在 上建置可擴展的 Web 爬取系統以用於 的 ESG 資料 AWS

AWS 方案指引

AWS 方案指引: 在 上建置可擴展的 Web 爬取系統以用於 的 ESG 資料 AWS

Table of Contents

簡介 1
目標對象 2
目標業務成果 2
架構 3
Web 爬蟲程式設計和操作 4
批次處理和資料處理 5
建置系統 6
準備資料集 6
建置 Web 爬蟲程式 7
擷取和處理 robots.txt 檔案 7
擷取和處理網站地圖 9
設計爬蟲程式 11
建置 AWS 基礎設施 18
最佳實務 20
Robots.txt 合規 20
爬蟲速率限制 20
使用者代理程式透明度 21
高效爬取 21
自適應方法 21
錯誤處理 21
批次爬取 21
安全 21
其他考量 22
常見問答集 23
如果無法使用 robots.txt 檔案該怎麼辦？ 23
如果 sitemaps.xml 檔案無法使用該怎麼辦？ 23
我可以使用無伺服器解決方案，而不是 Amazon EC2 或 Amazon ECS 嗎？ 23
為什麼爬蟲程式取得 403 狀態碼？ 24
後續步驟和資源 25
資源 25
工具 25
文件歷史紀錄 26
詞彙表 27
..... 27

A 27
B 30
C 31
D 34
E 37
F 39
G 40
H 41
I 42
L 44
M 45
O 49
P 51
Q 53
R 53
S 56
T 59
U 60
V 61
W 61
Z 62
..... lxiii

建置可擴展的 Web 爬取系統，以用於 上的 ESG 資料 AWS

Vijit Vashishtha 和 Mansi Doshi，Amazon Web Services

2025 年 1 月 (文件歷史記錄)

在評估潛在投資時，環境、社會和控管 (ESG) 因素是投資者的重要考量：

- 環境 – 專注於公司對自然世界的影響。它包含碳排放、資源管理和能源效率等因素。
- 社交 – 檢查公司如何管理與員工、供應商、客戶和社群的關係。它涵蓋了人力實務、多樣性和社群參與等層面。
- 控管 – 著眼於公司的領導階層、內部控制和擁有者權利。其中包括董事會組成、主管補償和商業道德。

具有強大 ESG 實務的公司在長期永續性和獲利能力方面，越來越被視作更好的定位。投資者對 ESG 資訊的需求不斷增加。能夠透過可靠、實用的 ESG 資料來證明其永續性登入資料的公司，更能吸引資金並保持競爭力。公司透過各種來源發佈 ESG 資料，例如新聞、文章和年報。由於此資訊是分散的，Web 爬蟲程式可協助您有效率地收集此資料。

此完整指南示範如何使用 AWS Fargate、Amazon Elastic Compute Cloud (Amazon EC2)、 AWS Batch和 Amazon Simple Storage Service (Amazon S3) 來建置強大、可擴展且負責任的資料收集管道。它討論了以下內容：

- 使用下列 架構可擴展的爬蟲系統 AWS 服務：
  - 用於執行爬蟲程式應用程式的 Fargate 或 Amazon EC2
  - AWS Batch 可有效率地協調大規模爬蟲任務
  - Amazon S3 提供安全且耐用的資料儲存
- 實作道德爬蟲的最佳實務，包括：
  - 遵守 robots.txt 和網站政策
  - 管理速率限制，以避免造成目標網站負擔過重
  - 確保資料隱私權和負責任地使用所收集的資訊
- 開發針對 AWS 基礎設施最佳化的 Python型爬蟲程式
- 最佳化爬蟲程式效能，同時維持安全標準

目標對象

本指南適用於希望從公有網站有效收集大量up-to-date ESG 資料的資料工程師和雲端架構師。它與涉及市場分析、永續財務評估或財務研究的專案特別相關。

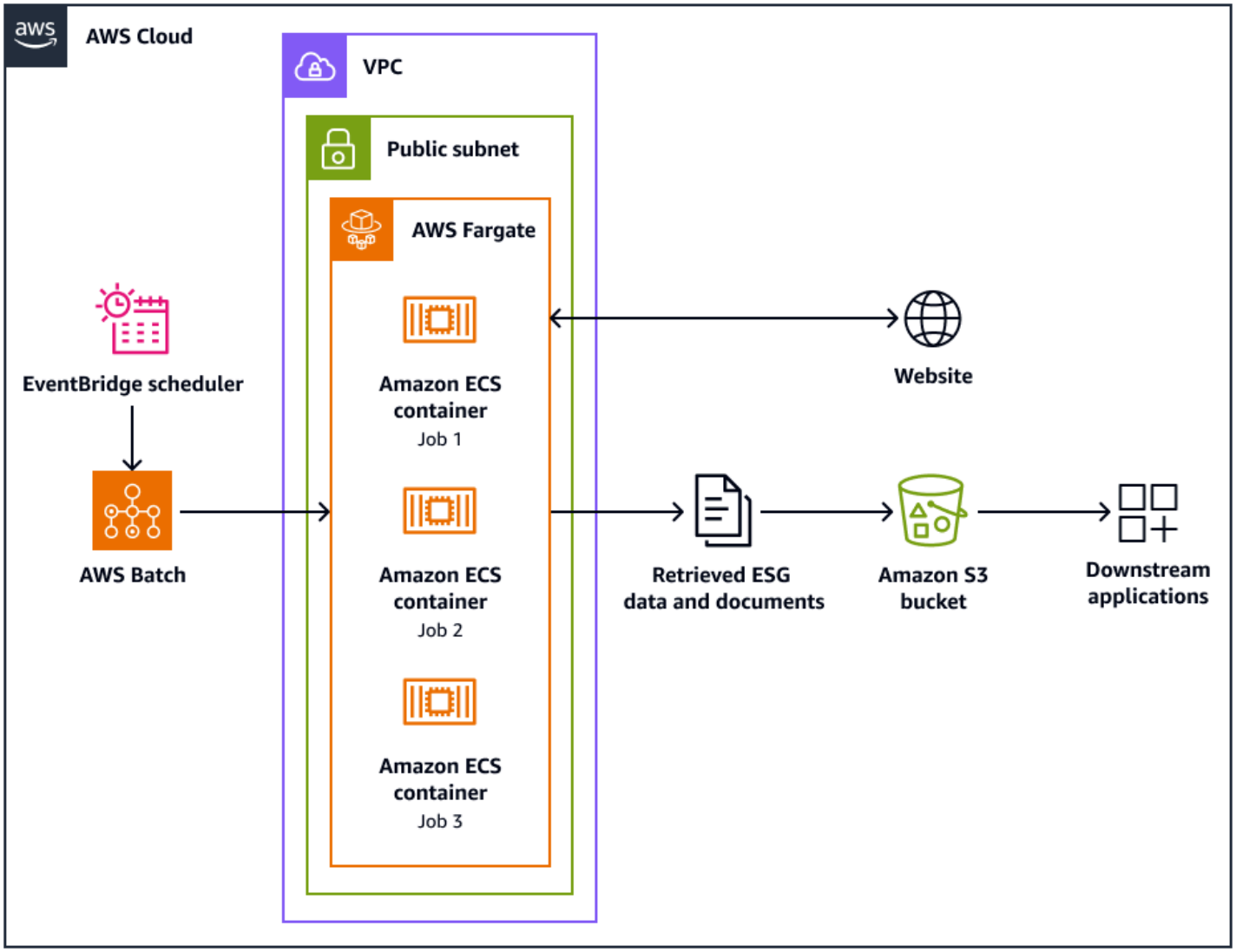
目標業務成果

以下是公司使用 ESG 資料的常見原因：

- 風險管理 – ESG 資料可協助您識別和降低與環境、社交和控管問題相關的潛在風險。
- 吸引投資者 – 許多投資者現在在做出投資決策時會考慮 ESG 因素。他們會將強大的 ESG 實務視為長期永續性和獲利能力的指標。
- 評價管理 – 良好的 ESG 效能可以增強公司在客戶、員工和一般大眾之間的評價。
- 法規合規 – 隨著 ESG 相關法規的增加，採用 ESG 實務有助於公司在合規要求方面保持領先。
- 創新和效率 – 專注於 ESG 因素可以推動產品、服務和營運方面的創新。這可提升效率並節省成本。
- 競爭優勢 – 強大的 ESG 效能可以讓公司與其競爭對手區分開來，並開啟新的市場機會。
- 利益相關者參與 – ESG 實務可協助公司更好地與各種利益相關者互動並滿足其期望，包括員工、客戶和當地社群。

上可擴展 Web 爬蟲系統的架構 AWS

下列架構圖顯示 Web 爬蟲程式系統，其設計旨在以符合道德的方式從網站擷取環境、社交和管理 (ESG) 資料。您可以使用針對 AWS 基礎設施最佳化的 Python型爬蟲程式。您可以使用 AWS Batch 來協調大規模爬蟲任務，並使用 Amazon Simple Storage Service (Amazon S3) 進行儲存。下游應用程式可以從 Amazon S3 儲存貯體擷取和存放資料。



該圖顯示以下工作流程：

1. Amazon EventBridge 排程器會以您排程的間隔啟動爬取程序。
2. AWS Batch 管理 Web 爬蟲程式任務的執行。 AWS Batch 任務佇列會保留和協調待定的爬蟲任務。

3. Web 爬取任務會在 Amazon Elastic Container Service (Amazon ECS) 容器中執行 AWS Fargate。任務會在虛擬私有雲端 (VPC) 的公有子網路中執行。
4. Web 爬蟲程式會爬取目標網站，並擷取 ESG 資料和文件，例如 PDF、CSV 或其他文件檔案。
5. Web 爬蟲程式會將擷取的資料和原始檔案存放在 Amazon S3 儲存貯體中。
6. 其他系統或應用程式會擷取或處理 Amazon S3 儲存貯體中儲存的資料和檔案。

Web 爬蟲程式設計和操作

有些網站專為在桌上型電腦或行動裝置上執行而設計。Web 爬蟲程式旨在支援使用桌面使用者代理程式或行動使用者代理程式。這些客服人員可協助您成功向目標網站提出請求。

初始化 Web 爬蟲程式之後，它會執行下列操作：

1. Web 爬蟲程式會呼叫 `setup()`方法。此方法會擷取和剖析 robots.txt 檔案。

> **Note**
>
> 您也可以設定 Web 爬蟲程式來擷取和剖析網站地圖。

2. Web 爬蟲程式會處理 robots.txt 檔案。如果在 robots.txt 檔案中指定爬蟲延遲，Web 爬蟲程式會擷取桌面使用者代理程式的爬蟲延遲。如果未於 robots.txt 檔案中指定爬蟲程式延遲，則 Web 爬蟲程式會使用隨機延遲。
3. Web 爬蟲程式會呼叫 `crawl()`方法，以啟動爬蟲程序。如果佇列中沒有 URLs，則會新增開始 URL。

> **Note**
>
> 爬蟲程式會持續進行，直到達到最大頁數或耗盡要爬蟲URLs。

4. 爬蟲程式會處理 URLs。對於佇列中的每個 URL，爬蟲程式會檢查 URL 是否已爬取。
5. 如果尚未抓取 URL，爬蟲程式會呼叫 `crawl_url()`方法，如下所示：
   a. 爬蟲程式會檢查 robots.txt 檔案，以判斷是否可以使用桌面使用者代理程式來爬取 URL。
   b. 如果允許，爬蟲程式會嘗試使用桌面使用者代理程式來抓取 URL。
   c. 如果不允許或桌面使用者代理程式無法爬取，則爬蟲程式會檢查 robots.txt 檔案，以判斷是否可以使用行動使用者代理程式來爬取 URL。
   d. 如果允許，爬蟲程式會嘗試使用行動使用者代理程式來抓取 URL。

6. 爬蟲程式會呼叫 `attempt_crawl()`方法，以擷取和處理內容。爬蟲程式會將 GET 請求傳送至具有適當標頭的 URL。如果請求失敗，爬蟲程式會使用重試邏輯。
7. 如果檔案是 HTML 格式，爬蟲程式會呼叫 `extract_esg_data()`方法。它使用 Beautiful Soup來剖析 HTML 內容。它使用關鍵字比對來擷取環境、社交和管理 (ESG) 資料。

   如果檔案是 PDF，爬蟲程式會呼叫 `save_pdf()`方法。爬蟲程式會下載 PDF 檔案並將其儲存至 Amazon S3 儲存貯體。
8. 爬蟲程式會呼叫 `extract_news_links()`方法。這會尋找並儲存新聞文章、新聞發佈和部落格文章的連結。
9. 爬蟲程式會呼叫 `extract_pdf_links()`方法。這會識別並存放 PDF 文件的連結。
10. 爬蟲程式會呼叫 `is_relevant_to_sustainable_finance()`方法。這會使用預先定義的關鍵字，檢查新聞或文章是否與永續金融相關。
11. 每次爬蟲嘗試後，爬蟲程式會使用 `delay()`方法實作延遲。如果在 robots.txt 檔案中指定了延遲，則會使用該值。否則，它會使用介於 1 到 3 秒之間的隨機延遲。
12. 爬蟲程式會呼叫 `save_esg_data()`方法，將 ESG 資料儲存至 CSV 檔案。CSV 檔案會儲存在 Amazon S3 儲存貯體中。
13. 爬蟲程式會呼叫 `save_news_links()`方法，將新聞連結儲存到 CSV 檔案，包括相關性資訊。CSV 檔案會儲存在 Amazon S3 儲存貯體中。
14. 爬蟲程式會呼叫 `save_pdf_links()`方法，將 PDF 連結儲存至 CSV 檔案。CSV 檔案會儲存在 Amazon S3 儲存貯體中。

批次處理和資料處理

爬蟲程序是以結構化的方式組織和執行。 AWS Batch 會為每個公司指派任務，以便它們以批次方式平行執行。每個批次都著重於單一公司的網域和子網域，因為您已在資料集中識別它們。不過，相同批次中的任務會依序執行，因此不會讓網站產生太多請求。這有助於應用程式更有效率地管理爬取工作負載，並確保為每個公司擷取所有相關資料。

透過將 Web 爬取組織成公司特定的批次，此容器會將收集的資料化。這有助於防止某家公司的資料與其他公司的資料混合。

批次處理可協助應用程式有效地從 Web 收集資料，同時根據目標公司及其各自的 Web 網域，維持清晰的結構和資訊分隔。這種方法有助於確保所收集資料的完整性和可用性，因為它整齊地組織並與適當的公司和網域相關聯。

在 上建置可擴展的 Web 爬取系統 AWS

本節說明如何建置架構一節所述的 Web 爬蟲程式。它包含建立公司及其相關聯 Web 屬性之強大資料集的系統性方法。此資料集是爬蟲活動的基礎。然後，本節說明如何在 中建置符合道德的 Web 爬蟲程式Python。

主題

- 準備資料集
- 建置 Web 爬蟲程式
- 建置 AWS 基礎設施

準備資料集

如果您尚未這麼做，請準備要從中收集資訊之網站的詳細資料集。此資料集應包含網站 URL 網域名稱和相關的子網域名稱。本節提供建置此資料集的step-by-step程序。

準備資料集

1. 定義範圍 – 確定您關注的產業或產業。決定要包含的公司數量。並定義您希望收集有關這些公司的任何條件，例如員工人數、地點或收入。
2. 識別資料來源 – 識別您可以用來收集這些公司相關資訊的資訊來源。範例包括商業目錄（例如 Crunchbase、Bloomberg 或 Forbes)、股票交換（例如 TZ 和 NASDAQ)、產業特定的關聯或出版物，或政府資料庫（例如 SEC 檔案）。
3. 建立資料表 – 在您偏好的工具中，例如 Microsoft Excel、Google Sheets 或資料庫管理系統中，建立用於收集每個公司準則的資料表。為每個條件包含資料欄。至少包含公司名稱、主要網域、子網域、產業、大小和位置的資料欄。
4. 收集初始公司資訊 – 收集每個公司的下列資訊，並將其輸入您建立的表格中：
    - 公司名稱
    - 產業或產業
    - 公司規模（員工人數）
    - 營收
    - 公司總部的位置
5. 收集網域資訊 – 對於每個公司，從主要網站 URL 擷取主要網域名稱，例如 `example.com`。您可以使用 WHOIS 網域查詢工具來驗證網域資訊。

6. 收集子網域資訊 – 針對每個公司，研究已註冊的子網域，例如 `blog.example.com`。您可以使用子網域列舉工具，例如 Sublist3r、OWASP Amass 或 Subfinder。您可以執行 Google 偏離（透過搜尋 `site:example.com`)、使用dig命令或 DNS 查詢工具檢查 DNS 記錄，或是分析 SSL 或 TLS 憑證。
7. 驗證和清除資料 – 檢閱、驗證和標準化您已收集的資料。例如，移除任何重複的項目、從網域和子網域移除不必要的 URL 資訊，並確認所有網域和子網域都處於作用中狀態。
8. （選用）將子網域分類 – 您可以將子網域分類為類型。以下是您可能遇到的類別的一些範例：
   - 部落格，例如 `blog.example.com`
   - 支援或協助，例如 `support.example.com`或 `help.example.com`
   - 電子商務，例如 `shop.example.com`或 `store.example.com`
   - 開發人員資源，例如 `dev.example.com`或 `api.example.com`
   - 區域或位置，例如 `us.example.com`或 `uk.example.com`
9. （選用）新增相關的中繼資料 – 您可以在資料集中記錄任何相關的中繼資料。例如，您可以新增上次更新的日期、資訊來源或子網域準確性的可信度分數。
10. 實作版本控制 – 使用 Git 等版本控制系統來追蹤資料表隨時間的變化。定期備份資料集。
11. 維護資料表 – 設定排程，例如每季更新資料表。標準化並實作新增公司或移除不再需要的公司的程序。可能的話，自動探索子網域。

建置 Web 爬蟲程式

如 架構節所述，應用程式會分批執行，每個公司各執行一個。

主題

- 擷取和處理 robots.txt 檔案
- 擷取和處理網站地圖
- 設計爬蟲程式

擷取和處理 robots.txt 檔案

準備資料集之後，您需要確認網域是否具有 robots.txt 檔案。對於 Web 爬蟲程式和其他機器人，bots.txt 檔案會指出他們可以造訪的網站哪些區段。遵守此檔案中的指示，是以道德方式編目網站的重要最佳實務。如需詳細資訊，請參閱本指南中的道德網路爬蟲程式最佳實務。

擷取和處理 robots.txt 檔案

1. 如果您尚未這麼做，請在終端機中執行下列命令來安裝requests程式庫：

```
pip install requests
```

2. 執行以下指令碼。此指令碼會執行下列操作：

   - 它定義了一個將網域做為輸入的check_robots_txt函數。
   - 它會建構 robots.txt 檔案的完整 URL。
   - 它會將 GET 請求傳送至 robots.txt 檔案的 URL。
   - 如果請求成功 （狀態碼 200)，則存在 robots.txt 檔案。
   - 如果請求失敗或傳回不同的狀態碼，則 robots.txt 檔案不存在或無法存取。

```
import requests
from urllib.parse import urljoin
def check_robots_txt(domain):
    # Ensure the domain starts with a protocol
    if not domain.startswith(('http://', 'https://')):
        domain = 'https://' + domain
    # Construct the full URL for robots.txt
    robots_url = urljoin(domain, '/robots.txt')
    try:
        # Send a GET request to the robots.txt URL
        response = requests.get(robots_url, timeout=5)
        # Check if the request was successful (status code 200)
        if response.status_code == 200:
            print(f"robots.txt found at {robots_url}")
            return True
        else:
            print(f"No robots.txt found at {robots_url} (Status code:
 {response.status_code})")
            return False
    except requests.RequestException as e:
        print(f"Error checking {robots_url}: {e}")
        return False
```

> **Note**
>
> 此指令碼會處理網路錯誤或其他問題的例外狀況。

3. 如果存在 robots.txt 檔案，請使用下列指令碼來下載它：

```
import requests

def download(self, url):
    response = requests.get(url, headers=self.headers, timeout=5)
    response.raise_for_status()  # Raise an exception for non-2xx responses
    return response.text

def download_robots_txt(self):
    # Append '/robots.txt' to the URL to get the robots.txt file's URL
    robots_url = self.url.rstrip('/') + '/robots.txt'
    try:
        response = download(robots_url)
        return response
    except requests.exceptions.RequestException as e:
        print(f"Error downloading robots.txt: {e}, \nGenerating sitemap using
 combinations...")
        return e
```

> **Note**
>
> 您可以根據您的使用案例自訂或修改這些指令碼。您也可以合併這些指令碼。

擷取和處理網站地圖

接下來，您需要處理網站地圖。您可以使用網站地圖將爬取重點放在重要頁面上。這可提高爬取效率。如需詳細資訊，請參閱本指南中的道德網路爬蟲程式最佳實務。

擷取和處理網站地圖

- 執行以下指令碼。此指令碼會定義以下`check_and_download_sitemap`函數：
  - 接受基本 URL、來自 robots.txt 的選用網站地圖 URL，以及使用者代理程式字串。
  - 檢查多個潛在的網站地圖位置，包括 robots.txt 中的位置 （如果提供）。

- 嘗試從每個位置下載網站地圖。
- 驗證下載的內容是否為 XML 格式。
- 呼叫 parse_sitemap函數以擷取 URLs。此函數：
  - 剖析網站地圖的 XML 內容。
  - 同時處理一般網站地圖和網站地圖索引檔案。
  - 如果遇到網站地圖索引，遞迴擷取子網站地圖。

```
import requests
from urllib.parse import urljoin
import xml.etree.ElementTree as ET

def check_and_download_sitemap(base_url, robots_sitemap_url=None,
 user_agent='SitemapBot/1.0'):
    headers = {'User-Agent': user_agent}
    sitemap_locations = [robots_sitemap_url, urljoin(base_url, '/sitemap.xml'),
 urljoin(base_url, '/sitemap_index.xml'),
        urljoin(base_url, '/sitemap/'), urljoin(base_url, '/sitemap/sitemap.xml')]

    for sitemap_url in sitemap_locations:
        if not sitemap_url:
            continue
        print(f"Checking for sitemap at: {sitemap_url}")
        try:
            response = requests.get(sitemap_url, headers=headers, timeout=10)
            if response.status_code == 200:
                content_type = response.headers.get('Content-Type', '')
                if 'xml' in content_type:
                    print(f"Successfully downloaded sitemap from {sitemap_url}")
                    return parse_sitemap(response.text)
                else:
                    print(f"Found content at {sitemap_url}, but it's not XML.
 Content-Type: {content_type}")
        except requests.RequestException as e:
            print(f"Error downloading sitemap from {sitemap_url}: {e}")
    print("No sitemap found.")
    return []

def parse_sitemap(sitemap_content):
    urls = []
    try:
```

```
        root = ET.fromstring(sitemap_content)
        # Handle both sitemap and sitemapindex
        for loc in root.findall('.//{http://www.sitemaps.org/schemas/
sitemap/0.9}loc'):
            urls.append(loc.text)

        # If it's a sitemap index, recursively fetch each sitemap
        if root.tag.endswith('sitemapindex'):
            all_urls = []
            for url in urls:
                print(f"Fetching sub-sitemap: {url}")
                sub_sitemap_urls = check_and_download_sitemap(url)
                all_urls.extend(sub_sitemap_urls)
            return all_urls
    except ET.ParseError as e:
        print(f"Error parsing sitemap XML: {e}")
    return urls


if __name__ == "__main__":
    base_url = input("Enter the base URL of the website: ")
    robots_sitemap_url = input("Enter the sitemap URL from robots.txt (or press
 Enter if none): ").strip() or None
    urls = check_and_download_sitemap(base_url, robots_sitemap_url)
    print(f"Found {len(urls)} URLs in sitemap:")
    for url in urls[:5]:  # Print first 5 URLs as an example
        print(url)
    if len(urls) > 5:
        print("...")
```

設計爬蟲程式

接著，您設計 Web 爬蟲程式。爬蟲程式旨在遵循本指南道德網路爬蟲程式的最佳實務中所述的最佳實務。此EthicalCrawler類別展示了道德爬蟲的幾個關鍵原則：

- 擷取和剖析 robots.txt 檔案 – 爬蟲程式會擷取目標網站的 robots.txt 檔案。
- 尊重爬蟲許可 – 爬蟲程式在爬蟲任何 URL 之前，會檢查 robots.txt 檔案中的規則是否允許該 URL 的爬蟲。如果不允許 URL，爬蟲程式會略過該 URL 並移至下一個 URL。
- 尊重爬蟲程式延遲 – 爬蟲程式會檢查 robots.txt 檔案中是否有爬蟲程式延遲指令。如果指定一個，爬蟲程式會在請求之間使用此延遲。否則，它會使用預設延遲。

- 使用者代理程式識別 – 爬蟲程式使用自訂使用者代理程式字串來識別網站本身。如有需要，網站擁有者可以設定特定規則來限制或允許您的爬蟲程式。
- 錯誤處理和正常降級 – 如果無法擷取或剖析 robots.txt 檔案，爬蟲程式會繼續進行保守的預設規則。它處理網路錯誤和非 200 HTTP 回應。
- 有限爬取 – 為了避免造成伺服器負擔過重，可以爬取的頁面數量有限制。

下列指令碼是虛擬程式碼，說明 Web 爬蟲程式的運作方式：

```
import requests
from urllib.parse import urljoin, urlparse
import time

class EthicalCrawler:
    def __init__(self, start_url, user_agent='EthicalBot/1.0'):
        self.start_url = start_url
        self.user_agent = user_agent
        self.domain = urlparse(start_url).netloc
        self.robots_parser = None
        self.crawl_delay = 1  # Default delay in seconds

    def can_fetch(self, url):
        if self.robots_parser:
            return self.robots_parser.allowed(url, self.user_agent)
        return True  # If no robots.txt, assume allowed but crawl conservatively

    def get_crawl_delay(self):
        if self.robots_parser:
            delay = self.robots_parser.agent(self.user_agent).delay
            if delay is not None:
                self.crawl_delay = delay
        print(f"Using crawl delay of {self.crawl_delay} seconds")

    def crawl(self, max_pages=10):
        self.get_crawl_delay()
        pages_crawled = 0
        urls_to_crawl = [self.start_url]
        while urls_to_crawl and pages_crawled < max_pages:
            url = urls_to_crawl.pop(0)
            if not self.can_fetch(url):
                print(f"robots.txt disallows crawling: {url}")
                continue
```

```
            try:
                response = requests.get(url, headers={'User-Agent': self.user_agent})
                if response.status_code == 200:
                    print(f"Successfully crawled: {url}")
                    # Here you would typically parse the content, extract links, etc.
                    # For this example, we'll just increment the counter
                    pages_crawled += 1
                else:
                    print(f"Failed to crawl {url}: HTTP {response.status_code}")
            except Exception as e:
                print(f"Error crawling {url}: {e}")

            # Respect the crawl delay
            time.sleep(self.crawl_delay)

        print(f"Crawling complete. Crawled {pages_crawled} pages.")
```

建置進階且符合道德的 Web 爬蟲程式，以收集 ESG 資料

1. 針對此系統中使用的進階道德網路爬蟲程式，複製下列程式碼範例：

```
import requests
from urllib.parse import urljoin, urlparse
import time
from collections import deque
import random
from bs4 import BeautifulSoup
import re
import csv
import os


class EnhancedESGCrawler:
    def __init__(self, start_url):
        self.start_url = start_url
        self.domain = urlparse(start_url).netloc
        self.desktop_user_agent = 'ESGEthicalBot/1.0'
        self.mobile_user_agent = 'Mozilla/5.0 (iPhone; CPU iPhone OS 14_0 like
 Mac OS X) AppleWebKit/605.1.15 (KHTML, like Gecko) Version/14.0 Mobile/15E148
 Safari/604.1'
        self.robots_parser = None
        self.crawl_delay = None
        self.urls_to_crawl = deque()
```

```
        self.crawled_urls = set()
        self.max_retries = 2
        self.session = requests.Session()
        self.esg_data = []
        self.news_links = []
        self.pdf_links = []

    def setup(self):
        self.fetch_robots_txt() # Provided in Previous Snippet
        self.fetch_sitemap() # Provided in Previous Snippet

    def can_fetch(self, url, user_agent):
        if self.robots_parser:
            return self.robots_parser.allowed(url, user_agent)
        return True

    def delay(self):
        if self.crawl_delay is not None:
            time.sleep(self.crawl_delay)
        else:
            time.sleep(random.uniform(1, 3))

    def get_headers(self, user_agent):
        return {'User-Agent': user_agent,
                'Accept': 'text/html,application/xhtml+xml,application/
xml;q=0.9,image/webp,*/*;q=0.8',
                'Accept-Language': 'en-US,en;q=0.5', 'Accept-Encoding': 'gzip,
 deflate, br', 'DNT': '1',
                'Connection': 'keep-alive', 'Upgrade-Insecure-Requests': '1'}

    def extract_esg_data(self, url, html_content):
        soup = BeautifulSoup(html_content, 'html.parser')
        esg_data = {
            'url': url,
            'environmental': self.extract_environmental_data(soup),
            'social': self.extract_social_data(soup),
            'governance': self.extract_governance_data(soup)
        }
        self.esg_data.append(esg_data)
        # Extract news links and PDFs
        self.extract_news_links(soup, url)
        self.extract_pdf_links(soup, url)

    def extract_environmental_data(self, soup):
```

```
        keywords = ['carbon footprint', 'emissions', 'renewable energy', 'waste
management', 'climate change']
        return self.extract_keyword_data(soup, keywords)

    def extract_social_data(self, soup):
        keywords = ['diversity', 'inclusion', 'human rights', 'labor practices',
'community engagement']
        return self.extract_keyword_data(soup, keywords)

    def extract_governance_data(self, soup):
        keywords = ['board structure', 'executive compensation', 'shareholder
rights', 'ethics', 'transparency']
        return self.extract_keyword_data(soup, keywords)

    def extract_keyword_data(self, soup, keywords):
        text = soup.get_text().lower()
        return {keyword: len(re.findall(r'\b' + re.escape(keyword) + r'\b', text))
for keyword in keywords}

    def extract_news_links(self, soup, base_url):
        news_keywords = ['news', 'press release', 'article', 'blog',
'sustainability']
        for a in soup.find_all('a', href=True):
            if any(keyword in a.text.lower() for keyword in news_keywords):
                full_url = urljoin(base_url, a['href'])
                if full_url not in self.news_links:
                    self.news_links.append({'url': full_url, 'text':
a.text.strip()})

    def extract_pdf_links(self, soup, base_url):
        for a in soup.find_all('a', href=True):
            if a['href'].lower().endswith('.pdf'):
                full_url = urljoin(base_url, a['href'])
                if full_url not in self.pdf_links:
                    self.pdf_links.append({'url': full_url, 'text':
a.text.strip()})

    def is_relevant_to_sustainable_finance(self, text):
        keywords = ['sustainable finance', 'esg', 'green bond', 'social impact',
'environmental impact',
                    'climate risk', 'sustainability report', 'corporate
responsibility']
        return any(keyword in text.lower() for keyword in keywords)
```

```
    def attempt_crawl(self, url, user_agent):
        for _ in range(self.max_retries):
            try:
                response = self.session.get(url,
 headers=self.get_headers(user_agent), timeout=10)
                if response.status_code == 200:
                    print(f"Successfully crawled: {url}")
                    if response.headers.get('Content-Type', '').startswith('text/
html'):
                        self.extract_esg_data(url, response.text)
                    elif response.headers.get('Content-Type',
 '').startswith('application/pdf'):
                        self.save_pdf(url, response.content)
                    return True
                else:
                    print(f"Failed to crawl {url}: HTTP {response.status_code}")
            except requests.RequestException as e:
                print(f"Error crawling {url} with {user_agent}: {e}")

            self.delay()
        return False

    def crawl_url(self, url):
        if not self.can_fetch(url, self.desktop_user_agent):
            print(f"Robots.txt disallows desktop user agent: {url}")
            if self.can_fetch(url, self.mobile_user_agent):
                print(f"Attempting with mobile user agent: {url}")
                return self.attempt_crawl(url, self.mobile_user_agent)
            else:
                print(f"Robots.txt disallows both user agents: {url}")
                return False

        return self.attempt_crawl(url, self.desktop_user_agent)

    def crawl(self, max_pages=100):
        self.setup()

        if not self.urls_to_crawl:
            self.urls_to_crawl.append(self.start_url)

        pages_crawled = 0
        while self.urls_to_crawl and pages_crawled < max_pages:
            url = self.urls_to_crawl.popleft()
            if url not in self.crawled_urls:
```

```
                if self.crawl_url(url):
                    pages_crawled += 1
                self.crawled_urls.add(url)
                self.delay()

        print(f"Crawling complete. Successfully crawled {pages_crawled} pages.")
        self.save_esg_data()
        self.save_news_links()
        self.save_pdf_links()

    def save_esg_data(self):
        with open('esg_data.csv', 'w', newline='', encoding='utf-8') as file:
            writer = csv.DictWriter(file, fieldnames=['url', 'environmental',
'social', 'governance'])
            writer.writeheader()
            for data in self.esg_data:
                writer.writerow({
                    'url': data['url'],
                    'environmental': ', '.join([f"{k}: {v}" for k, v in
data['environmental'].items()]),
                    'social': ', '.join([f"{k}: {v}" for k, v in
data['social'].items()]),
                    'governance': ', '.join([f"{k}: {v}" for k, v in
data['governance'].items()])
                })
        print("ESG data saved to esg_data.csv")

    def save_news_links(self):
        with open('news_links.csv', 'w', newline='', encoding='utf-8') as file:
            writer = csv.DictWriter(file, fieldnames=['url', 'text', 'relevant'])
            writer.writeheader()
            for news in self.news_links:
                writer.writerow({
                    'url': news['url'],
                    'text': news['text'],
                    'relevant':
self.is_relevant_to_sustainable_finance(news['text'])
                })
        print("News links saved to news_links.csv")

    def save_pdf_links(self):
        # Code for saving PDF in S3 or filesystem

    def save_pdf(self, url, content):
```

```
        # Code for saving PDF in S3 or filesystem

# Example usage
if __name__ == "__main__":
    start_url = input("Enter the starting URL to crawl for ESG data and news: ")
    crawler = EnhancedESGCrawler(start_url)
    crawler.crawl(max_pages=50)
```

2. 設定各種屬性，包括使用者代理程式、URLs的空集合，以及資料儲存清單。
3. 調整 `is_relevant_to_sustainable_finance()`方法中的關鍵字和相關性條件，以符合您的特定需求。
4. 請確定 robots.txt 檔案允許網路爬取網站，而且您使用 robots.txt 檔案中指定的網路爬取延遲和使用者代理程式。
5. 根據您的組織需要，考慮對提供的 Web 爬蟲程式指令碼進行下列自訂：
   - 實作 `fetch_sitemap()`方法來更有效率地探索 URL。
   - 增強生產用途的錯誤記錄和處理。
   - 實作更複雜的內容相關性分析。
   - 新增深度和廣度控制項以限制爬取範圍。

建置 AWS 基礎設施

您可以使用許多 AWS 服務 來建置網路爬取基礎設施。本指南的架構區段包含一個提議的解決方案。建議您考慮使用下列項目 AWS 服務 來建置 Web 爬蟲程式的支援基礎設施：

- 使用 Amazon Virtual Private Cloud (Amazon VPC) 建立 VPC 和子網路。
- 使用 Amazon EventBridge 排程器啟動爬取程序。
- 使用 AWS Batch 任務和任務佇列來管理 Web 爬蟲程式任務。
- 使用下列其中一個解決方案來執行 Web 爬蟲程式任務：
  - 上的 Amazon Elastic Container Service (Amazon ECS) 容器 AWS Fargate
  - Amazon Elastic Compute Cloud (Amazon EC2) 執行個體

> **Note**
>
> 如果您的應用程式可以處理中斷，請考慮透過 Spot Fleet 使用 Amazon EC2 Spot 執行個體。 https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/using-spot-instances.html

https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/work-with-spot-fleets.htmlSpot 執行個體機群可協助您大幅節省運算成本。

  - AWS Lambda 函數
- 將擷取的資料和原始檔案存放在 Amazon Simple Storage Service (Amazon S3) 儲存貯體中。

道德網路爬蟲程式的最佳實務

本節討論建置 Web 爬蟲應用程式以收集環境、社會和治理 (ESG) 資料的最佳實務和關鍵道德考量。透過遵循這些最佳實務，您可以保護您的專案和組織，並促進更負責任且永續的 Web 生態系統。這種方法可協助您存取寶貴的資料，並以尊重所有利益相關者的方式將其用於研究、業務和創新。

Robots.txt 合規

robots.txt 檔案用於網站，以與 Web 爬蟲程式和機器人進行通訊，了解哪些部分應該存取或不應存取。當 Web 爬蟲程式在網站上遇到 robots.txt 檔案時，它會剖析指示並相應地調整其爬蟲行為。這可防止爬蟲程式違反網站擁有者的指示，並維持網站與爬蟲程式之間的合作關係。因此，bots.txt 檔案有助於存取控制、保護敏感內容、負載管理和法律合規。

建議您遵循下列最佳實務：

- 請務必檢查並遵守 robots.txt 檔案中的規則。
- 爬取任何 URL 之前，請檢查桌面和行動使用者代理程式的規則。
- 如果網站僅允許行動使用者代理程式，請針對您的請求使用不同的代理程式標頭，例如行動代理程式標頭。

沒有 robots.txt 檔案不一定表示您無法或不應抓取網站。爬蟲應一律以負責任的方式完成，遵守網站的資源和擁有者的隱含權利。當 robots.txt 不存在時，以下是建議的最佳實務：

- 假設允許爬取，但請謹慎進行。
- 實作禮貌的爬蟲實務。
- 如果您打算執行大量爬取，請考慮聯絡網站擁有者以取得許可。

爬蟲速率限制

使用合理的網路爬取率，以避免伺服器過載。依 robots.txt 檔案或使用隨機延遲，在請求之間實作延遲。對於中小型網站，每 10-15 秒 1 次請求可能是適當的。對於較大的網站或具有明確爬蟲許可的網站，每秒 1-2 個請求可能是適當的。

使用者代理程式透明度

在使用者代理程式標頭中識別您的爬蟲程式。此 HTTP 標頭資訊旨在識別請求內容的裝置。一般而言， 機器人一詞會包含在 代理程式的名稱中。爬蟲程式和其他機器人有時會使用 標頭中的重要欄位來包含聯絡資訊。

高效爬取

使用網站擁有者開發的網站地圖，以專注於重要頁面。

自適應方法

如果桌面版本失敗，請將爬蟲程式設定為切換到行動使用者代理程式。這可以提供爬蟲程式存取，並減少網站伺服器上的壓力。

錯誤處理

確定爬蟲程式適當地處理各種 HTTP 狀態碼。例如，如果爬蟲程式遇到 429 狀態碼 (「請求太多」)，則爬蟲程式應暫停。如果爬蟲程式持續收到 403 個狀態碼 (「禁止」)，請考慮停止爬蟲。

批次爬取

我們建議您進行下列動作：

- 將任務分成較小的批次，而不是一次爬取所有 URLs。這有助於分配負載並降低遇到問題的風險，例如逾時或資源限制。
- 如果預期整體爬蟲任務會長時間執行，請考慮將其分成多個更小、更易於管理的任務。這可讓程序更具可擴展性和彈性。
- 如果要爬取URLs 數量相對較小，請考慮使用無伺服器解決方案，例如 AWS Lambda。Lambda 函數非常適合短期的事件驅動型任務，因為它們會自動擴展和處理資源管理。

安全

對於網路爬取運算任務，我們建議您將環境設定為僅允許傳出流量。這有助於透過將攻擊面降至最低並降低未經授權的傳入存取風險來增強安全性。僅允許傳出連線可讓爬蟲程序與目標網站通訊並擷取必要的資料，並限制任何可能危害系統的傳入流量。

其他考量

檢閱下列其他考量事項和最佳實務：

- 檢查網站服務條款或隱私權政策中的爬取準則。
- 在 HTML 中尋找可能提供爬蟲指令的`meta`標籤。
- 請注意您所在司法管轄區有關資料收集和使用的法律限制。
- 如果網站擁有者要求，請準備好停止爬取。

常見問答集

如果無法使用 robots.txt 檔案該怎麼辦？

沒有 robots.txt 檔案不一定表示您無法或不應抓取網站。爬蟲應一律以負責任的方式完成，遵守網站的資源和網站擁有者的隱含權利。

如果 sitemaps.xml 檔案無法使用該怎麼辦？

根據需求，您可以執行下列其中一項操作：

- 搜尋 HTML 網站地圖 – 尋找列出網站上重要頁面的 HTML 網站地圖頁面。這些通常會在頁尾中連結。
- 從首頁爬取 – 從首頁開始爬取，並遵循內部連結來探索其他頁面。
- 分析 URL 模式 – 分析網站的 URL 結構，以識別模式並以程式設計方式產生潛在的 URLs。
- 檢閱 robots.txt 檔案 – 檢查 robots.txt 檔案是否有任何不允許的頁面或目錄。這些可以提供有關網站結構的線索。
- 檢閱 API 端點 – 有些網站提供 API 端點，可用於擷取內容和結構資訊。
- 檢查搜尋引擎結果 – 使用搜尋引擎來尋找網站的索引頁面，方法是使用 網站：搜尋運算子，例如 `site:example.com`。
- 分析回連結 – 分析網站的回連結，以探索其他網站連結的重要頁面。
- 檢閱 Web 封存 – 檢查網際網路封存，例如 Wayback Machine，以取得可能有網站地圖或不同結構的舊版網站。
- 尋找內容管理系統 (CMS) 模式 – 如果您可以識別 CMS，請使用與該系統相關聯的常見 URL 模式。
- 確認 JavaScript 轉譯 – 如果網站高度依賴 JavaScript，請確定您的爬蟲程式可以轉譯 JavaScript 來探索動態載入的內容。對於某些網站，Sitemap.xml 檔案會在啟用 JavaScript 轉譯之後載入。

我可以使用無伺服器解決方案，而不是 Amazon EC2 或 Amazon ECS 嗎？

是。網路爬取的 AWS Lambda函數可以是可行的選項，特別是對於規模較小或模組化的爬取任務。不過，對於大規模、長時間執行的爬蟲操作，使用 Amazon Elastic Compute Cloud (Amazon EC2) 執行

個體或 Amazon Elastic Container Service (Amazon ECS) 的更傳統方法可能更合適。為網路爬取需求選擇正確的運算服務時，請務必仔細評估您的特定需求和權衡。

為什麼爬蟲程式取得 403 狀態碼？

HTTP 403 是 HTTP 狀態碼，表示禁止存取請求的資源。如果請求正確，伺服器就會了解請求，而且不會履行該請求。若要防止 403 狀態碼，您可以執行下列動作：

- 限制您的網路爬取率。
- 檢查 sitemap 或 robots.txt 檔案是否允許爬蟲程式存取 URL。
- 嘗試使用行動使用者代理程式，而非桌面使用者代理程式。

如果上述都無法運作，您應該遵守網站擁有者的決定，而不是編目頁面。

後續步驟和資源

收集原始環境、社會和管理 (ESG) 資料之後，您可以執行下列動作，從資料中擷取有意義的資訊：

1. 清除資料 – 資料可能包含大量與 ESG 因素和財務資料無關的不相關資訊。請務必移除此不相關的資料，並僅保留執行所需分析所需的資訊。您可以使用 yfinance 等工具來協助清理資料。
2. 擷取和轉換資料 – 從原始資料擷取相關特徵或變數，並將其轉換為適合分析的格式。您可以將資料轉換為表格格式，以提高可讀性和清晰度。您可以使用 之類的程式庫pandas來精簡資料。您也可以使用特徵工程、資料標準化和衍生指標來轉換資料。
3. 執行分析 – 您可以執行各種分析任務。這可能包括產生描述性統計資料、建立資料視覺化，以及進行探索性資料分析，以深入了解公司的 ESG 效能。
4. 套用機器學習 – 您可以使用清理和轉換的資料來訓練機器學習模型。這些模型可協助您識別目前正在展現財務永續性的公司，並預測其未來的永續性效能。

透過使用 Web 爬蟲程式和此資料評估程序，您可以有效地全面了解您正在評估的公司永續性實務和財務績效。您可以使用此資訊來告知投資決策、追蹤進度，以及支援永續商業實務。

資源

- 什麼是 Web 爬蟲程式？(Cloudflare 網站）
- ESG 投資指南 (Investopedia 網站）

工具

- Beautiful Soup (Beautiful Soup 文件）
- 處理表格式關聯式資料 (pandas 網站）

文件歷史紀錄

下表描述了本指南的重大變更。如果您想收到有關未來更新的通知，可以訂閱 RSS 摘要。

變更	描述	日期
初次出版	—	2025 年 1 月 6 日

AWS 規範性指引詞彙表

以下是 AWS Prescriptive Guidance 提供的策略、指南和模式中常用的術語。若要建議項目，請使用詞彙表末尾的提供意見回饋連結。

數字

7 R

將應用程式移至雲端的七種常見遷移策略。這些策略以 Gartner 在 2011 年確定的 5 R 為基礎，包括以下內容：

- 重構/重新架構 – 充分利用雲端原生功能來移動應用程式並修改其架構，以提高敏捷性、效能和可擴展性。這通常涉及移植作業系統和資料庫。範例：將您的現場部署 Oracle 資料庫遷移至 Amazon Aurora PostgreSQL 相容版本。
- 平台轉換 (隨即重塑) – 將應用程式移至雲端，並引入一定程度的優化以利用雲端功能。範例：將您的現場部署 Oracle 資料庫遷移至 中的 Amazon Relational Database Service (Amazon RDS) for Oracle AWS 雲端。
- 重新購買 (捨棄再購買) – 切換至不同的產品，通常從傳統授權移至 SaaS 模型。範例：將您的客戶關係管理 (CRM) 系統遷移至 Salesforce.com。
- 主機轉換 (隨即轉移) – 將應用程式移至雲端，而不進行任何變更以利用雲端功能。範例：將您的現場部署 Oracle 資料庫遷移至 中 EC2 執行個體上的 Oracle AWS 雲端。
- 重新放置 (虛擬機器監視器等級隨即轉移) – 將基礎設施移至雲端，無需購買新硬體、重寫應用程式或修改現有操作。您可以將伺服器從內部部署平台遷移到相同平台的雲端服務。範例：將 Microsoft Hyper-V應用程式遷移至 AWS。
- 保留 (重新檢視) – 將應用程式保留在來源環境中。其中可能包括需要重要重構的應用程式，且您希望將該工作延遲到以後，以及您想要保留的舊版應用程式，因為沒有業務理由來進行遷移。
- 淘汰 – 解除委任或移除來源環境中不再需要的應用程式。

A

ABAC

請參閱屬性型存取控制。

抽象服務

請參閱 受管服務。

ACID

請參閱原子性、一致性、隔離性、耐久性。

主動-主動式遷移

一種資料庫遷移方法，其中來源和目標資料庫保持同步 (透過使用雙向複寫工具或雙重寫入操作)，且兩個資料庫都在遷移期間處理來自連接應用程式的交易。此方法支援小型、受控制批次的遷移，而不需要一次性切換。它更靈活，但比主動-被動遷移需要更多的工作。

主動-被動式遷移

一種資料庫遷移方法，其中來源和目標資料庫保持同步，但只有來源資料庫處理來自連接應用程式的交易，同時將資料複寫至目標資料庫。目標資料庫在遷移期間不接受任何交易。

彙總函數

在一組資料列上運作的 SQL 函數，會計算群組的單一傳回值。彙總函數的範例包括 SUM和 MAX。

AI

請參閱人工智慧。

AIOps

請參閱人工智慧操作。

匿名化

在資料集中永久刪除個人資訊的程序。匿名化有助於保護個人隱私權。匿名資料不再被視為個人資料。

反模式

經常用於重複性問題的解決方案，其中解決方案具有反生產力、無效或比替代解決方案更有效。

應用程式控制

一種安全方法，僅允許使用核准的應用程式，以協助保護系統免受惡意軟體攻擊。

應用程式組合

有關組織使用的每個應用程式的詳細資訊的集合，包括建置和維護應用程式的成本及其商業價值。此資訊是產品組合探索和分析程序的關鍵，有助於識別要遷移、現代化和優化的應用程式並排定其優先順序。

人工智慧 (AI)

電腦科學領域，致力於使用運算技術來執行通常與人類相關的認知功能，例如學習、解決問題和識別模式。如需詳細資訊，請參閱什麼是人工智慧？

人工智慧操作 (AIOps)

使用機器學習技術解決操作問題、減少操作事件和人工干預以及提高服務品質的程序。如需有關如何在 AWS 遷移策略中使用 AIOps 的詳細資訊，請參閱操作整合指南。

非對稱加密

一種加密演算法，它使用一對金鑰：一個用於加密的公有金鑰和一個用於解密的私有金鑰。您可以共用公有金鑰，因為它不用於解密，但對私有金鑰存取應受到高度限制。

原子性、一致性、隔離性、耐久性 (ACID)

一組軟體屬性，即使在出現錯誤、電源故障或其他問題的情況下，也能確保資料庫的資料有效性和操作可靠性。

屬性型存取控制 (ABAC)

根據使用者屬性 (例如部門、工作職責和團隊名稱) 建立精細許可的實務。如需詳細資訊，請參閱《 AWS Identity and Access Management (IAM) 文件》中的 ABAC for AWS。

授權資料來源

您存放主要版本資料的位置，被視為最可靠的資訊來源。您可以將授權資料來源中的資料複製到其他位置，以處理或修改資料，例如匿名、修訂或假名化資料。

可用區域

中的不同位置 AWS 區域 ，可隔離其他可用區域中的故障，並提供相同區域中其他可用區域的低成本、低延遲網路連線。

AWS 雲端採用架構 (AWS CAF)

的指導方針和最佳實務架構 AWS ，可協助組織制定高效且有效的計劃，以成功地移至雲端。 AWS CAF 將指導方針組織到六個重點領域：業務、人員、治理、平台、安全和營運。業務、人員和控管層面著重於業務技能和程序；平台、安全和操作層面著重於技術技能和程序。例如，人員層面針對處理人力資源 (HR)、人員配備功能和人員管理的利害關係人。因此， AWS CAF 為人員開發、訓練和通訊提供指引，協助組織做好成功採用雲端的準備。如需詳細資訊，請參閱 AWS CAF 網站和 AWS CAF 白皮書。

AWS 工作負載資格架構 (AWS WQF)

一種工具，可評估資料庫遷移工作負載、建議遷移策略，並提供工作預估值。 AWS WQF 隨附於 AWS Schema Conversion Tool (AWS SCT)。它會分析資料庫結構描述和程式碼物件、應用程式程式碼、相依性和效能特性，並提供評估報告。

B

錯誤的機器人

旨在中斷或傷害個人或組織的機器人。

BCP

請參閱業務持續性規劃。

行為圖

資源行為的統一互動式檢視，以及一段時間後的互動。您可以將行為圖與 Amazon Detective 搭配使用來檢查失敗的登入嘗試、可疑的 API 呼叫和類似動作。如需詳細資訊，請參閱偵測文件中的行為圖中的資料。

大端序系統

首先儲存最高有效位元組的系統。另請參閱 Endianness。

二進制分類

預測二進制結果的過程 (兩個可能的類別之一)。例如，ML 模型可能需要預測諸如「此電子郵件是否是垃圾郵件？」等問題 或「產品是書還是汽車？」

Bloom 篩選條件

一種機率性、記憶體高效的資料結構，用於測試元素是否為集的成員。

藍/綠部署

一種部署策略，您可以在其中建立兩個不同但相同的環境。您可以在一個環境 （藍色） 中執行目前的應用程式版本，並在另一個環境 （綠色） 中執行新的應用程式版本。此策略可協助您快速復原，並將影響降至最低。

機器人

透過網際網路執行自動化任務並模擬人類活動或互動的軟體應用程式。有些機器人有用或有益，例如在網際網路上為資訊編製索引的 Web 爬蟲程式。有些其他機器人稱為惡意機器人，旨在中斷或傷害個人或組織。

殭屍網路

受到惡意軟體感染且受單一方控制之機器人的網路，稱為機器人繼承器或機器人運算子。殭屍網路是擴展機器人及其影響的最佳已知機制。

分支

程式碼儲存庫包含的區域。儲存庫中建立的第一個分支是主要分支。您可以從現有分支建立新分支，然後在新分支中開發功能或修正錯誤。您建立用來建立功能的分支通常稱為功能分支。當準備好發佈功能時，可以將功能分支合併回主要分支。如需詳細資訊，請參閱關於分支 (GitHub 文件)。

碎片存取

在特殊情況下，以及透過核准的程序，讓使用者能夠快速存取他們通常無權存取 AWS 帳戶 的 。如需詳細資訊，請參閱 Well-Architected 指南中的 AWS 實作打破玻璃程序指標。

棕地策略

環境中的現有基礎設施。對系統架構採用棕地策略時，可以根據目前系統和基礎設施的限制來設計架構。如果正在擴展現有基礎設施，則可能會混合棕地和綠地策略。

緩衝快取

儲存最常存取資料的記憶體區域。

業務能力

業務如何創造價值 (例如，銷售、客戶服務或營銷)。業務能力可驅動微服務架構和開發決策。如需詳細資訊，請參閱在 AWS上執行容器化微服務白皮書的圍繞業務能力進行組織部分。

業務連續性規劃 (BCP)

一種解決破壞性事件 (如大規模遷移) 對營運的潛在影響並使業務能夠快速恢復營運的計畫。

C

CAF

請參閱AWS 雲端採用架構。

Canary 部署

版本對最終使用者的緩慢和增量版本。當您有信心時，您可以部署新版本並完全取代目前的版本。

CCoE

請參閱 Cloud Center of Excellence。

CDC

請參閱變更資料擷取。

變更資料擷取 (CDC)

追蹤對資料來源 (例如資料庫表格) 的變更並記錄有關變更的中繼資料的程序。您可以將 CDC 用於各種用途，例如稽核或複寫目標系統中的變更以保持同步。

混沌工程

故意引入故障或破壞性事件，以測試系統的彈性。您可以使用 AWS Fault Injection Service (AWS FIS) 執行實驗，為您的 AWS 工作負載帶來壓力，並評估其回應。

CI/CD

請參閱持續整合和持續交付。

分類

有助於產生預測的分類程序。用於分類問題的 ML 模型可預測離散值。離散值永遠彼此不同。例如，模型可能需要評估影像中是否有汽車。

用戶端加密

在目標 AWS 服務 接收資料之前，在本機加密資料。

雲端卓越中心 (CCoE)

一個多學科團隊，可推動整個組織的雲端採用工作，包括開發雲端最佳實務、調動資源、制定遷移時間表以及領導組織進行大規模轉型。如需詳細資訊，請參閱 AWS 雲端 企業策略部落格上的 CCoE 文章。

雲端運算

通常用於遠端資料儲存和 IoT 裝置管理的雲端技術。雲端運算通常連接到邊緣運算技術。

雲端操作模型

在 IT 組織中，用於建置、成熟和最佳化一或多個雲端環境的操作模型。如需詳細資訊，請參閱建置您的雲端操作模型。

採用雲端階段

組織在遷移至 時通常會經歷的四個階段 AWS 雲端：

- 專案 – 執行一些與雲端相關的專案以進行概念驗證和學習用途
- 基礎 – 進行基礎投資以擴展雲端採用 (例如，建立登陸區域、定義 CCoE、建立營運模型)

- 遷移 – 遷移個別應用程式
- 重塑 – 優化產品和服務，並在雲端中創新

這些階段由 Stephen Orban 於部落格文章 The Journey Toward Cloud-First 和 Enterprise Strategy 部落格上的採用階段中定義。 AWS 雲端 如需有關它們如何與 AWS 遷移策略相關的詳細資訊，請參閱遷移整備指南。

CMDB

請參閱組態管理資料庫。

程式碼儲存庫

透過版本控制程序來儲存及更新原始程式碼和其他資產 (例如文件、範例和指令碼) 的位置。常見的雲端儲存庫包括 GitHub或 Bitbucket Cloud。程式碼的每個版本都稱為分支。在微服務結構中，每個儲存庫都專用於單個功能。單一 CI/CD 管道可以使用多個儲存庫。

冷快取

一種緩衝快取，它是空的、未填充的，或者包含過時或不相關的資料。這會影響效能，因為資料庫執行個體必須從主記憶體或磁碟讀取，這比從緩衝快取讀取更慢。

冷資料

很少存取且通常是歷史資料的資料。查詢這類資料時，通常可接受慢查詢。將此資料移至效能較低且成本較低的儲存層或類別，可以降低成本。

電腦視覺 (CV)

使用機器學習從數位影像和影片等視覺化格式分析和擷取資訊的 AI 欄位。例如，Amazon SageMaker AI 提供 CV 的影像處理演算法。

組態偏離

對於工作負載，組態會從預期狀態變更。這可能會導致工作負載變得不合規，而且通常是漸進和無意的。

組態管理資料庫 (CMDB)

儲存和管理有關資料庫及其 IT 環境的資訊的儲存庫，同時包括硬體和軟體元件及其組態。您通常在遷移的產品組合探索和分析階段使用 CMDB 中的資料。

一致性套件

您可以組合的 AWS Config 規則和修補動作集合，以自訂您的合規和安全檢查。您可以使用 YAML 範本，將一致性套件部署為 AWS 帳戶 和 區域中或整個組織的單一實體。如需詳細資訊，請參閱 AWS Config 文件中的一致性套件。

持續整合和持續交付 (CI/CD)

自動化軟體發行程序的來源、建置、測試、暫存和生產階段的程序。CI/CD 通常被描述為管道。CI/CD 可協助您將程序自動化、提升生產力、改善程式碼品質以及加快交付速度。如需詳細資訊，請參閱持續交付的優點。CD 也可表示持續部署。如需詳細資訊，請參閱持續交付與持續部署。

CV

請參閱電腦視覺。

D

靜態資料

網路中靜止的資料，例如儲存中的資料。

資料分類

根據重要性和敏感性來識別和分類網路資料的程序。它是所有網路安全風險管理策略的關鍵組成部分，因為它可以協助您確定適當的資料保護和保留控制。資料分類是 AWS Well-Architected Framework 中安全支柱的元件。如需詳細資訊，請參閱資料分類。

資料偏離

生產資料與用於訓練 ML 模型的資料之間有意義的變化，或輸入資料隨時間有意義的變更。資料偏離可以降低 ML 模型預測的整體品質、準確性和公平性。

傳輸中的資料

在您的網路中主動移動的資料，例如在網路資源之間移動。

資料網格

架構架構，提供分散式、分散式資料擁有權與集中式管理。

資料最小化

僅收集和處理嚴格必要資料的原則。在 中實作資料最小化 AWS 雲端 可以降低隱私權風險、成本和分析碳足跡。

資料周邊

AWS 環境中的一組預防性防護機制，可協助確保只有信任的身分才能從預期的網路存取信任的資源。如需詳細資訊，請參閱在 上建置資料周邊 AWS。

資料預先處理

將原始資料轉換成 ML 模型可輕鬆剖析的格式。預處理資料可能意味著移除某些欄或列，並解決遺失、不一致或重複的值。

資料來源

在整個生命週期中追蹤資料的原始伺服器和歷史記錄的程序，例如資料的產生、傳輸和儲存方式。

資料主體

正在收集和處理其資料的個人。

資料倉儲

支援商業智慧的資料管理系統，例如 分析。資料倉儲通常包含大量歷史資料，通常用於查詢和分析。

資料庫定義語言 (DDL)

用於建立或修改資料庫中資料表和物件之結構的陳述式或命令。

資料庫處理語言 (DML)

用於修改 (插入、更新和刪除) 資料庫中資訊的陳述式或命令。

DDL

請參閱資料庫定義語言。

深度整體

結合多個深度學習模型進行預測。可以使用深度整體來獲得更準確的預測或估計預測中的不確定性。

深度學習

一個機器學習子領域，它使用多層人工神經網路來識別感興趣的輸入資料與目標變數之間的對應關係。

深度防禦

這是一種資訊安全方法，其中一系列的安全機制和控制項會在整個電腦網路中精心分層，以保護網路和其中資料的機密性、完整性和可用性。當您在 上採用此策略時 AWS，您可以在 AWS Organizations 結構的不同層新增多個控制項，以協助保護資源。例如，defense-in-depth方法可能會結合多重要素驗證、網路分割和加密。

委派的管理員

在 中 AWS Organizations，相容的服務可以註冊 AWS 成員帳戶，以管理組織的帳戶和管理該服務的許可。此帳戶稱為該服務的委派管理員。如需詳細資訊和相容服務清單，請參閱 AWS Organizations 文件中的可搭配 AWS Organizations運作的服務。

部署

在目標環境中提供應用程式、新功能或程式碼修正的程序。部署涉及在程式碼庫中實作變更，然後在應用程式環境中建置和執行該程式碼庫。

開發環境

請參閱 環境。

偵測性控制

一種安全控制，用於在事件發生後偵測、記錄和提醒。這些控制是第二道防線，提醒您注意繞過現有預防性控制的安全事件。如需詳細資訊，請參閱在 AWS上實作安全控制中的偵測性控制。

開發值串流映射 (DVSM)

一種程序，用於識別對軟體開發生命週期中的速度和品質造成負面影響的限制並排定優先順序。DVSM 延伸了原本專為精簡製造實務設計的價值串流映射程序。它著重於透過軟體開發程序建立和移動價值所需的步驟和團隊。

數位分身

真實世界系統的虛擬呈現，例如建築物、工廠、工業設備或生產線。數位分身支援預測性維護、遠端監控和生產最佳化。

維度資料表

在星星結構描述中，較小的資料表包含有關事實資料表中量化資料的資料屬性。維度資料表屬性通常是文字欄位或離散數字，其行為類似於文字。這些屬性通常用於查詢限制、篩選和結果集標記。

災難

防止工作負載或系統在其主要部署位置中實現其業務目標的事件。這些事件可能是自然災難、技術故障或人為動作的結果，例如意外設定錯誤或惡意軟體攻擊。

災難復原 (DR)

您用來將災難造成的停機時間和資料遺失降至最低的策略和程序。如需詳細資訊，請參閱 AWS Well-Architected Framework 中的 上工作負載災難復原 AWS：雲端中的復原。

DML

請參閱資料庫處理語言。

領域驅動的設計

一種開發複雜軟體系統的方法，它會將其元件與每個元件所服務的不斷發展的領域或核心業務目標相關聯。Eric Evans 在其著作 Domain-Driven Design: Tackling Complexity in the Heart of Software (Boston: Addison-Wesley Professional, 2003) 中介紹了這一概念。如需有關如何將領域驅動的設計與 strangler fig 模式搭配使用的資訊，請參閱使用容器和 Amazon API Gateway 逐步現代化舊版 Microsoft ASP.NET (ASMX) Web 服務。

DR

請參閱災難復原。

偏離偵測

追蹤與基準組態的偏差。例如，您可以使用 AWS CloudFormation 來偵測系統資源中的偏離，也可以使用 AWS Control Tower 來偵測登陸區域中可能影響控管要求合規性的變更。 https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/using-cfn-stack-drift.html

DVSM

請參閱開發值串流映射。

E

EDA

請參閱探索性資料分析。

EDI

請參閱電子資料交換。

邊緣運算

提升 IoT 網路邊緣智慧型裝置運算能力的技術。與雲端運算相比，邊緣運算可以減少通訊延遲並改善回應時間。

電子資料交換 (EDI)

在組織之間自動交換商業文件。如需詳細資訊，請參閱什麼是電子資料交換。

加密

一種運算程序，可將人類可讀取的純文字資料轉換為加密文字。

加密金鑰

由加密演算法產生的隨機位元的加密字串。金鑰長度可能有所不同，每個金鑰的設計都是不可預測且唯一的。

端序

位元組在電腦記憶體中的儲存順序。大端序系統首先儲存最高有效位元組。小端序系統首先儲存最低有效位元組。

端點

請參閱 服務端點。

端點服務

您可以在虛擬私有雲端 (VPC) 中託管以與其他使用者共用的服務。您可以使用 建立端點服務，AWS PrivateLink 並將許可授予其他 AWS 帳戶 或 AWS Identity and Access Management (IAM) 委託人。這些帳戶或主體可以透過建立介面 VPC 端點私下連接至您的端點服務。如需詳細資訊，請參閱 Amazon Virtual Private Cloud (Amazon VPC) 文件中的建立端點服務。

企業資源規劃 (ERP)

一種系統，可自動化和管理企業的關鍵業務流程 （例如會計、MES 和專案管理）。

信封加密

使用另一個加密金鑰對某個加密金鑰進行加密的程序。如需詳細資訊，請參閱 AWS Key Management Service (AWS KMS) 文件中的信封加密。

環境

執行中應用程式的執行個體。以下是雲端運算中常見的環境類型：

- 開發環境 – 執行中應用程式的執行個體，只有負責維護應用程式的核心團隊才能使用。開發環境用來測試變更，然後再將開發環境提升到較高的環境。此類型的環境有時稱為測試環境。
- 較低的環境 – 應用程式的所有開發環境，例如用於初始建置和測試的開發環境。
- 生產環境 – 最終使用者可以存取的執行中應用程式的執行個體。在 CI/CD 管道中，生產環境是最後一個部署環境。
- 較高的環境 – 核心開發團隊以外的使用者可存取的所有環境。這可能包括生產環境、生產前環境以及用於使用者接受度測試的環境。

epic

在敏捷方法中，有助於組織工作並排定工作優先順序的功能類別。epic 提供要求和實作任務的高層級描述。例如， AWS CAF 安全概念包括身分和存取管理、偵測控制、基礎設施安全、資料保護和事件回應。如需有關 AWS 遷移策略中的 Epic 的詳細資訊，請參閱計畫實作指南。

ERP

請參閱企業資源規劃。

探索性資料分析 (EDA)

分析資料集以了解其主要特性的過程。您收集或彙總資料，然後執行初步調查以尋找模式、偵測異常並檢查假設。透過計算摘要統計並建立資料可視化來執行 EDA。

F

事實資料表

星狀結構描述中的中央資料表。它存放有關業務操作的量化資料。一般而言，事實資料表包含兩種類型的資料欄：包含度量的資料，以及包含維度資料表外部索引鍵的資料欄。

快速失敗

一種使用頻繁和增量測試來縮短開發生命週期的理念。這是敏捷方法的關鍵部分。

故障隔離界限

在 中 AWS 雲端，像是可用區域 AWS 區域、控制平面或資料平面等邊界會限制故障的影響，並有助於改善工作負載的彈性。如需詳細資訊，請參閱AWS 故障隔離界限。

功能分支

請參閱分支。

特徵

用來進行預測的輸入資料。例如，在製造環境中，特徵可能是定期從製造生產線擷取的影像。

功能重要性

特徵對於模型的預測有多重要。這通常表示為可以透過各種技術來計算的數值得分，例如 Shapley Additive Explanations (SHAP) 和積分梯度。如需詳細資訊，請參閱 的機器學習模型可解譯性 AWS。

特徵轉換

優化 ML 程序的資料，包括使用其他來源豐富資料、調整值、或從單一資料欄位擷取多組資訊。這可讓 ML 模型從資料中受益。例如，如果將「2021-05-27 00:15:37」日期劃分為「2021」、「五月」、「週四」和「15」，則可以協助學習演算法學習與不同資料元件相關聯的細微模式。

少量擷取提示

在要求 LLM 執行類似的任務之前，提供少量示範任務和所需輸出的範例。此技術是內容內學習的應用程式，其中模型會從內嵌在提示中的範例 (快照) 中學習。對於需要特定格式、推理或網域知識的任務，少量的提示非常有效。另請參閱零鏡頭提示。

FGAC

請參閱精細存取控制。

精細存取控制 (FGAC)

使用多個條件來允許或拒絕存取請求。

閃切遷移

一種資料庫遷移方法，透過變更資料擷取使用連續資料複寫，以盡可能在最短的時間內遷移資料，而不是使用分階段方法。目標是將停機時間降至最低。

FM

請參閱基礎模型。

基礎模型 (FM)

大型深度學習神經網路，已針對廣義和未標記資料的大量資料集進行訓練。FMs 能夠執行各種一般任務，例如了解語言、產生文字和影像，以及以自然語言交談。如需詳細資訊，請參閱什麼是基礎模型。

G

生成式 AI

已針對大量資料進行訓練的 AI 模型子集，可使用簡單的文字提示建立新的內容和成品，例如影像、影片、文字和音訊。如需詳細資訊，請參閱什麼是生成式 AI。

地理封鎖

請參閱地理限制。

地理限制 (地理封鎖)

Amazon CloudFront 中的選項，可防止特定國家/地區的使用者存取內容分發。您可以使用允許清單或封鎖清單來指定核准和禁止的國家/地區。如需詳細資訊，請參閱 CloudFront 文件中的限制內容的地理分佈。

Gitflow 工作流程

這是一種方法，其中較低和較高環境在原始碼儲存庫中使用不同分支。Gitflow 工作流程會被視為舊版，而以幹線為基礎的工作流程是現代、偏好的方法。

黃金影像

系統或軟體的快照，做為部署該系統或軟體新執行個體的範本。例如，在製造中，黃金映像可用於在多個裝置上佈建軟體，並有助於提高裝置製造操作的速度、可擴展性和生產力。

綠地策略

新環境中缺乏現有基礎設施。對系統架構採用綠地策略時，可以選擇所有新技術，而不會限制與現有基礎設施的相容性，也稱為棕地。如果正在擴展現有基礎設施，則可能會混合棕地和綠地策略。

防護機制

有助於跨組織單位 (OU) 來管控資源、政策和合規的高層級規則。預防性防護機制會強制執行政策，以確保符合合規標準。透過使用服務控制政策和 IAM 許可界限來將其實作。偵測性防護機制可偵測政策違規和合規問題，並產生提醒以便修正。它們是透過使用 AWS Config AWS Security Hub、Amazon GuardDuty、、Amazon Inspector AWS Trusted Advisor和自訂 AWS Lambda 檢查來實作。

H

HA

請參閱高可用性。

異質資料庫遷移

將來源資料庫遷移至使用不同資料庫引擎的目標資料庫 (例如，Oracle 至 Amazon Aurora)。異質遷移通常是重新架構工作的一部分，而轉換結構描述可能是一項複雜任務。AWS 提供有助於結構描述轉換的 AWS SCT。

高可用性 (HA)

在遇到挑戰或災難時，工作負載能夠在不介入的情況下持續運作。HA 系統的設計目的是自動容錯移轉、持續提供高品質的效能，並處理不同的負載和故障，並將效能影響降至最低。

歷史現代化

一種方法，用於現代化和升級操作技術 (OT) 系統，以更好地滿足製造業的需求。歷史資料是一種資料庫，用於從工廠中的各種來源收集和存放資料。

保留資料

從用於訓練機器學習模型的資料集中保留的部分歷史標記資料。您可以使用保留資料，透過比較模型預測與保留資料來評估模型效能。

異質資料庫遷移

將您的來源資料庫遷移至共用相同資料庫引擎的目標資料庫 (例如，Microsoft SQL Server 至 Amazon RDS for SQL Server)。同質遷移通常是主機轉換或平台轉換工作的一部分。您可以使用原生資料庫公用程式來遷移結構描述。

熱資料

經常存取的資料，例如即時資料或最近的轉譯資料。此資料通常需要高效能儲存層或類別，才能提供快速的查詢回應。

修補程序

緊急修正生產環境中的關鍵問題。由於其緊迫性，通常會在典型 DevOps 發行工作流程之外執行修補程式。

超級護理期間

在切換後，遷移團隊在雲端管理和監控遷移的應用程式以解決任何問題的時段。通常，此期間的長度為 1-4 天。在超級護理期間結束時，遷移團隊通常會將應用程式的責任轉移給雲端營運團隊。

I

IaC

將基礎設施視為程式碼。

身分型政策

連接至一或多個 IAM 主體的政策，可定義其在 AWS 雲端 環境中的許可。

閒置應用程式

90 天期間 CPU 和記憶體平均使用率在 5% 至 20% 之間的應用程式。在遷移專案中，通常會淘汰這些應用程式或將其保留在內部部署。

IIoT

請參閱工業物聯網。

不可變的基礎設施

為生產工作負載部署新基礎設施的模型，而不是更新、修補或修改現有的基礎設施。不可變基礎設施本質上比可變基礎設施更一致、可靠且可預測。如需詳細資訊，請參閱 AWS Well-Architected Framework 中的使用不可變基礎設施部署最佳實務。

傳入 (輸入) VPC

在 AWS 多帳戶架構中，接受、檢查和路由來自應用程式外部之網路連線的 VPC。AWS 安全參考架構建議您使用傳入、傳出和檢查 VPC 來設定網路帳戶，以保護應用程式與更廣泛的網際網路之間的雙向介面。

增量遷移

一種切換策略，您可以在其中將應用程式分成小部分遷移，而不是執行單一、完整的切換。例如，您最初可能只將一些微服務或使用者移至新系統。確認所有項目都正常運作之後，您可以逐步移動其他微服務或使用者，直到可以解除委任舊式系統。此策略可降低與大型遷移關聯的風險。

工業 4.0

2016 年 Klaus Schwab 推出的術語，透過連線能力、即時資料、自動化、分析和 AI/ML 的進展，指製造程序的現代化。

基礎設施

應用程式環境中包含的所有資源和資產。

基礎設施即程式碼 (IaC)

透過一組組態檔案來佈建和管理應用程式基礎設施的程序。IaC 旨在協助您集中管理基礎設施，標準化資源並快速擴展，以便新環境可重複、可靠且一致。

工業物聯網 (IIoT)

在製造業、能源、汽車、醫療保健、生命科學和農業等產業領域使用網際網路連線的感測器和裝置。如需詳細資訊，請參閱建立工業物聯網 (IIoT) 數位轉型策略。

檢查 VPC

在 AWS 多帳戶架構中，集中式 VPC 可管理 VPCs 之間 （在相同或不同的 中 AWS 區域)、網際網路和內部部署網路之間的網路流量檢查。AWS 安全參考架構建議您使用傳入、傳出和檢查 VPC 來設定網路帳戶，以保護應用程式與更廣泛的網際網路之間的雙向介面。

物聯網 (IoT)

具有內嵌式感測器或處理器的相連實體物體網路，其透過網際網路或本地通訊網路與其他裝置和系統進行通訊。如需詳細資訊，請參閱什麼是 IoT？

可解釋性

機器學習模型的一個特徵，描述了人類能夠理解模型的預測如何依賴於其輸入的程度。如需詳細資訊，請參閱 的機器學習模型可解譯性 AWS。

IoT

請參閱物聯網。

IT 資訊庫 (ITIL)

一組用於交付 IT 服務並使這些服務與業務需求保持一致的最佳實務。ITIL 為 ITSM 提供了基礎。

IT 服務管理 (ITSM)

與組織的設計、實作、管理和支援 IT 服務關聯的活動。如需有關將雲端操作與 ITSM 工具整合的資訊，請參閱操作整合指南。

ITIL

請參閱 IT 資訊庫。

ITSM

請參閱 IT 服務管理。

L

標籤型存取控制 (LBAC)

強制存取控制 (MAC) 的實作，其中使用者和資料本身都會獲得明確指派的安全標籤值。使用者安全標籤和資料安全標籤之間的交集會決定使用者可以看到哪些資料列和資料欄。

登陸區域

登陸區域是架構良好的多帳戶 AWS 環境，可擴展且安全。這是一個起點，您的組織可以從此起點快速啟動和部署工作負載與應用程式，並對其安全和基礎設施環境充滿信心。如需有關登陸區域的詳細資訊，請參閱設定安全且可擴展的多帳戶 AWS 環境。

大型語言模型 (LLM)

預先訓練大量資料的深度學習 AI 模型。LLM 可以執行多個任務，例如回答問題、摘要文件、將文字翻譯成其他語言，以及完成句子。如需詳細資訊，請參閱什麼是 LLMs。

大型遷移

遷移 300 部或更多伺服器。

LBAC

請參閱標籤型存取控制。

最低權限

授予執行任務所需之最低許可的安全最佳實務。如需詳細資訊，請參閱 IAM 文件中的套用最低權限許可。

隨即轉移

請參閱 7 個 R。

小端序系統

首先儲存最低有效位元組的系統。另請參閱 Endianness。

LLM

請參閱大型語言模型。

較低的環境

請參閱 環境。

M

機器學習 (ML)

一種使用演算法和技術進行模式識別和學習的人工智慧。機器學習會進行分析並從記錄的資料 (例如物聯網 (IoT) 資料) 中學習，以根據模式產生統計模型。如需詳細資訊，請參閱機器學習。

主要分支

請參閱分支。

惡意軟體

旨在危及電腦安全或隱私權的軟體。惡意軟體可能會中斷電腦系統、洩露敏感資訊，或取得未經授權的存取。惡意軟體的範例包括病毒、蠕蟲、勒索軟體、特洛伊木馬、間諜軟體和鍵盤記錄器。

受管服務

AWS 服務 會 AWS 操作基礎設施層、作業系統和平台，而您會存取端點來存放和擷取資料。Amazon Simple Storage Service (Amazon S3) 和 Amazon DynamoDB 是受管服務的範例。這些也稱為抽象服務。

製造執行系統 (MES)

一種軟體系統，用於追蹤、監控、記錄和控制生產程序，將原物料轉換為現場成品。

MAP

請參閱遷移加速計劃。

機制

建立工具、推動工具採用，然後檢查結果以進行調整的完整程序。機制是在操作時強化和改善自身的循環。如需詳細資訊，請參閱 AWS Well-Architected Framework 中的建置機制。

成員帳戶

除了屬於組織一部分的管理帳戶 AWS 帳戶 之外的所有 AWS Organizations。一個帳戶一次只能是一個組織的成員。

製造執行系統

請參閱製造執行系統。

訊息佇列遙測傳輸 (MQTT)

根據發佈/訂閱模式的輕量型machine-to-machine(M2M) 通訊協定，適用於資源受限的 IoT 裝置。

微服務

一種小型的獨立服務，它可透過定義明確的 API 進行通訊，通常由小型獨立團隊擁有。例如，保險系統可能包含對應至業務能力 (例如銷售或行銷) 或子領域 (例如購買、索賠或分析) 的微服務。微服務的優點包括靈活性、彈性擴展、輕鬆部署、可重複使用的程式碼和適應力。如需詳細資訊，請參閱使用無 AWS 伺服器服務整合微服務。

微服務架構

一種使用獨立元件來建置應用程式的方法，這些元件會以微服務形式執行每個應用程式程序。這些微服務會使用輕量型 API，透過明確定義的介面進行通訊。此架構中的每個微服務都可以進行

更新、部署和擴展，以滿足應用程式特定功能的需求。如需詳細資訊，請參閱在 上實作微服務 AWS。

Migration Acceleration Program (MAP)

一種 AWS 計畫，提供諮詢支援、訓練和服務，協助組織建立強大的營運基礎，以移至雲端，並協助抵銷遷移的初始成本。MAP 包括用於有條不紊地執行舊式遷移的遷移方法以及一組用於自動化和加速常見遷移案例的工具。

大規模遷移

將大部分應用程式組合依波次移至雲端的程序，在每個波次中，都會以更快的速度移動更多應用程式。此階段使用從早期階段學到的最佳實務和經驗教訓來實作團隊、工具和流程的遷移工廠，以透過自動化和敏捷交付簡化工作負載的遷移。這是 AWS 遷移策略的第三階段。

遷移工廠

可透過自動化、敏捷的方法簡化工作負載遷移的跨職能團隊。遷移工廠團隊通常包括營運、業務分析師和擁有者、遷移工程師、開發人員以及從事 Sprint 工作的 DevOps 專業人員。20% 至 50% 之間的企業應用程式組合包含可透過工廠方法優化的重複模式。如需詳細資訊，請參閱此內容集中的遷移工廠的討論和雲端遷移工廠指南。

遷移中繼資料

有關完成遷移所需的應用程式和伺服器的資訊。每種遷移模式都需要一組不同的遷移中繼資料。遷移中繼資料的範例包括目標子網路、安全群組和 AWS 帳戶。

遷移模式

可重複的遷移任務，詳細描述遷移策略、遷移目的地以及所使用的遷移應用程式或服務。範例：使用 AWS Application Migration Service 重新託管遷移至 Amazon EC2。

遷移組合評定 (MPA)

線上工具，提供驗證商業案例以遷移至 的資訊 AWS 雲端。MPA 提供詳細的組合評定 (伺服器適當規模、定價、總體擁有成本比較、遷移成本分析) 以及遷移規劃 (應用程式資料分析和資料收集、應用程式分組、遷移優先順序，以及波次規劃)。MPA 工具 （需要登入） 可供所有 AWS 顧問和 APN 合作夥伴顧問免費使用。

遷移準備程度評定 (MRA)

使用 AWS CAF 取得組織雲端整備狀態的洞見、識別優缺點，以及建立行動計劃以消除已識別差距的程序。如需詳細資訊，請參閱遷移準備程度指南。MRA 是 AWS 遷移策略的第一階段。

遷移策略

用來將工作負載遷移至 的方法 AWS 雲端。如需詳細資訊，請參閱此詞彙表中的 7 個 Rs 項目，並請參閱動員您的組織以加速大規模遷移。

機器學習 (ML)

請參閱機器學習。

現代化

將過時的 (舊版或單一) 應用程式及其基礎架構轉換為雲端中靈活、富有彈性且高度可用的系統，以降低成本、提高效率並充分利用創新。如需詳細資訊，請參閱《》中的現代化應用程式的策略 AWS 雲端。

現代化準備程度評定

這項評估可協助判斷組織應用程式的現代化準備程度；識別優點、風險和相依性；並確定組織能夠在多大程度上支援這些應用程式的未來狀態。評定的結果就是目標架構的藍圖、詳細說明現代化程序的開發階段和里程碑的路線圖、以及解決已發現的差距之行動計畫。如需詳細資訊，請參閱《》中的評估應用程式的現代化準備 AWS 雲端程度。

單一應用程式 (單一)

透過緊密結合的程序作為單一服務執行的應用程式。單一應用程式有幾個缺點。如果一個應用程式功能遇到需求激增，則必須擴展整個架構。當程式碼庫增長時，新增或改進單一應用程式的功能也會變得更加複雜。若要解決這些問題，可以使用微服務架構。如需詳細資訊，請參閱將單一體系分解為微服務。

MPA

請參閱遷移產品組合評估。

MQTT

請參閱訊息佇列遙測傳輸。

多類別分類

一個有助於產生多類別預測的過程 (預測兩個以上的結果之一)。例如，機器學習模型可能會詢問「此產品是書籍、汽車還是電話？」 或者「這個客戶對哪種產品類別最感興趣？」

可變基礎設施

更新和修改生產工作負載現有基礎設施的模型。為了提高一致性、可靠性和可預測性， AWS Well-Architected Framework 建議使用不可變基礎設施做為最佳實務。

O

OAC

請參閱原始存取控制。

OAI

請參閱原始存取身分。

OCM

請參閱組織變更管理。

離線遷移

一種遷移方法，可在遷移過程中刪除來源工作負載。此方法涉及延長停機時間，通常用於小型非關鍵工作負載。

OI

請參閱 操作整合。

OLA

請參閱操作層級協議。

線上遷移

一種遷移方法，無需離線即可將來源工作負載複製到目標系統。連接至工作負載的應用程式可在遷移期間繼續運作。此方法涉及零至最短停機時間，通常用於關鍵的生產工作負載。

OPC-UA

請參閱開放程序通訊 - 統一架構。

開放程序通訊 - 統一架構 (OPC-UA)

用於工業自動化的machine-to-machine(M2M) 通訊協定。OPC-UA 提供資料加密、身分驗證和授權機制的互通性標準。

操作水準協議 (OLA)

一份協議，闡明 IT 職能群組承諾向彼此提供的內容，以支援服務水準協議 (SLA)。

操作整備審查 (ORR)

問題和相關最佳實務的檢查清單，可協助您了解、評估、預防或減少事件和可能失敗的範圍。如需詳細資訊，請參閱 AWS Well-Architected Framework 中的操作準備度審查 (ORR)。

操作技術 (OT)

使用實體環境控制工業操作、設備和基礎設施的硬體和軟體系統。在製造業中，整合 OT 和資訊技術 (IT) 系統是工業 4.0 轉型的關鍵重點。

操作整合 (OI)

在雲端中將操作現代化的程序，其中包括準備程度規劃、自動化和整合。如需詳細資訊，請參閱操作整合指南。

組織追蹤

由 建立的線索 AWS CloudTrail 會記錄 AWS 帳戶 組織中所有 的所有事件 AWS Organizations。在屬於組織的每個 AWS 帳戶 中建立此追蹤，它會跟蹤每個帳戶中的活動。如需詳細資訊，請參閱 CloudTrail 文件中的建立組織追蹤。

組織變更管理 (OCM)

用於從人員、文化和領導力層面管理重大、顛覆性業務轉型的架構。OCM 透過加速變更採用、解決過渡問題，以及推動文化和組織變更，協助組織為新系統和策略做好準備，並轉移至新系統和策略。在 AWS 遷移策略中，此架構稱為人員加速，因為雲端採用專案所需的變更速度。如需詳細資訊，請參閱 OCM 指南。

原始存取控制 (OAC)

CloudFront 中的增強型選項，用於限制存取以保護 Amazon Simple Storage Service (Amazon S3) 內容。OAC 支援所有 S3 儲存貯體中的所有伺服器端加密 AWS KMS (SSE-KMS) AWS 區域，以及對 S3 儲存貯體的動態PUT和DELETE請求。

原始存取身分 (OAI)

CloudFront 中的一個選項，用於限制存取以保護 Amazon S3 內容。當您使用 OAI 時，CloudFront 會建立一個可供 Amazon S3 進行驗證的主體。經驗證的主體只能透過特定 CloudFront 分發來存取 S3 儲存貯體中的內容。另請參閱 OAC，它可提供更精細且增強的存取控制。

ORR

請參閱操作整備審核。

OT

請參閱操作技術。

傳出 (輸出) VPC

在 AWS 多帳戶架構中，處理從應用程式內啟動之網路連線的 VPC。AWS 安全參考架構建議您使用傳入、傳出和檢查 VPC 來設定網路帳戶，以保護應用程式與更廣泛的網際網路之間的雙向介面。

P

許可界限

附接至 IAM 主體的 IAM 管理政策，可設定使用者或角色擁有的最大許可。如需詳細資訊，請參閱 IAM 文件中的許可界限。

個人身分識別資訊 (PII)

直接檢視或與其他相關資料配對時，可用來合理推斷個人身分的資訊。PII 的範例包括名稱、地址和聯絡資訊。

PII

請參閱個人身分識別資訊。

手冊

一組預先定義的步驟，可擷取與遷移關聯的工作，例如在雲端中提供核心操作功能。手冊可以採用指令碼、自動化執行手冊或操作現代化環境所需的程序或步驟摘要的形式。

PLC

請參閱可程式設計邏輯控制器。

PLM

請參閱產品生命週期管理。

政策

可定義許可的物件（請參閱身分型政策）、指定存取條件（請參閱資源型政策），或定義組織中所有帳戶的最大許可 AWS Organizations（請參閱服務控制政策）。

混合持久性

根據資料存取模式和其他需求，獨立選擇微服務的資料儲存技術。如果您的微服務具有相同的資料儲存技術，則其可能會遇到實作挑戰或效能不佳。如果微服務使用最適合其需求的資料儲存，則

可以更輕鬆地實作並達到更好的效能和可擴展性。如需詳細資訊，請參閱在微服務中啟用資料持久性。

組合評定

探索、分析應用程式組合並排定其優先順序以規劃遷移的程序。如需詳細資訊，請參閱評估遷移準備程度。

述詞

傳回 `true`或 的查詢條件`false`，通常位於 `WHERE`子句中。

述詞下推

一種資料庫查詢最佳化技術，可在傳輸前篩選查詢中的資料。這可減少必須從關聯式資料庫擷取和處理的資料量，並改善查詢效能。

預防性控制

旨在防止事件發生的安全控制。這些控制是第一道防線，可協助防止對網路的未經授權存取或不必要變更。如需詳細資訊，請參閱在 AWS上實作安全控制中的預防性控制。

委託人

中可執行動作和存取資源 AWS 的實體。此實體通常是 AWS 帳戶、IAM 角色或使用者的根使用者。如需詳細資訊，請參閱 IAM 文件中角色術語和概念中的主體。

依設計的隱私權

透過整個開發程序將隱私權納入考量的系統工程方法。

私有託管區域

一種容器，它包含有關您希望 Amazon Route 53 如何回應一個或多個 VPC 內的域及其子域之 DNS 查詢的資訊。如需詳細資訊，請參閱 Route 53 文件中的使用私有託管區域。

主動控制

旨在防止部署不合規資源的安全控制。這些控制項會在佈建資源之前對其進行掃描。如果資源不符合控制項，則不會佈建。如需詳細資訊，請參閱 AWS Control Tower 文件中的控制項參考指南，並參閱實作安全控制項中的主動控制項。 AWS

產品生命週期管理 (PLM)

管理產品整個生命週期的資料和程序，從設計、開發和啟動，到成長和成熟，再到拒絕和移除。

生產環境

請參閱 環境。

可程式設計邏輯控制器 (PLC)

在製造中，高度可靠、可調整的電腦，可監控機器並自動化製造程序。

提示鏈結

使用一個 LLM 提示的輸出做為下一個提示的輸入，以產生更好的回應。此技術用於將複雜任務分解為子任務，或反覆精簡或展開初步回應。它有助於提高模型回應的準確性和相關性，並允許更精細、個人化的結果。

擬匿名化

將資料集中的個人識別符取代為預留位置值的程序。假名化有助於保護個人隱私權。假名化資料仍被視為個人資料。

發佈/訂閱 (pub/sub)

一種模式，可啟用微服務之間的非同步通訊，以提高可擴展性和回應能力。例如，在微服務型 MES 中，微服務可以將事件訊息發佈到其他微服務可訂閱的頻道。系統可以新增新的微服務，而無需變更發佈服務。

Q

查詢計劃

一系列步驟，如指示，用於存取 SQL 關聯式資料庫系統中的資料。

查詢計劃迴歸

在資料庫服務優化工具選擇的計畫比對資料庫環境進行指定的變更之前的計畫不太理想時。這可能因為對統計資料、限制條件、環境設定、查詢參數繫結的變更以及資料庫引擎的更新所導致。

R

RACI 矩陣

請參閱負責、負責、諮詢、告知 (RACI)。

RAG

請參閱擷取增強產生。

勒索軟體

一種惡意軟體，旨在阻止對計算機系統或資料的存取，直到付款為止。

RASCI 矩陣

請參閱負責、負責、諮詢、告知 (RACI)。

RCAC

請參閱資料列和資料欄存取控制。

僅供讀取複本

用於唯讀用途的資料庫複本。您可以將查詢路由至僅供讀取複本以減少主資料庫的負載。

重新架構師

請參閱 7 個 R。

復原點目標 (RPO)

自上次資料復原點以來可接受的時間上限。這會決定最後一個復原點與服務中斷之間可接受的資料遺失。

復原時間目標 (RTO)

服務中斷與服務還原之間的可接受延遲上限。

重構

請參閱 7 個 R。

區域

地理區域中的 AWS 資源集合。每個 AWS 區域 都獨立於其他 ，以提供容錯能力、穩定性和彈性。如需詳細資訊，請參閱指定 AWS 區域 您的帳戶可以使用哪些 。

迴歸

預測數值的 ML 技術。例如，為了解決「這房子會賣什麼價格？」的問題 ML 模型可以使用線性迴歸模型，根據已知的房屋事實 (例如，平方英尺) 來預測房屋的銷售價格。

重新託管

請參閱 7 個 R。

版本

在部署程序中，它是將變更提升至生產環境的動作。

重新定位

請參閱 7 個 R。

Replatform

請參閱 7 個 R。

回購

請參閱 7 個 R。

彈性

應用程式抵禦中斷或從中斷中復原的能力。在 中規劃彈性時，高可用性和災難復原是常見的考量 AWS 雲端。如需詳細資訊，請參閱AWS 雲端 彈性。

資源型政策

附接至資源的政策，例如 Amazon S3 儲存貯體、端點或加密金鑰。這種類型的政策會指定允許存取哪些主體、支援的動作以及必須滿足的任何其他條件。

負責者、當責者、事先諮詢者和事後告知者 (RACI) 矩陣

矩陣，定義所有涉及遷移活動和雲端操作之各方的角色和責任。矩陣名稱衍生自矩陣中定義的責任類型：負責人 (R)、責任 (A)、已諮詢 (C) 和知情 (I)。支援 (S) 類型為選用。如果您包含支援，則矩陣稱為 RASCI 矩陣，如果您排除它，則稱為 RACI 矩陣。

回應性控制

一種安全控制，旨在驅動不良事件或偏離安全基準的補救措施。如需詳細資訊，請參閱在 AWS上實作安全控制中的回應性控制。

保留

請參閱 7 個 R。

淘汰

請參閱 7 Rs。

檢索增強生成 (RAG)

一種生成式 AI 技術，其中 LLM 會在產生回應之前參考訓練資料來源以外的授權資料來源。例如，RAG 模型可能會對組織的知識庫或自訂資料執行語意搜尋。如需詳細資訊，請參閱什麼是 RAG。

輪換

定期更新秘密的程序，讓攻擊者更難存取登入資料。

資料列和資料欄存取控制 (RCAC)

使用已定義存取規則的基本、彈性 SQL 表達式。RCAC 包含資料列許可和資料欄遮罩。

RPO

請參閱復原點目標。

RTO

請參閱復原時間目標。

執行手冊

執行特定任務所需的一組手動或自動程序。這些通常是為了簡化重複性操作或錯誤率較高的程序而建置。

S

SAML 2.0

許多身分提供者 (IdP) 使用的開放標準。此功能會啟用聯合單一登入 (SSO)，讓使用者可以登入 AWS Management Console 或呼叫 AWS API 操作，而不必為您組織中的每個人在 IAM 中建立使用者。如需有關以 SAML 2.0 為基礎的聯合詳細資訊，請參閱 IAM 文件中的關於以 SAML 2.0 為基礎的聯合。

SCADA

請參閱監督控制和資料擷取。

SCP

請參閱服務控制政策。

秘密

您以加密形式存放的 AWS Secrets Manager機密或限制資訊，例如密碼或使用者登入資料。它由秘密值及其中繼資料組成。秘密值可以是二進位、單一字串或多個字串。如需詳細資訊，請參閱 Secrets Manager 文件中的 Secrets Manager 秘密中的什麼內容？。

依設計的安全性

透過整個開發程序將安全性納入考量的系統工程方法。

安全控制

一種技術或管理防護機制，它可預防、偵測或降低威脅行為者利用安全漏洞的能力。安全控制有四種主要類型：預防性、偵測性、回應性和主動性。

安全強化

減少受攻擊面以使其更能抵抗攻擊的過程。這可能包括一些動作，例如移除不再需要的資源、實作授予最低權限的安全最佳實務、或停用組態檔案中不必要的功能。

安全資訊與事件管理 (SIEM) 系統

結合安全資訊管理 (SIM) 和安全事件管理 (SEM) 系統的工具與服務。SIEM 系統會收集、監控和分析來自伺服器、網路、裝置和其他來源的資料，以偵測威脅和安全漏洞，並產生提醒。

安全回應自動化

預先定義和程式設計的動作，旨在自動回應或修復安全事件。這些自動化可做為偵測或回應式安全控制，協助您實作 AWS 安全最佳實務。自動化回應動作的範例包括修改 VPC 安全群組、修補 Amazon EC2 執行個體或輪換登入資料。

伺服器端加密

由 AWS 服務 接收資料的 在其目的地加密資料。

服務控制政策 (SCP)

為 AWS Organizations中的組織的所有帳戶提供集中控制許可的政策。SCP 會定義防護機制或設定管理員可委派給使用者或角色的動作限制。您可以使用 SCP 作為允許清單或拒絕清單，以指定允許或禁止哪些服務或動作。如需詳細資訊，請參閱 AWS Organizations 文件中的服務控制政策。

服務端點

的進入點 URL AWS 服務。您可以使用端點，透過程式設計方式連接至目標服務。如需詳細資訊，請參閱 AWS 一般參考 中的 AWS 服務 端點。

服務水準協議 (SLA)

一份協議，闡明 IT 團隊承諾向客戶提供的服務，例如服務正常執行時間和效能。

服務層級指標 (SLI)

服務效能方面的測量，例如其錯誤率、可用性或輸送量。

服務層級目標 (SLO)

代表服務運作狀態的目標指標，由服務層級指標測量。

共同責任模式

描述您與 共同 AWS 承擔雲端安全與合規責任的模型。 AWS 負責雲端的安全，而 負責雲端的安全。如需詳細資訊，請參閱共同責任模式。

SIEM

請參閱安全資訊和事件管理系統。

單點故障 (SPOF)

應用程式的單一關鍵元件故障，可能會中斷系統。

SLA

請參閱服務層級協議。

SLI

請參閱服務層級指標。

SLO

請參閱服務層級目標。

先拆分後播種模型

擴展和加速現代化專案的模式。定義新功能和產品版本時，核心團隊會進行拆分以建立新的產品團隊。這有助於擴展組織的能力和服務，提高開發人員生產力，並支援快速創新。如需詳細資訊，請參閱 中的階段式應用程式現代化方法 AWS 雲端。

SPOF

請參閱單一故障點。

星狀結構描述

使用一個大型事實資料表來存放交易或測量資料的資料庫組織結構，並使用一或多個較小的維度資料表來存放資料屬性。此結構旨在用於資料倉儲或商業智慧用途。

Strangler Fig 模式

一種現代化單一系統的方法，它會逐步重寫和取代系統功能，直到舊式系統停止使用為止。此模式源自無花果藤，它長成一棵馴化樹並最終戰勝且取代了其宿主。該模式由 Martin Fowler 引入，作

為重寫單一系統時管理風險的方式。如需有關如何套用此模式的範例，請參閱使用容器和 Amazon API Gateway 逐步現代化舊版 Microsoft ASP.NET (ASMX) Web 服務。

子網

您 VPC 中的 IP 地址範圍。子網必須位於單一可用區域。

監控控制和資料擷取 (SCADA)

在製造中，使用硬體和軟體來監控實體資產和生產操作的系統。

對稱加密

使用相同金鑰來加密及解密資料的加密演算法。

合成測試

以模擬使用者互動的方式測試系統，以偵測潛在問題或監控效能。您可以使用 Amazon CloudWatch Synthetics 來建立這些測試。

系統提示

一種向 LLM 提供內容、指示或指導方針以指示其行為的技術。系統提示有助於設定內容，並建立與使用者互動的規則。

T

標籤

做為中繼資料以組織 AWS 資源的鍵值對。標籤可協助您管理、識別、組織、搜尋及篩選資源。如需詳細資訊，請參閱標記您的 AWS 資源。

目標變數

您嘗試在受監督的 ML 中預測的值。這也被稱為結果變數。例如，在製造設定中，目標變數可能是產品瑕疵。

任務清單

用於透過執行手冊追蹤進度的工具。任務清單包含執行手冊的概觀以及要完成的一般任務清單。對於每個一般任務，它包括所需的預估時間量、擁有者和進度。

測試環境

請參閱 環境。

訓練

為 ML 模型提供資料以供學習。訓練資料必須包含正確答案。學習演算法會在訓練資料中尋找將輸入資料屬性映射至目標的模式 (您想要預測的答案)。它會輸出擷取這些模式的 ML 模型。可以使用 ML 模型，來預測您不知道的目標新資料。

傳輸閘道

可以用於互連 VPC 和內部部署網路的網路傳輸中樞。如需詳細資訊，請參閱 AWS Transit Gateway 文件中的什麼是傳輸閘道。

主幹型工作流程

這是一種方法，開發人員可在功能分支中本地建置和測試功能，然後將這些變更合併到主要分支中。然後，主要分支會依序建置到開發環境、生產前環境和生產環境中。

受信任的存取權

將許可授予您指定的服務，以代表您在組織中 AWS Organizations 及其帳戶中執行任務。受信任的服務會在需要該角色時，在每個帳戶中建立服務連結角色，以便為您執行管理工作。如需詳細資訊，請參閱 文件中的 AWS Organizations 搭配使用 AWS Organizations 與其他 AWS 服務。

調校

變更訓練程序的各個層面，以提高 ML 模型的準確性。例如，可以透過產生標籤集、新增標籤、然後在不同的設定下多次重複這些步驟來訓練 ML 模型，以優化模型。

雙比薩團隊

兩個比薩就能吃飽的小型 DevOps 團隊。雙披薩團隊規模可確保軟體開發中的最佳協作。

U

不確定性

這是一個概念，指的是不精確、不完整或未知的資訊，其可能會破壞預測性 ML 模型的可靠性。有兩種類型的不確定性：認知不確定性是由有限的、不完整的資料引起的，而隨機不確定性是由資料中固有的噪聲和隨機性引起的。如需詳細資訊，請參閱量化深度學習系統的不確定性指南。

未區分的任務

也稱為繁重工作，是建立和操作應用程式的必要工作，但不為最終使用者提供直接價值或提供競爭優勢。未區分任務的範例包括採購、維護和容量規劃。

較高的環境

請參閱 環境。

V

清空

一種資料庫維護操作，涉及增量更新後的清理工作，以回收儲存並提升效能。

版本控制

追蹤變更的程序和工具，例如儲存庫中原始程式碼的變更。

VPC 對等互連

兩個 VPC 之間的連線，可讓您使用私有 IP 地址路由流量。如需詳細資訊，請參閱 Amazon VPC 文件中的什麼是 VPC 對等互連。

漏洞

危及系統安全性的軟體或硬體瑕疵。

W

暖快取

包含經常存取的目前相關資料的緩衝快取。資料庫執行個體可以從緩衝快取讀取，這比從主記憶體或磁碟讀取更快。

暖資料

不常存取的資料。查詢這類資料時，通常可接受中等緩慢的查詢。

視窗函數

SQL 函數，對與目前記錄在某種程度上相關的資料列群組執行計算。視窗函數適用於處理任務，例如根據目前資料列的相對位置計算移動平均值或存取資料列的值。

工作負載

提供商業價值的資源和程式碼集合，例如面向客戶的應用程式或後端流程。

工作串流

遷移專案中負責一組特定任務的功能群組。每個工作串流都是獨立的，但支援專案中的其他工作串流。例如，組合工作串流負責排定應用程式、波次規劃和收集遷移中繼資料的優先順序。組合工作串流將這些資產交付至遷移工作串流，然後再遷移伺服器和應用程式。

WORM

請參閱寫入一次，讀取許多。

WQF

請參閱AWS 工作負載資格架構。

寫入一次，讀取許多 (WORM)

儲存模型，可一次性寫入資料，並防止資料遭到刪除或修改。授權使用者可以視需要多次讀取資料，但無法變更資料。此資料儲存基礎設施被視為不可變。

Z

零時差入侵

利用零時差漏洞的攻擊，通常是惡意軟體。

零時差漏洞

生產系統中未緩解的瑕疵或漏洞。威脅行為者可以使用這種類型的漏洞來攻擊系統。開發人員經常因為攻擊而意識到漏洞。

零鏡頭提示

提供 LLM 執行任務的指示，但沒有可協助引導任務的範例 (快照)。LLM 必須使用其預先訓練的知識來處理任務。零鏡頭提示的有效性取決於任務的複雜性和提示的品質。另請參閱少量擷取提示。

殭屍應用程式

CPU 和記憶體平均使用率低於 5% 的應用程式。在遷移專案中，通常會淘汰這些應用程式。

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。