为 ESG 数据构建可扩展的 Web 抓取系统 AWS

AWS 规范性指导

AWS 规范性指导: 为 ESG 数据构建可扩展的 Web 抓取系统 AWS

Table of Contents

为 ESG 数据构建可扩展的 Web 抓取系统 AWS

Vijit Vashishtha 和 Mansi Doshi，Amazon Web Services

2025 年 1 月（文档历史记录）

环境、社会和治理 (ESG) 因素是投资者在评估潜在投资时的关键考虑因素：

- 环境 ——关注公司对自然世界的影响。它包括碳排放、资源管理和能源效率等因素。
- 社交-研究公司如何管理与员工、供应商、客户和社区的关系。它涵盖了劳动实践、多元化和社区参与等方面。
- 治理-着眼于公司的领导层、内部控制和股东权利。它包括董事会组成、高管薪酬和商业道德。

拥有强大ESG实践的公司越来越多地被视为更有能力实现长期可持续性和盈利能力。投资者对ESG信息的需求不断增长。能够通过可靠、有用的ESG数据证明其可持续发展资格的公司更有能力吸引资金并保持竞争力。公司通过各种来源发布ESG数据，例如新闻、文章和年度报告。由于这些信息是分散的，因此网络爬虫可以帮助您高效地收集这些数据。

本综合指南演示了如何使用AWS Fargate亚马逊弹性计算云 (Amazon EC2) 和亚马逊简单存储服务 (Amazon S3) 来构建强大、可扩展且负责任的数据收集管道。AWS Batch它讨论了以下内容：

- 使用以下方法构建可扩展的抓取系统： AWS 服务
  - 用于运行爬虫应用程序的 Fargate 或 Amazon EC2
  - AWS Batch 用于高效协调大规模抓取作业
  - Amazon S3 可实现安全持久的数据存储
- 实施符合道德标准的抓取的最佳实践，包括：
  - 尊重 robots.txt 和网站政策
  - 管理速率限制以避免目标网站不堪重负
  - 确保数据隐私和负责任地使用所收集的信息
- 开发针对基础设施进行优化的Python基于爬虫的 AWS 爬虫
- 在保持道德标准的同时优化爬虫性能

目标受众

本指南适用于希望从公共网站高效收集大量 up-to-date ESG 数据的数据工程师和云架构师。它对于涉及市场分析、可持续财务评估或财务研究的项目尤其重要。

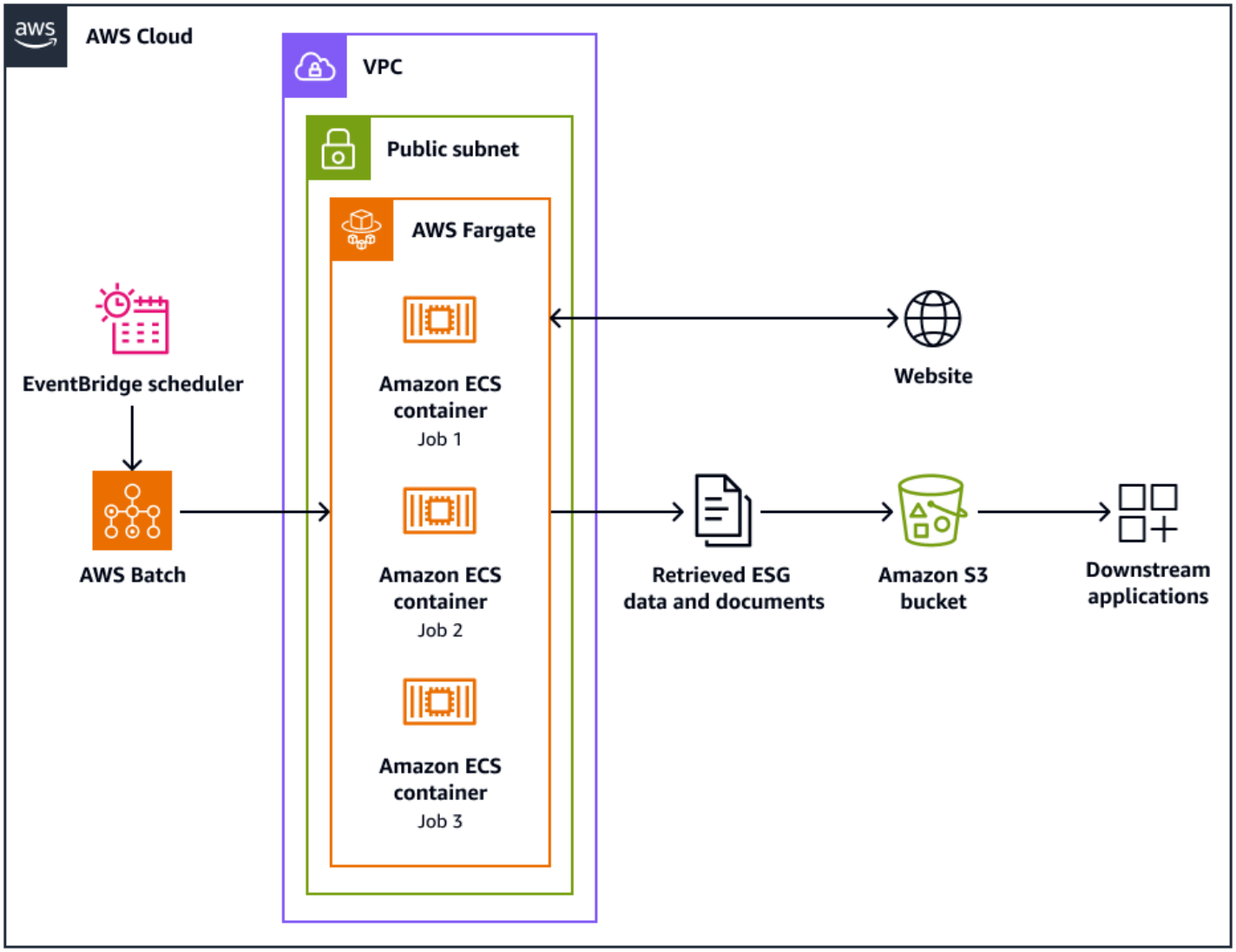
目标业务成果

以下是公司使用ESG数据的常见原因：

- 风险管理 — ESG 数据可帮助您识别和缓解与环境、社会和治理问题相关的潜在风险。
- 吸引投资者 — 许多投资者现在在做出投资决策时会考虑ESG因素。他们将强有力的ESG实践视为长期可持续性和盈利能力的指标。
- 声誉管理 — 良好的ESG绩效可以提高公司在客户、员工和公众中的声誉。
- 监管合规 — 随着ESG相关法规的增加，采用ESG实践有助于公司在合规要求之前保持领先地位。
- 创新和效率 — 关注 ESG 因素可以推动产品、服务和运营的创新。这可以提高效率并节省成本。
- 竞争优势 — 强劲的ESG表现可以使公司与竞争对手区分开来，并开辟新的市场机会。
- 利益相关者参与 — ESG实践可帮助公司更好地与包括员工、客户和当地社区在内的各种利益相关者互动并满足他们的期望。

可扩展的 Web 爬网系统的架构 AWS

以下架构图显示了一个网络爬虫系统，该系统旨在以合乎道德的方式从网站提取环境、社会和治理 (ESG) 数据。您使用针对基础 AWS 架构进行了优化的Python基于爬虫的爬虫。您 AWS Batch 用来编排大规模抓取任务，并使用亚马逊简单存储服务 (Amazon S3) 进行存储。下游应用程序可以从 Amazon S3 存储桶中提取和存储数据。



下图显示了如下工作流：

1. Amazon EventBridge Scheduler 按您安排的时间间隔启动抓取过程。
2. AWS Batch 管理网络爬虫作业的执行。 AWS Batch 作业队列保存并编排待处理的抓取作业。

3. 网络抓取任务在亚马逊弹性容器服务 (Amazon ECS) 容器中运行。 AWS Fargate这些任务在虚拟私有云 (VPC) 的公有子网中运行。
4. 网络爬虫抓取目标网站并检索 ESG 数据和文档，例如 PDF、CSV 或其他文档文件。
5. 网络爬虫将检索到的数据和原始文件存储在 Amazon S3 存储桶中。
6. 其他系统或应用程序接收或处理 Amazon S3 存储桶中存储的数据和文件。

网络爬虫的设计和操作

有些网站是专门为在台式机或移动设备上运行而设计的。Web Crawler 旨在支持使用桌面用户代理或移动用户代理。这些代理可以帮助您成功地向目标网站发出请求。

网络爬虫初始化后，它会执行以下操作：

1. 网络爬虫调用该`setup()`方法。此方法获取并解析 robots.txt 文件。

> **Note**
>
> 您也可以将网络爬虫配置为获取和解析站点地图。

2. 网络爬虫会处理 robots.txt 文件。如果在 robots.txt 文件中指定了抓取延迟，则网络爬虫会提取桌面用户代理的抓取延迟。如果未在 robots.txt 文件中指定抓取延迟，则网络爬虫会使用随机延迟。
3. Web Crawler 调用该`crawl()`方法，该方法启动爬网过程。如果队列中没 URLs 有，则会添加起始 URL。

> **Note**
>
> 爬虫会一直持续到达到最大页面数或用完 URLs 可以抓取为止。

4. 爬虫会处理. URLs 对于队列中的每个 URL，爬虫都会检查该 URL 是否已被抓取。
5. 如果网址尚未被抓取，则抓取器会按如下方式调用该`crawl_url()`方法：
   a. 爬虫会检查 robots.txt 文件以确定它是否可以使用桌面用户代理来抓取 URL。
   b. 如果允许，爬虫会尝试使用桌面用户代理抓取 URL。
   c. 如果不允许或者桌面用户代理无法抓取，则爬虫会检查 robots.txt 文件以确定它是否可以使用移动用户代理来抓取 URL。
   d. 如果允许，抓取工具会尝试使用移动用户代理来抓取 URL。

6. Crawler 调用该`attempt_crawl()`方法，该方法检索和处理内容。抓取工具向 URL 发送带有相应标头的 GET 请求。如果请求失败，Crawler 将使用重试逻辑。
7. 如果文件是 HTML 格式，则爬虫会调用该`extract_esg_data()`方法。它用于Beautiful Soup解析 HTML 内容。它使用关键字匹配来提取环境、社会和治理 (ESG) 数据。

   如果文件是 PDF，则爬虫会调用该`save_pdf()`方法。爬虫会下载 PDF 文件并将其保存到 Amazon S3 存储桶中。
8. 爬虫调用该`extract_news_links()`方法。它可以查找并存储指向新闻文章、新闻稿和博客文章的链接。
9. 爬虫调用该`extract_pdf_links()`方法。它可以识别和存储指向 PDF 文档的链接。
10. 爬虫调用该`is_relevant_to_sustainable_finance()`方法。它使用预定义的关键字来检查新闻或文章是否与可持续金融有关。
11. 每次尝试抓取后，爬虫都会使用该`delay()`方法实现延迟。如果在 robots.txt 文件中指定了延迟，则会使用该值。否则，它将使用 1 到 3 秒之间的随机延迟。
12. 爬网程序调用该`save_esg_data()`方法将 ESG 数据保存到 CSV 文件中。CSV 文件保存在亚马逊 S3 存储桶中。
13. Crawler 调用该`save_news_links()`方法将新闻链接保存到 CSV 文件，包括相关性信息。CSV 文件保存在亚马逊 S3 存储桶中。
14. Crawler 调用该`save_pdf_links()`方法将 PDF 链接保存到 CSV 文件。CSV 文件保存在亚马逊 S3 存储桶中。

批处理和数据处理

抓取过程是以结构化的方式组织和执行的。 AWS Batch 为每家公司分配任务，以便它们分批并行运行。每个批次都侧重于单个公司的域名和子域名，正如您在数据集中识别的那样。但是，同一批次中的作业会按顺序运行，这样它们就不会因为过多的请求而淹没网站。这有助于应用程序更有效地管理抓取工作负载，并确保捕获每家公司的所有相关数据。

通过将网络爬网组织成公司特定的批次，从而对收集的数据进行容器化。这有助于防止一家公司的数据与其他公司的数据混在一起。

批处理可帮助应用程序高效地从网络收集数据，同时根据目标公司及其各自的网络域保持清晰的结构和信息分离。这种方法有助于确保所收集数据的完整性和可用性，因为它组织得井井有条，并且与相应的公司和域名相关联。

在上构建可扩展的网络爬网系统 AWS

本节介绍如何构建本架构节中描述的网络爬虫。它包括一种系统的方法，用于创建强大的公司及其相关网络资产数据集。此数据集是您的抓取活动的基础。然后，本节介绍如何在中Python构建合乎道德的网络爬虫。

主题

- 准备数据集
- 构建 Web 爬虫
- 搭建 AWS 基础架构

准备数据集

如果您尚未这样做，请准备要从中收集信息的网站的详细数据集。此数据集应包括网站网址域名和相关的子域名。本节提供了构建此数据集的 step-by-step过程。

准备数据集

1. 定义范围 — 确定你关注的一个或多个行业。决定要包括多少家公司。并定义您要收集的有关这些公司的任何标准，例如员工人数、地点或收入。
2. 识别数据源-确定您可以使用哪些信息来源来收集有关这些公司的信息。示例包括企业名录（例如 Crunchbase、彭博社或福布斯）、证券交易所（例如纽约证券交易所和纳斯达克）、特定行业的协会或出版物或政府数据库（例如美国证券交易委员会的文件）。
3. 创建表格-在你首选的工具（例如 Microsoft Excel、Google 表格或数据库管理系统）中，创建一个用于收集有关每家公司的标准的表。为每个标准包括一列。至少要包括公司名称、主域名、子域名、行业、规模和位置等列。
4. 收集公司初始信息-收集有关每家公司的以下信息，并将其输入到您创建的表格中：
    - 公司名称
    - 行业或行业
    - 公司规模（员工人数）
    - Revenue (收入)
    - 公司总部所在地
5. 收集域名信息-对于每家公司，从主网站 URL 中提取主域名，例如`example.com`。您可以使用 WHOIS 域名查询工具验证域名信息。

6. 收集子域信息-对于每家公司，研究注册的子域名，例如。`blog.example.com`你可以使用子域枚举工具，例如 sub List3R、O WASP Amass 或 Sub finder。您可以执行 Google dorking（通过搜索`site:example.com`），使用`dig`命令或 DNS 查询工具检查 DNS 记录，也可以分析 SSL 或 TLS 证书。
7. 验证和清理数据-查看、验证和标准化您收集的数据。例如，删除所有重复的条目，从域名和子域中删除不必要的 URL 信息，并确认所有域名和子域名都处于活动状态。
8. （可选）对子域进行分类-您可以将子域名按类型进行分类。以下是您可能遇到的一些类别示例：
   - 博客，例如 `blog.example.com`
   - Support 或帮助，例如`support.example.com`或 `help.example.com`
   - 电子商务，例如`shop.example.com`或 `store.example.com`
   - 开发者资源，例如`dev.example.com`或 `api.example.com`
   - 地区或地点，例如`us.example.com`或 `uk.example.com`
9. （可选）添加相关元数据-您可以在数据集中记录任何相关的元数据。例如，您可以添加上次更新日期、信息来源或置信度分数，以提高子域的准确性。
10. 实现版本控制-使用版本控制系统（例如 Git）来跟踪表格随时间推移而发生的变化。定期备份数据集。
11. 维护表格-设置更新表格的时间表，例如每季度。标准化并实施添加新公司或删除不再需要的公司的流程。如果可能，自动发现子域名。

构建 Web 爬虫

如架构本节所述，该应用程序是分批运行的，每家公司一个。

主题

- 捕获和处理 robots.txt 文件
- 捕获和处理站点地图
- 设计爬行器

捕获和处理 robots.txt 文件

准备好数据集后，您需要确认该域是否有 robots.txt 文件。对于网络爬虫和其他机器人，robots.txt 文件会指明允许他们访问网站的哪些部分。遵守此文件中的说明是以合乎道德的方式抓取网站的重要最佳实践。有关更多信息，请参阅本指南中的符合道德标准的网络爬虫的最佳实践。

捕获和处理 robots.txt 文件

1. 如果您还没有这样做，请在终端中运行以下命令来安装该requests库：

```
pip install requests
```

2. 请运行以下脚本。该脚本执行以下操作：

   - 它定义了一个将域作为输入的check_robots_txt函数。
   - 它构建 robots.txt 文件的完整 URL。
   - 它向 robots.txt 文件的 URL 发送 GET 请求。
   - 如果请求成功（状态代码 200），则存在一个 robots.txt 文件。
   - 如果请求失败或返回不同的状态码，则 robots.txt 文件不存在或无法访问。

```
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**
>
> 此脚本处理网络错误或其他问题的异常。

3. 如果存在 robots.txt 文件，请使用以下脚本将其下载：

```
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**
>
> 这些脚本可以根据您的用例进行自定义或修改。您也可以组合这些脚本。

捕获和处理站点地图

接下来，您需要处理站点地图。您可以使用站点地图将抓取重点放在重要页面上。这提高了抓取效率。有关更多信息，请参阅本指南中的符合道德标准的网络爬虫的最佳实践。

捕获和处理站点地图

- 请运行以下脚本。此脚本定义了一个`check_and_download_sitemap`函数，该函数：
  - 接受基本 URL、来自 robots.txt 的可选站点地图 URL 和用户代理字符串。
  - 检查多个潜在的站点地图位置，包括来自 robots.txt 的站点地图位置（如果提供）。

- 尝试从每个位置下载站点地图。
- 验证下载的内容是否为 XML 格式。
- 调用该parse_sitemap函数以提取 URLs。这个函数：
  - 解析站点地图的 XML 内容。
  - 同时处理常规站点地图和站点地图索引文件。
  - 如果遇到站点地图索引，则以递归方式获取子站点地图。

```
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("...")
```

设计爬行器

接下来，您将设计网络爬虫。该爬虫旨在遵循本指南合乎道德的网络爬虫的最佳实践中描述的最佳实践。本EthicalCrawler课程演示了道德爬行的几个关键原则：

- 获取和解析 robots.txt 文件 — 抓取程序获取目标网站的 robots.txt 文件。
- 尊重抓取权限-在抓取任何 URL 之前，爬网程序会检查 robots.txt 文件中的规则是否允许抓取该 URL。如果某个 URL 被禁用，则抓取工具会跳过该网址并移至下一个网址。
- 尊重抓取延迟 — 爬虫会检查 robots.txt 文件中是否有抓取延迟指令。如果指定了延迟，Crawler 将在两次请求之间使用此延迟。否则，它将使用默认延迟。

- 用户代理识别-爬虫使用自定义的用户代理字符串向网站标识自己。如果需要，网站所有者可以设置特定的规则来限制或允许您的抓取工具。
- 错误处理和优雅降级 — 如果无法获取或解析 robots.txt 文件，则爬网程序会继续使用保守的默认规则。它可以处理网络错误和非 200 个 HTTP 响应。
- 有限抓取-为了避免服务器不堪重负，可以抓取的页面数量有限制。

以下脚本是解释网络爬虫工作原理的伪代码：

```
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.")
```

构建一款收集 ESG 数据的先进、合乎道德的网络爬虫

1. 复制此系统中使用的高级道德网络爬虫的以下代码示例：

```
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. 设置各种属性，包括用户代理、空集合 URLs和数据存储列表。
3. 调整`is_relevant_to_sustainable_finance()`方法中的关键字和相关性标准，以满足您的特定需求。
4. 确保 robots.txt 文件允许抓取网站，并且您使用的是 robots.txt 文件中指定的抓取延迟和用户代理。
5. 根据贵组织的需要，可以考虑对提供的网络爬虫脚本进行以下自定义：
   - 实现该`fetch_sitemap()`方法以提高网址发现的效率。
   - 增强用于生产的错误记录和处理。
   - 实施更复杂的内容相关性分析。
   - 添加深度和广度控件以限制爬行范围。

搭建 AWS 基础架构

你可以用很多 AWS 服务 方法来构建 Web 爬网基础架构。本指南的 “架构” 部分包括一个建议的解决方案。我们建议您考虑使用以下内容 AWS 服务 为网络爬虫构建支持基础架构：

- 使用亚马逊虚拟私有云（亚马逊 VPC）创建 VPC 和子网。
- 使用 Amazon EventBridge 计划程序启动抓取过程。
- 使用作业和作业队列管理 Web 爬虫 AWS Batch作业。
- 使用以下解决方案之一来运行 Web 爬网程序作业：
  - 亚马逊弹性容器服务 (Amazon ECS) Container Service 容器开启 AWS Fargate
  - 亚马逊弹性计算云 (Amazon EC2) 实例

> **Note**
>
> 如果您的应用程序可以应对中断，请考虑通过竞价型队列使用 Amazon EC2 竞价型实例。竞价型实例队列可以帮助您显著节省计算成本。

  - AWS Lambda 函数
- 将检索到的数据和原始文件存储在亚马逊简单存储服务 (Amazon S3) 存储桶中。

合乎道德的网络爬虫的最佳实践

本节讨论了构建收集环境、社会和治理 (ESG) 数据的网络爬取应用程序的最佳实践和关键道德考虑因素。通过遵守这些最佳实践，您可以保护您的项目和组织，并为更负责任和更可持续的网络生态系统做出贡献。这种方法可以帮助您访问有价值的数据，并以尊重所有利益相关者的方式将其用于研究、业务和创新。

Robots.txt 合规性

在网站上，robots.txt 文件用于与网络爬虫和机器人进行沟通，告知应该或不应该访问或爬取网站的哪些部分。当网络爬虫在网站上遇到 robots.txt 文件时，它会解析指令并相应地调整其抓取行为。这样可以防止爬虫违反网站所有者的指示，并保持网站与爬虫之间的合作关系。因此，robots.txt 文件有助于访问控制、敏感内容保护、负载管理和法律合规。

我们建议您遵循以下最佳实践：

- 务必检查并遵守 robots.txt 文件中的规则。
- 在抓取任何 URL 之前，请检查桌面和移动用户代理的规则。
- 如果网站仅允许移动用户代理，请使用不同的代理标头，例如移动代理标头，来处理您的请求。

没有 robots.txt 文件并不一定意味着你不能或不应该抓取网站。抓取应始终以负责任的方式进行，尊重网站的资源和所有者的隐含权利。以下是在没有 robots.txt 时推荐的最佳做法：

- 假设允许爬行，但请谨慎行事。
- 采取礼貌的抓取做法。
- 如果您计划进行大规模抓取，可以考虑向网站所有者寻求许可。

抓取速率限制

使用合理的抓取速率以避免服务器不堪重负。实现请求之间的延迟，可以由 robots.txt 文件指定，也可以使用随机延迟。对于中小型网站，每 10-15 秒 1 个请求可能是合适的。对于较大的网站或具有明确抓取权限的网站，每秒 1-2 个请求可能是合适的。

用户代理透明度

在用户代理标题中标识您的爬虫。此 HTTP 标头信息旨在识别请求内容的设备。通常，代理名称中包含 “机器人” 一词。爬虫和其他机器人有时会在标题中使用重要字段来包含联系信息。

高效爬行

使用网站所有者开发的站点地图，将注意力集中在重要页面上。

自适应方法

对爬虫进行编程，使其在桌面版本不成功时切换到移动用户代理。这可以为爬虫提供访问权限并减轻网站服务器的压力。

错误处理

确保爬网程序正确处理各种 HTTP 状态代码。例如，如果抓取工具遇到 429 状态码（“请求太多”），则应暂停。如果爬虫持续收到 403 状态码（“禁止”），则可以考虑停止抓取。

分批爬行

建议您执行以下操作：

- 与其 URLs 一次爬行所有任务，不如将任务分成较小的批次。这可以帮助分散负载并降低遇到问题（例如超时或资源限制）的风险。
- 如果预计整个抓取任务需要长时间运行，可以考虑将其分成多个更小、更易于管理的任务。这可以使流程更具可扩展性和弹性。
- 如果 URLs 要抓取的数量相对较少，可以考虑使用无服务器解决方案，例如。 AWS Lambda Lambda 函数非常适合短期的事件驱动型任务，因为它们可以自动扩展和处理资源管理。

安全性

对于网络爬网计算任务，我们建议您将环境配置为仅允许出站流量。这有助于通过最大限度地减少攻击面和降低未经授权的入站访问的风险来增强安全性。仅允许出站连接允许抓取过程与目标网站通信并检索必要的数据，并且可以限制任何可能危及系统的入站流量。

其他考虑因素

查看以下其他注意事项和最佳实践：

- 查看网站服务条款或隐私政策中的抓取指南。
- 在 HTML 中查找可能提供抓取指令的`meta`标签。
- 请注意您所在司法管辖区有关数据收集和使用的法律限制。
- 如果网站所有者提出要求，请做好停止抓取的准备。

常见问题解答

如果 robots.txt 文件不可用怎么办？

没有 robots.txt 文件并不一定意味着你不能或不应该抓取网站。抓取应始终以负责任的方式进行，尊重网站的资源和网站所有者的隐含权利。

如果 sitemaps.xml 文件不可用怎么办？

根据要求，您可以执行以下操作之一：

- 搜索 HTML 站点地图 — 查找列出网站上重要页面的 HTML 站点地图页面。它们通常链接在页脚中。
- 从主页抓取 — 从主页开始爬行，然后点击内部链接来发现其他页面。
- 分析网址模式 — 分析网站的网址结构以识别模式并以编程方式产生潜力 URLs。
- 查看 robots.txt 文件 — 检查 robots.txt 文件中是否存在任何不允许的页面或目录。这些可以提供有关网站结构的线索。
- 查看 API 端点 — 一些网站提供可用于检索内容和结构信息的 API 端点。
- 查看搜索引擎结果 — 使用搜索引擎通过以下网站查找网站的索引页面：搜索运算符，例如`site:example.com`。
- 分析反向链接 — 分析指向该网站的反向链接，以发现其他网站链接到的重要页面。
- 查看网络档案 — 查看互联网档案（例如 Wayback Mac hine），查看网站中可能有站点地图或不同结构的旧版本。
- 寻找内容管理系统 (CMS) 模式 — 如果您能识别 CMS，请使用与该系统关联的常用 URL 模式。
- 确认 JavaScript 渲染 — 如果网站严重依赖该网站 JavaScript，请确保您的抓取工具能够进行渲染 JavaScript 以发现动态加载的内容。对于某些网站，sitemap.xml 文件会在启用 JavaScript 渲染后加载。

我能否使用无服务器解决方案来代替 Amazon EC2 或 Amazon ECS？

是的。 AWS Lambda网页抓取功能可能是一个可行的选择，特别是对于规模较小或模块化程度更高的抓取任务。但是，对于大规模、长时间运行的抓取操作，使用亚马逊弹性计算云 (Amazon EC2) 实例

或亚马逊弹性容器服务 (Amazon ECS) Service 的更传统的方法可能更合适。在为您的网络抓取需求选择合适的计算服务时，请务必仔细评估您的具体要求和权衡取舍。

为什么爬虫会收到 403 状态码？

HTTP 403 是一个 HTTP 状态码，表示禁止访问所请求的资源。如果请求正确，则服务器理解了请求并且不会满足请求。要防止 403 状态码，您可以执行以下操作：

- 限制您的抓取速度。
- 检查站点地图或 robots.txt 文件是否允许抓取工具访问网址。
- 尝试使用移动用户代理，而不是桌面用户代理。

如果以上方法都不起作用，则应尊重网站所有者的决定，不要抓取该页面。

后续步骤和资源

收集原始环境、社会和治理 (ESG) 数据后，您可以执行以下操作从数据中提取有意义的信息：

1. 清理数据 — 数据可能包含大量与 ESG 因素和财务数据无关的不相关信息。重要的是要删除这些不相关的数据，只保留执行所需分析所需的信息。您可以使用诸如 yfinanc e 之类的工具来帮助清理数据。
2. 提取和转换数据-从原始数据中提取相关特征或变量，然后将其转换为适合分析的格式。您可以将数据转换为表格格式，以提高可读性和清晰度。你可以使用一个库，比如 pandas，以完善数据。您还可以使用特征工程、数据标准化和派生指标来转换数据。
3. 执行分析-您可以执行各种分析任务。这可能包括生成描述性统计数据、创建数据可视化以及进行探索性数据分析，以深入了解公司的ESG表现。
4. 应用机器学习 — 您可以使用清理和转换后的数据来训练机器学习模型。这些模型可以帮助您识别目前表现出财务可持续性的公司，并预测其未来的可持续发展表现。

通过使用网络爬虫和此数据评估流程，您可以有效地全面了解所评估公司的可持续发展实践和财务业绩。您可以使用这些信息为投资决策提供信息，跟踪进展并支持可持续的商业实践。

资源

- 什么是网络爬虫？ (Cloudflare 网站)
- 环境、社会及管治投资指南 (Investopedia 网站)

工具

- Beautiful Soup (Beautiful Soup 文档)
- 处理表格关系数据 (pandas 网站)

文档历史记录

下表介绍了本指南的一些重要更改。如果您希望收到有关未来更新的通知，可以订阅 RSS 源。

变更	说明	日期
初次发布	—	2025年1月6日

AWS 规范性指导词汇表

以下是 AWS 规范性指导提供的策略、指南和模式中的常用术语。若要推荐词条，请使用术语表末尾的提供反馈链接。

数字

7 R

将应用程序迁移到云中的 7 种常见迁移策略。这些策略以 Gartner 于 2011 年确定的 5 R 为基础，包括以下内容：

- Refactor/re-architect — 充分利用云原生功能来提高敏捷性、性能和可扩展性，从而移动应用程序并修改其架构。这通常涉及到移植操作系统和数据库。示例：将您的本地 Oracle 数据库迁移到亚马逊 Aurora PostgreSQL-Compatible 版。
- 更换平台：将应用程序迁移到云中，并进行一定程度的优化，以利用云功能。示例：将本地 Oracle 数据库迁移到 AWS Cloud中的 Amazon Relational Database Service（Amazon RDS）for Oracle。
- 重新购买：转换到其他产品，通常是从传统许可转向 SaaS 模式。示例：将您的客户关系管理 (CRM) 系统迁移到 Salesforce.com。
- 重新托管（直接迁移）：将应用程序迁移到云，无需进行任何更改即可利用云功能。示例：将本地 Oracle 数据库迁移到 AWS Cloud中 EC2 实例上的 Oracle。
- 重新放置（虚拟机监控器级直接迁移）：将基础设施迁移到云中，无需购买新硬件、重写应用程序或修改现有操作。您将服务器从本地平台迁移到同一平台的云服务中。示例：将Microsoft Hyper-V应用程序迁移到 AWS。
- 保留（重访）：将应用程序保留在源环境中。其中可能包括需要进行重大重构的应用程序，并且您希望将工作推迟到以后，以及您希望保留的遗留应用程序，因为迁移它们没有商业上的理由。
- 停用：停用或删除源环境中不再需要的应用程序。

A

A2A () Agent-to-Agent

一种支持任务委托和状态转移的代理到代理协作的状态协议。

ABAC

请参阅基于属性的访问控制。

抽象服务

请参阅托管服务。

ACID

请参阅原子性、一致性、隔离性、持久性。

主动-主动迁移

一种数据库迁移方法，在这种方法中，源数据库和目标数据库保持同步（通过使用双向复制工具或双写操作），两个数据库都在迁移期间处理来自连接应用程序的事务。这种方法支持小批量、可控的迁移，而不需要一次性割接。它比主动-被动迁移更灵活，但工作量更大。

主动-被动迁移

一种数据库迁移方法，在这种方法中，源数据库和目标数据库保持同步，但在将数据复制到目标数据库时，只有源数据库处理来自连接应用程序的事务。目标数据库在迁移期间不接受任何事务。

座席

一种能够使用工具自主推理、计划和采取行动来实现目标的人工智能系统。

特工行动

在生产环境中大规模构建、测试、部署和运行 AI 代理的操作实践。

聚合函数

一种 SQL 函数，它对一组行进行操作并计算该组的单个返回值。聚合函数的示例包括 SUM 和 MAX。

AI

请参阅人工智能。

AIOps

请参阅人工智能运营。

匿名化

永久删除数据集中个人信息的过程。匿名化可以帮助保护个人隐私。匿名化数据不再被视为个人数据。

反模式

一种用于解决反复出现的问题的常用解决方案，而在这类问题中，此解决方案适得其反、无效或不如替代方案有效。

应用程序控制

一种安全方法，仅允许使用经批准的应用程序，以帮助保护系统免受恶意软件的侵害。

应用程序组合

有关组织使用的每个应用程序的详细信息的集合，包括构建和维护该应用程序的成本及其业务价值。这些信息是产品组合发现和分析过程的关键，有助于识别需要进行迁移、现代化和优化的应用程序并确定其优先级。

人工智能（AI）

计算机科学领域致力于使用计算技术执行通常与人类相关的认知功能，例如学习、解决问题和识别模式。有关更多信息，请参阅什么是人工智能？

人工智能运营（AIOps）

使用机器学习技术解决运营问题、减少运营事故和人为干预以及提高服务质量的过程。有关如何在 AWS 迁移策略中使用 AIOps 的更多信息，请参阅运营集成指南。

非对称加密

一种加密算法，使用一对密钥，一个公钥用于加密，一个私钥用于解密。您可以共享公钥，因为它不用于解密，但对私钥的访问应受到严格限制。

原子性、一致性、隔离性、持久性（ACID）

一组软件属性，即使在出现错误、电源故障或其他问题的情况下，也能保证数据库的数据有效性和操作可靠性。

基于属性的访问权限控制（ABAC）

根据用户属性（如部门、工作角色和团队名称）创建精细访问权限的做法。有关更多信息，请参阅 AWS Identity and Access Management (I AM) 文档 AWS中的 AB AC。

权威数据来源

存储主要数据版本的位置，被认为是最可靠的信息源。您可以将数据从权威数据来源复制到其他位置，以便处理或修改数据，例如对数据进行匿名化、编辑或假名化。

可用区

中的一个不同位置 AWS 区域 ，不受其他可用区域故障的影响，并向同一区域中的其他可用区提供低成本、低延迟的网络连接。

AWS 云采用框架 (AWS CAF)

该框架包含指导方针和最佳实践 AWS ，可帮助组织制定高效且有效的计划，以成功迁移到云端。AWS CAF将指导分为六个重点领域，称为视角：业务、人员、治理、平台、安全和运营。业务、人员和治理角度侧重于业务技能和流程；平台、安全和运营角度侧重于技术技能和流程。例如，人员角度针对的是负责人力资源（HR）、人员配置职能和人员管理的利益相关者。从这个角度来看，AWS CAF 为人员发展、培训和沟通提供了指导，以帮助组织为成功采用云做好准备。有关更多信息，请参阅 AWS CAF 网站和 AWS CAF 白皮书。

AWS 工作负载资格框架 (AWS WQF)

一种评估数据库迁移工作负载、推荐迁移策略和提供工作估算的工具。 AWS WQF 包含在 AWS Schema Conversion Tool (AWS SCT) 中。它用来分析数据库架构和代码对象、应用程序代码、依赖关系和性能特征，并提供评测报告。

B

恶意机器人

一种旨在扰乱或伤害个人或组织的机器人。

BCP

请参阅业务连续性计划。

行为图

一段时间内资源行为和交互的统一交互式视图。您可以使用 Amazon Detective 的行为图来检查失败的登录尝试、可疑的 API 调用和类似的操作。有关更多信息，请参阅 Detective 文档中的行为图中的数据。

大端序系统

一个先存储最高有效字节的系统。另请参阅字节顺序。

二进制分类

一种预测二进制结果（两个可能的类别之一）的过程。例如，您的 ML 模型可能需要预测诸如“该电子邮件是否为垃圾邮件？”或“这个产品是书还是汽车？”之类的问题

bloom 筛选条件

一种概率性、内存高效的数据结构，用于测试元素是否为集合的成员。

blue/green 部署

一种部署策略，您可以创建两个独立但完全相同的环境。在一个环境中运行当前应用程序版本（蓝色），在另一个环境中运行新应用程序版本（绿色）。此策略可帮助您在影响最小的情况下快速回滚。

自动程序

一种通过互联网运行自动任务并模拟人类活动或交互的软件应用程序。有些机器人是有用或有益的，例如在互联网上索引信息的 Web 爬网程序。还有一些被称为恶意机器人的机器人，其目的是扰乱或伤害个人或组织。

僵尸网络

被恶意软件感染并受单方（称为僵尸网络控制者或僵尸网络操作者）控制的僵尸网络。僵尸网络是最著名的扩展机器人及其影响力的机制。

分支

代码存储库的一个包含区域。在存储库中创建的第一个分支是主分支。您可以从现有分支创建新分支，然后在新分支中开发功能或修复错误。为构建功能而创建的分支通常称为功能分支。当功能可以发布时，将功能分支合并回主分支。有关更多信息，请参阅关于分支（GitHub 文档）。

紧急（break-glass）访问

在特殊情况下，通过批准的流程，用户 AWS 账户 可以快速访问他们通常没有访问权限的内容。有关更多信息，请参阅指南中的 “实施破碎玻璃程序” AWS Well-Architected 指示器。

棕地策略

您环境中的现有基础设施。在为系统架构采用棕地策略时，您需要围绕当前系统和基础设施的限制来设计架构。如果您正在扩展现有基础设施，则可以将棕地策略和全新策略混合。

缓冲区缓存

存储最常访问的数据的内存区域。

业务能力

企业如何创造价值（例如，销售、客户服务或营销）。微服务架构和开发决策可以由业务能力驱动。有关更多信息，请参阅在 AWS上运行容器化微服务白皮书中的围绕业务能力进行组织部分。

业务连续性计划（BCP）

一项计划，旨在应对大规模迁移等破坏性事件对运营的潜在影响，并使企业能够快速恢复运营。

C

CAF

请参阅 AWS 云采用框架。

金丝雀部署

缓慢而渐进地向最终用户发布版本。当您确信无误后，即可部署新版本，并完全替换当前版本。

CCoE

请参阅云卓越中心。

CDC

请参阅更改数据捕获。

更改数据捕获（CDC）

跟踪数据来源（如数据库表）的更改并记录有关更改的元数据的过程。您可以将 CDC 用于各种目的，例如审计或复制目标系统中的更改以保持同步。

混沌工程

故意引入故障或破坏性事件来测试系统的韧性。您可以使用 AWS Fault Injection Service (AWS FIS) 来执行实验，对您的 AWS 工作负载施加压力并评估其响应。

CI/CD

请参阅持续集成和持续交付。

分类

一种有助于生成预测的分类流程。分类问题的 ML 模型预测离散值。离散值始终彼此不同。例如，一个模型可能需要评估图像中是否有汽车。

公民开发者

使用无code/low代码平台创建 AI 应用程序但没有专业技术技能的企业用户。

客户端加密

在目标 AWS 服务 收到数据之前，对数据进行本地加密。

云卓越中心（CCoE）

一个多学科团队，负责推动整个组织的云采用工作，包括开发云最佳实践、调动资源、制定迁移时间表、领导组织完成大规模转型。有关更多信息，请参阅 AWS Cloud 企业战略博客上的 CCoE 帖子。

云计算

通常用于远程数据存储和 IoT 设备管理的云技术。云计算通常连接到边缘计算技术。

云运营模型

在 IT 组织中，一种用于构建、完善和优化一个或多个云环境的运营模型。有关更多信息，请参阅构建您的云运营模型。

云采用阶段

组织迁移到 AWS Cloud中时通常会经历四个阶段：

- 项目 - 出于概念验证和学习目的，开展一些与云相关的项目
- 基础 - 进行基础投资以扩大云采用率（例如，创建登录区、定义 CCoE、建立运营模型）
- 迁移 - 迁移单个应用程序
- Re-invention — 优化产品和服务，在云端进行创新

Stephen Orban在 AWS Cloud 企业战略博客的博客文章《走向之旅 Cloud-First 和采用阶段》中定义了这些阶段。有关它们与 AWS 迁移策略的关系的信息，请参阅迁移准备指南。

CMDB

请参阅配置管理数据库。

代码存储库

通过版本控制过程存储和更新源代码和其他资产（如文档、示例和脚本）的位置。常见的云存储库包括 GitHub 或 Bitbucket Cloud。每个版本的代码都称为一个分支。在微服务结构中，每个存储库都专门用于一个功能。单个 CI/CD管道可以使用多个存储库。

冷缓存

一种空的、填充不足或包含过时或不相关数据的缓冲区缓存。这会影响性能，因为数据库实例必须从主内存或磁盘读取，这比从缓冲区缓存读取要慢。

冷数据

很少访问的数据，且通常是历史数据。查询此类数据时，通常可以接受慢速查询。将这些数据转移到性能较低且成本更低的存储层或类别可以降低成本。

计算机视觉（CV）

一种 AI 领域，它使用机器学习来分析和提取数字图像和视频等视觉格式中的信息。例如，Amazon SageMaker AI 为 CV 提供了图像处理算法。

配置偏移

对于工作负载而言，一种偏离预期状态的配置更改。这可能会导致工作负载变得不合规，且通常是渐进的，不是故意的。

配置管理数据库（CMDB）

一种存储库，用于存储和管理有关数据库及其 IT 环境的信息，包括硬件和软件组件及其配置。您通常在迁移的产品组合发现和分析阶段使用来自 CMDB 的数据。

合规性包

一系列 AWS Config 规则和补救措施，您可以汇编这些规则和补救措施，以自定义合规性和安全性检查。您可以使用 YAML 模板将一致性包作为单个实体部署在 AWS 账户 和区域或整个组织中。有关更多信息，请参阅 AWS Config 文档中的一致性包。

持续集成和持续交付 (CI/CD)

自动执行软件发布过程的源代码、构建、测试、暂存和生产阶段的过程。 CI/CD 通常被描述为管道。 CI/CD 可以帮助您实现流程自动化、提高生产力、提高代码质量和更快地交付。有关更多信息，请参阅持续交付的优势。CD 也可以表示持续部署。有关更多信息，请参阅持续交付与持续部署。

CV

请参阅计算机视觉。

D

静态数据

网络中静止的数据，例如存储中的数据。

数据分类

根据网络中数据的关键性和敏感性对其进行识别和分类的过程。它是任何网络安全风险管理策略的关键组成部分，因为它可以帮助您确定对数据的适当保护和保留控制。数据分类是《 AWS Well-Architected 框架》中安全支柱的组成部分。有关详细信息，请参阅数据分类。

数据漂移

生产数据与用来训练机器学习模型的数据之间的有意义差异，或者输入数据随时间推移的有意义变化。数据漂移可能降低机器学习模型预测的整体质量、准确性和公平性。

传输中数据

在网络中主动移动的数据，例如在网络资源之间移动的数据。

数据网格

一种架构框架，可提供分布式、去中心化的数据所有权以及集中式管理和治理。

数据最少化

仅收集并处理绝对必要数据的原则。在中进行数据最小化 AWS Cloud 可以降低隐私风险、成本和分析碳足迹。

数据边界

AWS 环境中的一组预防性防护措施，可帮助确保只有可信身份才能访问来自预期网络的可信资源。有关更多信息，请参阅在上构建数据边界。 AWS

数据预处理

将原始数据转换为 ML 模型易于解析的格式。预处理数据可能意味着删除某些列或行，并处理缺失、不一致或重复的值。

数据溯源

在数据的整个生命周期跟踪其来源和历史的过程，例如数据如何生成、传输和存储。

数据主体

正在收集和处理其数据的个人。

数据仓库

一种支持商业智能（例如分析）的数据管理系统。数据仓库通常包含大量历史数据，通常用于查询和分析。

数据库定义语言（DDL）

在数据库中创建或修改表和对象结构的语句或命令。

数据库操作语言（DML）

在数据库中修改（插入、更新和删除）信息的语句或命令。

DDL

请参阅数据库定义语言。

深度融合

组合多个深度学习模型进行预测。您可以使用深度融合来获得更准确的预测或估算预测中的不确定性。

深度学习

一个 ML 子字段使用多层人工神经网络来识别输入数据和感兴趣的目标变量之间的映射。

深度防御

一种信息安全方法，经过深思熟虑，在整个计算机网络中分层实施一系列安全机制和控制措施，以保护网络及其中数据的机密性、完整性和可用性。当你采用这种策略时 AWS，你会在 AWS Organizations 结构的不同层面添加多个控件来帮助保护资源。例如，深度防御方法可能将多因素身份验证、网络分段和加密结合起来。

委派管理员

在中 AWS Organizations，兼容的服务可以注册 AWS 成员帐户来管理组织的帐户并管理该服务的权限。此账户被称为该服务的委托管理员。有关更多信息和兼容服务列表，请参阅 AWS Organizations 文档中使用 AWS Organizations的服务。

部署

使应用程序、新功能或代码修复在目标环境中可用的过程。部署涉及在代码库中实现更改，然后在应用程序的环境中构建和运行该代码库。

开发环境

请参阅环境。

侦测性控制

一种安全控制，在事件发生后进行检测、记录日志和发出提醒。这些控制是第二道防线，提醒您注意绕过现有预防性控制的安全事件。有关更多信息，请参阅在 AWS上实施安全控制中的侦测性控制。

开发价值流映射（DVSM）

用于识别对软件开发生命周期中的速度和质量产生不利影响的限制因素并确定其优先级的流程。DVSM 扩展了最初为精益生产实践设计的价值流映射流程。其重点关注在软件开发过程中创造和转移价值所需的步骤和团队。

数字孪生

真实世界系统的虚拟再现，如建筑物、工厂、工业设备或生产线。数字孪生支持预测性维护、远程监控和生产优化。

维度表

星型架构中的一种较小的表，其中包含事实表中定量数据的数据属性。维度表属性通常是文本字段或行为类似于文本的离散数字。这些属性通常用于查询约束、筛选和结果集标注。

灾难

阻止工作负载或系统在其主要部署位置实现其业务目标的事件。这些事件可能是自然灾害、技术故障或人为操作的结果，例如无意的配置错误或恶意软件攻击。

灾难恢复（DR）

您用来最大程度地减少由灾难造成的停机时间和数据丢失的策略和流程。有关更多信息，请参阅《工作负载灾难恢复 AWS： AWS Well-Architected 框架中的云端恢复》。

DML

请参阅数据库操作语言。

领域驱动设计

一种开发复杂软件系统的方法，通过将其组件连接到每个组件所服务的不断发展的领域或核心业务目标。埃里克·埃文斯（Eric Evans）在他的《Domain-Driven 设计：解决软件核心的复杂性》（波士顿： Addison-Wesley 专业版，2003年）一书中介绍了这个概念。有关如何使用带有 strangler fig 模式的域驱动设计的信息，请参阅使用容器和 Amazon API Gateway 逐步实现传统微软 ASP.NET (ASMX) 网络服务的现代化。

DR

请参阅灾难恢复。

偏差检测

跟踪与基准配置的偏差。例如，您可以使用 AWS CloudFormation 来检测系统资源中的偏差，也可以使用 AWS Control Tower 来检测着陆区中可能影响监管要求合规性的变化。

DVSM

请参阅开发价值流映射。

E

EDA

请参阅探索性数据分析。

EDI

请参阅电子数据交换。

边缘计算

该技术可提高位于 IoT 网络边缘的智能设备的计算能力。与云计算比较时，边缘计算可以减少通信延迟并缩短响应时间。

电子数据交换（EDI）

组织之间业务文件的自动交换。有关更多信息，请参阅什么是电子数据交换。

加密

一种将人类可读的纯文本数据转换为加密文字的计算流程。

加密密钥

由加密算法生成的随机位的加密字符串。密钥的长度可能有所不同，而且每个密钥都设计为不可预测且唯一。

字节顺序

字节在计算机内存中的存储顺序。 Big-endian 系统首先存储最重要的字节。 Little-endian 系统首先存储最低有效字节。

端点

请参阅服务端点。

端点服务

一种可以在虚拟私有云（VPC）中托管，与其他用户共享的服务。您可以使用其他 AWS 账户 或 AWS Identity and Access Management (IAM) 委托人创建终端节点服务， AWS PrivateLink 并向其授予权限。这些账户或主体可通过创建接口 VPC 端点来私密地连接到您的端点服务。有关更多信息，请参阅 Amazon Virtual Private Cloud（Amazon VPC）文档中的创建端点服务。

企业资源规划（ERP）

一种自动化和管理企业关键业务流程（例如会计、MES 和项目管理）的系统。

信封加密

用另一个加密密钥对加密密钥进行加密的过程。有关更多信息，请参阅 AWS Key Management Service (AWS KMS) 文档中的信封加密。

环境

正在运行的应用程序的实例。以下是云计算中常见的环境类型：

- 开发环境 — 正在运行的应用程序的实例，只有负责维护应用程序的核心团队才能使用。开发环境用于测试更改，然后再将其提升到上层环境。这类环境有时称为测试环境。
- 下层环境 — 应用程序的所有开发环境，比如用于初始构建和测试的环境。
- 生产环境 — 最终用户可以访问的正在运行的应用程序的实例。在 CI/CD 管道中，生产环境是最后一个部署环境。
- 上层环境 — 除核心开发团队以外的用户可以访问的所有环境。这可能包括生产环境、预生产环境和用户验收测试环境。

epic

在敏捷方法学中，有助于组织工作和确定优先级的功能类别。epics 提供了对需求和实施任务的总体描述。例如， AWS CAF 安全史诗包括身份和访问管理、侦探控制、基础设施安全、数据保护和事件响应。有关 AWS 迁移策略中 epics 的更多信息，请参阅计划实施指南。

ERP

请参阅企业资源规划。

探索性数据分析（EDA）

分析数据集以了解其主要特征的过程。您收集或汇总数据，并进行初步调查，以发现模式、检测异常并检查假定情况。EDA 通过计算汇总统计数据和创建数据可视化得以执行。

F

事实表

星型架构中的中心表。它存储有关业务运营的定量数据。通常，事实表包含两种类型的列：包含度量的列和包含维度表外键的列。

快速失效机制

一种使用频繁且增量式的测试来缩短开发生命周期的理念。这是敏捷方法的关键部分。

故障隔离边界

在中 AWS Cloud，诸如可用区 AWS 区域、控制平面或数据平面之类的边界，它限制了故障的影响并有助于提高工作负载的弹性。有关更多信息，请参阅 AWS 故障隔离边界。

功能分支

请参阅分支。

特征

您用来进行预测的输入数据。例如，在制造环境中，特征可能是定期从生产线捕获的图像。

特征重要性

特征对于模型预测的重要性。这通常表示为数值分数，可以通过各种技术进行计算，例如 Shapley 加法解释（SHAP）和积分梯度。有关更多信息，请参阅机器学习模型的可解释性 AWS。

功能转换

为 ML 流程优化数据，包括使用其他来源丰富数据、扩展值或从单个数据字段中提取多组信息。这使得 ML 模型能从数据中获益。例如，如果您将“2021-05-27 00:15:37”日期分解为“2021”、“五月”、“星期四”和“15”，则可以帮助学习与不同数据成分相关的算法学习精细模式。

少样本提示

在要求 LLM 执行类似任务之前，先向其提供少量示例，以演示任务和预期输出。这种技术是情境学习的应用，模型可以从提示中嵌入的示例（镜头）中学习。 Few-shot 对于需要特定格式、推理或领域知识的任务，提示可能非常有效。另请参阅零样本提示。

FGAC

请参阅精细访问控制。

精细访问控制（FGAC）

使用多个条件允许或拒绝访问请求。

快闪迁移

一种数据库迁移方法，通过更改数据捕获使用连续数据复制，在极短的时间内迁移数据，而非使用分阶段方法。目标是将停机时间降至最低。

FM

请参阅基础模型。

基础模型（FM）

一个大型深度学习神经网络，它已使用海量的通用和未标注数据集进行训练。FM 能够执行各种常规任务，例如理解语言、生成文本和图像以及使用自然语言进行对话。有关更多信息，请参阅什么是基础模型。

FM 网关

一种集中式中介，用于控制和规范对基础模型的访问。也称为 LLM 网关。

G

生成式人工智能

AI 模型的一个子集，这些模型已经过大量数据训练，可以使用简单的文本提示来创建新的内容和构件，例如图像、视频、文本和音频。有关更多信息，请参阅什么是生成式人工智能。

地理阻止

请参阅地理限制。

地理限制（地理阻止）

在 Amazon 中 CloudFront，一种阻止特定国家/地区的用户访问内容分发的选项。您可以使用允许列表或阻止列表来指定已批准和已禁止的国家/地区。有关更多信息，请参阅 CloudFront 文档中的限制内容的地理分布。

GitFlow 工作流程

一种方法，在这种方法中，下层和上层环境在源代码存储库中使用不同的分支。Gitflow 工作流程被认为是传统的工作流程，而基于中继的工作流程则是现代的、首选的方法。

黄金映像

系统或软件的快照，用作部署该系统或软件的新实例的模板。例如，在制造业中，黄金映像可用于在多个设备上预调配软件，并有助于提高设备制造操作的速度、可扩展性和生产效率。

全新策略

在新环境中缺少现有基础设施。在对系统架构采用全新策略时，您可以选择所有新技术，而不受对现有基础设施（也称为棕地）兼容性的限制。如果您正在扩展现有基础设施，则可以将棕地策略和全新策略混合。

防护机制

一种高级规则，用于跨组织单位（OU）管理资源、策略和合规性。预防性防护机制会执行策略以确保符合合规性标准。它们是使用服务控制策略和 IAM 权限边界实现的。侦测性护栏会检测策略违规和合规性问题，并生成提醒以进行修复。它们通过使用 AWS Config、、Amazon、 AWS Security Hub CSPM GuardDuty AWS Trusted Advisor、Amazon Inspector 和自定义 AWS Lambda 支票来实现。

护栏 (AI)

用于过滤、验证和限制代理输入和输出的安全机制，有助于确保负责任和安全的 AI 行为。

H

HA

请参阅高可用性。

异构数据库迁移

将源数据库迁移到使用不同数据库引擎的目标数据库（例如，从 Oracle 迁移到 Amazon Aurora）。异构迁移通常是重新架构工作的一部分，而转换架构可能是一项复杂的任务。AWS 提供了 AWS SCT 来帮助实现架构转换。

高可用性（HA）

在遇到挑战或灾难时，工作负载无需干预即可连续运行的能力。HA 系统旨在自动进行故障转移、持续提供良好性能，并以最小的性能影响处理不同负载和故障。

历史数据库现代化

一种用于实现运营技术（OT）系统现代化和升级以更好满足制造业需求的方法。历史数据库是一种用于收集和存储工厂中各种来源数据的数据库。

保留数据

从用于训练机器学习模型的数据集中保留的一部分标注的历史数据。通过将模型预测与保留数据进行比较，您可以使用保留数据来评估模型性能。

人机在圈 (HitL)

一种工作流程模式，其中代理执行在关键决策点暂停以供人工审查和批准。

同构数据库迁移

将源数据库迁移到共享同一数据库引擎的目标数据库（例如，从 Microsoft SQL Server 迁移到 Amazon RDS for SQL Server）。同构迁移通常是更换主机或更换平台工作的一部分。您可以使用本机数据库实用程序来迁移架构。

热数据

经常访问的数据，例如实时数据或近期的转化数据。这些数据通常需要高性能存储层或存储类别才能提供快速的查询响应。

修补程序

针对生产环境中关键问题的紧急修复。由于其紧迫性，修补程序通常是在典型的 DevOps 发布工作流程之外进行的。

hypercare 周期

割接之后，迁移团队立即管理和监控云中迁移的应用程序以解决任何问题的时间段。通常，这个周期持续 1-4 天。在 hypercare 周期结束时，迁移团队通常会将应用程序的责任移交给云运营团队。

我

IaC

请参阅基础设施即代码。

基于身份的策略

附加到一个或多个 IAM 委托人的策略，用于定义他们在 AWS Cloud 环境中的权限。

空闲应用程序

90 天内平均 CPU 和内存使用率在 5% 到 20% 之间的应用程序。在迁移项目中，通常会停用这些应用程序或将其保留在本地。

IIoT

请参阅工业物联网。

不可变基础设施

一种模型，可为生产工作负载部署新的基础设施，而不是更新、修补或修改现有基础设施。不可变基础设施本质上比可变基础设施更一致、更可靠、更可预测。有关更多信息，请参阅框架中的使用不可变基础架构部署最佳实践。 AWS Well-Architected

入站（入口）VPC

在 AWS 多账户架构中，一种接受、检查和路由来自应用程序外部的网络连接的 VPC。AWS 安全参考架构建议使用入站、出站和检查 VPC 设置网络账户，保护应用程序与广泛的互联网之间的双向接口。

增量迁移

一种割接策略，在这种策略中，您可以将应用程序分成小部分进行迁移，而不是一次性完整割接。例如，您最初可能只将几个微服务或用户迁移到新系统。在确认一切正常后，您可以逐步迁移其他微服务或用户，直到停用遗留系统。这种策略降低了大规模迁移带来的风险。

工业 4.0

该术语由克劳斯·施瓦布（Klaus Schwab）在2016年推出，指的是通过连接性、实时数据、自动化、分析和的进步实现制造流程的现代化。 AI/ML

基础设施

应用程序环境中包含的所有资源和资产。

基础设施即代码（IaC）

通过一组配置文件预调配和管理应用程序基础设施的过程。IaC 旨在帮助您集中管理基础设施、实现资源标准化和快速扩展，使新环境具有可重复性、可靠性和一致性。

工业物联网（IIoT）

在工业领域使用联网的传感器和设备，例如制造业、能源、汽车、医疗保健、生命科学和农业。有关更多信息，请参阅制定工业物联网（IIoT）数字化转型策略。

检查 VPC

在 AWS 多账户架构中，一种集中式 VPC，用于管理 VPC（相同或不同 AWS 区域）、互联网和本地网络之间的网络流量检查。AWS 安全参考架构建议使用入站、出站和检查 VPC 设置网络账户，保护应用程序与广泛的互联网之间的双向接口。

物联网（IoT）

由带有嵌入式传感器或处理器的连接物理对象组成的网络，这些传感器或处理器通过互联网或本地通信网络与其他设备和系统进行通信。有关更多信息，请参阅什么是 IoT？

可解释性

它是机器学习模型的一种特征，描述了人类可以理解模型的预测如何取决于其输入的程度。有关更多信息，请参阅机器学习模型的可解释性 AWS。

物联网

请参阅物联网。

IT 信息库（ITIL）

提供 IT 服务并使这些服务符合业务要求的一套最佳实践。ITIL 是 ITSM 的基础。

IT 服务管理（ITSM）

为组织设计、实施、管理和支持 IT 服务的相关活动。有关将云运营与 ITSM 工具集成的信息，请参阅运营集成指南。

ITIL

请参阅 IT 信息库。

ITSM

请参阅 IT 服务管理。

L

基于标签的访问控制（LBAC）

强制访问控制（MAC）的一种实施方式，其中明确为用户和数据本身分配了安全标签值。用户安全标签和数据安全标签之间的交集决定了用户可以看到哪些行和列。

登录区

landing zone 是一个架构精良的多账户 AWS 环境，具有可扩展性和安全性。这是一个起点，您的组织可以从这里放心地在安全和基础设施环境中快速启动和部署工作负载和应用程序。有关登录区的更多信息，请参阅设置安全且可扩展的多账户 AWS 环境。

大语言模型（LLM）

一种基于大量数据进行预训练的深度学习 AI 模型。LLM 可以执行多项任务，例如回答问题、总结文档、将文本翻译成其他语言以及完成句子。有关更多信息，请参阅什么是 LLM。

大规模迁移

迁移 300 台或更多服务器。

LBAC

请参阅基于标签的访问控制。

最低权限

授予执行任务所需的最低权限的最佳安全实践。有关更多信息，请参阅 IAM 文档中的应用最低权限许可。

直接迁移

请参阅 7 R。

小端序系统

一个先存储最低有效字节的系统。另请参阅字节顺序。

LLM

请参阅大型语言模型。

下层环境

请参阅环境。

M

机器学习（ML）

一种使用算法和技术进行模式识别和学习的人工智能。ML 对记录的数据（例如物联网（IoT）数据）进行分析和学习，以生成基于模式的统计模型。有关更多信息，请参阅机器学习。

主分支

请参阅分支。

恶意软件

旨在危害计算机安全或隐私的软件。恶意软件可能会破坏计算机系统、泄露敏感信息或获得未经授权的访问权限。恶意软件的示例包括病毒、蠕虫、勒索软件、木马、间谍软件和键盘记录器。

托管式服务

AWS 服务 它 AWS 运行基础设施层、操作系统和平台，您可以访问端点来存储和检索数据。Amazon Simple Storage Service（Amazon S3）和 Amazon DynamoDB 就是托管服务的示例。这些服务也称为抽象服务。

制造执行系统（MES）

一种软件系统，用于跟踪、监控、记录和控制在车间将原材料转化为成品的生产过程。

MAP

请参阅迁移加速计划。

MCP

参见模型上下文协议。

模型上下文协议（MCP）

一种用于代理与工具通信的无状态协议。

MCP 服务器

一种通过模型上下文协议公开一个或多个工具的服务。

机制

一个完整的过程，您可以在其中创建工具，推动工具的采用，然后检查结果以进行调整。机制是一种在运作过程中自我强化和改善的循环。有关更多信息，请参阅在 AWS Well-Architected 框架中构建机制。

成员账户

AWS 账户 除属于组织中的管理账户之外的所有账户 AWS Organizations。一个账户一次只能是一个组织的成员。

MES

请参阅制造执行系统。

消息队列遥测传输（MQTT）

一种基于publish/subscribe模式的轻量级机器对机器 (M2M) 通信协议，适用于资源受限的物联网设备。

微服务

一种小型独立服务，通过明确定义的 API 进行通信，通常由小型独立团队拥有。例如，保险系统可能包括映射到业务能力（如销售或营销）或子域（如购买、理赔或分析）的微服务。微服务的好处包括敏捷、灵活扩展、易于部署、可重复使用的代码和恢复能力。有关更多信息，请参阅使用 AWS 无服务器服务集成微服务。

微服务架构

一种使用独立组件构建应用程序的方法，这些组件将每个应用程序进程作为微服务运行。这些微服务使用轻量级 API 通过明确定义的接口进行通信。该架构中的每个微服务都可以更新、部署和扩展，以满足对应用程序特定功能的需求。有关更多信息，请参阅在上实现微服务。 AWS

迁移加速计划（MAP）

AWS 该计划提供咨询支持、培训和服务，以帮助组织为迁移到云奠定坚实的运营基础，并帮助抵消迁移的初始成本。MAP 提供了一种以系统的方式执行遗留迁移的迁移方法，以及一套用于自动执行和加速常见迁移场景的工具。

大规模迁移

将大部分应用程序组合分波迁移到云中的过程，在每一波中以更快的速度迁移更多应用程序。本阶段使用从早期阶段获得的最佳实践和经验教训，实施由团队、工具和流程组成的迁移工厂，通过自动化和敏捷交付简化工作负载的迁移。这是 AWS 迁移策略的第三阶段。

迁移工厂

Cross-functional 通过自动化、敏捷的方法简化工作负载迁移的团队。迁移工厂团队通常包括运营、业务分析师和所有者、迁移工程师、开发 DevOps 人员和冲刺专业人员。20% 到 50% 的企业应用程序组合由可通过工厂方法优化的重复模式组成。有关更多信息，请参阅本内容集中有关迁移工厂的讨论和云迁移工厂指南。

迁移元数据

有关完成迁移所需的应用程序和服务器的信息。每种迁移模式都需要一套不同的迁移元数据。迁移元数据的示例包括目标子网、安全组和 AWS 账户。

迁移模式

一种可重复的迁移任务，详细列出了迁移策略、迁移目标以及所使用的迁移应用程序或服务。示例：使用 AWS 应用程序迁移服务重新托管向 Amazon EC2 的迁移。

迁移组合评测（MPA）

一种在线工具，提供了用于验证迁移到 AWS Cloud的业务案例的信息。MPA 提供了详细的组合评测（服务器规模调整、定价、TCO 比较、迁移成本分析）以及迁移计划（应用程序数据分析和数据收集、应用程序分组、迁移优先级排序和波次规划）。所有 AWS 顾问和 APN 合作伙伴顾问均可免费使用 MPA 工具（需要登录）。

迁移准备情况评测（MRA）

使用 AWS CAF 深入了解组织的云就绪状态、确定优势和劣势以及制定行动计划以缩小已发现差距的过程。有关更多信息，请参阅迁移准备指南。MRA 是 AWS 迁移策略的第一阶段。

迁移策略

将工作负载迁移到 AWS Cloud的方法。有关更多信息，请参见术语表中的 7 R 词条，以及动员您的组织以加快大规模迁移。

ML

请参阅机器学习。

现代化

将过时的（原有的或单体）应用程序及其基础设施转变为云中敏捷、弹性和高度可用的系统，以降低成本、提高效率和利用创新。有关更多信息，请参阅在 AWS Cloud中实现应用程序现代化的策略。

现代化准备情况评估

一种评估方式，有助于确定组织应用程序的现代化准备情况；确定收益、风险和依赖关系；确定组织能够在多大程度上支持这些应用程序的未来状态。评估结果是目标架构的蓝图、详细说明现代化进程发展阶段和里程碑的路线图以及解决已发现差距的行动计划。有关更多信息，请参阅在 AWS Cloud中评估应用程序的现代化准备情况。

单体应用程序（单体式）

作为具有紧密耦合进程的单个服务运行的应用程序。单体应用程序有几个缺点。如果某个应用程序功能的需求激增，则必须扩展整个架构。随着代码库的增长，添加或改进单体应用程序的功能也会变得更加复杂。若要解决这些问题，可以使用微服务架构。有关更多信息，请参阅将单体分解为微服务。

MPA

请参阅迁移组合评测。

MQTT

请参阅消息队列遥测传输。

多分类器

一种帮助为多个类别生成预测（预测两个以上结果之一）的过程。例如，ML 模型可能会询问“这个产品是书、汽车还是手机？”或“此客户最感兴趣什么类别的产品？”

可变基础设施

一种用于更新和修改生产工作负载的现有基础设施的模型。为了提高一致性、可靠性和可预测性，该 AWS Well-Architected 框架建议使用不可变基础设施作为最佳实践。

O

OAC

请参阅来源访问控制。

OAI

请参阅来源访问身份。

OCM

请参阅组织变革管理。

离线迁移

一种迁移方法，在这种方法中，源工作负载会在迁移过程中停止运行。这种方法会延长停机时间，通常用于小型非关键工作负载。

OI

请参阅运营集成。

OLA

请参阅运营级别协议。

在线迁移

一种迁移方法，在这种方法中，源工作负载无需离线即可复制到目标系统。在迁移过程中，连接工作负载的应用程序可以继续运行。这种方法的停机时间为零或最短，通常用于关键生产工作负载。

OPC-UA

请参阅开放流程通信 – 统一架构。

开放流程通信-统一架构 (OPC-UA)

一种用于工业自动化的机器对机器 (M2M) 通信协议。 OPC-UA 提供了数据加密、身份验证和授权方案的互操作性标准。

运营级别协议（OLA）

一项协议，阐明了 IT 职能部门承诺相互交付的内容，以支持服务水平协议（SLA）。

运营准备情况审查（ORR）

一份问题核对清单和关联的最佳实践，可帮助您了解、评估、预防或缩小事件和可能的故障的范围。有关更多信息，请参阅 AWS Well-Architected 框架中的运营准备情况审查 (ORR)。

运营技术（OT）

与物理环境配合使用以控制工业运营、设备和基础设施的硬件和软件系统。在制造业中，OT 和信息技术（IT）系统的集成是工业 4.0 转型的关键重点。

运营整合（OI）

在云中实现运营现代化的过程，包括就绪计划、自动化和集成。有关更多信息，请参阅运营整合指南。

组织跟踪

由 AWS CloudTrail 此创建的跟踪记录组织 AWS 账户 中所有人的所有事件 AWS Organizations。该跟踪是在每个 AWS 账户 中创建的，属于组织的一部分，并跟踪每个账户的活动。有关更多信息，请参阅 CloudTrail文档中的为组织创建跟踪。

组织变革管理（OCM）

一个从人员、文化和领导力角度管理重大、颠覆性业务转型的框架。OCM 通过加快变革采用、解决过渡问题以及推动文化和组织变革，帮助组织为新系统和战略做好准备和过渡。在 AWS 迁移策略中，该框架被称为人员加速，因为云采用项目需要变更的速度。有关更多信息，请参阅 OCM 指南。

来源访问控制（OAC）

在中 CloudFront，一个增强的选项，用于限制访问以保护您的亚马逊简单存储服务 (Amazon S3) 内容。OAC 全部支持所有 S3 存储桶 AWS 区域、使用 AWS KMS (SSE-KMS) 进行服务器端加密，以及对 S3 存储桶的动态PUT和DELETE请求。

来源访问身份（OAI）

在中 CloudFront，一个用于限制访问权限以保护您的 Amazon S3 内容的选项。当您使用 OAI 时，CloudFront 会创建一个 Amazon S3 可以对其进行身份验证的委托人。经过身份验证的委托人只能通过特定 CloudFront 分配访问 S3 存储桶中的内容。另请参阅 OAC，其中提供了更精细和增强的访问控制。

ORR

请参阅运营准备情况审查。

OT

请参阅运营技术。

出站（出口）VPC

在 AWS 多账户架构中，一种处理从应用程序内部启动的网络连接的 VPC。AWS 安全参考架构建议使用入站、出站和检查 VPC 设置网络账户，保护应用程序与广泛的互联网之间的双向接口。

P

权限边界

附加到 IAM 主体的 IAM 管理策略，用于设置用户或角色可以拥有的最大权限。有关更多信息，请参阅 IAM 文档中的权限边界。

个人身份信息（PII）

直接查看其他相关数据或与之配对时可用于合理推断个人身份的信息。PII 的示例包括姓名、地址和联系信息。

PII

请参阅个人身份信息。

playbook

一套预定义的步骤，用于捕获与迁移相关的工作，例如在云中交付核心运营功能。playbook 可以采用脚本、自动化运行手册的形式，也可以是操作现代化环境所需的流程或步骤的摘要。

PLC

请参阅可编程逻辑控制器。

PLM

请参阅产品生命周期管理。

policy

一个对象，可以定义权限（请参阅基于身份的策略）、指定访问条件（请参阅基于资源的策略）或定义 AWS Organizations 的组织中所有账户的最大权限（请参阅服务控制策略）。

多语言持久性

根据数据访问模式和其他要求，独立选择微服务的数据存储技术。如果您的微服务采用相同的数据存储技术，它们可能会遇到实现难题或性能不佳。如果微服务使用最适合其需求的数据存储，则可以更轻松地实现微服务，并获得更好的性能和可扩展性。

组合评测

一个发现、分析和确定应用程序组合优先级以规划迁移的过程。有关更多信息，请参阅评估迁移准备情况。

谓词

返回 `true` 或 `false` 的查询条件，通常位于 `WHERE` 子句中。

谓词下推

一种数据库查询优化技术，可在传输之前筛选查询中的数据。这将减少从关系数据库检索和处理的数据量，并提高查询性能。

预防性控制

一种安全控制，旨在防止事件发生。这些控制是第一道防线，帮助防止未经授权的访问或对网络的意外更改。有关更多信息，请参阅在 AWS上实施安全控制中的预防性控制。

主体

中 AWS 可以执行操作和访问资源的实体。此实体通常是 IAM 角色的根用户或用户。 AWS 账户有关更多信息，请参阅 IAM 文档中角色术语和概念中的主体。

隐私设计

一种在整个开发过程中都考虑隐私的系统工程方法。

私有托管区

私有托管区就是一个容器，其中包含的信息说明您希望 Amazon Route 53 如何响应一个或多个 VPC 中的某个域及其子域的 DNS 查询。有关更多信息，请参阅 Route 53 文档中的私有托管区的使用。

主动控制

一种安全控制，旨在防止部署不合规资源。这些控制会在资源预置之前对其进行扫描。如果资源与控制不兼容，则不会预置它。有关更多信息，请参阅 AWS Control Tower 文档中的控制参考指南，并参见在上实施安全控制中的主动控制 AWS。

产品生命周期管理（PLM）

对产品在其整个生命周期内的数据和流程的管理，从设计、开发和发布，到增长和成熟，再到衰退和淘汰。

生产环境

请参阅环境。

可编程逻辑控制器（PLC）

在制造业中，一种高度可靠、适应性强的计算机，用于监控机器并实现制造过程自动化。

提示串接

使用一个 LLM 提示的输出作为下一个提示的输入，以生成更好的响应。该技术用于将复杂的任务分解为子任务，或者迭代地完善或扩展初步响应。它有助于提高模型响应的准确性和相关性，并允许获得更精细的个性化结果。

假名化

用占位符值替换数据集中个人标识符的过程。假名化可以帮助保护个人隐私。假名化数据仍被视为个人数据。

publish/subscribe (pub/sub)

一种支持微服务间异步通信的模式，可提高可扩展性和响应能力。例如，在基于微服务的 MES 中，微服务可以将事件消息发布到其他微服务可以订阅的频道。系统可以在不更改发布服务的情况下添加新的微服务。

Q

查询计划

一系列用于访问 SQL 关系数据库系统中的数据的步骤，类似于指令。

查询计划回归

当数据库服务优化程序选择的最佳计划不如数据库环境发生特定变化之前时。这可能是由统计数据、约束、环境设置、查询参数绑定更改和数据库引擎更新造成的。

R

RACI 矩阵

请参阅责任、问责、咨询和知情（RACI）。

RAG

请参阅检索增强生成。

勒索软件

一种恶意软件，旨在阻止对计算机系统或数据的访问，直到付款为止。

RASCI 矩阵

请参阅责任、问责、咨询和知情（RACI）。

RCAC

请参阅行列访问控制。

只读副本

用于只读目的的数据库副本。您可以将查询路由到只读副本，以减轻主数据库的负载。

重新架构

请参阅 7 R。

恢复点目标（RPO）

自上一个数据恢复点以来可接受的最长时间。这决定了从上一个恢复点到服务中断之间可接受的数据丢失情况。

恢复时间目标（RTO）

服务中断和服务恢复之间可接受的最大延迟。

重构

请参阅 7 R。

Region

地理区域内的 AWS 资源集合。每一个 AWS 区域 都相互隔离，彼此独立，以提供容错、稳定性和弹性。有关更多信息，请参阅指定您的账户可以使用的 AWS 区域。

回归

一种预测数值的 ML 技术。例如，要解决“这套房子的售价是多少？”的问题 ML 模型可以使用线性回归模型，根据房屋的已知事实（如建筑面积）来预测房屋的销售价格。

重新托管

请参阅 7 R。

版本

在部署过程中，推动生产环境变更的行为。

重新放置

请参阅 7 R。

更换平台

请参阅 7 R。

重新购买

请参阅 7 R。

韧性

应用程序抵御中断或从中断中恢复的能力。在 AWS Cloud中规划韧性时，高可用性和灾难恢复是常见的考虑因素。有关更多信息，请参阅 AWS Cloud 韧性。

基于资源的策略

一种附加到资源的策略，例如 AmazonS3 存储桶、端点或加密密钥。此类策略指定了允许哪些主体访问、支持的操作以及必须满足的任何其他条件。

责任、问责、咨询和知情（RACI）矩阵

定义参与迁移活动和云运营的所有各方的角色和责任的矩阵。矩阵名称源自矩阵中定义的责任类型：负责（R）、问责（A）、咨询（C）和知情（I）。支持（S）类型是可选的。如果包括支持，则该矩阵称为 RASCI 矩阵，如果将其排除在外，则称为 RACI 矩阵。

响应性控制

一种安全控制，旨在推动对不良事件或偏离安全基线的情况进行修复。有关更多信息，请参阅在AWS上实施安全控制中的响应性控制。

保留

请参阅 7 R。

停用

请参阅 7 R。

检索增强生成（RAG）

一种生成式人工智能技术，其中 LLM 在生成响应之前引用其训练数据来源之外的权威数据来源。例如，RAG 模型可以对组织的知识库或自定义数据执行语义搜索。有关更多信息，请参阅什么是 RAG。

轮换

定期更新密钥以使攻击者更难访问凭证的过程。

行列访问控制（RCAC）

使用已定义访问规则的基本、灵活的 SQL 表达式。RCAC 由行权限和列掩码组成。

RPO

请参阅恢复点目标。

RTO

请参阅恢复时间目标。

运行手册

执行特定任务所需的一套手动或自动程序。它们通常是为了简化重复性操作或高错误率的程序而设计的。

S

SAML 2.0

许多身份提供商 (IdPs) 使用的开放标准。此功能支持联合单点登录 (SSO)，因此用户无需在 IAM 中为组织中的所有人创建用户即可登录 AWS 管理控制台 或调用 AWS API 操作。有关基于 SAML 2.0 的联合身份验证的更多信息，请参阅 IAM 文档中的关于基于 SAML 2.0 的联合身份验证。

SCADA

请参阅监督控制和数据采集。

SCP

请参阅服务控制策略。

机密密钥

在中 AWS Secrets Manager，您以加密形式存储的机密或受限信息，例如密码或用户凭证。它由密钥值及其元数据组成。密钥值可以是二进制、单个字符串或多个字符串。有关更多信息，请参阅 Secrets Manager 文档中的什么是 Amazon Secrets Manager 密钥？。

安全设计

一种在整个开发过程中都考虑安全的系统工程方法。

安全控制

一种技术或管理防护机制，可防止、检测或降低威胁行为体利用安全漏洞的能力。安全控制有以下四种类型：预防性、检测性、响应性和主动性。

安全固化

缩小攻击面，使其更能抵御攻击的过程。这可能包括删除不再需要的资源、实施授予最低权限的最佳安全实践或停用配置文件中不必要的功能等操作。

安全信息和事件管理（SIEM）系统

结合了安全信息管理（SIM）和安全事件管理（SEM）系统的工具和服务。SIEM 系统会收集、监控和分析来自服务器、网络、设备和其他来源的数据，以检测威胁和安全漏洞，并生成警报。

安全响应自动化

一种预定义的程序化操作，旨在自动响应或修复安全事件。这些自动化可作为侦探或响应式安全控制措施，帮助您实施 AWS 安全最佳实践。自动响应操作的示例包括修改 VPC 安全组、修补 Amazon EC2 实例或轮换凭证。

服务器端加密

由接收数据的人在目的地对数据 AWS 服务 进行加密。

服务控制策略（SCP）

一种策略，用于集中控制 AWS Organizations的组织中所有账户的权限。SCP 为管理员可以委托给用户或角色的操作定义了防护机制或设定了限制。您可以将 SCP 用作允许列表或拒绝列表，指定允许或禁止哪些服务或操作。有关更多信息，请参阅 AWS Organizations 文档中的服务控制策略。

服务端点

的入口点的 URL AWS 服务。您可以使用端点，通过编程方式连接到目标服务。有关更多信息，请参阅 AWS 一般参考 中的 AWS 服务 端点。

服务水平协议（SLA）

一份协议，阐明了 IT 团队承诺向客户交付的内容，比如服务正常运行时间和性能。

服务水平指示器（SLI）

对服务性能方面的衡量，例如错误率、可用性或吞吐量。

服务水平目标（SLO）

代表服务运行状况的目标指标，由服务水平指示器衡量。

责任共担模式

描述您在云安全与合规方面共同承担 AWS 的责任的模型。 AWS 负责云的安全，而您则负责云中的安全。有关更多信息，请参阅责任共担模式。

暗影人工智能

在组织内受管控渠道之外构建或使用的未经授权的 AI 应用程序。

SIEM

请参阅安全信息和事件管理系统。

单点故障（SPOF）

应用程序的单个关键组件出现故障，可能会中断系统。

SLA

请参阅服务水平协议。

SLI

请参阅服务水平指示器。

SLO

请参阅服务水平目标。

split-and-seed 模式

一种扩展和加速现代化项目的模式。随着新功能和产品发布的定义，核心团队会拆分以创建新的产品团队。这有助于扩展组织的能力和服务，提高开发人员的工作效率，支持快速创新。有关更多信息，请参阅在 AWS Cloud中实现应用程序现代化的分阶段方法。

SPOF

请参阅单点故障。

星型架构

一种数据库组织结构，它使用一个大型事实表来存储事务数据或测量数据，并使用一个或多个较小的维度表来存储数据属性。此结构专为在数据仓库中使用或用于商业智能目的而设计。

strangler fig 模式

一种通过逐步重写和替换系统功能直至可以停用原有的系统来实现单体系统现代化的方法。这种模式用无花果藤作为类比，这种藤蔓成长为一棵树，最终战胜并取代了宿主。该模式是由 Martin

Fowler 提出的，作为重写单体系统时管理风险的一种方法。有关如何应用此模式的示例，请参阅使用容器和 Amazon API Gateway 逐步实现传统微软 ASP.NET (ASMX) 网络服务的现代化。

子网

您的 VPC 内的一个 IP 地址范围。子网必须位于单个可用区中。

监督控制和数据采集（SCADA）

在制造业中，一种使用硬件和软件来监控实物资产和生产操作的系统。

对称加密

一种加密算法，它使用相同的密钥来加密和解密数据。

综合测试

以模拟用户交互的方式测试系统，以检测潜在问题或监控性能。你可以使用 Amazon S CloudWatch ynthetics 来创建这些测试。

系统提示

一种为 LLM 提供上下文、说明或准则以指导其行为的技术。系统提示有助于设置上下文并制定与用户交互的规则。

T

标签

Key-value 对充当用于组织 AWS 资源的元数据。标签有助于您管理、识别、组织、搜索和筛选 资源。有关更多信息，请参阅标记您的 AWS 资源。

目标变量

您在监督式 ML 中尝试预测的值。这也被称为结果变量。例如，在制造环境中，目标变量可能是产品缺陷。

任务列表

一种通过运行手册用于跟踪进度的工具。任务列表包含运行手册的概述和要完成的常规任务列表。对于每项常规任务，它包括预计所需时间、所有者和进度。

测试环境

请参阅环境。

训练

为您的 ML 模型提供学习数据。训练数据必须包含正确答案。学习算法在训练数据中查找将输入数据属性映射到目标（您希望预测的答案）的模式。然后输出捕获这些模式的 ML 模型。然后，您可以使用 ML 模型对不知道目标的新数据进行预测。

工具

代理可以调用以在外部系统中执行操作的函数或 API。

中转网关

中转网关是网络中转中心，您可用它来互连 VPC 和本地网络。有关更多信息，请参阅 AWS Transit Gateway 文档中的什么是公交网关。

基于中继的工作流程

一种方法，开发人员在功能分支中本地构建和测试功能，然后将这些更改合并到主分支中。然后，按顺序将主分支构建到开发、预生产和生产环境。

可信访问权限

向您指定的服务授予权限，该服务可以代表您在其账户中执行任务。 AWS Organizations 当需要服务相关的角色时，受信任的服务会在每个账户中创建一个角色，为您执行管理任务。有关更多信息，请参阅 AWS Organizations 文档中的AWS Organizations 与其他 AWS 服务一起使用。

优化

更改训练过程的各个方面，以提高 ML 模型的准确性。例如，您可以通过生成标签集、添加标签，并在不同的设置下多次重复这些步骤来优化模型，从而训练 ML 模型。

双披萨团队

一个小 DevOps 团队，你可以用两个披萨来喂食。双披萨团队的规模可确保在软件开发过程中充分协作。

U

不确定性

这一概念指的是不精确、不完整或未知的信息，这些信息可能会破坏预测式 ML 模型的可靠性。不确定性有两种类型：认知不确定性是由有限的、不完整的数据造成的，而偶然不确定性是由数据中固有的噪声和随机性导致的。

无差别任务

也称为繁重工作，即创建和运行应用程序所必需的工作，但不能为最终用户提供直接价值或竞争优势。无差别任务的示例包括采购、维护和容量规划。

上层环境

请参阅环境。

V

vacuum 操作

一种数据库维护操作，包括在增量更新后进行清理，以回收存储空间并提高性能。

版本控制

跟踪更改的过程和工具，例如存储库中源代码的更改。

VPC 对等连接

两个 VPC 之间的连接，允许您使用私有 IP 地址路由流量。有关更多信息，请参阅 Amazon VPC 文档中的什么是 VPC 对等连接。

漏洞

损害系统安全的软件缺陷或硬件缺陷。

W

热缓存

一种包含经常访问的当前相关数据的缓冲区缓存。数据库实例可以从缓冲区缓存读取，这比从主内存或磁盘读取要快。

暖数据

不常访问的数据。查询此类数据时，通常可以接受中速查询。

窗口函数

一种对与当前记录有某种关联的一组行执行计算的 SQL 函数。窗口函数对于处理任务很有用，例如计算移动平均值或根据当前行的相对位置访问行的值。

工作负载

一系列资源和代码，它们可以提供商业价值，如面向客户的应用程序或后端过程。

工作流

迁移项目中负责一组特定任务的职能小组。每个工作流都是独立的，但支持项目中的其他工作流。例如，组合工作流负责确定应用程序的优先级、波次规划和收集迁移元数据。组合工作流将这些资产交付给迁移工作流，然后迁移服务器和应用程序。

WORM

请参阅一次写入多次读取。

WQF

请参阅 AWS 工作负载资格鉴定框架。

一次写入多次读取（WORM）

一种存储模型，可一次写入数据并防止数据被删除或修改。授权用户可以根据需要多次读取数据，但无法对其进行更改。此数据存储基础设施被认为不可变。

Z

零日漏洞利用

一种利用零日漏洞的攻击，通常为恶意软件。

零日漏洞

生产系统中不可避免的缺陷或漏洞。威胁主体可能利用这种类型的漏洞攻击系统。开发人员经常因攻击而意识到该漏洞。

零样本提示

为 LLM 提供执行任务的说明，但没有可以帮助指导的示例（样本）。LLM 必须使用预先训练的知识来处理任务。零样本提示的有效性取决于任务的复杂性和提示的质量。另请参阅少样本提示。

僵尸应用程序

平均 CPU 和内存使用率低于 5% 的应用程序。在迁移项目中，通常会停用这些应用程序。

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