許可的 IAM 使用者、群組和角色，以推送、提取或管理 Amazon ECR 儲存庫。如需詳細資訊，請參閱[Amazon Elastic Container Registry 的安全性](#)。

跨帳戶和跨區域複寫

Amazon ECR 支援跨多個 AWS 帳戶和區域複寫映像，以提高可用性並減少延遲。如需詳細資訊，請參閱[Amazon ECR 中的私有映像複寫](#)。

加密

Amazon ECR 支援使用對靜態 Docker 映像進行伺服器端加密 AWS KMS。如需詳細資訊，請參閱[Amazon ECR 中的資料保護](#)。

AWS Command Line Interface 整合

AWS CLI 提供命令來與 Amazon ECR 儲存庫互動，例如建立、列出、推送和提取映像。

AWS 管理主控台

Amazon ECR 也可以透過 進行管理 AWS 管理主控台，提供易於使用的 Web 界面，以使用儲存庫和映像。

AWS CloudTrail

Amazon ECR 與 整合 AWS CloudTrail，可讓您記錄和稽核對 Amazon ECR 發出的 API 呼叫，以用於安全和合規目的。如需詳細資訊，請參閱[使用 記錄 Amazon ECR 動作 AWS CloudTrail](#)。

Amazon CloudWatch

Amazon ECR 提供可使用 監控的指標和日誌 Amazon CloudWatch，可讓您追蹤 Amazon ECR 儲存庫的效能和用量。如需詳細資訊，請參閱[Amazon ECR 儲存庫指標](#)。

受管簽署

受管簽署會在將映像推送至 Amazon ECR 時自動產生密碼編譯簽章，簡化容器映像簽署。如需詳細資訊，請參閱[受管簽署](#)。

Amazon ECR 中的常見使用案例

Amazon ECR 是由 提供的全受管 Docker 容器登錄服務 AWS。它提供安全且可擴展的儲存庫，用於存放和分發 Docker 容器映像，使其成為容器化應用程式部署中的重要元件。Amazon ECR 可簡化在各種 AWS 服務和內部部署環境中建置、分發和執行容器化應用程式的程序。

以下是 Amazon ECR 的一些主要使用案例：

容器映像儲存和分發

Amazon ECR 做為集中式儲存庫，用於在組織內存放和分發 Docker 容器映像，或供大眾使用。開發人員可以將容器映像推送到 Amazon ECR，然後從 Amazon EC2 AWS Fargate 或 Amazon EKS AWS 等內部的任何運算環境中提取它們。如需詳細資訊，請參閱[Amazon ECR 私有儲存庫](#)。

持續整合和持續部署 (CI/CD)

Amazon ECR 與 AWS CodeBuild AWS CodePipeline、和其他 CI/CD 工具無縫整合，實現容器化應用程式的自動化建置、測試和部署。容器映像可以作為 CI/CD 管道的一部分自動推送到 Amazon ECR，確保在不同環境中進行一致且可靠的部署。

微服務架構

Amazon ECR 非常適合微服務架構，其中應用程式分為封裝為容器的小型解耦服務。每個微服務都可以在 Amazon ECR 中存放自己的容器映像，以便獨立開發、部署和擴展個別服務。

混合和多雲端部署

Amazon ECR 支援從其他容器登錄檔提取容器映像，例如 Docker Hub 或第三方登錄檔。這可讓組織使用 Amazon ECR 作為容器映像的中央儲存庫，在混合或多雲端環境中維持一致的部署模型。

存取控制和安全性

Amazon ECR 提供精細存取控制機制，可讓組織控制誰可以從登錄檔推送或提取容器映像。它也會與整合 AWS Identity and Access Management 以進行身分驗證和授權，以確保安全存取容器映像。如需詳細資訊，請參閱[Amazon Elastic Container Registry 的安全性](#)。

映像漏洞掃描

Amazon ECR 提供自動掃描容器映像是否有軟體漏洞和潛在的錯誤組態，協助維護安全且合規的容器環境。如需詳細資訊，請參閱[掃描映像是否有 Amazon ECR 中的軟體漏洞](#)。

私有容器登錄檔

對於具有嚴格安全或合規要求的組織，Amazon ECR 可以用作私有容器登錄檔，確保敏感容器映像不會公開到公有登錄檔，並且只能在組織 AWS 環境中存取。如需詳細資訊，請參閱[Amazon ECR 私有登錄檔](#)。

使用 Amazon ECR 複寫進行全域分散式應用程式部署

利用 Amazon ECR 複寫功能，您可以將容器化 Web 應用程式映像集中在主要儲存庫中，實現跨多個 AWS 區域的自動化分佈，確保全球部署一致且低延遲，並減少營運負擔。如需詳細資訊，請參閱[Amazon ECR 中的私有映像複寫](#)

自動清除過時的容器映像

Amazon ECR 生命週期政策可根據定義規則自動清除過時的容器映像，例如存留期、計數或標籤、最佳化儲存成本、維護有組織的登錄檔、增強安全性和合規性，以及透過自動化簡化開發工作流程。如需詳細資訊，請參閱[在 Amazon ECR 中使用生命週期政策來自動化映像的清除](#)

Amazon ECR 的功能

Amazon ECR 提供以下功能：

- 生命週期政策有助於管理儲存庫中映像的生命週期。您定義會導致清理未使用映像的規則。您可以在將規則套用至儲存庫前先進行測試。如需詳細資訊，請參閱[在 Amazon ECR 中使用生命週期政策來自動化映像的清除](#)。
- 映像掃描有助於識別容器映像中的軟體漏洞。每個儲存庫都可以設定為在推送時掃描。這可確保會對每個推送到儲存庫的新映像進行掃描。之後，您可以擷取映像掃描的結果。如需詳細資訊，請參閱[掃描映像是否有 Amazon ECR 中的軟體漏洞](#)。
- 跨區域和跨帳戶複寫可讓您更輕鬆地將映像放置在所需的地方。這會設定為登錄設定，並且以每一區域為基礎。如需詳細資訊，請參閱[Amazon ECR 中的私有登錄檔設定](#)。

- 提取快取規則提供了一個方式，可在私有 Amazon ECR 登錄檔中快取上游登錄檔中的儲存庫。使用提取快取規則，Amazon ECR 將定期聯繫上游登錄檔，以確保 Amazon ECR 私有登錄檔中的快取映像是最新的。如需詳細資訊，請參閱[將上游登錄檔與 Amazon ECR 私有登錄檔同步](#)。
- 儲存庫建立範本可讓您在提取快取、推送時建立或複寫動作期間，定義 Amazon ECR 代表您建立的儲存庫設定。您可以為自動建立的儲存庫指定標籤不可變性、加密組態、儲存庫政策、生命週期政策和資源標籤。如需詳細資訊，請參閱[用於控制提取快取、推送時建立或複寫動作期間建立的儲存庫的範本](#)。
- 受管簽署會在將映像推送至 Amazon ECR 時自動產生密碼編譯簽章，簡化容器映像簽署。如需詳細資訊，請參閱[受管簽署](#)。

如何開始使用 Amazon ECR

如果您使用的是 Amazon Elastic Container Service (Amazon ECS) 或 Amazon Elastic Kubernetes Service (Amazon EKS)，請注意，這兩個服務的設定類似於 Amazon ECR 的設定，因為 Amazon ECR 是這兩個服務的延伸。

AWS Command Line Interface 搭配 Amazon ECR 使用時，請使用 AWS CLI 支援最新 Amazon ECR 功能的版本。如果您在中沒有看到 Amazon ECR 功能的支援 AWS CLI，請升級到最新版本的 AWS CLI。如需有關安裝最新版本的資訊 AWS CLI，請參閱AWS Command Line Interface 《使用者指南》中的[安裝或更新至最新版本的 AWS CLI](#)。

若要了解如何使用和 Docker 將 AWS CLI 容器映像推送至私有 Amazon ECR 儲存庫，請參閱[在 Amazon ECR 中移動映像的生命週期](#)。

Amazon ECR 定價

使用 Amazon ECR，您需要支付存放在儲存庫中的資料量、從您的影像推送和提取的資料傳輸，以及您選擇加入的影像動作，例如影像簽署和複寫。如需詳細資訊，請參閱[Amazon ECR 定價](#)。

在 Amazon ECR 中移動映像的生命週期

如果您是第一次使用 Amazon ECR，請使用下列步驟搭配 Docker CLI 和 AWS CLI 來建立範例映像、向預設登錄檔進行身分驗證，以及建立私有儲存庫。然後將映像推送至，並從私有儲存庫提取映像。完成範例映像後，請刪除範例映像和儲存庫。

若要使用 AWS 管理主控台 而非 AWS CLI，請參閱 [the section called “建立儲存庫以存放映像”](#)。

如需可用於管理 AWS 資源的其他工具的詳細資訊，包括 AWS SDKs、IDE 工具組和 Windows PowerShell 命令列工具，請參閱 <https://http://aws.amazon.com/tools/>。

先決條件

如果您尚未安裝最新的 AWS CLI 和 Docker 並準備好使用，請使用下列步驟來安裝這兩個工具。

安裝 AWS CLI

若要 AWS CLI 搭配 Amazon ECR 使用，請安裝 AWS CLI 最新版本。如需詳細資訊，請參閱《AWS Command Line Interface 使用者指南》中的 [安裝 AWS Command Line Interface](#)。

安裝 Docker

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

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

如果您已經安裝 Docker，請跳到「[步驟 1：建立 Docker 映像](#)」。

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

1. 使用最新版 Amazon Linux 2023 AMI 啟動執行個體。如需詳細資訊，請參閱《Amazon EC2 使用者指南》中的 [啟動執行個體](#)。
2. 連線到您的執行個體。如需詳細資訊，請參閱《Amazon EC2 使用者指南》中的 [連線至您的 Linux 執行個體](#)。
3. 更新已安裝的套裝服務，並在執行個體上封裝快取。

```
sudo yum update -y
```

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

```
sudo yum install docker
```

5. 啟動 Docker 服務。

```
sudo service docker start
```

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

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

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

```
docker info
```

Note

在某些情況下，您可能需要重新啟動執行個體，才能提供 `ec2-user` 存取 Docker 常駐程式的許可。如果您看到下列錯誤，請嘗試重新啟動執行個體：

```
Cannot connect to the Docker daemon. Is the docker daemon running on this host?
```

步驟 1：建立 Docker 映像

在此步驟中，您將建立簡易 Web 應用程式的 Docker 映像檔，並在本機系統或 Amazon EC2 執行個體上進行測試。

建立簡單 Web 應用程式的 Docker 映像

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

touch Dockerfile

2. 編輯您剛建立的 Dockerfile，並新增下列內容。

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

# Install dependencies
RUN yum update -y && \
    yum install -y httpd

# Install apache and write hello world message
RUN echo 'Hello World!' > /var/www/html/index.html

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

EXPOSE 80

CMD /root/run_apache.sh
```

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

3. 從 Dockerfile 建置 Docker 映像。

Note

在下列命令中，有些 Docker 版本可能需要 Dockerfile 的完整路徑，而不是下面所示的相對路徑。

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

- 列出您的容器映像。

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

輸出：

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

- 執行新建置的映像。-p 80:80 選項會將容器上的公開連接埠 80 映射至主機系統上的連接埠 80。如需 docker run 的詳細資訊，請前往「[Docker run 參考](#)」。

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

Note

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

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

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

您應該會看到網頁，內含您的 "Hello World!" 陳述式。

7. 輸入 Ctrl + c，以停止 Docker 容器。

步驟 2：建立儲存庫

現在您已擁有可推送至 Amazon ECR 的映像，您必須建立儲存庫以存放它。在此範例中，您建立一個稱為 `hello-repository` 的儲存庫，以在稍後推送 `hello-world:latest` 映像。若要建立一個儲存庫，請執行下列命令：

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

步驟 3：驗證您的預設登錄檔

安裝並設定之後 AWS CLI，請將 Docker CLI 驗證為您的預設登錄檔。docker 命令可以透過該方法使用 Amazon ECR 來推送和提取映像。AWS CLI 提供 `get-login-password` 命令來簡化身分驗證程序。

Note

您的 IAM 主體必須具有向登錄檔進行身分驗證的 `ecr:GetAuthorizationToken` 許可。如需詳細資訊，請參閱 [AWS Amazon Elastic Container Registry 的受管政策](#)。

若要使用 `get-login-password` 向 Amazon ECR 登錄檔驗證 Docker，請執行 `aws ecr get-login-password` 命令。將身分驗證字串傳遞給 `docker login` 命令時，使用 AWS 的值作為使用者名稱並指定您要驗證的 Amazon ECR 登錄檔 URI。如果是向多個登錄進行驗證，您必須針對每個登錄重複此命令。

Important

若您收到錯誤，請安裝或升級至最新版本的 AWS CLI。如需詳細資訊，請參閱《AWS Command Line Interface 使用者指南》中的 [安裝 AWS Command Line Interface](#)。

- [get-login-password](#) (AWS CLI)

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

- [Get-ECRLoginCommand](#) (AWS Tools for Windows PowerShell)

```
(Get-ECRLoginCommand).Password | docker login --username AWS --password-stdin aws_account_id.dkr.ecr.region.amazonaws.com
```

步驟 4：將映像推送至 Amazon ECR

現在您可推送映像至您在前一節建立的 Amazon ECR 儲存庫。在符合下列先決條件之後，使用 docker CLI 推送映像：

- docker 已安裝的最低版本：1.7。
- Amazon ECR 授權字符已使用設定 docker login。
- Amazon ECR 儲存庫存在，且使用者可存取並推送至儲存庫。

在那些必要條件滿足後，您可推送映像至您帳戶預設登錄檔中新建立的儲存庫。

標記映像並推送映像至 Amazon ECR

1. 列出您在本機儲存的映像以識別欲標記與推送的映像。

```
docker images
```

輸出：

REPOSITORY	TAG	IMAGE ID	CREATED
VIRTUAL SIZE			
hello-world	latest	e9ffedc8c286	4 minutes ago
241MB			

2. 標記映像以推送至您的儲存庫。

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

3. 推送映像。

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

輸出：

```
The push refers to a repository [aws_account_id.dkr.ecr.region.amazonaws.com/hello-repository] (len: 1)
e9ae3c220b23: Pushed
a6785352b25c: Pushed
0998bf8fb9e9: Pushed
0a85502c06c9: Pushed
latest: digest: sha256:215d7e4121b30157d8839e81c4e0912606fca105775bb0636EXAMPLE
size: 6774
```

步驟 5：從 Amazon ECR 提取映像

將映像推送至 Amazon ECR 儲存庫後，您可以從其他位置提取映像。符合下列先決條件後，使用 docker CLI 提取映像：

- docker 已安裝的最低版本：1.7。
- Amazon ECR 授權字符已使用設定 docker login。
- Amazon ECR 儲存庫存在，且使用者擁有從儲存庫提取的存取權。

在滿足那些必要條件後，您可提取您的映像。若要從 Amazon ECR 提取您的範例映像，請執行下列命令：

```
docker pull aws_account_id.dkr.ecr.region.amazonaws.com/hello-repository:latest
```

輸出：

```
latest: Pulling from hello-repository
0a85502c06c9: Pull complete
0998bf8fb9e9: Pull complete
a6785352b25c: Pull complete
e9ae3c220b23: Pull complete
Digest: sha256:215d7e4121b30157d8839e81c4e0912606fca105775bb0636EXAMPLE
Status: Downloaded newer image for aws_account_id.dkr.ecr.region.amazonaws.com/hello-repository:latest
```

步驟 6：刪除映像

如果您不再需要其中一個儲存庫中的映像，則可以刪除映像。若要刪除映像，請指定其所在的儲存庫，以及映像的 `imageTag` 或 `imageDigest` 值。下列範例會刪除儲存 `hello-repository` 庫中具有映像標籤的映像 `latest`。若要從儲存庫刪除您的範例映像，請執行下列命令：

```
aws ecr batch-delete-image \  
  --repository-name hello-repository \  
  --image-ids imageTag=latest \  
  --region region
```

步驟 7：刪除儲存庫

如果您不再需要整個映像儲存庫，您可以刪除儲存庫。下列範例使用 `--force` 旗標來刪除包含映像的儲存庫。若要刪除包含映像的儲存庫 (以及其中所有的映像)，您可執行以下命令：

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

最佳化 Amazon ECR 的效能

您可以使用下列有關設定和策略的建議，來最佳化使用 Amazon ECR 時的效能。

使用 Docker 1.10 與以上版本以取得同時分層上傳優勢

Docker 映像是由各層組成，亦是映像的中介建立階段。Dockerfile 中的每一行都會導致新層的建立。當您使用 Docker 1.10 或以上版本，Docker 預設會在同時上傳 Amazon ECR 時盡量推送越多的層，因此上傳時間會更快。

使用較小的基礎映像

整個 Docker Hub 可用的預設映像可能會包括許多您的應用程式不需要的依存項目。請考慮使用 Docker 社群中由其他人所建立並維護的較小映象，或者使用 Docker 的最小 Scratch 映像建立您自己的基礎映像。如需詳細資訊，請參閱 Docker 文件中的[建立基礎映像](#)。

先前在 Dockerfile 中放置最少更改的依存項目

Docker 快取層可加速建立時間。如果最後一次建立後分層便沒有任何變更，Docker 會使用快取的版本，而不會重新建立該分層。然而，每一分層會以之前的分層作為依據。如果分層變更了，Docker 不僅會重新編譯該分層，也會重新編譯任何之後的分層。

若要將重建 Docker 檔案與重新上傳分層的時間縮至最短，請考慮在您的 Docker 檔案中先置放變更頻率最少的依存項目。並稍後在堆疊中置放快速變更的依存項目 (例如您的應用程式原始碼)。

鏈結命令以避免不必要的檔案儲存

分層上建立的中介檔案即使在後續分層中被刪除了，仍將保持為該分層的一部分。請思考下列範例：

```
WORKDIR /tmp
RUN wget http://example.com/software.tar.gz
RUN wget tar -xvf software.tar.gz
RUN mv software/binary /opt/bin/myapp
RUN rm software.tar.gz
```

在此範例中，由第一、二個 RUN 命令所建立的分層仍將包括原始的 .tar.gz 檔案以及其所有解壓縮的內容。即使第四個 RUN 命令已將 .tar.gz 檔案刪除，這些命令可鏈結在一起成為單一的 RUN 陳述式，以確保這些不必要的檔案不會成為最終 Docker 映像的一部分：

```
WORKDIR /tmp
```

```
RUN wget http://example.com/software.tar.gz &&\
    wget tar -xvf software.tar.gz &&\
    mv software/binary /opt/bin/myapp &&\
    rm software.tar.gz
```

使用最接近的區域端點

您可透過使用最接近您執行中的應用程式的區域端點，以減少從 Amazon ECR 提取映像的延遲。如果您的應用程式正在 Amazon EC2 執行個體上執行，您可使用下列 shell 程式碼以從該執行個體的可用區域取得區域：

```
REGION=$(curl -s http://169.254.169.254/latest/meta-data/placement/availability-zone
|\
    sed -n 's/\(\d*\)[a-zA-Z]*$/\1/p')
```

可以使用 `--region` 參數將區域傳遞至 AWS CLI 命令，或使用 `aws configure` 命令將設定為設定檔的預設區域。您也可以在使用 AWS SDK 進行呼叫時設定區域。如需詳細資訊，請參閱您特定程式設計語言的開發套件文件。

向 Amazon ECR 登錄檔提出請求

您可以使用 IPv4-only 端點或雙堆疊 (IPv4 和 IPv6) 端點，在 Amazon ECR 私有登錄檔中推送、提取、刪除、檢視和管理 OCI 映像、Docker 映像和 OCI 相容成品。若要從 IPv4 網路提出請求，您可以使用雙堆疊或 IPv4 端點。若要從 IPv6 網路提出請求，請使用雙堆疊端點。如需使用 IPv4 和雙堆疊端點向 Amazon ECR Public 登錄檔提出請求的詳細資訊，請參閱[向 Amazon ECR Public 登錄檔提出請求](#)。透過 IPv6 存取 Amazon ECR 無需額外費用。如需定價的詳細資訊，請參閱[Amazon Elastic Container Registry 定價](#)。

Amazon ECR 端點是由 IPv4-only 端點或雙堆疊端點支援以外的屬性指定。這些屬性可能包括：

- 區域 – 每個端點都專屬於一個區域。
- 類型 – 端點選擇取決於您使用的是 AWS SDK 還是 OCI 相容和 Docker 命令列界面。
- 安全性 – 在特定區域中，Amazon ECR 提供符合 FIPS 標準的端點。如需符合 FIPS 標準的 Amazon ECR 端點清單的詳細資訊，請參閱[聯邦資訊處理標準 \(FIPS\) 140-3](#)。

如需 IPv4、雙堆疊、Docker 和 OCI 用戶端支援之服務端點的詳細資訊，其會處理來自 CLI 和 AWS SDKs Amazon ECR API AWS 呼叫，請參閱[服務端點](#)。

透過 IPv6 提出請求的入門

若要透過 IPv6 向 Amazon ECR 登錄檔提出請求，您需要使用雙堆疊端點。透過 IPv6 存取 Amazon ECR 登錄檔之前，請確認下列需求：

- 您的用戶端和網路必須支援 IPv6。
- Amazon ECR 透過 IPv6 支援下列請求類型：
 - OCI 和 Docker 用戶端請求：

```
<registry-id>.dkr-ecr.<aws-region>.on.aws
```

- AWS API 請求：

```
ecr.<aws-region>.api.aws
```

- 您必須更新任何使用來源 IP 地址篩選的 AWS Identity and Access Management (IAM) 或登錄政策，以包含 IPv6 地址範圍。如需詳細資訊，請參閱[在 IAM 原則中使用 IPv6 地址](#)。
- 當您使用 IPv6 時，伺服器存取日誌會以 IPv6 格式顯示 Remote IP 地址。更新現有的工具、指令碼和軟體，以剖析這些 IPv6-formatted 的 IP 地址。

Note

如果碰到與日誌檔案中 IPv6 地址相關的問題，請聯絡 [AWS 支援](#)。

測試 IP 地址相容性

如果使用 Linux/Unix 或 Mac OS X，您可以使用 `curl` 命令測試能否透過 IPv6 存取雙堆疊端點，如下例所示：

Example

```
curl --verbose https://ecr.us-west-2.api.aws
```

您會收到類似下列的資訊。如果您是透過 IPv6 連線，則連線的 IP 地址就會是 IPv6 地址。

```
* About to connect() to ecr.us-west-2.api.aws port 443 (#0)
* Trying IPv6 address... connected
* Connected to ecr.us-west-2.api.aws (IPv6 address) port 443 (#0)
> Host: ecr.us-west-2.api.aws
* Request completely sent off
```

如果您使用的是 Microsoft Windows 7 或 Windows 10，您可以使用 `ping` 命令來測試是否可以透過 IPv4 或 IPv6 存取雙堆疊端點，如下列範例所示。

```
ping ecr.us-west-2.api.aws
```

使用雙重堆疊端點透過 IPv6 提出請求

您可以使用雙堆疊端點透過 IPv6 進行 Amazon ECR API 呼叫。無論您使用 IPv4 或 IPv6，Amazon ECR API 操作的功能和效能都保持一致。

當您使用 AWS Command Line Interface (AWS CLI) AWS SDKs 時，您可以使用參數或旗標切換到雙堆疊端點，或直接在組態檔案中指定雙堆疊端點來覆寫預設的 Amazon ECR 端點，以啟用 IPv6。您也可以使用在預設設定檔中 `use_dualstack_endpoint` 設定為 `true` 的命令進行組態變更。如需的詳細資訊 `use_dualstack_endpoint`，請參閱 [雙堆疊和 FIPS 端點](#)。

Example 使用 命令進行組態變更

```
aws configure set default.ecr.use_dualstack_endpoint true
```

Example 使用 透過 IPv6 提出請求 AWS CLI

```
aws ecr describe-repositories --region us-west-2 --endpoint-url https://  
ecr.us-west-2.api.aws
```

從 docker CLI 使用 Amazon ECR 端點

登入 Amazon ECR 儲存庫並標記映像後，您可以在 Amazon ECR 登錄檔之間推送和提取 OCI 映像和 Docker 映像。下列範例示範使用雙堆疊端點的 `docker push` 和 `docker pull` 命令。

Example 使用 IPv4 端點推送 Docker 映像

```
docker push <registry-id>.dkr.ecr.us-west-1.amazonaws.com/my-repository:tag
```

Example 使用雙堆疊端點推送 Docker 映像

```
docker push <registry-id>.dkr-ecr.us-west-1.on.aws/my-repository:tag
```

Example 使用 IPv4 端點提取 Docker 映像

```
docker pull <registry-id>.dkr.ecr.us-west-1.amazonaws.com/my-repository:tag
```

Example 使用雙堆疊端點提取 Docker 映像

```
docker pull <registry-id>.dkr-ecr.us-west-1.on.aws/my-repository:tag
```

在 IAM 原則中使用 IPv6 地址

使用 IPv6 存取登錄檔之前，請確定使用 IP 地址篩選的 IAM 使用者和 Amazon ECR 登錄檔政策包含 IPv6 地址範圍。如果未更新 IP 地址篩選政策來處理 IPv6 地址，用戶端可能會在開始使用 IPv6 時錯誤遺失或存取登錄檔。如需 IAM 管理存取許可的詳細資訊，請參閱 [Amazon Elastic Container Registry 的 Identity and Access Management](#)。

篩選 IP 地址的 IAM 原則使用 [IP 地址條件運算子](#)。下列登錄政策範例示範如何使用 IP 地址條件運算子來識別允許的 IPv4 地址 `54.240.143.*` 範圍。超出此範圍的任何 IP 地址都會被拒絕存取登錄檔 (`exampleregistry`)。由於所有 IPv6 地址都超出允許的範圍，因此此政策會防止 IPv6 地址存取 `exampleregistry`。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "IPAllow",
      "Effect": "Allow",
      "Principal": "*",
      "Action": "ecr:*",
      "Resource": "arn:aws:ecr:*:*:repository/exampleregistry/*",
      "Condition": {
        "IpAddress": {"aws:SourceIp": "54.240.143.0/24"}
      }
    }
  ]
}
```

若要同時允許 IPv4 (54.240.143.0/24) 和 IPv6 (2001:DB8:1234:5678::/64) 地址範圍，請修改登錄政策的條件元素，如下列範例所示。您可以使用此Condition區塊格式來更新 IAM 使用者和登錄檔政策。

```
"Condition": {
  "IpAddress": {
    "aws:SourceIp": [
      "54.240.143.0/24",
      "2001:DB8:1234:5678::/64"
    ]
  }
}
```

⚠ Important

在使用 IPv6 之前，您必須更新使用 IP 地址篩選的所有相關 IAM 使用者和登錄政策。我們不建議在登錄政策中使用 IP 地址篩選。

您可以使用 <https://console.aws.amazon.com/iam/> 的 IAM 主控台來檢閱 IAM 使用者原則。如需 IAM 的詳細資訊，請參閱《[IAM 使用者指南](#)》。

Amazon ECR 私有登錄檔

Amazon ECR 私有登錄檔採用可用度高且可擴展的架構來託管您的容器映像。您可以使用自己的私有登錄檔來管理由 Docker 和開放容器計畫 (OCI) 映像和成品組成的私有映像儲存庫。每個 AWS 帳戶都提供預設的私有 Amazon ECR 登錄檔。如需 Amazon ECR 公有登錄檔的詳細資訊，請參閱《Amazon Elastic Container Registry Public 使用者指南》中的[公有登錄檔](#)。

私有登錄檔概念

- 預設私有登錄檔的 URL 為 `https://aws_account_id.dkr.ecr.region.amazonaws.com`。
- 根據預設，您的帳戶在私有登錄檔中擁有讀取與寫入存取權。不過，使用者需要許可才能呼叫 Amazon ECR APIs，以及將映像推送至私有儲存庫或從中提取映像。Amazon ECR 提供數個受管政策，以控制不同層級的使用者存取。如需詳細資訊，請參閱 [Amazon Elastic Container Registry 身分型政策的範例](#)。
- 必須授權您的 Docker 用戶端到私有登錄檔，才可使用 `docker push` 與 `docker pull` 命令來推送映像至該登錄檔中的儲存庫及自該儲存庫中提取映像。如需詳細資訊，請參閱 [Amazon ECR 中的私有登錄檔身分驗證](#)。
- 私有儲存庫可透過 使用者存取政策及儲存庫政策加以控制。如需有關儲存庫政策的詳細資訊，請參閱 [Amazon ECR 中的私有儲存庫政策](#)。
- 私有登錄檔中的儲存庫可以透過設定私有登錄檔的複寫，跨 AWS 自有私有登錄檔中的區域和跨個別帳戶進行複寫。如需詳細資訊，請參閱 [Amazon ECR 中的私有映像複寫](#)。

Amazon ECR 中的私有登錄檔身分驗證

您可以使用 AWS 管理主控台 AWS CLI、或 AWS SDKs 來建立和管理私有儲存庫。您可以使用這些方法來在映像上執行部分動作，例如列清單或刪除。這些用戶端使用標準 AWS 身分驗證方法。即使您可以使用 Amazon ECR API 推送並提取映像，您仍較有可能使用 Docker CLI 或依語言而定的 Docker 資料庫。

Docker CLI 不支援原生的 IAM 驗證方法。必須採取額外的步驟，Amazon ECR 才能驗證及授權 Docker 推送與提取請求。

下列各節詳述的登錄檔驗證方法可用。

使用 Amazon ECR 憑證協助程式

Amazon ECR 提供 Docker 憑證協助程式，可在推送和提取映像至 Amazon ECR 時更容易儲存和使用 Docker 憑證。如要了解安裝和設定步驟，請參閱 [Amazon ECR Docker 憑證協助程式](#)。

Note

Amazon ECR Docker 憑證協助程式目前不支援多重要素驗證 (MFA)。

使用授權字符

授權字符的許可權範圍與用來擷取身分驗證字符之 IAM 委託人的許可範圍相符。身分驗證字符是用來存取 IAM 委託人有權存取且有效期為 12 小時的任何 Amazon ECR 登錄檔。若要取得身分驗證字符，您必須使用 [GetAuthorizationToken](#) API 操作，才能擷取包含使用者名稱 AWS 和編碼密碼的 base64 編碼授權字符。AWS CLI `get-login-password` 命令透過擷取和解碼授權字符來簡化此過程，然後您可以將其輸送到 `docker login` 命令以進行驗證。

若要使用 `get-login` 向 Amazon ECR 私有登錄檔驗證 Docker

- 若要使用 `get-login-password` 向 Amazon ECR 登錄檔驗證 Docker，請執行 `aws ecr get-login-password` 命令。將身分驗證字符傳遞給 `docker login` 命令時，使用 AWS 的值作為使用者名稱並指定您要驗證的 Amazon ECR 登錄檔 URI。如果是向多個登錄進行驗證，您必須針對每個登錄重複此命令。

Important

若您收到錯誤，請安裝或升級至最新版本的 AWS CLI。如需詳細資訊，請參閱《AWS Command Line Interface 使用者指南》中的 [安裝 AWS Command Line Interface](#)。

- [get-login-password](#) (AWS CLI)

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

- [Get-ECRLoginCommand](#) (AWS Tools for Windows PowerShell)

```
(Get-ECRLoginCommand).Password | docker login --username AWS --password-stdin aws_account_id.dkr.ecr.region.amazonaws.com
```

使用 HTTP API 身分驗證

Amazon ECR 支援 [Docker 登錄檔 HTTP API](#)。但是，因 Amazon ECR 為私有登錄檔，您必須使用每個 HTTP 請求來提供驗證字符。您可以使用的 `-H` 選項新增 HTTP 授權標頭，`curl` 並傳遞 `get-authorization-token` AWS CLI 命令提供的授權字符。

使用 Amazon ECR HTTP API 進行身分驗證

1. 使用擷取授權字符，AWS CLI 並將其設定為環境變數。

```
TOKEN=$(aws ecr get-authorization-token --output text --query 'authorizationData[].authorizationToken')
```

2. 將 `$TOKEN` 變數傳遞到 `curl` 的 `-H` 選項來向 API 驗證身分。例如，下列命令會列出在 Amazon ECR 儲存庫中的映像標籤。如需詳細資訊，請參閱 [Docker 登錄檔 HTTP API](#) 參考文件。

```
curl -i -H "Authorization: Basic $TOKEN" https://aws_account_id.dkr.ecr.region.amazonaws.com/v2/amazonlinux/tags/list
```

其輸出如下：

```
HTTP/1.1 200 OK
Content-Type: text/plain; charset=utf-8
Date: Thu, 04 Jan 2018 16:06:59 GMT
Docker-Distribution-Api-Version: registry/2.0
Content-Length: 50
Connection: keep-alive

{"name":"amazonlinux","tags":["2017.09","latest"]}
```

Amazon ECR 中的私有登錄檔設定

Amazon ECR 使用私有登錄檔設定在登錄檔層級設定功能。私有登錄檔設定會針對每個區域分別設定。您可以使用私有登錄檔設定來設定下列功能。

- 登錄檔許可 – 登錄檔許可政策提供對複寫的控制，並提取快取許可。如需詳細資訊，請參閱[Amazon ECR 中的私有登錄檔許可](#)。
- 提取快取規則 – 提取快取規則用於從 Amazon ECR 私有登錄檔中的上游登錄檔快取映像。如需詳細資訊，請參閱[將上游登錄檔與 Amazon ECR 私有登錄檔同步](#)。
- 複寫組態 – 複寫組態用於控制您的儲存庫是否跨 AWS 區域 或 複製 AWS 帳戶。如需詳細資訊，請參閱[Amazon ECR 中的私有映像複寫](#)。
- 儲存庫建立範本 – 儲存庫建立範本用於定義 Amazon ECR 代表您建立新儲存庫時要套用的標準設定。例如，提取快取動作、推送時建立或複寫所建立的儲存庫。如需詳細資訊，請參閱[用於控制提取快取、推送時建立或複寫動作期間建立的儲存庫的範本](#)。
- 掃描組態 – 根據預設，您的登錄檔已啟用基本掃描。您可以啟用提供自動化連續掃描模式的增強型掃描，來掃描作業系統和程式設計語言套件的弱點。如需詳細資訊，請參閱[掃描映像是否有 Amazon ECR 中的軟體漏洞](#)。
- 提取時間更新排除 – 您可以設定提取時間更新排除，以防止提取特定映像時的上次提取時間更新。這適用於用於測試或 CI/CD 用途的影像，而您不希望提取時間影響生命週期政策決策。如需詳細資訊，請參閱[提取時間更新排除](#)。
- Blob 掛載組態 – Blob 掛載組態用於控制登錄檔中的儲存庫是否共用共同層，而不是存放重複層。如需詳細資訊，請參閱[Amazon ECR 中的 Blob 掛載](#)。

Amazon ECR 中的 Blob 掛載

Amazon ECR 支援稱為 Blob 掛載的功能，以在登錄檔中的儲存庫之間共用常見的映像層。啟用時，單一登錄檔中的儲存庫可以參考相同登錄檔中其他儲存庫的層，而不是儲存重複的複本。

啟用登錄 Blob 掛載時，Amazon ECR 會在包含掛載參數的推送操作期間檢查登錄檔中的現有層。如果圖層已存在於相同登錄檔中的另一個儲存庫中，Amazon ECR 會掛載現有的圖層，而不是上傳重複的圖層。

Note

如果 OCI 用戶端偵測到 Blob 可能已存在於不同的儲存庫中，則會自動包含掛載參數。只有在用戶端的 POST 請求中存在這些參數時，Amazon ECR 才會嘗試掛載。

Blob 掛載概念

- Blob 掛載僅適用於相同的登錄檔（相同的帳戶和區域）。

- 儲存庫必須使用相同的加密類型和金鑰。
- 透過提取快取建立的影像不支援 Blob 掛載。
- 如果您決定停用 Blob 掛載，已設定 Blob 掛載推送的現有映像會繼續運作，而圖層會保持掛載。

Blob 掛載組態

您可以使用 AWS 管理主控台 或 AWS CLI 來設定登錄檔的 Blob 掛載。

Note

使用者需要儲存庫的 `ecr:GetDownloadUrlForLayer` IAM 許可，才能從中掛載層。

AWS 管理主控台

使用下列步驟來更新登錄檔的 Blob 掛載組態 AWS 管理主控台。

開啟私有登錄檔的 Blob 掛載組態

1. 在 <https://console.aws.amazon.com/ecr/private-registry/repositories> 開啟 Amazon ECR 主控台
2. 從導覽列中，選擇區域。
3. 在導覽窗格中，選擇私有登錄檔、功能和設定，然後選擇 Blob 掛載。
4. 在 Blob 掛載頁面上，選擇啟用。

此時會顯示橫幅，指出 Blob 掛載組態已更新為啟用。

AWS CLI

使用以下命令，使用 更新登錄檔的 Blob 掛載組態 AWS CLI。

- ```
aws ecr put-account-setting --name BLOB_MOUNTING --value ENABLED
```

## Amazon ECR 中的私有登錄檔許可

Amazon ECR 使用登錄檔政策，將許可授予在私有登錄檔層級的 AWS 主體。

Amazon ECR 允許政策中的所有 ECR 動作，並在所有 ECR 請求中強制執行登錄政策。您可以使用登錄政策來授予複寫組態、提取快取規則建立和儲存庫建立等動作的許可。如需 API 動作的完整清單，請參閱 [Amazon ECR API 指南](#)。如需有關 Amazon ECR 私有登錄檔一般設定的資訊，請參閱 [Amazon ECR 中的私有登錄檔設定](#)。

#### Note

雖然可以將 `ecr:*` 動作新增至私有登錄檔政策，但最佳實務是僅根據您正在使用的功能新增所需的特定動作，而不是使用萬用字元。

## 主題

- [Amazon ECR 的私有登錄檔政策範例](#)
- [授予 Amazon ECR 中跨帳戶複寫的登錄檔許可](#)
- [授予 Amazon ECR 中提取快取的登錄檔許可](#)

## Amazon ECR 的私有登錄檔政策範例

以下範例顯示您可以用來控制使用者具有之 Amazon ECR 登錄檔許可的登錄檔許可政策陳述式。

#### Note

在每個範例中，如果從登錄檔政策中移除 `ecr:CreateRepository` 動作，則仍可進行複寫。但是，為了成功複寫，您需要在帳戶中建立具有相同名稱的儲存庫。

### 範例：允許來源帳戶中的所有 IAM 主體複寫所有儲存庫

下列登錄檔許可政策允許來源帳戶中的所有 IAM 主體（使用者和角色）複寫所有儲存庫。

注意下列事項：

- **重要：**當您在政策中將 AWS 帳戶 ID 指定為委託人時，您可以將存取權授予該帳戶中的所有 IAM 使用者和角色，而不只是根使用者。這可在整個帳戶中提供廣泛的存取。
- **安全考量：**帳戶層級許可授予指定帳戶中所有 IAM 實體的存取權。如需更嚴格的存取，請指定個別 IAM 使用者、角色，或使用條件陳述式進一步限制存取。

## JSON

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Sid": "ReplicationAccessCrossAccount",
 "Effect": "Allow",
 "Principal": {
 "AWS": "arn:aws:iam::111122223333:root"
 },
 "Action": [
 "ecr:CreateRepository",
 "ecr:ReplicateImage"
],
 "Resource": [
 "arn:aws:ecr:us-west-2:444455556666:repository/*"
]
 }
]
}
```

## 範例：允許來自多個帳戶的 IAM 主體

下列登錄檔許可政策有兩個陳述式。每個陳述式都允許來源帳戶中的所有 IAM 主體（使用者和角色）複寫所有儲存庫。

## JSON

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Sid": "ReplicationAccessCrossAccount1",
 "Effect": "Allow",
 "Principal": {
 "AWS": "arn:aws:iam::111122223333:root"
 },
 "Action": [
 "ecr:CreateRepository",

```

```

 "ecr:ReplicateImage"
],
 "Resource": [
 "arn:aws:ecr:us-west-2:123456789012:repository/*"
]
},
{
 "Sid": "ReplicationAccessCrossAccount2",
 "Effect": "Allow",
 "Principal": {
 "AWS": "arn:aws:iam::444455556666:root"
 },
 "Action": [
 "ecr:CreateRepository",
 "ecr:ReplicateImage"
],
 "Resource": [
 "arn:aws:ecr:us-west-2:123456789012:repository/*"
]
}
]
}

```

範例：允許來源帳戶中的所有 IAM 主體複寫字首為 的所有儲存庫 **prod-**。

下列登錄檔許可政策允許來源帳戶中的所有 IAM 主體（使用者和角色）複寫以 開頭的所有儲存庫 **prod-**。

JSON

```

{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Sid": "ReplicationAccessCrossAccount",
 "Effect": "Allow",
 "Principal": {
 "AWS": "arn:aws:iam::111122223333:root"
 },
 "Action": [
 "ecr:CreateRepository",
]
 }
]
}

```

```
 "ecr:ReplicateImage"
],
 "Resource": [
 "arn:aws:ecr:us-west-2:444455556666:repository/prod-*"
]
 }
]
}
```

## 授予 Amazon ECR 中跨帳戶複寫的登錄檔許可

跨帳戶政策類型用於將許可授予 AWS 委託人，允許將儲存庫從來源登錄檔複寫到您的登錄檔。根據預設，您可以在自己的登錄檔中設定跨區域複寫的許可。您只需要設定登錄檔政策，如果您授予另一個帳戶將內容複寫到登錄檔的許可。

登錄檔政策必須授予 `ecr:ReplicateImage` API 動作的許可。這個 API 為內部的 Amazon ECR API，可以在區域或帳戶之間複寫映像。您也可以授予 `ecr:CreateRepository` 許可，這允許 Amazon ECR 在您的登錄檔中建立儲存庫 (如果它們尚不存在)。如果未提供 `ecr:CreateRepository` 許可，則必須在登錄檔中手動建立具有與來源儲存庫相同名稱的儲存庫。如果兩者都未完成，複寫則會失敗。任何失敗的 `CreateRepository` 或 `ReplicateImage` API 動作都會顯示在 CloudTrail 中。

### 設定複寫許可政策 (AWS 管理主控台)

1. 在 <https://console.aws.amazon.com/ecr/> 開啟 Amazon ECR 主控台。
2. 從導覽列，選擇要在其中設定登錄檔政策的區域。
3. 在導覽窗格中，選擇私有登錄檔，選擇功能和設定，然後選擇許可。
4. 在 Registry permissions (登錄檔許可) 頁面上，選擇 Generate statement (產生陳述式)。
5. 使用政策產生器完成下列步驟以定義您的政策陳述式。
  - a. 針對政策類型，選擇複寫 - 跨帳戶。
  - b. 針對陳述式 ID，輸入唯一的陳述式 ID。此欄位用作 Sid 在登錄檔政策上。
  - c. 對於 Accounts (帳戶)，輸入您要授予許可的每個帳戶的帳戶 ID。指定多個帳戶 ID 時，以逗號分隔。
6. 選擇儲存。

## 設定複寫許可政策 (AWS CLI)

1. 建立名為 `registry_policy.json` 的檔案，並將其填入登錄檔政策。

### JSON

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Sid": "ReplicationAccessCrossAccount",
 "Effect": "Allow",
 "Principal": {
 "AWS": "arn:aws:iam::111122223333:root"
 },
 "Action": [
 "ecr:CreateRepository",
 "ecr:ReplicateImage"
],
 "Resource": [
 "arn:aws:ecr:us-west-2:444455556666:repository/*"
]
 }
]
}
```

2. 使用政策檔案建立登錄檔政策。

```
aws ecr put-registry-policy \
 --policy-text file://registry_policy.json \
 --region us-west-2
```

3. 擷取登錄檔的政策以確認。

```
aws ecr get-registry-policy \
 --region us-west-2
```

## 授予 Amazon ECR 中提取快取的登錄檔許可

Amazon ECR 私有登錄檔許可可用來設定個別 IAM 實體使用提取快取的許可範圍。如果 IAM 實體擁有的由 IAM 政策授予的許可多過登錄檔許可政策授予的許可，則 IAM 政策優先。

建立私有登錄檔的許可政策 (AWS 管理主控台)

1. 在 <https://console.aws.amazon.com/ecr/> 開啟 Amazon ECR 主控台。
2. 從導覽列選擇要在其中設定私有登錄檔許可陳述式的區域。
3. 在導覽窗格中，選擇私有登錄檔，選擇功能和設定，然後選擇許可。
4. 在 Registry permissions (登錄檔許可) 頁面上，選擇 Generate statement (產生陳述式)。
5. 針對您要建立的每個提取快取許可政策陳述式，執行下列動作。
  - a. 針對 Policy type (政策類型)，選擇 Pull through cache policy (提取快取政策)。
  - b. 針對 Statement id (陳述式 ID)，提供提取快取陳述式政策的名稱。
  - c. 針對 IAM entities (IAM 實體)，指定要包含在政策中的使用者、群組或角色。
  - d. 針對快取命名空間，選取要與政策建立關聯的提取快取規則。
  - e. 針對 Repository names (儲存庫名稱)，指定要套用規則的儲存庫基本名稱。例如，如果您想要在 Amazon ECR Public 上指定 Amazon Linux 儲存庫，則儲存庫名稱會是 `amazonlinux`。

# Amazon ECR 私有儲存庫

Amazon ECR 私有儲存庫包含您的 Docker 映像、開放式容器計畫 (OCI) 映像和 OCI 相容成品。您可以建立、監控和刪除映像儲存庫，並設定許可，以控制誰可以使用 Amazon ECR API 操作或 Amazon ECR 主控台的儲存庫區段來存取這些儲存庫。Amazon ECR 也與 Docker CLI 整合，因此您可以將映像從開發環境推送和提取到儲存庫。

## 主題

- [私有儲存庫概念](#)
- [建立 Amazon ECR 私有儲存庫以存放映像](#)
- [在 Amazon ECR 中檢視私有儲存庫的內容和詳細資訊](#)
- [在 Amazon ECR 中刪除私有儲存庫](#)
- [Amazon ECR 中的私有儲存庫政策](#)
- [在 Amazon ECR 中標記私有儲存庫](#)

## 私有儲存庫概念

- 根據預設，您的帳戶在預設登錄檔中擁有讀取與寫入存取權 (`aws_account_id.dkr.ecr.region.amazonaws.com`)。然而，使用者需要許可來對 Amazon ECR API 進行呼叫，並從您的儲存庫推送或提取映像。Amazon ECR 提供數個受管政策，以控制不同層級的使用者存取。如需詳細資訊，請參閱[Amazon Elastic Container Registry 身分型政策的範例](#)。
- 儲存庫可透過 使用者存取政策及個別儲存庫政策加以控制。如需詳細資訊，請參閱[Amazon ECR 中的私有儲存庫政策](#)。
- 儲存庫名稱可支援命名空間，您也可用該命名空間為相似的儲存庫分組。例如，如果有數個團隊使用相同的登錄檔，團隊 A 可使用 `team-a` 命名空間，而團隊 B 可使用 `team-b` 命名空間。如果這麼做，每個團隊都有自己的名為 `web-app` 的映像，每個映像都以團隊命名空間開頭。此組態允許在不干擾的情況下同時使用每個團隊上的這些映像。A 團隊的映像為 `team-a/web-app`，B 團隊的映像為 `team-b/web-app`。
- 您的映像可以在您自己的登錄檔和帳戶之間複寫到其他儲存庫。您可以藉由在登錄檔設定中指定複寫組態來執行這項操作。如需詳細資訊，請參閱[Amazon ECR 中的私有登錄檔設定](#)。
- 在登錄檔層級啟用 Blob 掛載時，儲存庫可以共用常見的映像層。

# 建立 Amazon ECR 私有儲存庫以存放映像

## Important

使用 AWS KMS (DSSE-KMS) 的雙層伺服器端加密僅適用於 AWS GovCloud (US) 區域。

建立 Amazon ECR 私有儲存庫，然後使用儲存庫來存放容器映像。遵循以下步驟，使用 AWS 管理主控台建立私有儲存庫。

### 建立儲存庫 (AWS 管理主控台)

1. 在 <https://console.aws.amazon.com/ecr/repositories> 開啟 Amazon ECR 主控台。
2. 從導覽列選擇您儲存庫所建立的區域。
3. 選擇私有儲存庫，然後選擇建立儲存庫。
4. 在 Repository name (儲存庫名稱) 中，為您的儲存庫輸入獨一無二的名稱。儲存庫名稱可以自行指定 (例如 nginx-web-app)。或者，可以在其前面加上命名空間，以將儲存庫分組到類別中 (例如 project-a/nginx-web-app)。

## Note

儲存庫名稱可以是最多 256 個字元的容器。名稱必須以字母開頭，且只能包含小寫字母、數字、連字號、底線、句號和斜線。不支援使用雙正斜線。

5. 針對影像標籤不可變性，選擇下列其中一個儲存庫的標籤可變性設定。
  - 可互斥 – 如果您想要覆寫影像標籤，請選擇此選項。建議用於使用提取快取動作的儲存庫，以確保 Amazon ECR 可以更新快取映像。此外，若要停用幾個可變標籤的標籤更新，請輸入標籤名稱或使用萬用字元 (\*) 來比對 Mutable 標籤排除文字方塊中的多個類似標籤。
  - 不可避免 – 如果您想要防止映像標籤遭到覆寫，且當推送具有現有標籤的映像時，它會套用至儲存庫中的所有標籤和排除項目，請選擇此選項。ImageTagAlreadyExistsException 如果您嘗試推送具有現有標籤的映像，Amazon ECR 會傳回。此外，若要啟用幾個不可變標籤的標籤更新，請輸入標籤名稱或使用萬用字元 (\*) 在不可分割標籤排除文字方塊中比對多個類似的標籤。

**Note**

不支援個別標籤可變性設定。

6. 針對加密組態，選擇 AES-256 或 AWS KMS。如需詳細資訊，請參閱[靜態加密](#)。
  - a. 如果選擇 AWS KMS，請在單層加密和雙層加密之間進行選擇。使用 AWS KMS 或雙層加密需支付額外費用。如需詳細資訊，請參閱 [Amazon ECR Service 定價](#)。
  - b. 根據預設，aws/ecr 會選擇具有別名的 AWS 受管金鑰。當您第一次建立啟用 AWS KMS 加密的儲存庫時，會在您的帳戶中建立此金鑰。選取客戶受管金鑰（進階）以選擇您自己的 AWS KMS 金鑰。AWS KMS 金鑰必須與叢集位於相同的區域。選取建立 AWS KMS 金鑰以導覽至 AWS KMS 主控台，以建立您自己的金鑰。
7. 對於映像掃描設定，雖然您可以在儲存庫層級指定基本掃描的掃描設定，但最佳實務是在私有登錄檔層級指定掃描組態。在私有登錄檔層級設定掃描設定可讓您在增強型掃描或基本掃描之間進行選擇，也可讓您定義篩選條件以指定應掃描的儲存庫。
8. 選擇建立。

## 建立儲存庫 (AWS CLI)

1. 您可以使用 AWS CLI 搭配 `aws ecr create-repository` 命令來建立儲存庫。

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

2. 如果您已定義儲存庫建立範本，您可以使用熟悉的 Amazon ECR 推送命令搭配所需的儲存庫名稱來推送映像，以建立儲存庫。Amazon ECR 將使用儲存庫建立範本的預先定義設定，自動為您建立儲存庫。如果您尚未定義儲存庫建立範本，對不存在映像儲存庫的請求將會失敗。

```
docker push aws_account_id.dkr.ecr.region.amazonaws.com/prefix/my-new-repository:tag
```

## 後續步驟

若要檢視將映像推送到儲存庫的步驟，請選取儲存庫，然後選擇檢視推送命令。如需將映像推送到您的儲存庫的詳細資訊，請參閱 [將映像推送至 Amazon ECR 私有儲存庫](#)。

## 在 Amazon ECR 中檢視私有儲存庫的內容和詳細資訊

建立私有儲存庫之後，您可以在 [中檢視儲存庫的詳細資訊](#) AWS 管理主控台：

- 在儲存庫中存放的是哪些映像
- 有關存放在儲存庫中每個映像的詳細資訊，包括每個映像的大小和 SHA 摘要
- 指定的儲存庫內容掃描頻率
- 儲存庫是否具有與其相關聯的作用中提取快取規則
- 儲存庫的加密設定

### Note

若以 Docker 1.9 版本開始，Docker 用戶端在將映像推送至 V2 Docker 登錄檔前，會先壓縮映像層。docker images 命令的輸出會顯示未壓縮的映像大小。因此，請記住，Docker 可能會傳回比 AWS 管理主控台中顯示的映像更大的映像。

### 檢視儲存庫資訊 (AWS 管理主控台)

1. 在 <https://console.aws.amazon.com/ecr/repositories> 開啟 Amazon ECR 主控台。
2. 從導覽列上，選擇包含要檢視的儲存庫區域。
3. 在導覽窗格中，選擇 Repositories (儲存庫)。
4. 在 Repositories (儲存庫) 頁面上，選擇 Private (私有) 分頁，然後選擇要檢視的儲存庫。
5. 在儲存庫詳細資訊頁面上，主控台預設為顯示 Images (映像) 檢視。使用導覽選單檢視與儲存庫有關的其他資訊。
  - 選擇 Summary (摘要) 檢視儲存庫詳細資訊和儲存庫的提取計數資料。
  - 選擇 Images (映像) 檢視關於儲存庫中映像標籤的資訊。若要檢視關於映像的詳細資訊，請選取映像標籤。如需詳細資訊，請參閱 [在 Amazon ECR 中檢視影像詳細資訊](#)。

如果有您想刪除的未標記映像，您可以選擇儲存庫左方的方塊以刪除並選擇 Delete(刪除)。如需詳細資訊，請參閱 [在 Amazon ECR 中刪除映像](#)。

- 選擇 Permissions (許可) 以檢視套用到儲存庫的儲存庫政策。如需詳細資訊，請參閱[Amazon ECR 中的私有儲存庫政策](#)。
- 選擇 Lifecycle Policy (生命週期政策) 以檢視套用到儲存庫的生命週期政策。您也可在此檢視生命週期事件歷史。如需詳細資訊，請參閱[在 Amazon ECR 中使用生命週期政策來自動化映像的清除](#)。
- 選擇 Tags (標籤) 以檢視套用到儲存庫的中繼資料標籤。

## 在 Amazon ECR 中刪除私有儲存庫

如果儲存庫已使用完畢，即可將其刪除。當您刪除 中的儲存庫時 AWS 管理主控台，儲存庫中包含的所有映像也會一併刪除；這無法復原。

### Important

也會刪除已刪除儲存庫中的影像。您無法復原此操作。

若要刪除儲存庫 (AWS 管理主控台)

1. 在 <https://console.aws.amazon.com/ecr/repositories> 開啟 Amazon ECR 主控台。
2. 從導覽列上，選擇包含要刪除的儲存庫區域。
3. 在導覽窗格中，選擇 Repositories (儲存庫)。
4. 在 Repositories (儲存庫) 頁面上，選擇 Private (私有) 分頁，然後選擇要刪除的儲存庫並選擇 Delete (刪除)。
5. 在 Delete *repository\_name* (刪除 repository\_name) 視窗中，確認應刪除的儲存庫已被選擇，再選擇 Delete (刪除)。

## Amazon ECR 中的私有儲存庫政策

Amazon ECR 使用資源型許可來控制儲存庫的存取。資源型許可可讓您指定哪些使用者或角色可存取儲存庫，以及他們可以對儲存庫執行哪些動作。根據預設，只有建立儲存庫 AWS 的帳戶可以存取儲存庫。您可以套用儲存庫政策，以允許對儲存庫進行其他存取。

### 主題

- [儲存庫政策與 IAM 政策的比較](#)

- [Amazon ECR 中的私有儲存庫政策範例](#)
- [在 Amazon ECR 中設定私有儲存庫政策陳述式](#)

## 儲存庫政策與 IAM 政策的比較

Amazon ECR 儲存庫政策是 IAM 政策的子集，其範圍和專門用於控制對單個 Amazon ECR 儲存庫的存取。IAM 政策通常用於套用整個 Amazon ECR 服務的許可，但也可用於控制對特定資源的存取。

在決定特定使用者或角色可以對儲存庫執行哪些動作時，Amazon ECR 儲存庫政策和 IAM 政策都會用到。如果使用者或角色透過儲存庫政策獲准執行某個動作，但是透過 IAM 政策被拒絕許可 (反之亦然)，則該動作將被拒絕。透過儲存庫政策或 IAM 政策任何一個即可允許使用者或角色的動作許可，但不需要兩者都允許。

### Important

Amazon ECR 要求使用者擁有透過 IAM 政策呼叫 `ecr:GetAuthorizationToken` API 的許可，然後才能對登錄檔進行身分驗證，並從任何 Amazon ECR 儲存庫推送或提取任何映像。Amazon ECR 提供多種受管 IAM 政策，以控制不同層級的使用者存取。如需詳細資訊，請參閱[Amazon Elastic Container Registry 身分型政策的範例](#)。

您可以使用這些政策類型的任何一個，來控制對您儲存庫的存取，如下列範例所示。

此範例顯示一個 Amazon ECR 儲存庫政策，允許特定使用者描述儲存庫和儲存庫中的映像。

JSON

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Sid": "ECRRepositoryPolicy",
 "Effect": "Allow",
 "Principal": {"AWS": "arn:aws:iam::111122223333:user/username"},
 "Action": [
 "ecr:DescribeImages",
 "ecr:DescribeRepositories"
],
 "Resource": "*"
 }
]
}
```

```
]
}
```

此範例顯示一個 IAM 政策，透過使用資源參數將政策的範圍限定在儲存庫 (以儲存庫的完整 ARN 指定)，可達成上述相同的目標。如需有關 Amazon 資源名稱 (ARN) 格式的詳細資訊，請參閱 [Resources](#)。

## JSON

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Sid": "AllowDescribeRepoImage",
 "Effect": "Allow",
 "Action": [
 "ecr:DescribeImages",
 "ecr:DescribeRepositories"
],
 "Resource": ["arn:aws:ecr:us-east-1:111122223333:repository/repository-name"]
 }
]
}
```

## Amazon ECR 中的私有儲存庫政策範例

### Important

此頁面上的儲存庫政策範例旨在套用至 Amazon ECR 私有儲存庫。如果直接與 IAM 主體搭配使用，除非為了將 Amazon ECR 儲存庫指定為資源而進行修改，否則它們將無法正常運作。如需有關設定儲存庫政策的詳細資訊，請參閱 [在 Amazon ECR 中設定私有儲存庫政策陳述式](#)。

Amazon ECR 儲存庫政策是 IAM 政策的子集，其範圍和專門用於控制對單個 Amazon ECR 儲存庫的存取。IAM 政策通常用於套用整個 Amazon ECR 服務的許可，但也可用於控制對特定資源的存取。如需詳細資訊，請參閱 [儲存庫政策與 IAM 政策的比較](#)。

以下儲存庫政策範例顯示您可以用來控制 Amazon ECR 私有儲存庫存取的許可陳述式。

### Important

Amazon ECR 要求使用者擁有透過 IAM 政策呼叫 `ecr:GetAuthorizationToken` API 的許可，然後才能對登錄檔進行身分驗證，並從任何 Amazon ECR 儲存庫推送或提取任何映像。Amazon ECR 提供多種受管 IAM 政策，以控制不同層級的使用者存取。如需詳細資訊，請參閱 [Amazon Elastic Container Registry 身分型政策的範例](#)。

## 範例：允許一或多個使用者

以下儲存庫政策允許一個或多個使用者向儲存庫推送和從儲存庫提取映像。

JSON

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Sid": "AllowPushPull",
 "Effect": "Allow",
 "Principal": {
 "AWS": [
 "arn:aws:iam::111122223333:user/push-pull-user-1",
 "arn:aws:iam::111122223333:user/push-pull-user-2"
]
 },
 "Action": [
 "ecr:BatchGetImage",
 "ecr:BatchCheckLayerAvailability",
 "ecr:CompleteLayerUpload",
 "ecr:GetDownloadUrlForLayer",
 "ecr:InitiateLayerUpload",
 "ecr:PutImage",
 "ecr:UploadLayerPart"
],
 "Resource": "*"
 }
]
}
```

## 範例：允許其他帳戶

下列儲存庫政策允許特定帳戶推入映像。

### Important

您授予許可的帳戶必須啟用您正在建立儲存庫政策的區域，否則會發生錯誤。

## JSON

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Sid": "AllowCrossAccountPush",
 "Effect": "Allow",
 "Principal": {
 "AWS": "arn:aws:iam::111122223333:root"
 },
 "Action": [
 "ecr:BatchCheckLayerAvailability",
 "ecr:CompleteLayerUpload",
 "ecr:InitiateLayerUpload",
 "ecr:PutImage",
 "ecr:UploadLayerPart"
],
 "Resource": "*"
 }
]
}
```

下列儲存庫政策允許部分使用者提取映像 (*pull-user-1* 與 *pull-user-2*)，同時提供其他使用者 (*admin-user*) 完整存取權限。

### Note

如需目前不支援的更複雜儲存庫政策 AWS 管理主控台，您可以使用 [set-repository-policy](#) AWS CLI 命令套用政策。

## JSON

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Sid": "AllowPull",
 "Effect": "Allow",
 "Principal": {
 "AWS": [
 "arn:aws:iam::111122223333:user/pull-user-1",
 "arn:aws:iam::111122223333:user/pull-user-2"
]
 },
 "Action": [
 "ecr:BatchGetImage",
 "ecr:GetDownloadUrlForLayer"
],
 "Resource": "*"
 },
 {
 "Sid": "AllowAll",
 "Effect": "Allow",
 "Principal": {
 "AWS": "arn:aws:iam::111122223333:user/admin-user"
 },
 "Action": [
 "ecr:*"
],
 "Resource": "*"
 }
]
}
```

## 範例：拒絕全部

下列儲存庫政策拒絕所有使用者擁有抽出映像的能力。

## JSON

```
{
```

```

"Version": "2012-10-17",
"Statement": [
 {
 "Sid": "DenyPull",
 "Effect": "Deny",
 "Principal": "*",
 "Action": [
 "ecr:BatchGetImage",
 "ecr:GetDownloadUrlForLayer"
],
 "Resource": "*"
 }
]
}

```

## 範例：限制特定 IP 地址的存取

在套用來自特定地址範圍的儲存庫時，下列範例拒絕任何使用者執行任何 Amazon ECR 操作的許可。

此陳述式中的條件會識別允許之網際網路通訊協定第 4 版 (IPv4) IP 地址的 54.240.143.\* 範圍。

Condition 區塊使用 NotIpAddress 條件和 aws:SourceIp 條件金鑰，這是 AWS 全局條件金鑰。如需有關這些條件索引鍵的詳細資訊，請參閱 [AWS 全域條件內容索引鍵](#)。aws:sourceIp IPv4 值會使用標準 CIDR 表示法。如需詳細資訊，請參閱《IAM 使用者指南》中的 [IP 地址條件運算子](#)。

## JSON

```

{
 "Version": "2012-10-17",
 "Id": "ECRPolicyId1",
 "Statement": [
 {
 "Sid": "IPAllow",
 "Effect": "Deny",
 "Principal": "*",
 "Action": "ecr:*",
 "Resource": "*",
 "Condition": {
 "NotIpAddress": {
 "aws:SourceIp": "54.240.143.0/24"
 }
 }
 }
]
}

```

```

 }
 }
]
}

```

## 範例：允許 AWS 服務

下列儲存庫政策允許 AWS CodeBuild 存取與該服務整合所需的 Amazon ECR API 動作。使用以下範例時，您應該使用 `aws:SourceArn` 和 `aws:SourceAccount` 條件索引鍵來調查可以承擔這些許可的資源。如需詳細資訊，請參閱《AWS CodeBuild 使用者指南》中的 [CodeBuild 的 Amazon ECR 範例](#)。

### JSON

```

{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Sid": "CodeBuildAccess",
 "Effect": "Allow",
 "Principal": {
 "Service": "codebuild.amazonaws.com"
 },
 "Action": [
 "ecr:BatchGetImage",
 "ecr:GetDownloadUrlForLayer"
],
 "Resource": "*",
 "Condition": {
 "ArnLike": {
 "aws:SourceArn": "arn:aws:codebuild:us-east-1:123456789012:project/project-name"
 },
 "StringEquals": {
 "aws:SourceAccount": "123456789012"
 }
 }
 }
]
}

```

## 在 Amazon ECR 中設定私有儲存庫政策陳述式

您可以 AWS 管理主控台 依照下列步驟，將存取政策陳述式新增至 中的儲存庫。您可以為每個儲存庫新增多個政策陳述式。如需範例政策，請參閱 [Amazon ECR 中的私有儲存庫政策範例](#)。

### Important

Amazon ECR 要求使用者擁有透過 IAM 政策呼叫 `ecr:GetAuthorizationToken` API 的許可，然後才能對登錄檔進行身分驗證，並從任何 Amazon ECR 儲存庫推送或提取任何映像。Amazon ECR 提供多種受管 IAM 政策，以控制不同層級的使用者存取。如需詳細資訊，請參閱 [Amazon Elastic Container Registry 身分型政策的範例](#)。

### 設定儲存庫政策陳述式

1. 在 <https://console.aws.amazon.com/ecr/repositories> 開啟 Amazon ECR 主控台。
2. 從導覽列上，選擇其中包含要設定政策陳述式的儲存庫之區域。
3. 在導覽窗格中，選擇 Repositories (儲存庫)。
4. 在 Repositories (儲存庫)頁面上，選擇要設定政策陳述式的儲存庫來檢視儲存庫內容。
5. 從儲存庫映像清單檢視的導覽窗格中，選擇 Permissions (許可)、Edit (編輯)。

### Note

如果您在導覽窗格中看不到 Permissions (許可) 選項，請確保您位於儲存庫映像清單檢視中。

6. 在 Edit permissions (編輯許可) 頁面上，選擇 Add statement (新增陳述式)。
7. 對於 Statement name (陳述式名稱)，輸入陳述式的名稱。
8. 對於 Effect (效果)，選擇會產生允許或明確拒絕的政策陳述式。
9. 對於 Principal (代理人)，請選擇要套用政策陳述式的使用者範圍。如需詳細資訊，請參閱《IAM 使用者指南》中的 [AWS IAM JSON 政策元素：條件](#)。
  - 您可以透過選取每個人 (\*) 核取方塊，將陳述式套用至所有已驗證 AWS 的使用者。
  - 對於 Service principal (服務委託人)，指定服務委託人名稱 (例如 `ecs.amazonaws.com`) 以套用陳述式到特定的服務。
  - 針對AWS 帳戶 IDs，指定 AWS 帳戶號碼 (例如 111122223333)，將陳述式套用至特定 AWS 帳戶下的所有使用者。您可以使用逗號分隔清單來指定多個帳戶。

**⚠ Important**

您授予許可的帳戶必須啟用您正在建立儲存庫政策的區域，否則會發生錯誤。

- 針對 IAM 實體，選取要套用陳述式之 AWS 帳戶下的角色或使用者。

**ℹ Note**

如需目前不支援的更複雜儲存庫政策 AWS 管理主控台，您可以使用 [set-repository-policy](#) AWS CLI 命令套用政策。

10. 對於 Actions (動作)，請選擇政策陳述式應從個別 API 操作清單中套用的 Amazon ECR API 操作範圍。
11. 當您完成時，請選擇 Save (儲存) 來設定政策。
12. 為要新增之每個儲存庫政策的重複之前的步驟。

## 在 Amazon ECR 中標記私有儲存庫

為了協助您管理 Amazon ECR 儲存庫，您可以使用 AWS 資源標籤，將自己的中繼資料指派給新的或現有的 Amazon ECR 儲存庫。例如，您可以為帳戶的 Amazon ECR 儲存庫定義一組標籤，協助您追蹤各個儲存庫的擁有者。

### 標籤基本概念

標籤對 Amazon ECR 來說不具有任何語意意義，並會嚴格解譯為字元字串。標籤不會自動指派給您的資源。您可以編輯標籤索引鍵和值，也可以隨時從資源中移除標籤。您可以將標籤的值設為空白字串，但您無法將標籤的值設為 Null。若您將與現有標籤具有相同鍵的標籤新增到該資源，則新值會覆寫舊值。如果您刪除資源，也會刪除任何該資源所使用的標籤。

您可以使用 Amazon ECR 主控台、AWS CLI 和 Amazon ECR API 來使用標籤。

使用 AWS Identity and Access Management (IAM)，您可以控制 AWS 帳戶中哪些使用者具有建立、編輯或刪除標籤的許可。如需 IAM 政策中標籤的資訊，請參閱 [the section called “使用標籤型存取控制”](#)。

## 標記您的資源以便計費

您新增至 Amazon ECR 儲存庫的標籤在成本與用量報告中啟用後，有助於檢視成本分配。如需詳細資訊，請參閱[Amazon ECR 用量報告](#)。

若想要查看合併資源的成本，您可根據具有相同標籤金鑰值的資源來整理您的帳單資訊。例如，您可以使用特定應用程式名稱來標記數個資源，然後整理帳單資訊以查看該應用程式跨數項服務的總成本。如需有關使用標籤設定成本分配報告的詳細資訊，請參閱《AWS Billing 使用者指南》中的[每月成本分配報告](#)。

### Note

若您才剛啟用報告，目前月份的資料會在 24 小時之後提供檢視。

## 將標籤新增至 Amazon ECR 中的私有儲存庫

您可以將標籤新增至私有儲存庫。

如需標籤名稱和最佳實務的相關資訊，請參閱《標記 AWS 資源使用者指南》中的[標籤命名限制和要求](#)以及[最佳實務](#)。

將標籤新增至儲存庫 (AWS 管理主控台)

1. 在 <https://console.aws.amazon.com/ecr/> 開啟 Amazon ECR 主控台。
2. 從導覽列中選取要使用的區域。
3. 在導覽窗格中，選擇 Repositories (儲存庫)。
4. 在儲存庫頁面上，選取要標記之儲存庫旁邊的核取方塊。
5. 從動作選單中選取儲存庫標籤。
6. 在儲存庫標籤頁面上，選取新增標籤、新增標籤。
7. 在編輯儲存庫標籤頁面上，指定每個標籤的索引鍵和值，然後選擇儲存。

將標籤新增至儲存庫 (AWS CLI 或 API)

您可以使用 或 AWS CLI API 新增或覆寫一或多個標籤。

- AWS CLI - [tag-resource](#)
- API 動作：[TagResource](#)

下列範例示範如何使用 新增標籤 AWS CLI。

#### 範例 1：標記儲存庫

下列命令會標記儲存庫。

```
aws ecr tag-resource \
 --resource-arn arn:aws:ecr:region:account_id:repository/repository_name \
 --tags Key=stack,Value=dev
```

#### 範例 2：使用多個標籤標記儲存庫

下列命令會將三個標籤新增至儲存庫。

```
aws ecr tag-resource \
 --resource-arn arn:aws:ecr:region:account_id:repository/repository_name \
 --tags Key=key1,Value=value1 Key=key2,Value=value2 Key=key3,Value=value3
```

#### 範例 3：列出儲存庫的標籤

下列命令會列出與儲存庫相關聯的標籤。

```
aws ecr list-tags-for-resource \
 --resource-arn arn:aws:ecr:region:account_id:repository/repository_name
```

#### 範例 4：建立儲存庫並新增標籤

以下命令建立名為 test-repo 的儲存庫，並新增索引鍵為 team 和值為 devs 的標籤。

```
aws ecr create-repository \
 --repository-name test-repo \
 --tags Key=team,Value=devs
```

## 從 Amazon ECR 中的私有儲存庫刪除標籤

您可以從私有儲存庫刪除標籤。

從私有儲存庫刪除標籤 (AWS 管理主控台)

1. 在 <https://console.aws.amazon.com/ecr/> 開啟 Amazon ECR 主控台。

2. 從導覽列中選取要使用的區域。
3. 在儲存庫頁面上，選取要移除標籤之儲存庫旁邊的核取方塊。
4. 從動作選單中選取儲存庫標籤。
5. 在儲存庫標籤頁面上，選取編輯。
6. 在編輯儲存庫標籤頁面上，針對您要刪除的每個標籤，選取移除圖示，然後選擇儲存。

### 從私有儲存庫刪除標籤 (AWS CLI)

您可以使用 `aws ecr untag-resource` 或 AWS CLI API 刪除一或多個標籤。

- AWS CLI - [untag-resource](#)
- API 動作 - [UntagResource](#)

下列範例顯示如何使用 `aws ecr untag-resource` 從儲存庫刪除標籤。

```
aws ecr untag-resource \
 --resource-arn arn:aws:ecr:region:account_id:repository/repository_name \
 --tag-keys tag_key
```

# Amazon ECR 中的私有映像

Amazon ECR 會將 Docker 映像、Open Container Initiative (OCI) 映像和 OCI 相容成品存放在私有儲存庫中。您可以使用 Docker CLI 或您偏好的用戶端來向您的儲存庫推送和提取映像。

透過 OCI v1.1 的 Amazon ECR 支援，您可以存放和管理 OCI [Referrers API](#) 定義的參考成品。成品包括簽章、軟體物料清單 (SBOMs)、Helm Chart、掃描結果和證明。容器映像的一組成品會與該容器一起傳輸，並儲存為單獨的映像，計算為儲存庫消耗的映像。

在 [Amazon ECR 中簽署映像](#) 和 [從 Amazon ECR 私有儲存庫刪除簽章和其他成品](#) 頁面提供如何使用簽章相關成品的範例。如需簽署容器映像的詳細資訊，請參閱《AWS Signer 開發人員指南》中的 [簽署容器映像](#)。

## 主題

- [將映像推送至 Amazon ECR 私有儲存庫](#)
- [從 Amazon ECR 私有儲存庫刪除簽章和其他成品](#)
- [在 Amazon ECR 中檢視影像詳細資訊](#)
- [從 Amazon ECR 私有儲存庫將映像提取到您的本機環境](#)
- [提取 Amazon Linux 容器映像](#)
- [在 Amazon ECR 中刪除映像](#)
- [在 Amazon ECR 中封存映像](#)
- [在 Amazon ECR 中重新標記映像](#)
- [防止在 Amazon ECR 中覆寫映像標籤](#)
- [Amazon ECR 中的容器映像資訊清單格式支援](#)
- [將 Amazon ECR 映像與 Amazon ECS 搭配使用](#)
- [將 Amazon ECR 映像與 Amazon EKS 搭配使用](#)

## 將映像推送至 Amazon ECR 私有儲存庫

您可以將 Docker 映像、資訊清單清單和開放容器計畫 (OCI) 映像和相容的成品推送至您的私有儲存庫。

Amazon ECR 提供將映像複寫到其他儲存庫的方法。透過在私有登錄檔設定中指定複寫組態，您可以在自己的登錄檔中跨區域複寫，並在不同的帳戶中複寫。如需詳細資訊，請參閱 [Amazon ECR 中的私有登錄檔設定](#)。

**Note**

如果您推送目前封存的映像，該映像會自動從封存中還原和移除。如需封存和還原映像的詳細資訊，請參閱[在 Amazon ECR 中封存映像](#)。

啟用登錄 Blob 掛載並包含掛載參數時，Amazon ECR 會在推送操作期間自動檢查登錄檔中的現有層。如果圖層已存在於相同登錄檔中的另一個儲存庫中，Amazon ECR 會掛載現有的圖層，而不是上傳重複的圖層。如需詳細資訊，請參閱[Amazon ECR 中的 Blob 掛載](#)。

**主題**

- [將映像推送至 Amazon ECR 私有儲存庫的 IAM 許可](#)
- [將 Docker 映像推送至 Amazon ECR 私有儲存庫](#)
- [將多架構映像推送至 Amazon ECR 私有儲存庫](#)
- [將 Helm Chart 推送至 Amazon ECR 私有儲存庫](#)

## 將映像推送至 Amazon ECR 私有儲存庫的 IAM 許可

使用者需要 IAM 許可，才能將映像推送至 Amazon ECR 私有儲存庫。遵循授予最低權限的最佳實務，您可以授予特定儲存庫的存取權。您也可以授予所有儲存庫的存取權。

使用者必須透過請求授權字符，向他們想要推送映像的每個 Amazon ECR 登錄檔進行身分驗證。Amazon ECR 提供多種 AWS 受管政策，以控制不同層級的使用者存取。如需詳細資訊，請參閱[AWS Amazon Elastic Container Registry 的受管政策](#)。

您也可以建立自己的 IAM 政策。下列 IAM 政策會授予將映像推送至特定儲存庫所需的許可。若要限制特定儲存庫的許可，請使用儲存庫的完整 Amazon Resource Name (ARN)。

**JSON**

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Effect": "Allow",
 "Action": [
 "ecr:CompleteLayerUpload",
```

```

 "ecr:UploadLayerPart",
 "ecr:InitiateLayerUpload",
 "ecr:BatchCheckLayerAvailability",
 "ecr:PutImage",
 "ecr:BatchGetImage"
],
 "Resource": "arn:aws:ecr:us-
east-1:111122223333:repository/repository-name"
 },
 {
 "Effect": "Allow",
 "Action": "ecr:GetAuthorizationToken",
 "Resource": "*"
 }
]
}

```

下列 IAM 政策會授予將映像推送至所有儲存庫所需的許可。

```

{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Effect": "Allow",
 "Action": [
 "ecr:CompleteLayerUpload",
 "ecr:GetAuthorizationToken",
 "ecr:UploadLayerPart",
 "ecr:InitiateLayerUpload",
 "ecr:BatchCheckLayerAvailability",
 "ecr:PutImage",
 "ecr:BatchGetImage"
],
 "Resource": "arn:aws:ecr:us-west-2:111122223333:repository/*"
 }
]
}

```

## 將 Docker 映像推送至 Amazon ECR 私有儲存庫

您可以使用 `docker push` 命令將容器映像推送到 Amazon ECR 儲存庫。

Amazon ECR 也支援建立和推送用於多架構映像的 Docker 資訊清單清單。如需相關資訊，請參閱[將多架構映像推送至 Amazon ECR 私有儲存庫](#)。

## 將 Docker 映像推送至 Amazon ECR 儲存庫

在推送映像之前，Amazon ECR 儲存庫必須存在，否則您必須定義儲存庫建立範本。如需詳細資訊，請參閱[建立 Amazon ECR 私有儲存庫以存放映像及用於控制提取快取、推送時建立或複寫動作期間建立的儲存庫的範本](#)。

1. 向打算推送映像的 Amazon ECR 登錄檔驗證您的 Docker 用戶端。所用的每個登錄檔皆必須取得身分驗證字符，字符有效期間為 12 個小時。如需詳細資訊，請參閱[Amazon ECR 中的私有登錄檔身分驗證](#)。

若要向 Amazon ECR 登錄檔驗證 Docker，請執行 `aws ecr get-login-password` 命令。將身分驗證字符傳遞給 `docker login` 命令時，使用 AWS 的值作為使用者名稱並指定您要驗證的 Amazon ECR 登錄檔 URI。如果是向多個登錄進行驗證，您必須針對每個登錄重複此命令。

### Important

若您收到錯誤，請安裝或升級至最新版本的 AWS CLI。如需詳細資訊，請參閱「AWS Command Line Interface 使用者指南」中的[安裝 AWS Command Line Interface](#)。

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

2. 如果您的映像儲存庫不存在於您打算推送的登錄檔中，且您已定義儲存庫建立範本，您可以使用儲存庫建立範本的字首和所需的儲存庫名稱來推送映像。ECR 會使用儲存庫建立範本的預先定義設定，自動為您建立儲存庫。

如果您沒有定義相符的儲存庫建立範本，則需要建立儲存庫。如需詳細資訊，請參閱[用於控制提取快取、推送時建立或複寫動作期間建立的儲存庫的範本](#) 或 [建立 Amazon ECR 私有儲存庫以存放映像](#)。

3. 找出要推送的本機映像。執行 `docker images` 命令，列出系統上的容器映像。

```
docker images
```

可用 `repository:tag` 值或映像 ID 從產生的命令輸出中找出映像。

- 在映像上標記要使用的 Amazon ECR 登錄檔、儲存庫和可選用的映像標籤名稱組合。登錄檔格式為 `aws_account_id.dkr.ecr.region.amazonaws.com`。儲存庫名稱應與您為映像建立的儲存庫名稱相符。如果省略映像標籤，系統將假設標籤為 `latest`。

下列範例會將 ID 為 `e9ae3c220b23` 的本機映像標記為 `aws_account_id.dkr.ecr.region.amazonaws.com/my-repository:tag`。

```
docker tag e9ae3c220b23 aws_account_id.dkr.ecr.region.amazonaws.com/my-repository:tag
```

- 使用 `docker push` 命令推送映像：

```
docker push aws_account_id.dkr.ecr.region.amazonaws.com/my-repository:tag
```

- (選用) 重覆執行 [Step 4](#) 和 [Step 5](#)，套用任何其他標籤至映像，並將標籤推送至 Amazon ECR。

## 將多架構映像推送至 Amazon ECR 私有儲存庫

您可以透過建立和推送 Docker 資訊清單清單，將多架構映像推送至 Amazon ECR 儲存庫。資訊清單列表是藉由指定一或多個映像名稱而建立的映像清單。在大多數情況下，資訊清單清單是從提供相同函數，但適用於不同作業系統或架構的映像建立的。資訊清單不是必要選項。如需詳細資訊，請參閱 [Docker 資訊清單](#)。

資訊清單列表可以像其他 Amazon ECR 映像一樣，在 Amazon ECS 任務定義或 Amazon EKS Pod 規格中提取或參考。

### 先決條件

- 在您的 Docker CLI 中，開啟實驗性功能。如需實驗功能的資訊，請參閱 Docker 文件中的 [實驗功能](#)。
- 在您推送映像之前，Amazon ECR 儲存庫必須存在。如需詳細資訊，請參閱 [the section called “建立儲存庫以存放映像”](#)。
- 您必須先將映像推送至儲存庫，才能建立 Docker 資訊清單。如需如何推送映像的資訊，請參閱 [將 Docker 映像推送至 Amazon ECR 私有儲存庫](#)。

## 將多架構 Docker 映像推送到 Amazon ECR 儲存庫

1. 向打算推送映像的 Amazon ECR 登錄檔驗證您的 Docker 用戶端。所用的每個登錄檔皆必須取得身分驗證字符，字符有效期間為 12 個小時。如需詳細資訊，請參閱[Amazon ECR 中的私有登錄檔身分驗證](#)。

若要向 Amazon ECR 登錄檔驗證 Docker，請執行 `aws ecr get-login-password` 命令。將身分驗證字符傳遞給 `docker login` 命令時，使用 AWS 的值作為使用者名稱並指定您要驗證的 Amazon ECR 登錄檔 URI。如果是向多個登錄進行驗證，您必須針對每個登錄重複此命令。

### Important

若您收到錯誤，請安裝或升級至最新版本的 AWS CLI。如需詳細資訊，請參閱《AWS Command Line Interface 使用者指南》中的[安裝 AWS Command Line Interface](#)。

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

2. 列出儲存庫中的映像，請確認映像標籤。

```
aws ecr describe-images --repository-name my-repository
```

3. 建立 Docker 資訊清單列表。manifest create 命令會驗證參考的映像是否已在您的儲存庫中，並在本機建立資訊清單。

```
docker manifest create aws_account_id.dkr.ecr.region.amazonaws.com/my-repository aws_account_id.dkr.ecr.region.amazonaws.com/my-repository:image_one_tag aws_account_id.dkr.ecr.region.amazonaws.com/my-repository:image_two
```

4. (選用) 檢查 Docker 資訊清單列表。這可讓您確認資訊清單列表中參照的每個映像資訊清單的大小和摘要。

```
docker manifest inspect aws_account_id.dkr.ecr.region.amazonaws.com/my-repository
```

5. 將 Docker 資訊清單列表推送至您的 Amazon ECR 儲存庫。

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

## 將 Helm Chart 推送至 Amazon ECR 私有儲存庫

您可以將開放容器計畫 (OCI) 成品推送至 Amazon ECR 儲存庫。若要查看此功能的範例，請使用下列步驟將 Helm Chart 推送至 Amazon ECR。

如需搭配 Amazon EKS 使用 Amazon ECR 託管 Helm Chart 的詳細資訊，請參閱 [在 Amazon EKS 叢集上安裝 Helm Chart](#)。

將 Helm Chart 推送到 Amazon ECR 儲存庫

1. 安裝 Helm 用戶端的最新版本。這些步驟是使用 Helm 版本 3.18.6 進行編寫。為了與 Amazon EKS 支援的 Kubernetes 版本相容，請使用 Helm 3.9 版或更新版本。如需詳細資訊，請參閱 [安裝 Helm](#)。
2. 使用下列步驟來建立測試 Helm Chart。如需詳細資訊，請參閱 [Helm Docs - 開始使用](#)。
  - a. 建立一個名為 `helm-test-chart` 的 Helm Chart 並清除 `templates` 目錄的內容。

```
helm create helm-test-chart
rm -rf ./helm-test-chart/templates/*
```

- b. ConfigMap 在 `templates` 資料夾中建立。

```
cd helm-test-chart/templates
cat <<EOF > configmap.yaml
apiVersion: v1
kind: ConfigMap
metadata:
 name: helm-test-chart-configmap
data:
 myvalue: "Hello World"
EOF
```

3. 封裝圖表。輸出將包含您在推送 Helm Chart 時使用的封裝圖表的檔案名稱。

```
cd ../..
helm package helm-test-chart
```

Output

```
Successfully packaged chart and saved it to: /Users/username/helm-test-chart-0.1.0.tgz
```

4. 建立儲存庫以存放 Helm Chart。儲存庫的名稱應與您在步驟 2 中建立 Helm Chart 時使用的名稱相符。如需詳細資訊，請參閱[建立 Amazon ECR 私有儲存庫以存放映像](#)。

```
aws ecr create-repository \
 --repository-name helm-test-chart \
 --region us-west-2
```

5. 將您的 Helm 用戶端驗證到您打算將 Helm Chart 推送到的 Amazon ECR 登錄檔。所用的每個登錄檔皆必須取得身分驗證字符，字符有效期間為 12 個小時。如需詳細資訊，請參閱[Amazon ECR 中的私有登錄檔身分驗證](#)。

```
aws ecr get-login-password \
 --region us-west-2 | helm registry login \
 --username AWS \
 --password-stdin aws_account_id.dkr.ecr.region.amazonaws.com
```

6. 使用 `helm push` 命令推送 Helm Chart。輸出應該包括 Amazon ECR 儲存庫 URI 和 SHA 摘要。

```
helm push helm-test-chart-0.1.0.tgz \
 oci://aws_account_id.dkr.ecr.region.amazonaws.com/
```

7. 描述您的 Helm Chart。

```
aws ecr describe-images \
 --repository-name helm-test-chart \
 --region us-west-2
```

在輸出中，確認 `artifactMediaType` 參數指出適當的成品類型。

```
{
 "imageDetails": [
 {
 "registryId": "aws_account_id",
 "repositoryName": "helm-test-chart",
 "imageDigest":
 "sha256:dd8aebdda7df991a0ffe0b3d6c0cf315fd582cd26f9755a347a52adEXAMPLE",
 "imageTags": [

```

```
 "0.1.0"
],
 "imageSizeInBytes": 1620,
 "imagePushedAt": "2021-09-23T11:39:30-05:00",
 "imageManifestMediaType": "application/vnd.oci.image.manifest.v1+json",
 "artifactMediaType": "application/vnd.cncf.helm.config.v1+json"
 }
]
}
```

8. (選用) 如需其他步驟，請安裝 Helm ConfigMap 並開始使用 Amazon EKS。如需詳細資訊，請參閱在 [Amazon EKS 叢集上安裝 Helm Chart](#)。

## 從 Amazon ECR 私有儲存庫刪除簽章和其他成品

您可以使用 ORAS 用戶端，從 Amazon ECR 私有儲存庫列出和刪除簽章和其他參考類型成品。刪除簽章和其他參考成品類似於刪除映像的方式（請參閱 [在 Amazon ECR 中刪除映像](#)）。以下是如何列出成品和刪除簽章的方法：

使用 ORAS CLI 管理影像成品

1. 安裝和設定 ORAS 用戶端。

如需有關安裝和設定 ORAS 用戶端的資訊，請參閱 ORAS 文件中的 [安裝](#)。

2. 若要列出 Amazon ECR 映像的可用成品，請使用 `oras discover`，後面接著映像名稱：

```
oras discover 111222333444.dkr.ecr.us-east-1.amazonaws.com/oci:helloworld
```

輸出看起來會與此類似：

```
111222333444.dkr.ecr.us-east-1.amazonaws.com/
oci@sha256:88c0c54329bfdc1d94d6f58cd3fcb1226d46f58670f44a8c689cb3c9b37b6925
application/vnd.cncf.notary.signature
sha256:387c10c1598ee18aae81dcfc86d0d06d116e46461d1c3cda8927e69c48108c42
sha256:6527bcec87adf1d55460666183b9d0968b3cd4e4bc34602d485206a219851171
```

3. 若要使用 ORAS CLI 刪除簽章，根據先前的範例，請執行下列命令：

```
oras manifest delete 111222333444.dkr.ecr.us-east-1.amazonaws.com/
oci@sha256:387c10c1598ee18aae81dcfc86d0d06d116e46461d1c3cda8927e69c48108c42
```

輸出看起來會與此類似：

```
Are you sure you want to delete the manifest "111222333444.dkr.ecr.us-
east-1.amazonaws.com/
oci@sha256:387c10c1598ee18aae81dcfc86d0d06d116e46461d1c3cda8927e69c48108c42" and
all tags associated with it? [y/N] y
```

4. 按 **y** 鍵。應刪除成品。

### 對成品刪除進行疑難排解

如果簽章刪除，例如剛才顯示的簽章刪除應該失敗，則會顯示類似以下的輸出。

```
Error response from registry: failed to delete 111222333444.dkr.ecr.us-
east-1.amazonaws.com/
oci@sha256:387c10c1598ee18aae81dcfc86d0d06d116e46461d1c3cda8927e69c48108c42:
unsupported: Requested image referenced by manifest list:
[sha256:005e2c97a6373e483799fa4ff29ac64a42dd10f08efcc166d6775f9b74943b5b]
```

刪除在 OCI 1.1 啟動之前推送的映像時，可能會發生此失敗。如錯誤中所述，您必須先刪除參考映像的資訊清單，才能刪除映像，如下所示：

1. 若要刪除與要刪除的簽章相關聯的資訊清單，請輸入：

```
oras manifest delete 111222333444.dkr.ecr.us-east-1.amazonaws.com/
oci@sha256:005e2c97a6373e483799fa4ff29ac64a42dd10f08efcc166d6775f9b74943b5b
```

輸出看起來會與此類似：

```
Are you sure you want to delete the manifest
"sha256:005e2c97a6373e483799fa4ff29ac64a42dd10f08efcc166d6775f9b74943b5b" and all
tags associated with it? [y/N] y
```

- 
2. 按 `y` 鍵。應刪除資訊清單。
3. 當資訊清單消失時，您可以刪除簽章：

```
oras manifest delete 111222333444.dkr.ecr.us-east-1.amazonaws.com/
oci@sha256:387c10c1598ee18aae81dcfc86d0d06d116e46461d1c3cda8927e69c48108c42
```

輸出看起來應該像這樣。按 `y` 鍵。

```
Are you sure you want to delete the manifest
"sha256:387c10c1598ee18aae81dcfc86d0d06d116e46461d1c3cda8927e69c48108c42" and all
tags associated with it? [y/N] y
Deleted [registry] 111222333444.dkr.ecr.us-east-1.amazonaws.com/
oci@sha256:387c10c1598ee18aae81dcfc86d0d06d116e46461d1c3cda8927e69c48108c42
```

- 
- 
- 
4. 若要查看簽章已刪除，請輸入：

```
oras discover 111222333444.dkr.ecr.us-east-1.amazonaws.com/oci:helloworld
```

輸出看起來會與此類似：

```
111222333444.dkr.ecr.us-east-1.amazonaws.com/
oci@sha256:88c0c54329bfdc1d94d6f58cd3fcb1226d46f58670f44a8c689cb3c9b37b6925
application/vnd.cncf.notary.signature
sha256:6527bcec87adf1d55460666183b9d0968b3cd4e4bc34602d485206a219851171
```

## 在 Amazon ECR 中檢視影像詳細資訊

將映像推送至儲存庫後，您可以檢視其相關資訊。包含的詳細資訊如下：

- 映像 URI
- 映像標籤
- 成品媒體類型

- 映像資訊清單類型
- 掃描狀態
- 映像大小 (以 MB 為單位)
- 將映像推送至儲存庫的時間
- 複寫狀態

### 檢視映像詳細資訊 (AWS 管理主控台)

1. 在 <https://console.aws.amazon.com/ecr/repositories> 開啟 Amazon ECR 主控台。
2. 從導覽列上，選擇包含映像的儲存庫的區域。
3. 在導覽窗格中，依序選擇私有登錄檔和儲存庫。
4. 在私有儲存庫頁面上，選擇要檢視的儲存庫。
5. 在 Repositories : **repository\_name** (儲存庫 : repository\_name) 頁面上，選擇要檢視的詳細資訊的映像。

## 從 Amazon ECR 私有儲存庫將映像提取到您的本機環境

如果您要執行在 Amazon ECR 中可用的 Docker 映像，您可以使用 `docker pull` 命令來將其提取至本機環境。您可以從預設登錄檔或與另一個 AWS 帳戶相關聯的登錄檔執行此操作。

若要在 Amazon ECS 任務定義中使用 Amazon ECR 映像，請參閱 [將 Amazon ECR 映像與 Amazon ECS 搭配使用](#)。

#### Important

您無法提取封存的映像。封存的映像必須先還原，才能提取。如需封存和還原映像的詳細資訊，請參閱 [在 Amazon ECR 中封存映像](#)。

#### Important

Amazon ECR 要求使用者擁有透過 IAM 政策呼叫 `ecr:GetAuthorizationToken` API 的許可，然後才能對登錄檔進行身分驗證，並從任何 Amazon ECR 儲存庫推送或提取任何映像。Amazon ECR 提供多種 AWS 受管政策，以控制不同層級的使用者存取。如需 Amazon

ECR AWS 受管政策的相關資訊，請參閱 [AWS Amazon Elastic Container Registry 的受管政策](#)。

## 從 Amazon ECR 儲存庫提取 Docker 映像

1. 向打算提取映像的 Amazon ECR 登錄檔驗證您的 Docker 用戶端。所用的每個登錄檔皆必須取得身分驗證字符，字符有效期間為 12 個小時。如需詳細資訊，請參閱 [Amazon ECR 中的私有登錄檔身分驗證](#)。
2. (選用) 找出要提取的映像。
  - 可用 `aws ecr describe-repositories` 命令列出登錄檔內的儲存庫：

```
aws ecr describe-repositories
```

上述登錄檔範例有一個名為 `amazonlinux` 的儲存庫。

- 可用 `aws ecr describe-images` 命令描述儲存庫內的映像：

```
aws ecr describe-images --repository-name amazonlinux
```

上述的儲存庫範例有標記為 `latest` 和 `2016.09`，映像摘要為 `sha256:f1d4ae3f7261a72e98c6ebefe9985cf10a0ea5bd762585a43e0700ed99863807` 的映像。

3. 使用 `docker pull` 命令提取映像。映像名稱格式應為 `registry/repository[:tag]` 才可依標籤提取，或為 `registry/repository[@digest]` 才可依摘要提取。

```
docker pull aws_account_id.dkr.ecr.us-west-2.amazonaws.com/amazonlinux:latest
```

### Important

如果出現 `repository-url not found: does not exist or no pull access` 錯誤，則需向 Amazon ECR 驗證您的 Docker 用戶端。如需詳細資訊，請參閱 [Amazon ECR 中的私有登錄檔身分驗證](#)。

## 提取 Amazon Linux 容器映像

Amazon Linux 容器映像是透過在 Amazon Linux AMI 中包含的相同軟體元件所建置。Amazon Linux 容器映像可在任何環境中使用，做為 Docker 工作負載的基礎映像。如果您將 Amazon Linux AMI 用於 Amazon EC2 中的應用程式，您可以使用 Amazon Linux 容器映像來容器化應用程式。

您可以在本機開發環境中使用 Amazon Linux 容器映像，然後使用 Amazon ECS 將應用程式推送至 AWS。如需詳細資訊，請參閱[將 Amazon ECR 映像與 Amazon ECS 搭配使用](#)。

Amazon Linux 容器映像可在 Amazon ECR Public 及 [Docker Hub](#) 使用。如需 Amazon Linux 容器映像的支援，請前往[AWS 開發人員論壇](#)。

從 Amazon ECR Public 中提取 Amazon Linux 容器映像

1. 向 Amazon Linux Public 登錄檔驗證您的 Docker 用戶端。驗證字符有效時間為 12 小時。如需詳細資訊，請參閱[Amazon ECR 中的私有登錄檔身分驗證](#)。

### Note

從版本 1.18.1.187 開始，可以在 AWS CLI 中使用 `ecr-public` 命令，但我們建議使用最新版的 AWS CLI。如需詳細資訊，請參閱《AWS Command Line Interface 使用者指南》中的[安裝 AWS Command Line Interface](#)。

```
aws ecr-public get-login-password --region us-east-1 | docker login --username AWS --password-stdin public.ecr.aws
```

其輸出如下：

```
Login succeeded
```

2. 使用 `docker pull` 命令提取 Amazon Linux 容器映像。若要在 Amazon ECR Public Gallery 上查看 Amazon Linux 容器映像，請參閱 [Amazon ECR Public Gallery - amazonlinux](#)。

```
docker pull public.ecr.aws/amazonlinux/amazonlinux:latest
```

3. (選用) 在本機執行容器。

```
docker run -it public.ecr.aws/amazonlinux/amazonlinux /bin/bash
```

## 從 Docker Hub 提取 Amazon Linux 容器映像

1. 使用 `docker pull` 命令提取 Amazon Linux 容器映像。

```
docker pull amazonlinux
```

2. (選用) 在本機執行容器。

```
docker run -it amazonlinux:latest /bin/bash
```

## 在 Amazon ECR 中刪除映像

如果您已完成使用映像，即可將其從儲存庫中刪除。如果您完成使用儲存庫，您可以刪除整個儲存庫及其中的所有映像。如需詳細資訊，請參閱[在 Amazon ECR 中刪除私有儲存庫](#)。

作為手動刪除映像的替代方法，您可以建立儲存庫生命週期政策，以更好地控制儲存庫中映像的生命週期管理。生命週期政策會為您自動執行此程序。如需詳細資訊，請參閱[在 Amazon ECR 中使用生命週期政策來自動化映像的清除](#)。

### Note

如果您的儲存庫混合了映像，其中一些會在 Amazon ECR 支援的 OCI 1.1 版之前推送，則某些簽章會有指向它們的映像索引或資訊清單清單。因此，當您刪除預先 OCI v1.1 映像時，您可能需要手動刪除參考映像的資訊清單清單，才能刪除成品。

### 刪除映像 (AWS 管理主控台)

1. 在 <https://console.aws.amazon.com/ecr/repositories> 開啟 Amazon ECR 主控台。
2. 從導覽列上，選擇包含要刪除之映像的區域。
3. 在導覽窗格中，選擇 Repositories (儲存庫)。
4. 在 Repositories (儲存庫) 頁面上，選擇包含要刪除之映像的儲存庫。
5. 在儲存庫：*positorage\_name* 頁面上，選取要刪除之影像左側的方塊，然後選擇刪除。
6. 在 Delete image(s) (刪除映像) 對話方塊中，確認應刪除的映像已被選擇，再選擇 Delete (刪除)。

## 刪除映像 (AWS CLI)

1. 在儲存庫中列出映像。標籤映像會同時具有映像摘要以及相關標籤的清單。未標籤的映像只會有映像摘要。

```
aws ecr list-images \
 --repository-name my-repo
```

2. (選用) 透過指定與要刪除的映像關聯的標籤來刪除映像的任何不需要的標籤。當從映像中刪除最後一個標籤時，映像也會遭到刪除。

```
aws ecr batch-delete-image \
 --repository-name my-repo \
 --image-ids imageTag=tag1 imageTag=tag2
```

3. 指定映像摘要以刪除標籤或未標籤的映像。透過參照摘要的方式刪除映像時，該映像及其所有標籤將遭刪除。

```
aws ecr batch-delete-image \
 --repository-name my-repo \
 --image-ids imageDigest=sha256:4f70ef7a4d29e8c0c302b13e25962d8f7a0bd304EXAMPLE
```

若要刪除多個映像，您可以在請求中指定多個映像標籤或映像摘要。

```
aws ecr batch-delete-image \
 --repository-name my-repo \
 --image-ids imageDigest=sha256:4f70ef7a4d29e8c0c302b13e25962d8f7a0bd304EXAMPLE
 imageDigest=sha256:f5t0e245ssffc302b13e25962d8f7a0bd304EXAMPLE
```

## 在 Amazon ECR 中封存映像

### 什麼是 ECR 封存儲存類別？

Amazon ECR 封存儲存類別是一種新的儲存類別，可為容器映像提供低成本、長期的儲存。Amazon ECR 提供兩種儲存類別：

- ECR 標準儲存類別 – 定期存取之作用中映像的預設儲存類別。

- ECR 封存儲存類別 – 很少存取但需要保留以進行合規或長期參考之映像的低成本儲存類別。與標準儲存體方案相比，封存儲存體方案可為大量映像節省長期映像保留的成本。如需詳細的定價資訊，請參閱 [Amazon ECR 定價](#)。

若要封存映像，您有兩個選項。首先，您可以設定生命週期規則，根據下列項目自動封存映像：

- 自推送映像以來的時間
- 自上次提取映像以來的時間
- 儲存庫中的映像數量

您也可以將設定設定為在封存影像一段指定期間之後將其永久刪除。請參閱 [在 Amazon ECR 中使用生命週期政策來自動化映像的清除](#)，了解詳細資訊。

您也可以使用 Amazon ECR 主控台或 封存映像 AWS CLI。請參閱 [封存映像](#)，了解詳細資訊。

當您需要再次使用封存的映像時，您可以將其還原至 ECR 標準儲存類別。您可以預期 ECR 會在 20 分鐘內還原映像。還原的映像的行為類似於新推送的映像，可在還原完成時立即使用。還原的映像受限於掃描、複寫和儲存庫生命週期政策。請參閱 [還原映像](#)，了解詳細資訊。

## 封存映像

您可以使用 Amazon ECR 主控台或 手動封存映像 AWS CLI，或使用生命週期政策自動封存映像。封存映像時：

- 影像會移至封存儲存體方案。
- 封存的映像無法提取。提取封存映像的請求將會失敗，並顯示 404 錯誤。
- 雖然無法提取映像，但仍然可以使用 `describe-images` 命令加以說明，或使用 `list-images` 命令列出。影像狀態會顯示為 ARCHIVED。
- 封存映像的最短儲存期間為 90 天。您無法設定生命週期政策來刪除已存檔不到 90 天的映像。如果您必須刪除已封存不到 90 天的映像，則需要使用 `batch-delete-image` API，但需支付 90 天最低儲存持續時間的費用。
- 映像會出現在儲存庫檢視的封存映像索引標籤中（只有在儲存庫中至少封存一個映像時，才會顯示此索引標籤）。
- 您可以透過手動選取要還原的映像，或將映像重新推送至儲存庫，將映像還原為作用中映像。
- 如果儲存庫具有生命週期政策來刪除具有封存時間等條件的映像，則會刪除映像。

## AWS 管理主控台

### 封存映像

1. 在 <https://console.aws.amazon.com/ecr/repositories> 開啟 Amazon ECR 主控台。
2. 從導覽列中，選擇包含儲存庫的區域，其中包含您要封存的映像。
3. 在導覽窗格中，選擇 Repositories (儲存庫)。
4. 在儲存庫頁面上，選擇包含您要封存之映像的儲存庫。
5. 選取您要存檔的映像。您將看到影像詳細資訊。
6. 若要封存映像，請選取封存按鈕，然後選取出現提示時確認。
7. 如果這是儲存庫中的第一個封存映像，則新封存映像索引標籤會顯示新封存映像。如果有其他封存的映像，此映像會新增至該索引標籤。

## AWS CLI

### 封存映像

- 使用 `update-image-storage-class` 命令來封存映像，方法是將其儲存類別更新為 ARCHIVE：

```
aws ecr update-image-storage-class \
 --repository-name my-repository \
 --image-id
 imageDigest=sha256:4f70ef7a4d29e8c0c302b13e25962d8f7a0bd304EXAMPLE \
 --target-storage-class ARCHIVE
```

### 使用生命週期政策封存映像

- 您可以使用生命週期政策來設定儲存庫的封存規則，以自動封存映像。生命週期政策可讓您根據下列條件自動封存映像：
  - 自推送映像以來的時間
  - 自上次提取映像以來的時間
  - 保持作用中的影像數量上限

您也可以設定生命週期政策，在映像封存一段指定的期間之後將其永久刪除。如需使用封存動作之生命週期政策的詳細資訊和範例，請參閱 [在 Amazon ECR 中使用生命週期政策來自動化映像的清除](#)。

#### Note

封存映像的最短儲存期間為 90 天。您無法設定生命週期政策來刪除已存檔不到 90 天的映像。如果您必須刪除已封存不到 90 天的映像，則需要使用 `batch-delete-image` API，但需支付 90 天最低儲存持續時間的費用。

當您使用 `describe-images` 命令描述映像時，封存的映像具有 `image-status` 的 `ARCHIVED`。您可以依篩選影像 `image-status`，僅檢視封存的影像或作用中的影像。

## 還原映像

當您還原封存的映像時，它會從 ECR Archive 儲存類別移回 ECR Standard 儲存類別。還原的映像會依標準儲存費率收費。還原程序會執行建立新映像時發生的類似動作：

- 當還原完成時，映像可供提取。還原通常需要 20 分鐘的時間，但可能會更快完成。
- 如果為儲存庫啟用推送時掃描，則會掃描還原的影像。請注意，在封存映像之前，之前的掃描結果將無法使用。
- 如果已為儲存庫設定複寫，則在還原時啟用複寫時，將會複寫還原的影像。
- 還原的映像會出現在作用中映像清單中。

還原映像通常需要 20 分鐘，但完成速度可能會更快。在還原程序期間，映像會維持在封存狀態，而且在還原完成之前無法提取。

### AWS 管理主控台

#### 還原封存的映像

1. 在 <https://console.aws.amazon.com/ecr/repositories> 開啟 Amazon ECR 主控台。
2. 從導覽列中，選擇包含儲存庫的區域，其中包含您要還原的封存映像。
3. 在導覽窗格中，選擇 Repositories (儲存庫)。
4. 在儲存庫頁面上，選擇包含封存映像的儲存庫。

5. 選擇封存映像索引標籤。
6. 選取您要還原的封存映像。
7. 選擇還原並確認還原動作。
8. 等待還原完成。還原完成後，映像會出現在作用中映像清單中。

## AWS CLI

### 還原封存的映像

- 使用 `update-image-storage-class` 命令將封存的映像更新為 `STANDARD`，以還原其儲存類別 `STANDARD`：

```
aws ecr update-image-storage-class \
 --repository-name my-repository \
 --image-id
 imageDigest=sha256:4f70ef7a4d29e8c0c302b13e25962d8f7a0bd304EXAMPLE \
 --target-storage-class STANDARD
```

當您使用 `describe-images` 命令描述映像時，正在還原的映像具有 `image-status` 的 `ACTIVATING`。您可以使用 `image-status` 值篩選映像 `ACTIVATING`，以檢視目前正在還原的映像。

還原封存映像的替代方法是將映像重新推送至儲存庫。當您推送目前封存的映像時，該映像會立即還原並從封存中移除。

## 在 Amazon ECR 中重新標記映像

透過 Docker 映像資訊清單 V2 結構描述 2 映像，您可以使用 `--image-tag` 命令的 `put-image` 選項以重新標記現有映像。您可以使用 Docker 來重新標記，而不需提取和推送映像。針對較大的映像，此程序節省了重新標記映像所需的大量網路頻寬與時間。

### 重新標記映像 (AWS CLI)

#### 使用 重新標記映像 AWS CLI

1. 使用 `batch-get-image` 命令以取得映像的映像資訊清單，來重新標記並將其寫入檔案。在此範例中，儲存庫 `amazonlinux` 中擁有 `latest` 標籤的映像資訊清單，被寫入名為 `MANIFEST` 的環境變數。

```
MANIFEST=$(aws ecr batch-get-image --repository-name amazonlinux --image-ids
imageTag=latest --output text --query 'images[].imageManifest')
```

2. 使用 `put-image` 命令的 `--image-tag` 選項，以新標籤將映像工作資訊清單檔案放至 Amazon ECR。在此範例中，該映像被標記為 **2017.03**。

#### Note

如果您的版本無法使用 `--image-tag` 選項 AWS CLI，請升級到最新版本。如需詳細資訊，請參閱《AWS Command Line Interface 使用者指南》中的 [安裝 AWS Command Line Interface](#)。

```
aws ecr put-image --repository-name amazonlinux --image-tag 2017.03 --image-
manifest "$MANIFEST"
```

3. 確認您的新映像標籤已連接至您的映像。在下方的輸出中，該映像有標籤 `latest` 與 `2017.03`。

```
aws ecr describe-images --repository-name amazonlinux
```

其輸出如下：

```
{
 "imageDetails": [
 {
 "imageSizeInBytes": 98755613,
 "imageDigest":
"sha256:8d00af8f076eb15a33019c2a3e7f1f655375681c4e5be157a26EXAMPLE",
 "imageTags": [
 "latest",
 "2017.03"
],
 "registryId": "aws_account_id",
 "repositoryName": "amazonlinux",
 "imagePushedAt": 1499287667.0
 }
]
}
```

## 重新標記映像 (AWS Tools for Windows PowerShell)

### 使用 重新標記映像 AWS Tools for Windows PowerShell

1. 使用 `Get-ECRIImageBatchcmdlet`取得映像的描述，以重新標記並寫入環境變數。在此範例中，儲存庫 `amazonlinux` 中具有 `##` 標籤的映像會寫入環境變數 `$Image`。

#### Note

如果您的系統沒有 `Get-ECRIImageBatch cmdlet` 可用的，請參閱 [AWS Tools for PowerShell 《使用者指南》](#) 中的 [設定 AWS Tools for Windows PowerShell](#)。

```
$Image = Get-ECRIImageBatch -ImageId @{ imageTag="latest" } -
RepositoryName amazonlinux
```

2. 將映像的資訊清單寫入 `$Manifest` 環境變數。

```
$Manifest = $Image.Images[0].ImageManifest
```

3. 使用 `-ImageTag`的選項，將映像資訊清單 `Write-ECRIImage cmdlet`放入具有新標籤的 Amazon ECR。在此範例中，該映像被標記為 `2017.09`。

```
Write-ECRIImage -RepositoryName amazonlinux -ImageManifest $Manifest -
ImageTag 2017.09
```

4. 確認您的新映像標籤已連接至您的映像。在下方的輸出中，該映像有標籤 `latest` 與 `2017.09`。

```
Get-ECRIImage -RepositoryName amazonlinux
```

其輸出如下：

```
ImageDigest ImageTag
----- -
sha256:359b948ea8866817e94765822787cd482279eed0c17bc674a7707f4256d5d497 latest
sha256:359b948ea8866817e94765822787cd482279eed0c17bc674a7707f4256d5d497 2017.09
```

## 防止在 Amazon ECR 中覆寫映像標籤

您可以在儲存庫中開啟標籤不可變性，以防止映像標籤遭到覆寫。開啟標籤不可變性後，如果您推送的映像具有已在儲存庫中的標籤，則會傳回 `ImageTagAlreadyExistsException` 錯誤。標籤不可變性會影響所有標籤。您無法讓某些標籤不可變，而其他標籤則不可變。

您可以使用 AWS 管理主控台 和 AWS CLI 工具來設定新儲存庫或現有儲存庫的影像標籤可變性。若要使用主控台步驟建立儲存庫，請參閱 [建立 Amazon ECR 私有儲存庫以存放映像](#)。

### 設定影像標籤可變性 (AWS 管理主控台)

#### 設定影像標籤可變性

1. 在 <https://console.aws.amazon.com/ecr/repositories> 開啟 Amazon ECR 主控台。
2. 從導覽列上，選擇包含要編輯之儲存庫的區域。
3. 在導覽窗格中，選擇私有登錄檔下的儲存庫。

如果您沒有看到儲存庫，請選擇私有登錄檔以展開選單，然後選擇儲存庫。

4. 在私有儲存庫頁面上，選擇您要設定影像標籤可變性設定的儲存庫名稱前面的選項按鈕。
5. 選擇動作，然後在編輯下選擇儲存庫。
6. 針對影像標籤不可變性，選擇下列其中一個儲存庫的標籤可變性設定。
  - 可互斥 – 如果您想要覆寫影像標籤，請選擇此選項。這是使用透過快取提取動作的儲存庫的建議設定，確保 Amazon ECR 能夠更新快取的映像。此外，若要停用幾個可變標籤的標籤更新，請輸入標籤名稱或使用萬用字元 (\*) 來比對 Mutable 標籤排除文字方塊中的多個類似標籤。
  - 不可變：如果您想要防止映像標籤遭到覆寫，而且在推送具有現有標籤的映像時，它適用於儲存庫中的所有標籤和排除項目，請選擇此選項。如果您嘗試推送具有現有標籤的映像，Amazon ECR 會傳回 `ImageTagAlreadyExistsException`。此外，若要啟用幾個不可變標籤的標籤更新，請輸入標籤名稱或使用萬用字元 (\*) 在不可分割標籤排除文字方塊中比對多個類似的標籤。
7. 對於 Image scan settings (映像掃描設定)，雖然您可以在儲存庫層級指定基本掃描的掃描設定，但最佳實務是在私有登錄檔層級指定掃描組態。在私有登錄檔層級指定掃描設定讓你能啟用增強型掃描或基本掃描，還能定義篩選條件來指定要掃描哪些儲存庫。如需詳細資訊，請參閱[掃描映像是否有 Amazon ECR 中的軟體漏洞](#)。
8. 對於 Encryption settings (加密設定)，這是一個僅能檢視的欄位，因為儲存庫建立後，就無法變更儲存庫的加密設定。

## 9. 選擇 Save (儲存) 更新儲存庫設定。

### 設定影像標籤可變性 (AWS CLI)

建立儲存庫並設定不可變標籤

使用以下其中一個命令來建立新的映像儲存庫，並設定不可變標籤。

- [create-repository](#) (AWS CLI) 具有影像標籤可變性

```
aws ecr create-repository --repository-name name --image-tag-mutability IMMUTABLE --region us-east-2
```

- [create-repository](#) (AWS CLI) 搭配影像標籤可變性排除篩選條件

```
aws ecr create-repository --repository-name name --image-tag-mutability IMMUTABLE_WITH_EXCLUSION --image-tag-mutability-exclusion-filters filterType=WILDCARD,filter=filter-text --region us-east-2
```

- 具有映像標籤可變性的 [New-ECRRepository](#) (AWS Tools for Windows PowerShell)

```
New-ECRRepository -RepositoryName name -ImageTagMutability IMMUTABLE -Region us-east-2 -Force
```

- 具有影像標籤可變性排除篩選條件的 [New-ECRRepository](#) (AWS Tools for Windows PowerShell)

```
New-ECRRepository -RepositoryName name -ImageTagMutability IMMUTABLE_WITH_EXCLUSION -ImageTagMutabilityExclusionFilter @{FilterType=WILDCARD Filter=filter-text} -Region us-east-2 -Force
```

更新儲存庫的映像標籤可變性設定

使用以下其中一個命令來更新現有儲存庫的映像標籤可變性設定。

- [put-image-tag-mutability](#) (AWS CLI) 具有影像標籤可變性

```
aws ecr put-image-tag-mutability --repository-name name --image-tag-mutability IMMUTABLE --region us-east-2
```

- [put-image-tag-mutability](#) (AWS CLI) 搭配影像標籤可變性排除篩選條件

```
aws ecr put-image-tag-mutability --repository-name name --image-tag-mutability IMMUTABLE_WITH_EXCLUSION --image-tag-mutability-exclusion-filters filterType=WILDCARD,filter=latest --region us-east-2
```

- [Write-ECRImageTagMutability](#) (AWS Tools for Windows PowerShell) 具有影像標籤可變性

```
Write-ECRImageTagMutability -RepositoryName name -ImageTagMutability IMMUTABLE -Region us-east-2 -Force
```

- [Write-ECRImageTagMutability](#) (AWS Tools for Windows PowerShell) 搭配影像標籤可變性排除篩選條件

```
Write-ECRImageTagMutability -RepositoryName name -ImageTagMutability IMMUTABLE_WITH_EXCLUSION -ImageTagMutabilityExclusionFilter @{FilterType=WILDCARD Filter=latest}
```

## Amazon ECR 中的容器映像資訊清單格式支援

Amazon ECR 支援以下的容器映像資訊清單格式：

- Docker 映像資訊清單 V2 結構描述 1 (需搭配 1.9 或更舊版本的 Docker 使用)
- Docker 映像資訊清單 V2 結構描述 2 (需搭配 1.10 或更新版本的 Docker 使用)
- Open Container Initiative (OCI) 規格 (v1.0 和 v1.1)

Docker 映像資訊清單 V2 結構描述 2 支援提供下列功能：

- 可在單數映像上使用多重標籤。
- 支援儲存 Windows 容器映像。

## Amazon ECR 映像資訊清單轉換

推送映像至 Amazon ECR 及提取映像時，容器引擎用戶端 (例如 Docker) 會與登錄檔通訊，以取得一致同意用戶端支援且供登錄檔用於映像的資訊清單格式。

使用 Docker 1.9 或更早版本推送映像至 Amazon ECR 時，映像的資訊清單格式將存放為 Docker 映像資訊清單 V2 結構描述 1。使用 Docker 1.10 或更新版本推送映像至 Amazon ECR 時，映像的資訊清單格式將存放為 Docker 映像資訊清單 V2 結構描述 2。

當您依標籤從 Amazon ECR 提取映像時，Amazon ECR 會傳回存放在儲存庫中的映像資訊清單格式。只有在用戶端支援該格式，才會傳回該格式。如果客戶端不理解存放的映像資訊清單格式，Amazon ECR 會將映像資訊清單轉換為可以理解的格式。例如，如果 Docker 1.9 用戶端請求以 Docker 映像資訊清單 V2 結構描述 2 形式存放的映像資訊清單，Amazon ECR 會傳回在 Docker 映像資訊清單 V2 結構描述 1 格式中的資訊清單。下表描述透過依標籤提取映像時，Amazon ECR 支援的可用轉換：

| 用戶端要求的結構描述 | 以 V2 結構描述 1 推送至 ECR   | 以 V2 結構描述 2 推送至 ECR | 以 OCI 推送至 ECR |
|------------|-----------------------|---------------------|---------------|
| V2 結構描述 1  | 不需翻譯                  | 翻譯為 V2 結構描述 1       | 沒有可用的翻譯       |
| V2 結構描述 2  | 不需翻譯，用戶端回退至 V2 結構描述 1 | 不需翻譯                | 翻譯為 V2 結構描述 2 |
| OCI        | 沒有可用的翻譯               | 翻譯為 OCI             | 不需翻譯          |

### Important

如果您依摘要提取映像，則沒有可用的翻譯。用戶端必須瞭解存放在 Amazon ECR 中的映像資訊清單格式。如果在 Docker 1.9 或更舊版本用戶端上依摘要要求 Docker 映像資訊清單 V2 結構描述 2 映像，映像提取將失敗。如需詳細資訊，請參閱 Docker 文件中的「[登錄檔相容性](#)」。

在此範例中，如果您依標籤要求相同的映像，Amazon ECR 則會將映像資訊清單翻譯為用戶端支援的格式。映像會提取成功。

## 將 Amazon ECR 映像與 Amazon ECS 搭配使用

您可以使用 Amazon ECR 私有儲存庫，來託管 Amazon ECS 任務可能從中提取的容器映像和成品。若要使用此功能，Amazon ECS 或 Fargate、容器代理程式必須具有進行 `ecr:BatchGetImage`、`ecr:GetDownloadUrlForLayer` 和 `ecr:GetAuthorizationToken` API 的許可。

## 所需的 IAM 許可

下表顯示了要針對每個啟動類型使用的 IAM 角色，這些角色為您的任務提供從 Amazon ECR 私有儲存庫提取所需的許可。Amazon ECS 提供受管 IAM 政策，其中包含必要許可。

| 啟動類型                         | IAM 角色                                                                                                                                                   | AWS 受管 IAM 政策                                                                                                                                          |
|------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------|
| Amazon EC2 執行個體上的 Amazon ECS | 使用容器執行個體 IAM 角色，此角色與 Amazon ECS 叢集中註冊的 Amazon EC2 執行個體相關聯。如需詳細資訊，請參閱《Amazon Elastic Container Service 開發人員指南》中的 <a href="#">容器執行個體 IAM 角色</a> 。          | AmazonEC2ContainerServiceforEC2Role<br><br>如需詳細資訊，請參閱《Amazon Elastic Container Service 開發人員指南》中的 <a href="#">AmazonEC2ContainerServiceforEC2Role</a> 。 |
| Fargate 上的 Amazon ECS        | 使用您在 Amazon ECS 任務定義中參照的任務執行 IAM 角色。如需詳細資訊，請參閱《Amazon Elastic Container Service 開發人員指南》中的 <a href="#">任務執行 IAM 角色</a> 。                                  | AmazonECSTaskExecutionRolePolicy<br><br>如需詳細資訊，請參閱《Amazon Elastic Container Service 開發人員指南》中的 <a href="#">AmazonECSTaskExecutionRolePolicy</a> 。       |
| 外部執行個體上的 Amazon ECS          | 使用容器執行個體 IAM 角色，此角色與 Amazon ECS 叢集中註冊的內部部署伺服器或虛擬機器 (VM) 相關聯。如需詳細資訊，請參閱《Amazon Elastic Container Service 開發人員指南》中的 <a href="#">容器執行個體 Amazon ECS 角色</a> 。 | AmazonEC2ContainerServiceforEC2Role<br><br>如需詳細資訊，請參閱《Amazon Elastic Container Service 開發人員指南》中的 <a href="#">AmazonEC2ContainerServiceforEC2Role</a> 。 |

**⚠ Important**

AWS 受管 IAM 政策包含您使用時可能不需要的其他許可。在此情況下，這些是從 Amazon ECR 私有儲存庫提取的最低必要許可。

JSON

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Effect": "Allow",
 "Action": [
 "ecr:BatchGetImage",
 "ecr:GetDownloadUrlForLayer",
 "ecr:GetAuthorizationToken"
],
 "Resource": "*"
 }
]
}
```

## 在 Amazon ECS 任務定義中指定 Amazon ECR 映像

建立 Amazon ECS 任務定義時，您可以指定託管在 Amazon ECR 私有儲存庫中的容器映像。在任務定義中，確保您的 Amazon ECR 映像使用完整的 `registry/repository:tag` 名稱。例如 `aws_account_id.dkr.ecr.region.amazonaws.com/my-repository:latest`。

下列任務定義片段會顯示您在 Amazon ECS 任務定義中，會用於指定 Amazon ECR 中託管的容器映像。

```
{
 "family": "task-definition-name",
 ...
 "containerDefinitions": [
 {
 "name": "container-name",
 "image": "aws_account_id.dkr.ecr.region.amazonaws.com/my-
repository:latest",
```

```
 ...
 }
],
...
}
```

## 將 Amazon ECR 映像與 Amazon EKS 搭配使用

您可以將 Amazon ECR 映像與 Amazon EKS 搭配使用。

從 Amazon ECR 參考映像時，您必須為映像使用完整的 `registry/repository:tag` 命名。例如 `aws_account_id.dkr.ecr.region.amazonaws.com/my-repository:latest`。

### 所需的 IAM 許可

如果您的 Amazon EKS 工作負載託管在受管節點、自我管理節點或上 AWS Fargate，請檢閱下列項目：

- 託管在受管或自我管理節點上的 Amazon EKS 工作負載：需要 Amazon EKS 工作者節點 IAM 角色 (NodeInstanceRole)。Amazon EKS 工作節點 IAM 角色必須包含適用於 Amazon ECR 的下列 IAM 政策許可。

#### JSON

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Effect": "Allow",
 "Action": [
 "ecr:BatchCheckLayerAvailability",
 "ecr:BatchGetImage",
 "ecr:GetDownloadUrlForLayer",
 "ecr:GetAuthorizationToken"
],
 "Resource": "*"
 }
]
}
```

**Note**

如果您使用 `eksctl` 或 [Amazon EKS 入門](#) 中的 CloudFormation 範本來建立叢集和工作者節點群組，這些 IAM 許可預設會套用至工作者節點 IAM 角色。

- 託管於的 Amazon EKS 工作負載 AWS Fargate：使用 Fargate Pod 執行角色，該角色提供 Pod 從私有 Amazon ECR 儲存庫提取映像的許可。如需詳細資訊，請參閱[建立 Fargate Pod 執行角色](#)。

## 在 Amazon EKS 叢集上安裝 Helm Chart

Amazon ECR 中託管的 Helm Chart 可以安裝在 Amazon EKS 叢集上。

### 先決條件

- 安裝 Helm 用戶端的最新版本。這些步驟是使用 Helm 版本 3.9.0 進行編寫。如需詳細資訊，請參閱[安裝 Helm](#)。
- 您至少已將 AWS CLI 的版本 1.23.9 或 2.6.3 安裝在自己的電腦上。如需詳細資訊，請參閱[安裝或更新最新版本的 AWS CLI](#)。
- 您已經將 Helm Chart 推送到您的 Amazon ECR 儲存庫。如需詳細資訊，請參閱[將 Helm Chart 推送至 Amazon ECR 私有儲存庫](#)。
- 您已設定 `kubectl` 與 Amazon EKS 合作。如需詳細資訊，請參閱《Amazon EKS 使用者指南》中的[為 Amazon EKS 建立 kubeconfig](#)。若下列命令在您的叢集上成功執行，就表示您的設定正確。

```
kubectl get svc
```

### 在 Amazon EKS 叢集上安裝 Helm Chart

- 對您的 Helm 用戶端驗證您的 Helm Chart 託管的 Amazon ECR 登錄檔。所用的每個登錄檔皆必須取得身分驗證字符，字符有效期間為 12 個小時。如需詳細資訊，請參閱[Amazon ECR 中的私有登錄檔身分驗證](#)。

```
aws ecr get-login-password \
 --region us-west-2 | helm registry login \
 --username AWS \
 --password-stdin aws_account_id.dkr.ecr.region.amazonaws.com
```

- 安裝圖表。將 `helm-test-chart` 取代為您的儲存庫，將 `0.1.0` 取代為 Helm Chart 的標籤。

```
helm install ecr-chart-demo oci://aws_account_id.dkr.ecr.region.amazonaws.com/helm-test-chart --version 0.1.0
```

輸出看起來會與此類似：

```
NAME: ecr-chart-demo
LAST DEPLOYED: Tue May 31 17:38:56 2022
NAMESPACE: default
STATUS: deployed
REVISION: 1
TEST SUITE: None
```

### 3. 驗證圖表安裝。

```
helm list -n default
```

輸出範例：

| NAME           | NAMESPACE             | REVISION    | UPDATED                             |
|----------------|-----------------------|-------------|-------------------------------------|
| STATUS         | CHART                 | APP VERSION |                                     |
| ecr-chart-demo | default               | 1           | 2022-06-01 15:56:40.128669157 +0000 |
| UTC deployed   | helm-test-chart-0.1.0 | 1.16.0      |                                     |

### 4. (選擇性) 請參閱已安裝的 Helm Chart ConfigMap。

```
kubectl describe configmap helm-test-chart-configmap
```

### 5. 完成後，您可以從叢集中移除圖表版本。

```
helm uninstall ecr-chart-demo
```

# 在 Amazon ECR 中簽署映像

Amazon ECR 與 整合 AWS Signer，為您提供兩種簽署容器映像的方式：受管簽署（自動、建議）和手動簽署（用戶端）。您可以將容器映像和簽署同時儲存在私有儲存庫中。

## 選擇簽署方法

Amazon ECR 支援兩種簽署容器映像的方法：

### 受管簽署（建議）

受管簽署會在將映像推送至 Amazon ECR 時自動產生密碼編譯簽章。此方法可簡化設定。受管簽署是大多數使用者的建議方法。如需詳細資訊，請參閱[受管簽署](#)。

### 手動簽署

手動簽署會使用 Notation CLI 和 AWS Signer 外掛程式來簽署映像，然後再將其推送至 Amazon ECR。此方法提供更多對簽署程序的控制，當您需要在推送工作流程之外簽署映像，或需要精細控制簽署操作時，會很有用。如需詳細資訊，請參閱[手動簽署](#)。

## 考量事項

使用 Amazon ECR 映像簽署時應考慮下列事項：

- 儲存在儲存庫中的簽署會計入每個儲存庫映像數目上限的服務配額。針對每個儲存庫配額的影像，每個簽章計為 1 個成品。如需詳細資訊，請參閱[Amazon ECR 服務配額](#)。
- 當儲存庫中有參考成品時，Amazon ECR 生命週期政策會在刪除主體映像的 24 小時內自動清除這些成品。

## 受管簽署

將映像推送至 Amazon ECR 時，Amazon ECR 受管簽署會使用 [AWS Signer](#) 產生密碼編譯簽章，以自動簽署您的容器映像。這不需要安裝和設定用戶端工具，並可讓您集中管理做為登錄組態的簽署。

## 先決條件

若要設定受管簽署，您可以使用參考一或多個簽署者簽署描述檔的 Amazon ECR 建立簽署組態，並選擇性地建立儲存庫篩選條件，以限制哪些儲存庫應簽署其映像。設定完成後，Amazon ECR 受管簽署會使用推送映像的實體身分，在推送映像時自動簽署映像。

您必須先具備下列項目，才能設定受管簽署：

- 簽署者簽署設定檔 — 建立至少一個簽署者[簽署設定檔](#)。簽署設定檔是獨特的 AWS 簽署者資源，可用於在 Amazon ECR 中執行簽署操作。簽署設定檔可讓您簽署和驗證程式碼成品，例如容器映像和 AWS Lambda 部署套件。每個簽署設定檔都會指定要簽署的簽署平台、平台 ID 和其他平台特定資訊。例如，簽署設定檔 ARN 看起來像這樣：`arn:partition:signer:region:account-id:/signing-profiles/profile-name`。
- IAM 許可 — 推送映像的 IAM 主體必須具有必要的 IAM 許可，才能存取相關的簽署者簽署設定檔和相關的 ECR 儲存庫。您需要修改 IAM 主體的身分型政策，以包含 ECR 儲存庫操作和簽署者簽署操作的許可。下列範例政策顯示必要的許可：

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Sid": "UploadSignaturePermissions",
 "Effect": "Allow",
 "Action": [
 "ecr:CompleteLayerUpload",
 "ecr:UploadLayerPart",
 "ecr:InitiateLayerUpload",
 "ecr:BatchCheckLayerAvailability",
 "ecr:PutImage"
],
 "Resource": "arn:aws:ecr:region:account-id:repository/repository-name"
 },
 {
 "Sid": "SignPermissions",
 "Effect": "Allow",
 "Action": [
 "signer:SignPayload"
],
 "Resource": "arn:aws:signer:region:account-id:/signing-profiles/signing-profile-name"
 }
]
}
```

```
}
]
}
```

透過 Amazon ECR 受管簽署，您可以建立多個簽署規則（每個登錄檔最多 10 個），以建立更強大的安全界限。例如，您可能會執行多個建置管道，並想要限制每個管道可以簽署的儲存庫。在每個規則中，您可以設定簽署設定檔並指定儲存庫名稱篩選條件。推送新映像時，Amazon ECR 會比對哪些簽署規則和簽署設定檔可以簽署映像。如果有多個相符項目，Amazon ECR 會產生多個簽章。

#### Note

如果您手動驗證簽章，您仍然需要安裝 Notation CLI。

#### Note

Amazon ECR 受管簽署可在使用 AWS Signer 進行容器映像簽署的所有 AWS 區域中使用。

## 開始使用

請依照下列步驟設定受管簽署。您向 Amazon ECR 提供簽署者簽署描述檔的參考，以及選擇性地限制哪些儲存庫應該簽署其映像的篩選條件。

### AWS 管理主控台

使用下列步驟，使用 [設定受管簽署 AWS 管理主控台](#)。

1. 開啟 [Amazon ECR 主控台](#)。在左側導覽窗格中，選取私有登錄檔、功能和設定、受管簽署。
2. 在簽署規則頁面上，選取建立規則。
3. 在簽署設定檔頁面的選取 AWS 簽署者設定檔下，選擇建立新 AWS 簽署者設定檔、輸入設定檔名稱，以及選擇性地變更簽署有效期間。然後選取下一步。
4. 在篩選條件頁面的選取儲存庫下，輸入儲存庫名稱篩選條件。然後選取下一步。
5. 在檢閱和建立頁面上，驗證您已輸入的 AWS 簽署者描述檔和儲存庫名稱篩選條件。如果一切看起來都正確，請選取儲存。

## AWS CLI

使用下列 AWS CLI 命令來設定受管簽署。

- 建立簽署規則

使用簽署設定檔 ARN 建立簽署組態。使用下列內容建立檔案：

```
{
 "rules": [
 {
 "signingProfileArn": "arn:aws:signer:region:account-id:/signing-
profiles/profile-name",
 "repositoryFilters": [
 {
 "filter": "test*",
 "filterType": "WILDCARD_MATCH"
 }
]
 }
]
}
```

然後執行以下命令：

```
aws ecr --region region \
 put-signing-configuration \
 --signing-configuration file://signing-config.json
```

您應該會看到包含簽署組態的 API 回應。

- 檢視您的簽署組態

擷取您的簽署組態：

```
aws ecr --region region \
 get-signing-configuration
```

您應該會看到包含簽署組態的 API 回應。

- 檢查映像簽署狀態

將映像推送到您的儲存庫。例如：

```
docker pull ubuntu

IMAGE_NAME="account-id.dkr.ecr.region.amazonaws.com/repository-name"
IMAGE_TAG="${IMAGE_NAME}:test-1"

docker tag ubuntu $IMAGE_TAG
docker push $IMAGE_TAG
```

推送之後，請使用映像標籤來檢查簽署狀態：

```
aws ecr --region region \
 describe-image-signing-status \
 --repository-name repository-name \
 --image-id imageTag=test-1
```

如果儲存庫名稱與簽署組態中定義的儲存庫篩選條件相符，您應該會在 API 回應中看到簽署狀態。如果狀態成功，您應該會看到簽章推送到您的儲存庫。

- 刪除您的簽署組態

刪除您的簽署組態：

```
aws ecr --region region \
 delete-signing-configuration
```

您應該會看到 API 回應，其中包含已刪除的簽署組態。

## 考量事項

下列限制和功能適用於受管簽署：

- 不支援跨區域簽署 — 簽署設定檔必須與 Amazon ECR 登錄檔位於相同的區域。您不能使用某個區域的簽署設定檔來簽署位於不同區域的登錄檔中的影像。
- 支援跨帳戶簽署 — 簽署設定檔可以在與您的 Amazon ECR 登錄檔不同的帳戶中。這可讓組織集中管理簽署設定檔，同時允許其他帳戶中的開發人員使用這些設定檔。如需詳細資訊，請參閱《AWS Signer 開發人員指南》中的[設定 Signer 的跨帳戶簽署](#)。
- 無法簽署簽章 — 您無法自行簽署簽章。只能簽署容器映像。

## 簽章驗證

簽署容器映像之後，您可以驗證簽章，以確保映像未遭到竄改，並且來自信任的來源。Amazon ECR 支援多種方法來驗證簽章：

### 使用 Amazon EKS 進行受管驗證

Amazon EKS 為自動簽章驗證提供原生整合。當您在 Amazon EKS 叢集中設定簽章驗證時，服務會在允許容器執行之前自動驗證映像簽章。如需設定簽章驗證的詳細資訊，請參閱《Amazon EKS 使用者指南》中的在[部署期間驗證容器映像簽章](#)。

### Amazon ECS 的 Lambda 許可控制器

Amazon ECS 提供服務生命週期掛鉤，可讓您在服務部署期間執行自訂邏輯。這些掛鉤可以在部署程序的特定點觸發 AWS Lambda 函數，可讓您在允許服務啟動之前驗證容器映像簽章。如需詳細資訊，請參閱《AWS Signer 開發人員指南》中的[驗證 Amazon ECS 的容器映像簽章](#)。

### 使用標記法 CLI 手動驗證

您可以使用標記法 CLI 手動驗證簽章。此方法會要求您在本機電腦或驗證環境中安裝和設定 Notation CLI。如需使用標記法 CLI 驗證映像的詳細指示，請參閱《AWS Signer 開發人員指南》中的[在本機登入後驗證映像](#)。

### 設定 Notation 用戶端的身分驗證

如果您使用手動簽署或使用 Notation CLI 手動驗證簽章，則必須設定 Notation 用戶端，以便向 Amazon ECR 進行身分驗證。如果您在安裝 Notation 用戶端的同一台主機上安裝了 Docker，則 Notation 將重複使用您用於 Docker 用戶端的相同身分驗證方法。Docker login 和 logout 命令將允許 Notation sign 和 verify 命令使用這些相同的登入資料，而且您不需要單獨驗證 Notation。如需設定您的 Notation 用戶端進行身分驗證的詳細資訊，請參閱 Notary Project 文件中的[使用 OCI 相容登錄檔進行驗證](#)。

如果您沒有使用 Docker 或其他使用 Docker 憑證的工具，則建議您使用 Amazon ECR Docker 憑證助手做為您的憑證存放區。如需有關如何安裝和設定 Amazon ECR 憑證助手的詳細資訊，請參閱《[Amazon ECR Docker 憑證助手](#)》。

# 手動簽署

手動簽署會使用標記 CLI 和 AWS Signer 外掛程式來簽署映像，然後再將其推送至 Amazon ECR。此方法提供更多對簽署程序的控制，當您需要在推送工作流程之外簽署映像，或需要精細控制簽署操作時很有用。

如需使用標記法 CLI 簽署容器映像的詳細說明 AWS Signer，請參閱《AWS Signer 開發人員指南》中的[在 Signer 中簽署容器映像](#)和相關主題。

## 先決條件

開始之前，請先達成以下先決條件。

- 安裝和設定最新版 AWS CLI。如需詳細資訊，請參閱《AWS Command Line Interface 使用者指南》中的「[安裝或更新最新版 AWS CLI](#)」。
- 安裝 Notation CLI 和適用於 Notation 的 AWS Signer 外掛程式。如需詳細資訊，請參閱《AWS Signer 開發人員指南》中的「[簽署容器映像的先決條件](#)」。
- 在 Amazon ECR 私有儲存庫中，儲存了需要簽署的容器映像。如需詳細資訊，請參閱[將映像推送至 Amazon ECR 私有儲存庫](#)。

# 掃描映像是否有 Amazon ECR 中的軟體漏洞

Amazon ECR 映像掃描有助於識別容器映像中的軟體漏洞。提供下列掃描類型。

## Important

在增強型掃描和基本掃描之間切換會導致先前建立的掃描不再可用。您必須再次設定掃描。不過，如果您切換回先前的掃描類型，則會提供已建立的掃描。

## Note

封存的映像無法掃描。封存的映像必須先還原，才能進行掃描。如需封存和還原映像的詳細資訊，請參閱[在 Amazon ECR 中封存映像](#)。

- 增強型掃描 – Amazon ECR 與 Amazon Inspector 整合，可自動持續掃描您的儲存庫。系統會掃描容器映像，檢查是否有作業系統和程式設計語言套件漏洞。如果出現新漏洞，掃描結果會更新，Amazon Inspector 會向 EventBridge 發出事件來通知您。增強型掃描提供下列項目：
  - 作業系統和程式設計語言套件漏洞
  - 兩種掃描頻率：推送時掃描和連續掃描
- 基本掃描 – Amazon ECR AWS 使用原生技術搭配 Common Vulnerabilities and Exposures (CVEs) 資料庫來掃描作業系統漏洞。

若使用基本型掃描，您可以將儲存庫設定為在推送時掃描，也可以執行手動掃描；Amazon ECR 會提供掃描問題清單。基本掃描提供下列項目：

- 作業系統掃描
- 兩種掃描頻率：手動和推送時掃描

## Important

新版本的 Amazon ECR Basic Scanning 不會使用 DescribeImages API 回應中的 imageScanFindingsSummary 和 imageScanStatus 屬性來傳回掃描結果。請改用 DescribeImageScanFindings API。如需詳細資訊，請參閱[DescribeImageScanFindings](#)。

## 選擇在 Amazon ECR 中掃描哪些儲存庫的篩選條件

當您為私有登錄檔設定映像掃描時，您可以使用篩選條件來選擇掃描哪些儲存庫。

如果使用基本掃描，您可以指定「依據推送篩選條件進行掃描」，這樣便可以指定有新映像推送時要對哪些儲存庫執行映像掃描。與推送篩選條件上的基本掃描不相符的任何儲存庫都會設定為手動掃描頻率，這表示要執行掃描，您必須手動觸發掃描。

如果使用增強掃描，您可以為「依據推送進行掃描」和「連續掃描」分別指定篩選條件。任何不符合增強掃描篩選條件的儲存庫都將停止掃描。如果您正在使用增強掃描並為「依據推送進行掃描」和「連續掃描」分別指定了篩選條件，發生同一儲存庫符合多個篩選條件的情況，則 Amazon ECR 會對該儲存庫強制優先執行「連續掃描」，而非「依據推送篩選條件進行掃描」。

### 篩選萬用字元

指定篩選條件時，沒有萬用字元的篩選條件會比對內含篩選條件的所有儲存庫名稱。具有萬用字元 (\*) 的篩選條件會比對任何儲存庫名稱，其中萬用字元會取代儲存庫名稱中零個以上的字元。

下表提供範例，其中儲存庫名稱在水平軸上表示，而範例篩選條件在垂直軸上指定。

|           | prod | repo-prod | Prod-repo | repo-prod-repo | prodrepo |
|-----------|------|-----------|-----------|----------------|----------|
| prod      | 是    | 是         | 是         | 是              | 是        |
| *prod     | 是    | 是         | 否         | 否              | 否        |
| prod*     | 是    | 否         | 是         | 否              | 是        |
| *prod*    | 是    | 是         | 是         | 是              | 是        |
| prod*repo | 否    | 否         | 是         | 否              | 是        |

## 在 Amazon ECR 中掃描映像是否有作業系統和程式設計語言套件漏洞

Amazon ECR 增強型掃描已與 Amazon Inspector 整合，可為容器映像提供漏洞掃描。系統會掃描容器映像，檢查是否有作業系統和程式設計語言套件漏洞。您可以使用 Amazon ECR 和 Amazon Inspector

直接檢視掃描問題清單。如需有關 Amazon Inspector 的詳細資訊，請參閱《Amazon Inspector 使用者指南》中的[使用 Amazon Inspector 掃描容器映像](#)。

透過增強型掃描，您可以選擇要將哪些儲存庫設定為自動連續掃描，以及將哪些儲存庫設定為推送時掃描。設定掃描篩選條件可完成此操作。

## 增強型掃描的注意事項

啟用 Amazon ECR 增強型掃描之前，請考慮下列事項。

- 使用此功能不會從 Amazon ECR 產生任何額外成本，但是，掃描映像檔則會從 Amazon Inspector 產生成本。此功能適用於支援 Amazon Inspector 的區域。如需詳細資訊，請參閱：
  - Amazon Inspector 定價 – [Amazon Inspector 定價](#)。
  - Amazon Inspector 支援的區域 – [區域和端點](#)。
- Amazon ECR 增強型掃描顯示如何在 Amazon EKS 和 Amazon ECS 上使用映像。您可以查看上次使用映像的時間，並識別有多少叢集使用每個映像。此資訊可協助您排定主動使用映像的漏洞修復優先順序。您可以快速判斷哪些叢集可能受到新發現的漏洞影響。如需如何請求這些資訊和檢視回應的詳細資訊，請參閱 [DescribeImageScanFindings](#)。
- Amazon Inspector 支援掃描特定作業系統。如需完整清單，請參閱《Amazon Inspector 使用者指南》中的[支援的作業系統：Amazon ECR 掃描](#)。
- Amazon Inspector 使用服務連結 IAM 角色，該角色提供為儲存庫提供增強型掃描所需的許可。為私有登錄檔開啟增強型掃描時，Amazon Inspector 會自動建立服務連結 IAM 角色。如需詳細資訊，請參閱《Amazon Inspector 使用者指南》中的[使用 Amazon Inspector 的服務連結角色](#)。
- 當您最初開啟私有登錄檔的增強型掃描時，Amazon Inspector 只會根據映像推送時間戳記，辨識過去 14 天內推送至 Amazon ECR 的映像。較舊的映像會具有 `SCAN_ELIGIBILITY_EXPIRED` 掃描狀態。如果您希望 Amazon Inspector 掃描這些映像，則必須將這些映像再次推送到儲存庫中。
- 為 Amazon ECR 私有登錄檔開啟增強型掃描時，系統只會使用增強型掃描來掃描符合掃描篩選條件的所有儲存庫。不符合篩選條件的儲存庫都會具備 `Off` 掃描頻率，且不會被掃描。不支援使用增強型掃描的手動掃描。如需詳細資訊，請參閱[選擇在 Amazon ECR 中掃描哪些儲存庫的篩選條件](#)。
- 如果您為「依據推送進行掃描」和「連續掃描」分別指定了篩選條件，發生同一儲存庫符合多個篩選條件的情況，則 Amazon ECR 會對該儲存庫強制優先執行「連續掃描」，而非「依據推送篩選條件進行掃描」。
- 開啟增強型掃描後，如果儲存庫的掃描頻率發生變更，Amazon ECR 會將事件傳送至 EventBridge。初始掃描完成且建立、更新或關閉映像掃描問題清單後，Amazon Inspector 會將事件發送到 EventBridge。

## 變更 Amazon Inspector 中影像的增強型掃描持續時間

啟用增強型掃描之後，Amazon ECR 會在設定的持續時間內持續掃描新推送的映像。根據預設，Amazon Inspector 會監控您的儲存庫，直到刪除映像或停用增強型掃描為止。您可以在 Amazon Inspector 主控台中設定推送日期持續時間（最長可達生命週期）和重新掃描持續時間，以符合環境的需求。當儲存庫的掃描持續時間經過時，掃描狀態會顯示為 `SCAN_ELIGIBILITY_EXPIRED`。如需在 Amazon Inspector 中設定 Amazon ECR 重新掃描持續時間設定的詳細資訊，請參閱 [《Amazon Inspector 使用者指南》](#) 中的 [設定 Amazon ECR 重新掃描持續時間](#)。Amazon Inspector

## 在 Amazon ECR 中增強型掃描所需的 IAM 許可

Amazon ECR 增強型掃描需要 Amazon Inspector 服務連結 IAM 角色，且啟用和使用增強型掃描的 IAM 委託人有權呼叫掃描所需的 Amazon Inspector API。為私有登錄檔開啟增強型掃描時，Amazon Inspector 會自動建立 Amazon Inspector 服務連結 IAM 角色。如需詳細資訊，請參閱 [《Amazon Inspector 使用者指南》](#) 中的 [使用 Amazon Inspector 的服務連結角色](#)。

下列 IAM 政策授予啟用及使用增強型掃描的必要許可。其中包含 Amazon Inspector 建立服務連結 IAM 角色所需的許可，以及開啟和關閉增強型掃描和擷取掃描問題清單所需的 Amazon Inspector API 許可。

### JSON

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Effect": "Allow",
 "Action": [
 "inspector2:Enable",
 "inspector2:Disable",
 "inspector2:ListFindings",
 "inspector2:ListAccountPermissions",
 "inspector2:ListCoverage"
],
 "Resource": "*"
 },
 {
 "Effect": "Allow",
 "Action": "iam:CreateServiceLinkedRole",
 "Resource": "*",
```

```
 "Condition": {
 "StringEquals": {
 "iam:AWSServiceName": [
 "inspector2.amazonaws.com"
]
 }
 }
]
}
```

## 在 Amazon ECR 中設定映像的增強型掃描

為您的私有登錄檔設定每個區域的增強型掃描。

確認您具有設定增強型掃描的適當 IAM 許可。如需相關資訊，請參閱 [在 Amazon ECR 中增強型掃描所需的 IAM 許可](#)。

### AWS 管理主控台

#### 開啟私有登錄檔的增強型掃描

1. 在 <https://console.aws.amazon.com/ecr/repositories> 開啟 Amazon ECR 主控台。
2. 從導覽列選擇要為其設定掃描組態的區域。
3. 在導覽窗格中，選擇私有登錄檔，然後選擇設定。
4. 在 Scanning configuration (掃描組態) 頁面，針對 Scan type (掃描類型) 選擇 Enhanced scanning (增強型掃描)。

根據預設，選取增強型掃描時，會持續掃描所有儲存庫。

5. 若要選擇要持續掃描的特定儲存庫，請清除持續掃描所有儲存庫方塊，然後定義篩選條件：

#### Important

沒有萬用字元的篩選條件會比對內含篩選條件的所有儲存庫名稱。具有萬用字元 (\*) 的篩選條件會比對儲存庫名稱，其中萬用字元會取代儲存庫名稱中零個以上的字元。若要查看篩選條件行為的範例，請參閱 [the section called “篩選萬用字元”](#)。

- a. 根據儲存庫名稱輸入篩選條件，然後選擇新增篩選條件。

- b. 決定推送映像時要掃描的儲存庫：
  - 若要在推送時掃描所有儲存庫，請選取推送時掃描所有儲存庫。
  - 若要在推送時選擇要掃描的特定儲存庫，請根據儲存庫名稱輸入篩選條件，然後選擇新增篩選條件。
6. 選擇儲存。
7. 在要開啟增強型掃描的各個區域重複這些步驟。

## AWS CLI

使用以下 AWS CLI 命令，使用 開啟私有登錄檔的增強型掃描 AWS CLI。您可以使用 `rules` 物件指定掃描篩選條件。

- [put-registry-scanning-configuration](#) (AWS CLI)

下列範例會為私有登錄檔開啟增強型掃描。在預設情況下，當未指定 `rules` 時，Amazon ECR 會將掃描組態設定為替所有儲存庫連續掃描。

```
aws ecr put-registry-scanning-configuration \
 --scan-type ENHANCED \
 --region us-east-2
```

下列範例會為私有登錄檔開啟增強型掃描，並指定掃描篩選條件。範例中的掃描篩選條件可針對在其名稱具有 `prod` 的所有儲存庫開啟連續掃描。

```
aws ecr put-registry-scanning-configuration \
 --scan-type ENHANCED \
 --rules '[{"repositoryFilters" : [{"filter": "prod", "filterType" :
 "WILDCARD"}], "scanFrequency" : "CONTINUOUS_SCAN"}]' \
 --region us-east-2
```

下列範例會為私有登錄檔開啟增強型掃描，並指定多個掃描篩選條件。範例中的掃描篩選條件可針對在其名稱中具有 `prod` 的所有儲存庫開啟連續掃描，並且僅針對所有其他儲存庫開啟推送時掃描。

```
aws ecr put-registry-scanning-configuration \
 --scan-type ENHANCED \
 --rules '[{"repositoryFilters" : [{"filter": "prod", "filterType" :
 "WILDCARD"}], "scanFrequency" : "CONTINUOUS_SCAN"}]' \
 --region us-east-2
```

```
--rules '[{"repositoryFilters" : [{"filter": "prod", "filterType" :
"WILDCARD"}], "scanFrequency" : "CONTINUOUS_SCAN"}, {"repositoryFilters" :
[{"filter": "*", "filterType" : "WILDCARD"}], "scanFrequency" : "SCAN_ON_PUSH"}]' \
--region us-west-2
```

## 在 Amazon ECR 中傳送用於增強型掃描的 EventBridge 事件

開啟增強型掃描後，如果儲存庫的掃描頻率發生變更，Amazon ECR 會將事件傳送至 EventBridge。當初始掃描完成，以及建立、更新或關閉映像掃描調查結果時，Amazon Inspector 會將事件傳送至 EventBridge。

### 儲存庫掃描頻率變更事件

為登錄檔開啟增強型掃描時，Amazon ECR 會在開啟了增強型掃描的資源發生變更時傳送下列事件。這包含正在建立的新儲存庫、正在變更的儲存庫掃描頻率，或是在開啟了增強型掃描的儲存庫中建立或刪除映像時。如需詳細資訊，請參閱[掃描映像是否有 Amazon ECR 中的軟體漏洞](#)。

```
{
 "version": "0",
 "id": "0c18352a-a4d4-6853-ef53-0abEXAMPLE",
 "detail-type": "ECR Scan Resource Change",
 "source": "aws.ecr",
 "account": "123456789012",
 "time": "2021-10-14T20:53:46Z",
 "region": "us-east-1",
 "resources": [],
 "detail": {
 "action-type": "SCAN_FREQUENCY_CHANGE",
 "repositories": [{
 "repository-name": "repository-1",
 "repository-arn": "arn:aws:ecr:us-east-1:123456789012:repository/repository-1",
 "scan-frequency": "SCAN_ON_PUSH",
 "previous-scan-frequency": "MANUAL"
 },
 {
 "repository-name": "repository-2",
 "repository-arn": "arn:aws:ecr:us-east-1:123456789012:repository/repository-2",
 "scan-frequency": "CONTINUOUS_SCAN",
 "previous-scan-frequency": "SCAN_ON_PUSH"
 },
 {
 "repository-name": "repository-3",
```

```

 "repository-arn": "arn:aws:ecr:us-east-1:123456789012:repository/repository-3",
 "scan-frequency": "CONTINUOUS_SCAN",
 "previous-scan-frequency": "SCAN_ON_PUSH"
 }
],
"resource-type": "REPOSITORY",
"scan-type": "ENHANCED"
}
}

```

### 初始映像掃描事件 (增強型掃描)

為登錄檔開啟增強型掃描時，Amazon Inspector 會在初始映像掃描完成時傳送下列事件。finding-severity-counts 參數只會在出現嚴重性等級時才傳回嚴重性等級的值。例如，如果映像在 CRITICAL 等級沒有問題清單，則不會傳回任何重要等級數值。如需詳細資訊，請參閱[在 Amazon ECR 中掃描映像是否有作業系統和程式設計語言套件漏洞](#)。

事件模式：

```

{
 "source": ["aws.inspector2"],
 "detail-type": ["Inspector2 Scan"]
}

```

輸出範例：

```

{
 "version": "0",
 "id": "739c0d3c-4f02-85c7-5a88-94a9EXAMPLE",
 "detail-type": "Inspector2 Scan",
 "source": "aws.inspector2",
 "account": "123456789012",
 "time": "2021-12-03T18:03:16Z",
 "region": "us-east-2",
 "resources": [
 "arn:aws:ecr:us-east-2:123456789012:repository/amazon/amazon-ecs-sample"
],
 "detail": {
 "scan-status": "INITIAL_SCAN_COMPLETE",
 "repository-name": "arn:aws:ecr:us-east-2:123456789012:repository/amazon/amazon-ecs-sample",
 "finding-severity-counts": {
 "CRITICAL": 7,

```

```

 "HIGH": 61,
 "MEDIUM": 62,
 "TOTAL": 158
 },
 "image-digest":
"sha256:36c7b282abd0186e01419f2e58743e1bf635808231049bbc9d77e5EXAMPLE",
 "image-tags": [
 "latest"
]
}
}

```

### 映像掃描問題清單更新事件 (增強型掃描)

為登錄檔開啟增強型掃描時，Amazon Inspector 會在建立、更新或關閉映像掃描問題清單時傳送下列事件。如需詳細資訊，請參閱 [在 Amazon ECR 中掃描映像是否有作業系統和程式設計語言套件漏洞](#)。

事件模式：

```

{
 "source": ["aws.inspector2"],
 "detail-type": ["Inspector2 Finding"]
}

```

輸出範例：

```

{
 "version": "0",
 "id": "42dbea55-45ad-b2b4-87a8-afaEXAMPLE",
 "detail-type": "Inspector2 Finding",
 "source": "aws.inspector2",
 "account": "123456789012",
 "time": "2021-12-03T18:02:30Z",
 "region": "us-east-2",
 "resources": [
 "arn:aws:ecr:us-east-2:123456789012:repository/amazon/amazon-ecs-sample/
sha256:36c7b282abd0186e01419f2e58743e1bf635808231049bbc9d77eEXAMPLE"
],
 "detail": {
 "awsAccountId": "123456789012",
 "description": "In libssh2 v1.9.0 and earlier versions, the SSH_MSG_DISCONNECT
logic in packet.c has an integer overflow in a bounds check, enabling an attacker to
specify an arbitrary (out-of-bounds) offset for a subsequent memory read. A crafted

```

```
SSH server may be able to disclose sensitive information or cause a denial of service
condition on the client system when a user connects to the server.",
 "findingArn": "arn:aws:inspector2:us-east-2:123456789012:finding/
be674aadd0f75ac632055EXAMPLE",
 "firstObservedAt": "Dec 3, 2021, 6:02:30 PM",
 "inspectorScore": 6.5,
 "inspectorScoreDetails": {
 "adjustedCvss": {
 "adjustments": [],
 "cvssSource": "REDHAT_CVE",
 "score": 6.5,
 "scoreSource": "REDHAT_CVE",
 "scoringVector": "CVSS:3.0/AV:N/AC:L/PR:N/UI:R/S:U/C:H/I:N/A:N",
 "version": "3.0"
 }
 },
 "lastObservedAt": "Dec 3, 2021, 6:02:30 PM",
 "packageVulnerabilityDetails": {
 "cvss": [
 {
 "baseScore": 6.5,
 "scoringVector": "CVSS:3.0/AV:N/AC:L/PR:N/UI:R/S:U/C:H/I:N/A:N",
 "source": "REDHAT_CVE",
 "version": "3.0"
 },
 {
 "baseScore": 5.8,
 "scoringVector": "AV:N/AC:M/Au:N/C:P/I:N/A:P",
 "source": "NVD",
 "version": "2.0"
 },
 {
 "baseScore": 8.1,
 "scoringVector": "CVSS:3.1/AV:N/AC:L/PR:N/UI:R/S:U/C:H/I:N/A:H",
 "source": "NVD",
 "version": "3.1"
 }
],
 "referenceUrls": [
 "https://access.redhat.com/errata/RHSA-2020:3915"
],
 "source": "REDHAT_CVE",
 "sourceUrl": "https://access.redhat.com/security/cve/CVE-2019-17498",
 "vendorCreatedAt": "Oct 16, 2019, 12:00:00 AM",
```

```
"vendorSeverity": "Moderate",
"vulnerabilityId": "CVE-2019-17498",
"vulnerablePackages": [
 {
 "arch": "X86_64",
 "epoch": 0,
 "name": "libssh2",
 "packageManager": "OS",
 "release": "12.amzn2.2",
 "sourceLayerHash":
"sha256:72d97abdfae3b3c933ff41e39779cc72853d7bd9dc1e4800c5294dEXAMPLE",
 "version": "1.4.3"
 }
],
"remediation": {
 "recommendation": {
 "text": "Update all packages in the vulnerable packages section to
their latest versions."
 }
},
"resources": [
 {
 "details": {
 "awsEcrContainerImage": {
 "architecture": "amd64",
 "imageHash":
"sha256:36c7b282abd0186e01419f2e58743e1bf635808231049bbc9d77e5EXAMPLE",
 "imageTags": [
 "latest"
],
 "platform": "AMAZON_LINUX_2",
 "pushedAt": "Dec 3, 2021, 6:02:13 PM",
 "lastInUseAt": "Dec 3, 2021, 6:02:13 PM",
 "inUseCount": 1,
 "registry": "123456789012",
 "repositoryName": "amazon/amazon-ecs-sample"
 }
 },
 "id": "arn:aws:ecr:us-east-2:123456789012:repository/amazon/amazon-ecs-
sample/sha256:36c7b282abd0186e01419f2e58743e1bf635808231049bbc9d77EXAMPLE",
 "partition": "N/A",
 "region": "N/A",
 "type": "AWS_ECR_CONTAINER_IMAGE"
 }
]
```

```
 }
],
 "severity": "MEDIUM",
 "status": "ACTIVE",
 "title": "CVE-2019-17498 - libssh2",
 "type": "PACKAGE_VULNERABILITY",
 "updatedAt": "Dec 3, 2021, 6:02:30 PM"
}
}
```

## 在 Amazon ECR 中擷取增強型掃描的調查結果

您可以擷取上次完成增強型映像掃描的掃描問題清單，然後在 Amazon Inspector 中開啟問題清單以查看更多詳細資訊。根據 Common Vulnerabilities and Exposures (CVEs) 資料庫，依嚴重性列出發現的軟體漏洞。

如需在掃描映像時某些常見問題的故障診斷詳細資訊，請參閱 [對 Amazon ECR 中的映像掃描進行故障診斷](#)。

### AWS 管理主控台

使用下列步驟以利用 AWS 管理主控台擷取映像掃描結果。

#### 擷取映像掃描問題清單

1. 在 <https://console.aws.amazon.com/ecr/> 開啟 Amazon ECR 主控台。
2. 從導覽列選擇儲存庫所在的區域。
3. 在導覽窗格中，選擇 Repositories (儲存庫)。
4. 在 Repositories (儲存庫) 頁面上，選擇包含要擷取掃描結果之映像的儲存庫。
5. 在影像頁面的影像標籤欄下，選取要擷取掃描問題清單的影像標籤。
6. 若要在 Amazon Inspector 主控台中檢視更多詳細資訊，請在名稱欄中選擇漏洞名稱。

### AWS CLI

使用以下 AWS CLI 命令，使用擷取影像掃描問題清單 AWS CLI。您可以使用 `imageTag` 或 `imageDigest` 指定映像，兩者皆可利用 [list-images](#) CLI 命令取得。

- [describe-image-scan-findings](#) (AWS CLI)

下列範例使用映像標籤。

```
aws ecr describe-image-scan-findings \
 --repository-name name \
 --image-id imageTag=tag_name \
 --region us-east-2
```

下列範例使用映像摘要。

```
aws ecr describe-image-scan-findings \
 --repository-name name \
 --image-id imageDigest=sha256_hash \
 --region us-east-2
```

## 掃描映像是否有 Amazon ECR 中的作業系統漏洞

Amazon ECR AWS 基本掃描使用原生技術來掃描容器映像是否有軟體漏洞。基本掃描可跨廣泛的熱門作業系統提供漏洞偵測，取得超過 50 個資料摘要，以產生常見漏洞和暴露 (CVEs) 的問題清單。這些來源包括廠商安全建議、資料饋送、威脅情報饋送，以及國家漏洞資料庫 (NVD) 和 MITRE。

[AWS 依區域列出的](#)所有區域都支援 Amazon ECR 基本掃描。

若有，Amazon ECR 會使用上游分發來源提供的 CVE 嚴重性程度。若無，則會使用通用漏洞評分系統 (CVSS) 分數。CVSS 分數可用於取得 NVD 漏洞嚴重性等級。如需詳細資訊，請參閱 [NVD 漏洞嚴重性等級](#)。

Amazon ECR 基本掃描支援篩選條件，以指定推送時要掃描的儲存庫。任何不符合推送時掃描篩選條件的儲存庫都會設定為手動掃描頻率，這表示您必須手動啟動掃描。影像每 24 小時可以掃描一次。如果已設定，則 24 小時包含推送時的初始掃描，以及任何手動掃描。透過基本掃描，您可以在指定的登錄檔中每 24 小時掃描最多 100,000 張映像。

可以為每個映像擷取上次完成的映像掃描結果。映像掃描完成後，Amazon ECR 會將事件傳送至 Amazon EventBridge。如需詳細資訊，請參閱 [Amazon ECR 事件和 EventBridge](#)。

## 作業系統支援基本掃描

作為安全最佳實務，為了持續涵蓋範圍，我們建議您繼續使用支援的作業系統版本。根據廠商政策，已終止的作業系統不會再使用修補程式更新，而且在許多情況下，不會再為這些作業系統發行新的安全建

議。此外，有些廠商會在受影響的作業系統達到標準支援結束時，從其摘要中移除現有的安全建議和偵測。分佈失去廠商的支援後，Amazon ECR 可能不再支援掃描其是否有漏洞。Amazon ECR 為已終止的作業系統產生的任何問題清單都應該僅用於資訊目的。如需支援作業系統和版本的完整清單，請參閱 [《Amazon Inspector 使用者指南》中的支援的作業系統 - Amazon Inspector 掃描](#)。Amazon Inspector

## 在 Amazon ECR 中設定影像的基本掃描

根據預設，Amazon ECR 會為所有私有登錄檔開啟基本掃描。因此，除非您已變更私有登錄檔上的掃描設定，否則不需要開啟基本掃描。

您可以使用下列步驟來定義推送篩選條件的一或多個掃描。

為您的私有登錄檔開啟基本掃描

1. 在 <https://console.aws.amazon.com/ecr/private-registry/repositories> 開啟 Amazon ECR 主控台
2. 從導覽列選擇要為其設定掃描組態的區域。
3. 在導覽窗格中，選擇私有登錄檔，掃描。
4. 在 Scanning configuration (掃描組態) 頁面，針對 Scan type (掃描類型) 選擇 Basic scanning (基本型掃描)。
5. 在預設情況下，系統將所有儲存庫設定為 Manual (手動) 掃描。您可選擇設定推送時掃描，方法是指定推送時掃描篩選條件。您可以為所有儲存庫或個別儲存庫設定推送時掃描。如需詳細資訊，請參閱 [選擇在 Amazon ECR 中掃描哪些儲存庫的篩選條件](#)。

### Note

如果針對儲存庫啟用推送時掃描，也會對封存後還原的映像進行掃描。還原的影像將不提供更舊的掃描。

## 手動掃描映像是否有 Amazon ECR 中的作業系統漏洞

如果您的儲存庫未設定為在推送時掃描，您可以手動啟動映像掃描。影像每 24 小時可以掃描一次。如果已設定，則 24 小時包含推送時的初始掃描，以及任何手動掃描。

如需在掃描映像時某些常見問題的故障診斷詳細資訊，請參閱 [對 Amazon ECR 中的映像掃描進行故障診斷](#)。

## AWS 管理主控台

使用下列步驟，利用 AWS 管理主控台以啟動手動映像掃描。

1. 在 <https://console.aws.amazon.com/ecr/private-registry/repositories> 開啟 Amazon ECR 主控台
2. 從導覽列選擇您儲存庫所建立的區域。
3. 在導覽窗格中，選擇 Repositories (儲存庫)。
4. 在 Repositories (儲存庫) 頁面上，選擇包含要掃描之映像的儲存庫。
5. 在 Images (映像) 頁面上，選取要掃描的映像，然後選擇 Scan (掃描)。

## AWS CLI

- [start-image-scan](#) (AWS CLI)

下列範例使用映像標籤。

```
aws ecr start-image-scan --repository-name name --image-id imageTag=tag_name --region us-east-2
```

下列範例使用映像摘要。

```
aws ecr start-image-scan --repository-name name --image-id imageDigest=sha256_hash --region us-east-2
```

## AWS Tools for Windows PowerShell

- [Get-ECRImageScanFinding](#) (AWS Tools for Windows PowerShell)

下列範例使用映像標籤。

```
Start-ECRImageScan -RepositoryName name -ImageId_ImageTag tag_name -Region us-east-2 -Force
```

下列範例使用映像摘要。

```
Start-ECRImageScan -RepositoryName name -ImageId_ImageDigest sha256_hash -
Region us-east-2 -Force
```

## 在 Amazon ECR 中擷取基本掃描的問題清單

您可以擷取上次完成的基本映像掃描的掃描問題清單。根據 Common Vulnerabilities and Exposures (CVEs) 資料庫，依嚴重性列出發現的軟體漏洞。

如需在掃描映像時某些常見問題的故障診斷詳細資訊，請參閱 [對 Amazon ECR 中的映像掃描進行故障診斷](#)。

### AWS 管理主控台

使用下列步驟以利用 AWS 管理主控台擷取映像掃描結果。

#### 擷取映像掃描問題清單

1. 在 <https://console.aws.amazon.com/ecr/private-registry/repositories> 開啟 Amazon ECR 主控台
2. 從導覽列選擇您儲存庫所建立的區域。
3. 在導覽窗格中，選擇 Repositories (儲存庫)。
4. 在 Repositories (儲存庫) 頁面上，選擇包含要擷取掃描結果之映像的儲存庫。
5. 在映像頁面的映像標籤欄下，選取要擷取掃描問題清單的映像標籤。

### AWS CLI

使用以下 AWS CLI 命令，使用擷取影像掃描問題清單 AWS CLI。您可以使用 `imageTag` 或 `imageDigest` 指定映像，兩者皆可利用 [list-images](#) CLI 命令取得。

- [describe-image-scan-findings](#) (AWS CLI)

下列範例使用映像標籤。

```
aws ecr describe-image-scan-findings --repository-name name --image-id
imageTag=tag_name --region us-east-2
```

下列範例使用映像摘要。

```
aws ecr describe-image-scan-findings --repository-name name --image-id
imageDigest=sha256_hash --region us-east-2
```

## AWS Tools for Windows PowerShell

- [Get-ECRImageScanFinding](#) (AWS Tools for Windows PowerShell)

下列範例使用映像標籤。

```
Get-ECRImageScanFinding -RepositoryName name -ImageId_ImageTag tag_name -
Region us-east-2
```

下列範例使用映像摘要。

```
Get-ECRImageScanFinding -RepositoryName name -ImageId_ImageDigest sha256_hash -
Region us-east-2
```

## 對 Amazon ECR 中的映像掃描進行故障診斷

以下是常見的映像掃描失敗。您可以透過顯示映像詳細資訊或透過 API 或使用 AWS CLI DescribeImageScanFindings API，在 Amazon ECR 主控台中檢視這類錯誤。

### UnsupportedImageError

嘗試對使用 Amazon ECR 不支援基本映像掃描的作業系統建置的映像執行基本型掃描時，您可能會收到 UnsupportedImageError 錯誤訊息。Amazon ECR 支援對 Amazon Linux、Amazon Linux 2、Debian、Ubuntu、CentOS、Oracle Linux、Alpine 和 RHEL Linux 發行版本的主要版本進行包漏洞掃描。一旦發行版本失去廠商的支援，Amazon ECR 可能不再支援掃描它是否有漏洞。Amazon ECR 不支援掃描從 [Docker scratch](#) 映像建置的映像。

#### Important

使用增強型掃描時，Amazon Inspector 支援掃描特定類型的作業系統和媒體。如需完整清單，請參閱《Amazon Inspector 使用者指南》中的 [支援的作業系統和媒體類型](#)。

傳回 UNDEFINED 嚴重性等級。

您可能會收到具有 UNDEFINED 嚴重性等級的掃描結果。下列是造成此問題的常見原因：

- CVE 來源未指派漏洞的優先順序。
- Amazon ECR 無法辨識指派給漏洞的優先順序。

若要判定漏洞的嚴重性和描述，您可以直接從來源檢視 CVE。

## 了解掃描狀態 SCAN\_ELIGIBILITY\_EXPIRED

當為您的私有登錄檔啟用使用 Amazon Inspector 的增強型掃描而且您正在檢視掃描弱點時，您可能會看到掃描狀態為 SCAN\_ELIGIBILITY\_EXPIRED。以下是造成此問題的最常見原因。

- 當您一開始為私有登錄檔開啟增強型掃描時，Amazon Inspector 只會根據映像推送時間戳記，辨識過去 30 天內推送至 Amazon ECR 的映像。較舊的映像會具有 SCAN\_ELIGIBILITY\_EXPIRED 掃描狀態。如果您希望 Amazon Inspector 掃描這些映像，則必須將這些映像再次推送到儲存庫中。
- 如果在 Amazon Inspector 主控台中變更 ECR re-scan duration (ECR 重新掃描持續時間)，且經過這段時間後，映像的掃描狀態變更為 inactive 並出現原因代碼 expired，且該映像的所有相關發現結果都排定為關閉。這會導致 Amazon ECR 主控台將掃描狀態列示為 SCAN\_ELIGIBILITY\_EXPIRED。

## 將上游登錄檔與 Amazon ECR 私有登錄檔同步

使用提取快取規則，您可以將上游登錄檔的內容與 Amazon ECR 私有登錄檔同步。

Amazon ECR 目前支援為下列上游登錄檔建立提取快取規則：

- Amazon ECR Public、Kubernetes 容器映像登錄檔和 Quay（不需要身分驗證）
- Docker Hub、Microsoft Azure Container Registry、GitHub Container Registry、GitLab Container Registry 和 Chainguard Registry（需要使用 AWS Secrets Manager 秘密進行身分驗證）
- Amazon ECR（需要使用 IAM AWS 角色進行身分驗證）

對於 GitLab Container Registry，Amazon ECR 僅支援使用 GitLab 的軟體即服務 (SaaS) 產品提取快取。如需使用 GitLab SaaS 產品的詳細資訊，請參閱 [GitLab.com](https://gitlab.com)

對於需要使用秘密進行身分驗證的上游登錄檔（例如 Docker Hub），您必須將登入資料存放在 AWS Secrets Manager 秘密中。您可以使用 Amazon ECR 主控台，為每個已驗證的上游登錄檔建立 Secrets Manager 秘密。如需使用 Secrets Manager 主控台建立 Secrets Manager 秘密的詳細資訊，請參閱 [將上游儲存庫登入資料存放在 AWS Secrets Manager 秘密中](#)。

對於 Amazon ECR，如果上游和下游 Amazon ECR 登錄檔屬於不同的 AWS 帳戶，您必須建立 IAM 角色。如需建立 IAM 角色的詳細資訊，請參閱 [跨帳戶 ECR 到 ECR 提取快取所需的 IAM 政策](#)。

建立上游登錄檔的提取快取規則之後，請使用 Amazon ECR 私有登錄檔 URI 從該上游登錄檔提取映像。接著 Amazon ECR 會建立儲存庫，並在您的私有登錄中快取該映像。對於具有指定標籤之快取映像的後續提取請求，Amazon ECR 會檢查上游登錄檔是否有具有該特定標籤之映像的新版本，並嘗試至少每 24 小時更新一次私有登錄檔中的映像。

## 儲存庫建立範本

Amazon ECR 已新增對儲存庫建立範本的支援，可讓您使用提取快取規則，為 Amazon ECR 代表您建立的新儲存庫指定初始組態。每個範本都包含一個儲存庫命名空間字首，用於將新儲存庫與特定範本匹配。範本可以指定所有儲存庫設定的組態，包括資源型存取政策、標籤不變性、加密和生命週期政策。儲存庫建立範本中的設定只會在建立儲存庫期間套用，對使用任何其他方法建立的現有儲存庫或儲存庫沒有任何影響。如需詳細資訊，請參閱 [用於控制提取快取、推送時建立或複寫動作期間建立的儲存庫的範本](#)。

## 使用提取快取規則的考量事項

使用 Amazon ECR 提取快取規則時，請考慮下列事項。

- 下列區域不支援提取快取規則的建立。
  - 中國 (北京) (cn-north-1)
  - 中國 (寧夏) (cn-northwest-1)
  - AWS GovCloud (美國東部) (us-gov-east-1)
  - AWS GovCloud (美國西部) (us-gov-west-1)
- AWS Lambda 不支援使用提取快取規則從 Amazon ECR 提取容器映像。
- 使用提取快取提取映像時，第一次提取映像時不支援 Amazon ECR FIPS 服務端點。不過，使用 Amazon ECR FIPS 服務端點可以處理後續的提取。
- 對於需要身分驗證的上游儲存庫，當第一次透過 Amazon ECR 私有登錄 URI 提取映像或更新快取時，映像提取是由與提取快取規則中設定的登入資料相關聯的使用者啟動。後續提取會直接從客戶私有登錄檔中的快取傳回映像。
- 對於不需要身分驗證的上游儲存庫，當映像透過 Amazon ECR 私有登錄 URI 提取時，映像提取是由 AWS IP 地址啟動。
- 當客戶透過 Amazon ECR 私有登錄檔 URI 提取快取映像時，Amazon ECR 會檢查過去 24 小時內是否已根據上游登錄檔驗證映像。如果 24 小時時段已過期，Amazon ECR 會在上游傳送請求，以檢查是否有較新的版本，並在快取存在時更新快取。如果視窗尚未過期，Amazon ECR 會在不聯絡上游的情況下提供快取映像。
- 呼叫 ListImageReferrers API 以提取建立的快取儲存庫會將符合 OCI 標準的參考者成品傳回私有快取。
- Amazon ECR 會檢查參考者成品是否已在過去 6 小時內更新。如果 6 小時時段已過期，Amazon ECR 會在上游傳送請求，以檢查是否有較新的版本，並在快取存在時更新快取。
- 如果 Amazon ECR 因任何原因無法從上游登錄檔更新映像，並且已提取映像，則仍會提取最後一個快取映像。
- 建立包含上游登錄檔憑證的 Secrets Manager 秘密時，秘密名稱必須使用 ecr-pullthroughcache/ 字首。秘密也必須位於在其中建立提取快取規則的相同帳戶和區域中。
- 使用提取快取規則提取多架構映像時，資訊清單和資訊清單中參照的每個映像都會提取至 Amazon ECR 儲存庫。如果只想提取特定架構，您可以使用與架構相關聯的映像摘要或標籤 (而不是與資訊清單相關聯的標籤) 來提取映像。

- Amazon ECR 使用服務連結 IAM 角色，該角色提供 Amazon ECR 建立儲存庫、擷取用於身分驗證的 Secrets Manager 秘密值，和代表您推送快取映像所需的許可。建立提取快取規則時，會自動建立服務連結 IAM 角色。如需詳細資訊，請參閱[用於提取快取的 Amazon ECR 服務連結角色](#)。
- 依預設，提取快取映像的 IAM 主體具有透過其 IAM 政策授予他們的許可。您可以使用 Amazon ECR 私有登錄檔許可政策，進一步設定 IAM 實體的許可範圍。如需詳細資訊，請參閱[使用登錄檔許可](#)。
- 使用提取快取工作流程建立的 Amazon ECR 儲存庫，會被視作與其他 Amazon ECR 儲存庫一樣。支援所有儲存庫功能，例如複寫和映像掃描。
- Amazon ECR 代表您使用提取快取動作建立新儲存庫時，下列預設設定會套用至儲存庫，除非有相符的儲存庫建立範本。您可以使用儲存庫建立範本，定義套用至 Amazon ECR 代表您建立的儲存庫的設定。如需詳細資訊，請參閱[用於控制提取快取、推送時建立或複寫動作期間建立的儲存庫的範本](#)。
  - 標籤不可變性 – 標籤不可變性指定是否可以覆寫影像標籤。根據預設，影像標籤是可變的（可以覆寫）。您可以在選取 Mutable 時，透過在 Mutable 標籤排除文字方塊中設定標籤排除篩選條件來修改標籤行為，或在選取 Immutable 時設定 Immutable 標籤排除文字方塊。
  - 加密 – 使用預設AES256加密。
  - 儲存庫許可 – 已省略，不會套用儲存庫許可政策。
  - 生命週期政策 – 已省略，未套用生命週期政策。
  - 資源標籤 – 已省略，不會套用資源標籤。
- 使用提取快取規則為儲存庫開啟映像標籤不變性，可防止 Amazon ECR 使用相同標籤更新映像。
- 第一次可能需要路由到網際網路時，使用提取快取規則提取映像時。在某些情況下，需要路由至網際網路，因此最好設定路由以避免任何失敗。因此，如果您已使用將 Amazon ECR 設定為使用介面 VPC 端點 AWS PrivateLink，則需要確保第一次提取具有網際網路的路由。其中一種方法是使用網際網路閘道在相同的 VPC 中建立公有子網路，然後將所有傳出流量從其私有子網路路由到公有子網路。使用提取快取規則提取的後續映像不需要此項目。如需詳細資訊，請參閱《Amazon Virtual Private Cloud 使用者指南》中的[路由選項範例](#)。

## 將上游登錄檔與 Amazon ECR 私有登錄檔同步所需的 IAM 許可

除了向私有登錄檔進行身分驗證以及推送和提取映像所需的 Amazon ECR API 許可之外，還需要以下額外許可才能有效使用提取快取規則。

- `ecr:CreatePullThroughCacheRule` – 授予建立提取快取規則的許可。必須透過身分型 IAM 政策授予此許可。

- `ecr:BatchImportUpstreamImage` – 授予檢索外部映像並將其匯入到您的私有登錄檔的許可。可以藉由使用身分型 IAM 政策的私有登錄檔許可政策或藉由使用資源型儲存庫許可政策來授予此許可。如需使用儲存庫許可的詳細資訊，請參閱 [Amazon ECR 中的私有儲存庫政策](#)。
- `ecr:CreateRepository` – 授予在私有登錄檔中建立儲存庫的許可。如果存放快取映像的儲存庫尚不存在，則需要此許可。可以由身分型 IAM 政策或私有登錄檔許可政策授予此許可。

## 使用登錄檔許可

Amazon ECR 私有登錄檔許可可用來設定個別 IAM 實體使用提取快取的許可範圍。如果 IAM 實體擁有的由 IAM 政策授予的許可多過登錄檔許可政策授予的許可，則 IAM 政策優先。例如，如果使用者已具有 `ecr:*` 許可，則在登錄檔層級不需要額外的許可。

### 建立私有登錄檔的許可政策 (AWS 管理主控台)

1. 在 <https://console.aws.amazon.com/ecr/> 開啟 Amazon ECR 主控台。
2. 從導覽列選擇要在其中設定私有登錄檔許可陳述式的區域。
3. 在導覽窗格中，選擇 Private registry (私有登錄檔)、Registry permissions (登錄檔許可)。
4. 在 Registry permissions (登錄檔許可) 頁面上，選擇 Generate statement (產生陳述式)。
5. 針對您要建立的每個提取快取許可政策陳述式，執行下列動作。
  - a. 針對 Policy type (政策類型)，選擇 Pull through cache policy (提取快取政策)。
  - b. 針對 Statement id (陳述式 ID)，提供提取快取陳述式政策的名稱。
  - c. 針對 IAM entities (IAM 實體)，指定要包含在政策中的使用者、群組或角色。
  - d. 針對 Repository namespace (儲存庫命名空間)，選取要與政策建立關聯的提取快取規則。
  - e. 針對 Repository names (儲存庫名稱)，指定要套用規則的儲存庫基本名稱。例如，如果您想要在 Amazon ECR Public 上指定 Amazon Linux 儲存庫，則儲存庫名稱會是 `amazonlinux`。

### 建立私有登錄檔的許可政策 (AWS CLI)

使用以下 AWS CLI 命令，使用指定私有登錄檔許可 AWS CLI。

1. 建立具有您登錄檔政策內容的名為 `ptc-registry-policy.json` 的本機檔案。下列範例會授予 `ecr-pull-through-cache-user` 許可，以建立儲存庫並從 Amazon ECR Public 中提取映像，其為與之前建立的提取快取規則相關聯的上游來源。

## JSON

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Sid": "PullThroughCacheFromReadOnlyRole",
 "Effect": "Allow",
 "Principal": {
 "AWS": "arn:aws:iam::111122223333:user/ecr-pull-through-cache-user"
 },
 "Action": [
 "ecr:CreateRepository",
 "ecr:BatchImportUpstreamImage"
],
 "Resource": "arn:aws:ecr:us-east-1:111122223333:repository/ecr-public/
*"
 }
]
}
```

**⚠ Important**

僅當存放快取映像的儲存庫尚不存在時，才需要 `ecr:CreateRepository` 許可。例如，如果儲存庫建立動作和映像提取動作是由不同的 IAM 主體 (例如管理員和開發人員) 所執行。

2. 使用 `put-registry-policy` 命令設定登錄檔政策。

```
aws ecr put-registry-policy \
 --policy-text file://ptc-registry.policy.json
```

## 後續步驟

準備好開始使用提取快取規則時，以下為接下來的步驟。

- 建立提取快取規則。如需詳細資訊，請參閱 [在 Amazon ECR 中建立提取快取規則](#)。

- 建立儲存庫建立範本。儲存庫建立範本賦予您控制權，於提取快取動作期間，定義用於 Amazon ECR 代表您建立的新儲存庫的設定。如需詳細資訊，請參閱[用於控制提取快取、推送時建立或複寫動作期間建立的儲存庫的範本](#)。

## 設定跨帳戶 ECR 到 ECR PTC 的許可

Amazon ECR 到 Amazon ECR (ECR 到 ECR) 提取快取功能可在區域、AWS 帳戶或兩者之間自動同步映像。使用 ECR 到 ECR PTC，您可以將映像推送到主要 Amazon ECR 登錄檔，並設定提取快取規則來快取下游 Amazon ECR 登錄檔中的映像。

### 跨帳戶 ECR 到 ECR 提取快取所需的 IAM 政策

若要跨不同 AWS 帳戶快取 Amazon ECR 登錄檔之間的映像，請在下游帳戶中建立 IAM 角色，並設定本節中的政策以提供下列許可：

- Amazon ECR 需要許可，才能代表您從上游 Amazon ECR 登錄檔提取映像。您可以透過建立 IAM 角色，然後在提取快取規則中指定這些角色來授予這些許可。
- 上游登錄擁有者也必須授予快取登錄擁有者必要的許可，才能將映像提取至資源政策。

#### 政策

- [建立 IAM 角色以定義提取快取許可](#)
- [為 IAM 角色建立信任政策](#)
- [在上游 Amazon ECR 登錄檔中建立資源政策](#)

### 建立 IAM 角色以定義提取快取許可

下列範例顯示許可政策，授予 IAM 角色代表您從上游 Amazon ECR 登錄檔提取映像的許可。當 Amazon ECR 擔任該角色時，會收到此政策中指定的許可。

#### JSON

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Sid": "VisualEditor0",
```

```

 "Effect": "Allow",
 "Action": [
 "ecr:GetDownloadUrlForLayer",
 "ecr:GetAuthorizationToken",
 "ecr:BatchImportUpstreamImage",
 "ecr:BatchGetImage",
 "ecr:GetImageCopyStatus",
 "ecr:InitiateLayerUpload",
 "ecr:UploadLayerPart",
 "ecr:CompleteLayerUpload",
 "ecr:PutImage"
],
 "Resource": "*"
}
]
}

```

## 為 IAM 角色建立信任政策

下列範例顯示信任政策，將 Amazon ECR 提取快取識別為可擔任該角色 AWS 的服務主體。

### JSON

```

{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Effect": "Allow",
 "Principal": {
 "Service": "pullthroughcache.ecr.amazonaws.com"
 },
 "Action": "sts:AssumeRole"
 }
]
}

```

## 在上游 Amazon ECR 登錄檔中建立資源政策

上游 Amazon ECR 登錄擁有者也必須新增登錄政策或儲存庫政策，才能授予下游登錄擁有者執行下列動作所需的許可。

**Note**

只有跨帳戶 ECR 到 ECR 提取快取組態才需要下列資源政策。對於相同帳戶、跨區域提取快取，您只需要前幾節中顯示的 IAM 角色政策和信任政策。當上游和下游登錄檔位於相同帳戶時，不需要根 AWS 帳戶主體許可。

```
{
 "Effect": "Allow",
 "Principal": {
 "AWS": "arn:aws:iam::444455556666:root"
 },
 "Action": [
 "ecr:BatchGetImage",
 "ecr:GetDownloadUrlForLayer",
 "ecr:BatchImportUpstreamImage",
 "ecr:GetImageCopyStatus"
],
 "Resource": "arn:aws:ecr:region:111122223333:repository/*"
}
```

## 在 Amazon ECR 中建立提取快取規則

對於每個包含您想要在 Amazon ECR 私有登錄檔中快取之映像的上游登錄檔，您必須建立提取快取規則。

對於需要使用秘密進行身分驗證的上游登錄檔，您必須將登入資料存放在 Secrets Manager 秘密中。您可以使用現有的秘密或建立新的秘密。您可以在 Amazon ECR 主控台或 Secrets Manager 主控台中建立 Secrets Manager 秘密。若要使用 Secrets Manager 主控台而非 Amazon ECR 主控台建立 Secrets Manager 秘密，請參閱 [將上游儲存庫登入資料存放在 AWS Secrets Manager 秘密中](#)。

### 先決條件

- 確認您具有適當的 IAM 許可來建立提取快取規則。如需相關資訊，請參閱 [將上游登錄檔與 Amazon ECR 私有登錄檔同步所需的 IAM 許可](#)。
- 對於需要使用秘密進行身分驗證的上游登錄：如果您想要使用現有的秘密，請確認 Secrets Manager 秘密符合下列要求：
  - 秘密的名稱開頭為 ecr-pullthroughcache/。AWS 管理主控台只會顯示具有 ecr-pullthroughcache/字首的 Secrets Manager 秘密。

- 秘密所在的帳戶和區域必須符合提取快取規則所在的帳戶和區域。

## 建立提取快取規則 (AWS 管理主控台)

下列步驟說明如何使用 Amazon ECR 主控台建立提取快取規則和 Secrets Manager 秘密。若要使用 Secrets Manager 主控台建立秘密，請參閱 [將上游儲存庫登入資料存放在 AWS Secrets Manager 秘密中](#)。

針對 Amazon ECR 公共，Kubernetes 容器登錄檔或 Quay

1. 在 <https://console.aws.amazon.com/ecr/> 開啟 Amazon ECR 主控台。
2. 從導覽列選擇要在其中進行私有登錄檔設定的區域。
3. 在導覽窗格中，選擇 Private registry (私有登錄檔)、Pull through cache (提取快取)。
4. 在 Pull through cache configuration (提取快取組態) 頁面上，選擇 Add rule (新增規則)。
5. 在步驟 1：指定來源頁面上，針對登錄檔，從上游登錄檔清單中選擇 Amazon ECR 公共、Kubernetes 或 Quay，接著選擇下一步。
6. 在步驟 2：指定目的地頁面上，針對 Amazon ECR 儲存庫字首，指定快取從來源公有登錄檔提取的映像時要使用的儲存庫命名空間字首，接著選擇下一步。預設會填入命名空間，但也可以指定自訂命名空間。
7. 在步驟 3：檢閱並建立頁面上，檢閱提取快取規則組態，接著選擇建立。
8. 為要建立的每個提取快取重複前面的步驟。系統會針對每個區域分別建立提取快取規則。

對於 Docker Hub

1. 在 <https://console.aws.amazon.com/ecr/> 開啟 Amazon ECR 主控台。
2. 從導覽列選擇要在其中進行私有登錄檔設定的區域。
3. 在導覽窗格中，選擇 Private registry (私有登錄檔)、Pull through cache (提取快取)。
4. 在 Pull through cache configuration (提取快取組態) 頁面上，選擇 Add rule (新增規則)。
5. 在步驟 1：指定來源頁面上，針對登錄檔選擇 Docker Hub、下一步。
6. 在步驟 2：設定身分驗證頁面上，針對上游憑證，您必須將 Docker Hub 的身分驗證憑證儲存在 AWS Secrets Manager 秘密中。您可以指定現有秘密，或使用 Amazon ECR 主控台建立新秘密。
  - a. 若要使用現有的秘密，請選擇使用現有的 AWS 秘密。針對秘密名稱，使用下拉式選單選取您現有的秘密，接著選擇下一步。

**Note**

AWS 管理主控台 只會顯示使用 `ecr-pullthroughcache/` 字首名稱的 Secrets Manager 秘密。秘密也必須位於在其中建立提取快取規則的相同帳戶和區域中。

- b. 若要建立新秘密，請選擇建立 AWS 秘密，執行下列動作，接著選擇下一步。
  - i. 針對秘密名稱，指定秘密的描述性名稱。秘密名稱必須含有 1 至 512 個 Unicode 字元。
  - ii. 針對 Docker Hub 電子郵件，指定您的 Docker Hub 電子郵件。
  - iii. 針對 Docker Hub 存取字符，請指定您的 Docker Hub 存取字符。如需建立 Docker Hub 存取字符的詳細資訊，請參閱 Docker 文件中的 [建立和管理存取字符](#)。
7. 在步驟 3：指定目的地頁面上，針對 Amazon ECR 儲存庫字首，指定快取從來源公有登錄檔提取的映像時要使用的儲存庫命名空間，接著選擇下一步。

預設會填入命名空間，但也可以指定自訂命名空間。
8. 在步驟 4：檢閱並建立頁面上，檢閱提取快取規則組態，接著選擇建立。
9. 為要建立的每個提取快取重複前面的步驟。系統會針對每個區域分別建立提取快取規則。

### 對於 GitHub Container Registry

1. 在 <https://console.aws.amazon.com/ecr/> 開啟 Amazon ECR 主控台。
2. 從導覽列選擇要在其中進行私有登錄檔設定的區域。
3. 在導覽窗格中，選擇 Private registry (私有登錄檔)、Pull through cache (提取快取)。
4. 在 Pull through cache configuration (提取快取組態) 頁面上，選擇 Add rule (新增規則)。
5. 在步驟 1：指定來源頁面上，針對登錄檔選擇 GitHub Container Registry、下一步。
6. 在步驟 2：設定身分驗證頁面上，針對上游憑證，您必須將 GitHub Container Registry 的身分驗證憑證儲存在 AWS Secrets Manager 秘密中。您可以指定現有秘密，或使用 Amazon ECR 主控台建立新秘密。
  - a. 若要使用現有的秘密，請選擇使用現有的 AWS 秘密。針對秘密名稱，使用下拉式選單選取您現有的秘密，接著選擇下一步。

**Note**

AWS 管理主控台 只會顯示使用 `ecr-pullthroughcache/` 字首名稱的 Secrets Manager 秘密。秘密也必須位於在其中建立提取快取規則的相同帳戶和區域中。

- b. 若要建立新秘密，請選擇建立 AWS 秘密，執行下列動作，接著選擇下一步。
  - i. 針對秘密名稱，指定秘密的描述性名稱。秘密名稱必須含有 1 至 512 個 Unicode 字元。
  - ii. 針對 GitHub Container Registry 使用者名稱，請指定您的 GitHub Container Registry 使用者名稱。
  - iii. 對於 GitHub Container Registry 存取字符，請指定您的 GitHub Container Registry 存取字符。如需建立 GitHub 存取字符的詳細資訊，請參閱 GitHub 文件中的 [管理個人存取權字符](#)。
7. 在步驟 3：指定目的地頁面上，針對 Amazon ECR 儲存庫字首，指定快取從來源公有登錄檔提取的映像時要使用的儲存庫命名空間，接著選擇下一步。

預設會填入命名空間，但也可以指定自訂命名空間。

8. 在步驟 4：檢閱並建立頁面上，檢閱提取快取規則組態，接著選擇建立。
9. 為要建立的每個提取快取重複前面的步驟。系統會針對每個區域分別建立提取快取規則。

### 對於 Microsoft Azure Container Registry

1. 在 <https://console.aws.amazon.com/ecr/> 開啟 Amazon ECR 主控台。
2. 從導覽列選擇要在其中進行私有登錄檔設定的區域。
3. 在導覽窗格中，選擇 Private registry (私有登錄檔)、Pull through cache (提取快取)。
4. 在 Pull through cache configuration (提取快取組態) 頁面上，選擇 Add rule (新增規則)。
5. 在步驟 1：指定來源頁面上，執行下列動作。
  - a. 針對登錄檔，選擇 Microsoft Azure Container Registry
  - b. 針對來源登錄檔 URL，指定 Microsoft Azure Container Registry 的名稱，接著選擇下一步。

**Important**

您只需要指定字首，因為會代表您填入 `.azurecr.io` 字尾。

6. 在步驟 2：設定身分驗證頁面上，針對上游憑證，您必須將 Microsoft Azure Container Registry 的身分驗證憑證儲存在 AWS Secrets Manager 秘密中。您可以指定現有秘密，或使用 Amazon ECR 主控台建立新秘密。
    - a. 若要使用現有的秘密，請選擇使用現有的 AWS 秘密。針對秘密名稱，使用下拉式選單選取您現有的秘密，接著選擇下一步。
-  **Note**

AWS 管理主控台 只會顯示使用 `ecr-pullthroughcache/` 字首名稱的 Secrets Manager 秘密。秘密也必須位於在其中建立提取快取規則的相同帳戶和區域中。
- b. 若要建立新秘密，請選擇建立 AWS 秘密，執行下列動作，接著選擇下一步。
      - i. 針對秘密名稱，指定秘密的描述性名稱。秘密名稱必須含有 1 至 512 個 Unicode 字元。
      - ii. 對於 Microsoft Azure Container Registry 使用者名稱，請指定您的 Microsoft Azure Container Registry 使用者名稱。
      - iii. 對於 Microsoft Azure Container Registry 存取字符，請指定您的 Microsoft Azure Container Registry 存取字符。如需建立 Microsoft Azure Container Registry 存取字符的詳細資訊，請參閱 Microsoft Azure 文件中的 [建立字符 - 入口網站](#)。
  7. 在步驟 3：指定目的地頁面上，針對 Amazon ECR 儲存庫字首，指定快取從來源公有登錄檔提取的映像時要使用的儲存庫命名空間，接著選擇下一步。

預設會填入命名空間，但也可以指定自訂命名空間。

8. 在步驟 4：檢閱並建立頁面上，檢閱提取快取規則組態，接著選擇建立。
9. 為要建立的每個提取快取重複前面的步驟。系統會針對每個區域分別建立提取快取規則。

#### 對於 GitLab 容器登錄檔

1. 在 <https://console.aws.amazon.com/ecr/> 開啟 Amazon ECR 主控台。
2. 從導覽列選擇要在其中進行私有登錄檔設定的區域。
3. 在導覽窗格中，選擇 Private registry (私有登錄檔)、Pull through cache (提取快取)。
4. 在 Pull through cache configuration (提取快取組態) 頁面上，選擇 Add rule (新增規則)。
5. 在步驟 1：指定來源頁面上，針對登錄檔，選擇 GitLab Container Registry，下一步。

6. 在步驟 2：設定身分驗證頁面上，對於上游憑證，您必須將 GitLab Container Registry 的身分驗證憑證存放在秘密 AWS Secrets Manager 中。您可以指定現有秘密，或使用 Amazon ECR 主控台建立新秘密。
  - a. 若要使用現有的秘密，請選擇使用現有的 AWS 秘密。針對秘密名稱，使用下拉式選單選取您現有的秘密，接著選擇下一步。如需有關使用 Secrets Manager 主控台建立 Secrets Manager 秘密的詳細資訊，請參閱 [將上游儲存庫登入資料存放在 AWS Secrets Manager 秘密中](#)。
  - b. 若要建立新秘密，請選擇建立 AWS 秘密，執行下列動作，接著選擇下一步。
    - i. 針對秘密名稱，指定秘密的描述性名稱。秘密名稱必須含有 1 至 512 個 Unicode 字元。
    - ii. 針對 GitLab Container Registry 使用者名稱，指定您的 GitLab Container Registry 使用者名稱。
    - iii. 對於 GitLab Container Registry 存取權杖，指定您的 GitLab Container Registry 存取權杖。如需建立 GitLab 容器登錄檔存取字符的詳細資訊，請參閱 GitLab 文件中的 [個人存取字符](#)、[群組存取字符](#) 或 [專案存取字符](#)。
7. 在步驟 3：指定目的地頁面上，針對 Amazon ECR 儲存庫字首，指定快取從來源公有登錄檔提取的映像時要使用的儲存庫命名空間，接著選擇下一步。


預設會填入命名空間，但也可以指定自訂命名空間。

8. 在步驟 4：檢閱並建立頁面上，檢閱提取快取規則組態，接著選擇建立。
9. 為要建立的每個提取快取重複前面的步驟。系統會針對每個區域分別建立提取快取規則。

對於您 AWS 帳戶中的 Amazon ECR 私有登錄檔

1. 在 <https://console.aws.amazon.com/ecr/> 開啟 Amazon ECR 主控台。
2. 從導覽列中，選擇您要在其中設定私有登錄檔設定的區域。
3. 在導覽窗格中，選擇 Private registry (私有登錄檔)、Pull through cache (提取快取)。
4. 在 Pull through cache configuration (提取快取組態) 頁面上，選擇 Add rule (新增規則)。

5. 在步驟 1：指定上游頁面上，針對登錄檔，選擇 Amazon ECR Private 和此帳戶。針對區域，選取上游 Amazon ECR 登錄檔的區域，然後選擇下一步。
6. 在步驟 2：指定命名空間頁面上，針對快取命名空間，選擇是否建立具有特定字首或無字首的提取快取儲存庫。如果您選取特定字首，則必須指定字首名稱，做為從上游登錄檔快取映像的命名空間的一部分。
7. 針對上游命名空間，選擇是否從上游登錄檔中存在的特定字首提取。如果未選取字首，您可以從上游登錄檔中的任何儲存庫提取。如果出現提示，請指定上游儲存庫字首，然後選擇下一步。


 Note

若要進一步了解自訂快取和上游命名空間，請參閱 [自訂 ECR 到 ECR 提取快取的儲存庫字首](#)。

8. 在步驟 3：檢閱並建立頁面上，檢閱提取快取規則組態，接著選擇建立。
9. 針對您要建立的每個提取快取重複這些步驟。系統會針對每個區域分別建立提取快取規則。

對於來自另一個 AWS 帳戶的 Amazon ECR 私有登錄


1. 在 <https://console.aws.amazon.com/ecr/> 開啟 Amazon ECR 主控台。
2. 從導覽列選擇要在其中進行私有登錄檔設定的區域。
3. 在導覽窗格中，選擇 Private registry (私有登錄檔)、Pull through cache (提取快取)。
4. 在 Pull through cache configuration (提取快取組態) 頁面上，選擇 Add rule (新增規則)。
5. 在步驟 1：指定上游頁面上，針對登錄檔，選擇 Amazon ECR 私有和跨帳戶。針對區域，選取上游 Amazon ECR 登錄檔的區域。針對帳戶，指定上游 Amazon ECR 登錄 AWS 檔的帳戶 ID，然後選擇下一步。
6. 在步驟 2：指定許可頁面上，針對 IAM 角色，選取要用於跨帳戶提取快取存取的角色，然後選擇建立。

 Note

請務必選取使用在 中建立之許可的 IAM 角色 [跨帳戶 ECR 到 ECR 提取快取所需的 IAM 政策](#)。

7. 在步驟 3：指定命名空間頁面上，針對快取命名空間，選擇是否建立具有特定字首或無字首的提取快取儲存庫。如果您選取特定字首，則必須指定字首名稱，做為從上游登錄檔快取映像的命名空間的一部分。

- 針對上游命名空間，選擇是否從上游登錄檔中存在的特定字首提取。如果未選取字首，您可以從上游登錄檔中的任何儲存庫提取。如果出現提示，請指定上游儲存庫字首，然後選擇下一步。


 Note

若要進一步了解自訂快取和上游命名空間，請參閱 [自訂 ECR 到 ECR 提取快取的儲存庫字首](#)。

- 在步驟 4：檢閱並建立頁面上，檢閱提取快取規則組態，接著選擇建立。
- 針對您要建立的每個提取快取重複這些步驟。系統會針對每個區域分別建立提取快取規則。

### 對於 Chainguard 登錄檔

- 在 <https://console.aws.amazon.com/ecr/> 開啟 Amazon ECR 主控台。
- 從導覽列選擇要在其中進行私有登錄檔設定的區域。
- 在導覽窗格中，選擇 Private registry (私有登錄檔)、Pull through cache (提取快取)。
- 在 Pull through cache configuration (提取快取組態) 頁面上，選擇 Add rule (新增規則)。
- 在步驟 1：指定來源頁面上，針對登錄檔，選擇 Chainguard 登錄檔，下一步。
- 在步驟 2：設定身分驗證頁面上，對於上游憑證，您必須將 Chainguard Registry 的身分驗證憑證存放在秘密 AWS Secrets Manager 中。您可以指定現有秘密，或使用 Amazon ECR 主控台建立新秘密。
  - 若要使用現有的秘密，請選擇使用現有的 AWS 秘密。針對秘密名稱，使用下拉式選單選取您現有的秘密，接著選擇下一步。如需有關使用 Secrets Manager 主控台建立 Secrets Manager 秘密的詳細資訊，請參閱 [將上游儲存庫登入資料存放在 AWS Secrets Manager 秘密中](#)。

 Note

AWS 管理主控台 只會顯示使用 ecr-pullthroughcache/字首名稱的 Secrets Manager 秘密。秘密也必須位於在其中建立提取快取規則的相同帳戶和區域中。

- 若要建立新秘密，請選擇建立 AWS 秘密，執行下列動作，接著選擇下一步。
  - 針對秘密名稱，指定秘密的描述性名稱。秘密名稱必須含有 1 至 512 個 Unicode 字元。
  - 針對 Chainguard 登錄檔使用者名稱，指定您的 Chainguard 登錄檔使用者名稱。

- iii. 針對 Chainguard Registry 提取字符，指定您的 Chainguard Registry 提取字符。如需建立 Chainguard 登錄檔提取字符的詳細資訊，請參閱 Chainguard 文件中的[使用提取字符進行驗證](#)。
7. 在步驟 3：指定目的地頁面上，針對 Amazon ECR 儲存庫字首，指定快取從來源登錄檔提取的影像時要使用的儲存庫命名空間，然後選擇下一步。

預設會填入命名空間，但也可以指定自訂命名空間。
8. 在步驟 4：檢閱並建立頁面上，檢閱提取快取規則組態，接著選擇建立。
9. 為要建立的每個提取快取重複前面的步驟。系統會針對每個區域分別建立提取快取規則。

## 建立提取快取規則 (AWS CLI)

使用 [create-pull-through-cache-rule](#) AWS CLI 命令來建立 Amazon ECR 私有登錄檔的提取快取規則。對於需要使用秘密進行身分驗證的上游登錄檔，您必須將登入資料存放在 Secrets Manager 秘密中。若要使用 Secrets Manager 主控台建立秘密，請參閱 [將上游儲存庫登入資料存放在 AWS Secrets Manager 秘密中](#)。

提供給每個受支援的登錄檔的下列範例。

對於 Amazon ECR Public

下列範例會為 Amazon ECR Public 登錄檔建立提取快取規則。其會指定 `ecr-public` 的儲存庫字首，這會導致使用提取快取規則建立的每個儲存庫具有 `ecr-public/upstream-repository-name` 的命名規則。

```
aws ecr create-pull-through-cache-rule \
 --ecr-repository-prefix ecr-public \
 --upstream-registry-url public.ecr.aws \
 --region us-east-2
```

對於 Kubernetes Container Registry

下列範例會為 Kubernetes 公有登錄檔建立提取快取規則。其會指定 `kubernetes` 的儲存庫字首，這會導致使用提取快取規則建立的每個儲存庫具有 `kubernetes/upstream-repository-name` 的命名規則。

```
aws ecr create-pull-through-cache-rule \
 --ecr-repository-prefix kubernetes \
 --upstream-registry-url public.ecr.aws \
 --region us-east-2
```

```
--ecr-repository-prefix kubernetes \
--upstream-registry-url registry.k8s.io \
--region us-east-2
```

## 對於 Quay

下列範例會為 Quay 公有登錄檔建立提取快取規則。其會指定 quay 的儲存庫字首，這會導致使用提取快取規則建立的每個儲存庫具有 quay/*upstream-repository-name* 的命名規則。

```
aws ecr create-pull-through-cache-rule \
 --ecr-repository-prefix quay \
 --upstream-registry-url quay.io \
 --region us-east-2
```

## 對於 Docker Hub

下列範例會為 Docker Hub 登錄檔建立提取快取規則。其會指定 docker-hub 的儲存庫字首，這會導致使用提取快取規則建立的每個儲存庫具有 docker-hub/*upstream-repository-name* 的命名規則。您必須指定秘密包含 Docker Hub 憑證的完整 Amazon Resource Name (ARN)。

```
aws ecr create-pull-through-cache-rule \
 --ecr-repository-prefix docker-hub \
 --upstream-registry-url registry-1.docker.io \
 --credential-arn arn:aws:secretsmanager:us-east-2:111122223333:secret:ecr-
pullthroughcache/example1234 \
 --region us-east-2
```

## 對於 GitHub Container Registry

下列範例會建立 GitHub Container Registry 的提取快取規則。其會指定 github 的儲存庫字首，這會導致使用提取快取規則建立的每個儲存庫具有 github/*upstream-repository-name* 的命名規則。您必須指定秘密包含 GitHub Container Registry 憑證的完整 Amazon Resource Name (ARN)。

```
aws ecr create-pull-through-cache-rule \
 --ecr-repository-prefix github \
 --upstream-registry-url ghcr.io \
 --credential-arn arn:aws:secretsmanager:us-east-2:111122223333:secret:ecr-
pullthroughcache/example1234 \
 --region us-east-2
```

## 對於 Microsoft Azure Container Registry

下列範例會建立 Microsoft Azure Container Registry 的提取快取規則。其會指定 `azure` 的儲存庫字首，這會導致使用提取快取規則建立的每個儲存庫具有 `azure/upstream-repository-name` 的命名規則。您必須指定秘密包含 Microsoft Azure Container Registry 憑證的完整 Amazon Resource Name (ARN)。

```
aws ecr create-pull-through-cache-rule \
 --ecr-repository-prefix azure \
 --upstream-registry-url myregistry.azurecr.io \
 --credential-arn arn:aws:secretsmanager:us-east-2:111122223333:secret:ecr-pullthroughcache/example1234 \
 --region us-east-2
```

## 對於 GitLab 容器登錄檔

下列範例會建立 GitLab Container Registry 的提取快取規則。其會指定 `gitlab` 的儲存庫字首，這會導致使用提取快取規則建立的每個儲存庫具有 `gitlab/upstream-repository-name` 的命名規則。您必須指定包含 GitLab Container Registry 登入資料之秘密的完整 Amazon Resource Name (ARN)。

```
aws ecr create-pull-through-cache-rule \
 --ecr-repository-prefix gitlab \
 --upstream-registry-url registry.gitlab.com \
 --credential-arn arn:aws:secretsmanager:us-east-2:111122223333:secret:ecr-pullthroughcache/example1234 \
 --region us-east-2
```

## 對於您 AWS 帳戶中的 Amazon ECR 私有登錄檔

下列範例會為同一 AWS 帳戶中跨區域 Amazon ECR 私有登錄檔建立提取快取規則。其會指定 `ecr` 的儲存庫字首，這會導致使用提取快取規則建立的每個儲存庫具有 `ecr/upstream-repository-name` 的命名規則。

```
aws ecr create-pull-through-cache-rule \
 --ecr-repository-prefix ecr \
 --upstream-registry-url aws_account_id.dkr.ecr.region.amazonaws.com \
 --region us-east-2
```

對於來自另一個 AWS 帳戶的 Amazon ECR 私有登錄

下列範例會為同一 AWS 帳戶中跨區域 Amazon ECR 私有登錄檔建立提取快取規則。其會指定 `ecr` 的儲存庫字首，這會導致使用提取快取規則建立的每個儲存庫具有 `ecr/upstream-repository-name` 的命名規則。您必須使用在 中建立的許可來指定 IAM 角色的完整 Amazon Resource Name (ARN) [在 Amazon ECR 中建立提取快取規則](#)。

```
aws ecr create-pull-through-cache-rule \
 --ecr-repository-prefix ecr \
 --upstream-registry-url aws_account_id.dkr.ecr.region.amazonaws.com \
 --custom-role-arn arn:aws:iam::aws_account_id:role/example-role \
 --region us-east-2
```

對於 Chainguard 登錄檔

下列範例會建立 Chainguard Registry 的提取快取規則。其會指定 `chainguard` 的儲存庫字首，這會導致使用提取快取規則建立的每個儲存庫具有 `chainguard/upstream-repository-name` 的命名規則。您必須指定包含 Chainguard 登錄檔登入資料之秘密的完整 Amazon Resource Name (ARN)。

```
aws ecr create-pull-through-cache-rule \
 --ecr-repository-prefix chainguard \
 --upstream-registry-url cgr.dev \
 --credential-arn arn:aws:secretsmanager:us-east-2:111122223333:secret:ecr-pullthroughcache/example1234 \
 --region us-east-2
```

## 後續步驟

建立提取快取規則之後，後續步驟如下：

- 建立儲存庫建立範本。儲存庫建立範本賦予您控制權，於提取快取動作期間，定義用於 Amazon ECR 代表您建立的新儲存庫的設定。如需詳細資訊，請參閱 [用於控制提取快取、推送時建立或複寫動作期間建立的儲存庫的範本](#)。
- 驗證提取快取規則。驗證提取快取規則時，Amazon ECR 會與上游登錄檔建立網路連線、驗證其是否可存取包含上游登錄檔憑證的 Secrets Manager 秘密，以及身分驗證是否成功。如需詳細資訊，請參閱 [驗證 Amazon ECR 中的提取快取規則](#)。
- 開始使用提取快取規則。如需詳細資訊，請參閱 [在 Amazon ECR 中使用提取快取規則提取映像](#)。

## 驗證 Amazon ECR 中的提取快取規則

建立提取快取規則之後，對於需要身分驗證的上游登錄檔，您可以驗證規則是否正常運作。驗證提取快取規則時，Amazon ECR 會與上游登錄檔建立網路連線、驗證是否可以存取包含上游登錄檔登入資料的 Secrets Manager 秘密，以及驗證身分驗證是否成功。

開始使用提取快取規則之前，請確認您擁有適當的 IAM 許可。如需詳細資訊，請參閱[將上游登錄檔與 Amazon ECR 私有登錄檔同步所需的 IAM 許可](#)。

### 若要驗證提取快取規則 (AWS 管理主控台)

下列步驟說明如何使用 Amazon ECR 主控台驗證提取快取規則。

1. 在 <https://console.aws.amazon.com/ecr/> 開啟 Amazon ECR 主控台。
2. 從導覽列選擇包含提取快取規則的區域以進行驗證。
3. 在導覽窗格中，選擇 Private registry (私有登錄檔)、Pull through cache (提取快取)。
4. 在提取快取組態頁面上，選取要驗證的提取快取規則。接著，使用動作下拉式選單並選擇檢視詳細資料。
5. 在提取快取規則詳細資料頁面上，使用動作下拉式選單，然後選擇驗證身分驗證。Amazon ECR 將顯示寫著結果的橫幅。
6. 針對您要驗證的每個提取快取規則重複這些步驟。

### 若要驗證提取快取規則 (AWS CLI)

[validate-pull-through-cache-rule](#) AWS CLI 命令用於驗證 Amazon ECR 私有登錄檔的提取快取規則。

下列範例會使用 `ecr-public` 命名空間字首。以要驗證的提取快取規則的字首值取代該值。

```
aws ecr validate-pull-through-cache-rule \
 --ecr-repository-prefix ecr-public \
 --region us-east-2
```

在回應中，`isValid` 參數會指出驗證是否成功。如果出現 `true`，代表 Amazon ECR 可以連到上游登錄檔，且身分驗證成功。如果出現 `false`，代表出現問題並且驗證失敗。該 `failure` 參數會指出原因。

## 在 Amazon ECR 中使用提取快取規則提取映像

下列範例顯示使用提取快取規則來提取映像時所要使用的命令語法。如果您在使用提取快取規則提取上游映像時收到錯誤，請參閱 [對 Amazon ECR 中的提取快取問題進行故障診斷](#) 以查看最常見的錯誤以及解決方式。

開始使用提取快取規則之前，請確認您擁有適當的 IAM 許可。如需詳細資訊，請參閱 [將上游登錄檔與 Amazon ECR 私有登錄檔同步所需的 IAM 許可](#)。

### Note

下列範例使用 AWS 管理主控台 使用的預設 Amazon ECR 儲存庫命名空間值。確保使用已設定的 Amazon ECR 私有儲存庫 URI。

### 對於 Amazon ECR Public

```
docker pull aws_account_id.dkr.ecr.region.amazonaws.com/ecr-public/repository_name/
image_name:tag
```

### Kubernetes 容器登錄檔

```
docker pull aws_account_id.dkr.ecr.region.amazonaws.com/kubernetes/repository_name/
image_name:tag
```

### Quay

```
docker pull aws_account_id.dkr.ecr.region.amazonaws.com/quay/repository_name/
image_name:tag
```

### Docker Hub

針對 Docker Hub 官方映像：

```
docker pull aws_account_id.dkr.ecr.region.amazonaws.com/docker-hub/
library/image_name:tag
```

**Note**

針對 Docker Hub 官方映像，必須包含 `/library` 字首。對於所有其他 Docker Hub 儲存庫，您應該省略 `/library` 字首。

針對所有其他 Docker Hub 映像：

```
docker pull aws_account_id.dkr.ecr.region.amazonaws.com/docker-hub/repository_name/
image_name:tag
```

## GitHub Container Registry

```
docker pull aws_account_id.dkr.ecr.region.amazonaws.com/github/repository_name/
image_name:tag
```

## Microsoft Azure Container Registry

```
docker pull aws_account_id.dkr.ecr.region.amazonaws.com/azure/repository_name/
image_name:tag
```

## GitLab 容器登錄檔

```
docker pull aws_account_id.dkr.ecr.region.amazonaws.com/gitlab/repository_name/
image_name:tag
```

## Chainguard 登錄檔

```
docker pull aws_account_id.dkr.ecr.region.amazonaws.com/chainguard/repository_name/
image_name:tag
```

## 將上游儲存庫登入資料存放在 AWS Secrets Manager 秘密中

為需要身分驗證的上游儲存庫建立提取快取規則時，您必須將憑證儲存在 Secrets Manager 秘密中。使用 Secrets Manager 秘密可能需要付費。如需詳細資訊，請參閱 [AWS Secrets Manager 定價](#)。

下列程序將逐步指引您為每個支援的上游儲存庫建立 Secret Secrets Manager 秘密的方法。您可以選擇性地使用 Amazon ECR 主控台內的建立提取快取規則工作流程來建立秘密，而不是使用 Secrets Manager 主控台建立秘密。如需詳細資訊，請參閱 [在 Amazon ECR 中建立提取快取規則](#)。

## Docker Hub

若要為您的 Docker Hub 憑證建立 Secrets Manager 秘密 (AWS 管理主控台)

1. 前往以下位置開啟機密管理員控制台：<https://console.aws.amazon.com/secretsmanager/>。
2. 選擇 Store a new secret (存放新機密)。
3. 在選擇秘密類型頁面上，執行下列動作。
  - a. 針對機密類型，選擇其他類型的機密。
  - b. 在鍵值對中，為您的 Docker Hub 憑證建立兩個資料列。秘密當中最多可以存放 65536 個位元組。
    - i. 針對第一個鍵值對，請指定 username 為鍵，並指定您的 Docker Hub 使用者名稱為值。
    - ii. 針對第二個鍵值對，請指定 accessToken 為鍵，並指定您的 Docker Hub 存取字符為值。如需建立 Docker Hub 存取字符的詳細資訊，請參閱 Docker 文件中的 [建立和管理存取字符](#)。
  - c. 針對加密金鑰，請保留預設的 aws/secretsmanager AWS KMS key 值，接著選擇下一步。使用此金鑰無需任何成本。如需詳細資訊，請參閱《AWS Secrets Manager 使用者指南》中的 [Secrets Manager 中的秘密加密和解密](#)。

### Important

您必須使用預設 aws/secretsmanager 加密金鑰來加密秘密。Amazon ECR 不支援為此使用客戶自管金鑰 (CMK)。

4. 在設定秘密頁面上，執行下列動作。
  - a. 輸入描述性的 Secret name (機密名稱) 和 Description (描述)。秘密名稱必須含有 1 至 512 個 Unicode 字元，並且以 ecr-pullthroughcache/ 作為字首。

**⚠ Important**

Amazon ECR AWS 管理主控台 只會顯示使用 `ecr-pullthroughcache/` 字首名稱的 Secrets Manager 秘密。

- b. (選用) 在 Tags (標籤) 區段，將標籤新增到秘密。關於標記策略，請參閱《AWS Secrets Manager 使用者指南》中的 [標記 Secrets Manager 秘密](#)。請勿在標籤中存放敏感資訊，因為標籤並未加密。
  - c. (選用) 若要將資源政策新增至秘密，請在 Resource permissions (資源使用權限) 中選擇 Edit permissions (編輯許可)。如需詳細資訊，請參閱《AWS Secrets Manager 使用者指南》中的 [將許可政策連接至 Secrets Manager 秘密](#)。
  - d. (選用) 在複寫秘密中，若要將秘密複寫至另一個秘密 AWS 區域，請選擇複寫秘密。您可以立即複寫秘密，也可以稍後返回複寫。如需詳細資訊，請參閱《AWS Secrets Manager 使用者指南》中的 [將秘密複寫至其他地區](#)。
  - e. 選擇下一步。
5. (選用) 在 Configure rotation (設定輪換) 頁面上，可開啟自動輪換。您也可以暫時關閉輪換，稍後再將其開啟。如需詳細資訊，請參閱《AWS Secrets Manager 使用者指南》中的 [輪換 Secrets Manager 秘密](#)。選擇下一步。
  6. 在 Review (檢閱) 頁面上，檢閱機密詳細資訊，然後選擇 Store (存放)。

Secrets Manager 會傳回秘密清單。如果您的新秘密沒有顯示，請選擇重新整理按鈕。

## GitHub Container Registry

若要為 GitHub Container Registry 憑證建立 Secrets Manager 秘密 (AWS 管理主控台)


1. 前往以下位置開啟機密管理員控制台：<https://console.aws.amazon.com/secretsmanager/>。
2. 選擇 Store a new secret (存放新機密)。
3. 在選擇秘密類型頁面上，執行下列動作。
  - a. 針對機密類型，選擇其他類型的機密。
  - b. 在鍵/值對中，為您的 GitHub 憑證建立兩個資料列。秘密當中最多可以存放 65536 個位元組。
    - i. 針對第一個鍵值對，請指定 `username` 為鍵，並指定您的 GitHub 使用者名稱為值。

- ii. 針對第二個鍵值對，請指定 `accessToken` 為鍵，並指定您的 GitHub 存取字符為值。如需建立 GitHub 存取字符的詳細資訊，請參閱 GitHub 文件中的[管理個人存取權字符](#)。
- c. 針對加密金鑰，請保留預設的 `aws/secretsmanager` AWS KMS key 值，接著選擇下一步。使用此金鑰無需任何成本。如需詳細資訊，請參閱《AWS Secrets Manager 使用者指南》中的[Secrets Manager 中的秘密加密和解密](#)。

 Important

您必須使用預設 `aws/secretsmanager` 加密金鑰來加密秘密。Amazon ECR 不支援為此使用客戶自管金鑰 (CMK)。

4. 在 `Configure secret` (設定秘密) 頁面上，執行下列動作：
  - a. 輸入描述性的 `Secret name` (機密名稱) 和 `Description` (描述)。秘密名稱必須含有 1 至 512 個 Unicode 字元，並且以 `ecr-pullthroughcache/` 作為字首。

 Important

Amazon ECR AWS 管理主控台 只會顯示使用 `ecr-pullthroughcache/` 字首名稱的 `Secrets Manager` 秘密。

- b. (選用) 在 `Tags` (標籤) 區段，將標籤新增到秘密。關於標記策略，請參閱《AWS Secrets Manager 使用者指南》中的[標記 Secrets Manager 秘密](#)。請勿在標籤中存放敏感資訊，因為標籤並未加密。
  - c. (選用) 若要將資源政策新增至秘密，請在 `Resource permissions` (資源使用權限) 中選擇 `Edit permissions` (編輯許可)。如需詳細資訊，請參閱《AWS Secrets Manager 使用者指南》中的[將許可政策連接至 Secrets Manager 秘密](#)。
  - d. (選用) 在複寫秘密中，若要將秘密複寫至另一個秘密 AWS 區域，請選擇複寫秘密。您可以立即複寫秘密，也可以稍後返回複寫。如需詳細資訊，請參閱《AWS Secrets Manager 使用者指南》中的[將秘密複寫至其他地區](#)。
  - e. 選擇下一步。
5. (選用) 在 `Configure rotation` (設定輪換) 頁面上，可開啟自動輪換。您也可以暫時關閉輪換，稍後再將其開啟。如需詳細資訊，請參閱《AWS Secrets Manager 使用者指南》中的[輪換 Secrets Manager 秘密](#)。選擇下一步。
  6. 在 `Review` (檢閱) 頁面上，檢閱機密詳細資訊，然後選擇 `Store` (存放)。

Secrets Manager 會傳回秘密清單。如果您的新秘密沒有顯示，請選擇重新整理按鈕。

## Microsoft Azure Container Registry

若要為您的 Microsoft Azure Container Registry 憑證建立 Secrets Manager 秘密 (AWS 管理主控台)

1. 前往以下位置開啟機密管理員控制台：<https://console.aws.amazon.com/secretsmanager/>。
2. 選擇 Store a new secret (存放新機密)。
3. 在選擇秘密類型頁面上，執行下列動作。
  - a. 針對機密類型，選擇其他類型的機密。
  - b. 在鍵/值對中，為您的 Microsoft Azure 憑證建立兩個資料列。秘密當中最多可以存放 65536 個位元組。
    - i. 針對第一個鍵值對，請指定 username 為鍵，並指定您的 Microsoft Azure Container Registry 使用者名稱為值。
    - ii. 針對第二個鍵值對，請指定 accessToken 為鍵，並指定您的 Microsoft Azure Container Registry 存取字符為值。如需建立 Microsoft Azure 存取字符的詳細資訊，請參閱 Microsoft Azure 文件中的[建立字符 - 入口網站](#)。
  - c. 針對加密金鑰，請保留預設的 aws/secretsmanager AWS KMS key 值，接著選擇下一步。使用此金鑰無需任何成本。如需詳細資訊，請參閱《AWS Secrets Manager 使用者指南》中的[Secrets Manager 中的秘密加密和解密](#)。

### Important

您必須使用預設 aws/secretsmanager 加密金鑰來加密秘密。Amazon ECR 不支援為此使用客戶自管金鑰 (CMK)。

4. 在 Configure secret (設定秘密) 頁面上，執行下列動作：
  - a. 輸入描述性的 Secret name (機密名稱) 和 Description (描述)。秘密名稱必須含有 1 至 512 個 Unicode 字元,並且以 ecr-pullthroughcache/ 作為字首。

**⚠ Important**

Amazon ECR AWS 管理主控台 只會顯示使用 `ecr-pullthroughcache/` 字首名稱的 Secrets Manager 秘密。

- b. (選用) 在 Tags (標籤) 區段，將標籤新增到秘密。關於標記策略，請參閱《AWS Secrets Manager 使用者指南》中的 [標記 Secrets Manager 秘密](#)。請勿在標籤中存放敏感資訊，因為標籤並未加密。
  - c. (選用) 若要將資源政策新增至秘密，請在 Resource permissions (資源使用權限) 中選擇 Edit permissions (編輯許可)。如需詳細資訊，請參閱《AWS Secrets Manager 使用者指南》中的 [將許可政策連接至 Secrets Manager 秘密](#)。
  - d. (選用) 在複寫秘密中，若要將秘密複寫至另一個秘密 AWS 區域，請選擇複寫秘密。您可以立即複寫秘密，也可以稍後返回複寫。如需詳細資訊，請參閱《AWS Secrets Manager 使用者指南》中的 [將秘密複寫至其他地區](#)。
  - e. 選擇下一步。
5. (選用) 在 Configure rotation (設定輪換) 頁面上，可開啟自動輪換。您也可以暫時關閉輪換，稍後再將其開啟。如需詳細資訊，請參閱《AWS Secrets Manager 使用者指南》中的 [輪換 Secrets Manager 秘密](#)。選擇下一步。
  6. 在 Review (檢閱) 頁面上，檢閱機密詳細資訊，然後選擇 Store (存放)。


Secrets Manager 會傳回秘密清單。如果您的新秘密沒有顯示，請選擇重新整理按鈕。

## GitLab Container Registry

為您的 GitLab Container Registry 登入資料建立 Secrets Manager 秘密 (AWS 管理主控台)


1. 前往以下位置開啟機密管理員控制台：<https://console.aws.amazon.com/secretsmanager/>。
2. 選擇 Store a new secret (存放新機密)。
3. 在選擇秘密類型頁面上，執行下列動作。
  - a. 針對機密類型，選擇其他類型的機密。
  - b. 在金鑰/值對中，為您的 GitLab 登入資料建立兩列。秘密當中最多可以存放 65536 個位元組。
    - i. 對於第一個金鑰/值對，將指定 username 為金鑰，將 GitLab Container Registry 使用者名稱指定為值。

- ii. 對於第二個金鑰/值對，請將指定 `accessToken` 為金鑰，並將 GitLab Container Registry 存取權杖指定為值。如需建立 GitLab 容器登錄檔存取字符的詳細資訊，請參閱 GitLab 文件中的 [個人存取字符](#)、[群組存取字符](#) 或 [專案存取字符](#)。
- c. 針對加密金鑰，請保留預設的 `aws/secretsmanager` AWS KMS key 值，接著選擇下一步。使用此金鑰無需任何成本。如需詳細資訊，請參閱《AWS Secrets Manager 使用者指南》中的 [Secrets Manager 中的秘密加密和解密](#)。

 Important

您必須使用預設 `aws/secretsmanager` 加密金鑰來加密秘密。Amazon ECR 不支援為此使用客戶自管金鑰 (CMK)。

4. 在 `Configure secret` (設定秘密) 頁面上，執行下列動作：
  - a. 輸入描述性的 `Secret name` (機密名稱) 和 `Description` (描述)。秘密名稱必須含有 1 至 512 個 Unicode 字元，並且以 `ecr-pullthroughcache/` 作為字首。

 Important

Amazon ECR AWS 管理主控台 只會顯示使用 `ecr-pullthroughcache/` 字首名稱的 `Secrets Manager` 秘密。

- b. (選用) 在 `Tags` (標籤) 區段，將標籤新增到秘密。關於標記策略，請參閱《AWS Secrets Manager 使用者指南》中的 [標記 Secrets Manager 秘密](#)。請勿在標籤中存放敏感資訊，因為標籤並未加密。
  - c. (選用) 若要將資源政策新增至秘密，請在 `Resource permissions` (資源使用權限) 中選擇 `Edit permissions` (編輯許可)。如需詳細資訊，請參閱《AWS Secrets Manager 使用者指南》中的 [將許可政策連接至 Secrets Manager 秘密](#)。
  - d. (選用) 在複寫秘密中，若要將秘密複寫至另一個秘密 AWS 區域，請選擇複寫秘密。您可以立即複寫秘密，也可以稍後返回複寫。如需詳細資訊，請參閱《AWS Secrets Manager 使用者指南》中的 [將秘密複寫至其他地區](#)。
  - e. 選擇下一步。
5. (選用) 在 `Configure rotation` (設定輪換) 頁面上，可開啟自動輪換。您也可以暫時關閉輪換，稍後再將其開啟。如需詳細資訊，請參閱《AWS Secrets Manager 使用者指南》中的 [輪換 Secrets Manager 秘密](#)。選擇下一步。
6. 在 `Review` (檢閱) 頁面上，檢閱機密詳細資訊，然後選擇 `Store` (存放)。

Secrets Manager 會傳回秘密清單。如果您的新秘密沒有顯示，請選擇重新整理按鈕。

## Chainguard Registry

為您的 Chainguard 登入資料建立 Secrets Manager 秘密 (AWS 管理主控台)

1. 前往以下位置開啟機密管理員控制台：<https://console.aws.amazon.com/secretsmanager/>。
2. 選擇 Store a new secret (存放新機密)。
3. 在選擇秘密類型頁面上，執行下列動作。
  - a. 針對機密類型，選擇其他類型的機密。
  - b. 在金鑰/值對中，為您的 Chainguard 登入資料建立兩列。秘密當中最多可以存放 65536 個位元組。
    - i. 對於第一個金鑰/值對，將指定username為金鑰，將 Chainguard Registry 使用者名稱指定為值。
    - ii. 對於第二個金鑰/值對，將指定accessToken為金鑰，將 Chainguard Registry 存取權杖指定為值。如需建立 Chainguard 登錄檔提取字符的詳細資訊，請參閱 Chainguard 文件中的[使用提取字符進行驗證](#)。
  - c. 針對加密金鑰，請保留預設的 aws/secretsmanager AWS KMS key 值，接著選擇下一步。使用此金鑰無需任何成本。如需詳細資訊，請參閱《AWS Secrets Manager 使用者指南》中的[Secrets Manager 中的秘密加密和解密](#)。

### Important

您必須使用預設 aws/secretsmanager 加密金鑰來加密秘密。Amazon ECR 不支援為此使用客戶自管金鑰 (CMK)。

4. 在 Configure secret (設定秘密) 頁面上，執行下列動作：
  - a. 輸入描述性的 Secret name (機密名稱) 和 Description (描述)。秘密名稱必須含有 1 至 512 個 Unicode 字元,並且以 ecr-pullthroughcache/ 作為字首。

**⚠ Important**

Amazon ECR AWS 管理主控台 只會顯示使用 `ecr-pullthroughcache/` 字首名稱的 Secrets Manager 秘密。

- b. (選用) 在 Tags (標籤) 區段，將標籤新增到秘密。關於標記策略，請參閱《AWS Secrets Manager 使用者指南》中的[標記 Secrets Manager 秘密](#)。請勿在標籤中存放敏感資訊，因為標籤並未加密。
  - c. (選用) 若要將資源政策新增至秘密，請在 Resource permissions (資源使用權限) 中選擇 Edit permissions (編輯許可)。如需詳細資訊，請參閱《AWS Secrets Manager 使用者指南》中的[將許可政策連接至 Secrets Manager 秘密](#)。
  - d. (選用) 在複寫秘密中，若要將秘密複寫至另一個秘密 AWS 區域，請選擇複寫秘密。您可以立即複寫秘密，也可以稍後返回複寫。如需詳細資訊，請參閱《AWS Secrets Manager 使用者指南》中的[將秘密複寫至其他地區](#)。
  - e. 選擇下一步。
5. (選用) 在 Configure rotation (設定輪換) 頁面上，可開啟自動輪換。您也可以暫時關閉輪換，稍後再將其開啟。如需詳細資訊，請參閱《AWS Secrets Manager 使用者指南》中的[輪換 Secrets Manager 秘密](#)。選擇下一步。
  6. 在 Review (檢閱) 頁面上，檢閱機密詳細資訊，然後選擇 Store (存放)。

Secrets Manager 會傳回秘密清單。如果您的新秘密沒有顯示，請選擇重新整理按鈕。

## 自訂 ECR 到 ECR 提取快取的儲存庫字首

提取快取規則支援 `ecr` 儲存庫字首和上游儲存庫字首。`ecr` 儲存庫字首是與規則相關聯的 Amazon ECR 快取登錄檔中的儲存庫命名空間字首。使用此字首的所有儲存庫都會針對規則中定義的上游登錄檔提取啟用快取的儲存庫。例如，字首 `prod` 適用於以 `prod/` 開頭的所有儲存庫。若要將範本套用至登錄檔中沒有相關聯提取快取規則的所有儲存庫，請使用 `ROOT` 做為字首。

**⚠ Important**

總會有一個假設 / 套用至字首的結尾。如果您指定 `ecr-public` 為字首，Amazon ECR 會將其視為 `ecr-public/`。

上游儲存庫字首符合上游儲存庫名稱。根據預設，它會設定為 ROOT，允許與任何上游儲存庫相符。只有在 Amazon ECR 儲存庫字首具有非 值時，才能設定上游儲存庫字首。ROOT

下表顯示快取儲存庫名稱與上游儲存庫名稱之間的映射，根據其提取快取規則的字首組態。

| 快取命名空間     | 上游命名空間      | 映射關係 ( 快取儲存庫 → 上游儲存庫 )                                                                                   |
|------------|-------------|----------------------------------------------------------------------------------------------------------|
| ecr-public | ROOT ( 預設 ) | ecr-public/my-app/<br>image1 → my-app/im<br>age1<br><br>ecr-public/my-app/<br>image2 → my-app/im<br>age2 |
| ROOT       | ROOT        | my-app/image1 → my-<br>app/image1                                                                        |
| team-a     | team-a      | team-a/myapp/image1 →<br>team-a/myapp/image1                                                             |
| my-app     | 上游應用程式      | my-app/image1 →<br>upstream-app/image1                                                                   |

## 對 Amazon ECR 中的提取快取問題進行故障診斷

在使用提取快取規則提取上游映像時，以下是您可能會收到的最常見錯誤。

### 儲存庫不存在

指示儲存庫不存在的錯誤大多是因為儲存庫不存在於您的 Amazon ECR 私有登錄檔中，或是因為未授予 `ecr:CreateRepository` 許可給提取上游映像的 IAM 主體。若要解決此錯誤，您應該確認您提取命令中的儲存庫 URI 正確，已授予所需要的 IAM 許可給提取上游映像的 IAM 主體，或者在進行上游映像提取之前已在您的 Amazon ECR 私有登錄檔中建立了要將上游映像推送到的儲存庫。如需所需 IAM 許可的詳細資訊，請參閱 [將上游登錄檔與 Amazon ECR 私有登錄檔同步所需的 IAM 許可](#)。

以下為此錯誤的範例。

```
Error response from daemon: repository 111122223333.dkr.ecr.us-east-1.amazonaws.com/
ecr-public/amazonlinux/amazonlinux not found: name unknown: The repository with
name 'ecr-public/amazonlinux/amazonlinux' does not exist in the registry with id
'111122223333'
```

## 找不到請求的映像

指示找不到映像的錯誤大多是因為映像不存在於上游登錄檔中，或是因為未授予 `ecr:BatchImportUpstreamImage` 許可給提取上游映像的 IAM 主體，但已在您的 Amazon ECR 私有登錄檔中建立了儲存庫。若要解決此錯誤，您應該確認上游映像和映像標籤名稱正確，且其已存在且已將所需要的 IAM 許可授予提取上游映像的 IAM 主體。如需所需 IAM 許可的詳細資訊，請參閱 [將上游登錄檔與 Amazon ECR 私有登錄檔同步所需的 IAM 許可](#)。

以下為此錯誤的範例。

```
Error response from daemon: manifest for 111122223333.dkr.ecr.us-
east-1.amazonaws.com/ecr-public/amazonlinux/amazonlinux:latest not found: manifest
unknown: Requested image not found
```

## 403 從 Docker Hub 儲存庫提取時禁止

從標記為 Docker 官方映像的 Docker Hub 儲存庫中提取時，您必須在您使用的 URI 中包含 `/library/`。例如 `aws_account_id.dkr.ecr.region.amazonaws.com/docker-hub/library/image_name:tag`。如果您省略 `/library/` Docker Hub 官方映像的，當您嘗試使用提取快取規則提取映像時，將會傳回 403 Forbidden 錯誤。如需詳細資訊，請參閱 [在 Amazon ECR 中使用提取快取規則提取映像](#)。

以下為此錯誤的範例。

```
Error response from daemon: failed to resolve reference "111122223333.dkr.ecr.us-
west-2.amazonaws.com/docker-hub/amazonlinux:2023": pulling from host
111122223333.dkr.ecr.us-west-2.amazonaws.com failed with status code [manifests
2023]: 403 Forbidden
```

# Amazon ECR 中的私有映像複寫

您可以設定 Amazon ECR 私有登錄檔，以支援儲存庫的複寫。Amazon ECR 支援跨區域和跨帳戶複寫。若要進行跨帳戶複寫，目的地帳戶必須設定登錄檔許可政策，以允許從來源登錄檔進行複寫。如需詳細資訊，請參閱[Amazon ECR 中的私有登錄檔許可](#)。

## 主題

- [跨帳戶複寫政策需求](#)
- [私有映像複寫的考量](#)
- [Amazon ECR 的私有映像複寫範例](#)
- [在 Amazon ECR 中設定私有映像複寫](#)
- [在 Amazon ECR 中移除私有映像複寫設定](#)

## 跨帳戶複寫政策需求

若要讓跨帳戶 ECR 複寫正常運作，您必須了解哪個帳戶需要設定哪些政策。本節說明來源和目的地帳戶的政策需求。

### 政策組態概觀

跨帳戶 ECR 複寫只需要目的地帳戶的政策組態。來源帳戶不需要任何特殊儲存庫或登錄檔政策。

- 來源帳戶：在登錄檔設定中設定複寫規則。來源儲存庫不需要其他政策。
- 目的地帳戶：設定登錄檔許可政策，以允許來源帳戶複寫映像。

### 目的地登錄檔政策需求

目的地帳戶必須設定登錄許可政策，授予來源帳戶執行下列動作的許可：

- `ecr:ReplicateImage` - 允許來源帳戶將映像複寫到目的地登錄檔
- `ecr:CreateRepository` - 允許 ECR 在目的地登錄檔中自動建立尚未存在的儲存庫

**⚠ Important**

如果您未授予 `ecr:CreateRepository` 許可，您必須先在目的地帳戶中手動建立具有相同名稱的儲存庫，才能成功複製。

範例目的地登錄檔政策：

JSON

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Sid": "AllowCrossAccountReplication",
 "Effect": "Allow",
 "Principal": {
 "AWS": "arn:aws:iam::111122223333:root"
 },
 "Action": [
 "ecr:ReplicateImage",
 "ecr:CreateRepository"
],
 "Resource": "*"
 }
]
}
```

## 來源帳戶需求

來源帳戶只需要：

- 在登錄檔設定中設定複製規則，以指定目的地帳戶和區域
- 確保設定複製的 IAM 主體具有必要的 ECR 許可

來源儲存庫不需要其他政策。來源儲存庫不需要授予複製許可的儲存庫政策。

## 常見的誤解

以下是 ECR 跨帳戶複寫政策的常見誤解：

- 誤解：來源儲存庫需要允許目的地帳戶複寫映像的政策。

實際：來源儲存庫不需要任何特殊政策即可複寫。

- 誤解：來源和目的地帳戶都需要登錄政策。

實際：只有目的地帳戶需要登錄檔許可政策。

- 誤解：儲存庫政策和登錄檔政策是相同的。

事實：儲存庫政策控制對個別儲存庫的存取，而登錄政策則控制登錄檔層級的操作，例如複寫。

## 對複寫失敗進行故障診斷

如果跨帳戶複寫失敗，請檢查下列項目：

- 確認目的地帳戶已設定登錄檔許可政策
- 確保登錄政策同時包含 `ecr:ReplicateImage` 和 `ecr:CreateRepository` 動作
- 確認已在目的地登錄政策中正確指定來源帳戶 ID
- 檢查目的地儲存庫是否存在（如果 `ecr:CreateRepository` 未授予）
- 檢閱 CloudTrail 日誌是否有失敗 `CreateRepository` 或 `ReplicateImage` API 呼叫

## 私有映像複寫的考量

使用私有映像複寫時應考慮以下事項。

- 只有在設定複寫之後，才會將儲存庫內容推送或還原至儲存庫。不會複寫儲存庫中預先存在的任何內容。如果在複寫開啟後還原映像，則會進行複寫。如果在複寫開啟之前還原，則不會複寫。
- 發生複寫時，跨區域和帳戶的儲存庫名稱將保持不變。Amazon ECR 不支援在複寫期間變更儲存庫名稱。
- 第一次設定私有登錄檔進行複寫時，Amazon ECR 會代表您建立服務連結 IAM 角色。服務連結 IAM 角色授予 Amazon ECR 複寫服務在登錄檔中建立儲存庫和複寫映像所需的許可。如需詳細資訊，請參閱 [使用 Amazon ECR 的服務連結角色](#)。

- 若要進行跨帳戶複寫，私有登錄檔目的地必須授予許可，才能允許來源登錄複寫其映像。這是藉由設定私有登錄檔許可政策來完成。如需詳細資訊，請參閱[Amazon ECR 中的私有登錄檔許可](#)。
- 如果私有登錄檔的許可政策變更為移除許可，先前授予的任何進行中複寫都可能完成。
- 若要進行跨區域複寫，來源帳戶和目的地帳戶都必須先選擇加入區域，才能在該區域內或對該區域發生任何複寫動作。如需詳細資訊，請參閱《Amazon Web Services 一般參考》中的[管理 AWS 區域](#)。
- AWS 分割區之間不支援跨區域複寫。例如，無法將 us-west-2 中的儲存庫複寫到 cn-north-1。如需 AWS 分割區的詳細資訊，請參閱《AWS 一般參考》中的[ARN 格式](#)。
- 私有登錄檔的複寫設定可能包含最多 25 個唯一目的地，跨越所有規則，最多有 10 個規則總計。每個規則最多可包含 100 個篩選條件。例如，這允許為包含用於生產和測試之映像的儲存庫指定個別規則。
- 複寫組態支援篩選藉由指定儲存庫字首來複寫私有登錄檔中的儲存庫。如需範例，請參閱[範例：使用儲存庫篩選條件設定跨區域複寫](#)。
- 每個影像推送或影像還原只會執行複寫動作一次。例如，如果您將跨區域複寫從 us-west-2 至 us-east-1 和來自 us-east-1 至 us-east-2 進行設定，映像會推送至 us-west-2 僅複寫至 us-east-1，它不會再複寫至 us-east-2。這種行為適用於跨區域和跨帳戶複寫。
- 大多數映像可在不到 30 分鐘的時間內複製，但在極少數情況下，複製可能需要更長的時間。
- 登錄檔複寫不會執行任何刪除動作或封存動作。複寫的映像和儲存庫可以在不再使用時刪除或封存。
- 如果要複寫的影像已封存在目的地，則會在目的地還原。
- 在來源區域中封存映像時，不會將其封存在複寫組態指定的目的地區域中。
- 儲存庫政策 (包括 IAM 政策和生命週期政策) 不會進行複寫，除了它們定義的儲存庫外，也不會有任何影響。
- 預設不會複寫儲存庫設定，您可以使用儲存庫建立範本複寫儲存庫設定。這些設定包括標籤可變性、加密、儲存庫許可和生命週期政策。如需儲存庫建立範本的詳細資訊，請參閱[用於控制提取快取、推送時建立或複寫動作期間建立的儲存庫的範本](#)。
- 如果在儲存庫上啟用標籤不變性，而且複寫使用與現有映像相同標籤的映像，則會複寫映像，但不會包含重複的標籤。這可能導致映像未標籤。
- 複寫映像時，如果已設定 Blob 掛載，ECR 將檢查以確保來源儲存庫中的任何層已存在於目的地登錄檔中。如果目的地登錄檔中已存在任何圖層，ECR 會掛載這些圖層。

**Note**

如果來源登錄檔與其目的地登錄檔不同，則需要為 ECR 的兩個登錄檔啟用 Blob 掛載，以掛載複寫層。

## Amazon ECR 的私有映像複寫範例

以下範例顯示私有映像複寫作業的常見使用方式。如果您使用 設定複寫 AWS CLI，您可以在建立 JSON 檔案時使用 JSON 範例做為起點。如果您使用 設定複寫 AWS 管理主控台，當您在檢閱和提交頁面上檢閱複寫規則時，將會看到類似的 JSON。

### 範例：將跨區域複寫設定為單一目的地區域

下列顯示在單一登錄檔內設定跨區域複寫的範例。此範例假設您的帳戶 ID 為 111122223333 並且您正在 us-west-2 以外的區域中指定此複寫組態。

```
{
 "rules": [
 {
 "destinations": [
 {
 "region": "us-west-2",
 "registryId": "111122223333"
 }
]
 }
]
}
```

### 範例：使用儲存庫篩選條件設定跨區域複寫

下列範例說明為符合字首名稱值的儲存庫設定跨區域複寫。此範例假設您的帳戶 ID 為 111122223333 並且您正在 us-west-1 以外的區域中指定此複寫組態，並且儲存庫的字首為 prod。

```
{
 "rules": [{
 "destinations": [{
 "region": "us-west-1",
```

```
"registryId": "111122223333"
}],
"repositoryFilters": [{
 "filter": "prod",
 "filterType": "PREFIX_MATCH"
}]
}]
}
```

## 範例：設定跨區域複寫至多個目的地區域

下列顯示在單一登錄檔內設定跨區域複寫的範例。此範例假設您的帳戶 ID 為 `111122223333` 且您在 `us-west-1` 或 `us-west-2` 以外的區域中指定此複寫組態。

```
{
 "rules": [
 {
 "destinations": [
 {
 "region": "us-west-1",
 "registryId": "111122223333"
 },
 {
 "region": "us-west-2",
 "registryId": "111122223333"
 }
]
 }
]
}
```

## 範例：設定跨帳戶複寫

下列顯示為登錄檔設定跨帳戶複寫的範例。此範例會設定複寫到 `444455556666` 帳戶和 `us-west-2` 區域。

### Important

若要進行跨帳戶複寫，目的地帳戶必須設定登錄檔許可政策，以允許複寫發生。如需詳細資訊，請參閱 [Amazon ECR 中的私有登錄檔許可](#)。

```
{
 "rules": [
 {
 "destinations": [
 {
 "region": "us-west-2",
 "registryId": "444455556666"
 }
]
 }
]
}
```

## 範例：指定組態中的多個規則

以下顯示為登錄檔設定多個複寫規則的範例。此範例使用一個規則來設定 **111122223333** 帳戶的複寫，該規則會將字首為 **prod** 的儲存庫複寫至 **us-west-2** 區域，並將字首為 **test** 的儲存庫複寫至 **us-east-2** 區域。複寫組態最多可包含 10 個規則，每個規則最多可指定 25 個目的地。

```
{
 "rules": [{
 "destinations": [{
 "region": "us-west-2",
 "registryId": "111122223333"
 }],
 "repositoryFilters": [{
 "filter": "prod",
 "filterType": "PREFIX_MATCH"
 }]
 },
 {
 "destinations": [{
 "region": "us-east-2",
 "registryId": "111122223333"
 }],
 "repositoryFilters": [{
 "filter": "test",
 "filterType": "PREFIX_MATCH"
 }]
 }
]
}
```

## 範例：移除所有複寫設定

以下顯示從登錄檔中移除所有複寫設定的範例。若要移除複寫設定，您必須設定空規則陣列。

```
{
 "rules": []
}
```

### Important

移除複寫設定不會刪除任何先前複寫的儲存庫或映像。如果不再需要複寫的內容，您必須手動刪除。

## 在 Amazon ECR 中設定私有映像複寫

為您的私有登錄檔設定每個區域的複寫。您可以設定跨區域複寫或跨帳戶複寫。

如需複寫作業常見使用方式的範例，請參閱[Amazon ECR 的私有映像複寫範例](#)。

### 設定登錄檔複寫設定 (AWS 管理主控台)

1. 在 <https://console.aws.amazon.com/ecr/repositories> 開啟 Amazon ECR 主控台。
2. 從導覽列，選擇要為其設定登錄檔複寫設定的區域。
3. 在導覽窗格中，選擇 Private registry (私有登錄檔)。
4. 在私有登錄頁面上，選擇設定，然後選擇複寫組態下的編輯。
5. 在 Replication (複寫) 頁面上，選擇 Add replication rule (新增複寫規則)。
6. 在 Destination types (目的地類型) 頁面上，選擇要啟用跨區域複寫、跨帳戶複寫或兩者，然後選擇 Next (下一步)。
7. 如果啟用跨區域複寫，則對於 Configure destination regions (設定目的地區域)，選擇一或多個 Destination regions (目的地區域)，然後選擇 Next (下一步)。
8. 如果啟用跨帳戶複寫，則對於 Cross-account replication (跨帳戶複寫)，選擇登錄檔的跨帳戶複寫設定。對於 Destination account (目的地帳戶)，輸入目的地帳戶的帳戶 ID 和一或多個 Destination regions (目的地區域) 以進行複寫。選擇 Destination account + (目的地帳戶 +)，將其他帳戶設定為複寫目的地。

**⚠ Important**

若要進行跨帳戶複寫，目的地帳戶必須設定登錄檔許可政策，以允許複寫發生。如需詳細資訊，請參閱[Amazon ECR 中的私有登錄檔許可](#)。

9. (選用) 在 Add filters (新增篩選條件) 頁面上，指定複寫規則的一或多個篩選條件，然後選擇 Add (新增)。對於您要與複寫動作產生關聯的每個篩選條件重複此步驟。必須將篩選器指定為儲存庫名稱字首。如果未新增篩選器，則會複寫所有儲存庫的內容。一旦新增所有篩選條件，選擇 Next (下一步)。
10. 在 Review and submit (檢閱並提交) 頁面上，檢閱複寫規則組態，然後選擇 Submit rule (提交規則)。

## 設定登錄檔複寫設定 (AWS CLI)

1. 建立包含要為登錄檔定義的複寫規則的 JSON 檔案。複寫組態最多可包含 10 個規則，且所有規則最多 25 個唯一目的地，每個規則 100 個篩選條件。若要在您自己的帳戶內設定跨區域複寫，請指定您自己的帳戶 ID。如需更多範例，請參閱[Amazon ECR 的私有映像複寫範例](#)。

```
{
 "rules": [{
 "destinations": [{
 "region": "destination_region",
 "registryId": "destination_accountId"
 }],
 "repositoryFilters": [{
 "filter": "repository_prefix_name",
 "filterType": "PREFIX_MATCH"
 }]
 }]
}
```

2. 登錄的複寫組態。

```
aws ecr put-replication-configuration \
 --replication-configuration file://replication-settings.json \
 --region us-west-2
```

3. 確認您的登錄檔設定。

```
aws ecr describe-registry \
 --region us-west-2
```

## 在 Amazon ECR 中移除私有映像複寫設定

若要移除或停用私有登錄檔的複寫設定，您需要設定空的複寫組態。中沒有專用移除命令 AWS CLI。

### 移除登錄檔複寫設定 (AWS 管理主控台)

1. 在 <https://console.aws.amazon.com/ecr/repositories> 開啟 Amazon ECR 主控台。
2. 從導覽列中，選擇要從中移除登錄檔複寫設定的區域。
3. 在導覽窗格中，選擇 Private registry (私有登錄檔)。
4. 在私有登錄頁面上，選擇設定，然後選擇複寫組態下的編輯。
5. 選擇每個規則的刪除選項，移除所有現有的複寫規則。
6. 選擇儲存以套用空的複寫組態。

### 移除登錄檔複寫設定 (AWS CLI)

1. 使用空白規則陣列建立 JSON 檔案，以移除所有複寫設定。

```
{
 "rules": []
}
```

2. 將空的複寫組態套用至您的登錄檔。

```
aws ecr put-replication-configuration \
 --replication-configuration file://empty-replication-settings.json \
 --region us-west-2
```

3. 確認複寫設定已移除。

```
aws ecr describe-registry \
 --region us-west-2
```

輸出應會顯示空白 replicationConfiguration 且沒有規則。

**⚠ Important**

移除複寫設定不會刪除任何先前複寫的儲存庫或映像。如果不再需要複寫的內容，您必須手動刪除。

# 用於控制提取快取、推送時建立或複寫動作期間建立的儲存庫的範本

使用 Amazon ECR 儲存庫建立範本來定義 Amazon ECR 代表您建立的儲存庫設定。儲存庫建立範本中的設定只會在建立儲存庫期間套用，對使用任何其他方法建立的現有儲存庫或儲存庫沒有任何影響。目前，儲存庫建立範本可用於在建立儲存庫期間套用這些功能的設定：

- 提取快取
- 推送時建立
- 複寫

## 儲存庫建立範本的運作方式

有時 Amazon ECR 需要代表您建立新的私有儲存庫。例如：

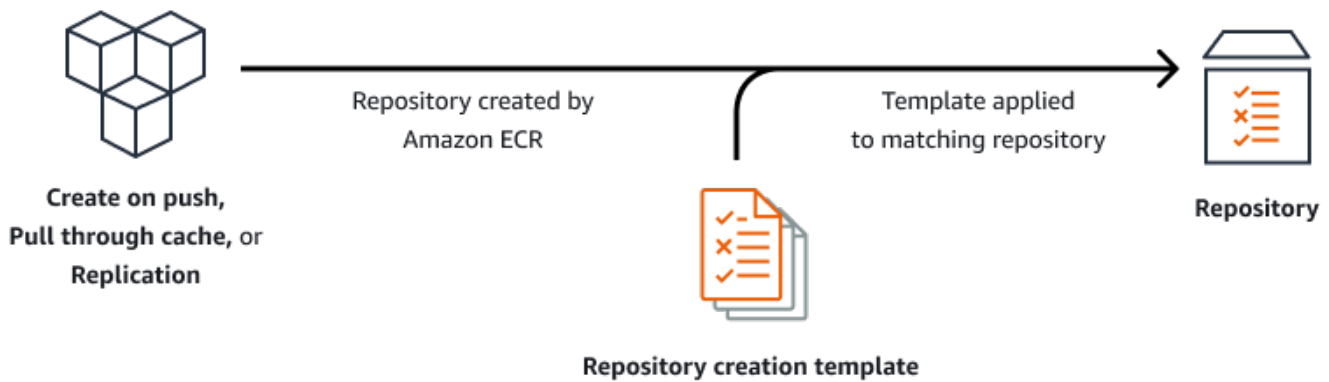
- 第一次使用提取快取規則擷取上游儲存庫的內容，並將其存放在 Amazon ECR 私有登錄檔中。
- 當您將映像推送至尚不存在的儲存庫時。
- 當您希望 Amazon ECR 將儲存庫複寫到另一個區域或帳戶時。

當沒有符合您提取快取規則或複寫儲存庫的儲存庫建立範本時，Amazon ECR 會使用新儲存庫的預設設定。這些預設設定包括關閉標籤不變性、使用 AES-256 加密，以及不套用任何儲存庫或生命週期政策。

如果沒有符合影像推送目標儲存庫的儲存庫建立範本，Amazon ECR 將不會建立具有預設設定的儲存庫。

使用儲存庫建立範本可讓您定義 Amazon ECR 套用至透過提取快取、推送時建立和複寫動作所建立的新儲存庫的設定。您可以為新儲存庫定義標籤不變性、加密組態、儲存庫許可、生命週期政策和資源標籤。

下圖顯示使用儲存庫建立範本時 Amazon ECR 所使用的工作流程。



以下詳細說明儲存庫建立範本中的每個參數。

## 字首

字首是與範本相關聯的儲存庫命名空間字首。使用此字首建立的所有儲存庫都會套用此範本中定義的設定。例如，`prod` 字首會套用至開頭為 `prod/` 的所有儲存庫。同樣地，`prod/team` 字首會套用至開頭為 `prod/team/` 的所有儲存庫。在包含兩個範本的登錄檔中，如果一個範本具有字首 "prod"，而另一個範本具有字首 "prod/team"，則具有字首 "prod/team" 的範本將套用至名稱開頭為 "prod/team/" 的所有儲存庫。

若要將範本套用至登錄檔中沒有關聯建立範本的所有儲存庫，您可以使用 `ROOT` 做為字首。

### ⚠ Important

總會有一個假設 / 套用至字首的結尾。如果您指定 `ecr-public` 為字首，Amazon ECR 會將其視為 `ecr-public/`。使用提取快取規則時，您在規則建立期間指定的儲存庫字首，也應該將其指定為儲存庫建立範本字首。

## Description

此範本描述是選用的，用於描述儲存庫建立範本的目的。

## 套用至

套用至設定的 `pullThroughCache` 會決定要使用此範本建立哪些 Amazon ECR 建立的儲存庫。有效值為 `PULL_THROUGH_CACHE`、`CREATE_ON_PUSH` 和 `REPLICATION`。例如，您第一次使用提取快取規則來擷取上游儲存庫的內容，並將其存放在 Amazon ECR 私有儲存庫時。如果沒有符合您提取快取規則的存放庫建立範本，Amazon ECR 會使用新儲存庫的預設設定。

## 儲存庫建立角色

儲存庫建立角色是 IAM 角色 ARN，Amazon ECR 會在透過儲存庫建立範本建立和設定儲存庫時擔任此角色。在範本中使用儲存庫標籤和/或 KMS 時，必須提供此角色，否則儲存庫建立會失敗。

## 影像標籤可變性

用於使用範本建立的儲存庫之標籤可變性設定。如果省略此參數，則會使用可變預設設定，以允許覆寫映像標籤。建議使用此設定，用於透過提取快取動作所建立的儲存庫的範本。這可確保標籤相同時，Amazon ECR 可以更新快取的映像。

如果已指定不可變，儲存庫中的所有映像標籤不可變，這會阻止它們遭覆寫。

## 加密組態

### Important

使用 AWS KMS (DSSE-KMS) 的雙層伺服器端加密僅適用於 AWS GovCloud (US) 區域。

用於使用範本建立之儲存庫的加密組態。

如果您使用 KMS 加密類型，儲存庫內容將搭配使用伺服器端加密與存放在 AWS KMS 中的 AWS Key Management Service 金鑰進行加密。當您使用 AWS KMS 加密資料時，您可以使用 Amazon ECR 的預設 AWS 受管 AWS KMS 金鑰，或指定您已建立的自有 AWS KMS 金鑰。您可以進一步選擇搭配使用單層或雙層加密 AWS KMS。如需詳細資訊，請參閱 [靜態加密](#)。如果您使用 KMS 加密類型並搭配跨區域複寫使用，您可能需要額外的許可。如需詳細資訊，請參閱 [建立用於複寫的 KMS 金鑰政策](#)。

如果您使用 AES256 加密類型，Amazon ECR 搭配使用伺服器端加密與 Amazon S3 受管加密金鑰，該方法使用 AES-256 加密演算法對儲存庫中的映像進行加密。如需詳細資訊，請參閱《Amazon Simple Storage Service 使用者指南》中的 [透過 Amazon S3 受管加密金鑰 \(SSE-S3\) 使用伺服器端加密來保護資料](#)。

## 儲存庫許可

套用至使用範本建立之儲存庫的儲存庫政策。儲存庫政策使用資源型許可來控制儲存庫的存取。資源型權限可讓您指定哪些 IAM 使用者或角色可以存取儲存庫，以及他們可以執行的動作。根據預設，只有建立儲存庫 AWS 的帳戶才能存取儲存庫。您可以套用政策文件來授予或拒絕儲存庫的其他許可。如需詳細資訊，請參閱 [Amazon ECR 中的私有儲存庫政策](#)。

## 儲存庫生命週期政策

用於使用範本建立的儲存庫之生命週期政策。生命週期政策提供對私有儲存庫中映像的生命週期管理的更多控制。生命週期政策包含一個或多項規則，其中的每一項規則都會定義 Amazon ECR 的動作。這提供藉由依據年齡或計數讓映像過期的方式，自動清理您的容器映像。如需詳細資訊，請參閱 [在 Amazon ECR 中使用生命週期政策來自動化映像的清除](#)。

## 資源標籤

資源標籤是要套用至儲存庫的中繼資料，可協助您分類和組織儲存庫。每個標籤皆包含由您定義的一個金鑰與一個選用值。如果您使用具有跨區域複寫的儲存庫建立範本，則需要將此許可套用至目的地登錄政策。

## 在 Amazon ECR 中建立儲存庫建立範本

您可以建立儲存庫建立範本，以定義在提取快取、在推送時建立或複寫動作期間，用於 Amazon ECR 代表您建立之儲存庫的設定。建立儲存庫建立範本後，所有建立的新儲存庫都會套用這些設定。這並不影響任何先前建立的儲存庫。

使用範本設定儲存庫時，您可以選擇指定 KMS 金鑰和資源標籤。如果您想要在一或多個範本中使用 KMS 金鑰、資源標籤或兩者的組合，您需要：

- [建立儲存庫建立範本的自訂政策](#)。
- [為儲存庫建立範本建立 IAM 角色](#)。

設定完成後，您可以將自訂角色連接至登錄檔中的特定範本。

## 建立儲存庫建立範本的 IAM 許可

IAM 主體需要下列許可才能管理儲存庫建立範本。必須使用身分型 IAM 政策授予此許可。

- `ecr:CreateRepositoryCreationTemplate` – 准許建立儲存庫建立範本。
- `ecr:UpdateRepositoryCreationTemplate` – 准許更新儲存庫建立範本。
- `ecr:DescribeRepositoryCreationTemplates` – 准許列出登錄檔中的儲存庫建立範本。
- `ecr>DeleteRepositoryCreationTemplate` – 准許刪除儲存庫建立範本。
- `ecr:CreateRepository` – 准許建立 Amazon ECR 儲存庫。

- `ecr:PutLifecyclePolicy` – 准許建立生命週期政策，並將其套用至儲存庫。僅當儲存庫建立範本包含生命週期政策時，才需要此許可。
- `ecr:SetRepositoryPolicy` – 准許為儲存庫建立許可政策。僅當儲存庫建立範本包含儲存庫政策時，才需要此許可。
- `iam:PassRole` – 准許允許實體將角色傳遞至服務或應用程式。需要擔任角色才能代表您執行動作的服務和應用程式需要此許可。

## 建立儲存庫建立範本的自訂政策

您可以使用 AWS 管理主控台 來定義隨後將與 IAM 角色相關聯的政策。然後，在設定儲存庫建立範本時，可以使用此 IAM 角色做為儲存庫建立角色。

### AWS 管理主控台

使用 JSON 政策編輯器為儲存庫建立範本建立自訂政策。

1. 登入 AWS 管理主控台，並在 <https://console.aws.amazon.com/iam/> 開啟 IAM 主控台。
2. 在左側的導覽窗格中，選擇 Policies (政策)。
3. 選擇建立政策。
4. 在政策編輯器中，選擇 JSON 選項。
5. 在 JSON 欄位中輸入下列政策。

### JSON

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Effect": "Allow",
 "Action": [
 "ecr:CreateRepository",
 "ecr:ReplicateImage",
 "ecr:TagResource"
],
 "Resource": "*"
 }
]
}
```

```
 "Effect": "Allow",
 "Action": [
 "kms:CreateGrant",
 "kms:RetireGrant",
 "kms:DescribeKey"
],
 "Resource": "*"
 }
]
```

6. 解決[政策驗證](#)期間產生的任何安全警告、錯誤或一般警告，然後選擇下一步。
7. 將許可新增至政策後，請選擇下一步。
8. 在檢視與建立頁面上，為您正在建立的政策輸入政策名稱與描述 (選用)。檢視此政策中定義的許可，來查看您的政策所授予的許可。
9. 選擇 Create policy (建立政策) 儲存您的新政策。
10. 建立角色以為建立範本指派此政策，請參閱 [為儲存庫建立範本建立 IAM 角色](#)。

## 為儲存庫建立範本建立 IAM 角色

您可以使用 AWS 管理主控台 來建立角色，當您在範本中使用儲存庫標籤或 KMS 的儲存庫建立範本中指定儲存庫建立角色時，Amazon ECR 可以使用該角色。

### AWS 管理主控台

建立角色。

1. 登入 AWS 管理主控台，並在 <https://console.aws.amazon.com/iam/> 開啟 IAM 主控台。
2. 在主控台的導覽窗格中，選擇 Roles (角色)，然後選擇 Create role (建立角色)。
3. 選擇自訂信任政策角色類型。
4. 在自訂信任政策區段中，貼上下列的自訂信任政策：

JSON

```
{
 "Version": "2012-10-17",
 "Statement": [
```

```
{
 "Effect": "Allow",
 "Principal": {
 "Service": "ecr.amazonaws.com"
 },
 "Action": "sts:AssumeRole"
}
]
```

5. 選擇下一步。
6. 從新增許可頁面，從許可政策清單中選取您先前建立的自訂政策旁的核取方塊，然後選擇下一步。
7. 針對 Role name (角色名稱)，輸入您的角色名稱。角色名稱在您的 中必須是唯一的 AWS 帳戶。角色名稱用在政策中或作為 ARN 的一部分時，角色名稱區分大小寫。當主控台客戶顯示角色名稱時 (例如在登入程序期間)，角色名稱不區分大小寫。因為有各種實體可能會參考此角色，所以建立角色之後，您就無法編輯其名稱。
8. (選用) 在 Description (說明) 中，輸入新角色的說明。
9. 檢閱角色，然後選擇建立角色。

## 建立儲存庫建立範本

完成範本的必要先決條件後，您就可以繼續建立儲存庫建立範本。

### AWS 管理主控台


若要建立儲存庫建立範本 (AWS 管理主控台)

1. 在 <https://console.aws.amazon.com/ecr/> 開啟 Amazon ECR 主控台。
2. 從導覽列選擇要在其中建立儲存庫建立範本的區域。
3. 在導覽窗格中，選擇私有登錄檔，儲存庫建立範本。
4. 在儲存庫建立範本頁面上，選擇建立範本。
5. 在步驟 1：定義範本頁面上，針對範本詳細資料，選擇特定字首以將範本套用至特定儲存庫命名空間字首，或選擇 ECR 登錄檔中的任何字首，將範本套用至與區域中任何其他範本不相符的所有儲存庫。

- a. 如果您選擇特定字首，請針對字首指定要套用範本的儲存庫命名空間字首。總會有一個假設 / 套用至字首的結尾。例如，prod 字首會套用至開頭為 prod/ 的所有儲存庫。同樣地，prod/team 字首會套用至開頭為 prod/team/ 的所有儲存庫。
  - b. 如果您選擇 ECR 登錄檔中的任何字首，字首將設定為 ROOT。
6. 對於適用於的，指定此範本將套用到哪些 Amazon ECR 工作流程。選項包括 PULL\_THROUGH\_CACHE、CREATE\_ON\_PUSH 和 REPLICATION。
  7. 針對範本描述，請指定範本的選擇性描述，然後選擇下一步。
  8. 在步驟 2：新增儲存庫建立組態頁面中，指定要套用至使用範本建立之儲存庫的儲存庫設定組態。
    - a. 針對 Image tag mutability (映像標籤可變性)，選擇要使用的標籤可變性設定。如需詳細資訊，請參閱[防止在 Amazon ECR 中覆寫映像標籤](#)。
      - 可互斥 – 如果您想要覆寫影像標籤，請選擇此選項。建議用於使用提取快取動作的儲存庫，以確保 Amazon ECR 可以更新快取映像。此外，若要停用幾個可變標籤的標籤更新，請輸入標籤名稱或使用萬用字元 (\*) 來比對 Mutable 標籤排除文字方塊中的多個類似標籤。
      - 不可變：如果您想要防止映像標籤遭到覆寫，而且在推送具有現有標籤的映像時，它適用於儲存庫中的所有標籤和排除項目，請選擇此選項。ImageTagAlreadyExistsException 如果您嘗試推送具有現有標籤的映像，Amazon ECR 會傳回。此外，若要啟用幾個不可變標籤的標籤更新，請輸入標籤名稱或使用萬用字元 (\*) 在不可分割標籤排除文字方塊中比對多個類似的標籤。
    - b. 針對加密組態，請選擇要使用的加密設定。如需詳細資訊，請參閱[靜態加密](#)。

選取 AES-256 時，Amazon ECR 會使用伺服器端加密與 Amazon Simple Storage Service 受管加密金鑰，該加密金鑰使用行業標準的 AES-256 加密演算法對靜態資料進行加密。此服務無須額外付費。

選取 AWS KMS 時，Amazon ECR 會使用儲存在 AWS Key Management Service (AWS KMS) 中的金鑰進行伺服器端加密。當您使用 AWS KMS 加密資料時，您可以使用由 Amazon ECR 管理的預設 AWS 受管金鑰，或指定自己的 AWS KMS 金鑰，稱為客戶受管金鑰。

 Note

一旦建立儲存庫，就無法變更儲存庫的加密設定。

- c. 針對儲存庫許可，請指定要套用至使用此範本建立之儲存庫的儲存庫許可政策。您可以選擇性地使用下拉式清單，針對最常見的使用案例選取其中一個 JSON 範例。如需詳細資訊，請參閱[Amazon ECR 中的私有儲存庫政策](#)。
  - d. 針對儲存庫生命週期政策，請指定要套用至使用此範本建立的儲存庫之儲存庫生命週期政策。您可以選擇性地使用下拉式清單，針對最常見的使用案例選取其中一個 JSON 範例。如需詳細資訊，請參閱[在 Amazon ECR 中使用生命週期政策來自動化映像的清除](#)。
  - e. 對於儲存庫 AWS 標籤，請以鍵值對的形式指定中繼資料，以與使用此範本建立的儲存庫建立關聯，然後選擇下一步。如需詳細資訊，請參閱[在 Amazon ECR 中標記私有儲存庫](#)。
  - f. 對於儲存庫建立角色，從下拉式功能表中選取自訂 IAM 角色，以便在範本中使用儲存庫標籤或 KMS 時用於儲存庫建立範本（如需詳細資訊為[儲存庫建立範本建立 IAM 角色](#)，請參閱）。然後選擇下一步。
9. 在步驟 3：檢閱和建立頁面上，檢閱您為儲存庫建立範本指定的設定。選擇編輯選項來進行變更。一旦完成，請選擇建立。

## AWS CLI

[create-repository-creation-template](#) AWS CLI 命令用於為您的私有登錄檔建立儲存庫建立範本。

若要建立儲存庫建立範本 (AWS CLI)

1. 使用 AWS CLI 為 [create-repository-creation-template](#) 命令產生骨架。

```
aws ecr create-repository-creation-template \
--generate-cli-skeleton
```

命令的輸出會顯示儲存庫建立範本的完整語法。

```
{
 "appliedFor":[""], // string array, but valid are PULL_THROUGH_CACHE,
 CREATE_ON_PUSH, and REPLICATION
 "prefix": "string",
 "description": "string",
 "imageTagMutability":
 "MUTABLE"|"IMMUTABLE"|"IMMUTABLE_WITH_EXCLUSION"|"MUTABLE_WITH_EXCLUSION",
 "imageTagMutabilityExclusionFilters": [
 "filterType": "WILDCARD",
 "filter": "string"
```

```

],
 "repositoryPolicy": "string",
 "lifecyclePolicy": "string"
 "encryptionConfiguration": {
 "encryptionType": "AES256"|"KMS",
 "kmsKey": "string"
 },
 "resourceTags": [
 {
 "Key": "string",
 "Value": "string"
 }
],
 "customRoleArn": "string", // must be a valid IAM Role ARN
}

```

2. 使用上一個步驟的 `repository-creation-template.json` 輸出建立名為 `prod/*` 的檔案。此範本會為在下建立的任何儲存庫設定 KMS 加密金鑰，`prod/*` 該儲存庫政策會啟用將映像推送和提取到未來的儲存庫、設定生命週期政策，使映像過期超過兩週，並設定自訂角色，讓 ECR 存取 KMS 金鑰，並將資源標籤指派給 `examplekey` 未來的儲存庫。

```

{
 "prefix": "prod",
 "description": "For repositories cached from my PTC rule and in my replication configuration that start with 'prod/'",
 "appliedFor": ["PULL_THROUGH_CACHE", "CREATE_ON_PUSH", "REPLICATION"],
 "encryptionConfiguration": {
 "encryptionType": "KMS",
 "kmsKey": "arn:aws:kms:us-west-2:111122223333:key/a1b2c3d4-5678-90ab-cdef-example11111"
 },
 "resourceTags": [
 {
 "Key": "examplekey",
 "Value": "examplevalue"
 }
],
 "imageTagMutability": "IMMUTABLE_WITH_EXCLUSION",
 "imageTagMutabilityExclusionFilters": [
 {
 "filterType": "WILDCARD",
 "filter": "latest"
 }
],
}

```

```

 {
 "filterType": "WILDCARD",
 "filter": "beta*"
 }
]
 "repositoryPolicy": "{ \"Version\": \"2012-10-17\", \"Statement\":
 [{ \"Sid\": \"AllowPushPullIAMRole\", \"Effect\": \"Allow\", \"Principal\": { \"AWS\":
 \"arn:aws:iam::111122223333:user/IAMUsername\" }, \"Action\": [\"ecr:BatchGetImage
 \", \"ecr:BatchCheckLayerAvailability\", \"ecr:CompleteLayerUpload\",
 \"ecr:GetDownloadUrlForLayer\", \"ecr:InitiateLayerUpload\", \"ecr:PutImage\",
 \"ecr:UploadLayerPart\"]] }] }",
 "lifecyclePolicy": "{ \"rules\": [{ \"rulePriority\": 1, \"description\": \"Expire
 images older than 14 days\", \"selection\": { \"tagStatus\": \"any\", \"countType
 \": \"sinceImagePushed\", \"countUnit\": \"days\", \"countNumber\": 14 }, \"action\":
 { \"type\": \"expire\" } }] }",
 "customRoleArn": "arn:aws:iam::111122223333:role/myRole"
}

```

3. 使用下列命令來建立儲存庫建立範本。請確定您在 `repository-creation-template.json` 下列範例中指定在上一個步驟中建立的組態檔案名稱，以取代。

```

aws ecr create-repository-creation-template \
 --cli-input-json file://repository-creation-template.json

```

## 更新儲存庫建立範本

如果您需要變更儲存庫的組態，您可以編輯儲存庫建立範本。編輯儲存庫建立範本後，新組態將套用至現有範本。

### Important

這並不影響任何先前建立的儲存庫。

## AWS 管理主控台

### 編輯儲存庫建立範本 (AWS 管理主控台)

1. 在 <https://console.aws.amazon.com/ecr/> 開啟 Amazon ECR 主控台。
2. 從導覽列中，選擇要編輯的儲存庫建立範本所在的區域。

3. 在導覽窗格中，選擇私有登錄檔，然後選擇設定。
4. 從導覽列中，選擇儲存庫建立範本。
5. 在儲存庫建立範本頁面上，選取要編輯的儲存庫建立範本。
6. 從動作下拉式功能表中，選擇編輯。
7. 檢閱並更新組態設定。
8. 選擇更新以套用新的建立範本組態。

## AWS CLI

### 編輯儲存庫建立範本 (AWS CLI)

- 使用 [update-repository-creation-template](#) 命令來更新現有的儲存庫建立範本。您必須指定範本 `prefix` 的值。下列範例會使用 `prod` 字首更新儲存庫建立範本。

```
aws ecr update-repository-creation-template \
 --prefix prod \
 --image-tag-mutability="IMMUTABLE_WITH_EXCLUSION" \
 --image-tag-mutability-exclusion-filters filterType=WILDCARD, filter=latest
```

命令的輸出會顯示更新儲存庫建立範本的詳細資訊。

## 在 Amazon ECR 中刪除儲存庫建立範本

如果您使用完儲存庫建立範本，您可以將其刪除。刪除儲存庫建立範本後，任何在提取快取或複寫動作期間關聯字首下新建立的儲存庫都會繼承預設設定，除非找到另一個相符的範本，請參閱 [儲存庫建立範本的運作方式](#)。

### Important

這並不影響任何先前建立的儲存庫。

## AWS 管理主控台

### 若要刪除儲存庫建立範本 (AWS 管理主控台)

1. 在 <https://console.aws.amazon.com/ecr/> 開啟 Amazon ECR 主控台。

2. 從導覽列上，選擇要刪除的儲存庫建立範本所在的區域。
3. 在導覽窗格中，選擇私有登錄檔，儲存庫建立範本。
4. 在儲存庫建立範本頁面上，選取要刪除的儲存庫建立範本。
5. 從動作下拉式選單中，選擇刪除。

## AWS CLI

### 若要刪除儲存庫建立範本 (AWS CLI)

- 使用 [delete-repository-creation-template.html](#) 命令來刪除現有的儲存庫建立範本。您必須指定範本prefix的值。下列範例會刪除具有 prod字首的儲存庫建立範本。

```
aws ecr delete-repository-creation-template \
 --prefix prod
```

命令的輸出會顯示已刪除儲存庫建立範本的詳細資訊。

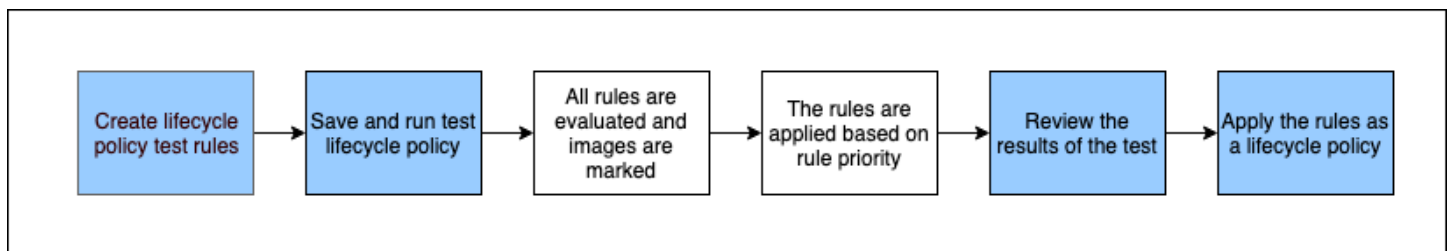
# 在 Amazon ECR 中使用生命週期政策來自動化映像的清除

Amazon ECR 生命週期政策提供對私有儲存庫中映像的生命週期管理的更多控制。生命週期政策包含一或多個規則，每個規則都會定義 Amazon ECR 的動作。根據生命週期政策中的過期條件，可以根據生命週期政策中指定的條件在 24 小時內封存或過期映像。當 Amazon ECR 根據生命週期政策執行動作時，此動作會擷取為中的事件 AWS CloudTrail。如需詳細資訊，請參閱[使用 記錄 Amazon ECR 動作 AWS CloudTrail](#)。

## 生命週期政策如何運作

生命週期政策由一或多個規則組成，用以決定儲存庫中的映像過期與否。在考慮使用生命週期政策時，請務必使用生命週期政策預覽來確認生命週期政策到期的映像，然後再將其套用至儲存庫。將生命週期政策套用至儲存庫後，您可預期映像將在符合到期條件後的 24 小時內過期。當 Amazon ECR 依據生命週期政策執行動作時，在 AWS CloudTrail 中會將此動作視為事件。如需詳細資訊，請參閱[使用 記錄 Amazon ECR 動作 AWS CloudTrail](#)。

以下圖表顯示生命週期政策工作流程。



1. 建立一或多個測試規則。
2. 儲存測試規則並執行預覽。
3. 生命週期政策評估工具會檢視所有規則，並標記每個規則會影響的映像。
4. 然後，生命週期政策評估器會根據規則優先順序套用規則，並顯示儲存庫中哪些映像設定為過期或封存。規則優先順序數字越低表示優先順序越高。例如，優先順序為 1 的規則優先於優先順序為 2 的規則。
5. 檢閱測試結果，確保標記為過期或封存的映像符合您預期。
6. 套用測試規則作為儲存庫的生命週期政策。
7. 建立生命週期政策後，您應該預期映像會在符合過期條件後的 24 小時內過期或封存。

## 生命週期政策評估規則

生命週期政策評估工具負責剖析生命週期政策的純文字 JSON、評估所有規則，然後根據規則優先順序套用這些規則至儲存庫中的映像。以下詳細說明生命週期政策評估工具的邏輯。如需範例，請參閱 [Amazon ECR 中的生命週期政策範例](#)。

- 當儲存庫中有參考成品時，Amazon ECR 生命週期政策會在刪除或封存主體映像的 24 小時內自動過期或封存這些成品。
- 不論規則優先順序，都會同時評估所有規則。評估所有規則之後，就會根據規則優先順序進行套用。
- 映像已過期或僅由一或零個規則封存。
- 符合規則標記需求的映像，無法由優先順序較低的規則過期或封存。
- 規則永遠無法標記由較高優先順序規則標記的影像，但仍可以將其識別為尚未過期或封存。
- 選取特定儲存類別的所有規則集必須包含一組唯一的字首。
- 只允許選取一個特定儲存類別的規則來選取未標記的影像。
- 如果資訊清單清單參考影像，則必須先刪除或封存資訊清單清單，才能過期或封存該影像。
- 過期一律由 `pushed_at_time` 或 `排序`，`transitioned_at_time` 並一律在較新的映像之前過期。如果影像已封存，然後在過去的任何時間點還原，`last_activated_at` 則會使用影像的而非 `pushed_at_time`。
- 生命週期政策規則可指定 `tagPatternList` 或 `tagPrefixList`，但不能同時指定兩者。不過，生命週期策略可包含多項規則，不同規則可同時使用模式和字首清單。如果 `tagPatternList` 或 `tagPrefixList` 值中的所有標籤都與任何影像的標籤相符，則會成功比對影像。
- 唯有當 `tagStatus` 是 `tagged` 時，才能使用 `tagPatternList` 或 `tagPrefixList` 參數。
- 使用 `tagPatternList` 時，如果映像與萬用字元篩選條件相符，便會成功配對映像。例如，如果 `prod*` 套用篩選條件，它會比對名稱開頭為 `prod` 的影像標籤 `prod1`，`prod` 例如 `prod`、`production-team1`。同樣地，如果 `*prod*` 套用篩選條件，則會比對名稱包含 `repo-production` 或 `prod` 的影像標籤 `prod-team`。

### Important

每個字串最多可有 4 個萬用字元 (`*`)，例如 `["*test*1*2*3", "test*1*2*3*"]` 是有效字串，但 `["test*1*2*3*4*5*6"]` 則為無效。

- 使用 `tagPrefixList` 時，如果 `tagPrefixList` 值中的所有萬用字元篩選條件都符合任何影像的標籤，則映像會成功比對。

- `countUnit` 參數僅在 `countType` 為 `sinceImagePushed`、`sinceImagePulled` 或 時使用 `sinceImageTransitioned`。
- 使用 時 `countType = imageCountMoreThan`，影像會根據 來從最小到最舊排序，`pushed_at_time` 然後所有大於指定計數的影像都會過期或封存。
- 使用 `countType = sinceImagePushed` 時，其 `pushed_at_time` 早於根據 指定天數的所有映像 `countNumber` 都會過期或封存。
- 使用 `countType = sinceImagePulled` 時，`countNumber` 會封存其 `last_recorded_pulltime` 早於根據 指定天數的所有映像。如果從未提取影像，`pushed_at_time` 則會使用影像的 而非 `last_recorded_pulltime`。如果影像已封存，然後在過去的任何時間點還原，但自影像還原以來從未提取，`last_activated_at` 則會使用影像的 ，而非 `last_recorded_pulltime`。
- 使用 `countType = sinceImageTransitioned` 時，所有 `last_archived_at` 早於 所指定天數的封存映像 `countNumber` 都會過期。
- 過期一律由 排序，`pushed_at_time` 並在較新的映像之前一律過期。

## 在 Amazon ECR 中建立生命週期政策預覽

您可以使用生命週期政策預覽來查看生命週期政策對映像儲存庫的影響，然後再套用它。在將生命週期政策套用至儲存庫之前，將預覽視為最佳實務。

### Note

如果您使用 Amazon ECR 複寫跨不同區域或帳戶複製儲存庫，請注意，生命週期政策只能對建立儲存庫所在區域中的儲存庫採取動作。因此，如果您已開啟複寫功能，您可能需要考慮在複寫儲存庫的每個區域和帳戶中建立生命週期政策。

### 建立生命週期政策預覽 (AWS 管理主控台)

1. 在 <https://console.aws.amazon.com/ecr/repositories> 開啟 Amazon ECR 主控台。
2. 從導覽列上，選擇其中包含要執行生命週期政策預覽的儲存庫之區域。
3. 在導覽窗格中，依序選擇私有登錄檔和儲存庫。
4. 在私有儲存庫頁面中選取儲存庫，然後使用動作下拉式清單選擇生命週期政策。
5. 在儲存庫的生命週期政策規則頁面中，依序選擇編輯測試規則和建立規則。

6. 為每個生命週期政策測試規則輸入下列詳細資訊。
  - a. 在 Rule priority (規則優先順序) 中，輸入規則優先順序的編號。規則優先順序會決定生命週期政策規則的套用順序。數字越小表示優先順序越高。例如，優先順序為 1 的規則優先於優先順序為 2 的規則。
  - b. 在 Rule description (規則描述) 中，輸入生命週期政策規則的描述。
  - c. 針對映像狀態，請選擇已標記 (萬用字元比對)、已標記 (前綴比對)、未標記或任何。

**⚠ Important**

若您指定多個標籤，只會選擇含有所有指定標籤的映像。

- d. 如果您在映像狀態選擇已標記 (萬用字元比對)，可針對指定萬用字元比對標籤指定含萬用字元 (\*) 的映像標籤清單，以透過生命週期政策對其執行動作。例如，若您的映像已標記為 prod、prod1、prod2 等等，您可能需要指定 prod\* 對所有映像執行動作。若您指定多個標籤，只會選擇含有所有指定標籤的映像。

**⚠ Important**

每個字串最多可有 4 個萬用字元 (\*)，例如 ["\*test\*1\*2\*3", "test\*1\*2\*3\*"] 是有效字串，但 ["test\*1\*2\*3\*4\*5\*6"] 則為無效。

- e. 如果您在映像狀態選擇已標記 (前綴比對)，則可針對指定前綴比對標籤指定映像標籤清單，以透過生命週期政策對其執行動作。
  - f. 針對相符條件，選擇自影像建立以來的天數、自上次記錄提取時間以來的天數、自影像封存以來的天數或影像計數，然後指定值。
  - g. 針對規則動作，選擇過期或封存。
  - h. 選擇儲存。
7. 重複操作步驟 5-7 來建立其他測試生命週期政策規則。
  8. 若要執行生命週期政策預覽，請選擇 Save and run test (儲存並執行測試)。
  9. 在 Image matches for test lifecycle rules (測試生命週期規則的映像符合數) 下，檢視您的生命週期政策預覽影響。
  10. 若您對預覽結果感到滿意，請選擇 Apply as lifecycle policy (套用為生命週期政策) 來使用指定規則建立生命週期政策。您應該預期套用生命週期政策後，受影響的映像會在 24 小時內過期或封存。
  11. 如果對預覽結果不滿意，您可以刪除一或多個測試生命週期規則，再建立一或多個規則加以取代，然後重複測試。

# 在 Amazon ECR 中為儲存庫建立生命週期政策

使用生命週期政策來建立一組過期或封存未使用儲存庫映像的規則。建立生命週期政策後，受影響的映像會在 24 小時內過期或封存。

## Note

如果您使用 Amazon ECR 複寫跨不同區域或帳戶複製儲存庫，請注意，生命週期政策只能對建立儲存庫所在區域中的儲存庫採取動作。因此，如果您已開啟複寫功能，您可能需要考慮在複寫儲存庫的每個區域和帳戶中建立生命週期政策。

## 先決條件

最佳實務：建立生命週期政策預覽，以確認生命週期政策規則已過期或封存的映像是您想要的。如需說明，請參閱在 [Amazon ECR 中建立生命週期政策預覽](#)。

## 建立生命週期政策 (AWS 管理主控台)

1. 在 <https://console.aws.amazon.com/ecr/repositories> 開啟 Amazon ECR 主控台。
2. 從導覽列上，選擇其中包含要建立生命週期政策的儲存庫之區域。
3. 在導覽窗格中，依序選擇私有登錄檔和儲存庫。
4. 在私有儲存庫頁面中選取儲存庫，然後使用動作下拉式清單選擇生命週期政策。
5. 在儲存庫生命週期政策頁面中，選擇建立規則。
6. 為您的生命週期政策規則輸入下列詳細資訊。
  - a. 在 Rule priority (規則優先順序) 中，輸入規則優先順序的編號。規則優先順序會決定生命週期政策規則的套用順序。規則優先順序數字越低表示優先順序越高。例如，優先順序為 1 的規則優先於優先順序為 2 的規則。
  - b. 在 Rule description (規則描述) 中，輸入生命週期政策規則的描述。
  - c. 針對映像狀態，請選擇已標記 (萬用字元比對)、已標記 (前綴比對)、未標記或任何。

## Important

若您指定多個標籤，只會選擇含有所有指定標籤的映像。

- d. 如果您在映像狀態選擇已標記 (萬用字元比對)，可針對指定萬用字元比對標籤指定含萬用字元 (\*) 的映像標籤清單，以透過生命週期政策對其執行動作。例如，若您的映像已標記為 prod、prod1、prod2 等等，您可能需要指定 prod\* 對所有映像執行動作。若您指定多個標籤，只會選擇含有所有指定標籤的映像。

**⚠ Important**

每個字串最多可有 4 個萬用字元 (\*)，例如 ["\*test\*1\*2\*3", "test\*1\*2\*3\*"] 是有效字串，但 ["test\*1\*2\*3\*4\*5\*6"] 則為無效。

- e. 如果您在映像狀態選擇已標記 (前綴比對)，則可針對指定前綴比對標籤指定映像標籤清單，以透過生命週期政策對其執行動作。
  - f. 針對相符條件，選擇影像建立後天數、上次記錄提取時間後天數、影像封存後天數或影像計數，然後指定值。
  - g. 針對規則動作，選擇過期或封存。
  - h. 選擇儲存。
7. 重複操作步驟 5-7 來建立其他生命週期政策規則。

## 建立生命週期政策 (AWS CLI)

1. 取得要建立生命週期政策的儲存庫名稱。

```
aws ecr describe-repositories
```

2. 建立名為 policy.json 的本機檔案使用生命週期政策的內容。如需生命週期政策範例，請參閱「[Amazon ECR 中的生命週期政策範例](#)」。
3. 透過指定儲存庫名稱並參考您建立的生命週期政策 JSON 檔案來建立生命週期政策。

```
aws ecr put-lifecycle-policy \
 --repository-name repository-name \
 --lifecycle-policy-text file://policy.json
```

## Amazon ECR 中的生命週期政策範例

以下是顯示語法的生命週期政策範例。

若要查看政策屬性的詳細資訊，請參閱 [Amazon ECR 中的生命週期政策屬性](#)。如需使用 建立生命週期政策的說明 AWS CLI，請參閱 [建立生命週期政策 \(AWS CLI\)](#)。

## 生命週期政策範本

在與儲存庫建立關聯之前，會評估生命週期政策的內容。以下是生命週期政策的 JSON 語法範本。

```
{
 "rules": [
 {
 "rulePriority": integer,
 "description": "string",
 "selection": {
 "tagStatus": "tagged"|"untagged"|"any",
 "tagPatternList": list<string>,
 "tagPrefixList": list<string>,
 "storageClass": "standard"|"archive",
 "countType":
"imageCountMoreThan"|"sinceImagePushed"|"sinceImagePulled"|"sinceImageTransitioned",
 "countUnit": "string",
 "countNumber": integer
 },
 "action": {
 "type": "expire"|"transition",
 "targetStorageClass": "archive"
 }
 }
]
}
```

## 篩選映像存在時間

以下範例顯示政策的生命週期政策語法，該政策能尋找以 prod 標籤開頭的映像，使用 prod\* 的 tagPatternList，將存在時間同樣超過 14 天的映像設為過期。

```
{
 "rules": [
 {
 "rulePriority": 1,
 "description": "Expire images older than 14 days",
 "selection": {
 "tagStatus": "tagged",
```

```
 "tagPatternList": ["prod*"],
 "countType": "sinceImagePushed",
 "countUnit": "days",
 "countNumber": 14
 },
 "action": {
 "type": "expire"
 }
}
]
```

## 篩選上次提取的時間

下列範例顯示政策的生命週期政策語法，該政策會將映像轉換為未在90幾天內提取的封存儲存。

```
{
 "rules": [
 {
 "rulePriority": 1,
 "description": "Archive images not pulled in 90 days",
 "selection": {
 "tagStatus": "any",
 "countType": "sinceImagePulled",
 "countUnit": "days",
 "countNumber": 90
 },
 "action": {
 "type": "transition",
 "targetStorageClass": "archive"
 }
 }
]
}
```

### Important

`sinceImagePulled` 計數類型必須與 `transition` 動作搭配使用。它無法與 `expire` 動作搭配使用。若要根據提取活動刪除映像，請先將其轉換為使用 封存儲存體 `sinceImagePulled`，然後使用 `sinceImageTransitioned` 搭配 `expire` 動作來刪除它們。在刪除之前，映像必須位於封存儲存中至少 90 天。

## 篩選封存轉換時間

下列範例顯示政策的生命週期政策語法，該政策會將封存儲存超過 365 天的封存映像過期。

```
{
 "rules": [
 {
 "rulePriority": 1,
 "description": "Expire images archived for more than 365 days",
 "selection": {
 "tagStatus": "any",
 "storageClass": "archive",
 "countType": "sinceImageTransitioned",
 "countUnit": "days",
 "countNumber": 365
 },
 "action": {
 "type": "expire"
 }
 }
]
}
```

### Important

`sinceImageTransitioned` 計數類型必須與 `expire` 動作和 `archive` 儲存體方案搭配使用。在刪除之前，映像必須位於封存儲存中至少 90 天。

## 篩選映像計數

以下範例顯示政策的生命週期政策語法，該政策能只保留一個未標記的映像，並將其他所有映像設為過期。

```
{
 "rules": [
 {
 "rulePriority": 1,
 "description": "Keep only one untagged image, expire all others",
 "selection": {
 "tagStatus": "untagged",

```

```
 "countType": "imageCountMoreThan",
 "countNumber": 1
 },
 "action": {
 "type": "expire"
 }
}
]
```

## 篩選多個規則

以下是在生命週期政策中使用多個規則的範例。範例儲存庫與指定的生命週期政策與結果說明同時提供。

### 範例 A

儲存庫內容：

- Image A, Taglist: ["beta-1", "prod-1"], Pushed: 10 days ago
- Image B, Taglist: ["beta-2", "prod-2"], Pushed: 9 days ago
- Image C, Taglist: ["beta-3"], Pushed: 8 days ago

生命週期政策文字：

```
{
 "rules": [
 {
 "rulePriority": 1,
 "description": "Rule 1",
 "selection": {
 "tagStatus": "tagged",
 "tagPatternList": ["prod*"],
 "countType": "imageCountMoreThan",
 "countNumber": 1
 },
 "action": {
 "type": "expire"
 }
 },
 {
```

```

 "rulePriority": 2,
 "description": "Rule 2",
 "selection": {
 "tagStatus": "tagged",
 "tagPatternList": ["beta*"],
 "countType": "imageCountMoreThan",
 "countNumber": 1
 },
 "action": {
 "type": "expire"
 }
 }
]
}

```

生命週期政策的邏輯會是：

- 規則 1 找出含有前綴 prod 標記的映像。將會從最舊的映像開始標記，直到沒有或剩餘很少符合的映像。標記映像 A 為過期。
- 規則 2 找出含有前綴 beta 標記的映像。將會從最舊的映像開始標記，直到沒有或剩餘很少符合的映像。標記映像 A 與映像 B 為過期。但是，映像 A 已被規則 1 看到，而若映像 B 已過期，則會因違反規則 1 而被略過。
- 結果：映像 A 已過期。

## 範例：B

這是與前一個範例相同的儲存庫，但是規則優先順序已變更以說明結果。

儲存庫內容：

- Image A, Taglist: ["beta-1", "prod-1"], Pushed: 10 days ago
- Image B, Taglist: ["beta-2", "prod-2"], Pushed: 9 days ago
- Image C, Taglist: ["beta-3"], Pushed: 8 days ago

生命週期政策文字：

```

{
 "rules": [
 {

```

```
 "rulePriority": 1,
 "description": "Rule 1",
 "selection": {
 "tagStatus": "tagged",
 "tagPatternList": ["beta*"],
 "countType": "imageCountMoreThan",
 "countNumber": 1
 },
 "action": {
 "type": "expire"
 }
 },
 {
 "rulePriority": 2,
 "description": "Rule 2",
 "selection": {
 "tagStatus": "tagged",
 "tagPatternList": ["prod*"],
 "countType": "imageCountMoreThan",
 "countNumber": 1
 },
 "action": {
 "type": "expire"
 }
 }
]
```

生命週期政策的邏輯會是：

- 規則 1 找出含有前綴 beta 標記的映像。將會從最舊的映像開始標記，直到沒有或剩餘很少符合的映像。將看到所有三個映像並標記映像 A 與映像 B 為過期。
- 規則 2 找出含有前綴 prod 標記的映像。將會從最舊的映像開始標記，直到沒有或剩餘很少符合的映像。將不會看到映像，因所有可用映像已被規則 1 看到，因此將不會標記其他映像。
- 結果：映像 A 與 B 皆已過期。

## 篩選單一規則中的多個標籤

以下範例說明在單一規則中採用多標籤模式的生命週期政策語法。範例儲存庫與指定的生命週期政策與結果說明同時提供。

## 範例 A

當單一規則指定多個標籤模式，映像必須符合所有列出的標籤模式。

儲存庫內容：

- Image A, Taglist: ["alpha-1"], Pushed: 12 days ago
- Image B, Taglist: ["beta-1"], Pushed: 11 days ago
- Image C, Taglist: ["alpha-2", "beta-2"], Pushed: 10 days ago
- Image D, Taglist: ["alpha-3"], Pushed: 4 days ago
- Image E, Taglist: ["beta-3"], Pushed: 3 days ago
- Image F, Taglist: ["alpha-4", "beta-4"], Pushed: 2 days ago

```
{
 "rules": [
 {
 "rulePriority": 1,
 "description": "Rule 1",
 "selection": {
 "tagStatus": "tagged",
 "tagPatternList": ["alpha*", "beta*"],
 "countType": "sinceImagePushed",
 "countNumber": 5,
 "countUnit": "days"
 },
 "action": {
 "type": "expire"
 }
 }
]
}
```

生命週期政策的邏輯會是：

- 規則 1 找出含有前綴 alpha 和 beta 標籤的映像。看到映像 C 與 F。應標記存在時間大於五天的映像，應是映像 C。
- 結果：映像 C 已過期。

## 範例：B

下列範例說明非專屬的標籤。

儲存庫內容：

- Image A, Taglist: ["alpha-1", "beta-1", "gamma-1"], Pushed: 10 days ago
- Image B, Taglist: ["alpha-2", "beta-2"], Pushed: 9 days ago
- Image C, Taglist: ["alpha-3", "beta-3", "gamma-2"], Pushed: 8 days ago

```
{
 "rules": [
 {
 "rulePriority": 1,
 "description": "Rule 1",
 "selection": {
 "tagStatus": "tagged",
 "tagPatternList": ["alpha*", "beta*"],
 "countType": "imageCountMoreThan",
 "countNumber": 1
 },
 "action": {
 "type": "expire"
 }
 }
]
}
```

生命週期政策的邏輯會是：

- 規則 1 找出含有前綴 alpha 和 beta 標籤的映像。看到所有映像。將會從最舊的映像開始標記，直到沒有或剩餘很少符合的映像。標記映像 A 與 B 為過期。
- 結果：映像 A 與 B 皆已過期。

## 篩選所有映像

以下生命週期政策範例說明含有不同篩選條件的映像。範例儲存庫與指定的生命週期政策與結果說明同時提供。

## 範例 A

下列顯示套用到所有規則的政策之生命週期政策語法，但是只保留一個映像並將其他所有映像設為過期。

儲存庫內容：

- Image A, Taglist: ["alpha-1"], Pushed: 4 days ago
- Image B, Taglist: ["beta-1"], Pushed: 3 days ago
- Image C, Taglist: [], Pushed: 2 days ago
- Image D, Taglist: ["alpha-2"], Pushed: 1 day ago

```
{
 "rules": [
 {
 "rulePriority": 1,
 "description": "Rule 1",
 "selection": {
 "tagStatus": "any",
 "countType": "imageCountMoreThan",
 "countNumber": 1
 },
 "action": {
 "type": "expire"
 }
 }
]
}
```

生命週期政策的邏輯會是：

- 規則 1 找出所有映像。會看到映像 A、B、C 與 D。除了最新映像外，應將所有映像設為過期。標記映像 A、B、C 為過期。
- 結果：映像 A、B 與 C 皆已過期。

## 範例：B

以下範例說明在單一規則中整合所有規則類型的生命週期政策。

**儲存庫內容：**

- Image A, Taglist: ["alpha-", "beta-1", "-1"], Pushed: 4 days ago
- Image B, Taglist: [], Pushed: 3 days ago
- Image C, Taglist: ["alpha-2"], Pushed: 2 days ago
- Image D, Taglist: ["git hash"], Pushed: 1 day ago
- Image E, Taglist: [], Pushed: 1 day ago

```
{
 "rules": [
 {
 "rulePriority": 1,
 "description": "Rule 1",
 "selection": {
 "tagStatus": "tagged",
 "tagPatternList": ["alpha*"],
 "countType": "imageCountMoreThan",
 "countNumber": 1
 },
 "action": {
 "type": "expire"
 }
 },
 {
 "rulePriority": 2,
 "description": "Rule 2",
 "selection": {
 "tagStatus": "untagged",
 "countType": "sinceImagePushed",
 "countUnit": "days",
 "countNumber": 1
 },
 "action": {
 "type": "expire"
 }
 },
 {
 "rulePriority": 3,
 "description": "Rule 3",
 "selection": {
 "tagStatus": "any",
```

```

 "countType": "imageCountMoreThan",
 "countNumber": 1
 },
 "action": {
 "type": "expire"
 }
}
]
}

```

生命週期政策的邏輯會是：

- 規則 1 找出含有前綴 alpha 標記的映像。找出映像 A 與映像 C。應保留最新映像並將其他標記為過期。標記映像 A 為過期。
- 規則 2 找出未標記的映像。找出映像 B 與映像 E。應標記所有超過一天的映像為過期。標記映像 B 為過期。
- 規則 3 找出所有映像。找出映像 A、B、C、D 與 E。應保留最新映像並將其他標記為過期。但是，無法標記映像 A、B、C 或 E，因為會以較高的優先順序規則來找出它們。標記映像 D 為過期。
- 結果：映像 A、B 與 D 皆已過期。

## 存檔範例

下列範例顯示存檔映像而非刪除映像的生命週期政策。

### 封存超過指定天數的影像

下列範例顯示生命週期政策，以超過 prod30 天的標籤封存映像：

```

{
 "rules": [
 {
 "rulePriority": 1,
 "description": "Archive production images older than 30 days",
 "selection": {
 "tagStatus": "tagged",
 "tagPatternList": ["prod*"],
 "countType": "sinceImagePushed",
 "countUnit": "days",
 "countNumber": 30
 },
 },
],
}

```

```
 "action": {
 "type": "transition",
 "targetStorageClass": "archive"
 }
 }
]
```

## 封存未在指定的天數內提取的映像

下列範例顯示生命週期政策，該政策會封存 90 天內尚未提取的影像：

```
{
 "rules": [
 {
 "rulePriority": 1,
 "description": "Archive images not pulled in 90 days",
 "selection": {
 "tagStatus": "any",
 "countType": "sinceImagePulled",
 "countUnit": "days",
 "countNumber": 90
 },
 "action": {
 "type": "transition",
 "targetStorageClass": "archive"
 }
 }
]
}
```

## 合併封存和過期規則

下列範例顯示的生命週期政策會封存超過 30 天的映像，然後永久過期已封存超過 365 天的映像：

### Note

封存映像的最短儲存期間為 90 天。您無法設定生命週期政策來刪除已存檔不到 90 天的映像。如果您必須刪除已封存不到 90 天的映像，則需要使用 `batch-delete-image` API，但需支付 90 天最低儲存持續時間的費用。

```
{
 "rules": [
 {
 "rulePriority": 1,
 "description": "Archive images older than 30 days",
 "selection": {
 "tagStatus": "any",
 "countType": "sinceImagePushed",
 "countUnit": "days",
 "countNumber": 30
 },
 "action": {
 "type": "transition",
 "targetStorageClass": "archive"
 }
 },
 {
 "rulePriority": 2,
 "description": "Expire images archived for more than 365 days",
 "selection": {
 "tagStatus": "any",
 "storageClass": "archive",
 "countType": "sinceImageTransitioned",
 "countUnit": "days",
 "countNumber": 365
 },
 "action": {
 "type": "expire"
 }
 }
]
}
```

## Amazon ECR 中的生命週期政策屬性

生命週期政策具有下列屬性。

若要查看生命週期政策的範例，請參閱 [Amazon ECR 中的生命週期政策範例](#)。如需使用 建立生命週期政策的說明 AWS CLI，請參閱 [建立生命週期政策 \(AWS CLI\)](#)。

## 規則優先順序

### rulePriority

類型：整數

必要：是

設定套用的規則之順序，從最低值到最高值。首先1套用優先順序為 的生命週期政策規則，接著套用優先順序2為 的規則，以此類推。當您新增規則到生命週期政策時，必須指定唯一的 rulePriority 值。政策中的值不需要跨規則循序。擁有 any 的 tagStatus 值必須有 rulePriority 的最高值與最後評估的值。

## Description

### description

類型：字串

必要：否

(選用) 說明生命週期政策中的規則用途。

## 標籤狀態

### tagStatus

類型：字串

必要：是

決定您新增的生命週期政策規則是否為映像指定標籤。可接受選項為 tagged、untagged、或 any。若您指定 any，所有映像都對規則進行評估。如果您指定 tagged，則還必須指定 tagPrefixList 值或 tagPatternList 值。如果您指定 untagged，則必須同時省略 tagPrefixList 和 tagPatternList。

## 標籤模式清單

### tagPatternList

類型：list[string]

必填：是，如果 tagStatus 設為已標記且未指定 tagPrefixList

為已標記的映像建立生命週期政策時，最佳做法是使用 tagPatternList 來指定預計會過期的標籤。您必須指定以逗號分隔的映像標籤模式清單，其中可能包含萬用字元 (\*)，以便您透過生命週期政策執行動作。例如，若您的映像標記為 prod、prod1、prod2 等等，您可能需使用標籤模式清單 prod\* 來指定所有標籤。若您指定多個標籤，只會選擇含有所有指定標籤的映像。

#### Important

每個字串最多可有 4 個萬用字元 (\*)，例如 ["\*test\*1\*2\*3", "test\*1\*2\*3\*"] 是有效字串，但 ["test\*1\*2\*3\*4\*5\*6"] 則為無效。

## 標籤字首清單

### tagPrefixList

類型：list[string]

必填：是，如果 tagStatus 設為已標記且未指定 tagPatternList

僅在您指定 "tagStatus": "tagged" 且未指定 tagPatternList 時使用。您必須指定以逗號分隔的映像標籤前綴清單，用以使用生命週期政策來採取動作。例如，若您的映像被標記為 prod、prod1、prod2 以此類推，您可能需要使用標籤前綴 prod 來指定所有映像。若您指定多個標籤，只會選擇含有所有指定標籤的映像。

## 儲存類別

### storageClass

類型：字串

必要：是，如果 countType 為 sinceImageTransitioned

規則只會選取此儲存類別的影像。使用 `imageCountMoreThan`、`sinceImagePushed` 或 `countType` 的 `sinceImagePulled`，唯一支援的值為 `standard`。使用計數類型 `sinceImageTransitioned`，這是必要的，唯一支援的值是 `archive`。如果您省略此選項，`standard` 則會使用的值。

## 計數類型

### `countType`

類型：字串

必要：是

指定計數類型以套用到映像。

若 `countType` 設為 `imageCountMoreThan`，您也指定 `countNumber` 來建立設定存在於您的儲存庫中的映像數量限制之規則。如果 `countType` 設為 `sinceImagePushed`、`sinceImagePulled` 或 `sinceImageTransitioned`，您也可以指定 `countUnit` 和 `countNumber`，以指定存在於儲存庫中映像的時間限制。

## 計數單位

### `countUnit`

類型：字串

必要：是，只有在 `countType` 設為 `sinceImagePushed`、`sinceImagePulled` 或 `sinceImageTransitioned`

除了代表天數的 `days` 外，請指定 `countNumber` 的技術單位來作為時間單位。

這只能在 `countType` 為 `sinceImagePushed`、`sinceImagePulled` 或 `sinceImageTransitioned`；如果您在 `countType` 為任何其他值時指定計數單位，則會發生錯誤。

## Count (計數)

### `countNumber`

類型：整數

必要：是

指定計數號碼。可接受的值為正整數 (0 不是接受值)。

如果使用的 `countType` 為 `imageCountMoreThan`，那麼值便是您想要保留於儲存庫中的最大映像數量。如果使用的 `countType` 為 `sinceImagePushed`，那麼值便是您想要保留於儲存庫中的映像存在時間上限。如果 `countType` 使用的是 `sinceImagePulled`，則值為自上次提取映像以來的天數上限。如果 `countType` 使用的是 `sinceImageTransitioned`，則值為自封存映像以來的天數上限。

## Action

`type`

類型：字串

必要：是

指定一種動作類型。支援的值為 `expire` (刪除映像) 和 `transition` (將映像移至封存儲存)。

`targetStorageClass`

類型：字串

必要：是，如果 `type` 為 `transition`

您希望生命週期政策轉換映像的儲存類別。 `archive` 是唯一支援的值。

# 提取時間更新排除

Amazon ECR 會更新每次提取的 `LastRecordedPullTime` 時間戳記，但 AWS Inspector 提取除外。提取時間更新排除項目可讓您指定在提取影像時不應更新影像提取時間的 IAM 角色 ARNs，例如由第三方掃描器提取（例如 CrowdStrike、Snyk 和 Trivy）。這適用於用於測試或 CI/CD 用途的影像，而您不希望提取時間影響生命週期政策決策。

當排除清單中的角色提取映像時，提取時間保持不變。任何其他角色都會繼續更新提取時間（目前行為）。每個帳戶最多可設定 100 個排除。

## 管理提取時間更新排除

若要管理提取時間更新排除，您需要下列 IAM 許可：

- `ecr:CreatePullTimeUpdateExclusion` – 准許將角色 ARN 新增至排除清單。
- `ecr>DeletePullTimeUpdateExclusion` – 准許從排除清單中移除角色 ARN。
- `ecr:ListPullTimeUpdateExclusions` – 准許列出排除清單中的所有角色 ARNs。

### Note

您不需要 `iam:PassRole` 許可。Amazon ECR 不會擔任角色來執行動作；它只會使用排除組態 ARNs 來判斷是否應更新映像的提取時間。

您可以使用 Amazon ECR 主控台或 CLI AWS 管理提取時間更新排除。

### AWS 管理主控台

#### 管理提取時間更新排除 (AWS 管理主控台)

1. 在 <https://console.aws.amazon.com/ecr/private-registry/repositories> 開啟 Amazon ECR 主控台
2. 從導覽列中選擇區域。
3. 在導覽窗格中，選擇私有登錄檔、功能和設定，然後選擇提取時間更新排除。
4. 若要新增排除，請選擇新增排除，輸入角色 ARN，然後選擇新增。
5. 若要移除排除，請從清單中選擇角色 ARN，然後選擇刪除。

- 若要檢視所有排除，清單會顯示所有已設定的角色 ARNs。

## AWS CLI

### 建立提取時間更新排除

- 使用 `create-pull-time-update-exclusion` 命令將角色 ARN 新增至排除清單：

```
aws ecr create-pull-time-update-exclusion \
 --role-arn arn:aws:iam::123456789012:role/scanner-role
```

命令會傳回角色 ARN 和建立時間戳記：

```
{
 "roleArn": "arn:aws:iam::123456789012:role/scanner-role",
 "createdAt": 1745531331.0
}
```

### 刪除提取時間更新排除

- 使用 `delete-pull-time-update-exclusion` 命令從排除清單中移除角色 ARN：

```
aws ecr delete-pull-time-update-exclusion \
 --role-arn arn:aws:iam::123456789012:role/scanner-role
```

命令會傳回已刪除的角色 ARN：

```
{
 "roleArn": "arn:aws:iam::123456789012:role/scanner-role"
}
```

### 列出提取時間更新排除

- 使用 `list-pull-time-update-exclusions` 命令列出排除清單中的所有角色 ARNs：

```
aws ecr list-pull-time-update-exclusions
```

如果未設定排除，命令會傳回空清單：

```
{
 "pullTimeUpdateExclusions": []
}
```

如果已設定排除，命令會傳回角色 ARNs 的清單：

```
{
 "pullTimeUpdateExclusions": [
 "arn:aws:iam::123456789012:role/security-role"
]
}
```

2. 若要分頁結果，請使用 `--max-results` 和 `--next-token` 參數：

```
aws ecr list-pull-time-update-exclusions \
 --max-results 4
```

命令會傳回最多指定數量的結果，`nextToken` 如果有更多結果可用，則傳回：

```
{
 "pullTimeUpdateExclusions": [
 "arn:aws:iam::123456789012:role/security-role1",
 "arn:aws:iam::123456789012:role/security-role2",
 "arn:aws:iam::123456789012:role/security-role3",
 "arn:aws:iam::123456789012:role/security-role4"
],
 "nextToken": "ukD72mdD/mC8b5xV3susmJzzaTgp3hKwR9nRUW1yZZ79..."
}
```

若要擷取結果的下一頁，請使用上一個回應 `nextToken` 中的：

```
aws ecr list-pull-time-update-exclusions \
 --max-results 4 \
 --next-token ukD72mdD/mC8b5xV3susmJzzaTgp3hKwR9nRUW1yZZ79...
```

## 提取時間更新排除的考量事項

使用提取時間更新排除項目時，請考慮下列事項：

- 列出排除項目的預設頁面大小為 100。您可以搭配 `maxResults` 和 `nextToken` 參數使用分頁。
- 僅接受正確 ARNs 格式的有效 IAM 角色 ARN。
- 如果您嘗試建立已存在的排除，您將會收到 `ExclusionAlreadyExistsException` 錯誤。如果您嘗試刪除不存在的排除，您將會收到 `ExclusionNotFoundException` 錯誤。

# Amazon Elastic Container Registry 的安全性

的雲端安全性 AWS 是最高優先順序。身為 AWS 客戶，您可以受益於資料中心和網路架構，該架構專為滿足最安全敏感組織的需求而建置。

安全性是 AWS 與您之間共同責任。[共同責任模型](#) 將此描述為雲端的安全和雲端內的安全：

- 雲端的安全性 – AWS 負責保護在 Cloud AWS 中執行 AWS 服務的基礎設施。AWS 也為您提供可安全使用的服務。在 [AWS 合規計劃](#) 中，第三方稽核員會定期測試並驗證我們的安全功效。若要了解適用於 Amazon ECR 的合規計劃，請參閱 [合規計劃範圍內的 AWS 服務](#)。
- 雲端的安全性 – 您的責任取決於您使用 AWS 的服務。您也必須對其他因素負責，包括資料的機密性、您的公司的要求和適用法律和法規。

本文件有助於您了解如何在使用 Amazon ECR 時套用共同責任模型。下列主題說明如何將 Amazon ECR 設定為符合您的安全與合規目標。您也會了解如何使用其他 AWS 服務來協助您監控和保護 Amazon ECR 資源。

## 主題

- [Amazon Elastic Container Registry 的 Identity and Access Management](#)
- [Amazon ECR 中的資料保護](#)
- [Amazon Elastic Container Registry 的合規驗證](#)
- [Amazon Elastic Container Registry 的基礎設施安全性](#)
- [預防跨服務混淆代理人](#)

## Amazon Elastic Container Registry 的 Identity and Access Management

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 管理員可以控制驗證 (已登入) 和授權 (具有許可) 來使用 Amazon ECR 資源。IAM 是您可以免費使用 AWS 服務的。

## 主題

- [目標對象](#)
- [使用身分驗證](#)

- [使用政策管理存取權](#)
- [Amazon Elastic Container Registry 如何與 IAM 搭配使用](#)
- [Amazon Elastic Container Registry 身分型政策的範例](#)
- [使用標籤型存取控制](#)
- [AWS Amazon Elastic Container Registry 的 受管政策](#)
- [使用 Amazon ECR 的服務連結角色](#)
- [針對 Amazon Elastic Container Registry Identity and Access 進行故障診斷](#)

## 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 [針對 Amazon Elastic Container Registry Identity and Access 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 [Amazon Elastic Container Registry 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 [Amazon Elastic Container Registry 身分型政策的範例](#))

## 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](#)。

## IAM 使用者和群組

IAM 使用者[https://docs.aws.amazon.com/IAM/latest/UserGuide/id\\_users.html](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](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 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。您無法在以資源為基礎的政策中使用來自 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，請參閱《IAM 使用者指南》中的[工作階段政策](#)。

## 多種政策類型

當請求套用多種類型的政策時，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](#)。

## Amazon Elastic Container Registry 如何與 IAM 搭配使用

在您使用 IAM 管理對 Amazon ECR 的存取權之前，您應該瞭解哪些 IAM 功能可以與 Amazon ECR 搭配使用。若要全面了解 Amazon ECR 和其他 AWS 服務如何與 IAM 搭配使用，請參閱《IAM 使用者指南》中的與 IAM [AWS 搭配使用的服務](#)。

### 主題

- [Amazon ECR 身分型政策](#)
- [Amazon ECR 資源型政策](#)
- [以 Amazon ECR 標籤為基礎的授權](#)
- [Amazon ECR IAM 角色](#)

## Amazon ECR 身分型政策

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。Amazon ECR 支援特定動作、資源和條件索引鍵。若要了解您在 JSON 政策中使用的所有元素，請參閱 IAM 使用者指南中的 [JSON 政策元素參考](#)。

### 動作

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。

Amazon ECR 中的政策動作會在動作之前使用下列字首：`ecr:`。例如，若要授予某人使用 Amazon ECR CreateRepository API 操作建立 Amazon ECR 儲存庫的許可，請在其政策中加入 `ecr:CreateRepository` 動作。政策陳述式必須包含 Action 或 NotAction 元素。Amazon ECR 會定義自己的一組動作，描述您可以使用此服務執行的任務。

若要在單一陳述式中指定多個動作，請用逗號分隔，如下所示：

```
"Action": [
 "ecr:action1",
 "ecr:action2"
```

您也可以使用萬用字元 (\*) 來指定多個動作。例如，若要指定開頭是 Describe 文字的所有動作，請包含以下動作：

```
"Action": "ecr:Describe*"
```

若要查看 Amazon ECR 動作的清單，請參閱《IAM 使用者指南》中的 [Amazon Elastic Container Registry 的動作、資源與條件索引鍵](#)。

### Resources

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": "*"
```

Amazon ECR 儲存庫資源有以下 ARN：

```
arn:${Partition}:ecr:${Region}:${Account}:repository/${Repository-name}
```

如需 ARNs 格式的詳細資訊，請參閱 [Amazon Resource Name \(ARNs AWS 和服務命名空間\)](#)。

例如，若要在陳述式中指定 us-east-1 區域的 my-repo 儲存庫，請使用下列 ARN：

```
"Resource": "arn:aws:ecr:us-east-1:123456789012:repository/my-repo"
```

若要指定屬於特定帳戶的所有儲存庫，請使用萬用字元 (\*)：

```
"Resource": "arn:aws:ecr:us-east-1:123456789012:repository/*"
```

若要在單一陳述式中指定多項資源，請使用逗號分隔 ARN。

```
"Resource": [
 "resource1",
 "resource2"
```

若要查看 Amazon ECR 資源類型及其 ARN 的清單，請參閱《IAM 使用者指南》中的 [Amazon Elastic Container Registry 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 [Amazon Elastic Container Registry 定義的動作](#)。

## 條件索引鍵

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](#)。

Amazon ECR 會定義自己的一組條件索引鍵，也支援使用一些全域條件索引鍵。若要查看所有 AWS 全域條件金鑰，請參閱《IAM 使用者指南》中的 [AWS 全域條件內容金鑰](#)。

大部分 Amazon ECR 動作均支援 `aws:ResourceTag` 和 `ecr:ResourceTag` 條件索引鍵。如需詳細資訊，請參閱[使用標籤型存取控制](#)。

若要查看 Amazon ECR 條件索引鍵的清單，請參閱《IAM 使用者指南》中的 [Amazon Elastic Container Registry 定義的條件索引鍵](#)。若要了解您可以搭配哪些動作和資源使用條件索引鍵，請參閱 [Amazon Elastic Container Registry 定義的動作](#)。

## 範例

若要檢視 Amazon ECR 身分型政策的範例，請參閱 [Amazon Elastic Container Registry 身分型政策的範例](#)。

## Amazon ECR 資源型政策

資源型政策是 JSON 政策文件，這些文件會指定指定的委託人可對 Amazon ECR 資源以及在怎樣的條件下執行哪些動作。Amazon ECR 支援 Amazon ECR 存放庫的資源型許可政策。資源型政策可讓您依資源將使用許可授予至其他帳戶。您也可以使用資源型政策來允許 AWS 服務存取 Amazon ECR 儲存庫。

若要啟用跨帳戶存取，您可以指定在其他帳戶內的所有帳戶或 IAM 實體，作為[資源型政策的委託人](#)。新增跨帳戶主體至資源型政策，只是建立信任關係的一半。當主體和資源位於不同的 AWS 帳戶中時，您還必須授予主體實體存取資源的許可。透過將身分型政策連接到實體來授予許可。不過，如果資源型政策將存取權授予相同帳戶中的委託人，則不需要身分型政策中的其他 Amazon ECR 儲存庫許可。如需詳細資訊，請參閱《IAM 使用者指南》中的 [IAM 角色與資源型政策有何差異](#)。

Amazon ECR 服務僅支援一種稱為儲存庫政策之資源型政策，且已連接到儲存庫。此政策會定義哪些委託人實體 (帳戶、使用者、角色和聯合身分使用者) 可在該儲存庫上執行動作。若要了解如何將資源型政策連接到儲存庫，請參閱 [Amazon ECR 中的私有儲存庫政策](#)。

### Note

在 Amazon ECR 儲存器政策中，政策元素 `Sid` 支援 IAM 政策中不支援的附加字元和間距。

## 範例

若要檢視 Amazon ECR 資源型政策的範例，請參閱 [Amazon ECR 中的私有儲存庫政策範例](#)。

## 以 Amazon ECR 標籤為基礎的授權

您可以將標籤連接至 Amazon ECR 資源，或是在請求中將標籤傳遞至 Amazon ECR。若要根據標籤控制存取，請使用 `ecr: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。如需標記 Amazon ECR 資源的詳細資訊，請參閱 [在 Amazon ECR 中標記私有儲存庫](#)。

若要檢視身分型政策範例，以根據該資源上的標籤來限制存取資源，請參閱 [使用標籤型存取控制](#)。

## Amazon ECR IAM 角色

[IAM 角色](#) 是您 AWS 帳戶中具有特定許可的實體。

### 搭配使用臨時憑證與 Amazon ECR

您可以搭配聯合使用暫時憑證、擔任 IAM 角色，或是擔任跨帳戶角色。您可以透過呼叫 [AssumeRole](#) 或 [GetFederationToken](#) 等 AWS STS API 操作來取得臨時安全登入資料。

Amazon ECR 支援使用臨時憑證。

### 服務連結角色

[服務連結角色](#) 可讓 AWS 服務存取其他服務中的資源，以代表您完成動作。服務連結角色會顯示在您的 IAM 帳戶中，並由該服務所擁有。IAM 管理員可以檢視，但不能編輯服務連結角色的許可。

Amazon ECR 支援服務連結角色。如需詳細資訊，請參閱 [使用 Amazon ECR 的服務連結角色](#)。

## Amazon Elastic Container Registry 身分型政策的範例

根據預設，使用者和角色不具備建立或修改 Amazon ECR 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 Amazon ECR 所定義之動作和資源類型的詳細資訊，包括每種資源類型的 ARN 格式，請參閱《服務授權參考》中的 [Amazon Elastic Container Registry 的動作、資源和條件索引鍵](#)。

若要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 [在 JSON 索引標籤上建立政策](#)。

### 主題

- [政策最佳實務](#)
- [使用 Amazon ECR 主控台](#)
- [允許使用者檢視自己的許可](#)
- [存取一個 Amazon ECR 儲存庫](#)

## 政策最佳實務

身分型政策會判斷您帳戶中的某個人員是否可以建立、存取或刪除 Amazon ECR 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，以進一步減少許可。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 Amazon ECR 主控台

若要存取 Amazon Elastic Container Registry 主控台，您必須擁有最基本的一組許可。這些許可必須允許您列出和檢視 AWS 帳戶中 Amazon ECR 資源的詳細資訊。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。

為了確保這些實體仍然可以使用 Amazon ECR 主控台，請將 AmazonEC2ContainerRegistryReadOnly AWS 受管政策新增至實體。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](#)。

若要檢視此政策的許可，請參閱《AWS 受管政策參考》中的[AmazonElasticContainerRegistryPublicReadOnly](#)。

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，只需允許存取符合您嘗試執行之 API 操作的動作就可以了。

## 允許使用者檢視自己的許可

此範例會示範如何建立政策，允許 IAM 使用者檢視附加到他們使用者身分的內嵌及受管政策。此政策包含在主控台或使用或 AWS CLI AWS API 以程式設計方式完成此動作的許可。

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Sid": "ViewOwnUserInfo",
 "Effect": "Allow",
 "Action": [
 "iam:GetUserPolicy",
 "iam:ListGroupForUser",
 "iam:ListAttachedUserPolicies",
 "iam:ListUserPolicies",
 "iam:GetUser"
],
 "Resource": ["arn:aws:iam::*:user/${aws:username}"]
 },
 {
 "Sid": "NavigateInConsole",
 "Effect": "Allow",
 "Action": [
 "iam:GetGroupPolicy",
 "iam:GetPolicyVersion",
 "iam:GetPolicy",
 "iam:ListAttachedGroupPolicies",
 "iam:ListGroupPolicies",
 "iam:ListPolicyVersions",
 "iam:ListPolicies",
 "iam:ListUsers"
]
 }
]
}
```

```
],
 "Resource": "*"
 }
]
}
```

## 存取一個 Amazon ECR 儲存庫

在此範例中，您想要授予 AWS 帳戶中的使用者存取其中一個 Amazon ECR 儲存庫 `my-repo`。您也希望允許使用者推送、提取及列出映像。

### JSON

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Sid": "GetAuthorizationToken",
 "Effect": "Allow",
 "Action": [
 "ecr:GetAuthorizationToken"
],
 "Resource": "*"
 },
 {
 "Sid": "ManageRepositoryContents",
 "Effect": "Allow",
 "Action": [
 "ecr:BatchCheckLayerAvailability",
 "ecr:GetDownloadUrlForLayer",
 "ecr:GetRepositoryPolicy",
 "ecr:DescribeRepositories",
 "ecr:ListImages",
 "ecr:DescribeImages",
 "ecr:BatchGetImage",
 "ecr:InitiateLayerUpload",
 "ecr:UploadLayerPart",
 "ecr:CompleteLayerUpload",
 "ecr:PutImage"
],
 "Resource": "arn:aws:ecr:us-east-1:123456789012:repository/my-repo"
 }
]
}
```

```
]
}
```

## 使用標籤型存取控制

Amazon ECR CreateRepository API 動作可讓您在建立儲存庫時指定標籤。如需詳細資訊，請參閱 [在 Amazon ECR 中標記私有儲存庫](#)。

若要讓使用者在建立時標記儲存庫，他們必須具備建立資源之動作 (如 `ecr:CreateRepository`) 的使用許可。若標籤於資源建立動作指定，Amazon 會針對 `ecr:CreateRepository` 動作執行其他授權，以確認使用者具備建立標籤的許可。

您可以透過 IAM 政策來使用標籤型存取控制，範例如下。

以下政策只允許使用者將儲存庫建立或標記為 `key=environment,value=dev`。

### JSON

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Sid": "AllowCreateTaggedRepository",
 "Effect": "Allow",
 "Action": [
 "ecr:CreateRepository"
],
 "Resource": "*",
 "Condition": {
 "StringEquals": {
 "aws:RequestTag/environment": "dev"
 }
 }
 },
 {
 "Sid": "AllowTagRepository",
 "Effect": "Allow",
 "Action": [
 "ecr:TagResource"
],
 "Resource": "*",
 }
]
}
```

```

 "Condition": {
 "StringEquals": {
 "aws:RequestTag/environment": "dev"
 }
 }
]
}

```

下列政策可讓使用者從所有儲存庫提取映像，除非它們標記為 `key=environment,value=prod`。

JSON

```

{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Effect": "Allow",
 "Action": [
 "ecr:BatchGetImage",
 "ecr:GetDownloadUrlForLayer"
],
 "Resource": "*"
 },
 {
 "Effect": "Deny",
 "Action": [
 "ecr:BatchGetImage",
 "ecr:GetDownloadUrlForLayer"
],
 "Resource": "*",
 "Condition": {
 "StringEquals": {
 "ecr:ResourceTag/environment": "prod"
 }
 }
 }
]
}

```

## AWS Amazon Elastic Container Registry 的 受管政策

AWS 受管政策是由 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](#)。

Amazon ECR 提供數個受管政策，您可以連接到 IAM 身分或 Amazon EC2 執行個體。這些受管政策允許對 Amazon ECR 資源和 API 操作的存取進行不同層級的控制。如需這些政策中所提及之各 API 操作的詳細資訊，請參閱 Amazon Elastic Container Registry API 參考中的 [Actions](#) (動作)。

### 主題

- [AmazonEC2ContainerRegistryFullAccess](#)
- [AmazonEC2ContainerRegistryPowerUser](#)
- [AmazonEC2ContainerRegistryPullOnly](#)
- [AmazonEC2ContainerRegistryReadOnly](#)
- [AWSECRPullThroughCache\\_ServiceRolePolicy](#)
- [ECRReplicationServiceRolePolicy](#)
- [ECRTemplateServiceRolePolicy](#)
- [AWS 受管政策的 Amazon ECR 更新](#)

### AmazonEC2ContainerRegistryFullAccess

您可將 AmazonEC2ContainerRegistryFullAccess 政策連接到 IAM 身分。此政策會授予 Amazon ECR 資源的管理存取權，並授予 IAM 身分（例如使用者、群組或角色）存取權給與 Amazon ECR 整合 AWS 的服務，以使用所有 Amazon ECR 功能。使用此政策允許存取 中提供的所有 Amazon ECR 功能 AWS 管理主控台。

若要檢視此政策的許可，請參閱《AWS 受管政策參考》中的 [AmazonEC2ContainerRegistryFullAccess](#)。

## AmazonEC2ContainerRegistryPowerUser

您可將 AmazonEC2ContainerRegistryPowerUser 政策連接到 IAM 身分。此政策授予管理許可，允許 IAM 使用者讀取和寫入儲存庫，但不允許他們刪除儲存庫或變更套用於他們的政策文件。

若要檢視此政策的許可，請參閱《AWS 受管政策參考》中的

[AmazonEC2ContainerRegistryPowerUser](#)。

## AmazonEC2ContainerRegistryPullOnly

您可將 AmazonEC2ContainerRegistryPullOnly 政策連接到 IAM 身分。此政策授予從 Amazon ECR 提取容器映像的許可。如果登錄檔已啟用提取快取，它也會允許提取從上游登錄檔匯入映像。

若要檢視此政策的許可，請參閱《AWS 受管政策參考》中的

[AmazonEC2ContainerRegistryPullOnly](#)。

## AmazonEC2ContainerRegistryReadOnly

您可將 AmazonEC2ContainerRegistryReadOnly 政策連接到 IAM 身分。此政策向 Amazon ECR 授予唯讀許可。這包括在儲存庫中列出儲存庫和映像的功能。它還包括使用 Docker CLI 從 Amazon ECR 提取映像的能力。

若要檢視此政策的許可，請參閱《AWS 受管政策參考》中的

[AmazonEC2ContainerRegistryReadOnly](#)。

## AWSECRPullThroughCache\_ServiceRolePolicy

您無法將 AWSECRPullThroughCache\_ServiceRolePolicy 受管 IAM 政策連接至 IAM 實體。此政策會連接至服務連結角色，可讓 Amazon ECR 透過提取快取工作流程將映像推送到儲存庫。如需詳細資訊，請參閱[用於提取快取的 Amazon ECR 服務連結角色](#)。

## ECRReplicationServiceRolePolicy

您無法將 ECRReplicationServiceRolePolicy 受管 IAM 政策連接至 IAM 實體。此政策會連接到服務連結角色，而此角色可讓 Amazon ECR 代表您執行動作。如需詳細資訊，請參閱[使用 Amazon ECR 的服務連結角色](#)。

## ECRTemplateServiceRolePolicy

您無法將 ECRTemplateServiceRolePolicy 受管 IAM 政策連接至 IAM 實體。此政策會連接到服務連結角色，而此角色可讓 Amazon ECR 代表您執行動作。如需詳細資訊，請參閱[使用 Amazon ECR 的服務連結角色](#)。

## AWS 受管政策的 Amazon ECR 更新

檢視自此服務開始追蹤 Amazon ECR AWS 受管政策更新以來的詳細資訊。如需有關此頁面變更的自動提醒，請訂閱 Amazon ECR 文件歷史記錄頁面上的 RSS 摘要。

| 變更                                                                 | 描述                                                                                                                                                        | Date             |
|--------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------|------------------|
| <a href="#">用於提取快取的 Amazon ECR 服務連結角色</a> – 更新現有政策                 | Amazon ECR 將新的許可新增到 <code>AWSECRPullThroughCache_ServiceRolePolicy</code> 政策。這些許可允許 Amazon ECR 從 ECR 私有登錄檔提取映像。使用提取快取規則從另一個 Amazon ECR 私有登錄檔快取映像時，這是必要的。  | 2025 年 3 月 12 日  |
| <a href="#">AmazonEC2ContainerRegistryPullOnly</a> – 新政策           | Amazon ECR 新增了新的政策，授予 Amazon ECR 僅提取許可。                                                                                                                   | 2024 年 10 月 10 日 |
| <a href="#">ECRTemplateServiceRolePolicy</a> – 新政策                 | Amazon ECR 已新增新政策。此政策與儲存庫建立範本功能 <code>ECRTemplateServiceRolePolicy</code> 的服務連結角色相關聯。                                                                     | 2024 年 6 月 20 日  |
| <a href="#">AWSECRPullThroughCache_ServiceRolePolicy</a> – 更新至現有政策 | Amazon ECR 將新的許可新增到 <code>AWSECRPullThroughCache_ServiceRolePolicy</code> 政策。這些許可允許 Amazon ECR 擷取 Secrets Manager 秘密的加密內容。這在使用提取快取規則從需要驗證的上游登錄檔快取映像時是必要的。 | 2023 年 11 月 15 日 |

| 變更                                                              | 描述                                                                                             | Date             |
|-----------------------------------------------------------------|------------------------------------------------------------------------------------------------|------------------|
| <a href="#">AWSECRPullThroughCache_ServiceRolePolicy</a> : 全新政策 | Amazon ECR 已新增新政策。此政策與提取快取功能的 AWSServiceRoleForECRPullThroughCache 服務連結角色相關聯。                  | 2021 年 11 月 29 日 |
| <a href="#">ECRReplicationServiceRolePolicy</a> : 全新政策          | Amazon ECR 已新增新政策。此政策與複寫功能的 AWSServiceRoleForECRReplication 服務連結角色相關聯。                         | 2020 年 12 月 4 日  |
| <a href="#">AmazonEC2ContainerRegistryFullAccess</a> – 更新現有政策   | Amazon ECR 將新的許可新增到 AmazonEC2ContainerRegistryFullAccess 政策。這些許可允許委託人建立 Amazon ECR 服務連結角色。     | 2020 年 12 月 4 日  |
| <a href="#">AmazonEC2ContainerRegistryReadOnly</a> – 更新現有政策     | Amazon ECR 將新的許可新增到 AmazonEC2ContainerRegistryReadOnly 政策，允許委託人讀取生命週期政策、列出標籤，以及描述映像的掃描問題清單。    | 2019 年 12 月 10 日 |
| <a href="#">AmazonEC2ContainerRegistryPowerUser</a> – 更新現有政策    | Amazon ECR 將新的許可新增到 AmazonEC2ContainerRegistryPowerUser 政策。它們允許委託人讀取生命週期政策、列出標籤，以及描述映像的掃描問題清單。 | 2019 年 12 月 10 日 |

| 變更                                                            | 描述                                                                                                                   | Date             |
|---------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------|------------------|
| <a href="#">AmazonEC2ContainerRegistryFullAccess</a> – 更新現有政策 | Amazon ECR 將新的許可新增到 AmazonEC2ContainerRegistryFullAccess 政策。它們允許主體查詢 CloudTrail 擷取的管理事件或 AWS CloudTrail Insights 事件。 | 2017 年 11 月 10 日 |
| <a href="#">AmazonEC2ContainerRegistryReadOnly</a> – 更新現有政策   | Amazon ECR 將新的許可新增到 AmazonEC2ContainerRegistryReadOnly 政策。他們允許委託人描述 Amazon ECR 映像。                                   | 2016 年 10 月 11 日 |
| <a href="#">AmazonEC2ContainerRegistryPowerUser</a> – 更新現有政策  | Amazon ECR 將新的許可新增到 AmazonEC2ContainerRegistryPowerUser 政策。他們允許委託人描述 Amazon ECR 映像。                                  | 2016 年 10 月 11 日 |
| <a href="#">AmazonEC2ContainerRegistryReadOnly</a> – 新政策      | Amazon ECR 新增了將唯讀許可授予 Amazon ECR 的新政策。這些許可包括在儲存庫中列出儲存庫和映像的功能。它們還包括使用 Docker CLI 從 Amazon ECR 提取映像的能力。                | 2015 年 12 月 21 日 |
| <a href="#">AmazonEC2ContainerRegistryPowerUser</a> – 新政策     | Amazon ECR 新增了新的政策，授予管理許可，允許使用者讀取和寫入儲存庫，但不允許他們刪除儲存庫或變更套用的政策文件。                                                       | 2015 年 12 月 21 日 |

| 變更                                                         | 描述                                          | Date             |
|------------------------------------------------------------|---------------------------------------------|------------------|
| <a href="#">AmazonEC2ContainerRegistryFullAccess</a> – 新政策 | Amazon ECR 已新增新政策。此政策授予 Amazon ECR 的完整存取權限。 | 2015 年 12 月 21 日 |
| Amazon ECR 開始追蹤變更                                          | Amazon ECR 開始追蹤 AWS 受管政策的變更。                | 2021 年 6 月 24 日  |

## 使用 Amazon ECR 的服務連結角色

Amazon Elastic Container Registry (Amazon ECR) 使用 AWS Identity and Access Management (IAM) [服務連結角色](#) 來提供使用複寫和提取快取功能所需的許可。服務連結角色是直接連結至 Amazon ECR 的一種特殊 IAM 角色類型。服務連結的角色是由 Amazon ECR 預先定義。其中包含了該服務需要的所有許可，可支援私有登錄檔的複寫和提取快取功能。設定登錄檔的複寫或提取快取之後，系統將代表您自動建立服務連結角色。如需詳細資訊，請參閱[Amazon ECR 中的私有登錄檔設定](#)。

服務連結角色可讓使用 Amazon ECR 設定複寫和提取快取的過程更為輕鬆。這是因為使用它，您不必手動新增所有必要的許可。Amazon ECR 定義其服務連結角色的許可，除非另有定義，否則僅有 Amazon ECR 可以擔任其角色。已定義的許可包括信任政策和許可政策。許可政策無法附加到其他任何 IAM 實體。

您只能在登錄檔停用複寫或提取快取後，才能刪除對應的服務連結角色。這可確保您不會意外移除 Amazon ECR 針對這些功能所要求的許可。

關於支援服務連結角色的其他服務，如需相關資訊，請參閱[與 IAM 搭配運作的 AWS 服務](#)。在此連結至頁面上，在 Service-linked role (服務連結角色) 欄位中尋找具有 Yes (是) 的服務。選擇具有連結的 Yes (是)，以檢視該服務的相關服務連結角色文件。

### 主題

- [Amazon ECR 服務連結角色的支援區域](#)
- [用於複寫的 Amazon ECR 服務連結角色](#)
- [用於提取快取的 Amazon ECR 服務連結角色](#)
- [適用於儲存庫建立範本的 Amazon ECR 服務連結角色](#)

## Amazon ECR 服務連結角色的支援區域

Amazon ECR 在所有提供 Amazon ECR 服務的區域中支援使用服務連結的角色。如需 Amazon ECR 區域可用性的詳細資訊，請參閱《[AWS 區域與端點](#)》。

## 用於複寫的 Amazon ECR 服務連結角色

Amazon ECR 使用名為 `AWSServiceRoleForECRReplication` 的服務連結角色，可讓 Amazon ECR 跨多個帳戶複寫映像。

### Amazon ECR 的服務連結角色許可

`AWSServiceRoleForECRReplication` 服務連結角色信任下列服務可擔任該角色：

- `replication.ecr.amazonaws.com`

以下 `ECRReplicationServiceRolePolicy` 角色許可政策允許 Amazon ECR 對資源使用以下動作：

### JSON

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Effect": "Allow",
 "Action": [
 "ecr:CreateRepository",
 "ecr:ReplicateImage"
],
 "Resource": "*"
 }
]
}
```

#### Note

`ReplicateImage` 是 Amazon ECR 用於複寫的內部 API，無法直接呼叫。

您必須設定許可，IAM 實體 (例如，使用者、群組或角色) 才可建立、編輯或刪除服務連結角色。如需詳細資訊，請參閱《IAM 使用者指南》中的[服務連結角色許可](#)。

### 建立 Amazon ECR 的服務連結角色

您不需要手動建立 Amazon ECR 服務連結角色。當您在 AWS 管理主控台 AWS CLI、或 AWS API 中設定登錄檔的複寫設定時，Amazon ECR 會為您建立服務連結角色。

若您刪除此服務連結角色，之後需要再次建立，您可以在帳戶中使用相同程序重新建立角色。當您設定登錄檔的複寫設定時，Amazon ECR 會再次為您建立服務連結角色。

### 編輯 Amazon ECR 的服務連結角色

Amazon ECR 不允許手動編輯 AWSServiceRoleForECRReplication 服務連結角色。因為可能有各種實體會參考服務連結角色，所以您無法在建立角色之後變更其名稱。然而，您可使用 IAM 來編輯角色描述。如需詳細資訊，請參閱《IAM 使用者指南》中的[編輯服務連結角色](#)。

### 刪除 Amazon ECR 的服務連結角色

若您不再使用需要服務連結角色的功能或服務，我們建議您刪除該角色。如此一來，您就沒有未主動監控或維護的未使用實體。不過，您必須在每個區域中先移除您登錄檔的複寫組態，才能手動刪除服務連結角色。

#### Note

如果您嘗試在 Amazon ECR 服務仍在角色時刪除資源，則刪除動作可能會失敗。若此情況發生，請等待數分鐘後並再次嘗試。

### 刪除 AWSServiceRoleForECRReplication 所使用的 Amazon ECR 資源

1. 在 <https://console.aws.amazon.com/ecr/> 開啟 Amazon ECR 主控台。
2. 從導覽列，選擇您複寫組態所設定的區域。
3. 在導覽窗格中，選擇 Private registry (私有登錄檔)。
4. 在私有登錄檔頁面上，於複寫組態區段中，選擇編輯。
5. 若要刪除所有複寫規則，請選擇全部刪除。此步驟需要確認。

### 使用 IAM 手動刪除服務連結角色

使用 IAM 主控台、AWS CLI、或 AWS API 來刪除 `AWSServiceRoleForECRReplication` 服務連結角色。如需詳細資訊，請參閱《IAM 使用者指南》中的[刪除服務連結角色](#)。

## 用於提取快取的 Amazon ECR 服務連結角色

Amazon ECR 使用名為 `AWSServiceRoleForECRPullThroughCache` 的服務連結角色，其允許 Amazon ECR 代表您執行動作，以完成提取快取動作。如需提取快取的詳細資訊，請參閱「[用於控制提取快取、推送時建立或複寫動作期間建立的儲存庫的範本](#)」。

Amazon ECR 的服務連結角色許可

`AWSServiceRoleForECRPullThroughCache` 服務連結角色信任下列服務來擔任此角色。

- `pullthroughcache.ecr.amazonaws.com`

### 許可詳細資訊

此 `AWSECRPullThroughCache_ServiceRolePolicy` 許可政策連接至服務連結角色。此受管政策授予 Amazon ECR 執行下列動作的許可。如需詳細資訊，請參閱[AWSECRPullThroughCache\\_ServiceRolePolicy](#)。

- `ecr` – 允許 Amazon ECR 服務將映像提取並推送至私有儲存庫。
- `secretsmanager:GetSecretValue` – 允許 Amazon ECR 服務擷取 AWS Secrets Manager 秘密的加密內容。使用提取快取規則從需要在私有登錄檔中進行身分驗證的上游登錄檔快取映像時，需要此選項。該許可僅適用於具有 `ecr-pullthroughcache/` 名稱字首的秘密。

該 `AWSECRPullThroughCache_ServiceRolePolicy` 政策包含下列 JSON。

### JSON

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Sid": "ECR",
 "Effect": "Allow",
 "Action": [
 "ecr:GetAuthorizationToken",
 "ecr:BatchCheckLayerAvailability",
```

```

 "ecr:InitiateLayerUpload",
 "ecr:UploadLayerPart",
 "ecr:CompleteLayerUpload",
 "ecr:PutImage",
 "ecr:BatchGetImage",
 "ecr:BatchImportUpstreamImage",
 "ecr:GetDownloadUrlForLayer",
 "ecr:GetImageCopyStatus"
],
 "Resource": "*"
},
{
 "Sid": "SecretsManager",
 "Effect": "Allow",
 "Action": [
 "secretsmanager:GetSecretValue"
],
 "Resource": "arn:aws:secretsmanager:*:*:secret:ecr-pullthroughcache/
*",
 "Condition": {
 "StringEquals": {
 "aws:ResourceAccount": "${aws:PrincipalAccount}"
 }
 }
}
]
}

```

您必須設定許可，IAM 實體 (例如，使用者、群組或角色) 才可建立、編輯或刪除服務連結角色。如需詳細資訊，請參閱《IAM 使用者指南》中的[服務連結角色許可](#)。

### 建立 Amazon ECR 的服務連結角色

您不需要為提取快取手動建立 Amazon ECR 服務連結角色。當您在 AWS CLI、AWS 管理主控台或 AWS API 中建立私有登錄檔的提取快取規則時，Amazon ECR 會為您建立服務連結角色。

若您刪除此服務連結角色，之後需要再次建立，您可以在帳戶中使用相同程序重新建立角色。當您為私有登錄檔建立提取快取規則時，Amazon ECR 會再次為您建立服務連結角色 (如果尚不存在)。

## 編輯 Amazon ECR 的服務連結角色

Amazon ECR 不允許手動編輯 `AWSServiceRoleForECRPullThroughCache` 服務連結角色。因為可能有各種實體會參考服務連結角色，所以您無法在建立角色之後變更其名稱。然而，您可使用 IAM 來編輯角色描述。如需詳細資訊，請參閱《IAM 使用者指南》中的[編輯服務連結角色](#)。

## 刪除 Amazon ECR 的服務連結角色

若您不再使用需要服務連結角色的功能或服務，我們建議您刪除該角色。如此一來，您就沒有未主動監控或維護的未使用實體。不過，您必須在每個區域中先刪除登錄檔的提取快取規則，才能手動刪除服務連結角色。

### Note

如果您嘗試在 Amazon ECR 服務仍在角色時刪除資源，則刪除動作可能會失敗。若此情況發生，請等待數分鐘後並再次嘗試。

若要刪除 `AWSServiceRoleForECRPullThroughCache` 服務連結角色所使用的 Amazon ECR 資源

1. 在 <https://console.aws.amazon.com/ecr/> 開啟 Amazon ECR 主控台。
2. 從導覽列選擇您建立提取快取規則的區域。
3. 在導覽窗格中，選擇 Private registry (私有登錄檔)。
4. 在私有登錄檔頁面上，於提取快取組態區段中，選擇編輯。
5. 針對您建立的每個提取快取規則，請選取規則，然後選擇刪除規則。

## 使用 IAM 手動刪除服務連結角色

使用 IAM 主控台 AWS CLI、或 AWS API 來刪除 `AWSServiceRoleForECRPullThroughCache` 服務連結角色。如需詳細資訊，請參閱《IAM 使用者指南》中的[刪除服務連結角色](#)。

## 適用於儲存庫建立範本的 Amazon ECR 服務連結角色

Amazon ECR 使用名為 `AWSServiceRoleForECRTemplate` 的服務連結角色，授予 Amazon ECR 代表您執行動作以完成儲存庫建立範本動作的許可。

## Amazon ECR 的服務連結角色許可

`AWSServiceRoleForECRTemplate` 服務連結角色信任下列服務擔任該角色。

- `ecr.amazonaws.com`

## 許可詳細資訊

此 [ECRTemplateServiceRolePolicy](#) 許可政策連接至服務連結角色。此受管政策授予 Amazon ECR 代表您執行儲存庫建立動作的許可。

該 `ECRTemplateServiceRolePolicy` 政策包含下列 JSON。

## JSON

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Sid": "CreateRepositoryWithTemplate",
 "Effect": "Allow",
 "Action": [
 "ecr:CreateRepository"
],
 "Resource": "*"
 }
]
}
```

您必須設定許可，IAM 實體 (例如，使用者、群組或角色) 才可建立、編輯或刪除服務連結角色。如需詳細資訊，請參閱《IAM 使用者指南》中的[服務連結角色許可](#)。

## 建立 Amazon ECR 的服務連結角色

您不需要為儲存庫建立範本手動建立 Amazon ECR 服務連結角色。當您在 AWS 管理主控台、AWS CLI 或 AWS API 中為私有登錄檔建立儲存庫建立範本規則時，Amazon ECR 會為您建立服務連結角色。

若您刪除此服務連結角色，之後需要再次建立，您可以在帳戶中使用相同程序重新建立角色。當您為私有登錄檔建立儲存庫建立規則時，如果服務連結角色不存在，Amazon ECR 會再次為您建立該角色。

## 編輯 Amazon ECR 的服務連結角色

Amazon ECR 不允許手動編輯 `AWSServiceRoleForECRTemplate` 服務連結角色。因為可能有各種實體會參考服務連結角色，所以您無法在建立角色之後變更其名稱。然而，您可使用 IAM 來編輯角色描述。如需詳細資訊，請參閱《IAM 使用者指南》中的[編輯服務連結角色](#)。

## 刪除 Amazon ECR 的服務連結角色

若您不再使用需要服務連結角色的功能或服務，我們建議您刪除該角色。如此一來，您就沒有未主動監控或維護的未使用實體。不過，您必須先刪除每個區域中登錄檔的儲存庫建立規則，才能手動刪除服務連結角色。

### Note

如果您嘗試在 Amazon ECR 服務仍在使用角色時刪除資源，則刪除動作可能會失敗。若此情況發生，請等待數分鐘後並再次嘗試。

## 刪除 `AWSServiceRoleForECRTemplate` 服務連結角色所使用的 Amazon ECR 資源

1. 在 <https://console.aws.amazon.com/ecr/> 開啟 Amazon ECR 主控台。
2. 從導覽列中，選擇建立儲存庫建立規則的區域。
3. 在導覽窗格中，選擇 Private registry (私有登錄檔)。
4. 在私有登錄檔頁面的儲存庫建立範本區段中，選擇編輯。
5. 針對您建立的每個儲存庫建立規則，選取規則，然後選擇刪除規則。

## 使用 IAM 手動刪除服務連結角色

使用 IAM 主控台、AWS CLI、或 AWS API 來刪除 `AWSServiceRoleForECRTemplate` 服務連結角色。如需詳細資訊，請參閱《IAM 使用者指南》中的[刪除服務連結角色](#)。

## 針對 Amazon Elastic Container Registry Identity and Access 進行故障診斷

請使用以下資訊來協助您診斷和修正使用 Amazon ECR 和 IAM 時可能遇到的常見問題。

### 主題

- [我未獲授權，不得在 Amazon ECR 中執行動作](#)
- [我未獲得執行 `iam : PassRole` 的授權](#)

- [我想要允許以外的人員 AWS 帳戶 存取我的 Amazon ECR 資源](#)

## 我未獲授權，不得在 Amazon ECR 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 `ecr:GetWidget` 許可時發生。

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
ecr: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 `ecr: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，允許您將角色傳遞給 Amazon ECR。

有些 AWS 服務可讓您將現有角色傳遞給該服務，而不是建立新的服務角色或服務連結角色。如需執行此作業，您必須擁有將角色傳遞至該服務的許可。

當名為 marymajor 的 IAM 使用者嘗試使用主控台在 Amazon ECR 中執行動作時，發生下列範例錯誤。但是，動作請求服務具備服務角色授予的許可。Mary 沒有將角色傳遞給服務的許可。

```
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 Amazon ECR 資源

您可以建立一個角色，讓其他帳戶中的使用者或您組織外部的人員存取您的資源。您可以指定要允許哪些信任物件取得該角色。針對支援基於資源的政策或存取控制清單 (ACL) 的服務，您可以使用那些政策來授予人員存取您的資源的許可。

如需進一步了解，請參閱以下內容：

- 若要了解 Amazon ECR 是否支援這些功能，請參閱 [Amazon Elastic Container Registry 如何與 IAM 搭配使用](#)。
- 若要了解如何 AWS 帳戶 在您擁有的 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 IAM 使用者中提供存取權](#)。
- 若要了解如何將資源的存取權提供給第三方 AWS 帳戶，請參閱 [《IAM 使用者指南》中的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](#)。

## Amazon ECR 中的資料保護

[共同責任模型](#)適用於 AWS Amazon Elastic Container Service 中的資料保護。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詳細資訊，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 Amazon ECS 或使用主控台、API AWS CLI 或 AWS SDKs 的其他 AWS 服務 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。

## 主題

- [靜態加密](#)

## 靜態加密

### Important

使用 AWS KMS (DSSE-KMS) 的雙層伺服器端加密僅適用於 AWS GovCloud (US) 區域。

Amazon ECR 將映像存放在 Amazon ECR 管理的 Amazon S3 儲存貯體中。根據預設，Amazon ECR 會使用伺服器端加密與 Amazon S3 受管加密金鑰，該加密金鑰使用 AES-256 加密演算法對靜止資料進行加密。這不需要您採取任何動作，並且免費提供。如需詳細資訊，請參閱《Amazon Simple Storage Service 使用者指南》中的[透過 Amazon S3 受管加密金鑰 \(SSE-S3\) 使用伺服器端加密來保護資料](#)。

如需進一步控制 Amazon ECR 儲存庫的加密，您可以使用伺服器端加密搭配存放在 AWS Key Management Service () 中的 KMS 金鑰 AWS KMS。當您使用 AWS KMS 加密資料時，您可以使用由 Amazon ECR AWS 受管金鑰管理的預設值，或指定您自己的 KMS 金鑰（稱為客戶受管金鑰）。如需詳細資訊，請參閱《Amazon Simple Storage Service 使用者指南》中的[使用伺服器端加密搭配存放在 AWS KMS \(SSE-KMS\) 中的 KMS 金鑰來保護資料](#)。

您可以選擇使用雙層伺服器端加密搭配 AWS KMS (DSSE-KMS)，將兩層加密套用至 Amazon ECR 映像。DSSE-KMS 選項類似於 SSE-KMS，但會套用兩層個別加密，而不是一層加密。如需詳細資訊，請參閱[使用雙層伺服器端加密搭配 AWS KMS 金鑰 \(DSSE-KMS\)](#)。

每個 Amazon ECR 儲存庫都有一個加密組態，這是在建立儲存庫時所設定。您可以在每個儲存庫上使用不同的加密組態。如需詳細資訊，請參閱[建立 Amazon ECR 私有儲存庫以存放映像](#)。

在啟用 AWS KMS 加密的情況下建立儲存庫時，會使用 KMS 金鑰來加密儲存庫的內容。此外，Amazon ECR 會將 AWS KMS 授予新增至 KMS 金鑰，並將 Amazon ECR 儲存庫做為承授者委託人。

以下提供了對 Amazon ECR 如何與 AWS KMS 整合以加密和解密您的儲存庫的高等程度的了解：

1. 建立儲存庫時，Amazon ECR 會傳送 [DescribeKey](#) 呼叫給 AWS KMS，以驗證和擷取加密組態中指定的 KMS 金鑰的 Amazon Resource Name (ARN)。
2. Amazon ECR 將兩個 [CreateGrant](#) 請求傳送至 AWS KMS，以在 KMS 金鑰上建立授予，以允許 Amazon ECR 使用資料金鑰加密和解密資料。

3. 推送映像時，會向發出 [GenerateDataKey](#) 請求 AWS KMS，指定用於加密映像層和資訊清單的 KMS 金鑰。
4. AWS KMS 會產生新的資料金鑰、在指定的 KMS 金鑰下加密，並傳送加密的資料金鑰，以與映像層中繼資料和映像資訊清單一起存放。
5. 提取映像時，會對發出 [解密](#) 請求 AWS KMS，並指定加密的資料金鑰。
6. AWS KMS 會解密加密的資料金鑰，並將解密的資料金鑰傳送至 Amazon S3。
7. 使用資料金鑰在提取映像層之前解密映像層。
8. 刪除儲存庫時，Amazon ECR 會將兩個 [RetireGrant](#) 請求傳送至 AWS KMS，以淘汰為儲存庫建立的授予。

## 考量事項

搭配 Amazon ECR 使用 AWS KMS 型加密 (SSE-KMS 或 DSSE-KMS) 時，應考慮下列幾點。

- 如果您使用 KMS 加密建立 Amazon ECR 儲存庫，但未指定 KMS 金鑰，Amazon ECR `aws/ecr` 預設會使用 AWS 受管金鑰 具有別名的。當您第一次建立啟用 KMS 加密的儲存庫時，就會在您的帳戶中建立此 KMS 金鑰。
- 儲存庫加密組態無法在建立儲存庫之後變更。
- 當您使用 KMS 加密搭配您自己的 KMS 金鑰時，金鑰必須與您的儲存庫位於相同的區域。
- 不應撤銷 Amazon ECR 代表您建立的授予。如果您撤銷授予 Amazon ECR 許可以使用帳戶中的 AWS KMS 金鑰的授予，Amazon ECR 無法存取此資料、加密推送到儲存庫的新映像，或在提取時解密這些映像。當您撤銷 Amazon ECR 的授予時，則會立即進行變更。若要撤銷存取權，請刪除儲存庫，而不是撤銷授權。刪除儲存庫後，Amazon ECR 會代表您淘汰授予。
- 使用 AWS KMS 金鑰會產生相關費用。如需詳細資訊，請參閱 [AWS Key Management Service 定價](#)。
- 使用雙層伺服器端加密會產生相關費用。如需詳細資訊，請參閱 [Amazon ECR 定價](#)

## 所需的 IAM 許可

使用 AWS KMS 伺服器端加密建立或刪除 Amazon ECR 儲存庫時，所需的許可取決於您使用的特定 KMS 金鑰。

使用 AWS 受管金鑰 for Amazon ECR 時所需的 IAM 許可

根據預設，當 Amazon ECR 儲存庫啟用 AWS KMS 加密，但未指定 KMS 金鑰時，會使用適用於 Amazon ECR AWS 受管金鑰的。當 Amazon ECR 的 AWS 受管 KMS 金鑰用於加密儲存庫時，具有

建立儲存庫許可的任何主體也可以在儲存庫上啟用 AWS KMS 加密。但是，刪除儲存庫的 IAM 委託人必須具有 `kms:RetireGrant` 許可。這可讓您淘汰在建立儲存庫時新增至 AWS KMS 金鑰的授予。

下列範例 IAM 政策可以作為內嵌政策新增至使用者，以確保使用者具有刪除啟用加密的儲存庫所需的最低許可。使用資源參數可指定用於加密儲存庫的 KMS 金鑰。

## JSON

```
{
 "Version": "2012-10-17",
 "Id": "ecr-kms-permissions",
 "Statement": [
 {
 "Sid": "AllowAccessToRetireTheGrantsAssociatedWithTheKey",
 "Effect": "Allow",
 "Action": [
 "kms:RetireGrant"
],
 "Resource": "arn:aws:kms:us-
west-2:111122223333:key/b8d9ae76-080c-4043-92EXAMPLE"
 }
]
}
```

### 使用客戶受管金鑰時所需的 IAM 許可

使用客戶受管金鑰建立已啟用 AWS KMS 加密的儲存庫時，建立儲存庫的使用者或角色需要 KMS 金鑰政策和 IAM 政策的許可。

建立您自己的 KMS 金鑰時，您可以使用預設金鑰政策 AWS KMS 建立，或者您可以自行指定。為了確保客戶受管金鑰仍可由帳戶擁有者管理，KMS 金鑰的金鑰政策應允許帳戶根使用者的所有 AWS KMS 動作。其他範圍的權限可能會新增至金鑰政策，但至少應該授予根使用者管理 KMS 金鑰的許可。若要允許 KMS 金鑰僅用於源自 Amazon ECR 的請求，您可以使用 [kms:ViaService 條件索引鍵](#) 搭配 `ecr.<region>.amazonaws.com` 值。

下列範例金鑰政策可讓擁有 KMS 金鑰 AWS 的帳戶（根使用者）完整存取 KMS 金鑰。如需此範例金鑰政策的詳細資訊，請參閱《AWS Key Management Service 開發人員指南》中的 [允許存取 AWS 帳戶並啟用 IAM 政策](#)。

## JSON

```
{
 "Version": "2012-10-17",
 "Id": "ecr-key-policy",
 "Statement": [
 {
 "Sid": "EnableIAMUserPermissions",
 "Effect": "Allow",
 "Principal": {
 "AWS": "arn:aws:iam::111122223333:root"
 },
 "Action": "kms:*",
 "Resource": "*"
 }
]
}
```

除了必要的 Amazon ECR `kms:DescribeKey` 許可之外，建立儲存庫的 IAM 使用者 `kms:RetireGrant`、IAM 角色或 AWS 帳戶還必須具有 `kms:CreateGrant`、和 許可。

**Note**

必須將 `kms:RetireGrant` 許可新增到建立儲存庫的使用者或角色的 IAM 政策中。可以將 `kms:CreateGrant` 和 `kms:DescribeKey` 許可新增到 KMS 金鑰的金鑰政策或建立儲存庫的使用者或角色的 IAM 政策。如需 AWS KMS 許可運作方式的詳細資訊，請參閱《AWS Key Management Service 開發人員指南》中的 [AWS KMS API 許可：動作和資源參考](#)。

下列範例 IAM 政策可以作為內嵌政策新增至使用者，以確保使用者具有建立啟用加密的儲存庫所需的最低許可，並在使用完儲存庫後刪除該儲存庫。使用資源參數可指定用於加密儲存庫的 AWS KMS key。

## JSON

```
{
 "Version": "2012-10-17",
 "Id": "ecr-kms-permissions",
```

```

 "Statement": [
 {
 "Sid":
"AllowAccessToCreateAndRetireTheGrantsAssociatedWithTheKeyAsWellAsDescribeTheKey",
 "Effect": "Allow",
 "Action": [
 "kms:CreateGrant",
 "kms:RetireGrant",
 "kms:DescribeKey"
],
 "Resource": "arn:aws:kms:us-
west-2:111122223333:key/b8d9ae76-080c-4043-92EXAMPLE"
 }
]
 }

```

允許使用者在建立儲存庫時在主控台中列出 KMS 金鑰

使用 Amazon ECR 主控台建立儲存庫時，您可以授予許可以允許使用者在為儲存庫啟用加密時列出區域中的客戶受管 KMS 金鑰。下列 IAM 政策範例顯示使用主控台時列出 KMS 金鑰和別名所需的許可。

JSON

```

{
 "Version": "2012-10-17",
 "Statement": {
 "Effect": "Allow",
 "Action": [
 "kms:ListKeys",
 "kms:ListAliases",
 "kms:DescribeKey"
],
 "Resource": "*"
 }
}

```

## 監控 Amazon ECR 與 的互動 AWS KMS

您可以使用 AWS CloudTrail 來追蹤 Amazon ECR 代表您傳送給 AWS KMS 的請求。CloudTrail 日誌中的日誌項目包含加密內容索引鍵，可讓它們更容易識別。

## Amazon ECR 加密內容

加密內容是一組鍵/值組，其中包含任意非私密資料。當您在加密資料的請求中包含加密內容時，會以 AWS KMS 加密方式將加密內容繫結至加密的資料。若要解密資料，您必須傳遞相同的加密內容。

在其 [GenerateDataKey](#) 和 [Decrypt](#) 請求中 AWS KMS，Amazon ECR 使用兩個名稱/值對的加密內容，以識別正在使用的儲存庫和 Amazon S3 儲存貯體。如以下範例所示。名稱不會改變，但組合的加密內容值對於每個值都是不同的。

```
"encryptionContext": {
 "aws:s3:arn": "arn:aws:s3::us-west-2-starport-manifest-bucket/EXAMPLE1-90ab-cdef-fedc-ba987BUCKET1/sha256:a7766145a775d39e53a713c75b6fd6d318740e70327aaa3ed5d09e0ef33fc3df",
 "aws:ecr:arn": "arn:aws:ecr:us-west-2:111122223333:repository/repository-name"
}
```

您可以使用加密內容來識別稽核紀錄和日誌 (例如 [AWS CloudTrail](#) 和 Amazon CloudWatch Logs) 中的這些密碼編譯操作，以及在政策和授權中作為授權的條件。

Amazon ECR 加密內容包含兩個名稱值組。

- `aws:s3:arn` – 第一個名稱-值配對會識別儲存貯體。金鑰為 `aws:s3:arn`。值為 Amazon S3 儲存貯體 Amazon Resource Name (ARN)。

```
"aws:s3:arn": "ARN of an Amazon S3 bucket"
```

例如，如果儲存貯體的 ARN 是 `arn:aws:s3::us-west-2-starport-manifest-bucket/EXAMPLE1-90ab-cdef-fedc-ba987BUCKET1/sha256:a7766145a775d39e53a713c75b6fd6d318740e70327aaa3ed5d09e0ef33fc3df`，加密內容會包含下列對組。

```
"arn:aws:s3::us-west-2-starport-manifest-bucket/EXAMPLE1-90ab-cdef-fedc-ba987BUCKET1/sha256:a7766145a775d39e53a713c75b6fd6d318740e70327aaa3ed5d09e0ef33fc3df"
```

- `aws:ecr:arn` – 第二個名稱-值對會識別儲存庫的 Amazon Resource Name (ARN)。金鑰為 `aws:ecr:arn`。值為儲存庫的 ARN。

```
"aws:ecr:arn": "ARN of an Amazon ECR repository"
```

例如，如果儲存庫的 ARN 是 `arn:aws:ecr:us-west-2:111122223333:repository/repository-name`，加密內容會包含下列對組。

```
"aws:ecr:arn": "arn:aws:ecr:us-west-2:111122223333:repository/repository-name"
```

## 疑難排解

使用主控台刪除 Amazon ECR 儲存庫時，如果儲存庫已成功刪除，但 Amazon ECR 無法淘汰新增至您儲存庫的 KMS 金鑰的授予，則會收到以下錯誤訊息。

```
The repository [{repository-name}] has been deleted successfully but the grants created by the kmsKey [{kms_key}] failed to be retired
```

發生這種情況時，您可以自行淘汰儲存庫的 AWS KMS 授予。

### 手動淘汰儲存庫的 AWS KMS 授予

1. 列出用於儲存庫之 AWS KMS 金鑰的授予。此 `key-id` 值會包含在您從主控台收到的錯誤中。您也可以使用 `list-keys` 命令來列出帳戶中特定區域中的 AWS 受管金鑰 和客戶受管 KMS 金鑰。

```
aws kms list-grants \
 --key-id b8d9ae76-080c-4043-9237-c815bfc21dfc \
 --region us-west-2
```

輸出包含 `EncryptionContextSubset`，其中包含儲存庫的 Amazon Resource Name (ARN)。這可用於確定新增到金鑰的哪個授予是您要淘汰的授予。`GrantId` 值會在下一個步驟中淘汰授予時使用。

2. 淘汰為儲存庫新增之 AWS KMS 金鑰的每個授予。將 `GrantId` 的值替換為上一步輸出的授予 ID。

```
aws kms retire-grant \
 --key-id b8d9ae76-080c-4043-9237-c815bfc21dfc \
 --grant-id GrantId \
 --region us-west-2
```

## Amazon Elastic Container Registry 的合規驗證

若要了解 是否 AWS 服務 在特定合規計劃範圍內，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[下載報告 in 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](#)。

## Amazon Elastic Container Registry 的基礎設施安全性

Amazon Elastic Container Registry 是受管服務，受到 AWS 全球網路安全的保護。如需 AWS 安全服務以及 如何 AWS 保護基礎設施的相關資訊，請參閱[AWS 雲端安全](#)。若要使用基礎設施安全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 Amazon ECR。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。

您可從任何網路位置呼叫這些 API 操作，而 Amazon ECR 確實可支援基於資源的存取政策，以納入依據來源 IP 地址的限制。您也可以使用 Amazon ECR 政策，以便從特定 Amazon Virtual Private Cloud (Amazon VPC) 端點或特定 VPC 中控制存取權。實際上，這只會隔離網路中特定 VPC 對指定 Amazon ECR 資源 AWS 的網路存取。如需詳細資訊，請參閱[Amazon ECR 介面 VPC 端點 \(AWS PrivateLink\)](#)。

### Amazon ECR 介面 VPC 端點 (AWS PrivateLink)

您可以將 Amazon ECR 設定為使用介面 VPC 端點，進而提升 VPC 的安全狀態。VPC 端點採用 AWS PrivateLink，這項技術可讓您透過私有 IP 地址 (IPv4 和 IPv6) 私密存取 Amazon ECR APIs。AWS PrivateLink 會將 VPC 和 Amazon ECR 之間的所有網路流量限制在 Amazon 網路。您不需要網際網路閘道、NAT 裝置或虛擬私有閘道。

如需 AWS PrivateLink 和 VPC 端點的詳細資訊，請參閱《Amazon [VPC 使用者指南](#)》中的 [VPC 端點](#)。

## Amazon ECR VPC 端點的考量事項

在您設定 Amazon ECR 的 VPC 端點之前，請注意以下幾點考量。

- 若要允許 Amazon EC2 執行個體上託管的 Amazon ECS 任務從 Amazon ECR 提取私有映像，請建立 Amazon ECS 的介面 VPC 端點。如需詳細資訊，請參閱《Amazon Elastic Container Service 開發人員指南》中的[介面 VPC 端點 \(AWS PrivateLink\)](#)。
- Fargate 上託管的從 Amazon ECR 提取容器映像的 Amazon ECS 任務可以透過向任務的任務執行 IAM 角色新增條件索引鍵來限制對其任務使用的特定 VPC 和服務使用的 VPC 端點的存取。如需詳細資訊，請參閱《Amazon Elastic Container Service 開發人員指南》中的[透過介面端點提取 Amazon ECR 映像的 Fargate 任務的可選 IAM 許可](#)。
- 連接到 VPC 端點的安全群組，必須允許從 VPC 的私有子網路，透過 443 埠傳入的連線。
- Amazon ECR VPC 端點支援雙堆疊 (IPv4 和 IPv6) 連線。當您建立雙堆疊 VPC 端點時，它可以處理透過 IPv4 和 IPv6 私有 IP 地址的流量。
- VPC 端點透過美國東部 (維吉尼亞北部) 的 AWS API SDK 端點支援 Amazon ECR Public 儲存庫。
- VPC 端點僅支援透過 Amazon Route 53 AWS 提供的 DNS。如果您想要使用自己的 DNS，您可以使用條件式 DNS 轉送。如需詳細資訊，請參閱《Amazon VPC 使用者指南》中的[DHCP 選項集](#)。
- 如果您的容器有現有的 Amazon S3 連線，則在您新增 Amazon S3 閘道端點時，其連線可能會短暫中斷。如果您想避免發生中斷情況，請建立採用 Amazon S3 閘道端點的新 VPC，然後將 Amazon ECS 叢集及其容器遷移至新的 VPC。
- 第一次使用提取快取規則提取映像時，如果您已使用 AWS PrivateLink 將 Amazon ECR 設定為使用介面 VPC 端點，那麼您需要在同一個 VPC 中使用 NAT 閘道建立公有子網路，然後將所有傳出流量從其私有子網路路由到 NAT 閘道，以便提取至工作。隨後的映像提取不需要這樣做。如需詳細資訊，請參閱《Amazon Virtual Private Cloud 使用者指南》中的[案例：從私有子網路存取網際網路](#)。
- 對於需要 FIPS 140-3 驗證密碼編譯模組的工作負載，Amazon ECR 支援 VPC 端點的 FIPS 端點。

### Windows 映像的考量

基於 Windows 作業系統的映像包括受授權限制無法分配的成品。預設情況下，當您將 Windows 映像推送到 Amazon ECR 儲存庫時，不推送包含這些成品的層，因為它們被視為外來層。當成品由 Microsoft 提供時，外部層將從 Microsoft Azure 基礎設施中擷取。因此，為了讓您的容器能夠從 Azure 中提取這些外部層，除了建立 VPC 端點之外，還需要其他步驟。

當使用 Docker 常駐程式中的 `--allow-nondistributable-artifacts` 標誌將 Windows 映像推送到 Amazon ECR 時，可以覆寫此行為。啟用後，此標誌將把授權層推送到 Amazon ECR，這樣就可以透過 VPC 端點從 Amazon ECR 中提取這些映像，而無需額外存取 Azure。

### Important

使用 `--allow-nondistributable-artifacts` 標誌並不排除您遵守 Windows 容器基礎映像授權條款的義務；您不能發佈 Windows 內容以進行公有或第三方重新分佈。允許在您自己的環境中使用。

若要在 Docker 安裝中啟用此標誌，您必須修改 Docker 常駐程式組態檔案，其根據 Docker 的安裝情況，通常可以在 Docker Engine (Docker 引擎) 區段下的設定或偏好設定選單中設定，也可以直接編輯 `C:\ProgramData\docker\config\daemon.json` 檔案。

以下為所需組態的範例。將該值替換為要將映像推送到的儲存庫 URI。

```
{
 "allow-nondistributable-artifacts": [
 "111122223333.dkr.ecr.us-west-2.amazonaws.com"
]
}
```

修改 Docker 常駐程式組態檔案後，在嘗試推送映像之前，必須重新啟動 Docker 常駐程式。透過驗證基本層是否已推送到儲存庫來確認推送是否有效。

### Note

Windows 映像的基本圖很大。層大小將導致較長的推送時間以及 Amazon ECR 中這些映像的額外存放成本。基於這些原因，我們建議僅在嚴格要求減少建置時間和持續儲存成本時使用此選項。例如，`mcr.microsoft.com/windows/servercore` 映像 Amazon ECR 中壓縮時大約為 1.7 GiB。

## 為 Amazon ECR 建立 VPC 端點

若要為 Amazon ECR 服務建立 VPC 端點，請使用《Amazon VPC 使用者指南》中的 [Creating an Interface Endpoint](#) (建立介面端點) 程序。


Amazon ECR VPC 端點支援雙堆疊 (IPv4 和 IPv6) 連線。當您建立雙堆疊 VPC 端點時，它會自動處理透過 IPv4 和 IPv6 私有 IP 地址的流量。端點會根據用戶端的網路組態和端點的功能，使用適當的 IP 版本路由流量。

如果您有現有的 IPv4-only VPC 端點，並想要遷移至雙堆疊端點，您可以修改現有的端點以支援雙堆疊連線，或建立新的雙堆疊端點。建立或修改端點時，請確定您的 VPC 和子網路支援您要使用的 IP 版本。建立雙堆疊端點之後，端點將同時支援 IPv4 和 IPv6。

在 Amazon EC2 執行個體上託管的 Amazon ECS 任務需要 Amazon ECR 端點和 Amazon S3 閘道端點。

使用平台版本 1.4.0 或更高版本在 Fargate 上託管的 Amazon ECS 任務需要 Amazon ECR VPC 終端節點和 Amazon S3 閘道端點。

使用平台版本 1.3.0 或更早版本的 Fargate 上託管的 Amazon ECS 任務只需要 `com.amazonaws.region.ecr.dkr` Amazon ECR VPC 端點和 Amazon S3 閘道端點。

 Note

建立端點的順序並不重要。


`com.amazonaws.region.ecr.dkr`

此端點用於 Docker 登錄檔 API。Docker 用戶端命令 (例如 `push` 和 `pull`) 會使用此端點。

當您建立此端點時，需確實啟用私有 DNS 主機名稱。若要執行此操作，請務必在建立 VPC 端點時，選取 Amazon VPC 主控台中的 `Enable Private DNS Name` (啟用私有 DNS 名稱) 選項。

對於符合 FIPS 140-3 的連線，請使用 FIPS 端點名稱 `com.amazonaws.region.ecr-fips.dkr`

`com.amazonaws.region.ecr.api`

 Note

指定的 `##` 代表 Amazon ECR 支援的 AWS 區域的區域識別符，例如 `us-east-2` 美國東部 (俄亥俄) 區域。

對於符合 FIPS 140-3 的連線，請使用 FIPS 端點名稱：`com.amazonaws.region.ecr-fips.dkr` 和 `com.amazonaws.region.ecr-fips.api`。

此端點用於呼叫 Amazon ECR API。DescribeImages 和 CreateRepository 等 API 動作會移至此端點。

建立此端點時，您可以選擇啟用私有 DNS 主機名稱。當您建立 VPC 端點時，在 VPC 主控台中選取 Enable Private DNS Name (啟用私有 DNS 名稱) 來啟用此設計。如果您為 VPC 端點啟用私有 DNS 主機名稱，請將 SDK 或更新 AWS CLI 為最新版本，以便在使用 SDK 或 AWS CLI 時不需要指定端點 URL。

對於符合 FIPS 140-3 的連線，請使用 FIPS 端點名稱 `com.amazonaws.region.ecr-fips.api`。

如果您啟用私有 DNS 主機名稱，並使用 2019 年 1 月 24 日之前發行的 SDK 或 AWS CLI 版本，您必須使用 `--endpoint-url` 參數來指定介面端點。以下範例會顯示端點 URL 的格式。

```
aws ecr create-repository --repository-name name --endpoint-url https://
api.ecr.region.amazonaws.com
```

如果您不啟用 VPC 端點的私有 DNS 主機名稱，就必須透過 `--endpoint-url` 參數來指定介面端點的 VPC 端點 ID。以下範例會顯示端點 URL 的格式。

```
aws ecr create-repository --repository-name name --endpoint-url
https://VPC_endpoint_ID.api.ecr.region.vpce.amazonaws.com
```

對於 FIPS 140-3 相容連線，請使用 FIPS 端點 URL：

```
aws ecr create-repository --repository-name name --endpoint-url https://api.ecr-
fips.region.amazonaws.com
```

## 建立 Amazon S3 閘道端點

若要讓您的 Amazon ECS 任務從 Amazon ECR 提取私有映像，您必須為所有 Amazon S3 建立閘道端點。閘道端點是必要的，因為 Amazon ECR 使用 Amazon S3 來存放映像分層。容器從 Amazon ECR 下載映像時，必須存取 Amazon ECR 以取得映像資訊清單，並透過 Amazon S3 下載實際映像層。以下是 Amazon S3 儲存貯體的 Amazon Resource Name (ARN)，該儲存貯體包含每個 Docker 映像的分層。

```
arn:aws:s3:::prod-region-starport-layer-bucket/*
```

**Note**

如果為 Amazon ECR 建立雙堆疊 VPC 端點，則您也需要建立雙堆疊 Amazon S3 Gateway 或介面端點。如需詳細資訊，請參閱 [S3 文件](#)。

使用《Amazon VPC 使用者指南》中的 [Creating a gateway endpoint](#) (建立閘道端點) 操作程序為 Amazon ECR 建立以下 Amazon S3 閘道端點。建立端點時，請務必為您的 VPC 選取路由表。

com.amazonaws.*region*.s3

Amazon S3 閘道端點會使用 IAM 政策文件來限制服務的存取權限。您可以使用 Full Access (完整存取) 政策，因為系統會以此政策為基礎，繼續套用您對任務 IAM 角色或其他 IAM 使用者政策的限制。如果您想將 Amazon S3 儲存貯體存取權限限制為使用 Amazon ECR 所需的最低許可，請參閱 [Amazon ECR 的最低 Amazon S3 儲存貯體許可](#)。

Amazon ECR 的最低 Amazon S3 儲存貯體許可

Amazon S3 閘道端點會使用 IAM 政策文件來限制服務的存取權限。若要僅允許 Amazon ECR 的最低 Amazon S3 儲存貯體許可，請限制對 Amazon ECR 在為端點建立 IAM 政策文件時使用的 Amazon S3 儲存貯體的存取。

下表會說明 Amazon ECR 所需的 Amazon S3 儲存貯體政策許可。

| 權限                                                        | Description                                                                                          |
|-----------------------------------------------------------|------------------------------------------------------------------------------------------------------|
| arn:aws:s3:::prod- <i>region</i> -starport-layer-bucket/* | 提供對包含每個 Docker 映像分層之 Amazon S3 儲存貯體的存取權限。表示 Amazon ECR 支援的 AWS 區域的區域識別符，例如美國東部 (俄亥俄) 區域的 us-east-2 。 |

**範例**

下方範例會說明如何提供 Amazon ECR 操作所需的 Amazon S3 儲存貯體存取權限。

```
{
 "Statement": [
 {
```

```

 "Sid": "Access-to-specific-bucket-only",
 "Principal": "*",
 "Action": [
 "s3:GetObject"
],
 "Effect": "Allow",
 "Resource": ["arn:aws:s3:::prod-region-starport-layer-bucket/*"]
 }
]
}

```

## 建立 CloudWatch Logs 端點

使用 Fargate 啟動類型的 Amazon ECS 任務 (使用 VPC 而不使用網際網路閘道，也使用 awslogs 記錄驅動程序向 CloudWatch Logs 傳送記錄資訊) 需要為 CloudWatch Logs 建立 `com.amazonaws.region.logs` 介面 VPC 端點。如需詳細資訊，請參閱《Amazon CloudWatch Logs 使用者指南》中的 [CloudWatch Logs 搭配介面 VPC 端點一起使用](#)。

## 為您的 Amazon ECR VPC 端點建立端點政策

當您建立或修改端點時，VPC 端點政策是您連接至端點的 IAM 資源政策。如果您在建立端點時未連接政策，會為您 AWS 連接允許完整存取服務的預設政策。端點政策不會覆寫或取代使用者政策或服務特定的政策。這個另行區分的政策會控制從端點到所指定之服務的存取。端點政策必須以 JSON 格式撰寫。如需詳細資訊，請參閱《Amazon VPC 使用者指南》中的 [使用 VPC 端點控制服務的存取](#)。

我們建議建立單一 IAM 資源政策，並將其連接到兩個 Amazon ECR VPC 端點。

以下是 Amazon ECR 端點政策的範例。此政策可讓特定 IAM 角色從 Amazon ECR 提取映像。

```

{
 "Statement": [{
 "Sid": "AllowPull",
 "Principal": {
 "AWS": "arn:aws:iam::1234567890:role/role_name"
 },
 "Action": [
 "ecr:BatchGetImage",
 "ecr:GetDownloadUrlForLayer",
 "ecr:GetAuthorizationToken"
],
 "Effect": "Allow",
 }],
}

```

```
"Resource": "*"
}]
}
```

下列端點政策範例可防止指定的儲存庫遭到刪除。

```
{
 "Statement": [{
 "Sid": "AllowAll",
 "Principal": "*",
 "Action": "*",
 "Effect": "Allow",
 "Resource": "*"
 },
 {
 "Sid": "PreventDelete",
 "Principal": "*",
 "Action": "ecr:DeleteRepository",
 "Effect": "Deny",
 "Resource": "arn:aws:ecr:region:1234567890:repository/repository_name"
 }
]
}
```

下列端點政策範例會將前兩個範例結合成單一政策。

```
{
 "Statement": [{
 "Sid": "AllowAll",
 "Effect": "Allow",
 "Principal": "*",
 "Action": "*",
 "Resource": "*"
 },
 {
 "Sid": "PreventDelete",
 "Effect": "Deny",
 "Principal": "*",
 "Action": "ecr:DeleteRepository",
 "Resource": "arn:aws:ecr:region:1234567890:repository/repository_name"
 },
 {
 "Sid": "AllowPull",
```

```
"Effect": "Allow",
"Principal": {
 "AWS": "arn:aws:iam::1234567890:role/role_name"
},
"Action": [
 "ecr:BatchGetImage",
 "ecr:GetDownloadUrlForLayer",
 "ecr:GetAuthorizationToken"
],
"Resource": "*"
}
]
```

## 修改 Amazon ECR 的 VPC 端點政策

1. 在 <https://console.aws.amazon.com/vpc/> 開啟 Amazon VPC 主控台。
2. 在導覽窗格中選擇 Endpoints (端點)。
3. 如果您尚未建立 Amazon ECR 的 VPC 端點，請參閱 [為 Amazon ECR 建立 VPC 端點](#)。
4. 選取要新增政策的 Amazon ECR VPC 端點，然後選擇畫面下半部的 Policy (政策) 索引標籤。
5. 選擇 Edit Policy (編輯政策)，並對政策做出變更。
6. 選擇 Save (儲存) 以儲存政策。

## 共用子網路

無法在與您共用的子網路中建立、描述、修改或刪除 VPC 端點。不過，可以在與您共用的子網路中使用 VPC 端點。

## 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](#) 或 [aws:SourceAccount](#) 全域條件內容索引鍵，來限制 Amazon ECR 給予另一項服務對資源的許可。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，以及資源的完整 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:servicename:region: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。

`aws:SourceArn` 的值必須是 `ResourceDescription`。

下列範例顯示如何在 Amazon ECR 儲存庫政策中使用 `aws:SourceArn` 和 `aws:SourceAccount` 全域條件內容金鑰，以允許 AWS CodeBuild 存取與該服務整合所需的 Amazon ECR API 動作，同時防止混淆代理人問題。

## JSON

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Sid": "CodeBuildAccess",
 "Effect": "Allow",
 "Principal": {
 "Service": "codebuild.amazonaws.com"
 },
 "Action": [
 "ecr:BatchGetImage",
 "ecr:GetDownloadUrlForLayer"
],
 "Resource": "*",
 "Condition": {
 "ArnLike": {
 "aws:SourceArn": "arn:aws:codebuild:us-east-1:123456789012:project/project-name"
 },
 "StringEquals": {
 "aws:SourceAccount": "123456789012"
 }
 }
 }
]
}
```

```
}
]
}
```

# Amazon ECR 監控

您可以使用 Amazon CloudWatch 來監控 Amazon ECR API 用量，其會收集來自 Amazon ECR 的原始資料，並處理為可讀且近乎即時的指標。這些統計資料會記錄兩週，以便您可以存取歷史資訊並了解 API 使用情況。Amazon ECR 指標資料會自動以 1 分鐘的期間傳送到 CloudWatch。如需有關 CloudWatch 的詳細資訊，請參閱《Amazon CloudWatch 使用者指南》<https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/>。

Amazon ECR 會根據您的 API 用量，針對授權、映像推送和映像提取動作提供指標。

監控是維護 Amazon ECR 和 AWS 解決方案可靠性、可用性和效能的重要部分。我們建議您從組成 AWS 解決方案的資源收集監控資料，以便在發生多點故障時更輕鬆地偵錯。不過，開始監控 Amazon ECR 之前，您應該建立監控計劃，其中回答下列問題：

- 監控目標是什麼？
- 要監控哪些資源？
- 監控這些資源的頻率為何？
- 要使用哪些監控工具？
- 誰將執行監控任務？
- 發生問題時應該通知誰？

下一步是在各個時間點和不同的負載條件下測量效能，以在您的環境中確立 Amazon ECR 正常效能的基準。當您監控 Amazon ECR 時，請存放歷史記錄監控資料，如此才能與新的效能資料做比較、辨識正常效能模式和效能異常狀況、規劃問題處理方式。

## 主題

- [視覺化您的服務配額和設定警報](#)
- [Amazon ECR 用量指標](#)
- [Amazon ECR 用量報告](#)
- [Amazon ECR 儲存庫指標](#)
- [Amazon ECR 事件和 EventBridge](#)
- [使用記錄 Amazon ECR 動作 AWS CloudTrail](#)

## 視覺化您的服務配額和設定警報

您可以使用 CloudWatch 主控台視覺化您的服務配額，並查看目前用量與服務配額的比較情況。您也可以設定警示，以便在接近配額時收到通知。

視覺化服務配額並選擇是否設定警示

1. 透過 <https://console.aws.amazon.com/cloudwatch/> 開啟 CloudWatch 主控台。
2. 在導覽窗格中，選擇指標。
3. 在 All metrics (所有指標) 索引標籤上，選擇 Usage (用量)，然後選擇 By AWS Resource (依 AWS 資源)。

服務配額用量指標清單隨即出現。

4. 選取其中一個指標旁的核取方塊。

圖形會顯示您目前使用該 AWS 資源的情況。

5. 若要將服務配額新增至圖表，請執行下列動作：
  - a. 選擇 Graphed metrics (圖表化指標) 標籤。
  - b. 選擇 Math expression (數學表達式)、Start with an empty expression (以空表達式開始)。然後在新資料列的 Details (詳細資訊) 下，輸入 **SERVICE\_QUOTA(m1)**。

圖表中會新增一行，顯示指標所示資源的服務配額。

6. 若要查看目前用量的配額百分比，請新增表達式或變更目前的 SERVICE\_QUOTA 表達式。新的表達式請使用 **m1/60/SERVICE\_QUOTA(m1)\*100**。
7. (選用) 若要設定接近服務配額的警示通知，請執行下列動作：
  - a. 在 **m1/60/SERVICE\_QUOTA(m1)\*100** 資料列的 Actions (動作) 下，選擇警示圖示。它看起來像一個鈴鐺。

警示建立頁面隨即出現。

- b. 確定在 Conditions (條件) 下，Threshold type (閾值類型) 為 Static (靜態)，而 Whenever Expression1 is (每當 Expression1) 設為 Greater (大於)。在 than (比較值) 下輸入 **80**。當您的用量超過配額的 80% 時，即會建立進入 ALARM 狀態的警示。
- c. 選擇下一步。
- d. 在下一頁中，選取 Amazon SNS 主題或建立新主題。當警示進入 ALARM 狀態時，即會通知此主題。然後選擇下一步。

- e. 在下一頁中，輸入警示的名稱和說明，然後選擇 Next (下一步)。
- f. 選擇 Create alarm (建立警示)。

## Amazon ECR 用量指標

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

Amazon ECR 用量指標對應至 AWS 服務配額。您可以設定警示，在您的用量接近服務配額時發出警示。如需 Amazon ECR 服務配額的詳細資訊，請參閱 [Amazon ECR 服務配額](#)。

Amazon ECR 在 AWS/Usage 命名空間中發佈下列指標。

| 指標            | Description                                                                          |
|---------------|--------------------------------------------------------------------------------------|
| CallCount     | <p>從您帳戶呼叫 API 動作的次數。資源由與指標相關聯的維度定義。</p> <p>此指標最有用的統計資料為 SUM，代表所定義期間來自所有作者群之值的總和。</p> |
| ResourceCount | <p>您帳戶中指定的資源數量。資源由與指標相關聯的維度定義。</p> <p>此指標最有用的統計資料是 MAXIMUM，代表 5 分鐘期間使用的資源數目上限。</p>   |

下列維度用於精簡 Amazon ECR 發佈的 API 用量指標。

| 維度       | Description                                         |
|----------|-----------------------------------------------------|
| Service  | 包含資源 AWS 的服務名稱。對於 Amazon ECR 用量指標，此維度的值為 ECR。       |
| Type     | 正在報告的實體類型。目前，Amazon ECR API 用量指標的唯一有效值為 API。        |
| Resource | 正在執行的資源類型。目前，Amazon ECR 會針對下列 API 動作傳回 API 用量的相關資訊。 |

| 維度    | Description                                                                                                                                                                                                                                                                              |
|-------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|       | <ul style="list-style-type: none"> <li>• GetAuthorizationToken</li> <li>• BatchCheckLayerAvailability</li> <li>• InitiateLayerUpload</li> <li>• UploadLayerPart</li> <li>• CompleteLayerUpload</li> <li>• PutImage</li> <li>• BatchGetImage</li> <li>• GetDownloadUrlForLayer</li> </ul> |
| Class | 正在追蹤的資源類別。目前，Amazon ECR 不會使用類別維度。                                                                                                                                                                                                                                                        |

下列維度用於精簡 Amazon ECR 發佈的資源用量指標。

| 維度         | Description                                                                                                                                                |
|------------|------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Service    | 包含資源 AWS 的服務名稱。對於 Amazon ECR 用量指標，此維度的值為 ECR。                                                                                                              |
| Type       | 正在報告的實體類型。目前，Amazon ECR 資源用量指標的唯一有效值為 RESOURCE。                                                                                                            |
| Resource   | <p>正在執行的資源類型。目前，Amazon ECR 會針對下列指標傳回資源用量的相關資訊。</p> <ul style="list-style-type: none"> <li>• RepositoryCount</li> <li>• ImagesPerRepositoryCount</li> </ul> |
| ResourceId | 產生用量之資源的識別符。目前，ResourceId 僅與相關，ImagesPerRepositoryCount 其值格式為 repository/your_repository_name。例如："repository/my-repo" 會傳回儲存庫中名為 "my-repo" 的影像數量。           |

# Amazon ECR 用量報告

AWS 提供名為 Cost Explorer 的免費報告工具，可讓您分析 Amazon ECR 資源的成本和用量。

使用 Cost Explorer 來檢視用量和成本的圖表。您可以檢視前 13 個月的資料，以及預測未來三個月的可能花費。您可以使用 Cost Explorer 查看一段時間內在 AWS 資源上花費多少、找出需進一步調查的領域，以及查看您可用來了解成本的趨勢。您也可以指定資料的時間範圍，以及根據天或月檢視時間資料。

成本與用量報告中的計量資料會顯示所有 Amazon ECR 儲存庫的用量。如需詳細資訊，請參閱[標記您的資源以便計費](#)。

如需建立 AWS 成本和用量報告的詳細資訊，請參閱 AWS Billing 《使用者指南》中的[AWS 成本和用量報告](#)。

## Amazon ECR 儲存庫指標

Amazon ECR 向 Amazon CloudWatch 傳送儲存庫提取計數指標。Amazon ECR 指標資料會在 1 分鐘內自動傳送到 CloudWatch。如需有關 CloudWatch 的詳細資訊，請參閱[Amazon CloudWatch 使用者指南](#)。

### 主題

- [啟用 CloudWatch 指標](#)
- [可用的指標與維度](#)
- [使用 CloudWatch 主控台檢視 Amazon ECR 指標](#)

## 啟用 CloudWatch 指標

Amazon ECR 為所有儲存庫自動傳送儲存庫指標。無需採取任何手動步驟。

## 可用的指標與維度

以下區段列出 Amazon ECR 傳送至 Amazon CloudWatch 的指標和維度。

### Amazon ECR 指標

Amazon ECR 為您提供指標以監控儲存庫。您可以測量提取計數。

AWS/ECR 命名空間包含下列指標。

## RepositoryPullCount

儲存庫中映像的提取總數。

有效維度：RepositoryName。

有效統計資訊：Average、Minimum、Maximum、Sum、Sample Count。最實用的統計是 Sum。

單位：整數。

## Amazon ECR 指標的維度

Amazon ECR 指標使用 AWS/ECR 命名空間，並提供下列維度的指標。

### RepositoryName

此維度可篩選您為指定儲存庫中的所有容器映像請求的資料。

## 使用 CloudWatch 主控台檢視 Amazon ECR 指標

您可以在 CloudWatch 主控台上檢視 Amazon ECR 儲存庫指標。CloudWatch 主控台提供微調和可自訂的資源顯示。如需更多資訊，請參閱 [Amazon CloudWatch 使用者指南](#)。

## Amazon ECR 事件和 EventBridge

Amazon EventBridge 可讓您自動化您的 AWS 服務，並自動回應系統事件，例如應用程式可用性問題或資源變更。AWS 服務的事件會以接近即時的方式傳送到 EventBridge。您可撰寫簡單的規則，來指示您在意的事件，以及包含當事件符合規則時所要自動執行的動作。可以自動觸發的動作如下：

- 將事件新增至 CloudWatch Logs 中的日誌群組
- 叫用 AWS Lambda 函數
- 調用 Amazon EC2 執行命令
- 將事件轉傳至 Amazon Kinesis Data Streams
- 啟用 AWS Step Functions 狀態機器
- 通知 Amazon SNS 主題或 Amazon SQS 佇列

如需詳細資訊，請參閱《Amazon EventBridge 使用者指南》中的 [Amazon EventBridge 入門](#)。

## 來自 Amazon ECR 的範例事件

以下是 Amazon ECR 的範例事件。盡可能發出事件。

### 已完成映像推送的事件

每次完成映像推送時，會傳送下列事件。如需詳細資訊，請參閱[將 Docker 映像推送至 Amazon ECR 私有儲存庫](#)。

```
{
 "version": "0",
 "id": "13cde686-328b-6117-af20-0e5566167482",
 "detail-type": "ECR Image Action",
 "source": "aws.ecr",
 "account": "123456789012",
 "time": "2019-11-16T01:54:34Z",
 "region": "us-west-2",
 "resources": [],
 "detail": {
 "result": "SUCCESS",
 "repository-name": "my-repository-name",
 "image-digest":
 "sha256:7f5b2640fe6fb4f46592dfd3410c4a79dac4f89e4782432e0378abcd1234",
 "action-type": "PUSH",
 "image-tag": "latest"
 }
}
```

### 提取快取動作的事件

嘗試提取快取動作時，會傳送下列事件。如需詳細資訊，請參閱[將上游登錄檔與 Amazon ECR 私有登錄檔同步](#)。

```
{
 "version": "0",
 "id": "85fc3613-e913-7fc4-a80c-a3753e4aa9ae",
 "detail-type": "ECR Pull Through Cache Action",
 "source": "aws.ecr",
 "account": "123456789012",
 "time": "2023-02-29T02:36:48Z",
 "region": "us-west-2",
 "resources": [
 "arn:aws:ecr:us-west-2:123456789012:repository/docker-hub/alpine"
]
}
```

```

],
 "detail": {
 "rule-version": "1",
 "sync-status": "SUCCESS",
 "ecr-repository-prefix": "docker-hub",
 "repository-name": "docker-hub/alpine",
 "upstream-registry-url": "public.ecr.aws",
 "image-tag": "3.17.2",
 "image-digest":
 "sha256:4aa08ef415aecc80814cb42fa41b658480779d80c77ab15EXAMPLE",
 }
}

```

### 已完成映像掃描的事件 (基本型掃描)

啟用登錄檔的基本型掃描時，當每個映像掃描完成時，就會傳送下列事件。finding-severity-counts 參數只會在出現嚴重性等級時才傳回嚴重性等級的值。例如，如果映像在 CRITICAL 等級沒有問題清單，則不會傳回任何重要等級數值。如需詳細資訊，請參閱[掃描映像是否有 Amazon ECR 中的作業系統漏洞](#)。

#### Note

如需有關 Amazon Inspector 在啟用增強型掃描時發出之事件的詳細資訊，請參閱 [在 Amazon ECR 中傳送用於增強型掃描的 EventBridge 事件](#)。

```

{
 "version": "0",
 "id": "85fc3613-e913-7fc4-a80c-a3753e4aa9ae",
 "detail-type": "ECR Image Scan",
 "source": "aws.ecr",
 "account": "123456789012",
 "time": "2019-10-29T02:36:48Z",
 "region": "us-east-1",
 "resources": [
 "arn:aws:ecr:us-east-1:123456789012:repository/my-repository-name"
],
 "detail": {
 "scan-status": "COMPLETE",
 "repository-name": "my-repository-name",
 "finding-severity-counts": {
 "CRITICAL": 10,

```

```

 "MEDIUM": 9
 },
 "image-digest":
 "sha256:7f5b2640fe6fb4f46592dfd3410c4a79dac4f89e4782432e0378abcd1234",
 "image-tags": []
}
}

```

### 啟用增強型掃描之資源上的變更通知事件 (增強掃描)

為登錄檔啟用增強型掃描時，Amazon ECR 會在啟用了增強型掃描的資源發生變更時傳送下列事件。這包括正在建立的新儲存庫、正在變更的儲存庫掃描頻率，或是在啟用了增強型掃描的儲存庫中建立或刪除映像時。如需詳細資訊，請參閱[掃描映像是否有 Amazon ECR 中的軟體漏洞](#)。

```

{
 "version": "0",
 "id": "0c18352a-a4d4-6853-ef53-0ab8638973bf",
 "detail-type": "ECR Scan Resource Change",
 "source": "aws.ecr",
 "account": "123456789012",
 "time": "2021-10-14T20:53:46Z",
 "region": "us-east-1",
 "resources": [],
 "detail": {
 "action-type": "SCAN_FREQUENCY_CHANGE",
 "repositories": [{
 "repository-name": "repository-1",
 "repository-arn": "arn:aws:ecr:us-east-1:123456789012:repository/repository-1",
 "scan-frequency": "SCAN_ON_PUSH",
 "previous-scan-frequency": "MANUAL"
 },
 {
 "repository-name": "repository-2",
 "repository-arn": "arn:aws:ecr:us-east-1:123456789012:repository/repository-2",
 "scan-frequency": "CONTINUOUS_SCAN",
 "previous-scan-frequency": "SCAN_ON_PUSH"
 },
 {
 "repository-name": "repository-3",
 "repository-arn": "arn:aws:ecr:us-east-1:123456789012:repository/repository-3",
 "scan-frequency": "CONTINUOUS_SCAN",
 "previous-scan-frequency": "SCAN_ON_PUSH"
 }
]
}

```

```
],
 "resource-type": "REPOSITORY",
 "scan-type": "ENHANCED"
}
}
```

## 映像刪除的事件

刪除映像時傳送以下事件。如需詳細資訊，請參閱[在 Amazon ECR 中刪除映像](#)。

```
{
 "version": "0",
 "id": "dd3b46cb-2c74-f49e-393b-28286b67279d",
 "detail-type": "ECR Image Action",
 "source": "aws.ecr",
 "account": "123456789012",
 "time": "2019-11-16T02:01:05Z",
 "region": "us-west-2",
 "resources": [],
 "detail": {
 "result": "SUCCESS",
 "repository-name": "my-repository-name",
 "image-digest":
 "sha256:7f5b2640fe6fb4f46592dfd3410c4a79dac4f89e4782432e0378abcd1234",
 "action-type": "DELETE",
 "image-tag": "latest"
 }
}
```

## 映像封存動作的事件

封存映像時，會傳送下列事件。target-storage-class 欄位將設定為 ARCHIVE。事件包含資訊清單和成品媒體類型，以識別要封存的內容類型。

```
{
 "version": "0",
 "id": "4f5ec4d5-4de4-7aad-a046-EXAMPLE",
 "detail-type": "ECR Image Action",
 "source": "aws.ecr",
 "account": "123456789012",
 "time": "2019-08-06T00:58:09Z",
 "region": "us-east-1",
 "resources": [],
```

```

 "detail": {
 "action-type": "UPDATE_STORAGE_CLASS",
 "target-storage-class": "ARCHIVE",
 "image-digest":
"sha256:f98d67af8e53a536502bfc600de3266556b06ed635a32d60aa7a5fe6d7e609d7",
 "repository-name": "ubuntu",
 "result": "SUCCESS",
 "manifest-media-type": "application/vnd.oci.image.manifest.v1+json",
 "artifact-media-type": "application/vnd.oci.image.config.v1+json"
 }
 }
}

```

### 映像還原動作的事件

還原封存映像時，會傳送下列事件。target-storage-class 欄位將設定為 STANDARD。事件包含顯示影像上次還原時間last-activated-at的欄位。

```

{
 "version": "0",
 "id": "7b8fc5e6-5ef5-8bbe-b157-EXAMPLE",
 "detail-type": "ECR Image Action",
 "source": "aws.ecr",
 "account": "123456789012",
 "time": "2019-08-06T01:15:22Z",
 "region": "us-east-1",
 "resources": [],
 "detail": {
 "action-type": "UPDATE_STORAGE_CLASS",
 "target-storage-class": "STANDARD",
 "image-digest":
"sha256:f98d67af8e53a536502bfc600de3266556b06ed635a32d60aa7a5fe6d7e609d7",
 "repository-name": "ubuntu",
 "result": "SUCCESS",
 "manifest-media-type": "application/vnd.oci.image.manifest.v1+json",
 "artifact-media-type": "application/vnd.oci.image.config.v1+json",
 "last-activated-at": "2025-10-10T19:13:02.74Z"
 }
}

```

### 推薦者還原動作的事件

還原封存的推薦者（參考成品，例如 SBOM、簽章或證明）時，會傳送下列事件。請注意，detail-type ECR Referrer Action 是將其與一般映像動作區分開來。manifest-media-type

和 `artifact-media-type` 欄位可識別要還原的特定參考者類型。在此範例中，正在還原 SBOM 成品。

```
{
 "version": "0",
 "id": "8c9gd6f7-6fg6-9ccf-c268-EXAMPLE",
 "detail-type": "ECR Referrer Action",
 "source": "aws.ecr",
 "account": "123456789012",
 "time": "2019-08-06T01:20:45Z",
 "region": "us-east-1",
 "resources": [],
 "detail": {
 "action-type": "UPDATE_STORAGE_CLASS",
 "target-storage-class": "STANDARD",
 "image-digest":
 "sha256:f98d67af8e53a536502bfc600de3266556b06ed635a32d60aa7a5fe6d7e609d7",
 "repository-name": "sbom",
 "result": "SUCCESS",
 "manifest-media-type": "application/vnd.cncf.oras.artifact.manifest.v1+json",
 "artifact-media-type": "text/sbom+json",
 "last-activated-at": "2025-10-10T19:13:02.74Z"
 }
}
```

已完成映像複寫的事件

下列事件會在每個映像複寫完成時傳送。如需詳細資訊，請參閱[Amazon ECR 中的私有映像複寫](#)。

```
{
 "version": "0",
 "id": "c8b133b1-6029-ee73-e2a1-4f466b8ba999",
 "detail-type": "ECR Replication Action",
 "source": "aws.ecr",
 "account": "123456789012",
 "time": "2024-05-08T20:44:54Z",
 "region": "us-east-1",
 "resources": [
 "arn:aws:ecr:us-east-1:123456789012:repository/docker-hub/alpine"
],
 "detail": {
 "result": "SUCCESS",
 "repository-name": "docker-hub/alpine",
 }
}
```

```

 "image-digest":
 "sha256:7f5b2640fe6fb4f46592dfd3410c4a79dac4f89e4782432e0378abcd1234",
 "source-account": "123456789012",
 "action-type": "REPLICATE",
 "source-region": "us-west-2",
 "image-tag": "3.17.2"
 }
}

```

## 失敗映像複寫的事件

當映像複寫失敗時，會傳送下列事件。result 欄位將包含 FAILED，而額外的錯誤資訊可能會包含在事件詳細資訊中。

```

{
 "version": "0",
 "id": "d9c244c2-7130-ff84-f3b2-5g577c9cb000",
 "detail-type": "ECR Replication Action",
 "source": "aws.ecr",
 "account": "123456789012",
 "time": "2024-05-08T20:45:12Z",
 "region": "us-east-1",
 "resources": [
 "arn:aws:ecr:us-east-1:123456789012:repository/my-app"
],
 "detail": {
 "result": "FAILED",
 "repository-name": "my-app",
 "image-digest":
 "sha256:8g6c3751gf7gc5g47603ege4511d5a80ead5g90f5893543f1489bde2345",
 "source-account": "123456789012",
 "action-type": "REPLICATE",
 "source-region": "us-west-2",
 "image-tag": "latest"
 }
}

```

## 使用記錄 Amazon ECR 動作 AWS CloudTrail

Amazon ECR 已與服務整合 AWS CloudTrail，此服務提供使用者、角色或 AWS 服務在 Amazon ECR 中所採取動作的記錄。CloudTrail 會將下列 Amazon ECR 動作擷取為事件：

- 所有 API 呼叫，包括來自 Amazon ECR 主控台的呼叫
- 由於儲存庫上的加密設定而採取的所有動作
- 根據生命週期政策規則採取的所有動作，包括成功和失敗的動作

#### Important

由於個別 CloudTrail 事件的大小限制，對於有 10 個或更多映像過期的生命週期政策動作，Amazon ECR 會向 CloudTrail 發送多個事件。此外，Amazon ECR 每個映像最多包含 100 個標籤。

建立追蹤後，便可將 CloudTrail 事件持續交付至 Amazon S3 儲存貯體，包括 Amazon ECR 的事件。即使您未設定追蹤，依然可以透過 CloudTrail 主控台的事件歷史記錄檢視最新事件。使用此資訊，您就可以判斷傳送至 Amazon ECR 的請求、提出請求的 IP 地址、提出請求的對象、提出請求的時間，以及其他詳細資訊。

如需詳細資訊，請參閱《AWS CloudTrail 使用者指南》<https://docs.aws.amazon.com/awscloudtrail/latest/userguide/>。

## CloudTrail 中的 Amazon ECR 資訊

當您建立 AWS 帳戶時，會在您的帳戶上啟用 CloudTrail。在 Amazon ECR 中發生活動時，該活動將與 Event history (事件歷史紀錄) 中的其他 AWS 服務事件一起記錄在 CloudTrail 事件中。您可以在 AWS 帳戶中檢視、搜尋和下載最近的事件。如需詳細資訊，請參閱《使用 CloudTrail 事件歷史記錄檢視事件》<https://docs.aws.amazon.com/awscloudtrail/latest/userguide/view-cloudtrail-events.html>。

若要持續記錄您 AWS 帳戶中的事件，包括 Amazon ECR 的事件，請建立追蹤。線索能讓 CloudTrail 將日誌檔案交付至 Amazon S3 儲存貯體。當您在主控台建立追蹤記錄時，可以將追蹤記錄套用至單一區域或所有區域。線索會記錄 AWS 分割區中的事件，並將日誌檔案交付至您指定的 Amazon S3 儲存貯體。此外，您可以設定其他 AWS 服務，以分析和處理 CloudTrail 日誌中所收集的事件資料。如需詳細資訊，請參閱：

- [為 AWS 您的帳戶建立追蹤](#)
- [AWS 服務與 CloudTrail 日誌的整合](#)
- [設定 CloudTrail 的 Amazon SNS 通知](#)
- [接收多個區域的 CloudTrail 日誌檔案及接收多個帳戶的 CloudTrail 日誌檔案](#)

CloudTrail 會記錄所有 Amazon ECR API 動作，並記錄在 [Amazon Elastic Container Registry API 參考](#) 中。執行常見任務時，CloudTrail 日誌檔案中會為屬於該任務的每個 API 動作產生不同區段。例如，建立儲存庫時，會在 CloudTrail 日誌檔案中產生 `GetAuthorizationToken`、`CreateRepository` 及 `SetRepositoryPolicy` 區段。當您將映像推送到儲存庫、`InitiateLayerUpload`、`PutImage`、和 `UploadLayerPart` `CompleteLayerUpload` 時，如果啟用 Blob 掛載，就會產生 `MountLayer` 區段。提取映像時，會產生 `GetDownloadUrlForLayer` 和 `BatchGetImage` 區段。當您封存或還原時，會產生映像 `UpdateImageStorageClass` 區段。當支援 OCI 1.1 規格的 OCI 用戶端使用 `Referrers` API 擷取映像的參考者清單或參考成品時，會發出 `ListImageReferrers` CloudTrail 事件。如需這些常見任務的範例，請參閱 [CloudTrail 日誌項目範例](#)。

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：

- 該請求是否使用根或使用者登入資料提出
- 提出該請求時，是否使用了特定角色或聯合身分使用者的臨時安全憑證
- 請求是否由其他 AWS 服務提出

如需詳細資訊，請參閱 [CloudTrail userIdentity 元素](#)。

## 了解 Amazon ECR 日誌檔案項目

追蹤是一種組態，能讓事件以日誌檔案的形式交付到您指定的 Amazon S3 儲存貯體。CloudTrail 日誌檔案包含一或多個日誌專案。一個事件為任何來源提出的單一請求，並包含請求動作、請求的日期和時間、請求參數等其他資訊。CloudTrail 日誌檔案並非依公有 API 呼叫追蹤記錄的堆疊排序，因此不會以任何特定順序出現。

### CloudTrail 日誌項目範例

以下是一些常見 Amazon ECR 任務的 CloudTrail 日誌項目範例。

這些範例已格式化，以提升可讀性。在 CloudTrail 日誌檔案中，所有項目和事件會合併為單一系列。此外，這個範例中受限於單一 Amazon ECR 項目。在真正的 CloudTrail 日誌檔案中，您會看到來自多個 AWS 服務的項目和事件。

**⚠ Important**

sourceIPAddress 是提出請求的 IP 地址。對於源自 服務主控台的動作，報告的地址適用於您的基礎資源，而不是主控台 Web 伺服器。對於 中的服務 AWS，只會顯示 DNS 名稱。即使已修訂為 AWS 服務 DNS 名稱，我們仍會使用用戶端來源 IP 來評估身分驗證。

**主題**

- [範例：建立儲存庫動作](#)
- [範例：AWS KMS CreateGrant 建立 Amazon ECR 儲存庫時的 API 動作](#)
- [範例：映像推送動作](#)
- [範例：映像提取動作](#)
- [範例：映像生命週期政策動作](#)
- [範例：映像封存動作](#)
- [範例：映像還原動作](#)
- [範例：映像參考程式動作](#)

**範例：建立儲存庫動作**

以下範例顯示的是展示 CreateRepository 動作的 CloudTrail 日誌項目。

```
{
 "eventVersion": "1.04",
 "userIdentity": {
 "type": "AssumedRole",
 "principalId": "AIDACKCEVSQ6C2EXAMPLE:account_name",
 "arn": "arn:aws:sts::123456789012:user/Mary_Major",
 "accountId": "123456789012",
 "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
 "sessionContext": {
 "attributes": {
 "mfaAuthenticated": "false",
 "creationDate": "2018-07-11T21:54:07Z"
 },
 "sessionIssuer": {
 "type": "Role",
 "principalId": "AIDACKCEVSQ6C2EXAMPLE",
 "arn": "arn:aws:iam::123456789012:role/Admin",
```

```

 "accountId": "123456789012",
 "userName": "Admin"
 }
},
"eventTime": "2018-07-11T22:17:43Z",
"eventSource": "ecr.amazonaws.com",
"eventName": "CreateRepository",
"awsRegion": "us-east-2",
"sourceIPAddress": "203.0.113.12",
"userAgent": "console.amazonaws.com",
"requestParameters": {
 "repositoryName": "testrepo"
},
"responseElements": {
 "repository": {
 "repositoryArn": "arn:aws:ecr:us-east-2:123456789012:repository/testrepo",
 "repositoryName": "testrepo",
 "repositoryUri": "123456789012.dkr.ecr.us-east-2.amazonaws.com/testrepo",
 "createdAt": "Jul 11, 2018 10:17:44 PM",
 "registryId": "123456789012"
 }
},
"requestID": "cb8c167e-EXAMPLE",
"eventID": "e3c6f4ce-EXAMPLE",
"resources": [
 {
 "ARN": "arn:aws:ecr:us-east-2:123456789012:repository/testrepo",
 "accountId": "123456789012"
 }
],
"eventType": "AwsApiCall",
"recipientAccountId": "123456789012"
}

```

### 範例：AWS KMS **CreateGrant** 建立 Amazon ECR 儲存庫時的 API 動作

下列範例顯示 CloudTrail 日誌項目，在建立已啟用 KMS 加密的 Amazon ECR 儲存庫時示範 AWS KMS **CreateGrant** 動作。對於使用 KMS 加密建立的每個儲存庫都已啟用，您應該會在 CloudTrail 中看到兩個 **CreateGrant** 日誌項目。

```

{
 "eventVersion": "1.05",

```

```
"userIdentity": {
 "type": "IAMUser",
 "principalId": "AIDAIEP6W46J43IG7LXAQ",
 "arn": "arn:aws:iam::123456789012:user/Mary_Major",
 "accountId": "123456789012",
 "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
 "userName": "Mary_Major",
 "sessionContext": {
 "sessionIssuer": {

 },
 "webIdFederationData": {

 },
 "attributes": {
 "mfaAuthenticated": "false",
 "creationDate": "2020-06-10T19:22:10Z"
 }
 },
 "invokedBy": "AWS Internal"
},
"eventTime": "2020-06-10T19:22:10Z",
"eventSource": "kms.amazonaws.com",
"eventName": "CreateGrant",
"awsRegion": "us-west-2",
"sourceIPAddress": "203.0.113.12",
"userAgent": "console.amazonaws.com",
"requestParameters": {
 "keyId": "4b55e5bf-39c8-41ad-b589-18464af7758a",
 "granteePrincipal": "ecr.us-west-2.amazonaws.com",
 "operations": [
 "GenerateDataKey",
 "Decrypt"
],
 "retiringPrincipal": "ecr.us-west-2.amazonaws.com",
 "constraints": {
 "encryptionContextSubset": {
 "aws:ecr:arn": "arn:aws:ecr:us-west-2:123456789012:repository/testrepo"
 }
 }
},
"responseElements": {
 "grantId": "3636af9adfee1accb67b83941087dcd45e7fadc4e74ff0103bb338422b5055f3"
},
```

```

"requestID": "047b7dea-b56b-4013-87e9-a089f0f6602b",
"eventID": "af4c9573-c56a-4886-baca-a77526544469",
"readOnly": false,
"resources": [
 {
 "accountId": "123456789012",
 "type": "AWS::KMS::Key",
 "ARN": "arn:aws:kms:us-west-2:123456789012:key/4b55e5bf-39c8-41ad-
b589-18464af7758a"
 }
],
"eventType": "AwsApiCall",
"recipientAccountId": "123456789012"
}

```

### 範例：映像推送動作

下列範例顯示 CloudTrail 日誌項目，此項目示範了使用 PutImage 動作的映像推送。

#### Note

推送映像時，您也會在 CloudTrail 日誌中看到 `InitiateLayerUpload`、`UploadLayerPart` 和 `CompleteLayerUpload` 參考。

```

{
 "eventVersion": "1.04",
 "userIdentity": {
 "type": "IAMUser",
 "principalId": "AIDACKCEVSQ6C2EXAMPLE:account_name",
 "arn": "arn:aws:sts::123456789012:user/Mary_Major",
 "accountId": "123456789012",
 "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
 "userName": "Mary_Major",
 "sessionContext": {
 "attributes": {
 "mfaAuthenticated": "false",
 "creationDate": "2019-04-15T16:42:14Z"
 }
 }
 },
 "eventTime": "2019-04-15T16:45:00Z",

```

```

"eventSource": "ecr.amazonaws.com",
"eventName": "PutImage",
"awsRegion": "us-east-2",
"sourceIPAddress": "AWS Internal",
"userAgent": "AWS Internal",
"requestParameters": {
 "repositoryName": "testrepo",
 "imageTag": "latest",
 "registryId": "123456789012",
 "imageManifest": "{\n \"schemaVersion\": 2,\n \"mediaType\": \"application/
vnd.docker.distribution.manifest.v2+json\",\n \"config\": {\n \"mediaType\":
\"application/vnd.docker.container.image.v1+json\",\n \"size\": 5543,\n
 \"digest\": \"sha256:000b9b805af1cdb60628898c9f411996301a1c13afd3dbef1d8a16ac6dbf503a
\\n },\n \"layers\": [\n {\n \"mediaType\": \"application/
vnd.docker.image.rootfs.diff.tar.gzip\",\n \"size\": 43252507,\n
 \"digest\": \"sha256:3b37166ec61459e76e33282dda08f2a9cd698ca7e3d6bc44e6a6e7580cdeff8e
\\n },\n {\n \"mediaType\": \"application/
vnd.docker.image.rootfs.diff.tar.gzip\",\n \"size\": 846,\n \"digest
\": \"sha256:504facff238fde83f1ca8f9f54520b4219c5b8f80be9616ddc52d31448a044bd
\\n },\n {\n \"mediaType\": \"application/
vnd.docker.image.rootfs.diff.tar.gzip\",\n \"size\": 615,\n \"digest
\": \"sha256:ebbcacd28e101968415b0c812b2d2dc60f969e36b0b08c073bf796e12b1bb449\"\\n
 },\n {\n \"mediaType\": \"application/
vnd.docker.image.rootfs.diff.tar.gzip\",\n \"size\": 850,\n \"digest
\": \"sha256:c7fb3351ecad291a88b92b600037e2435c84a347683d540042086fe72c902b8a
\\n },\n {\n \"mediaType\": \"application/
vnd.docker.image.rootfs.diff.tar.gzip\",\n \"size\": 168,\n \"digest\":
\"sha256:2e3debadcbf7e542e2aefbce1b64a358b1931fb403b3e4aeca27cb4d809d56c2\"\\n },
\n {\n \"mediaType\": \"application/vnd.docker.image.rootfs.diff.tar.gzip
\",,\n \"size\": 37720774,\n \"digest\":
\"sha256:f8c9f51ad524d8ae9bf4db69cd3e720ba92373ec265f5c390ffb21bb0c277941\"\\n
 },\n {\n \"mediaType\": \"application/
vnd.docker.image.rootfs.diff.tar.gzip\",\n \"size\": 30432107,\n
 \"digest\": \"sha256:813a50b13f61cf1f8d25f19fa96ad3aa5b552896c83e86ce413b48b091d7f01b
\\n },\n {\n \"mediaType\": \"application/
vnd.docker.image.rootfs.diff.tar.gzip\",\n \"size\": 197,\n \"digest
\": \"sha256:7ab043301a6187ea3293d80b30ba06c7bf1a0c3cd4c43d10353b31bc0cecfe7d
\\n },\n {\n \"mediaType\": \"application/
vnd.docker.image.rootfs.diff.tar.gzip\",\n \"size\": 154,\n \"digest
\": \"sha256:67012cca8f31dc3b8ee2305e7762fee20c250513effdedb38a1c37784a5a2e71\"\\n
 },\n {\n \"mediaType\": \"application/
vnd.docker.image.rootfs.diff.tar.gzip\",\n \"size\": 176,\n \"digest
\": \"sha256:3bc892145603fffc9b1c97c94e2985b4cb19ca508750b15845a5d97becbd1a0e
\\n },\n {\n \"mediaType\": \"application/

```

```

vnd.docker.image.rootfs.diff.tar.gzip",\n \size": 183,\n \digest
\": \sha256:6f1c79518f18251d35977e7e46bfa6c6b9cf50df2a79d4194941d95c54258d18\"\n
 },\n {\n \mediaType\": \application/
vnd.docker.image.rootfs.diff.tar.gzip",\n \size\": 212,\n \digest
\": \sha256:b7bcfbc2e2888afebede4dd1cd5eebf029bb6315feeaf0b56e425e11a50afe42\"\n
 },\n {\n \mediaType\": \application/
vnd.docker.image.rootfs.diff.tar.gzip",\n \size\": 212,\n \digest\":
\sha256:2b220f8b0f32b7c2ed8eaafe1c802633bbd94849b9ab73926f0ba46cdae91629\"\n }\n
]\n}
},
"responseElements": {
 "image": {
 "repositoryName": "testrepo",
 "imageManifest": "{\n \schemaVersion\": 2,\n \mediaType\": \application/
vnd.docker.distribution.manifest.v2+json",\n \config\": {\n \mediaType\":
\application/vnd.docker.container.image.v1+json",\n \size\": 5543,\n
\digest\": \sha256:000b9b805af1cdb60628898c9f411996301a1c13afd3dbef1d8a16ac6dbf503a
\"\n },\n \layers\": [\n {\n \mediaType\": \application/
vnd.docker.image.rootfs.diff.tar.gzip",\n \size\": 43252507,\n
\digest\": \sha256:3b37166ec61459e76e33282dda08f2a9cd698ca7e3d6bc44e6a6e7580cdeff8e
\"\n },\n {\n \mediaType\": \application/
vnd.docker.image.rootfs.diff.tar.gzip",\n \size\": 846,\n \digest
\": \sha256:504facff238fde83f1ca8f9f54520b4219c5b8f80be9616ddc52d31448a044bd
\"\n },\n {\n \mediaType\": \application/
vnd.docker.image.rootfs.diff.tar.gzip",\n \size\": 615,\n \digest
\": \sha256:ebbcacd28e101968415b0c812b2d2dc60f969e36b0b08c073bf796e12b1bb449\"\n
 },\n {\n \mediaType\": \application/
vnd.docker.image.rootfs.diff.tar.gzip",\n \size\": 850,\n \digest
\": \sha256:c7fb3351ecad291a88b92b600037e2435c84a347683d540042086fe72c902b8a
\"\n },\n {\n \mediaType\": \application/
vnd.docker.image.rootfs.diff.tar.gzip",\n \size\": 168,\n \digest\":
\sha256:2e3debadcbf7e542e2aefbce1b64a358b1931fb403b3e4aeca27cb4d809d56c2\"\n },
\n {\n \mediaType\": \application/vnd.docker.image.rootfs.diff.tar.gzip
\",,\n \size\": 37720774,\n \digest\":
\sha256:f8c9f51ad524d8ae9bf4db69cd3e720ba92373ec265f5c390ffb21bb0c277941\"\n
 },\n {\n \mediaType\": \application/
vnd.docker.image.rootfs.diff.tar.gzip",\n \size\": 30432107,\n
\digest\": \sha256:813a50b13f61cf1f8d25f19fa96ad3aa5b552896c83e86ce413b48b091d7f01b
\"\n },\n {\n \mediaType\": \application/
vnd.docker.image.rootfs.diff.tar.gzip",\n \size\": 197,\n \digest
\": \sha256:7ab043301a6187ea3293d80b30ba06c7bf1a0c3cd4c43d10353b31bc0cecfe7d
\"\n },\n {\n \mediaType\": \application/
vnd.docker.image.rootfs.diff.tar.gzip",\n \size\": 154,\n \digest
\": \sha256:67012cca8f31dc3b8ee2305e7762fee20c250513effdedb38a1c37784a5a2e71\"\n

```

```

 },\n {\n \"mediaType\": \"application/
vnd.docker.image.rootfs.diff.tar.gzip\",,\n \"size\": 176,\n \"digest
\": \"sha256:3bc892145603fffc9b1c97c94e2985b4cb19ca508750b15845a5d97becbd1a0e
\"\n },\n {\n \"mediaType\": \"application/
vnd.docker.image.rootfs.diff.tar.gzip\",,\n \"size\": 183,\n \"digest
\": \"sha256:6f1c79518f18251d35977e7e46bfa6c6b9cf50df2a79d4194941d95c54258d18\"\n
 },\n {\n \"mediaType\": \"application/
vnd.docker.image.rootfs.diff.tar.gzip\",,\n \"size\": 212,\n \"digest
\": \"sha256:b7bcfbc2e2888afebede4dd1cd5eebf029bb6315feeaf0b56e425e11a50afe42\"\n
 },\n {\n \"mediaType\": \"application/
vnd.docker.image.rootfs.diff.tar.gzip\",,\n \"size\": 212,\n \"digest\":
\"sha256:2b220f8b0f32b7c2ed8eaafe1c80263bbd94849b9ab73926f0ba46cdae91629\"\n }
]\n}]",
 "registryId": "123456789012",
 "imageId": {
 "imageDigest":
"sha256:98c8b060c21d9adbb6b8c41b916e95e6307102786973ab93a41e8b86d1fc6d3e",
 "imageTag": "latest"
 }
}
},
"requestID": "cf044b7d-5f9d-11e9-9b2a-95983139cc57",
"eventID": "2bfd4ee2-2178-4a82-a27d-b12939923f0f",
"resources": [{
 "ARN": "arn:aws:ecr:us-east-2:123456789012:repository/testrepo",
 "accountId": "123456789012"
}],
"eventType": "AwsApiCall",
"recipientAccountId": "123456789012"
}

```

### 範例：映像提取動作

下列範例顯示 CloudTrail 日誌項目，此項目示範了使用 BatchGetImage 動作的映像提取。

#### Note

提取映像時，如果您在本機尚未有映像，則也會在 CloudTrail 日誌中看到 `GetDownloadUrlForLayer` 參考。

```
{
```

```
"eventVersion": "1.04",
"userIdentity": {
 "type": "IAMUser",
 "principalId": "AIDACKCEVSQ6C2EXAMPLE:account_name",
 "arn": "arn:aws:sts::123456789012:user/Mary_Major",
 "accountId": "123456789012",
 "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
 "userName": "Mary_Major",
 "sessionContext": {
 "attributes": {
 "mfaAuthenticated": "false",
 "creationDate": "2019-04-15T16:42:14Z"
 }
 }
},
"eventTime": "2019-04-15T17:23:20Z",
"eventSource": "ecr.amazonaws.com",
"eventName": "BatchGetImage",
"awsRegion": "us-east-2",
"sourceIPAddress": "ecr.amazonaws.com",
"userAgent": "ecr.amazonaws.com",
"requestParameters": {
 "imageIds": [{
 "imageTag": "latest"
 }],
 "acceptedMediaTypes": [
 "application/json",
 "application/vnd.oci.image.manifest.v1+json",
 "application/vnd.oci.image.index.v1+json",
 "application/vnd.docker.distribution.manifest.v2+json",
 "application/vnd.docker.distribution.manifest.list.v2+json",
 "application/vnd.docker.distribution.manifest.v1+prettyjws"
],
 "repositoryName": "testrepo",
 "registryId": "123456789012"
},
"responseElements": null,
"requestID": "2a1b97ee-5fa3-11e9-a8cd-cd2391aeda93",
"eventID": "c84f5880-c2f9-4585-9757-28fa5c1065df",
"resources": [{
 "ARN": "arn:aws:ecr:us-east-2:123456789012:repository/testrepo",
 "accountId": "123456789012"
}],
"eventType": "AwsApiCall",
```

```
"recipientAccountId": "123456789012"
}
```

### 範例：映像生命週期政策動作

以下範例顯示 CloudTrail 日誌項目，其示範映像何時會因為生命週期政策規則而導致過期。您可以藉由為事件名稱欄位篩選 PolicyExecutionEvent，找出事件類型。

當您測試生命週期政策預覽時，Amazon ECR 會產生事件名稱欄位為的 CloudTrail 日誌項目 DryRunEvent，其結構與完全相同 PolicyExecutionEvent。透過將事件名稱變更為 DryRunEvent，您可以改為篩選乾執行事件。

#### Important

由於個別 CloudTrail 事件的大小限制，對於有 10 個或更多映像過期的生命週期政策動作，Amazon ECR 會向 CloudTrail 發送多個事件。此外，Amazon ECR 每個映像最多包含 100 個標籤。

```
{
 "eventVersion": "1.05",
 "userIdentity": {
 "accountId": "123456789012",
 "invokedBy": "AWS Internal"
 },
 "eventTime": "2020-03-12T20:22:12Z",
 "eventSource": "ecr.amazonaws.com",
 "eventName": "PolicyExecutionEvent",
 "awsRegion": "us-west-2",
 "sourceIPAddress": "AWS Internal",
 "userAgent": "AWS Internal",
 "requestParameters": null,
 "responseElements": null,
 "eventID": "9354dd7f-9aac-4e9d-956d-12561a4923aa",
 "readOnly": true,
 "resources": [
 {
 "ARN": "arn:aws:ecr:us-west-2:123456789012:repository/testrepo",
 "accountId": "123456789012",
 "type": "AWS::ECR::Repository"
 }
],
}
```

```
"eventType": "AwsServiceEvent",
"recipientAccountId": "123456789012",
"serviceEventDetails": {
 "repositoryName": "testrepo",
 "lifecycleEventPolicy": {
 "lifecycleEventRules": [
 {
 "rulePriority": 1,
 "description": "remove all images > 2",
 "lifecycleEventSelection": {
 "tagStatus": "Any",
 "tagPrefixList": [],
 "countType": "Image count more than",
 "countNumber": 2
 },
 "action": "expire"
 }
],
 "lastEvaluatedAt": 0,
 "policyVersion": 1,
 "policyId": "ceb86829-58e7-9498-920c-aa042e33037b"
 },
 "lifecycleEventImageActions": [
 {
 "lifecycleEventImage": {
 "digest":
"sha256:ddba4d27a7ffc3f86dd6c2f92041af252a1f23a8e742c90e6e1297bfa1bc0c45",
 "tagStatus": "Tagged",
 "tagList": [
 "alpine"
],
 "pushedAt": 1584042813000
 },
 "rulePriority": 1
 },
 {
 "lifecycleEventImage": {
 "digest":
"sha256:6ab380c5a5acf71c1b6660d645d2cd79cc8ce91b38e0352cbf9561e050427baf",
 "tagStatus": "Tagged",
 "tagList": [
 "centos"
],
 "pushedAt": 1584042842000
 }
 }
]
}
```

```

 },
 "rulePriority": 1
 }
],
"lifecycleEventFailureDetails": [
 {
 "lifecycleEventImage": {
 "digest":
"sha256:9117e1bc28cd20751e584b4ccd19b1178d14cf02d134b04ce6be0cc51bff762a",
 "tagStatus": "Untagged",
 "tagList": [],
 "pushedAt": 1584042844000
 },
 "rulePriority": 1,
 "failureCode": "ImageReferencedByManifestList",
 "failureReason": "Requested image referenced by manifest list:
[sha256:4b27c83d44a18c31543039d9e8b2786043ec6c8d00804d5800c5148d6b6f65bc]"
 }
]
}
}

```

### 範例：映像封存動作

下列範例顯示 CloudTrail 日誌項目，示範使用 UpdateImageStorageClass 動作將 targetStorageClass 設定為 來封存映像 ARCHIVE。

```

{
 "eventVersion": "1.11",
 "userIdentity": {
 "type": "IAMUser",
 "principalId": "AIDACKCEVSQ6C2EXAMPLE:account_name",
 "arn": "arn:aws:sts::123456789012:user/Mary_Major",
 "accountId": "123456789012",
 "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
 "userName": "Mary_Major",
 "sessionContext": {
 "attributes": {
 "mfaAuthenticated": "false",
 "creationDate": "2019-04-15T16:42:14Z"
 }
 }
 }
},

```

```
"eventTime": "2019-04-15T16:45:00Z",
"eventSource": "ecr.amazonaws.com",
"eventName": "UpdateImageStorageClass",
"awsRegion": "us-east-2",
"sourceIPAddress": "AWS Internal",
"userAgent": "AWS Internal",
"requestParameters": {
 "repositoryName": "testrepo",
 "imageId": {
 "imageDigest":
"sha256:98c8b060c21d9adbb6b8c41b916e95e6307102786973ab93a41e8b86d1fc6d3e"
 },
 "targetStorageClass": "ARCHIVE",
 "registryId": "123456789012"
},
"responseElements": {
 "image": {
 "registryId": "123456789012",
 "repositoryName": "testrepo",
 "imageId": {
 "imageDigest":
"sha256:98c8b060c21d9adbb6b8c41b916e95e6307102786973ab93a41e8b86d1fc6d3e"
 },
 "imageStatus": "ARCHIVED"
 }
},
"requestID": "cf044b7d-EXAMPLE",
"eventID": "2bfd4ee2-EXAMPLE",
"readOnly": false,
"resources": [{
 "ARN": "arn:aws:ecr:us-east-2:123456789012:repository/testrepo",
 "accountId": "123456789012"
}],
"eventType": "AwsApiCall",
"managementEvent": true,
"recipientAccountId": "123456789012",
"eventCategory": "Management"
}
```

## 範例：映像還原動作

下列範例顯示示範正在還原映像的 CloudTrail 日誌項目。當您還原封存的映像時，會產生兩個事件：

### 1. 啟動還原時的 API 呼叫事件

## 2. 非同步還原操作完成時的服務事件

### API 呼叫事件（還原啟動）

下列範例顯示初始 API 呼叫，以使用 UpdateImageStorageClass 動作將 targetStorageClass 設定為 來還原映像 STANDARD。回應會將影像狀態顯示為 ACTIVATING。

```
{
 "eventVersion": "1.11",
 "userIdentity": {
 "type": "IAMUser",
 "principalId": "AIDACKCEVSQ6C2EXAMPLE:account_name",
 "arn": "arn:aws:sts::123456789012:user/Mary_Major",
 "accountId": "123456789012",
 "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
 "userName": "Mary_Major",
 "sessionContext": {
 "attributes": {
 "mfaAuthenticated": "false",
 "creationDate": "2019-04-15T16:42:14Z"
 }
 }
 },
 "eventTime": "2019-04-15T16:45:00Z",
 "eventSource": "ecr.amazonaws.com",
 "eventName": "UpdateImageStorageClass",
 "awsRegion": "us-east-2",
 "sourceIPAddress": "AWS Internal",
 "userAgent": "AWS Internal",
 "requestParameters": {
 "repositoryName": "testrepo",
 "imageId": {
 "imageDigest":
"sha256:98c8b060c21d9adbb6b8c41b916e95e6307102786973ab93a41e8b86d1fc6d3e"
 },
 "targetStorageClass": "STANDARD",
 "registryId": "123456789012"
 },
 "responseElements": {
 "image": {
 "registryId": "123456789012",
 "repositoryName": "testrepo",
 "imageId": {
```

```

 "imageDigest":
 "sha256:98c8b060c21d9adbb6b8c41b916e95e6307102786973ab93a41e8b86d1fc6d3e"
 },
 "imageStatus": "ACTIVATING"
 }
},
"requestID": "cf044b7d-EXAMPLE",
"eventID": "2bfd4ee2-EXAMPLE",
"readOnly": false,
"resources": [{
 "ARN": "arn:aws:ecr:us-east-2:123456789012:repository/testrepo",
 "accountId": "123456789012"
}],
"eventType": "AwsApiCall",
"managementEvent": true,
"recipientAccountId": "123456789012",
"eventCategory": "Management"
}

```

## 服務事件（還原完成）

下列範例顯示非同步還原操作完成時產生的服務事件。您可以藉由為事件名稱欄位篩選 `ImageActivationEvent`，找出事件類型。`serviceEventDetails` 區段包含還原結果和最終影像狀態。

```

{
 "eventVersion": "1.11",
 "userIdentity": {
 "accountId": "123456789012",
 "invokedBy": "AWS Internal"
 },
 "eventTime": "2020-03-12T20:22:12Z",
 "eventSource": "ecr.amazonaws.com",
 "eventName": "ImageActivationEvent",
 "awsRegion": "us-west-2",
 "sourceIPAddress": "AWS Internal",
 "userAgent": "AWS Internal",
 "requestParameters": null,
 "responseElements": null,
 "eventID": "9354dd7f-EXAMPLE",
 "readOnly": true,
 "resources": [
 {

```

```

 "ARN": "arn:aws:ecr:us-west-2:123456789012:repository/testrepo",
 "accountId": "123456789012",
 "type": "AWS::ECR::Repository"
 }
],
"eventType": "AwsServiceEvent",
"managementEvent": true,
"recipientAccountId": "123456789012",
"serviceEventDetails": {
 "repositoryName": "testrepo",
 "imageDigest":
"sha256:98c8b060c21d9adbb6b8c41b916e95e6307102786973ab93a41e8b86d1fc6d3e",
 "targetStorageClass": "STANDARD",
 "result": "SUCCESS",
 "imageStatus": "ACTIVE"
},
"eventCategory": "Management"
}

```

### 範例：映像參考程式動作

下列範例顯示 AWS CloudTrail 日誌項目，示範 OCI 1.1 合規用戶端何時使用 Referrers API 擷取映像的參考者或參考成品清單。

```

{
 "eventVersion": "1.08",
 "userIdentity": {
 "type": "AssumedRole",
 "principalId": "AIDACKCEVSQ6C2EXAMPLE:account_name",
 "arn": "arn:aws:sts::123456789012:user/Mary_Major",
 "accountId": "123456789012",
 "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
 "sessionContext": {
 "sessionIssuer": {
 "type": "Role",
 "principalId": "AIDACKCEVSQ6C2EXAMPLE",
 "arn": "arn:aws:iam::123456789012:role/Admin",
 "accountId": "123456789012",
 "userName": "Admin"
 },
 "webIdFederationData": {},
 "attributes": {
 "creationDate": "2024-10-08T16:38:39Z",

```

```
 "mfaAuthenticated": "false"
 },
 "ec2RoleDelivery": "2.0"
 },
 "invokedBy": "ecr.amazonaws.com"
 },
 "eventTime": "2024-10-08T17:22:51Z",
 "eventSource": "ecr.amazonaws.com",
 "eventName": "ListImageReferrers",
 "awsRegion": "us-east-2",
 "sourceIPAddress": "ecr.amazonaws.com",
 "userAgent": "ecr.amazonaws.com",
 "requestParameters": {
 "registryId": "123456789012",
 "repositoryName": "testrepo",
 "subjectId": {
 "imageDigest":
"sha256:000b9b805af1cdb60628898c9f411996301a1c13afd3dbef1d8a16ac6dbf503a"
 },
 "nextToken": "urD72mdD/mC8b5-EXAMPLE"
 },
 "responseElements": null,
 "requestID": "cb8c167e-EXAMPLE",
 "eventID": "e3c6f4ce-EXAMPLE",
 "readOnly": true,
 "resources": [
 {
 "accountId": "123456789012",
 "ARN": "arn:aws:ecr:us-east-2:123456789012:repository/testrepo"
 }
],
 "eventType": "AwsApiCall",
 "managementEvent": true,
 "recipientAccountId": "123456789012",
 "eventCategory": "Management"
}
```

# 搭配 SDK 使用 Amazon ECR AWS

AWS 軟體開發套件 (SDKs) 適用於許多熱門的程式設計語言。每個 SDK 都提供 API、程式碼範例和說明文件，讓開發人員能夠更輕鬆地以偏好的語言建置應用程式。

| SDK 文件                                       | 代碼範例                                               |
|----------------------------------------------|----------------------------------------------------|
| <a href="#">適用於 C++ 的 AWS SDK</a>            | <a href="#">適用於 C++ 的 AWS SDK 程式碼範例</a>            |
| <a href="#">AWS CLI</a>                      | <a href="#">AWS CLI 程式碼範例</a>                      |
| <a href="#">適用於 Go 的 AWS SDK</a>             | <a href="#">適用於 Go 的 AWS SDK 程式碼範例</a>             |
| <a href="#">適用於 Java 的 AWS SDK</a>           | <a href="#">適用於 Java 的 AWS SDK 程式碼範例</a>           |
| <a href="#">適用於 JavaScript 的 AWS SDK</a>     | <a href="#">適用於 JavaScript 的 AWS SDK 程式碼範例</a>     |
| <a href="#">適用於 Kotlin 的 AWS SDK</a>         | <a href="#">適用於 Kotlin 的 AWS SDK 程式碼範例</a>         |
| <a href="#">適用於 .NET 的 AWS SDK</a>           | <a href="#">適用於 .NET 的 AWS SDK 程式碼範例</a>           |
| <a href="#">適用於 PHP 的 AWS SDK</a>            | <a href="#">適用於 PHP 的 AWS SDK 程式碼範例</a>            |
| <a href="#">AWS Tools for PowerShell</a>     | <a href="#">AWS Tools for PowerShell 程式碼範例</a>     |
| <a href="#">適用於 Python (Boto3) 的 AWS SDK</a> | <a href="#">適用於 Python (Boto3) 的 AWS SDK 程式碼範例</a> |
| <a href="#">適用於 Ruby 的 AWS SDK</a>           | <a href="#">適用於 Ruby 的 AWS SDK 程式碼範例</a>           |
| <a href="#">適用於 Rust 的 AWS SDK</a>           | <a href="#">適用於 Rust 的 AWS SDK 程式碼範例</a>           |
| <a href="#">適用於 SAP ABAP 的 AWS SDK</a>       | <a href="#">適用於 SAP ABAP 的 AWS SDK 程式碼範例</a>       |
| <a href="#">適用於 Swift 的 AWS SDK</a>          | <a href="#">適用於 Swift 的 AWS SDK 程式碼範例</a>          |

## 可用性範例

找不到所需的內容嗎？請使用本頁面底部的提供意見回饋連結申請程式碼範例。

# 使用 AWS SDKs Amazon ECR 程式碼範例

下列程式碼範例示範如何搭配 AWS 軟體開發套件 (SDK) 使用 Amazon ECR。

基本概念是程式碼範例，這些範例說明如何在服務內執行基本操作。

Actions 是大型程式的程式碼摘錄，必須在內容中執行。雖然動作會告訴您如何呼叫個別服務函數，但您可以在其相關情境中查看內容中的動作。

案例是向您展示如何呼叫服務中的多個函數或與其他 AWS 服務組合來完成特定任務的程式碼範例。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱 [搭配 SDK 使用 Amazon ECR AWS](#)。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

## 程式碼範例

- [使用 AWS SDKs 的 Amazon ECR 基本範例](#)
  - [Hello Amazon ECR](#)
  - [使用 AWS SDK 了解 Amazon ECR 的基本概念](#)
  - [使用 AWS SDKs 的 Amazon ECR 動作](#)
    - [CreateRepository 搭配 AWS SDK 或 CLI 使用](#)
    - [DeleteRepository 搭配 AWS SDK 或 CLI 使用](#)
    - [DescribeImages 搭配 AWS SDK 或 CLI 使用](#)
    - [DescribeRepositories 搭配 AWS SDK 或 CLI 使用](#)
    - [GetAuthorizationToken 搭配 AWS SDK 或 CLI 使用](#)
    - [GetRepositoryPolicy 搭配 AWS SDK 或 CLI 使用](#)
    - [ListImages 搭配 AWS SDK 或 CLI 使用](#)
    - [PushImageCmd 搭配 AWS SDK 使用](#)
    - [PutLifecyclePolicy 搭配 AWS SDK 或 CLI 使用](#)
    - [SetRepositoryPolicy 搭配 AWS SDK 或 CLI 使用](#)
    - [StartLifecyclePolicyPreview 搭配 AWS SDK 或 CLI 使用](#)
- [使用 AWS SDKs Amazon ECR 案例](#)
  - [Amazon ECR 入門](#)

# 使用 AWS SDKs 的 Amazon ECR 基本範例

下列程式碼範例示範如何搭配 AWS SDK 使用 Amazon Elastic Container Registry 的基本功能。

## 範例

- [Hello Amazon ECR](#)
- [使用 AWS SDK 了解 Amazon ECR 的基本概念](#)
- [使用 AWS SDKs 的 Amazon ECR 動作](#)
  - [CreateRepository 搭配 AWS SDK 或 CLI 使用](#)
  - [DeleteRepository 搭配 AWS SDK 或 CLI 使用](#)
  - [DescribeImages 搭配 AWS SDK 或 CLI 使用](#)
  - [DescribeRepositories 搭配 AWS SDK 或 CLI 使用](#)
  - [GetAuthorizationToken 搭配 AWS SDK 或 CLI 使用](#)
  - [GetRepositoryPolicy 搭配 AWS SDK 或 CLI 使用](#)
  - [ListImages 搭配 AWS SDK 或 CLI 使用](#)
  - [PushImageCmd 搭配 AWS SDK 使用](#)
  - [PutLifecyclePolicy 搭配 AWS SDK 或 CLI 使用](#)
  - [SetRepositoryPolicy 搭配 AWS SDK 或 CLI 使用](#)
  - [StartLifecyclePolicyPreview 搭配 AWS SDK 或 CLI 使用](#)

## Hello Amazon ECR

下列程式碼範例示範如何開始使用 Amazon ECR。

### Java

適用於 Java 2.x 的 SDK

#### Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
import software.amazon.awssdk.regions.Region;
```

```
import software.amazon.awssdk.services.ecr.EcrClient;
import software.amazon.awssdk.services.ecr.model.EcrException;
import software.amazon.awssdk.services.ecr.model.ListImagesRequest;
import software.amazon.awssdk.services.ecr.paginators.ListImagesIterable;

public class HelloECR {

 public static void main(String[] args) {
 final String usage = ""
 Usage: <repositoryName>

 Where:
 repositoryName - The name of the Amazon ECR repository.
 "";

 if (args.length != 1) {
 System.out.println(usage);
 System.exit(1);
 }

 String repoName = args[0];
 EcrClient ecrClient = EcrClient.builder()
 .region(Region.US_EAST_1)
 .build();

 listImageTags(ecrClient, repoName);
 }

 public static void listImageTags(EcrClient ecrClient, String repoName){
 ListImagesRequest listImagesPaginator = ListImagesRequest.builder()
 .repositoryName(repoName)
 .build();

 ListImagesIterable imagesIterable =
 ecrClient.listImagesPaginator(listImagesPaginator);
 imagesIterable.stream()
 .flatMap(r -> r.imageIds().stream())
 .forEach(image -> System.out.println("The docker image tag is: "
+image.imageTag()));
 }
}
```

- 如需 API 詳細資訊，請參閱《AWS SDK for Java 2.x API 參考》中的 [listImages](#)。

## Kotlin

### 適用於 Kotlin 的 SDK

#### Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
import aws.sdk.kotlin.services.ecr.EcrClient
import aws.sdk.kotlin.services.ecr.model.ListImagesRequest
import kotlin.system.exitProcess

suspend fun main(args: Array<String>) {
 val usage = """
 Usage: <repositoryName>

 Where:
 repositoryName - The name of the Amazon ECR repository.

 """.trimIndent()

 if (args.size != 1) {
 println(usage)
 exitProcess(1)
 }

 val repoName = args[0]
 listImageTags(repoName)
}

suspend fun listImageTags(repoName: String?) {
 val listImages =
 ListImagesRequest {
 repositoryName = repoName
 }

 EcrClient.fromEnvironment { region = "us-east-1" }.use { ecrClient ->
 val imageResponse = ecrClient.listImages(listImages)
 imageResponse.imageIds?.forEach { imageId ->
 println("Image tag: ${imageId.imageTag}")
 }
 }
}
```

```
 }
 }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的 AWS SDK API 參考》中的 [listImages](#)。

## Python

### 適用於 Python 的 SDK (Boto3)

#### Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
import boto3
import argparse
from boto3 import client

def hello_ecr(ecr_client: client, repository_name: str) -> None:
 """
 Use the AWS SDK for Python (Boto3) to create an Amazon Elastic Container
 Registry (Amazon ECR)
 client and list the images in a repository.
 This example uses the default settings specified in your shared credentials
 and config files.

 :param ecr_client: A Boto3 Amazon ECR Client object. This object wraps
 the low-level Amazon ECR service API.
 :param repository_name: The name of an Amazon ECR repository in your account.
 """
 print(
 f"Hello, Amazon ECR! Let's list some images in the repository
{repository_name}:\n"
)
 paginator = ecr_client.get_paginator("list_images")
 page_iterator = paginator.paginate(
 repositoryName=repository_name, PaginationConfig={"MaxItems": 10}
)
```

```
image_names: [str] = []
for page in page_iterator:
 for schedule in page["imageIds"]:
 image_names.append(schedule["imageTag"])

print(f"{len(image_names)} image(s) retrieved.")
for schedule_name in image_names:
 print(f"\t{schedule_name}")

if __name__ == "__main__":
 parser = argparse.ArgumentParser(description="Run hello Amazon ECR.")
 parser.add_argument(
 "--repository-name",
 type=str,
 help="the name of an Amazon ECR repository in your account.",
 required=True,
)
 args = parser.parse_args()

 hello_ecr(boto3.client("ecr"), args.repository_name)
```

- 如需 API 詳細資訊，請參閱《AWS SDK for Python (Boto3) API 參考》中的 [listImages](#)。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱 [搭配 SDK 使用 Amazon ECR AWS](#)。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

## 使用 AWS SDK 了解 Amazon ECR 的基本概念

下列程式碼範例示範如何：

- 建立 Amazon ECR 儲存庫。
- 設定儲存庫政策。
- 擷取儲存庫 URI。
- 取得 Amazon ECR 授權字符。
- 設定 Amazon ECR 儲存庫的生命週期政策。
- 將 Docker 映像檔推送至 Amazon ECR 儲存庫。
- 驗證 Amazon ECR 儲存庫中是否有映像存在。

- 列出您帳戶的 Amazon ECR 儲存庫，並取得其詳細資訊。
- 刪除 Amazon ECR 儲存庫。

## Java

### 適用於 Java 2.x 的 SDK

#### Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

執行展示 Amazon ECR 功能的互動式情境。

```
import software.amazon.awssdk.services.ecr.model.EcrException;
import
 software.amazon.awssdk.services.ecr.model.RepositoryPolicyNotFoundException;

import java.util.Scanner;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
 * started.html
 *
 * This Java code example requires an IAM Role that has permissions to interact
 * with the Amazon ECR service.
 *
 * To create an IAM role, see:
 *
 * https://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles_create.html
 *
 * This Java scenario example requires a local docker image named echo-text.
 * Without a local image,
 * this Java program will not successfully run. For more information including
 * how to create the local
```

```
* image, see:
*
* /scenarios/basics/ecr/README
*
*/
public class ECRScenario {
 public static final String DASHES = new String(new char[80]).replace("\0",
"-");
 public static void main(String[] args) {
 final String usage = ""
 Usage: <iamRoleARN> <accountId>

 Where:
 iamRoleARN - The IAM role ARN that has the necessary permissions
to access and manage the Amazon ECR repository.
 accountId - Your AWS account number.
 """;

 if (args.length != 2) {
 System.out.println(usage);
 return;
 }

 ECRActions ecrActions = new ECRActions();
 String iamRole = args[0];
 String accountId = args[1];
 String localImageName;

 Scanner scanner = new Scanner(System.in);
 System.out.println("""
 The Amazon Elastic Container Registry (ECR) is a fully-managed
Docker container registry
 service provided by AWS. It allows developers and organizations to
securely
 store, manage, and deploy Docker container images.
 ECR provides a simple and scalable way to manage container images
throughout their lifecycle,
 from building and testing to production deployment.\s

 The `EcrAsyncClient` interface in the AWS SDK for Java 2.x provides
a set of methods to
 programmatically interact with the Amazon ECR service. This allows
developers to
```

automate the storage, retrieval, and management of container images as part of their application deployment pipelines. With ECR, teams can focus on building and deploying their applications without having to worry about the underlying infrastructure required to host and manage a container registry.

This scenario walks you through how to perform key operations for this service.

Let's get started...

You have two choices:

- 1 - Run the entire program.
- 2 - Delete an existing Amazon ECR repository named echo-text (created from a previous execution of this program that did not complete).

```
""");

while (true) {
 String input = scanner.nextLine();
 if (input.trim().equalsIgnoreCase("1")) {
 System.out.println("Continuing with the program...");
 System.out.println("");
 break;
 } else if (input.trim().equalsIgnoreCase("2")) {
 String repoName = "echo-text";
 ecrActions.deleteECRRepository(repoName);
 return;
 } else {
 // Handle invalid input.
 System.out.println("Invalid input. Please try again.");
 }
}

waitForInputToContinue(scanner);
System.out.println(DASHES);

System.out.println("""
1. Create an ECR repository.
```

The first task is to ensure we have a local Docker image named echo-text.

If this image exists, then an Amazon ECR repository is created.

An ECR repository is a private Docker container repository provided by Amazon Web Services (AWS). It is a managed service that makes it easy to store, manage, and deploy Docker container images.

```
"");

// Ensure that a local docker image named echo-text exists.
boolean doesExist = ecrActions.isEchoTextImagePresent();
String repoName;
if (!doesExist){
 System.out.println("The local image named echo-text does not exist");
 return;
} else {
 localImageName = "echo-text";
 repoName = "echo-text";
}

try {
 String repoArn = ecrActions.createECRRepository(repoName);
 System.out.println("The ARN of the ECR repository is " + repoArn);

} catch (IllegalArgumentException e) {
 System.err.println("Invalid repository name: " + e.getMessage());
 return;
} catch (RuntimeException e) {
 System.err.println("An error occurred while creating the ECR repository: " + e.getMessage());
 e.printStackTrace();
 return;
}

waitForInputToContinue(scanner);

System.out.println(DASHES);
System.out.println("""
2. Set an ECR repository policy.

Setting an ECR repository policy using the `setRepositoryPolicy` function is crucial for maintaining the security and integrity of your container images. The repository policy allows you to define specific rules and restrictions for accessing and managing the images stored within your ECR repository.
```

```
 """);
waitForInputToContinue(scanner);
try {
 ecrActions.setRepoPolicy(repoName, iamRole);

} catch (RepositoryPolicyNotFoundException e) {
 System.err.println("Invalid repository name: " + e.getMessage());
 return;
} catch (EcrException e) {
 System.err.println("An ECR exception occurred: " + e.getMessage());
 return;
} catch (RuntimeException e) {
 System.err.println("An error occurred while creating the ECR
repository: " + e.getMessage());
 return;
}
waitForInputToContinue(scanner);

System.out.println(DASHES);
System.out.println("""
3. Display ECR repository policy.

Now we will retrieve the ECR policy to ensure it was successfully set.
""");
waitForInputToContinue(scanner);
try {
 String policyText = ecrActions.getRepoPolicy(repoName);
 System.out.println("Policy Text:");
 System.out.println(policyText);

} catch (EcrException e) {
 System.err.println("An ECR exception occurred: " + e.getMessage());
 return;
} catch (RuntimeException e) {
 System.err.println("An error occurred while creating the ECR
repository: " + e.getMessage());
 return;
}

waitForInputToContinue(scanner);

System.out.println(DASHES);
System.out.println("""
4. Retrieve an ECR authorization token.
```

You need an authorization token to securely access and interact with the Amazon ECR registry.

The `getAuthorizationToken` method of the `EcrAsyncClient` is responsible for securely accessing

and interacting with an Amazon ECR repository. This operation is responsible for obtaining a

valid authorization token, which is required to authenticate your requests to the ECR service.

Without a valid authorization token, you would not be able to perform any operations on the

ECR repository, such as pushing, pulling, or managing your Docker images.

```
""");
waitForInputToContinue(scanner);
try {
 ecrActions.getAuthToken();

} catch (EcrException e) {
 System.err.println("An ECR exception occurred: " + e.getMessage());
 return;
} catch (RuntimeException e) {
 System.err.println("An error occurred while retrieving the
authorization token: " + e.getMessage());
 return;
}
waitForInputToContinue(scanner);

System.out.println(DASHES);
System.out.println("""
5. Get the ECR Repository URI.
```

The URI of an Amazon ECR repository is important. When you want to deploy a container image to

a container orchestration platform like Amazon Elastic Kubernetes Service (EKS)

or Amazon Elastic Container Service (ECS), you need to specify the full image URI,

which includes the ECR repository URI. This allows the container runtime to pull the

correct container image from the ECR repository.

```
""");
waitForInputToContinue(scanner);
```

```
try {
 ecrActions.getRepositoryURI(repoName);

} catch (EcrException e) {
 System.err.println("An ECR exception occurred: " + e.getMessage());
 return;

} catch (RuntimeException e) {
 System.err.println("An error occurred while retrieving the URI: " +
e.getMessage());
 return;
}
waitForInputToContinue(scanner);

System.out.println(DASHES);
System.out.println("""
 6. Set an ECR Lifecycle Policy.

 An ECR Lifecycle Policy is used to manage the lifecycle of Docker
images stored in your ECR repositories.
 These policies allow you to automatically remove old or unused Docker
images from your repositories,
 freeing up storage space and reducing costs.

 This example policy helps to maintain the size and efficiency of the
container registry
 by automatically removing older and potentially unused images,
ensuring that the
 storage is optimized and the registry remains up-to-date.
 """);
waitForInputToContinue(scanner);
try {
 ecrActions.setLifeCyclePolicy(repoName);

} catch (RuntimeException e) {
 System.err.println("An error occurred while setting the lifecycle
policy: " + e.getMessage());
 e.printStackTrace();
 return;
}
waitForInputToContinue(scanner);

System.out.println(DASHES);
```

```
System.out.println("""
7. Push a docker image to the Amazon ECR Repository.
```

The `pushImageCmd()` method pushes a local Docker image to an Amazon ECR repository.

It sets up the Docker client by connecting to the local Docker host using the default port.

It then retrieves the authorization token for the ECR repository by making a call to the AWS SDK.

The method uses the authorization token to create an `AuthConfig` object, which is used to authenticate

the Docker client when pushing the image. Finally, the method tags the Docker image with the specified

repository name and image tag, and then pushes the image to the ECR repository using the Docker client.

If the push operation is successful, the method prints a message indicating that the image was pushed to ECR.

```
""");
waitForInputToContinue(scanner);

try {
 ecrActions.pushDockerImage(repoName, localImageName);

} catch (RuntimeException e) {
 System.err.println("An error occurred while pushing a local Docker
image to Amazon ECR: " + e.getMessage());
 e.printStackTrace();
 return;
}
waitForInputToContinue(scanner);

System.out.println(DASHES);
System.out.println("8. Verify if the image is in the ECR Repository.");
waitForInputToContinue(scanner);
try {
 ecrActions.verifyImage(repoName, localImageName);

} catch (EcrException e) {
 System.err.println("An ECR exception occurred: " + e.getMessage());
 return;
} catch (RuntimeException e) {
 System.err.println("An error occurred " + e.getMessage());
 e.printStackTrace();
```

```

 return;
 }
 waitForInputToContinue(scanner);

 System.out.println(DASHES);
 System.out.println("9. As an optional step, you can interact with the
image in Amazon ECR by using the CLI.");
 System.out.println("Would you like to view instructions on how to use the
CLI to run the image? (y/n)");
 String ans = scanner.nextLine().trim();
 if (ans.equalsIgnoreCase("y")) {
 String instructions = ""
 1. Authenticate with ECR - Before you can pull the image from Amazon
ECR, you need to authenticate with the registry. You can do this using the AWS
CLI:

 aws ecr get-login-password --region us-east-1 | docker login --
username AWS --password-stdin %s.dkr.ecr.us-east-1.amazonaws.com

 2. Describe the image using this command:

 aws ecr describe-images --repository-name %s --image-ids imageTag=
%s

 3. Run the Docker container and view the output using this command:

 docker run --rm %s.dkr.ecr.us-east-1.amazonaws.com/%s:%s
 """;
 instructions = String.format(instructions, accountId, repoName,
localImageName, accountId, repoName, localImageName);
 System.out.println(instructions);
 }
 waitForInputToContinue(scanner);

 System.out.println(DASHES);
 System.out.println("10. Delete the ECR Repository.");
 System.out.println(
 ""
 If the repository isn't empty, you must either delete the contents of the
repository
 or use the force option (used in this scenario) to delete the repository
and have Amazon ECR delete all of its contents
 on your behalf.
);
}

```

```
 """);
 System.out.println("Would you like to delete the Amazon ECR Repository?
(y/n)");
 String delAns = scanner.nextLine().trim();
 if (delAns.equalsIgnoreCase("y")) {
 System.out.println("You selected to delete the AWS ECR resources.");

 try {
 ecrActions.deleteECRRepository(repoName);

 } catch (EcrException e) {
 System.err.println("An ECR exception occurred: " +
e.getMessage());
 return;
 } catch (RuntimeException e) {
 System.err.println("An error occurred while deleting the Docker
image: " + e.getMessage());
 e.printStackTrace();
 return;
 }
 }

 System.out.println(DASHES);
 System.out.println("This concludes the Amazon ECR SDK scenario");
 System.out.println(DASHES);
 }

 private static void waitForInputToContinue(Scanner scanner) {
 while (true) {
 System.out.println("");
 System.out.println("Enter 'c' followed by <ENTER> to continue:");
 String input = scanner.nextLine();

 if (input.trim().equalsIgnoreCase("c")) {
 System.out.println("Continuing with the program...");
 System.out.println("");
 break;
 } else {
 // Handle invalid input.
 System.out.println("Invalid input. Please try again.");
 }
 }
 }
}
```

## Amazon ECR SDK 方法的包裝函式類別。

```
import com.github.dockerjava.api.DockerClient;
import com.github.dockerjava.api.exception.DockerClientException;
import com.github.dockerjava.api.model.AuthConfig;
import com.github.dockerjava.api.model.Image;
import com.github.dockerjava.core.DockerClientBuilder;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.client.config.ClientOverrideConfiguration;
import software.amazon.awssdk.http.async.SdkAsyncHttpClient;
import software.amazon.awssdk.http.nio.netty.NettyNioAsyncHttpClient;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ecr.EcrAsyncClient;
import software.amazon.awssdk.services.ecr.model.AuthorizationData;
import software.amazon.awssdk.services.ecr.model.CreateRepositoryRequest;
import software.amazon.awssdk.services.ecr.model.CreateRepositoryResponse;
import software.amazon.awssdk.services.ecr.model.DeleteRepositoryRequest;
import software.amazon.awssdk.services.ecr.model.DeleteRepositoryResponse;
import software.amazon.awssdk.services.ecr.model.DescribeImagesRequest;
import software.amazon.awssdk.services.ecr.model.DescribeImagesResponse;
import software.amazon.awssdk.services.ecr.model.DescribeRepositoriesRequest;
import software.amazon.awssdk.services.ecr.model.DescribeRepositoriesResponse;
import software.amazon.awssdk.services.ecr.model.EcrException;
import software.amazon.awssdk.services.ecr.model.GetAuthorizationTokenResponse;
import software.amazon.awssdk.services.ecr.model.GetRepositoryPolicyRequest;
import software.amazon.awssdk.services.ecr.model.GetRepositoryPolicyResponse;
import software.amazon.awssdk.services.ecr.model.ImageIdentifier;
import software.amazon.awssdk.services.ecr.model.Repository;
import
 software.amazon.awssdk.services.ecr.model.RepositoryPolicyNotFoundException;
import software.amazon.awssdk.services.ecr.model.SetRepositoryPolicyRequest;
import software.amazon.awssdk.services.ecr.model.SetRepositoryPolicyResponse;
import
 software.amazon.awssdk.services.ecr.model.StartLifecyclePolicyPreviewRequest;
import
 software.amazon.awssdk.services.ecr.model.StartLifecyclePolicyPreviewResponse;
import com.github.dockerjava.api.command.DockerCmdExecFactory;
import com.github.dockerjava.netty.NettyDockerCmdExecFactory;
import java.time.Duration;
import java.util.Base64;
```

```
import java.util.List;
import java.util.concurrent.CompletableFuture;
import java.util.concurrent.CompletionException;

public class ECRActions {
 private static EcrAsyncClient ecrClient;

 private static DockerClient dockerClient;

 private static Logger logger = LoggerFactory.getLogger(ECRActions.class);

 /**
 * Creates an Amazon Elastic Container Registry (Amazon ECR) repository.
 *
 * @param repoName the name of the repository to create.
 * @return the Amazon Resource Name (ARN) of the created repository, or an
 empty string if the operation failed.
 * @throws IllegalArgumentException If repository name is invalid.
 * @throws RuntimeException if an error occurs while creating the
 repository.
 */
 public String createECRRepository(String repoName) {
 if (repoName == null || repoName.isEmpty()) {
 throw new IllegalArgumentException("Repository name cannot be null or
empty");
 }

 CreateRepositoryRequest request = CreateRepositoryRequest.builder()
 .repositoryName(repoName)
 .build();

 CompletableFuture<CreateRepositoryResponse> response =
getAsyncClient().createRepository(request);
 try {
 CreateRepositoryResponse result = response.join();
 if (result != null) {
 System.out.println("The " + repoName + " repository was created
successfully.");
 return result.repository().repositoryArn();
 } else {
 throw new RuntimeException("Unexpected response type");
 }
 } catch (CompletionException e) {
 Throwable cause = e.getCause();

```

```
 if (cause instanceof EcrException ex) {
 if
("RepositoryAlreadyExistsException".equals(ex.awsErrorDetails().errorCode())) {
 System.out.println("The Amazon ECR repository already exists,
moving on...");
 DescribeRepositoriesRequest describeRequest =
DescribeRepositoriesRequest.builder()
 .repositoryNames(repoName)
 .build();
 DescribeRepositoriesResponse describeResponse =
getAsyncClient().describeRepositories(describeRequest).join();
 return
describeResponse.repositories().get(0).repositoryArn();
 } else {
 throw new RuntimeException(ex);
 }
 } else {
 throw new RuntimeException(e);
 }
 }
}

/**
 * Deletes an ECR (Elastic Container Registry) repository.
 *
 * @param repoName the name of the repository to delete.
 * @throws IllegalArgumentException if the repository name is null or empty.
 * @throws EcrException if there is an error deleting the repository.
 * @throws RuntimeException if an unexpected error occurs during the deletion
process.
 */
public void deleteECRRepository(String repoName) {
 if (repoName == null || repoName.isEmpty()) {
 throw new IllegalArgumentException("Repository name cannot be null or
empty");
 }

 DeleteRepositoryRequest repositoryRequest =
DeleteRepositoryRequest.builder()
 .force(true)
 .repositoryName(repoName)
 .build();
```

```

 CompletableFuture<DeleteRepositoryResponse> response =
getAsyncClient().deleteRepository(repositoryRequest);
 response.whenComplete((deleteRepositoryResponse, ex) -> {
 if (deleteRepositoryResponse != null) {
 System.out.println("You have successfully deleted the " +
repoName + " repository");
 } else {
 Throwable cause = ex.getCause();
 if (cause instanceof EcrException) {
 throw (EcrException) cause;
 } else {
 throw new RuntimeException("Unexpected error: " +
cause.getMessage(), cause);
 }
 }
 });

 // Wait for the CompletableFuture to complete
 response.join();
 }

private static DockerClient getDockerClient() {
 String osName = System.getProperty("os.name");
 if (osName.startsWith("Windows")) {
 // Make sure Docker Desktop is running.
 String dockerHost = "tcp://localhost:2375"; // Use the Docker Desktop
default port.
 DockerCmdExecFactory dockerCmdExecFactory = new
NettyDockerCmdExecFactory().withReadTimeout(20000).withConnectTimeout(20000);
 dockerClient =
DockerClientBuilder.getInstance(dockerHost).withDockerCmdExecFactory(dockerCmdExecFactory)
 } else {
 dockerClient = DockerClientBuilder.getInstance().build();
 }
 return dockerClient;
}

/**
 * Retrieves an asynchronous Amazon Elastic Container Registry (ECR) client.
 *
 * @return the configured ECR asynchronous client.
 */

```

```
private static EcrAsyncClient getAsyncClient() {

 /**
 * The `NettyNioAsyncHttpClient` class is part of the AWS SDK for Java,
 * version 2,
 * and it is designed to provide a high-performance, asynchronous HTTP
 * client for interacting with AWS services.
 * It uses the Netty framework to handle the underlying network
 * communication and the Java NIO API to
 * provide a non-blocking, event-driven approach to HTTP requests and
 * responses.
 */
 SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
 .maxConcurrency(50) // Adjust as needed.
 .connectionTimeout(Duration.ofSeconds(60)) // Set the connection
 timeout.
 .readTimeout(Duration.ofSeconds(60)) // Set the read timeout.
 .writeTimeout(Duration.ofSeconds(60)) // Set the write timeout.
 .build();

 ClientOverrideConfiguration overrideConfig =
 ClientOverrideConfiguration.builder()
 .apiCallTimeout(Duration.ofMinutes(2)) // Set the overall API call
 timeout.
 .apiCallAttemptTimeout(Duration.ofSeconds(90)) // Set the individual
 call attempt timeout.
 .build();

 if (ecrClient == null) {
 ecrClient = EcrAsyncClient.builder()
 .region(Region.US_EAST_1)
 .httpClient(httpClient)
 .overrideConfiguration(overrideConfig)
 .build();
 }
 return ecrClient;
}

/**
 * Sets the lifecycle policy for the specified repository.
 *
 * @param repoName the name of the repository for which to set the lifecycle
 * policy.
 */
```

```
public void setLifecyclePolicy(String repoName) {
 /*
 * This policy helps to maintain the size and efficiency of the container
 registry
 by automatically removing older and potentially unused images,
 ensuring that the storage is optimized and the registry remains up-to-
 date.
 */
 String polText = ""
 {
 "rules": [
 {
 "rulePriority": 1,
 "description": "Expire images older than 14 days",
 "selection": {
 "tagStatus": "any",
 "countType": "sinceImagePushed",
 "countUnit": "days",
 "countNumber": 14
 },
 "action": {
 "type": "expire"
 }
 }
]
 }
 """;

 StartLifecyclePolicyPreviewRequest lifecyclePolicyPreviewRequest =
 StartLifecyclePolicyPreviewRequest.builder()
 .lifecyclePolicyText(polText)
 .repositoryName(repoName)
 .build();

 CompletableFuture<StartLifecyclePolicyPreviewResponse> response =
 getAsyncClient().startLifecyclePolicyPreview(lifecyclePolicyPreviewRequest);
 response.whenComplete((lifecyclePolicyPreviewResponse, ex) -> {
 if (lifecyclePolicyPreviewResponse != null) {
 System.out.println("Lifecycle policy preview started
 successfully.");
 } else {
 if (ex.getCause() instanceof EcrException) {
 throw (EcrException) ex.getCause();
 } else {

```

```
 String errorMessage = "Unexpected error occurred: " +
ex.getMessage();
 throw new RuntimeException(errorMessage, ex);
 }
}
});
// Wait for the CompletableFuture to complete.
response.join();
}

/**
 * Verifies the existence of an image in an Amazon Elastic Container Registry
(Amazon ECR) repository asynchronously.
 *
 * @param repositoryName The name of the Amazon ECR repository.
 * @param imageTag The tag of the image to verify.
 * @throws EcrException if there is an error retrieving the image
information from Amazon ECR.
 * @throws CompletionException if the asynchronous operation completes
exceptionally.
 */
public void verifyImage(String repositoryName, String imageTag) {
 DescribeImagesRequest request = DescribeImagesRequest.builder()
 .repositoryName(repositoryName)
 .imageIds(ImageIdentifier.builder().imageTag(imageTag).build())
 .build();

 CompletableFuture<DescribeImagesResponse> response =
getAsyncClient().describeImages(request);
 response.whenComplete((describeImagesResponse, ex) -> {
 if (ex != null) {
 if (ex instanceof CompletionException) {
 Throwable cause = ex.getCause();
 if (cause instanceof EcrException) {
 throw (EcrException) cause;
 } else {
 throw new RuntimeException("Unexpected error: " +
cause.getMessage(), cause);
 }
 } else {
 throw new RuntimeException("Unexpected error: " +
ex.getCause());
 }
 }
 });
}
```

```
 } else if (describeImagesResponse != null && !
describeImagesResponse.imageDetails().isEmpty()) {
 System.out.println("Image is present in the repository.");
 } else {
 System.out.println("Image is not present in the repository.");
 }
 });

 // Wait for the CompletableFuture to complete.
 response.join();
}

/**
 * Retrieves the repository URI for the specified repository name.
 *
 * @param repoName the name of the repository to retrieve the URI for.
 * @return the repository URI for the specified repository name.
 * @throws EcrException if there is an error retrieving the repository
information.
 * @throws CompletionException if the asynchronous operation completes
exceptionally.
 */
public void getRepositoryURI(String repoName) {
 DescribeRepositoriesRequest request =
DescribeRepositoriesRequest.builder()
 .repositoryNames(repoName)
 .build();

 CompletableFuture<DescribeRepositoriesResponse> response =
getAsyncClient().describeRepositories(request);
 response.whenComplete((describeRepositoriesResponse, ex) -> {
 if (ex != null) {
 Throwable cause = ex.getCause();
 if (cause instanceof InterruptedException) {
 Thread.currentThread().interrupt();
 String errorMessage = "Thread interrupted while waiting for
asynchronous operation: " + cause.getMessage();
 throw new RuntimeException(errorMessage, cause);
 } else if (cause instanceof EcrException) {
 throw (EcrException) cause;
 } else {
 String errorMessage = "Unexpected error: " +
cause.getMessage();
 throw new RuntimeException(errorMessage, cause);
 }
 }
 });
}
```

```
 }
 } else {
 if (describeRepositoriesResponse != null) {
 if (!describeRepositoriesResponse.repositories().isEmpty()) {
 String repositoryUri =
describeRepositoriesResponse.repositories().get(0).repositoryUri();
 System.out.println("Repository URI found: " +
repositoryUri);
 } else {
 System.out.println("No repositories found for the given
name.");
 }
 } else {
 System.err.println("No response received from
describeRepositories.");
 }
 }
});
response.join();
}

/**
 * Retrieves the authorization token for Amazon Elastic Container Registry
(ECR).
 * This method makes an asynchronous call to the ECR client to retrieve the
authorization token.
 * If the operation is successful, the method prints the token to the
console.
 * If an exception occurs, the method handles the exception and prints the
error message.
 *
 * @throws EcrException if there is an error retrieving the authorization
token from ECR.
 * @throws RuntimeException if there is an unexpected error during the
operation.
 */
public void getAuthToken() {
 CompletableFuture<GetAuthorizationTokenResponse> response =
getAsyncClient().getAuthorizationToken();
 response.whenComplete((authorizationTokenResponse, ex) -> {
 if (authorizationTokenResponse != null) {
 AuthorizationData authorizationData =
authorizationTokenResponse.authorizationData().get(0);
 String token = authorizationData.authorizationToken();
 }
 });
}
```

```
 if (!token.isEmpty()) {
 System.out.println("The token was successfully retrieved.");
 }
 } else {
 if (ex.getCause() instanceof EcrException) {
 throw (EcrException) ex.getCause();
 } else {
 String errorMessage = "Unexpected error occurred: " +
ex.getMessage();
 throw new RuntimeException(errorMessage, ex); // Rethrow the
exception
 }
 }
});
response.join();
}

/**
 * Gets the repository policy for the specified repository.
 *
 * @param repoName the name of the repository.
 * @throws EcrException if an AWS error occurs while getting the repository
policy.
 */
public String getRepoPolicy(String repoName) {
 if (repoName == null || repoName.isEmpty()) {
 throw new IllegalArgumentException("Repository name cannot be null or
empty");
 }

 GetRepositoryPolicyRequest getRepositoryPolicyRequest =
GetRepositoryPolicyRequest.builder()
 .repositoryName(repoName)
 .build();

 CompletableFuture<GetRepositoryPolicyResponse> response =
getAsyncClient().getRepositoryPolicy(getRepositoryPolicyRequest);
 response.whenComplete((resp, ex) -> {
 if (resp != null) {
 System.out.println("Repository policy retrieved successfully.");
 } else {
 if (ex.getCause() instanceof EcrException) {
 throw (EcrException) ex.getCause();
 } else {

```

```
 String errorMessage = "Unexpected error occurred: " +
ex.getMessage();
 throw new RuntimeException(errorMessage, ex);
 }
}
});

GetRepositoryPolicyResponse result = response.join();
return result != null ? result.policyText() : null;
}

/**
 * Sets the repository policy for the specified ECR repository.
 *
 * @param repoName the name of the ECR repository.
 * @param iamRole the IAM role to be granted access to the repository.
 * @throws RepositoryPolicyNotFoundException if the repository policy does
not exist.
 * @throws EcrException if there is an unexpected error
setting the repository policy.
 */
public void setRepoPolicy(String repoName, String iamRole) {
 /**
 This example policy document grants the specified AWS principal the
 permission to perform the
 `ecr:BatchGetImage` action. This policy is designed to allow the
 specified principal
 to retrieve Docker images from the ECR repository.
 */
 String policyDocumentTemplate = ""
 {
 "Version": "2012-10-17",
 "Statement" : [{
 "Sid" : "new statement",
 "Effect" : "Allow",
 "Principal" : {
 "AWS" : "%s"
 },
 "Action" : "ecr:BatchGetImage"
 }]
 }
 """;

 String policyDocument = String.format(policyDocumentTemplate, iamRole);
}
```

```
 SetRepositoryPolicyRequest setRepositoryPolicyRequest =
SetRepositoryPolicyRequest.builder()
 .repositoryName(repoName)
 .policyText(policyDocument)
 .build();

 CompletableFuture<SetRepositoryPolicyResponse> response =
getAsyncClient().setRepositoryPolicy(setRepositoryPolicyRequest);
 response.whenComplete((resp, ex) -> {
 if (resp != null) {
 System.out.println("Repository policy set successfully.");
 } else {
 Throwable cause = ex.getCause();
 if (cause instanceof RepositoryPolicyNotFoundException) {
 throw (RepositoryPolicyNotFoundException) cause;
 } else if (cause instanceof EcrException) {
 throw (EcrException) cause;
 } else {
 String errorMessage = "Unexpected error: " +
cause.getMessage();
 throw new RuntimeException(errorMessage, cause);
 }
 }
 });
 response.join();
 }

 /**
 * Pushes a Docker image to an Amazon Elastic Container Registry (ECR)
 repository.
 *
 * @param repoName the name of the ECR repository to push the image to.
 * @param imageName the name of the Docker image.
 */
 public void pushDockerImage(String repoName, String imageName) {
 System.out.println("Pushing " + imageName + " to Amazon ECR will take a
few seconds.");
 CompletableFuture<AuthConfig> authResponseFuture =
getAsyncClient().getAuthorizationToken()
 .thenApply(response -> {
 String token =
response.authorizationData().get(0).authorizationToken();
 String decodedToken = new
String(Base64.getDecoder().decode(token));
```

```
String password = decodedToken.substring(4);

DescribeRepositoriesResponse descrRepoResponse =
getAsyncClient().describeRepositories(b -> b.repositoryNames(repoName)).join();
Repository repoData =
descrRepoResponse.repositories().stream().filter(r ->
r.repositoryName().equals(repoName)).findFirst().orElse(null);
assert repoData != null;
String registryURL = repoData.repositoryUri().split("/")[0];

AuthConfig authConfig = new AuthConfig()
 .withUsername("AWS")
 .withPassword(password)
 .withRegistryAddress(registryURL);
return authConfig;
})
.thenCompose(authConfig -> {
 DescribeRepositoriesResponse descrRepoResponse =
getAsyncClient().describeRepositories(b -> b.repositoryNames(repoName)).join();
 Repository repoData =
descrRepoResponse.repositories().stream().filter(r ->
r.repositoryName().equals(repoName)).findFirst().orElse(null);
 getDockerClient().tagImageCmd(imageName + ":latest",
repoData.repositoryUri() + ":latest", imageName).exec();
 try {

getDockerClient().pushImageCmd(repoData.repositoryUri()).withTag("echo-
text").withAuthConfig(authConfig).start().awaitCompletion();
 System.out.println("The " + imageName + " was pushed to
ECR");

 } catch (InterruptedException e) {
 throw (RuntimeException) e.getCause();
 }
 return CompletableFuture.completedFuture(authConfig);
});

authResponseFuture.join();
}

// Make sure local image echo-text exists.
public boolean isEchoTextImagePresent() {
 try {
 List<Image> images = getDockerClient().listImagesCmd().exec();
```

```
 boolean helloWorldFound = false;
 for (Image image : images) {
 String[] repoTags = image.getRepoTags();
 if (repoTags != null) {
 for (String tag : repoTags) {
 if (tag.startsWith("echo-text")) {
 System.out.println(tag);
 helloWorldFound = true;
 }
 }
 }
 }
 if (helloWorldFound) {
 System.out.println("The local image named echo-text exists.");
 return true;
 } else {
 System.out.println("The local image named echo-text does not exist.");
 return false;
 }
 } catch (DockerClientException ex) {
 logger.error("ERROR: " + ex.getMessage());
 return false;
 }
}
}
```

- 如需 API 詳細資訊，請參閱《AWS SDK for Java 2.x API 參考》中的下列主題。
  - [CreateRepository](#)
  - [DeleteRepository](#)
  - [DescribeImages](#)
  - [DescribeRepositories](#)
  - [GetAuthorizationToken](#)
  - [GetRepositoryPolicy](#)
  - [SetRepositoryPolicy](#)
  - [StartLifecyclePolicyPreview](#)

## Kotlin

### 適用於 Kotlin 的 SDK

#### Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

執行展示 Amazon ECR 功能的互動式情境。

```
import java.util.Scanner

/**
 * Before running this Kotlin code example, set up your development environment,
 * including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html
 *
 * This code example requires an IAM Role that has permissions to interact with
 * the Amazon ECR service.
 *
 * To create an IAM role, see:
 *
 * https://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles_create.html
 *
 * This code example requires a local docker image named echo-text. Without a
 * local image,
 * this program will not successfully run. For more information including how to
 * create the local
 * image, see:
 *
 * /scenarios/basics/ecr/README
 */

val DASHES = String(CharArray(80)).replace("\u0000", "-")

suspend fun main(args: Array<String>) {
 val usage =
```

```
""
Usage: <iamRoleARN> <accountId>

Where:
 iamRoleARN - The IAM role ARN that has the necessary permissions to
access and manage the Amazon ECR repository.
 accountId - Your AWS account number.

"".trimIndent()

if (args.size != 2) {
 println(usage)
 return
}

var iamRole = args[0]
var localImageName: String
var accountId = args[1]
val ecrActions = ECRActions()
val scanner = Scanner(System.`in`)

println(
 ""
 The Amazon Elastic Container Registry (ECR) is a fully-managed Docker
container registry
 service provided by AWS. It allows developers and organizations to
securely
 store, manage, and deploy Docker container images.
 ECR provides a simple and scalable way to manage container images
throughout their lifecycle,
 from building and testing to production deployment.

 The `EcrClient` service client that is part of the AWS SDK for Kotlin
provides a set of methods to
 programmatically interact with the Amazon ECR service. This allows
developers to
 automate the storage, retrieval, and management of container images as
part of their application
 deployment pipelines. With ECR, teams can focus on building and deploying
their
 applications without having to worry about the underlying infrastructure
required to
 host and manage a container registry.
```

This scenario walks you through how to perform key operations for this service.

Let's get started...

You have two choices:

- 1 - Run the entire program.
- 2 - Delete an existing Amazon ECR repository named echo-text (created from a previous execution of this program that did not complete).

```

 """.trimIndent(),
)

 while (true) {
 val input = scanner.nextLine()
 if (input.trim { it <= ' ' }.equals("1", ignoreCase = true)) {
 println("Continuing with the program...")
 println("")
 break
 } else if (input.trim { it <= ' ' }.equals("2", ignoreCase = true)) {
 val repoName = "echo-text"
 ecrActions.deleteECRRepository(repoName)
 return
 } else {
 // Handle invalid input.
 println("Invalid input. Please try again.")
 }
 }
}

```

```

waitForInputToContinue(scanner)
println(DASHES)
println(
 """
 1. Create an ECR repository.

```

The first task is to ensure we have a local Docker image named echo-text.

If this image exists, then an Amazon ECR repository is created.

An ECR repository is a private Docker container repository provided by Amazon Web Services (AWS). It is a managed service that makes it easy to store, manage, and deploy Docker container images.

```

 """.trimIndent(),

```

```
)

// Ensure that a local docker image named echo-text exists.
val doesExist = ecrActions.listLocalImages()
val repoName: String
if (!doesExist) {
 println("The local image named echo-text does not exist")
 return
} else {
 localImageName = "echo-text"
 repoName = "echo-text"
}

val repoArn = ecrActions.createECRRepository(repoName).toString()
println("The ARN of the ECR repository is $repoArn")
waitForInputToContinue(scanner)

println(DASHES)
println(
 """
 2. Set an ECR repository policy.

 Setting an ECR repository policy using the `setRepositoryPolicy` function
 is crucial for maintaining
 the security and integrity of your container images. The repository
 policy allows you to
 define specific rules and restrictions for accessing and managing the
 images stored within your ECR
 repository.

 """.trimIndent(),
)
waitForInputToContinue(scanner)
ecrActions.setRepoPolicy(repoName, iamRole)
waitForInputToContinue(scanner)

println(DASHES)
println(
 """
 3. Display ECR repository policy.

 Now we will retrieve the ECR policy to ensure it was successfully set.
 """.trimIndent(),
)
```

```
waitForInputToContinue(scanner)
val policyText = ecrActions.getRepoPolicy(repoName)
println("Policy Text:")
println(policyText)
waitForInputToContinue(scanner)
```

```
println(DASHES)
println(
```

```
 ""
```

```
 4. Retrieve an ECR authorization token.
```

You need an authorization token to securely access and interact with the Amazon ECR registry.

The `getAuthorizationToken` method of the `EcrAsyncClient` is responsible for securely accessing

and interacting with an Amazon ECR repository. This operation is responsible for obtaining a

valid authorization token, which is required to authenticate your requests to the ECR service.

Without a valid authorization token, you would not be able to perform any operations on the

ECR repository, such as pushing, pulling, or managing your Docker images.

```
 """.trimIndent(),
)
waitForInputToContinue(scanner)
ecrActions.getAuthToken()
waitForInputToContinue(scanner)
```

```
println(DASHES)
println(
```

```
 ""
```

```
 5. Get the ECR Repository URI.
```

The URI of an Amazon ECR repository is important. When you want to deploy a container image to

a container orchestration platform like Amazon Elastic Kubernetes Service (EKS)

or Amazon Elastic Container Service (ECS), you need to specify the full image URI,

which includes the ECR repository URI. This allows the container runtime to pull the

```
 correct container image from the ECR repository.

 """".trimIndent(),
)
 waitForInputToContinue(scanner)
 val repositoryURI: String? = ecrActions.getRepositoryURI(repoName)
 println("The repository URI is $repositoryURI")
 waitForInputToContinue(scanner)
```

```
println(DASHES)
```

```
println(
```

```
 """
```

```
 6. Set an ECR Lifecycle Policy.
```

An ECR Lifecycle Policy is used to manage the lifecycle of Docker images stored in your ECR repositories.

These policies allow you to automatically remove old or unused Docker images from your repositories, freeing up storage space and reducing costs.

```
 """".trimIndent(),
```

```
)
```

```
 waitForInputToContinue(scanner)
```

```
 val pol = ecrActions.setLifeCyclePolicy(repoName)
```

```
 println(pol)
```

```
 waitForInputToContinue(scanner)
```

```
println(DASHES)
```

```
println(
```

```
 """
```

```
 7. Push a docker image to the Amazon ECR Repository.
```

The `pushImageCmd()` method pushes a local Docker image to an Amazon ECR repository.

It sets up the Docker client by connecting to the local Docker host using the default port.

It then retrieves the authorization token for the ECR repository by making a call to the AWS SDK.

The method uses the authorization token to create an `AuthConfig` object, which is used to authenticate

the Docker client when pushing the image. Finally, the method tags the Docker image with the specified

repository name and image tag, and then pushes the image to the ECR repository using the Docker client.

If the push operation is successful, the method prints a message indicating that the image was pushed to ECR.

```

 """.trimIndent(),
)

 waitForInputToContinue(scanner)
 ecrActions.pushDockerImage(repoName, localImageName)
 waitForInputToContinue(scanner)

 println(DASHES)
 println("8. Verify if the image is in the ECR Repository.")
 waitForInputToContinue(scanner)
 ecrActions.verifyImage(repoName, localImageName)
 waitForInputToContinue(scanner)

 println(DASHES)
 println("9. As an optional step, you can interact with the image in Amazon
 ECR by using the CLI.")
 println("Would you like to view instructions on how to use the CLI to run the
 image? (y/n)")
 val ans = scanner.nextLine().trim()
 if (ans.equals("y", true)) {
 val instructions = """
 1. Authenticate with ECR - Before you can pull the image from Amazon ECR,
 you need to authenticate with the registry. You can do this using the AWS CLI:

 aws ecr get-login-password --region us-east-1 | docker login --
 username AWS --password-stdin $accountId.dkr.ecr.us-east-1.amazonaws.com

 2. Describe the image using this command:

 aws ecr describe-images --repository-name $repoName --image-ids
 imageTag=$localImageName

 3. Run the Docker container and view the output using this command:

 docker run --rm $accountId.dkr.ecr.us-east-1.amazonaws.com/$repoName:
 $localImageName
 """
 println(instructions)
 }
}

```

```
waitForInputToContinue(scanner)

println(DASHES)
println("10. Delete the ECR Repository.")
println(
 """
 If the repository isn't empty, you must either delete the contents of the
repository
 or use the force option (used in this scenario) to delete the repository
and have Amazon ECR delete all of its contents
 on your behalf.

 """.trimIndent(),
)
println("Would you like to delete the Amazon ECR Repository? (y/n)")
val delAns = scanner.nextLine().trim { it <= ' ' }
if (delAns.equals("y", ignoreCase = true)) {
 println("You selected to delete the AWS ECR resources.")
 waitForInputToContinue(scanner)
 ecrActions.deleteECRRepository(repoName)
}

println(DASHES)
println("This concludes the Amazon ECR SDK scenario")
println(DASHES)
}

private fun waitForInputToContinue(scanner: Scanner) {
 while (true) {
 println("")
 println("Enter 'c' followed by <ENTER> to continue:")
 val input = scanner.nextLine()
 if (input.trim { it <= ' ' }.equals("c", ignoreCase = true)) {
 println("Continuing with the program...")
 println("")
 break
 } else {
 // Handle invalid input.
 println("Invalid input. Please try again.")
 }
 }
}
}
```

## Amazon ECR SDK 方法的包裝函式類別。

```
import aws.sdk.kotlin.services.ecr.EcrClient
import aws.sdk.kotlin.services.ecr.model.CreateRepositoryRequest
import aws.sdk.kotlin.services.ecr.model.DeleteRepositoryRequest
import aws.sdk.kotlin.services.ecr.model.DescribeImagesRequest
import aws.sdk.kotlin.services.ecr.model.DescribeRepositoriesRequest
import aws.sdk.kotlin.services.ecr.model.EcrException
import aws.sdk.kotlin.services.ecr.model.GetRepositoryPolicyRequest
import aws.sdk.kotlin.services.ecr.model.ImageIdentifier
import aws.sdk.kotlin.services.ecr.model.RepositoryAlreadyExistsException
import aws.sdk.kotlin.services.ecr.model.SetRepositoryPolicyRequest
import aws.sdk.kotlin.services.ecr.model.StartLifecyclePolicyPreviewRequest
import com.github.dockerjava.api.DockerClient
import com.github.dockerjava.api.command.DockerCmdExecFactory
import com.github.dockerjava.api.model.AuthConfig
import com.github.dockerjava.core.DockerClientBuilder
import com.github.dockerjava.netty.NettyDockerCmdExecFactory
import java.io.IOException
import java.util.Base64

class ECRActions {
 private var dockerClient: DockerClient? = null

 private fun getDockerClient(): DockerClient? {
 val osName = System.getProperty("os.name")
 if (osName.startsWith("Windows")) {
 // Make sure Docker Desktop is running.
 val dockerHost = "tcp://localhost:2375" // Use the Docker Desktop
 default port.
 val dockerCmdExecFactory: DockerCmdExecFactory =
 NettyDockerCmdExecFactory().withReadTimeout(20000).withConnectTimeout(20000)
 dockerClient =
 DockerClientBuilder.getInstance(dockerHost).withDockerCmdExecFactory(dockerCmdExecFactory)
 } else {
 dockerClient = DockerClientBuilder.getInstance().build()
 }
 return dockerClient
 }

 /**
 * Sets the lifecycle policy for the specified repository.
 */
}
```

```
*
* @param repoName the name of the repository for which to set the lifecycle
policy.
*/
suspend fun setLifecyclePolicy(repoName: String): String? {
 val polText =
 """
 {
 "rules": [
 {
 "rulePriority": 1,
 "description": "Expire images older than 14 days",
 "selection": {
 "tagStatus": "any",
 "countType": "sinceImagePushed",
 "countUnit": "days",
 "countNumber": 14
 },
 "action": {
 "type": "expire"
 }
 }
]
 }

 """.trimIndent()
 val lifecyclePolicyPreviewRequest =
 StartLifecyclePolicyPreviewRequest {
 lifecyclePolicyText = polText
 repositoryName = repoName
 }

 // Execute the request asynchronously.
 EcrClient.fromEnvironment { region = "us-east-1" }.use { ecrClient ->
 val response =
 ecrClient.startLifecyclePolicyPreview(lifecyclePolicyPreviewRequest)
 return response.lifecyclePolicyText
 }
}

/**
 * Retrieves the repository URI for the specified repository name.
 */
```

```
* @param repoName the name of the repository to retrieve the URI for.
* @return the repository URI for the specified repository name.
*/
suspend fun getRepositoryURI(repoName: String?): String? {
 require(!(repoName == null || repoName.isEmpty())) { "Repository name
cannot be null or empty" }
 val request =
 DescribeRepositoriesRequest {
 repositoryNames = listOf(repoName)
 }

 EcrClient.fromEnvironment { region = "us-east-1" }.use { ecrClient ->
 val describeRepositoriesResponse =
ecrClient.describeRepositories(request)
 if (!describeRepositoriesResponse.repositories?.isEmpty()!!) {
 return
describeRepositoriesResponse?.repositories?.get(0)?.repositoryUri
 } else {
 println("No repositories found for the given name.")
 return ""
 }
 }
}

/**
 * Retrieves the authorization token for Amazon Elastic Container Registry
(ECR).
 *
 */
suspend fun getAuthToken() {
 EcrClient.fromEnvironment { region = "us-east-1" }.use { ecrClient ->
 // Retrieve the authorization token for ECR.
 val response = ecrClient.getAuthorizationToken()
 val authorizationData = response.authorizationData?.get(0)
 val token = authorizationData?.authorizationToken
 if (token != null) {
 println("The token was successfully retrieved.")
 }
 }
}

/**
```

```
* Gets the repository policy for the specified repository.
*
* @param repoName the name of the repository.
*/
suspend fun getRepoPolicy(repoName: String?): String? {
 require(!(repoName == null || repoName.isEmpty())) { "Repository name
cannot be null or empty" }

 // Create the request
 val getRepositoryPolicyRequest =
 GetRepositoryPolicyRequest {
 repositoryName = repoName
 }
 EcrClient.fromEnvironment { region = "us-east-1" }.use { ecrClient ->
 val response =
ecrClient.getRepositoryPolicy(getRepositoryPolicyRequest)
 val responseText = response.policyText
 return responseText
 }
}

/**
* Sets the repository policy for the specified ECR repository.
*
* @param repoName the name of the ECR repository.
* @param iamRole the IAM role to be granted access to the repository.
*/
suspend fun setRepoPolicy(
 repoName: String?,
 iamRole: String?,
) {
 val policyDocumentTemplate =
 """
 {
 "Version": "2012-10-17",
 "Statement" : [{
 "Sid" : "new statement",
 "Effect" : "Allow",
 "Principal" : {
 "AWS" : "$iamRole"
 },
 "Action" : "ecr:BatchGetImage"
 }]
 }
 """
}
```

```
 }

 """.trimIndent()
 val setRepositoryPolicyRequest =
 SetRepositoryPolicyRequest {
 repositoryName = repoName
 policyText = policyDocumentTemplate
 }

 EcrClient.fromEnvironment { region = "us-east-1" }.use { ecrClient ->
 val response =
 ecrClient.setRepositoryPolicy(setRepositoryPolicyRequest)
 if (response != null) {
 println("Repository policy set successfully.")
 }
 }
 }

}

/**
 * Creates an Amazon Elastic Container Registry (Amazon ECR) repository.
 *
 * @param repoName the name of the repository to create.
 * @return the Amazon Resource Name (ARN) of the created repository, or an
 * empty string if the operation failed.
 * @throws RepositoryAlreadyExistsException if the repository exists.
 * @throws EcrException if an error occurs while creating the
 * repository.
 */
suspend fun createECRRepository(repoName: String?): String? {
 val request =
 CreateRepositoryRequest {
 repositoryName = repoName
 }

 return try {
 EcrClient.fromEnvironment { region = "us-east-1" }.use { ecrClient ->
 val response = ecrClient.createRepository(request)
 response.repository?.repositoryArn
 }
 } catch (e: RepositoryAlreadyExistsException) {
 println("Repository already exists: $repoName")
 repoName?.let { getRepoARN(it) }
 } catch (e: EcrException) {
```

```
 println("An error occurred: ${e.message}")
 null
 }
}

suspend fun getRepoARN(repoName: String): String? {
 // Fetch the existing repository's ARN.
 val describeRequest =
 DescribeRepositoriesRequest {
 repositoryNames = listOf(repoName)
 }
 EcrClient.fromEnvironment { region = "us-east-1" }.use { ecrClient ->
 val describeResponse =
 ecrClient.describeRepositories(describeRequest)
 return describeResponse.repositories?.get(0)?.repositoryArn
 }
}

fun listLocalImages(): Boolean = try {
 val images = getDockerClient()?.listImagesCmd()?.exec()
 images?.any { image ->
 image.repoTags?.any { tag -> tag.startsWith("echo-text") } ?: false
 } ?: false
} catch (ex: Exception) {
 println("ERROR: ${ex.message}")
 false
}

/**
 * Pushes a Docker image to an Amazon Elastic Container Registry (ECR)
 * repository.
 *
 * @param repoName the name of the ECR repository to push the image to.
 * @param imageName the name of the Docker image.
 */
suspend fun pushDockerImage(
 repoName: String,
 imageName: String,
) {
 println("Pushing $imageName to $repoName will take a few seconds")
 val authConfig = getAuthConfig(repoName)

 EcrClient.fromEnvironment { region = "us-east-1" }.use { ecrClient ->
```

```
 val desRequest =
 DescribeRepositoriesRequest {
 repositoryNames = listOf(repoName)
 }

 val describeRepoResponse = ecrClient.describeRepositories(desRequest)
 val repoData =
 describeRepoResponse.repositories?.firstOrNull
{ it.repositoryName == repoName }
 ?: throw RuntimeException("Repository not found: $repoName")

 val tagImageCmd = getDockerClient()?.tagImageCmd("$imageName",
"${repoData.repositoryUri}", imageName)
 if (tagImageCmd != null) {
 tagImageCmd.exec()
 }
 val pushImageCmd =
 repoData.repositoryUri?.let {
 dockerClient?.pushImageCmd(it)
 // ?.withTag("latest")
 ?.withAuthConfig(authConfig)
 }

 try {
 if (pushImageCmd != null) {
 pushImageCmd.start().awaitCompletion()
 }
 println("The $imageName was pushed to Amazon ECR")
 } catch (e: IOException) {
 throw RuntimeException(e)
 }
 }
}

/**
 * Verifies the existence of an image in an Amazon Elastic Container Registry
(Amazon ECR) repository asynchronously.
 *
 * @param repositoryName The name of the Amazon ECR repository.
 * @param imageTag The tag of the image to verify.
 */
suspend fun verifyImage(
 repoName: String?,
```

```
 imageTagVal: String?,
) {
 require(!(repoName == null || repoName.isEmpty())) { "Repository name
cannot be null or empty" }
 require(!(imageTagVal == null || imageTagVal.isEmpty())) { "Image tag
cannot be null or empty" }

 val imageId =
 ImageIdentifier {
 imageTag = imageTagVal
 }
 val request =
 DescribeImagesRequest {
 repositoryName = repoName
 imageIds = listOf(imageId)
 }

 EcrClient.fromEnvironment { region = "us-east-1" }.use { ecrClient ->
 val describeImagesResponse = ecrClient.describeImages(request)
 if (describeImagesResponse != null && !
describeImagesResponse.imageDetails?.isEmpty()!!) {
 println("Image is present in the repository.")
 } else {
 println("Image is not present in the repository.")
 }
 }
 }
}

/**
 * Deletes an ECR (Elastic Container Registry) repository.
 *
 * @param repoName the name of the repository to delete.
 */
suspend fun deleteECRRepository(repoName: String) {
 if (repoName.isNullOrEmpty()) {
 throw IllegalArgumentException("Repository name cannot be null or
empty")
 }

 val repositoryRequest =
 DeleteRepositoryRequest {
 force = true
 repositoryName = repoName
 }
}
```

```
 }

 EcrClient.fromEnvironment { region = "us-east-1" }.use { ecrClient ->
 ecrClient.deleteRepository(repositoryRequest)
 println("You have successfully deleted the $repoName repository")
 }
}

// Return an AuthConfig.
private suspend fun getAuthConfig(repoName: String): AuthConfig {
 EcrClient.fromEnvironment { region = "us-east-1" }.use { ecrClient ->
 // Retrieve the authorization token for ECR.
 val response = ecrClient.getAuthorizationToken()
 val authorizationData = response.authorizationData?.get(0)
 val token = authorizationData?.authorizationToken
 val decodedToken = String(Base64.getDecoder().decode(token))
 val password = decodedToken.substring(4)

 val request =
 DescribeRepositoriesRequest {
 repositoryNames = listOf(repoName)
 }

 val descrRepoResponse = ecrClient.describeRepositories(request)
 val repoData = descrRepoResponse.repositories?.firstOrNull
 { it.repositoryName == repoName }
 val registryURL: String =
 repoData?.repositoryUri?.split("/")?.get(0) ?: ""

 return AuthConfig()
 .withUsername("AWS")
 .withPassword(password)
 .withRegistryAddress(registryURL)
 }
}
}
```

- 如需 API 詳細資訊，請參閱《AWS SDK for Kotlin API 參考》中的下列主題。
  - [CreateRepository](#)
  - [DeleteRepository](#)
  - [DescribeImages](#)

- [DescribeRepositories](#)
- [GetAuthorizationToken](#)
- [GetRepositoryPolicy](#)
- [SetRepositoryPolicy](#)
- [StartLifecyclePolicyPreview](#)

## Python

適用於 Python 的 SDK (Boto3)

### Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

在命令提示中執行互動式案例。

```
class ECRGettingStarted:
 """
 A scenario that demonstrates how to use Boto3 to perform basic operations
 using
 Amazon ECR.
 """

 def __init__(
 self,
 ecr_wrapper: ECRWrapper,
 docker_client: docker.DockerClient,
):
 self.ecr_wrapper = ecr_wrapper
 self.docker_client = docker_client
 self.tag = "echo-text"
 self.repository_name = "ecr-basics"
 self.docker_image = None
 self.full_tag_name = None
 self.repository = None

 def run(self, role_arn: str) -> None:
 """
```

```
Runs the scenario.
```

```
"""
```

```
print(
 """
```

The Amazon Elastic Container Registry (ECR) is a fully-managed Docker container registry

service provided by AWS. It allows developers and organizations to securely store, manage, and deploy Docker container images.

ECR provides a simple and scalable way to manage container images throughout their lifecycle,

from building and testing to production deployment.

The `ECRWrapper` class is a wrapper for the Boto3 `ecr` client. The `ecr` client provides a set of methods to

programmatically interact with the Amazon ECR service. This allows developers to automate the storage, retrieval, and management of container images as part of their application

deployment pipelines. With ECR, teams can focus on building and deploying their applications without having to worry about the underlying infrastructure required to

host and manage a container registry.

This scenario walks you through how to perform key operations for this service.

Let's get started...

```
"""
```

```
)
press_enter_to_continue()
print_dashes()
print(
 f"""
```

\* Create an ECR repository.

An ECR repository is a private Docker container repository provided by Amazon Web Services (AWS). It is a managed service that makes it easy to store, manage, and deploy Docker container images.

```
"""
```

```
)
print(f"Creating a repository named {self.repository_name}")
self.repository =
self.ecr_wrapper.create_repository(self.repository_name)
print(f"The ARN of the ECR repository is
{self.repository['repositoryArn']}")
repository_uri = self.repository["repositoryUri"]
press_enter_to_continue()
```

```
 print_dashes()

 print(
 f"""
* Build a Docker image.

Create a local Docker image if it does not already exist.
A Python Docker client is used to execute Docker commands.
You must have Docker installed and running.
 """
)
 print(f"Building a docker image from 'docker_files/Dockerfile'")
 self.full_tag_name = f"{repository_uri}:{self.tag}"
 self.docker_image = self.docker_client.images.build(
 path="docker_files", tag=self.full_tag_name
)[0]
 print(f"Docker image {self.full_tag_name} successfully built.")
 press_enter_to_continue()
 print_dashes()

 if role_arn is None:
 print(
 """
* Because an IAM role ARN was not provided, a role policy will not be set for
this repository.
 """
)
 else:
 print(
 """
* Set an ECR repository policy.

Setting an ECR repository policy using the `setRepositoryPolicy` function is
crucial for maintaining
the security and integrity of your container images. The repository policy allows
you to
define specific rules and restrictions for accessing and managing the images
stored within your ECR
repository.
 """
)

 self.grant_role_download_access(role_arn)
 print(f"Download access granted to the IAM role ARN {role_arn}")
```

```
 press_enter_to_continue()
 print_dashes()

 print(
 """
* Display ECR repository policy.

Now we will retrieve the ECR policy to ensure it was successfully set.
 """
)

 policy_text =
self.ecr_wrapper.get_repository_policy(self.repository_name)
 print("Policy Text:")
 print(f"{policy_text}")
 press_enter_to_continue()
 print_dashes()

 print(
 """
* Retrieve an ECR authorization token.

You need an authorization token to securely access and interact with the Amazon
 ECR registry.
The `get_authorization_token` method of the `ecr` client is responsible for
 securely accessing
and interacting with an Amazon ECR repository. This operation is responsible for
 obtaining a
valid authorization token, which is required to authenticate your requests to the
 ECR service.

Without a valid authorization token, you would not be able to perform any
 operations on the
ECR repository, such as pushing, pulling, or managing your Docker images.
 """
)

 authorization_token = self.ecr_wrapper.get_authorization_token()
 print("Authorization token retrieved.")
 press_enter_to_continue()
 print_dashes()
 print(
 """
* Get the ECR Repository URI.
```

The URI of an Amazon ECR repository is important. When you want to deploy a container image to a container orchestration platform like Amazon Elastic Kubernetes Service (EKS) or Amazon Elastic Container Service (ECS), you need to specify the full image URI, which includes the ECR repository URI. This allows the container runtime to pull the correct container image from the ECR repository.

```

"""
)
repository_descriptions = self.ecr_wrapper.describe_repositories(
 [self.repository_name]
)
repository_uri = repository_descriptions[0]["repositoryUri"]
print(f"Repository URI found: {repository_uri}")
press_enter_to_continue()
print_dashes()

```

```

print(
 """

```

\* Set an ECR Lifecycle Policy.

An ECR Lifecycle Policy is used to manage the lifecycle of Docker images stored in your ECR repositories.

These policies allow you to automatically remove old or unused Docker images from your repositories, freeing up storage space and reducing costs.

This example policy helps to maintain the size and efficiency of the container registry by automatically removing older and potentially unused images, ensuring that the storage is optimized and the registry remains up-to-date.

```

"""
)
press_enter_to_continue()
self.put_expiration_policy()
print(f"An expiration policy was added to the repository.")
print_dashes()

```

```

print(
 """

```

\* Push a docker image to the Amazon ECR Repository.

The Docker client uses the authorization token is used to authenticate the when pushing the image to the ECR repository.

```
 """
)
 decoded_authorization =
base64.b64decode(authorization_token).decode("utf-8")
 username, password = decoded_authorization.split(":")

 resp = self.docker_client.api.push(
 repository=repository_uri,
 auth_config={"username": username, "password": password},
 tag=self.tag,
 stream=True,
 decode=True,
)
 for line in resp:
 print(line)

 print_dashes()

 print("* Verify if the image is in the ECR Repository.")
 image_descriptions = self.ecr_wrapper.describe_images(
 self.repository_name, [self.tag]
)
 if len(image_descriptions) > 0:
 print("Image found in ECR Repository.")
 else:
 print("Image not found in ECR Repository.")
 press_enter_to_continue()
 print_dashes()

 print(
 """ As an optional step, you can interact with the image in Amazon ECR
 by using the CLI."
)
 if q.ask(
 "Would you like to view instructions on how to use the CLI to run the
 image? (y/n)",
 q.is_yesno,
):
 print(
 f"""
```

1. Authenticate with ECR - Before you can pull the image from Amazon ECR, you need to authenticate with the registry. You can do this using the AWS CLI:

```
aws ecr get-login-password --region us-east-1 | docker login --username AWS
--password-stdin {repository_uri.split("/")}[0]}
```

2. Describe the image using this command:

```
aws ecr describe-images --repository-name {self.repository_name} --image-ids
imageTag={self.tag}
```

3. Run the Docker container and view the output using this command:

```
docker run --rm {self.full_tag_name}
"""
)

 self.cleanup(True)

def cleanup(self, ask: bool):
 """
 Deletes the resources created in this scenario.
 :param ask: If True, prompts the user to confirm before deleting the
resources.
 """
 if self.repository is not None and (
 not ask
 or q.ask(
 f"Would you like to delete the ECR repository
'{self.repository_name}'? (y/n) "
)
):
 print(f"Deleting the ECR repository '{self.repository_name}'.")
 self.ecr_wrapper.delete_repository(self.repository_name)

 if self.full_tag_name is not None and (
 not ask
 or q.ask(
 f"Would you like to delete the local Docker image
'{self.full_tag_name}'? (y/n) "
)
):
 print(f"Deleting the docker image '{self.full_tag_name}'.")
 self.docker_client.images.remove(self.full_tag_name)
```

```
def grant_role_download_access(self, role_arn: str):
 """
 Grants the specified role access to download images from the ECR
 repository.

 :param role_arn: The ARN of the role to grant access to.
 """
 policy_json = {
 "Version": "2012-10-17",
 "Statement": [
 {
 "Sid": "AllowDownload",
 "Effect": "Allow",
 "Principal": {"AWS": role_arn},
 "Action": ["ecr:BatchGetImage"],
 }
],
 }

 self.ecr_wrapper.set_repository_policy(
 self.repository_name, json.dumps(policy_json)
)

def put_expiration_policy(self):
 """
 Puts an expiration policy on the ECR repository.
 """
 policy_json = {
 "rules": [
 {
 "rulePriority": 1,
 "description": "Expire images older than 14 days",
 "selection": {
 "tagStatus": "any",
 "countType": "sinceImagePushed",
 "countUnit": "days",
 "countNumber": 14,
 },
 "action": {"type": "expire"},
 }
]
 }
```

```
 self.ecr_wrapper.put_lifecycle_policy(
 self.repository_name, json.dumps(policy_json)
)

if __name__ == "__main__":
 parser = argparse.ArgumentParser(
 description="Run Amazon ECR getting started scenario."
)
 parser.add_argument(
 "--iam-role-arn",
 type=str,
 default=None,
 help="an optional IAM role ARN that will be granted access to download
images from a repository.",
 required=False,
)
 parser.add_argument(
 "--no-art",
 action="store_true",
 help="accessibility setting that suppresses art in the console output.",
)
 args = parser.parse_args()
 no_art = args.no_art
 iam_role_arn = args.iam_role_arn
 demo = None
 a_docker_client = None
 try:
 a_docker_client = docker.from_env()
 if not a_docker_client.ping():
 raise docker.errors.DockerException("Docker is not running.")
 except docker.errors.DockerException as err:
 logging.error(
 """

The Python Docker client could not be created.
Do you have Docker installed and running?
Here is the error message:
%s
""",
 err,
)
 sys.exit("Error with Docker.")
```

```
try:
 an_ecr_wrapper = ECRWrapper.from_client()
 demo = ECRGettingStarted(an_ecr_wrapper, a_docker_client)
 demo.run(iam_role_arn)

except Exception as exception:
 logging.exception("Something went wrong with the demo!")
 if demo is not None:
 demo.cleanup(False)
```

包裝 Amazon ECR 動作的 ECRWrapper 類別。

```
class ECRWrapper:
 def __init__(self, ecr_client: client):
 self.ecr_client = ecr_client

 @classmethod
 def from_client(cls) -> "ECRWrapper":
 """
 Creates a ECRWrapper instance with a default Amazon ECR client.

 :return: An instance of ECRWrapper initialized with the default Amazon
 ECR client.
 """
 ecr_client = boto3.client("ecr")
 return cls(ecr_client)

 def create_repository(self, repository_name: str) -> dict[str, any]:
 """
 Creates an ECR repository.

 :param repository_name: The name of the repository to create.
 :return: A dictionary of the created repository.
 """
 try:
 response =
self.ecr_client.create_repository(repositoryName=repository_name)
 return response["repository"]
 except ClientError as err:
```

```
 if err.response["Error"]["Code"] ==
"RepositoryAlreadyExistsException":
 print(f"Repository {repository_name} already exists.")
 response = self.ecr_client.describe_repositories(
 repositoryNames=[repository_name]
)
 return self.describe_repositories([repository_name])[0]
 else:
 logger.error(
 "Error creating repository %s. Here's why %s",
 repository_name,
 err.response["Error"]["Message"],
)
 raise

def delete_repository(self, repository_name: str):
 """
 Deletes an ECR repository.

 :param repository_name: The name of the repository to delete.
 """
 try:
 self.ecr_client.delete_repository(
 repositoryName=repository_name, force=True
)
 print(f"Deleted repository {repository_name}.")
 except ClientError as err:
 logger.error(
 "Couldn't delete repository %s.. Here's why %s",
 repository_name,
 err.response["Error"]["Message"],
)
 raise

def set_repository_policy(self, repository_name: str, policy_text: str):
 """
 Sets the policy for an ECR repository.

 :param repository_name: The name of the repository to set the policy for.
 :param policy_text: The policy text to set.
 """
 try:
```

```
 self.ecr_client.set_repository_policy(
 repositoryName=repository_name, policyText=policy_text
)
 print(f"Set repository policy for repository {repository_name}.")
 except ClientError as err:
 if err.response["Error"]["Code"] ==
"RepositoryPolicyNotFoundException":
 logger.error("Repository does not exist. %s.", repository_name)
 raise
 else:
 logger.error(
 "Couldn't set repository policy for repository %s. Here's why
%s",
 repository_name,
 err.response["Error"]["Message"],
)
 raise

def get_repository_policy(self, repository_name: str) -> str:
 """
 Gets the policy for an ECR repository.

 :param repository_name: The name of the repository to get the policy for.
 :return: The policy text.
 """
 try:
 response = self.ecr_client.get_repository_policy(
 repositoryName=repository_name
)
 return response["policyText"]
 except ClientError as err:
 if err.response["Error"]["Code"] ==
"RepositoryPolicyNotFoundException":
 logger.error("Repository does not exist. %s.", repository_name)
 raise
 else:
 logger.error(
 "Couldn't get repository policy for repository %s. Here's why
%s",
 repository_name,
 err.response["Error"]["Message"],
)
 raise
```

```
def get_authorization_token(self) -> str:
 """
 Gets an authorization token for an ECR repository.

 :return: The authorization token.
 """
 try:
 response = self.ecr_client.get_authorization_token()
 return response["authorizationData"][0]["authorizationToken"]
 except ClientError as err:
 logger.error(
 "Couldn't get authorization token. Here's why %s",
 err.response["Error"]["Message"],
)
 raise

def describe_repositories(self, repository_names: list[str]) -> list[dict]:
 """
 Describes ECR repositories.

 :param repository_names: The names of the repositories to describe.
 :return: The list of repository descriptions.
 """
 try:
 response = self.ecr_client.describe_repositories(
 repositoryNames=repository_names
)
 return response["repositories"]
 except ClientError as err:
 logger.error(
 "Couldn't describe repositories. Here's why %s",
 err.response["Error"]["Message"],
)
 raise

def put_lifecycle_policy(self, repository_name: str, lifecycle_policy_text:
str):
 """
 Puts a lifecycle policy for an ECR repository.
```

```
 :param repository_name: The name of the repository to put the lifecycle
policy for.
 :param lifecycle_policy_text: The lifecycle policy text to put.
 """
 try:
 self.ecr_client.put_lifecycle_policy(
 repositoryName=repository_name,
 lifecyclePolicyText=lifecycle_policy_text,
)
 print(f"Put lifecycle policy for repository {repository_name}.")
 except ClientError as err:
 logger.error(
 "Couldn't put lifecycle policy for repository %s. Here's why %s",
 repository_name,
 err.response["Error"]["Message"],
)
 raise

def describe_images(
 self, repository_name: str, image_ids: list[str] = None
) -> list[dict]:
 """
 Describes ECR images.

 :param repository_name: The name of the repository to describe images
for.
 :param image_ids: The optional IDs of images to describe.
 :return: The list of image descriptions.
 """
 try:
 params = {
 "repositoryName": repository_name,
 }
 if image_ids is not None:
 params["imageIds"] = [{"imageTag": tag} for tag in image_ids]

 paginator = self.ecr_client.get_paginator("describe_images")
 image_descriptions = []
 for page in paginator.paginate(**params):
 image_descriptions.extend(page["imageDetails"])
 return image_descriptions
 except ClientError as err:
 logger.error(
```

```
 "Couldn't describe images. Here's why %s",
 err.response["Error"]["Message"],
)
 raise
```

- 如需 API 詳細資訊，請參閱《適用於 Python (Boto3) 的 AWS SDK API 參考》中的下列主題。
  - [CreateRepository](#)
  - [DeleteRepository](#)
  - [DescribeImages](#)
  - [DescribeRepositories](#)
  - [GetAuthorizationToken](#)
  - [GetRepositoryPolicy](#)
  - [SetRepositoryPolicy](#)
  - [StartLifecyclePolicyPreview](#)

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱 [搭配 SDK 使用 Amazon ECR AWS](#)。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

## 使用 AWS SDKs 的 Amazon ECR 動作

下列程式碼範例示範如何使用 AWS SDKs 執行個別 Amazon ECR 動作。每個範例均包含 GitHub 的連結，您可以在連結中找到設定和執行程式碼的相關說明。

這些摘錄會呼叫 Amazon ECR API，是必須在內容中執行之大型程式的程式碼摘錄。您可以在 [使用 AWS SDKs Amazon ECR 案例](#) 中查看內容中的動作。

下列範例僅包含最常使用的動作。如需完整清單，請參閱《[Amazon Elastic Container Registry API 參考](#)》。

### 範例

- [CreateRepository 搭配 AWS SDK 或 CLI 使用](#)
- [DeleteRepository 搭配 AWS SDK 或 CLI 使用](#)

- [DescribeImages 搭配 AWS SDK 或 CLI 使用](#)
- [DescribeRepositories 搭配 AWS SDK 或 CLI 使用](#)
- [GetAuthorizationToken 搭配 AWS SDK 或 CLI 使用](#)
- [GetRepositoryPolicy 搭配 AWS SDK 或 CLI 使用](#)
- [ListImages 搭配 AWS SDK 或 CLI 使用](#)
- [PushImageCmd 搭配 AWS SDK 使用](#)
- [PutLifecyclePolicy 搭配 AWS SDK 或 CLI 使用](#)
- [SetRepositoryPolicy 搭配 AWS SDK 或 CLI 使用](#)
- [StartLifecyclePolicyPreview 搭配 AWS SDK 或 CLI 使用](#)

## CreateRepository 搭配 AWS SDK 或 CLI 使用

下列程式碼範例示範如何使用 CreateRepository。

動作範例是大型程式的程式碼摘錄，必須在內容中執行。您可以在下列程式碼範例的內容中看到此動作：

- [了解基本概念](#)
- [Amazon ECR 入門](#)

### CLI

#### AWS CLI

##### 範例 1：建立儲存庫

下列 create-repository 範例會在帳戶預設登錄檔中指定的命名空間內建立儲存庫。

```
aws ecr create-repository \
 --repository-name project-a/sample-repo
```

輸出：

```
{
 "repository": {
 "registryId": "123456789012",
 "repositoryName": "project-a/sample-repo",
```

```
 "repositoryArn": "arn:aws:ecr:us-west-2:123456789012:repository/project-
a/sample-repo"
 }
}
```

如需詳細資訊，請參閱《Amazon ECR 使用者指南》中的[建立儲存庫](#)。

### 範例 2：建立以映像標籤不變性設定的儲存庫

下列 `create-repository` 範例會在帳戶預設登錄檔中，建立針對標籤不變性而設定的儲存庫。

```
aws ecr create-repository \
 --repository-name project-a/sample-repo \
 --image-tag-mutability IMMUTABLE
```

輸出：

```
{
 "repository": {
 "registryId": "123456789012",
 "repositoryName": "project-a/sample-repo",
 "repositoryArn": "arn:aws:ecr:us-west-2:123456789012:repository/project-
a/sample-repo",
 "imageTagMutability": "IMMUTABLE"
 }
}
```

如需詳細資訊，請參閱《Amazon ECR 使用者指南》中的[映像標籤可變性](#)。

### 範例 3：建立以掃描組態設定的儲存庫

下列 `create-repository` 範例會建立儲存庫，其設定為在帳戶預設登錄檔中的映像推送時，執行漏洞掃描。

```
aws ecr create-repository \
 --repository-name project-a/sample-repo \
 --image-scanning-configuration scanOnPush=true
```

輸出：

```
{
```

```
"repository": {
 "registryId": "123456789012",
 "repositoryName": "project-a/sample-repo",
 "repositoryArn": "arn:aws:ecr:us-west-2:123456789012:repository/project-
a/sample-repo",
 "imageScanningConfiguration": {
 "scanOnPush": true
 }
}
```

如需詳細資訊，請參閱《Amazon ECR 使用者指南》中的[映像掃描](#)。

- 如需 API 詳細資訊，請參閱《AWS CLI 命令參考》中的[CreateRepository](#)。

## Java

### SDK for Java 2.x

#### Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
/**
 * Creates an Amazon Elastic Container Registry (Amazon ECR) repository.
 *
 * @param repoName the name of the repository to create.
 * @return the Amazon Resource Name (ARN) of the created repository, or an
empty string if the operation failed.
 * @throws IllegalArgumentException If repository name is invalid.
 * @throws RuntimeException if an error occurs while creating the
repository.
 */
public String createECRRepository(String repoName) {
 if (repoName == null || repoName.isEmpty()) {
 throw new IllegalArgumentException("Repository name cannot be null or
empty");
 }

 CreateRepositoryRequest request = CreateRepositoryRequest.builder()
```


```
 .repositoryName(repoName)
 .build();

 CompletableFuture<CreateRepositoryResponse> response =
getAsyncClient().createRepository(request);
 try {
 CreateRepositoryResponse result = response.join();
 if (result != null) {
 System.out.println("The " + repoName + " repository was created
successfully.");
 return result.repository().repositoryArn();
 } else {
 throw new RuntimeException("Unexpected response type");
 }
 } catch (CompletionException e) {
 Throwable cause = e.getCause();
 if (cause instanceof EcrException ex) {
 if
("RepositoryAlreadyExistsException".equals(ex.awsErrorDetails().errorCode())) {
 System.out.println("The Amazon ECR repository already exists,
moving on...");
 DescribeRepositoriesRequest describeRequest =
DescribeRepositoriesRequest.builder()
 .repositoryNames(repoName)
 .build();
 DescribeRepositoriesResponse describeResponse =
getAsyncClient().describeRepositories(describeRequest).join();
 return
describeResponse.repositories().get(0).repositoryArn();
 } else {
 throw new RuntimeException(ex);
 }
 } else {
 throw new RuntimeException(e);
 }
 }
}
```

- 如需 API 詳細資訊，請參閱《AWS SDK for Java 2.x API 參考》中的 [CreateRepository](#)。

## Kotlin

## 適用於 Kotlin 的 SDK

 Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
/**
 * Creates an Amazon Elastic Container Registry (Amazon ECR) repository.
 *
 * @param repoName the name of the repository to create.
 * @return the Amazon Resource Name (ARN) of the created repository, or an
 * empty string if the operation failed.
 * @throws RepositoryAlreadyExistsException if the repository exists.
 * @throws EcrException if an error occurs while creating the
 * repository.
 */
suspend fun createECRRepository(repoName: String?): String? {
 val request =
 CreateRepositoryRequest {
 repositoryName = repoName
 }

 return try {
 EcrClient.fromEnvironment { region = "us-east-1" }.use { ecrClient ->
 val response = ecrClient.createRepository(request)
 response.repository?.repositoryArn
 }
 } catch (e: RepositoryAlreadyExistsException) {
 println("Repository already exists: $repoName")
 repoName?.let { getRepoARN(it) }
 } catch (e: EcrException) {
 println("An error occurred: ${e.message}")
 null
 }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的 AWS SDK API 參考》中的 [CreateRepository](#)。

## Python

### 適用於 Python 的 SDK (Boto3)

#### Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
class ECRWrapper:
 def __init__(self, ecr_client: client):
 self.ecr_client = ecr_client

 @classmethod
 def from_client(cls) -> "ECRWrapper":
 """
 Creates a ECRWrapper instance with a default Amazon ECR client.

 :return: An instance of ECRWrapper initialized with the default Amazon
 ECR client.
 """
 ecr_client = boto3.client("ecr")
 return cls(ecr_client)

 def create_repository(self, repository_name: str) -> dict[str, any]:
 """
 Creates an ECR repository.

 :param repository_name: The name of the repository to create.
 :return: A dictionary of the created repository.
 """
 try:
 response =
self.ecr_client.create_repository(repositoryName=repository_name)
 return response["repository"]
 except ClientError as err:
```

```

 if err.response["Error"]["Code"] ==
"RepositoryAlreadyExistsException":
 print(f"Repository {repository_name} already exists.")
 response = self.ecr_client.describe_repositories(
 repositoryNames=[repository_name]
)
 return self.describe_repositories([repository_name])[0]
 else:
 logger.error(
 "Error creating repository %s. Here's why %s",
 repository_name,
 err.response["Error"]["Message"],
)
 raise

```

- 如需 API 詳細資訊，請參閱《AWS SDK for Python (Boto3) API 參考》中的 [CreateRepository](#)。

## SAP ABAP

### 適用於 SAP ABAP 的開發套件

#### Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```

TRY.
 " iv_repository_name = 'my-repository'
 oo_result = lo_ecr->createrepository(
 iv_repositoryname = iv_repository_name).
 DATA(lv_repository_uri) = oo_result->get_repository()-
>get_repositoryuri().
 MESSAGE |Repository created with URI: { lv_repository_uri }| TYPE 'I'.
CATCH /aws1/cx_ecrrepositoryalrexex.
 " If repository already exists, retrieve it
 DATA lt_repo_names TYPE /aws1/
cl_ecrrepositorynamels00=>tt_repositorynamelist.

```

```
APPEND NEW /aws1/cl_ecrrepositorynames00(iv_value =
iv_repository_name) TO lt_repo_names.
DATA(lo_describe_result) = lo_ecr-
>describerepositories(it_repositorynames = lt_repo_names).
DATA(lt_repos) = lo_describe_result->get_repositories().
IF lines(lt_repos) > 0.
 READ TABLE lt_repos INDEX 1 INTO DATA(lo_repo).
 oo_result = NEW /aws1/cl_ecrcrrepositoryrsp(io_repository =
lo_repo).
 MESSAGE |Repository { iv_repository_name } already exists.| TYPE 'I'.
ENDIF.
ENDTRY.
```

- 如需 API 詳細資訊，請參閱《適用於 AWS SAP ABAP 的 SDK API 參考》中的 [CreateRepository](#)。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱 [搭配 SDK 使用 Amazon ECR AWS](#)。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

## DeleteRepository 搭配 AWS SDK 或 CLI 使用

下列程式碼範例示範如何使用 DeleteRepository。

動作範例是大型程式的程式碼摘錄，必須在內容中執行。您可以在下列程式碼範例的內容中看到此動作：

- [了解基本概念](#)
- [Amazon ECR 入門](#)

### CLI

#### AWS CLI

##### 刪除儲存庫

下列 delete-repository 範例命令會強制刪除帳戶預設登錄檔中指定的儲存庫。如果儲存庫包含映像，則需要 --force 旗標。

```
aws ecr delete-repository \
 --repository-name ubuntu \
```

```
--force
```

輸出：

```
{
 "repository": {
 "registryId": "123456789012",
 "repositoryName": "ubuntu",
 "repositoryArn": "arn:aws:ecr:us-west-2:123456789012:repository/ubuntu"
 }
}
```

如需詳細資訊，請參閱《Amazon ECR 使用者指南》中的[刪除儲存庫](#)。

- 如需 API 詳細資訊，請參閱《AWS CLI 命令參考》中的 [DeleteRepository](#)。

## Java

### SDK for Java 2.x

#### Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
/**
 * Deletes an ECR (Elastic Container Registry) repository.
 *
 * @param repoName the name of the repository to delete.
 * @throws IllegalArgumentException if the repository name is null or empty.
 * @throws EcrException if there is an error deleting the repository.
 * @throws RuntimeException if an unexpected error occurs during the deletion
 process.
 */
public void deleteECRRepository(String repoName) {
 if (repoName == null || repoName.isEmpty()) {
 throw new IllegalArgumentException("Repository name cannot be null or
empty");
 }
}
```

```
 DeleteRepositoryRequest repositoryRequest =
DeleteRepositoryRequest.builder()
 .force(true)
 .repositoryName(repoName)
 .build();

 CompletableFuture<DeleteRepositoryResponse> response =
getAsyncClient().deleteRepository(repositoryRequest);
 response.whenComplete((deleteRepositoryResponse, ex) -> {
 if (deleteRepositoryResponse != null) {
 System.out.println("You have successfully deleted the " +
repoName + " repository");
 } else {
 Throwable cause = ex.getCause();
 if (cause instanceof EcrException) {
 throw (EcrException) cause;
 } else {
 throw new RuntimeException("Unexpected error: " +
cause.getMessage(), cause);
 }
 }
 });

 // Wait for the CompletableFuture to complete
 response.join();
 }
}
```

- 如需 API 詳細資訊，請參閱《AWS SDK for Java 2.x API 參考》中的 [DeleteRepository](#)。

## Kotlin

### 適用於 Kotlin 的 SDK

#### Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
/**
```

```

 * Deletes an ECR (Elastic Container Registry) repository.
 *
 * @param repoName the name of the repository to delete.
 */
suspend fun deleteECRRepository(repoName: String) {
 if (repoName.isNullOrEmpty()) {
 throw IllegalArgumentException("Repository name cannot be null or
empty")
 }

 val repositoryRequest =
 DeleteRepositoryRequest {
 force = true
 repositoryName = repoName
 }

 EcrClient.fromEnvironment { region = "us-east-1" }.use { ecrClient ->
 ecrClient.deleteRepository(repositoryRequest)
 println("You have successfully deleted the $repoName repository")
 }
}

```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的 AWS SDK API 參考》中的 [DeleteRepository](#)。

## Python

### 適用於 Python 的 SDK (Boto3)

#### Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```

class ECRWrapper:
 def __init__(self, ecr_client: client):
 self.ecr_client = ecr_client

 @classmethod
 def from_client(cls) -> "ECRWrapper":
 """

```

```
Creates a ECRWrapper instance with a default Amazon ECR client.

:return: An instance of ECRWrapper initialized with the default Amazon
ECR client.
"""
 ecr_client = boto3.client("ecr")
 return cls(ecr_client)

def delete_repository(self, repository_name: str):
 """
 Deletes an ECR repository.

 :param repository_name: The name of the repository to delete.
 """
 try:
 self.ecr_client.delete_repository(
 repositoryName=repository_name, force=True
)
 print(f"Deleted repository {repository_name}.")
 except ClientError as err:
 logger.error(
 "Couldn't delete repository %s.. Here's why %s",
 repository_name,
 err.response["Error"]["Message"],
)
 raise
```

- 如需 API 詳細資訊，請參閱《AWS SDK for Python (Boto3) API 參考》中的 [DeleteRepository](#)。

## SAP ABAP

適用於 SAP ABAP 的開發套件

### Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
TRY.
 " iv_repository_name = 'my-repository'
 lo_ecr->deleterepository(
 iv_repositoryname = iv_repository_name
 iv_force = abap_true).
 MESSAGE |Repository { iv_repository_name } deleted.| TYPE 'I'.
CATCH /aws1/cx_ecrrepositorynotfndex.
 MESSAGE 'Repository not found.' TYPE 'I'.
ENDTRY.
```

- 如需 API 詳細資訊，請參閱《適用於 AWS SAP ABAP 的 SDK API 參考》中的 [DeleteRepository](#)。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱 [搭配 SDK 使用 Amazon ECR AWS](#)。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

## DescribeImages 搭配 AWS SDK 或 CLI 使用

下列程式碼範例示範如何使用 DescribeImages。

動作範例是大型程式的程式碼摘錄，必須在內容中執行。您可以在下列程式碼範例的內容中看到此動作：

- [了解基本概念](#)

### CLI

#### AWS CLI

描述儲存庫中的映像

下列 describe-images 範例顯示 儲存 cluster-autoscaler 庫中具有標籤 之映像的詳細資訊 v1.13.6。

```
aws ecr describe-images \
 --repository-name cluster-autoscaler \
 --image-ids imageTag=v1.13.6
```

輸出：

```
{
 "imageDetails": [
 {
 "registryId": "012345678910",
 "repositoryName": "cluster-autoscaler",
 "imageDigest":
"sha256:4a1c6567c38904384ebc64e35b7eeddd8451110c299e3368d2210066487d97e5",
 "imageTags": [
 "v1.13.6"
],
 "imageSizeInBytes": 48318255,
 "imagePushedAt": 1565128275.0
 }
]
}
```

- 如需 API 詳細資訊，請參閱《AWS CLI 命令參考》中的 [DescribeImages](#)。

## Java

### SDK for Java 2.x

#### Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
/**
 * Verifies the existence of an image in an Amazon Elastic Container Registry
 (Amazon ECR) repository asynchronously.
 *
 * @param repositoryName The name of the Amazon ECR repository.
 * @param imageTag The tag of the image to verify.
 * @throws EcrException if there is an error retrieving the image
 information from Amazon ECR.
 * @throws CompletionException if the asynchronous operation completes
 exceptionally.
 */
public void verifyImage(String repositoryName, String imageTag) {
 DescribeImagesRequest request = DescribeImagesRequest.builder()
```

```
.repositoryName(repositoryName)
.imageIds(ImageIdentifier.builder().imageTag(imageTag).build())
.build();


CompletableFuture<DescribeImagesResponse> response =
getAsyncClient().describeImages(request);
response.whenComplete((describeImagesResponse, ex) -> {
 if (ex != null) {
 if (ex instanceof CompletionException) {
 Throwable cause = ex.getCause();
 if (cause instanceof EcrException) {
 throw (EcrException) cause;
 } else {
 throw new RuntimeException("Unexpected error: " +
cause.getMessage(), cause);
 }
 } else {
 throw new RuntimeException("Unexpected error: " +
ex.getCause());
 }
 } else if (describeImagesResponse != null && !
describeImagesResponse.imageDetails().isEmpty()) {
 System.out.println("Image is present in the repository.");
 } else {
 System.out.println("Image is not present in the repository.");
 }
});

// Wait for the CompletableFuture to complete.
response.join();
}
```

- 如需 API 詳細資訊，請參閱《AWS SDK for Java 2.x API 參考》中的 [DescribeImages](#)。

## Kotlin

## 適用於 Kotlin 的 SDK

 Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
/**
 * Verifies the existence of an image in an Amazon Elastic Container Registry
 (Amazon ECR) repository asynchronously.
 *
 * @param repositoryName The name of the Amazon ECR repository.
 * @param imageTag The tag of the image to verify.
 */
suspend fun verifyImage(
 repoName: String?,
 imageTagVal: String?,
) {
 require(!(repoName == null || repoName.isEmpty())) { "Repository name
cannot be null or empty" }
 require(!(imageTagVal == null || imageTagVal.isEmpty())) { "Image tag
cannot be null or empty" }

 val imageId =
 ImageIdentifier {
 imageTag = imageTagVal
 }
 val request =
 DescribeImagesRequest {
 repositoryName = repoName
 imageIds = listOf(imageId)
 }

 EcrClient.fromEnvironment { region = "us-east-1" }.use { ecrClient ->
 val describeImagesResponse = ecrClient.describeImages(request)
 if (describeImagesResponse != null && !
describeImagesResponse.imageDetails?.isEmpty()!!) {
 println("Image is present in the repository.")
 }
 }
}
```

```
 } else {
 println("Image is not present in the repository.")
 }
 }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的 AWS SDK API 參考》中的 [DescribeImages](#)。

## Python

### 適用於 Python 的 SDK (Boto3)

#### Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
class ECRWrapper:
 def __init__(self, ecr_client: client):
 self.ecr_client = ecr_client

 @classmethod
 def from_client(cls) -> "ECRWrapper":
 """
 Creates a ECRWrapper instance with a default Amazon ECR client.

 :return: An instance of ECRWrapper initialized with the default Amazon
 ECR client.
 """
 ecr_client = boto3.client("ecr")
 return cls(ecr_client)

 def describe_images(
 self, repository_name: str, image_ids: list[str] = None
) -> list[dict]:
 """
 Describes ECR images.
```

```
for.
 :param repository_name: The name of the repository to describe images
 :param image_ids: The optional IDs of images to describe.
 :return: The list of image descriptions.
 """
 try:
 params = {
 "repositoryName": repository_name,
 }
 if image_ids is not None:
 params["imageIds"] = [{"imageTag": tag} for tag in image_ids]

 paginator = self.ecr_client.get_paginator("describe_images")
 image_descriptions = []
 for page in paginator.paginate(**params):
 image_descriptions.extend(page["imageDetails"])
 return image_descriptions
 except ClientError as err:
 logger.error(
 "Couldn't describe images. Here's why %s",
 err.response["Error"]["Message"],
)
 raise
```

- 如需 API 詳細資訊，請參閱《適用於 Python (Boto3) 的 AWS SDK API 參考》中的 [DescribeImages](#)。

## SAP ABAP

### 適用於 SAP ABAP 的開發套件

#### Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

TRY.

```
" iv_repository_name = 'my-repository'
```

```
 " it_image_ids = VALUE #((NEW /aws1/cl_ecrimageidentifier(iv_imagetag
= 'latest')))
 IF it_image_ids IS NOT INITIAL.
 oo_result = lo_ecr->describeimages(
 iv_repositoryname = iv_repository_name
 it_imageids = it_image_ids).
 ELSE.
 oo_result = lo_ecr->describeimages(
 iv_repositoryname = iv_repository_name).
 ENDIF.
 DATA(lt_image_details) = oo_result->get_imagedetails().
 MESSAGE |Found { lines(lt_image_details) } images in repository.| TYPE
'I'.
 CATCH /aws1/cx_ecrrepositorynotfound.
 MESSAGE 'Repository not found.' TYPE 'I'.
 CATCH /aws1/cx_ecrimagenotfound.
 MESSAGE 'Image not found.' TYPE 'I'.
 CATCH /aws1/cx_ecrinvalidparameter.
 MESSAGE 'Invalid parameter provided.' TYPE 'I'.
 ENDTRY.
```

- 如需 API 詳細資訊，請參閱《適用於 AWS SAP ABAP 的 SDK API 參考》中的 [DescribeImages](#)。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱 [搭配 SDK 使用 Amazon ECR AWS](#)。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

## DescribeRepositories 搭配 AWS SDK 或 CLI 使用

下列程式碼範例示範如何使用 DescribeRepositories。

動作範例是大型程式的程式碼摘錄，必須在內容中執行。您可以在下列程式碼範例的內容中看到此動作：

- [了解基本概念](#)

### CLI

#### AWS CLI

##### 描述登錄檔中的儲存庫

此範例描述帳戶預設登錄檔中的儲存庫。

命令：

```
aws ecr describe-repositories
```

輸出：

```
{
 "repositories": [
 {
 "registryId": "012345678910",
 "repositoryName": "ubuntu",
 "repositoryArn": "arn:aws:ecr:us-west-2:012345678910:repository/
ubuntu"
 },
 {
 "registryId": "012345678910",
 "repositoryName": "test",
 "repositoryArn": "arn:aws:ecr:us-west-2:012345678910:repository/test"
 }
]
}
```

- 如需 API 詳細資訊，請參閱《AWS CLI 命令參考》中的 [DescribeRepositories](#)。

## Java

### SDK for Java 2.x

#### Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
/**
 * Retrieves the repository URI for the specified repository name.
 *
 * @param repoName the name of the repository to retrieve the URI for.
 * @return the repository URI for the specified repository name.
```

```
 * @throws EcrException if there is an error retrieving the repository
information.
 * @throws CompletionException if the asynchronous operation completes
exceptionally.
 */
 public void getRepositoryURI(String repoName) {
 DescribeRepositoriesRequest request =
DescribeRepositoriesRequest.builder()
 .repositoryNames(repoName)
 .build();

 CompletableFuture<DescribeRepositoriesResponse> response =
getAsyncClient().describeRepositories(request);
 response.whenComplete((describeRepositoriesResponse, ex) -> {
 if (ex != null) {
 Throwable cause = ex.getCause();
 if (cause instanceof InterruptedException) {
 Thread.currentThread().interrupt();
 String errorMessage = "Thread interrupted while waiting for
asynchronous operation: " + cause.getMessage();
 throw new RuntimeException(errorMessage, cause);
 } else if (cause instanceof EcrException) {
 throw (EcrException) cause;
 } else {
 String errorMessage = "Unexpected error: " +
cause.getMessage();
 throw new RuntimeException(errorMessage, cause);
 }
 } else {
 if (describeRepositoriesResponse != null) {
 if (!describeRepositoriesResponse.repositories().isEmpty()) {
 String repositoryUri =
describeRepositoriesResponse.repositories().get(0).repositoryUri();
 System.out.println("Repository URI found: " +
repositoryUri);
 } else {
 System.out.println("No repositories found for the given
name.");
 }
 } else {
 System.err.println("No response received from
describeRepositories.");
 }
 }
 })
 }
}
```

```

 });
 response.join();
}

```

- 如需 API 詳細資訊，請參閱《AWS SDK for Java 2.x API 參考》中的 [DescribeRepositories](#)。

## Kotlin

### 適用於 Kotlin 的 SDK

#### Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```

/**
 * Retrieves the repository URI for the specified repository name.
 *
 * @param repoName the name of the repository to retrieve the URI for.
 * @return the repository URI for the specified repository name.
 */
suspend fun getRepositoryURI(repoName: String?): String? {
 require(!(repoName == null || repoName.isEmpty())) { "Repository name
cannot be null or empty" }
 val request =
 DescribeRepositoriesRequest {
 repositoryNames = listOf(repoName)
 }

 EcrClient.fromEnvironment { region = "us-east-1" }.use { ecrClient ->
 val describeRepositoriesResponse =
 ecrClient.describeRepositories(request)
 if (!describeRepositoriesResponse.repositories?.isEmpty()!!) {
 return
 describeRepositoriesResponse?.repositories?.get(0)?.repositoryUri
 } else {
 println("No repositories found for the given name.")
 }
 }
}

```

```
 return ""
 }
}
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的 AWS SDK API 參考》中的 [DescribeRepositories](#)。

## Python

### 適用於 Python 的 SDK (Boto3)

#### Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
class ECRWrapper:
 def __init__(self, ecr_client: client):
 self.ecr_client = ecr_client

 @classmethod
 def from_client(cls) -> "ECRWrapper":
 """
 Creates a ECRWrapper instance with a default Amazon ECR client.

 :return: An instance of ECRWrapper initialized with the default Amazon
 ECR client.
 """
 ecr_client = boto3.client("ecr")
 return cls(ecr_client)

 def describe_repositories(self, repository_names: list[str]) -> list[dict]:
 """
 Describes ECR repositories.

 :param repository_names: The names of the repositories to describe.
 :return: The list of repository descriptions.
```

```
"""
try:
 response = self.ecr_client.describe_repositories(
 repositoryNames=repository_names
)
 return response["repositories"]
except ClientError as err:
 logger.error(
 "Couldn't describe repositories. Here's why %s",
 err.response["Error"]["Message"],
)
 raise
```

- 如需 API 詳細資訊，請參閱《AWS SDK for Python (Boto3) API 參考》中的 [DescribeRepositories](#)。

## Rust

### 適用於 Rust 的 SDK

#### Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
async fn show_repos(client: &aws_sdk_ecr::Client) -> Result<(),
aws_sdk_ecr::Error> {
 let rsp = client.describe_repositories().send().await?;

 let repos = rsp.repositories();

 println!("Found {} repositories:", repos.len());

 for repo in repos {
 println!(" ARN: {}", repo.repository_arn().unwrap());
 println!(" Name: {}", repo.repository_name().unwrap());
 }
}
```

```
 Ok(())
}
```

- 如需 API 詳細資訊，請參閱《AWS SDK for Rust API 參考》中的 [DescribeRepositories](#)。

## SAP ABAP

### 適用於 SAP ABAP 的開發套件

#### Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
TRY.
 " it_repository_names = VALUE #((NEW /aws1/
 cl_ecrrepositorynames00(iv_value = 'my-repository')))
 oo_result = lo_ecr->describerepositories(
 it_repositorynames = it_repository_names).
 DATA(lt_repositories) = oo_result->get_repositories().
 MESSAGE |Found { lines(lt_repositories) } repositories.| TYPE 'I'.
CATCH /aws1/cx_ecrrepositorynotfound.
 MESSAGE 'Repository not found.' TYPE 'I'.
ENDTRY.
```

- 如需 API 詳細資訊，請參閱《適用於 AWS SAP ABAP 的 SDK API 參考》中的 [DescribeRepositories](#)。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱 [搭配 SDK 使用 Amazon ECR AWS](#)。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

## GetAuthorizationToken 搭配 AWS SDK 或 CLI 使用

下列程式碼範例示範如何使用 GetAuthorizationToken。

動作範例是大型程式的程式碼摘錄，必須在內容中執行。您可以在下列程式碼範例的內容中看到此動作：

- [了解基本概念](#)

## CLI

### AWS CLI

取得預設登錄檔的授權字符

下列 `get-authorization-token` 範例命令會取得預設登錄檔的授權字符。

```
aws ecr get-authorization-token
```

輸出：

```
{
 "authorizationData": [
 {
 "authorizationToken": "QVdT0kN...",
 "expiresAt": 1448875853.241,
 "proxyEndpoint": "https://123456789012.dkr.ecr.us-
west-2.amazonaws.com"
 }
]
}
```

- 如需 API 詳細資訊，請參閱《AWS CLI 命令參考》中的 [GetAuthorizationToken](#)。

## Java

### SDK for Java 2.x

#### Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
/**
 * Retrieves the authorization token for Amazon Elastic Container Registry
 * (ECR).
```

```
* This method makes an asynchronous call to the ECR client to retrieve the
authorization token.
* If the operation is successful, the method prints the token to the
console.
* If an exception occurs, the method handles the exception and prints the
error message.
*
* @throws EcrException if there is an error retrieving the authorization
token from ECR.
* @throws RuntimeException if there is an unexpected error during the
operation.
*/
public void getAuthToken() {
 CompletableFuture<GetAuthorizationTokenResponse> response =
getAsyncClient().getAuthorizationToken();
 response.whenComplete((authorizationTokenResponse, ex) -> {
 if (authorizationTokenResponse != null) {
 AuthorizationData authorizationData =
authorizationTokenResponse.authorizationData().get(0);
 String token = authorizationData.authorizationToken();
 if (!token.isEmpty()) {
 System.out.println("The token was successfully retrieved.");
 }
 } else {
 if (ex.getCause() instanceof EcrException) {
 throw (EcrException) ex.getCause();
 } else {
 String errorMessage = "Unexpected error occurred: " +
ex.getMessage();
 throw new RuntimeException(errorMessage, ex); // Rethrow the
exception
 }
 }
 });
 response.join();
}
```

- 如需 API 詳細資訊，請參閱《AWS SDK for Java 2.x API 參考》中的 [GetAuthorizationToken](#)。

## Kotlin

### 適用於 Kotlin 的 SDK

#### Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
/**
 * Retrieves the authorization token for Amazon Elastic Container Registry
 * (ECR).
 */
suspend fun getAuthToken() {
 EcrClient.fromEnvironment { region = "us-east-1" }.use { ecrClient ->
 // Retrieve the authorization token for ECR.
 val response = ecrClient.getAuthorizationToken()
 val authorizationData = response.authorizationData?.get(0)
 val token = authorizationData?.authorizationToken
 if (token != null) {
 println("The token was successfully retrieved.")
 }
 }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的 AWS SDK API 參考》中的 [GetAuthorizationToken](#)。

## Python

### 適用於 Python 的 SDK (Boto3)

#### Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
class ECRWrapper:
 def __init__(self, ecr_client: client):
 self.ecr_client = ecr_client

 @classmethod
 def from_client(cls) -> "ECRWrapper":
 """
 Creates a ECRWrapper instance with a default Amazon ECR client.

 :return: An instance of ECRWrapper initialized with the default Amazon
 ECR client.
 """
 ecr_client = boto3.client("ecr")
 return cls(ecr_client)

 def get_authorization_token(self) -> str:
 """
 Gets an authorization token for an ECR repository.

 :return: The authorization token.
 """
 try:
 response = self.ecr_client.get_authorization_token()
 return response["authorizationData"][0]["authorizationToken"]
 except ClientError as err:
 logger.error(
 "Couldn't get authorization token. Here's why %s",
 err.response["Error"]["Message"],
)
 raise
```

- 如需 API 詳細資訊，請參閱《AWS SDK for Python (Boto3) API 參考》中的 [GetAuthorizationToken](#)。

## SAP ABAP

### 適用於 SAP ABAP 的開發套件

#### Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
TRY.
 oo_result = lo_ecr->getauthorizationtoken().
 DATA(lt_auth_data) = oo_result->get_authorizationdata().
 IF lines(lt_auth_data) > 0.
 READ TABLE lt_auth_data INDEX 1 INTO DATA(lo_auth_data).
 DATA(lv_token) = lo_auth_data->get_authorizationtoken().
 MESSAGE 'Authorization token retrieved.' TYPE 'I'.
 ENDIF.
 CATCH /aws1/cx_ecrserverexception.
 MESSAGE 'Server exception occurred.' TYPE 'I'.
ENDTRY.
```

- 如需 API 詳細資訊，請參閱《適用於 AWS SAP ABAP 的 SDK API 參考》中的 [GetAuthorizationToken](#)。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱 [搭配 SDK 使用 Amazon ECR AWS](#)。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

## GetRepositoryPolicy 搭配 AWS SDK 或 CLI 使用

下列程式碼範例示範如何使用 GetRepositoryPolicy。

動作範例是大型程式的程式碼摘錄，必須在內容中執行。您可以在下列程式碼範例的內容中看到此動作：

- [了解基本概念](#)

## CLI

## AWS CLI

## 擷取儲存庫的儲存庫政策

下列 `get-repository-policy` 範例顯示 `cluster-autoscaler` 儲存庫的儲存庫政策的詳細資訊。

```
aws ecr get-repository-policy \
 --repository-name cluster-autoscaler
```


輸出：

```
{
 "registryId": "012345678910",
 "repositoryName": "cluster-autoscaler",
 "policyText": "{
 \"Version\" : \"2008-10-17\",
 \"Statement\" :
 [{
 \"Sid\" : \"allow public pull\",
 \"Effect\" : \"Allow\",
 \"Principal\" : \"*\",
 \"Action\" : [\"ecr:BatchCheckLayerAvailability\",
 \"ecr:BatchGetImage\", \"ecr:GetDownloadUrlForLayer\"]
 }]
 }"
```

- 如需 API 詳細資訊，請參閱《AWS CLI 命令參考》中的 [GetRepositoryPolicy](#)。

## Java

## SDK for Java 2.x

 Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
/**
 * Gets the repository policy for the specified repository.
 *
 * @param repoName the name of the repository.
 * @throws EcrException if an AWS error occurs while getting the repository
 policy.
```

```
 */
 public String getRepoPolicy(String repoName) {
 if (repoName == null || repoName.isEmpty()) {
 throw new IllegalArgumentException("Repository name cannot be null or
empty");
 }

 GetRepositoryPolicyRequest getRepositoryPolicyRequest =
GetRepositoryPolicyRequest.builder()
 .repositoryName(repoName)
 .build();

 CompletableFuture<GetRepositoryPolicyResponse> response =
getAsyncClient().getRepositoryPolicy(getRepositoryPolicyRequest);
 response.whenComplete((resp, ex) -> {
 if (resp != null) {
 System.out.println("Repository policy retrieved successfully.");
 } else {
 if (ex.getCause() instanceof EcrException) {
 throw (EcrException) ex.getCause();
 } else {
 String errorMessage = "Unexpected error occurred: " +
ex.getMessage();
 throw new RuntimeException(errorMessage, ex);
 }
 }
 });

 GetRepositoryPolicyResponse result = response.join();
 return result != null ? result.policyText() : null;
 }
}
```

- 如需 API 詳細資訊，請參閱《AWS SDK for Java 2.x API 參考》中的 [GetRepositoryPolicy](#)。

## Kotlin

### 適用於 Kotlin 的 SDK

#### Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
/**
 * Gets the repository policy for the specified repository.
 *
 * @param repoName the name of the repository.
 */
suspend fun getRepoPolicy(repoName: String?): String? {
 require(!(repoName == null || repoName.isEmpty())) { "Repository name
cannot be null or empty" }

 // Create the request
 val getRepositoryPolicyRequest =
 GetRepositoryPolicyRequest {
 repositoryName = repoName
 }
 EcrClient.fromEnvironment { region = "us-east-1" }.use { ecrClient ->
 val response =
ecrClient.getRepositoryPolicy(getRepositoryPolicyRequest)
 val responseText = response.policyText
 return responseText
 }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的 AWS SDK API 參考》中的 [GetRepositoryPolicy](#)。

## Python

### 適用於 Python 的 SDK (Boto3)

#### Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
class ECRWrapper:
 def __init__(self, ecr_client: client):
 self.ecr_client = ecr_client

 @classmethod
 def from_client(cls) -> "ECRWrapper":
 """
 Creates a ECRWrapper instance with a default Amazon ECR client.

 :return: An instance of ECRWrapper initialized with the default Amazon
 ECR client.
 """
 ecr_client = boto3.client("ecr")
 return cls(ecr_client)

 def get_repository_policy(self, repository_name: str) -> str:
 """
 Gets the policy for an ECR repository.

 :param repository_name: The name of the repository to get the policy for.
 :return: The policy text.
 """
 try:
 response = self.ecr_client.get_repository_policy(
 repositoryName=repository_name
)
 return response["policyText"]
 except ClientError as err:
 if err.response["Error"]["Code"] ==
 "RepositoryPolicyNotFoundException":
 logger.error("Repository does not exist. %s.", repository_name)
```

```
 raise
 else:
 logger.error(
 "Couldn't get repository policy for repository %s. Here's why
%s",
 repository_name,
 err.response["Error"]["Message"],
)
 raise
```

- 如需 API 詳細資訊，請參閱《AWS SDK for Python (Boto3) API 參考》中的 [GetRepositoryPolicy](#)。

## SAP ABAP

### 適用於 SAP ABAP 的開發套件

#### Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
TRY.
 " iv_repository_name = 'my-repository'
 oo_result = lo_ecr->getrepositorypolicy(
 iv_repositoryname = iv_repository_name).
 DATA(lv_policy_text) = oo_result->get_policytext().
 MESSAGE 'Repository policy retrieved.' TYPE 'I'.
CATCH /aws1/cx_ecrrepositorynotfound.
 MESSAGE 'Repository not found.' TYPE 'I'.
CATCH /aws1/cx_ecrrepositoryplynot00.
 MESSAGE 'Repository policy not found.' TYPE 'I'.
ENDTRY.
```

- 如需 API 詳細資訊，請參閱《適用於 AWS SAP ABAP 的 SDK API 參考》中的 [GetRepositoryPolicy](#)。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱 [搭配 SDK 使用 Amazon ECR AWS](#)。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

## ListImages 搭配 AWS SDK 或 CLI 使用

下列程式碼範例示範如何使用 ListImages。

### CLI

#### AWS CLI

在儲存庫中列出映像

下列 list-images 範例顯示 cluster-autoscaler 儲存庫中的映像清單。

```
aws ecr list-images \
 --repository-name cluster-autoscaler
```

輸出：

```
{
 "imageIds": [
 {
 "imageDigest":
"sha256:99c6fb4377e9a420a1eb3b410a951c9f464eff3b7dbc76c65e434e39b94b6570",
 "imageTag": "v1.13.8"
 },
 {
 "imageDigest":
"sha256:99c6fb4377e9a420a1eb3b410a951c9f464eff3b7dbc76c65e434e39b94b6570",
 "imageTag": "v1.13.7"
 },
 {
 "imageDigest":
"sha256:4a1c6567c38904384ebc64e35b7eeddd8451110c299e3368d2210066487d97e5",
 "imageTag": "v1.13.6"
 }
]
}
```

- 如需 API 詳細資訊，請參閱《AWS CLI 命令參考》中的 [ListImages](#)。

## Rust

### 適用於 Rust 的 SDK

#### Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
async fn show_images(
 client: &aws_sdk_ecr::Client,
 repository: &str,
) -> Result<(), aws_sdk_ecr::Error> {
 let rsp = client
 .list_images()
 .repository_name(repository)
 .send()
 .await?;

 let images = rsp.image_ids();

 println!("found {} images", images.len());

 for image in images {
 println!(
 "image: {}:{}",
 image.image_tag().unwrap(),
 image.image_digest().unwrap()
);
 }

 Ok(())
}
```

- 如需 API 詳細資訊，請參閱《AWS SDK for Rust API 參考》中的 [ListImages](#)。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱 [搭配 SDK 使用 Amazon ECR AWS](#)。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

## PushImageCmd 搭配 AWS SDK 使用

下列程式碼範例示範如何使用 PushImageCmd。

### Java

適用於 Java 2.x 的 SDK

#### Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
/**
 * Pushes a Docker image to an Amazon Elastic Container Registry (ECR)
 * repository.
 *
 * @param repoName the name of the ECR repository to push the image to.
 * @param imageName the name of the Docker image.
 */
public void pushDockerImage(String repoName, String imageName) {
 System.out.println("Pushing " + imageName + " to Amazon ECR will take a
few seconds.");
 CompletableFuture<AuthConfig> authResponseFuture =
getAsyncClient().getAuthorizationToken()
 .thenApply(response -> {
 String token =
response.authorizationData().get(0).authorizationToken();
 String decodedToken = new
String(Base64.getDecoder().decode(token));
 String password = decodedToken.substring(4);

 DescribeRepositoriesResponse descrRepoResponse =
getAsyncClient().describeRepositories(b -> b.repositoryNames(repoName)).join();
 Repository repoData =
descrRepoResponse.repositories().stream().filter(r ->
r.repositoryName().equals(repoName)).findFirst().orElse(null);
 assert repoData != null;
 String registryURL = repoData.repositoryUri().split("/")[0];

 AuthConfig authConfig = new AuthConfig()
```

```
 .withUsername("AWS")
 .withPassword(password)
 .withRegistryAddress(registryURL);
 return authConfig;
})
.thenCompose(authConfig -> {
 DescribeRepositoriesResponse descrRepoResponse =
getAsyncClient().describeRepositories(b -> b.repositoryNames(repoName)).join();
 Repository repoData =
descrRepoResponse.repositories().stream().filter(r ->
r.repositoryName().equals(repoName)).findFirst().orElse(null);
 getDockerClient().tagImageCmd(imageName + ":latest",
repoData.repositoryUri() + ":latest", imageName).exec();
 try {

getDockerClient().pushImageCmd(repoData.repositoryUri()).withTag("echo-
text").withAuthConfig(authConfig).start().awaitCompletion();
 System.out.println("The " + imageName + " was pushed to
ECR");

 } catch (InterruptedException e) {
 throw (RuntimeException) e.getCause();
 }
 return CompletableFuture.completedFuture(authConfig);
});

authResponseFuture.join();
}
```

- 如需 API 詳細資訊，請參閱《AWS SDK for Java 2.x API 參考》中的 [PushImageCmd](#)。

## Kotlin

### 適用於 Kotlin 的 SDK

#### Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
/**
 * Pushes a Docker image to an Amazon Elastic Container Registry (ECR)
 repository.
 *
 * @param repoName the name of the ECR repository to push the image to.
 * @param imageName the name of the Docker image.
 */
suspend fun pushDockerImage(
 repoName: String,
 imageName: String,
) {
 println("Pushing $imageName to $repoName will take a few seconds")
 val authConfig = getAuthConfig(repoName)

 EcrClient.fromEnvironment { region = "us-east-1" }.use { ecrClient ->
 val desRequest =
 DescribeRepositoriesRequest {
 repositoryNames = listOf(repoName)
 }

 val describeRepoResponse = ecrClient.describeRepositories(desRequest)
 val repoData =
 describeRepoResponse.repositories?.firstOrNull
 { it.repositoryName == repoName }
 ?: throw RuntimeException("Repository not found: $repoName")

 val tagImageCmd = getDockerClient()?.tagImageCmd("$imageName",
 "${repoData.repositoryUri}", imageName)
 if (tagImageCmd != null) {
 tagImageCmd.exec()
 }
 val pushImageCmd =
 repoData.repositoryUri?.let {
 dockerClient?.pushImageCmd(it)
 // ?.withTag("latest")
 ?.withAuthConfig(authConfig)
 }

 try {
 if (pushImageCmd != null) {
 pushImageCmd.start().awaitCompletion()
 }
 }
 }
}
```

```
 println("The $imageName was pushed to Amazon ECR")
 } catch (e: IOException) {
 throw RuntimeException(e)
 }
}
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的 AWS SDK API 參考》中的 [PushImageCmd](#)。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱 [搭配 SDK 使用 Amazon ECR AWS](#)。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

## PutLifecyclePolicy 搭配 AWS SDK 或 CLI 使用

下列程式碼範例示範如何使用 PutLifecyclePolicy。

### CLI

#### AWS CLI

##### 建立生命週期政策

下列 put-lifecycle-policy 範例會在帳戶預設登錄檔中，為指定的儲存庫建立生命週期政策。

```
aws ecr put-lifecycle-policy \
 --repository-name "project-a/amazon-ecs-sample" \
 --lifecycle-policy-text "file://policy.json"
```

policy.json 的內容：

```
{
 "rules": [
 {
 "rulePriority": 1,
 "description": "Expire images older than 14 days",
 "selection": {
 "tagStatus": "untagged",
 "countType": "sinceImagePushed",
 "countUnit": "days",
 "countNumber": 14
 }
 }
]
}
```

```

 },
 "action": {
 "type": "expire"
 }
}
]
}

```

輸出：

```

{
 "registryId": "<aws_account_id>",
 "repositoryName": "project-a/amazon-ecs-sample",
 "lifecyclePolicyText": "{\"rules\": [{\"rulePriority\": 1, \"description\": \"Expire images older than 14 days\", \"selection\": {\"tagStatus\": \"untagged\", \"countType\": \"sinceImagePushed\", \"countUnit\": \"days\", \"countNumber\": 14}, \"action\": {\"type\": \"expire\"}}]}"
}

```

如需詳細資訊，請參閱《Amazon ECR 使用者指南》中的[生命週期政策](#)。

- 如需 API 詳細資訊，請參閱《AWS CLI 命令參考》中的 [PutLifecyclePolicy](#)。

## Python

適用於 Python 的 SDK (Boto3)

### Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```

class ECRWrapper:
 def __init__(self, ecr_client: client):
 self.ecr_client = ecr_client

 @classmethod
 def from_client(cls) -> "ECRWrapper":
 """
 Creates a ECRWrapper instance with a default Amazon ECR client.

```

```
 :return: An instance of ECRWrapper initialized with the default Amazon
 ECR client.
 """
 ecr_client = boto3.client("ecr")
 return cls(ecr_client)

 def put_lifecycle_policy(self, repository_name: str, lifecycle_policy_text:
 str):
 """
 Puts a lifecycle policy for an ECR repository.

 :param repository_name: The name of the repository to put the lifecycle
 policy for.
 :param lifecycle_policy_text: The lifecycle policy text to put.
 """
 try:
 self.ecr_client.put_lifecycle_policy(
 repositoryName=repository_name,
 lifecyclePolicyText=lifecycle_policy_text,
)
 print(f"Put lifecycle policy for repository {repository_name}.")
 except ClientError as err:
 logger.error(
 "Couldn't put lifecycle policy for repository %s. Here's why %s",
 repository_name,
 err.response["Error"]["Message"],
)
 raise
```

放置到期日政策的範例。

```
def put_expiration_policy(self):
 """
 Puts an expiration policy on the ECR repository.
 """
 policy_json = {
 "rules": [
 {
 "rulePriority": 1,
```

```

 "description": "Expire images older than 14 days",
 "selection": {
 "tagStatus": "any",
 "countType": "sinceImagePushed",
 "countUnit": "days",
 "countNumber": 14,
 },
 "action": {"type": "expire"},
 }
]
}

self.ecr_wrapper.put_lifecycle_policy(
 self.repository_name, json.dumps(policy_json)
)

```

- 如需 API 詳細資訊，請參閱《AWS SDK for Python (Boto3) API 參考》中的 [PutLifecyclePolicy](#)。

## SAP ABAP

### 適用於 SAP ABAP 的開發套件

#### Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```

TRY.
 " iv_repository_name = 'my-repository'
 " iv_lifecycle_policy_text = '{"rules":
[{"rulePriority":1,"description":"Expire images older than 14 days",...}]}'
 lo_ecr->putlifecyclepolicy(
 iv_repositoryname = iv_repository_name
 iv_lifecyclepolicytext = iv_lifecycle_policy_text).
 MESSAGE |Lifecycle policy set for repository { iv_repository_name }.|
TYPE 'I'.
CATCH /aws1/cx_ecrrepositorynotfndex.

```

```
MESSAGE 'Repository not found.' TYPE 'I'.
CATCH /aws1/cx_ecrvalidationex.
MESSAGE 'Invalid lifecycle policy format.' TYPE 'I'.
ENDTRY.
```

- 如需 API 詳細資訊，請參閱《適用於 AWS SAP ABAP 的 SDK API 參考》中的 [PutLifecyclePolicy](#)。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱 [搭配 SDK 使用 Amazon ECR AWS](#)。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

## SetRepositoryPolicy 搭配 AWS SDK 或 CLI 使用

下列程式碼範例示範如何使用 SetRepositoryPolicy。

動作範例是大型程式的程式碼摘錄，必須在內容中執行。您可以在下列程式碼範例的內容中看到此動作：

- [了解基本概念](#)

### CLI

#### AWS CLI

設定儲存庫的儲存庫政策

下列 set-repository-policy 範例會將檔案中包含的儲存庫政策，連接至 cluster-autoscaler 儲存庫。

```
aws ecr set-repository-policy \
 --repository-name cluster-autoscaler \
 --policy-text file://my-policy.json
```

my-policy.json 的內容：

```
{
 "Version": "2012-10-17",
 "Statement" : [
 {
 "Sid" : "allow public pull",
```

```

 "Effect" : "Allow",
 "Principal" : "*",
 "Action" : [
 "ecr:BatchCheckLayerAvailability",
 "ecr:BatchGetImage",
 "ecr:GetDownloadUrlForLayer"
]
 }
]
}

```

輸出：

```

{
 "registryId": "012345678910",
 "repositoryName": "cluster-autoscaler",
 "policyText": "{\n \"Version\" : \"2008-10-17\",\n \"Statement\" :\n [\n {\n \"Sid\" : \"allow public pull\",\n \"Effect\" : \"Allow\",\n \"Principal\" : \"*\",\n \"Action\" : [\"ecr:BatchCheckLayerAvailability\",\n \"ecr:BatchGetImage\", \"ecr:GetDownloadUrlForLayer\"]\n }\n]\n}"
}

```

- 如需 API 詳細資訊，請參閱《AWS CLI 命令參考》中的 [SetRepositoryPolicy](#)。

## Java

### SDK for Java 2.x

#### Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```

/**
 * Sets the repository policy for the specified ECR repository.
 *
 * @param repoName the name of the ECR repository.
 * @param iamRole the IAM role to be granted access to the repository.
 * @throws RepositoryPolicyNotFoundException if the repository policy does
 * not exist.

```

```
 * @throws EcrException if there is an unexpected error
 setting the repository policy.
 */
 public void setRepoPolicy(String repoName, String iamRole) {
 /*
 This example policy document grants the specified AWS principal the
 permission to perform the
 `ecr:BatchGetImage` action. This policy is designed to allow the
 specified principal
 to retrieve Docker images from the ECR repository.
 */
 String policyDocumentTemplate = ""
 {
 "Version": "2012-10-17",
 "Statement" : [{
 "Sid" : "new statement",
 "Effect" : "Allow",
 "Principal" : {
 "AWS" : "%s"
 },
 "Action" : "ecr:BatchGetImage"
 }]
 }
 """;

 String policyDocument = String.format(policyDocumentTemplate, iamRole);
 SetRepositoryPolicyRequest setRepositoryPolicyRequest =
 SetRepositoryPolicyRequest.builder()
 .repositoryName(repoName)
 .policyText(policyDocument)
 .build();

 CompletableFuture<SetRepositoryPolicyResponse> response =
 getAsyncClient().setRepositoryPolicy(setRepositoryPolicyRequest);
 response.whenComplete((resp, ex) -> {
 if (resp != null) {
 System.out.println("Repository policy set successfully.");
 } else {
 Throwable cause = ex.getCause();
 if (cause instanceof RepositoryPolicyNotFoundException) {
 throw (RepositoryPolicyNotFoundException) cause;
 } else if (cause instanceof EcrException) {
 throw (EcrException) cause;
 } else {

```

```
 String errorMessage = "Unexpected error: " +
cause.getMessage();
 throw new RuntimeException(errorMessage, cause);
 }
}
});
response.join();
}
```

- 如需 API 詳細資訊，請參閱《AWS SDK for Java 2.x API 參考》中的 [SetRepositoryPolicy](#)。

## Kotlin

### 適用於 Kotlin 的 SDK

#### Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
/**
 * Sets the repository policy for the specified ECR repository.
 *
 * @param repoName the name of the ECR repository.
 * @param iamRole the IAM role to be granted access to the repository.
 */
suspend fun setRepoPolicy(
 repoName: String?,
 iamRole: String?,
) {
 val policyDocumentTemplate =
 """
 {
 "Version": "2012-10-17",
 "Statement" : [{
 "Sid" : "new statement",
 "Effect" : "Allow",
 "Principal" : {
 "AWS" : "$iamRole"
 }
 }
]
 }
 """
}
```

```

 },
 "Action" : "ecr:BatchGetImage"
 }]
}

"".trimIndent()
val setRepositoryPolicyRequest =
 SetRepositoryPolicyRequest {
 repositoryName = repoName
 policyText = policyDocumentTemplate
 }

EcrClient.fromEnvironment { region = "us-east-1" }.use { ecrClient ->
 val response =
ecrClient.setRepositoryPolicy(setRepositoryPolicyRequest)
 if (response != null) {
 println("Repository policy set successfully.")
 }
}
}
}

```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的 AWS SDK API 參考》中的 [SetRepositoryPolicy](#)。

## Python

### 適用於 Python 的 SDK (Boto3)

#### Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```

class ECRWrapper:
 def __init__(self, ecr_client: client):
 self.ecr_client = ecr_client

 @classmethod
 def from_client(cls) -> "ECRWrapper":

```

```
"""
Creates a ECRWrapper instance with a default Amazon ECR client.

:return: An instance of ECRWrapper initialized with the default Amazon
ECR client.
"""
ecr_client = boto3.client("ecr")
return cls(ecr_client)

def set_repository_policy(self, repository_name: str, policy_text: str):
 """
 Sets the policy for an ECR repository.

 :param repository_name: The name of the repository to set the policy for.
 :param policy_text: The policy text to set.
 """
 try:
 self.ecr_client.set_repository_policy(
 repositoryName=repository_name, policyText=policy_text
)
 print(f"Set repository policy for repository {repository_name}.")
 except ClientError as err:
 if err.response["Error"]["Code"] ==
"RepositoryPolicyNotFoundException":
 logger.error("Repository does not exist. %s.", repository_name)
 raise
 else:
 logger.error(
 "Couldn't set repository policy for repository %s. Here's why
%s",
 repository_name,
 err.response["Error"]["Message"],
)
 raise
```

授予 IAM 角色下載存取權的範例。

```
def grant_role_download_access(self, role_arn: str):
 """
```

Grants the specified role access to download images from the ECR repository.

```

:param role_arn: The ARN of the role to grant access to.
"""
policy_json = {
 "Version": "2012-10-17",
 "Statement": [
 {
 "Sid": "AllowDownload",
 "Effect": "Allow",
 "Principal": {"AWS": role_arn},
 "Action": ["ecr:BatchGetImage"],
 }
],
}

self.ecr_wrapper.set_repository_policy(
 self.repository_name, json.dumps(policy_json)
)

```

- 如需 API 詳細資訊，請參閱《AWS SDK for Python (Boto3) API 參考》中的 [SetRepositoryPolicy](#)。

## SAP ABAP

適用於 SAP ABAP 的開發套件

### Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

TRY.

```

" iv_repository_name = 'my-repository'
" iv_policy_text = '{"Version": "2012-10-17", "Statement": [...]}'
lo_ecr->setrepositorypolicy(
 iv_repositoryname = iv_repository_name

```

```
 iv_policytext = iv_policy_text).
 MESSAGE |Policy set for repository { iv_repository_name }.| TYPE 'I'.
 CATCH /aws1/cx_ecrrepositorynotfound.
 MESSAGE 'Repository not found.' TYPE 'I'.
ENDTRY.
```

- 如需 API 詳細資訊，請參閱《適用於 AWS SAP ABAP 的 SDK API 參考》中的 [SetRepositoryPolicy](#)。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱 [搭配 SDK 使用 Amazon ECR AWS](#)。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

## StartLifecyclePolicyPreview 搭配 AWS SDK 或 CLI 使用

下列程式碼範例示範如何使用 StartLifecyclePolicyPreview。

動作範例是大型程式的程式碼摘錄，必須在內容中執行。您可以在下列程式碼範例的內容中看到此動作：

- [了解基本概念](#)

### CLI

#### AWS CLI

##### 建立生命週期政策預覽

下列 start-lifecycle-policy-preview 範例會為指定的儲存庫建立由 JSON 檔案定義的生命週期政策預覽。

```
aws ecr start-lifecycle-policy-preview \
 --repository-name "project-a/amazon-ecs-sample" \
 --lifecycle-policy-text "file://policy.json"
```

policy.json 的內容：

```
{
 "rules": [
 {
```

```

 "rulePriority": 1,
 "description": "Expire images older than 14 days",
 "selection": {
 "tagStatus": "untagged",
 "countType": "sinceImagePushed",
 "countUnit": "days",
 "countNumber": 14
 },
 "action": {
 "type": "expire"
 }
 }
]
}

```

輸出：

```

{
 "registryId": "012345678910",
 "repositoryName": "project-a/amazon-ecs-sample",
 "lifecyclePolicyText": "{\n \"rules\": [\n {\n\n \"rulePriority\": 1,\n \"description\": \"Expire images older than 14\n days\",\n \"selection\": {\n \"tagStatus\": \"untagged\n \",\n \"countType\": \"sinceImagePushed\",\n \"countUnit\": \"days\",\n \"countNumber\": 14\n },\n \"action\": {\n \"type\": \"expire\"\n }\n }\n]\n}\n",
 "status": "IN_PROGRESS"
}

```

- 如需 API 詳細資訊，請參閱《AWS CLI 命令參考》中的 [StartLifecyclePolicyPreview](#)。

## Java

### SDK for Java 2.x

#### Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
/**
 * Verifies the existence of an image in an Amazon Elastic Container Registry
 (Amazon ECR) repository asynchronously.
 *
 * @param repositoryName The name of the Amazon ECR repository.
 * @param imageTag The tag of the image to verify.
 * @throws EcrException if there is an error retrieving the image
 information from Amazon ECR.
 * @throws CompletionException if the asynchronous operation completes
 exceptionally.
 */
public void verifyImage(String repositoryName, String imageTag) {
 DescribeImagesRequest request = DescribeImagesRequest.builder()
 .repositoryName(repositoryName)
 .imageIds(ImageIdentifier.builder().imageTag(imageTag).build())
 .build();

 CompletableFuture<DescribeImagesResponse> response =
getAsyncClient().describeImages(request);
 response.whenComplete((describeImagesResponse, ex) -> {
 if (ex != null) {
 if (ex instanceof CompletionException) {
 Throwable cause = ex.getCause();
 if (cause instanceof EcrException) {
 throw (EcrException) cause;
 } else {
 throw new RuntimeException("Unexpected error: " +
cause.getMessage(), cause);
 }
 } else {
 throw new RuntimeException("Unexpected error: " +
ex.getCause());
 }
 } else if (describeImagesResponse != null && !
describeImagesResponse.imageDetails().isEmpty()) {
 System.out.println("Image is present in the repository.");
 } else {
 System.out.println("Image is not present in the repository.");
 }
 });

 // Wait for the CompletableFuture to complete.
 response.join();
}
```

```
}
```

- 如需 API 詳細資訊，請參閱《AWS SDK for Java 2.x API 參考》中的 [StartLifecyclePolicyPreview](#)。

## Kotlin

### 適用於 Kotlin 的 SDK

#### Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
/**
 * Verifies the existence of an image in an Amazon Elastic Container Registry
 (Amazon ECR) repository asynchronously.
 *
 * @param repositoryName The name of the Amazon ECR repository.
 * @param imageTag The tag of the image to verify.
 */
suspend fun verifyImage(
 repoName: String?,
 imageTagVal: String?,
) {
 require(!(repoName == null || repoName.isEmpty())) { "Repository name
cannot be null or empty" }
 require(!(imageTagVal == null || imageTagVal.isEmpty())) { "Image tag
cannot be null or empty" }

 val imageId =
 ImageIdentifier {
 imageTag = imageTagVal
 }
 val request =
 DescribeImagesRequest {
 repositoryName = repoName
 imageIds = listOf(imageId)
```

```
 }

 EcrClient.fromEnvironment { region = "us-east-1" }.use { ecrClient ->
 val describeImagesResponse = ecrClient.describeImages(request)
 if (describeImagesResponse != null && !
describeImagesResponse.imageDetails?.isEmpty()!!) {
 println("Image is present in the repository.")
 } else {
 println("Image is not present in the repository.")
 }
 }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的 AWS SDK API 參考》中的 [StartLifecyclePolicyPreview](#)。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱 [搭配 SDK 使用 Amazon ECR AWS](#)。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

## 使用 AWS SDKs Amazon ECR 案例

下列程式碼範例示範如何在 Amazon ECR AWS SDKs 中實作常見案例。這些案例說明如何透過在 Amazon ECR 中呼叫多個函數或與其他函數結合，來完成特定任務 AWS 服務。每個案例均包含完整原始碼的連結，您可在連結中找到如何設定和執程式碼的相關指示。

案例的目標是獲得中等水平的經驗，協助您了解內容中的服務動作。

### 範例

- [Amazon ECR 入門](#)

## Amazon ECR 入門

以下程式碼範例顯示做法：

- 建立 Docker 映像
- 建立 Amazon ECR 儲存庫
- 刪除資源

## Bash

### AWS CLI 搭配 Bash 指令碼

#### Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在[範例開發人員教學課程](#)儲存庫中設定和執行。

```
#!/bin/bash

Amazon ECR Getting Started Script
This script demonstrates the lifecycle of a Docker image in Amazon ECR

Set up logging
LOG_FILE="ecr-tutorial.log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "======"
echo "Amazon ECR Getting Started Tutorial"
echo "======"
echo "This script will:"
echo "1. Create a Docker image"
echo "2. Create an Amazon ECR repository"
echo "3. Authenticate to Amazon ECR"
echo "4. Push the image to Amazon ECR"
echo "5. Pull the image from Amazon ECR"
echo "6. Clean up resources (optional)"
echo "======"

Check prerequisites
echo "Checking prerequisites..."

Check if AWS CLI is installed
if ! command -v aws &> /dev/null; then
 echo "ERROR: AWS CLI is not installed. Please install it before running this
 script."
 echo "Visit https://docs.aws.amazon.com/cli/latest/userguide/install-
 cliv2.html for installation instructions."
 exit 1
fi
```

```
Check if AWS CLI is configured
if ! aws sts get-caller-identity &> /dev/null; then
 echo "ERROR: AWS CLI is not configured properly. Please run 'aws configure'
 to set up your credentials."
 exit 1
fi

Check if Docker is installed
if ! command -v docker &> /dev/null; then
 echo "ERROR: Docker is not installed. Please install Docker before running
 this script."
 echo "Visit https://docs.docker.com/get-docker/ for installation
 instructions."
 exit 1
fi

Check if Docker daemon is running
if ! docker info &> /dev/null; then
 echo "ERROR: Docker daemon is not running. Please start Docker and try
 again."
 exit 1
fi

echo "All prerequisites met."

Initialize variables
REPO_URI=""
TIMEOUT_CMD="timeout 300" # 5-minute timeout for long-running commands

Function to handle errors
handle_error() {
 echo "ERROR: $1"
 echo "Check the log file for details: $LOG_FILE"

 echo "====="
 echo "Resources created:"
 echo "- Docker image: hello-world (local)"
 if [-n "$REPO_URI"]; then
 echo "- ECR Repository: hello-repository"
 echo "- ECR Image: $REPO_URI:latest"
 fi
 echo "====="
}
```

```
 echo "Attempting to clean up resources..."
 cleanup
 exit 1
}

Function to clean up resources
cleanup() {
 echo "======"
 echo "Cleaning up resources..."

 # Delete the image from ECR if it exists
 if [-n "$REPO_URI"]; then
 echo "Deleting image from ECR repository..."
 aws ecr batch-delete-image --repository-name hello-repository --image-ids
imageTag=latest || echo "Failed to delete image, it may not exist or may have
already been deleted."
 fi

 # Delete the ECR repository if it exists
 if [-n "$REPO_URI"]; then
 echo "Deleting ECR repository..."
 aws ecr delete-repository --repository-name hello-repository --force ||
echo "Failed to delete repository, it may not exist or may have already been
deleted."
 fi

 # Remove local Docker image
 echo "Removing local Docker image..."
 docker rmi hello-world:latest 2>/dev/null || echo "Failed to remove local
image, it may not exist or may have already been deleted."
 if [-n "$REPO_URI"]; then
 docker rmi "$REPO_URI:latest" 2>/dev/null || echo "Failed to remove
tagged image, it may not exist or may have already been deleted."
 fi

 echo "Cleanup completed."
 echo "======"
}

Step 1: Create a Docker image
echo "Step 1: Creating a Docker image"

Create Dockerfile
echo "Creating Dockerfile..."
```

```
cat > Dockerfile << 'EOF'
FROM public.ecr.aws/amazonlinux/amazonlinux:latest

Install dependencies
RUN yum update -y && \
 yum install -y httpd

Install apache and write hello world message
RUN echo 'Hello World!' > /var/www/html/index.html

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

EXPOSE 80

CMD /root/run_apache.sh
EOF

Build Docker image
echo "Building Docker image..."
$TIMEOUT_CMD docker build -t hello-world . || handle_error "Failed to build
 Docker image or operation timed out after 5 minutes"

Verify image was created
echo "Verifying Docker image..."
docker images --filter reference=hello-world || handle_error "Failed to list
 Docker images"

echo "Docker image created successfully."

Step 2: Create an Amazon ECR repository
echo "Step 2: Creating an Amazon ECR repository"

Get AWS account ID
echo "Getting AWS account ID..."
AWS_ACCOUNT_ID=$(aws sts get-caller-identity --query Account --output text)
if [[-z "$AWS_ACCOUNT_ID" || "$AWS_ACCOUNT_ID" == *"error"*]]; then
 handle_error "Failed to get AWS account ID. Make sure your AWS credentials
 are configured correctly."
fi
echo "AWS Account ID: $AWS_ACCOUNT_ID"
```

```
Get current region
AWS_REGION=$(aws configure get region)
if [[-z "$AWS_REGION"]]; then
 AWS_REGION="us-east-1" # Default to us-east-1 if no region is configured
 echo "No AWS region configured, defaulting to $AWS_REGION"
else
 echo "Using AWS region: $AWS_REGION"
fi

Create ECR repository
echo "Creating ECR repository..."
REPO_RESULT=$(aws ecr create-repository --repository-name hello-repository --
tags key=project,value=doc-smith key=tutorial,value=amazon-elastic-container-
registry-gs)
if [[-z "$REPO_RESULT" || "$REPO_RESULT" == *"error"*]]; then
 handle_error "Failed to create ECR repository"
fi

Extract repository URI
REPO_URI="$AWS_ACCOUNT_ID.dkr.ecr.$AWS_REGION.amazonaws.com/hello-repository"
echo "Repository URI: $REPO_URI"

Step 3: Authenticate to Amazon ECR
echo "Step 3: Authenticating to Amazon ECR"

echo "Getting ECR login password..."
aws ecr get-login-password --region "$AWS_REGION" | docker login --username
AWS --password-stdin "$AWS_ACCOUNT_ID.dkr.ecr.$AWS_REGION.amazonaws.com" ||
handle_error "Failed to authenticate to ECR"

echo "Successfully authenticated to ECR."

Step 4: Push the image to Amazon ECR
echo "Step 4: Pushing the image to Amazon ECR"

Tag the image
echo "Tagging Docker image..."
docker tag hello-world:latest "$REPO_URI:latest" || handle_error "Failed to tag
Docker image"

Push the image with timeout
echo "Pushing image to ECR..."
```

```
$TIMEOUT_CMD docker push "$REPO_URI:latest" || handle_error "Failed to push image
to ECR or operation timed out after 5 minutes"

echo "Successfully pushed image to ECR."

Step 5: Pull the image from Amazon ECR
echo "Step 5: Pulling the image from Amazon ECR"

Remove local image to ensure we're pulling from ECR
echo "Removing local tagged image..."
docker rmi "$REPO_URI:latest" || echo "Warning: Failed to remove local tagged
image"

Pull the image with timeout
echo "Pulling image from ECR..."
$TIMEOUT_CMD docker pull "$REPO_URI:latest" || handle_error "Failed to pull image
from ECR or operation timed out after 5 minutes"

echo "Successfully pulled image from ECR."

List resources created
echo "======"
echo "Resources created:"
echo "- Docker image: hello-world (local)"
echo "- ECR Repository: hello-repository"
echo "- ECR Image: $REPO_URI:latest"
echo "======"

Ask user if they want to clean up resources
echo ""
echo "======"
echo "CLEANUP CONFIRMATION"
echo "======"
echo "Do you want to clean up all created resources? (y/n): "
read -r CLEANUP_CHOICE

if [["$CLEANUP_CHOICE" =~ ^[Yy]$]]; then
 # Step 6: Delete the image from ECR
 echo "Step 6: Deleting the image from ECR"

 DELETE_IMAGE_RESULT=$(aws ecr batch-delete-image --repository-name hello-
repository --image-ids imageTag=latest)
 if [[-z "$DELETE_IMAGE_RESULT" || "$DELETE_IMAGE_RESULT" == *"error"*]];
then
```

```
 echo "Warning: Failed to delete image from ECR"
else
 echo "Successfully deleted image from ECR."
fi

Step 7: Delete the ECR repository
echo "Step 7: Deleting the ECR repository"

DELETE_REPO_RESULT=$(aws ecr delete-repository --repository-name hello-
repository --force)
if [[-z "$DELETE_REPO_RESULT" || "$DELETE_REPO_RESULT" == *"error"*]]; then
 echo "Warning: Failed to delete ECR repository"
else
 echo "Successfully deleted ECR repository."
fi

Remove local Docker images
echo "Removing local Docker images..."
docker rmi hello-world:latest 2>/dev/null || echo "Warning: Failed to remove
local image"

echo "All resources have been cleaned up."
else
 echo "Resources were not cleaned up. You can manually clean up later with:"
 echo "aws ecr batch-delete-image --repository-name hello-repository --image-
ids imageTag=latest"
 echo "aws ecr delete-repository --repository-name hello-repository --force"
 echo "docker rmi hello-world:latest"
 echo "docker rmi $REPO_URI:latest"
fi

echo "======"
echo "Tutorial completed!"
echo "Log file: $LOG_FILE"
echo "======"
```

- 如需 API 詳細資訊，請參閱《AWS CLI 命令參考》中的下列主題。
  - [BatchDeleteImage](#)
  - [CreateRepository](#)
  - [DeleteRepository](#)
  - [GetCallerIdentity](#)

- [GetLoginPassword](#)

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱 [搭配 SDK 使用 Amazon ECR AWS](#)。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

## Amazon ECR 服務配額

下表提供 Amazon Elastic Container Registry (Amazon ECR) 的預設服務配額。

| 名稱             | 預設                | 可調整      | Description                                                  |
|----------------|-------------------|----------|--------------------------------------------------------------|
| 每 24 小時的基本映像掃描 | 每個受支援的區域：100,000  | 否        | 目前帳戶和區域中可使用基本掃描在 24 小時內掃描的影像數量上限。此限制包括推送時掃描和手動掃描。            |
| 複寫組態中每個規則的篩選條件 | 每個受支援的區域：100      | 否        | 複寫組態中每個規則的篩選條件的數量上限。                                         |
| 每個儲存庫的映像       | 每個受支援的區域：100,000  | <u>是</u> | 每個儲存庫映像的數量上限。                                                |
| 分層部分           | 每個支援的區域：4,200     | 否        | 分層部分數目上限。這僅適用於直接使用 Amazon ECR API 動作來啟動分段上傳以進行映像推送操作時。       |
| 生命週期政策長度       | 每個支援的區域：30,720    | 否        | 生命週期政策中的字元數上限。                                               |
| 最大分層部分大小       | 每個受支援的區域：10       | 否        | 分層部分大小上限 (MiB)。這僅適用於直接使用 Amazon ECR API 動作來啟動分段上傳以進行映像推送操作時。 |
| 最大分層大小         | 每個受支援的區域：52,000 個 | 否        | 每層的大小上限 (MiB)。                                               |

| 名稱                                | 預設                | 可調整      | Description                                                                                                                   |
|-----------------------------------|-------------------|----------|-------------------------------------------------------------------------------------------------------------------------------|
| 最小分層部分大小                          | 每個受支援的區域：5        | 否        | 分層部分大小下限 (MiB)。這僅適用於直接使用 Amazon ECR API 動作來啟動分段上傳以進行映像推送操作時。                                                                  |
| 依登錄檔提取快取規則                        | 每個受支援的區域：50       | 否        | 提取快取規則的數量上限。                                                                                                                  |
| BatchCheckLayerAvailability 請求的速率 | 每個受支援的區域：每秒 1,000 | <u>是</u> | 目前區域中每秒可進行 BatchCheckLayerAvailability 請求的數量上限。將映像推送至儲存器時會檢查每個映像圖層，驗證之前是否上傳過。如果已上傳，則會略過映像圖層。                                  |
| BatchGetImage 請求的速率               | 每個支援的區域：每秒 2,000  | <u>是</u> | 目前區域中每秒可進行 BatchGetImage 請求的數量上限。提取映像時，系統會叫用 BatchGetImage API 一次以擷取映像資訊清單。如果您請求增加此 API 的配額，請同時檢視 GetDownloadUriForLayer 使用量。 |

| 名稱                           | 預設                   | 可調整      | Description                                                                                                                             |
|------------------------------|----------------------|----------|-----------------------------------------------------------------------------------------------------------------------------------------|
| CompleteLayerUpload 請求的速率    | 每個支援的區域：<br>每秒 100   | <u>是</u> | 目前區域中每秒可進行 CompleteLayerUpload 請求的數量上限。推送映像時，每個新映像圖層都會叫用 CompleteLayerUpload API 一次，以確認上傳已完成。                                           |
| GetAuthorizationToken 請求的速率  | 每個支援的區域：<br>每秒 500   | <u>是</u> | 目前區域中每秒可進行 GetAuthorizationToken 請求的數量上限。                                                                                               |
| GetDownloadUrlForLayer 請求的速率 | 每個支援的區域：<br>每秒 3,000 | <u>是</u> | 目前區域中每秒可進行 GetDownloadUrlForLayer 請求的數量上限。提取映像時，未快取的每個映像圖層都會叫用一次 GetDownloadUrlForLayer API。如果您請求增加此 API 的配額，請同時檢視 BatchGetImage 使用量。   |
| InitiateLayerUpload 請求的速率    | 每個支援的區域：<br>每秒 100   | <u>是</u> | 目前區域中每秒可進行 InitiateLayerUpload 請求的數量上限。推送映像時，未上傳的每個映像圖層都會叫用 InitiateLayerUpload API 一次。是否上傳映像圖層會由 BatchCheckLayerAvailability API 動作決定。 |

| 名稱                    | 預設                   | 可調整               | Description                                                                                    |
|-----------------------|----------------------|-------------------|------------------------------------------------------------------------------------------------|
| PutImage 請求的速率        | 每個支援的區域：<br>每秒 10    | <a href="#">是</a> | 目前區域中每秒可進行 PutImage 請求的數量上限。推送映像並上傳所有新映像圖層時，系統會叫用 PutImage API 一次，以建立或更新與映像關聯標籤的映像資訊清單。        |
| UploadLayerPart 請求的速率 | 每個支援的區域：<br>每秒 500   | <a href="#">是</a> | 目前區域中每秒可進行 UploadLayerPart 請求的數量上限。推送映像時，每個新映像層都會分成多個部分上傳，而每個新映像層部分都會呼叫一次 UploadLayerPart API。 |
| 映像掃描速率                | 每個支援的區域：<br>1        | 否                 | 每 24 小時每張映像的映像掃描次數上限。                                                                          |
| 已註冊的儲存庫               | 每個受支援的區域：<br>100,000 | <a href="#">是</a> | 目前區域中，您可以在此帳戶中建立的儲存庫數量上限。                                                                      |
| 每個生命週期政策的規則           | 每個受支援的區域：<br>50      | 否                 | 生命週期政策中規則的數量上限                                                                                 |
| 每個複寫組態的規則             | 每個受支援的區域：<br>10      | 否                 | 複寫組態中規則的數量上限。                                                                                  |
| 每個映像的標籤               | 每個支援的區域：<br>1,000    | 否                 | 每個映像的標籤數目上限。                                                                                   |
| 複寫組態中跨所有規則的唯一目的地      | 每個受支援的區域：<br>25      | 否                 | 複寫組態中跨所有規則的唯一目的地的數量上限。                                                                         |

## 在中管理您的 Amazon ECR 服務配額 AWS 管理主控台

Amazon ECR 已與 Service Quotas 整合，這項 AWS 服務可讓您從中央位置檢視和管理配額。如需詳細資訊，請參閱 Service Quotas 使用者指南中的 [什麼是 Service Quotas ?](#)。

Service Quotas 可讓您輕鬆查詢所有 Amazon ECR 服務配額的值。

### 檢視 Amazon ECR 服務配額 (AWS 管理主控台)

1. 開啟 Service Quotas 主控台，網址為 <https://console.aws.amazon.com/servicequotas/>。
2. 在導覽窗格中，選擇 AWS services (AWS 服務)。
3. 從 AWS 服務清單中，搜尋並選取 Amazon Elastic Container Registry (Amazon ECR)。

在服務配額清單中，您可以看到服務配額名稱、套用值（如果有的話）、AWS 預設配額，以及配額值是否可以調整。

4. 若要檢視服務配額的其他資訊（例如說明），請選擇配額名稱。

若要請求提高配額，請參閱 [《Service Quotas 使用者指南》](#) 中的請求提高配額。

## 建立 CloudWatch 警示以監控 API 用量指標

Amazon ECR 提供 CloudWatch 用量指標，其對應至與登錄驗證、映像推送和映像提取動作相關的每個 APIs AWS 的服務配額。在 Service Quotas 主控台中，您可以在圖表上視覺化您的用量，並設定在用量接近服務配額時提醒您的警示。如需詳細資訊，請參閱 [Amazon ECR 用量指標](#)。

使用下列步驟，根據其中一個 Amazon ECR API 用量指標來建立 CloudWatch 警示。

### 根據您的 Amazon ECR 用量配額建立警示 (AWS 管理主控台)

1. 開啟 Service Quotas 主控台，網址為 <https://console.aws.amazon.com/servicequotas/>。
2. 在導覽窗格中，選擇 AWS services (AWS 服務)。
3. 從 AWS 服務清單中，搜尋並選取 Amazon Elastic Container Registry (Amazon ECR)。
4. 在 Service quotas (服務配額) 清單中，選取您要為其建立警示的 Amazon ECR 用量配額。
5. 在 Amazon CloudWatch Events 警示區段中，選擇 Create (建立)。
6. 針對 Alarm threshold (警示閾值)，選擇您想要多少百分比的套用配額值設為警示值。
7. 針對 Alarm name (警示名稱)，輸入警示的名稱，然後選擇 Create (建立)。

# Amazon ECR 故障診斷

本章可協助您尋找 Amazon ECR 的診斷資訊，並提供常見問題和錯誤訊息的疑難排解步驟。

## 主題

- [對使用 Amazon ECR 時的 Docker 命令和問題進行故障診斷](#)
- [Amazon ECR 錯誤訊息故障診斷](#)

## 對使用 Amazon ECR 時的 Docker 命令和問題進行故障診斷

在某些情況下，對 Amazon ECR 執行 Docker 命令可能會導致錯誤訊息。以下說明了一些一般錯誤訊息與潛在解決方案。

## 主題

- [Docker 日誌不包含預期的錯誤訊息](#)
- [當從 Amazon ECR 儲存庫提取映像時，出現錯誤：「Filesystem Verification Failed」\(檔案系統驗證失敗\) 或「404: Image Not Found」\(404：找不到映像\)](#)
- [當從 Amazon ECR 提取映像時出現錯誤：「Filesystem Layer Verification Failed」\(檔案系統分層驗證失敗\)](#)
- [當推送至儲存庫時，出現 HTTP 403 錯誤或「無基本身分驗證憑證」錯誤](#)

## Docker 日誌不包含預期的錯誤訊息

若要開始偵錯任何 Docker 相關問題，請先在主機執行個體上執行的 Docker 協助程式上開啟 Docker 偵錯輸出。如果您在 Amazon ECS 容器執行個體上使用從 Amazon ECR 提取的影像，請參閱《Amazon Elastic Container Service 開發人員指南》中的[從 Docker 協助程式設定詳細輸出](#)。

當從 Amazon ECR 儲存庫提取映像時，出現錯誤：「Filesystem Verification Failed」(檔案系統驗證失敗) 或「404: Image Not Found」(404：找不到映像)

在 Docker 1.9 或以上版本中使用 docker pull 命令從 Amazon ECR 儲存庫映像時，您可能收到錯誤 Filesystem verification failed。當您使用 Docker 1.9 以前的版本時，您可能會收到錯誤 404: Image not found。

以下列出一些可能的原因及其說明。

### 本機磁碟已滿

若您正在執行 `docker pull` 的本機磁碟已滿，則在本機檔案上計算的 SHA-1 雜湊，可能與 Amazon ECR 計算的 SHA-1 雜湊不同。檢查您的本機磁碟有足夠的剩餘空間以儲存您提取的 Docker 映像。您也可以刪除舊的映像，以為新的映像騰出空間。使用 `docker images` 命令以查看所有本機下載且包含容量大小的 Docker 映像清單。

由於網路錯誤，用戶端無法連接至遠端儲存庫

對 Amazon ECR 儲存庫的呼叫需要正常運作的網路連線。確認您的網路設定，並確認其他工具與應用程式可正常存取網路資源。如果您在私有子網路中的 Amazon EC2 執行個體上執行 `docker pull`，請確認該子網路擁有對網際網路的路由。使用網路地址轉譯 (NAT) 伺服器或受管 NAT 閘道。

目前，對 Amazon ECR 儲存庫的呼叫也需要透過您的企業防火牆的網路存取至 Amazon Simple Storage Service (Amazon S3)。如果您的組織使用允許服務端點的防火牆軟體或 NAT 裝置，請務必確保您目前區域的 Amazon S3 服務端點是受到允許的。

如果您正在 HTTP 代理後使用 Docker，您可以使用適當的代理設定來設定 Docker。如需詳細資訊，請參閱 Docker 文件中的 [HTTP 代理](#)。

## 當從 Amazon ECR 提取映像時出現錯誤：「Filesystem Layer Verification Failed」(檔案系統分層驗證失敗)

當您使用 `docker pull` 命令提取映像時，您可能會收到錯誤 `image image-name not found`。若您檢查 Docker 日誌時，您可能會看到像下列的錯誤：

```
filesystem layer verification failed for digest sha256:2b96f...
```

此錯誤表示一個或更多的映像分層下載失敗。以下列出一些可能的原因及其說明。

您可能正在使用較舊版本的 Docker

在您使用低於 1.10 版本的 Docker 時，有少許機率會出現此錯誤。將您的 Docker 用戶端更新為 1.10 或更新的版本。

## 您的用戶端遭遇網路或磁碟錯誤

完全的磁碟或網路錯誤可能阻礙一個或更多分層的下載，如同稍早討論過的 `Filesystem verification failed` 訊息。遵循上述建議，以確保您的檔案系統未滿，且已在您的網路中啟用存取 Amazon S3。

## 當推送至儲存庫時，出現 HTTP 403 錯誤或「無基本身分驗證憑證」錯誤

即使您已透過 `aws ecr get-login-password` 命令成功驗證至 Docker，有時候您可能仍會收到 HTTP 403 (Forbidden) 錯誤，或是從 `docker push` 或 `docker pull` 命令收到錯誤訊息 `no basic auth credentials`。以下是此問題的一些已知原因：

### 您已驗證至不同區域

驗證請求是繫結至特定區域的，且無法跨區域使用。例如，如果您從美國西部 (奧勒岡) 取得了授權字符，您便不得將其用於驗證您美國東部 (維吉尼亞北部) 中的儲存庫。若要解決此問題，請確認您已從與儲存器所在相同的區域擷取身分驗證字符。如需詳細資訊，請參閱 [the section called “登錄檔身分驗證”](#)。

### 您已完成身分驗證，可推送至您沒有權限的儲存庫

您沒有推送至儲存庫的必要權限。如需詳細資訊，請參閱 [Amazon ECR 中的私有儲存庫政策](#)。

### 您的字符已過期

使用 `GetAuthorizationToken` 操作取得的授權字符預設過期期間是 12 小時。

### wincred 憑證管理工具中的錯誤

某些版本的 Docker for Windows 使用名為的登入資料管理員 `wincred`，無法正確處理產生的 Docker 登入命令 `aws ecr get-login-password` (如需詳細資訊，請參閱 [CredsStore 使用私有儲存庫失敗](#))。您可以執行輸出的 Docker 登入命令，但若您試著推送或提取映像時，那些命令將失敗。您可以透過從 `aws ecr get-login-password` 輸出 Docker 登入命令的登錄檔引數中移除 `https://` 機制以解決此錯誤。無 HTTPS 機制的範例 Docker 登入命令如下所示。

```
docker login -u AWS -p <password> <aws_account_id>.dkr.ecr.<region>.amazonaws.com
```

## Amazon ECR 錯誤訊息故障診斷

在某些情況下，您已透過 Amazon ECR 主控台啟動的 API 呼叫，或 AWS CLI 會結束並顯示錯誤訊息。以下說明了一些一般錯誤訊息與潛在解決方案。

## HTTP 429: Too Many Requests or ThrottleException (過多請求或 ThrottleException)

您可能會收到一或多個 Amazon ECR 動作 429: Too Many Requests 或 API 呼叫的 ThrottleException 錯誤。這表示您正在以短間隔重複呼叫 Amazon ECR 中的單一端點，因此您的請求已被調節。調節會發生在當從單一使用者至單一端點的呼叫在一段時間內超出特定閾值時。

Amazon ECR 中的每個 API 操作都有與其相關聯的速率調節。例如，[GetAuthorizationToken](#) 動作的調節為每秒 20 次交易 (TPS)，爆增上限允許為每秒 200 次交易。在每一區域中，每一帳戶都會接收到可儲存高達 200 GetAuthorizationToken 點的儲存貯體。這些點數以每秒 20 次的速率補充。如果您的儲存貯體有 200 點，您一秒可以達到每秒 200 次的 GetAuthorizationToken API 交易，然後再無限期保持每秒 20 次交易。如需 Amazon ECR APIs 速率限制的詳細資訊，請參閱 [Amazon ECR 服務配額](#)。

欲處理調節錯誤，以增量退避實施重試功能至您的程式碼。如需詳細資訊，請參閱 AWS SDKs 和工具參考指南中的 [重試行為](#)。另一個選項是請求提高速率限制，您可以使用 Service Quotas 主控台執行此操作。如需詳細資訊，請參閱 [在中管理您的 Amazon ECR 服務配額 AWS 管理主控台](#)。

## HTTP 403: "User [arn] is not authorized to perform [operation]" (使用者 [arn] 未授權您執行此 [操作])

當您嘗試以 Amazon ECR 執行動作時，您可能會接收到下列錯誤：

```
$ aws ecr get-login-password
A client error (AccessDeniedException) occurred when calling the GetAuthorizationToken
operation:
 User: arn:aws:iam::account-number:user/username is not authorized to perform:
 ecr:GetAuthorizationToken on resource: *
```

這表示您的使用者沒有被授予使用 Amazon ECR 的權限，或者那些權限未正確設定。特別是，如果您正在針對一個 Amazon ECR 儲存庫執行動作，確認使用者已被授予權限以存取該儲存庫。如需相關建立和驗證 Amazon ECR 許可的詳細資訊，請參閱 [Amazon Elastic Container Registry 的 Identity and Access Management](#)。

## HTTP 404：「儲存庫不存在」錯誤

如果您指定的 Docker Hub 儲存庫目前尚未存在，Docker Hub 會自動建立。使用 Amazon ECR，新的儲存庫必須在可使用前確實建立。這會避免新的儲存庫被意外的建立 (例如輸入了錯別字)，並且也確

為了適當的安全性存取政策確實的指派至任何新的儲存庫。如需有關建立儲存庫的詳細資訊，請參閱 [Amazon ECR 私有儲存庫](#)。

## 錯誤：無法從非 TTY 裝置執行互動式登入

如果您收到錯誤訊息 `Cannot perform an interactive login from a non TTY device`，以下疑難排解步驟應該會有所幫助。

- 確認您使用的是第 2 AWS CLI 版，而且系統上沒有衝突的第 1 AWS CLI 版。如需詳細資訊，請參閱 [安裝或更新最新版本的 AWS CLI](#)。
- 確認您已 AWS CLI 使用有效的登入資料設定。如需詳細資訊，請參閱 [安裝或更新最新版本的 AWS CLI](#)。
- 驗證 AWS CLI 命令的語法是否正確。

# 搭配 Amazon ECR 使用 Podman

Podman 搭配 Amazon ECR 使用 可讓組織利用 的安全性和簡單性，Podman同時受益於 Amazon ECR 用於容器映像管理的可擴展性和可靠性。透過遵循概述的步驟和命令，開發人員和管理員可以簡化其容器工作流程、增強安全性，並最佳化資源使用率。隨著容器化持續獲得動力，使用 Podman 和 Amazon ECR 提供強大且靈活的解決方案，用於管理和部署容器化應用程式。

## 使用 Podman 向 Amazon ECR 驗證

使用 與 Amazon ECR 互動之前 Podman，需要身分驗證。這可以透過執行 `aws ecr get-login-password` 命令來擷取身分驗證字符，然後使用該字符搭配 `podman login` 命令來向 Amazon ECR 進行身分驗證來實現。

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

## 搭配 Podman 使用 Amazon ECR 登入資料協助程式

Amazon ECR 提供可與 Podman 搭配使用的 Docker 登入資料協助程式。登入資料協助程式可讓您在將映像推送和提取至 Amazon ECR 時，更輕鬆地存放和使用 Docker 登入資料。如要了解安裝和設定步驟，請參閱 [Amazon ECR Docker 憑證協助程式](#)。

### Important

Podman 僅部分支援 docker-creds-helper 規格。Podman 支援 Docker 組態中的 credHelpers 關鍵字，但不支援 credsStore 關鍵字。

若要搭配 Podman 使用 Amazon ECR 登入資料協助程式，請使用下列 credHelpers 格式設定 Docker 組態檔案：

```
{
 "credHelpers": {
 "public.ecr.aws": "ecr-login",
 "<aws_account_id>.dkr.ecr.<region>.amazonaws.com": "ecr-login"
 }
}
```

Podman 不支援下列 credsStore 組態：

```
{
 "credsStore": "ecr-login"
}
```

### Note

Amazon ECR Docker 憑證協助程式目前不支援多重要素驗證 (MFA)。

## 使用 Podman 從 Amazon ECR 提取映像

身分驗證成功後，可以使用 `podman pull` 命令搭配完整的 Amazon ECR 儲存庫 URI 從 Amazon ECR 提取容器映像。

```
podman pull aws_account_id.dkr.ecr.region.amazonaws.com/repository_name:tag
```

## 使用 執行 Amazon ECR 的容器 Podman

提取所需的映像後，可以使用 `podman run` 命令執行個體化容器。

```
podman run -d aws_account_id.dkr.ecr.region.amazonaws.com/repository_name:tag
```

## 使用 將映像推送至 Amazon ECR Podman

若要將本機映像推送至 Amazon ECR，必須先使用以 Amazon ECR 儲存庫 URI 標記映像 `podman tag`，然後使用 `podman push` 命令將映像上傳至 Amazon ECR。

```
podman
tag local_image:tag aws_account_id.dkr.ecr.region.amazonaws.com/repository_name:tag
podman push aws_account_id.dkr.ecr.region.amazonaws.com/repository_name:tag
```

## 文件歷史紀錄

下表說明自上次發行 Amazon ECR 後，對文件的重要變更。我們也會經常更新文件，以處理您傳送給我們的意見回饋。

| 變更                                   | 描述                                                                                                                                                                                                     | Date             |
|--------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------|
| 已完成以 Clair 為基礎的掃描棄用                  | 截至 2026 年 2 月 2 日，Clair 型映像掃描已棄用，且所有 ECR 帳戶都已遷移至 AWS 原生基本掃描。文件已更新，移除 Clair 特定的內容，並反映 AWS 原生現在是唯一基本掃描的實作。                                                                                               | 2026 年 2 月 2 日   |
| ECR 受管簽署                             | Amazon ECR 現在支援受管容器映像簽署，以增強您的安全狀態，並消除設定簽署的操作開銷。容器映像簽署可讓您驗證映像是否來自信任的來源。如需詳細資訊，請參閱 <a href="#">受管簽署</a> 。                                                                                                | 2025 年 11 月 21 日 |
| IPv6 支援 for AWS PrivateLink (VPC 端點) | 新增對採用 AWS PrivateLink 技術的 Amazon ECR VPC 端點的雙堆疊 (IPv4 和 IPv6) 連線支援。您現在可以建立雙堆疊 VPC 端點，以處理透過 IPv4 和 IPv6 私有 IP 地址的流量。如需詳細資訊，請參閱 <a href="#">Amazon ECR 介面 VPC 端點 (AWS PrivateLink)</a> 。                 | 2025 年 11 月 21 日 |
| ECR 封存功能                             | Amazon ECR 已新增對封存映像以進行長期保留的支援。如需詳細資訊，請參閱在 <a href="#">Amazon ECR 中封存映像</a> 。                                                                                                                           | 2025 年 11 月 19 日 |
| 更新以包含影像標籤不可變排除模式的支援                  | Amazon ECR 已更新映像標記功能，以在建立和更新儲存庫時包含映像標籤不可變性排除模式。您現在可以透過定義萬用字元模式 (例如 latest"、v*.beta、dev-*) 來指定可在儲存庫上啟用標籤不可變性時更新的標籤，以從不可變規則中排除特定標籤，同時維持所有其他標籤的不可變性。如需詳細資訊，請參閱 <a href="#">建立 Amazon ECR 私有儲存庫以存放映像</a> 。 | 2025 年 7 月 23 日  |
| 更新增強型影像掃描，以提供影像用量洞察                  | Amazon ECR 已更新增強型影像掃描功能，以包含 Amazon EKS 和 Amazon ECS 上影像使用方式的可見                                                                                                                                         | 2025 年 6 月 16 日  |

| 變更                                                                    | 描述                                                                                                                                                                                                                                                                                                                                             | Date             |
|-----------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------|
|                                                                       | 性。如需詳細資訊，請參閱在 <a href="#">Amazon ECR 中掃描映像以取得作業系統和程式設計語言套件漏洞</a> 。                                                                                                                                                                                                                                                                             |                  |
| IPv6 支援                                                               | 新增支援使用 IPv4-only 和雙堆疊 (IPv4 和 IPv6) 端點向 Amazon ECR 登錄檔提出請求。如需詳細資訊，請參閱 <a href="#">向 Amazon ECR 登錄檔提出請求</a> 。                                                                                                                                                                                                                                   | 2025 年 4 月 30 日  |
| 新增 Amazon ECR 私有登錄支援以提取快取                                             | Amazon ECR 新增支援為 Amazon ECR 私有登錄檔建立提取快取規則。如需詳細資訊，請參閱 <a href="#">將上游登錄檔與 Amazon ECR 私有登錄檔同步及用於提取快取的 Amazon ECR 服務連結角色</a> 。                                                                                                                                                                                                                    | 2025 年 3 月 12 日  |
| 新增設定登錄政策範圍的支援                                                         | Amazon ECR 新增了為私有登錄檔設定登錄政策範圍的支援。如需詳細資訊，請參閱 <a href="#">Amazon ECR 和 Amazon ECR 私有登錄檔中的私有登錄檔許可</a> 。<br><a href="https://docs.aws.amazon.com/AmazonECR/latest/userguide/Registries.html#registry-permissions-account-settings">https://docs.aws.amazon.com/AmazonECR/latest/userguide/Registries.html#registry-permissions-account-settings</a> | 2024 年 12 月 23 日 |
| <a href="#">AmazonEC2ContainerRegistryPullOnly</a> – 新政策              | Amazon ECR 新增了新的政策，將僅限提取許可授予 Amazon ECR。                                                                                                                                                                                                                                                                                                       | 2024 年 10 月 10 日 |
| CloudTrail 事件中的 Docker/OCI 用戶端代理操作現在指向 <code>ecr.amazonaws.com</code> | 值會 <code>ecr.amazonaws.com</code> 取代與 Docker/OCI 用戶端端點相關聯之 CloudTrail 事件 AWS Internal 的使用者代理程式 (userAgent) 和來源 IP 地址 (sourceIPAddress) 欄位中的值。如需範例，請參閱 <a href="#">範例：映像提取動作</a> 和 <a href="#">範例：映像推送動作</a> 。                                                                                                                                  | 2024 年 7 月 1 日   |
| 新增儲存庫建立範本之新 Amazon ECR 服務連結角色的說明。                                     | Amazon ECR 使用名為 <code>AWSServiceRoleForECRTemplate</code> 的服務連結角色，授予 Amazon ECR 代表您執行動作以完成儲存庫建立範本動作的許可。如需詳細資訊，請參閱 <a href="#">適用於儲存庫建立範本的 Amazon ECR 服務連結角色</a> 。                                                                                                                                                                              | 2024 年 6 月 20 日  |

| 變更                                                                 | 描述                                                                                                                                           | Date             |
|--------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------|------------------|
| 新增ECRTemplateServiceRolePolicy 服務連結角色。                             | 新增ECRTemplateServiceRolePolicy 服務連結角色。如需詳細資訊，請參閱 <a href="#">ECRTemplateServiceRolePolicy</a>                                                | 2024 年 6 月 20 日  |
| 已將跨區域和跨帳戶複寫新增至中國區域。                                                | Amazon ECR 已將支援新增至中國區域，以篩選複寫的儲存庫。如需詳細資訊，請參閱 <a href="#">Amazon ECR 中的私有映像複寫</a>                                                              | 2024 年 5 月 15 日  |
| 新增 GitLab 容器登錄檔以提取快取規則                                             | Amazon ECR 新增了為 GitLab 容器登錄建立提取快取規則的支援。如需詳細資訊，請參閱 <a href="#">將上游登錄檔與 Amazon ECR 私有登錄檔同步</a> 。                                               | 2024 年 5 月 8 日   |
| Amazon ECR 生命週期政策新增萬用字元使用支援                                        | Amazon ECR 在生命週期政策規則中使用 tagPatternList 參數，藉以新增在生命週期政策中使用萬用字元的支援。如需詳細資訊，請參閱在 <a href="#">Amazon ECR 中使用生命週期政策來自動化映像的清除</a> 。                  | 2023 年 12 月 18 日 |
| Amazon ECR 儲存庫建立範本                                                 | Amazon ECR 新增儲存庫建立範本支援。如需詳細資訊，請參閱 <a href="#">用於控制提取快取、推送時建立或複寫動作期間建立的儲存庫的範本</a> 。                                                           | 2023 年 11 月 15 日 |
| Amazon ECR 提取快取已新增支援已驗證的上游登錄檔                                      | Amazon ECR 已新增支援，開放使用需驗證才能提取快取規則的上游登錄檔。如需詳細資訊，請參閱 <a href="#">將上游登錄檔與 Amazon ECR 私有登錄檔同步</a> 。                                               | 2023 年 11 月 15 日 |
| <a href="#">AWSECRPullThroughCache_ServiceRolePolicy</a> - 更新至現有政策 | Amazon ECR 將新的許可新增到 AWSECRPullThroughCache_ServiceRolePolicy 政策。這些許可允許 Amazon ECR 擷取 Secrets Manager 秘密的加密內容。這在使用提取快取規則從需要驗證的上游登錄檔快取映像時是必要的。 | 2023 年 11 月 15 日 |
| Amazon ECR 映像簽署                                                    | Amazon ECR 和 AWS Signer 新增了使用 Notary 用戶端建立和推送容器映像簽章的支援。如需詳細資訊，請參閱在 <a href="#">Amazon ECR 中簽署映像</a> 。                                        | 2023 年 6 月 6 日   |

| 變更                                         | 描述                                                                                                                                                                     | Date            |
|--------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------|
| 新增了 Kubernetes 容器登錄檔以提取快取規則                | Amazon ECR 新增了對 Kubernetes 容器登錄檔建立提取快取規則的支援。如需詳細資訊，請參閱 <a href="#">將上游登錄檔與 Amazon ECR 私有登錄檔同步</a> 。                                                                    | 2023 年 6 月 1 日  |
| Amazon ECR 增強型掃描持續時間支援                     | Amazon Inspector 新增在啟用增強型掃描時設定儲存庫監控的持續時間的支援。如需詳細資訊，請參閱 <a href="#">變更 Amazon Inspector 中影像的增強型掃描持續時間</a> 。                                                             | 2022 年 6 月 28 日 |
| Amazon ECR 向 Amazon CloudWatch 傳送儲存庫提取計數指標 | Amazon ECR 向 Amazon CloudWatch 傳送儲存庫提取計數指標。如需詳細資訊，請參閱 <a href="#">Amazon ECR 儲存庫指標</a> 。                                                                               | 2022 年 1 月 6 日  |
| 擴展複寫支援                                     | Amazon ECR 已新增用於篩選要複寫哪些儲存庫的支援。如需詳細資訊，請參閱 <a href="#">Amazon ECR 中的私有映像複寫</a> 。                                                                                         | 2021 年 9 月 21 日 |
| AWS Amazon ECR 的受管政策                       | Amazon ECR 新增了 AWS 受管政策的文件。如需詳細資訊，請參閱 <a href="#">AWS Amazon Elastic Container Registry 的受管政策</a> 。                                                                    | 2021 年 6 月 24 日 |
| 跨區域和跨帳戶複寫                                  | Amazon ECR 已新增為您的私有登錄檔設定複寫設定的支援。如需詳細資訊，請參閱 <a href="#">Amazon ECR 中的私有登錄檔設定</a> 。                                                                                      | 2020 年 12 月 8 日 |
| OCI 成品支援                                   | Amazon ECR 已新增推送和提取開放容器計畫 (OCI) 成品的支援。新的參數 artifactMediaType 已新增至 DescribeImages API 回應來指出成品的類型。<br><br>如需詳細資訊，請參閱 <a href="#">將 Helm Chart 推送至 Amazon ECR 私有儲存庫</a> 。 | 2020 年 8 月 24 日 |
| 靜態加密                                       | Amazon ECR 已新增對使用存放在 AWS Key Management Service (AWS KMS) 中的客戶受管金鑰的伺服器端加密為您的儲存庫設定加密的支援。<br><br>如需詳細資訊，請參閱 <a href="#">靜態加密</a> 。                                       | 2020 年 7 月 29 日 |

| 變更                        | 描述                                                                                                                                                                                             | Date             |
|---------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------|
| 多架構映像                     | <p>Amazon ECR 已新增對建立和推送用於多架構映像之 Docker 資訊清單列表的支援。</p> <p>如需詳細資訊，請參閱<a href="#">將多架構映像推送至 Amazon ECR 私有儲存庫</a>。</p>                                                                             | 2020 年 4 月 28 日  |
| Amazon ECR 用量指標           | <p>Amazon ECR 新增 CloudWatch 用量指標，可讓您查看帳戶的資源用量。您也可以從 CloudWatch 和 Service Quotas 主控台建立 CloudWatch 警示，以在用量接近您套用的服務配額時收到警示。</p> <p>如需詳細資訊，請參閱<a href="#">Amazon ECR 用量指標</a>。</p>                 | 2020 年 2 月 28 日  |
| 更新 Amazon ECR 服務配額        | <p>已更新 Amazon ECR 服務配額以包含每個 API 配額。</p> <p>如需詳細資訊，請參閱<a href="#">Amazon ECR 服務配額</a>。</p>                                                                                                      | 2020 年 2 月 19 日  |
| 新增 get-login-password 的命令 | <p>已新增 get-login-password 的支援，可提供簡單且安全的授權字串擷取方式。</p> <p>如需詳細資訊，請參閱<a href="#">使用授權字串</a>。</p>                                                                                                  | 2020 年 2 月 4 日   |
| 映像掃描                      | <p>新增對映像掃描的支援，這有助於識別容器映像中的軟體漏洞。Amazon ECR 使用開放原始碼 CoreOS Clair 專案的 Common Vulnerabilities and Exposures (CVE) 資料庫，並提供您掃描結果的清單。</p> <p>如需詳細資訊，請參閱<a href="#">掃描映像是否有 Amazon ECR 中的軟體漏洞</a>。</p> | 2019 年 10 月 24 日 |
| VPC 端點政策                  | <p>新增對 Amazon ECR 介面 VPC 端點設定 IAM 政策的支援。</p> <p>如需詳細資訊，請參閱<a href="#">為您的 Amazon ECR VPC 端點建立端點政策</a>。</p>                                                                                     | 2019 年 9 月 26 日  |

| 變更                                      | 描述                                                                                                                                                                                                     | Date             |
|-----------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------|
| 映像標籤可變性                                 | <p>新增支援將儲存庫設定為不可變的，以防止映像標籤被覆寫。</p> <p>如需詳細資訊，請參閱<a href="#">防止在 Amazon ECR 中覆寫映像標籤</a>。</p>                                                                                                            | 2019 年 7 月 25 日  |
| 介面 VPC 端點 (AWS PrivateLink)             | <p>新增支援設定採用 AWS PrivateLink 技術的介面 VPC 端點。這可讓您在 VPC 與 Amazon ECR 之間建立私有連線，而不需要透過網際網路、NAT 執行個體、VPN 連接或 Direct Connect 進行存取。</p> <p>如需詳細資訊，請參閱<a href="#">Amazon ECR 介面 VPC 端點 (AWS PrivateLink)</a>。</p> | 2019 年 1 月 25 日  |
| 資源標記                                    | <p>Amazon ECR 新增支援將中繼資料標籤新增至您的儲存庫。</p> <p>如需詳細資訊，請參閱<a href="#">在 Amazon ECR 中標記私有儲存庫</a>。</p>                                                                                                         | 2018 年 12 月 18 日 |
| Amazon ECR 名稱變更                         | <p>Amazon Elastic Container Registry 已重新命名 (之前稱為 Amazon EC2 Container Registry)。</p>                                                                                                                   | 2017 年 11 月 21 日 |
| 生命週期政策                                  | <p>Amazon ECR 生命週期政策可讓您指定儲存庫中映像的生命週期管理。</p> <p>如需詳細資訊，請參閱<a href="#">在 Amazon ECR 中使用生命週期政策來自動化映像的清除</a>。</p>                                                                                          | 2017 年 10 月 11 日 |
| Amazon ECR 對 Docker 映像資訊清單 2 結構描述 2 的支援 | <p>Amazon ECR 現在支援 Docker 映像工作資訊清單檔案 V2 結構描述 2 (需搭配 1.10 或更新版本的 Docker 使用)。</p> <p>如需詳細資訊，請參閱<a href="#">Amazon ECR 中的容器映像資訊清單格式支援</a>。</p>                                                            | 2017 年 1 月 27 日  |
| Amazon ECR 一般可用性                        | <p>Amazon Elastic Container Registry (Amazon ECR) 是安全、可擴展且可靠的受管 AWS Docker 登錄服務。</p>                                                                                                                   | 2015 年 12 月 21 日 |

本文為英文版的機器翻譯版本，如內容有任何歧義或不一致之處，概以英文版為準。