layer AWS CDK 指南

AWS 方案指引

AWS 方案指引: layer AWS CDK 指南

Table of Contents

layer AWS CDK 指南

Steven Guggenheimer，Amazon Web Services (AWS)

2023 年 12 月 (文件歷史記錄)

背後的主要概念之一 AWS Cloud Development Kit (AWS CDK) 與在冷天保持暖和的概念非常相似。該概念稱為分層。在冷天穿著襯衫、夾克，有時甚至更大型的夾克，具體取決於它有多冷。然後，如果您進入其中且加熱器正在閃電，您可以脫下一個或兩個夾克層，這樣您就不會太熱。 AWS CDK 使用分層為使用雲端元件提供不同層級的抽象。分層可確保在部署基礎設施做為程式碼 (IAC) 堆疊時，您永遠不需要撰寫太多程式碼或對資源屬性的存取過少。

如果您不使用 AWS CDK，則必須手動撰寫AWS CloudFormation範本；也就是說，您只會利用單一 layer 來強制您撰寫比平常更多的程式碼。另一方面，如果 AWS CDK 是要抽象化 CloudFormation 中通常不需要寫出的所有內容，您將無法處理任何邊緣案例。

為了解決此問題， 會將資源佈建 AWS CDK 分割成三個不同的層級：

- 第 1 層 – CloudFormation 層：CloudFormation 資源和資源 AWS CDK 幾乎完全相同的最基本層。
- 圖層 2 – 精選圖層：CloudFormation 資源抽象至程式設計類別的圖層，可簡化機庫下的大部分樣板 CloudFormation 語法。此 layer 組成大部分的 AWS CDK。
- 第 3 層 – 模式層：最抽象的層，您可以使用第 1 層和第 2 層提供的建置區塊來自訂特定使用案例的程式碼。

每個 layer 中的每個項目都是稱為 之特殊 AWS CDK 類別的執行個體`Construct`。根據AWS 文件，建構是「 AWS CDK 應用程式的基本建置區塊。建構代表「雲端元件」，並封裝建立元件 AWS CloudFormation 所需的一切。」 這些層內的建構稱為 L1、L2 和 L3 建構，取決於其所屬層。在本指南中，我們將瀏覽每一 AWS CDK 層，以了解它們的用途及其重要性。

本指南適用於技術經理、主管和開發人員，他們有興趣更深入探索讓 AWS CDK 工作成為工作的核心概念。 AWS CDK 是熱門的工具，但團隊通常會錯過其提供的大部分內容。當您開始了解本指南中所述的概念時，您可以釋放充滿可能性的全新世界，並最佳化團隊的資源佈建程序。

在本指南中：

- 最佳實務
- 常見問答集
- 資源

第 1 層建構

L1 建構模組是 的建置區塊， AWS CDK 且很容易與其他建構模組區別為字首 `Cfn`。例如， 中的 Amazon DynamoDB `Table` 套件 AWS CDK 包含一個建構，即 L2 建構。對應的 L1 建構稱為 `CfnTable`，直接代表 CloudFormation DynamoDB `Table`。雖然 AWS CDK 應用程式通常不會直接使用 L1 建構，但無法在 AWS CDK 未存取第一層的情況下使用 。不過，在大多數情況下，開發人員習慣使用的 L2 和 L3 建構會大量依賴 L1 建構。因此，您可以將 L1 建構視為 CloudFormation 與 之間的橋樑 AWS CDK。

的唯一目的是使用標準編碼語言 AWS CDK 產生 CloudFormation 範本。在您執行 cdk 合成 CLI 命令並產生產生的 CloudFormation 範本後， AWS CDK任務即完成。cdk 部署命令僅為方便起見，但執行該命令時所執行的動作完全發生在 CloudFormation 中。將 AWS CDK 程式碼轉譯為 CloudFormation 理解格式的拼圖片段是 L1 建構。

L1 建構的 AWS CDK–CloudFormation 生命週期

建立和使用 L1 建構的程序包含下列步驟：

1. AWS CDK 建置程序會將 CloudFormation 規格轉換為 L1 建構的程式設計程式碼。
2. 開發人員編寫程式碼，直接或間接參考 L1 建構做為 AWS CDK 應用程式的一部分。
3. 開發人員執行 cdk 合成命令，將程式設計程式碼轉換回 CloudFormation 規格 （範本） 指定的格式。
4. 開發人員會執行 cdk 部署命令，在這些範本中將 CloudFormation 堆疊部署到 AWS 帳戶環境。

讓我們做一些練習。前往 GitHub 上的AWS CDK 開放原始碼儲存庫，挑選隨機 AWS 服務，然後前往該服務的 AWS CDK 套件 （位於資料夾 `packages`、`aws-cdk-lib`、`aws-<servicename>`、)`lib`。在此範例中，讓我們挑選 Amazon S3，但這適用於任何服務。如果您查看該套件的主要 index.ts 檔案，您會看到一行顯示：

```
export * from './s3.generated';
```

不過，您不會在對應目錄中的任何位置看到 `s3.generated` 檔案。這是因為 L1 建構會在 AWS CDK 建置程序期間從 CloudFormation 資源規格自動產生。因此，只有在您執行套件的 AWS CDK 建置命令之後，才會在套件`s3.generated`中看到 。

AWS CloudFormation 資源規格

AWS CloudFormation 資源規格將基礎設施定義為 的程式碼 (IAC)， AWS 並決定 CloudFormation 範本中的程式碼如何轉換為帳戶中 AWS 的資源。此規格會在每個區域層級上以 JSON 格式定義 AWS 資源。每個資源都會獲得遵循格式 的唯一資源類型名稱`provider::service::resource`。例如，Amazon S3 儲存貯體的資源類型名稱為 `AWS::S3::Bucket`，Amazon S3 存取點的資源類型名稱則為 `AWS::S3::AccessPoint`。您可以使用 資源規格中定義的語法，在 CloudFormation 範本中轉譯這些 AWS CloudFormation 資源類型。當 AWS CDK 建置程序執行時，每個資源類型也會成為 L1 建構。

因此，每個 L1 建構都是其對應 CloudFormation 資源的程式設計鏡像影像。當您執行個體化 L1 建構時，您可以在 CloudFormation 範本中套用的每個屬性都可用，而當您執行個體化對應的 L1 建構時，也需要每個必要的 CloudFormation 屬性做為引數。下表將 CloudFormation 範本中顯示的 S3 儲存貯體與定義為 an AWS CDK L1 建構的相同 S3 儲存貯體進行比較。

CloudFormation 範本

```
"amzns3demobucket": {
    "Type": "AWS::S3::Bucket",
    "Properties": {
      "BucketName": "amzn-s3-demo-
bucket",
      "BucketEncryption": {
        "ServerSideEncryptionConfig
uration": [
          {
            "ServerSideEncrypt
ionByDefault": {
              "SSEAlgorithm": "AES256"
            }
          }
        ]
      },
      "MetricsConfigurations": [
        {
          "Id": "myConfig"
        }
      ],
      "OwnershipControls": {
        "Rules": [
          {
```

L1 建構

```
new CfnBucket(this, "amzns3de
mobucket", {
  bucketName: "amzn-s3-demo-bucket",
  bucketEncryption: {
    serverSideEncryptionConfigu
ration: [
      {
        serverSideEncryptionByDefau
lt: {
          sseAlgorithm: "AES256"
        }
      }
    ]
  },
  metricsConfigurations: [
    {
      id: "myConfig"
    }
  ],
  ownershipControls: {
    rules: [
      {
        objectOwnership: "BucketOw
nerPreferred"
```

```
            "ObjectOwnership":
 "BucketOwnerPreferred"
          }
        ]
      },
      "PublicAccessBlockConfigura
tion": {
        "BlockPublicAcls": true,
        "BlockPublicPolicy": true,
        "IgnorePublicAcls": true,
        "RestrictPublicBuckets": true
      },
      "VersioningConfiguration": {
        "Status": "Enabled"
      }
    }
  }
```

```
      }
    ]
  },
  publicAccessBlockConfiguration: {
    blockPublicAcls: true,
    blockPublicPolicy: true,
    ignorePublicAcls: true,
    restrictPublicBuckets: true
  },
  versioningConfiguration: {
    status: "Enabled"
  }
});
```

如您所見，L1 建構是 CloudFormation 資源程式碼中的確切資訊清單。沒有捷徑或簡化，因此必須寫入的樣板文字量大致相同。不過，使用 AWS CDK 的一大優點是，它有助於消除許多樣板 CloudFormation 語法。那麼，這種情況如何發生？ 這就是 L2 建構的來源。

第 2 層建構

AWS CDK 開放原始碼儲存庫主要是使用 TypeScript 程式設計語言撰寫，由許多套件和模組組成。主要套件程式庫稱為 aws-cdk-lib，大致分為每個 AWS 服務一個套件，但情況並非總是如此。如前所述，L1 建構會在建置程序期間自動產生，所以當您查看儲存庫時，看到的所有程式碼為何？ 這些是 L2 建構，這是 L1 建構的抽象。

這些套件也包含 TypeScript 類型、列舉和介面的集合，以及新增更多功能的協助程式類別，但這些項目都提供 L2 建構。所有 L2 建構函數會在執行個體化時在其建構函數中呼叫其對應的 L1 建構函數，而建立的 L1 建構函數可以從第 2 層存取，如下所示：

```
const role = new Bucket(this, "amzn-s3-demo-bucket", {/*...BucketProps*/});
const cfnBucket = role.node.defaultChild;
```

L2 建構會使用預設屬性、便利方法和其他語法含糖，並將其套用至 L1 建構。這消除了直接在 CloudFormation 中佈建資源所需的大部分重複性和詳細程度。

所有 L2 建構模組都會在幕後建置其對應的 L1 建構模組。不過，L2 建構不會實際擴展 L1 建構。L1 和 L2 建構都會繼承稱為 Construct 的特殊類別。在 AWS CDK Construct 類別的第 1 版中， 類別是內建在開發套件中，但在第 2 版中則是獨立的套件。因此，雲端開發套件 (CDKTF) 等其他套件可以將其納入做為相依性。繼承類別的任何Construct類別都是 L1, L2 或 L3 建構。L2 建構模組會直接擴展此類別，而 L1 建構模組則會擴展名為 的類別CfnResource，如下表所示。

L1 繼承樹	L2 繼承樹
L1 建構	L2 建構
→ CfnResource 類別 CfnResource	→ 類別建構
→→ 抽象類別 CfnRefElement	
→→→ 抽象類別 CfnElement	
→→→→ 類別建構	

如果 L1 和 L2 建構都繼承 Construct類別，為什麼不 L2 建構只延伸 L1？ 類別Construct與層 1 之間的類別會將 L1 建構鎖定到位，做為 CloudFormation 資源的鏡像影像。它們包含抽象方法 （ 下

游類別必須包含的方法），例如 _toCloudFormation，這會強制建構直接輸出 CloudFormation 語法。L2 建構會略過這些類別，並直接擴展該Construct類別。這可讓他們彈性地抽象 L1 建構所需的大部分程式碼，方法是在建構函數中分別建置這些程式碼。

上一節針對來自 CloudFormation 範本的 S3 儲存貯體，以及轉譯為 L1 建構的相同 S3 儲存貯體，side-by-side比較。該比較顯示屬性和語法幾乎相同，L1 建構與 CloudFormation 建構相比，只會儲存三行或四行。現在，讓我們比較 L1 建構與相同 S3 儲存貯體的 L2 建構：

S3 儲存貯體的 L1 建構

```
new CfnBucket(this, "amzns3demobucket", {
  bucketName: "amzn-s3-demo-bucket",
  bucketEncryption: {
    serverSideEncryptionConfiguration: [
      {
        serverSideEncryptionByDefault: {
          sseAlgorithm: "AES256"
        }
      }
    ]
  },
  metricsConfigurations: [
    {
      id: "myConfig"
    }
  ],
  ownershipControls: {
    rules: [
      {
        objectOwnership: "BucketOwnerPreferred"
      }
    ]
  },
  publicAccessBlockConfiguration: {
    blockPublicAcls: true,
    blockPublicPolicy: true,
    ignorePublicAcls: true,
    restrictPublicBuckets: true
```

S3 儲存貯體的 L2 建構

```
new Bucket(this, "amzns3demobucket",
 {
  bucketName: "amzn-s3-demo-bucket",
  encryption: BucketEncryption.S3_MANAGED,
  metrics: [
    {
      id: "myConfig"
    },
  ],
  objectOwnership: ObjectOwnership.BUCKET_OWNER_PREFERRED,
  blockPublicAccess: BlockPublicAccess.BLOCK_ALL,
  versioned: true
});
```

```
  },
  versioningConfiguration: {
    status: "Enabled"
  }
});
```

如您所見，L2 建構體小於 L1 建構體大小的一半。L2 建構會使用多種技術來完成此整合。其中一些技術適用於單一 L2 建構模組，但其他技術可以在多個建構模組之間重複使用，以便將其分隔成自己的類別以重複使用。L2 建構會以多種方式合併 CloudFormation 語法，如以下章節所述。

預設屬性

合併用於佈建資源的程式碼最簡單的方法是將最常見的屬性設定轉換為預設值。 AWS CDK 可存取強大的程式設計語言，而 CloudFormation 則無法存取，因此這些預設值通常具有條件性。有時可以從 AWS CDK 程式碼中消除數行 CloudFormation 組態，因為這些設定可以從傳遞給建構模組的其他屬性值推斷。

結構、類型和界面

雖然 AWS CDK 提供多種程式設計語言，但會以 TypeScript 原生撰寫，因此語言的類型系統可用來定義組成 L2 建構的類型。深入探索該類型的系統超出本指南的範圍；如需詳細資訊，請參閱 TypeScript 文件。總而言之，TypeScript `type`會描述特定變數所保留的資料類型。這可以是基本資料，例如 `string`，或更複雜的資料，例如 `object`。TypeScript `interface`是表達 TypeScript 物件類型的另一種方式，而 `struct`是介面的另一種名稱。

TypeScript 不會使用 struct 一詞，但如果您在 AWS CDK API 參考中查看，您會看到 struct 實際上只是程式碼中的另一個 TypeScript 界面。API 參考也稱特定界面為界面。如果結構和界面是相同的，為什麼文件會 AWS CDK 區分它們？

AWS CDK 稱為 結構的界面代表 L2 建構所使用的任何物件。這包括在執行個體化期間傳遞給 L2 建構的屬性引數的物件類型，例如 `BucketProps` S3 儲存貯體建構`TableProps`和 DynamoDB Table 建構，以及 中使用的其他 TypeScript 介面 AWS CDK。簡而言之，如果它是 內的 TypeScript 介面 AWS CDK ，而且其名稱不是字母 的字首`I`，則 會將其 AWS CDK 呼叫為結構。

相反地， AWS CDK 使用 術語界面來表示純物件需要被視為特定建構函數或協助程式類別的適當表示的基本元素。也就是說，界面說明 L2 建構的公有屬性必須是什麼。所有 AWS CDK 界面名稱都是字母前綴的現有建構或協助程式類別的名稱`I`。所有 L2 建構模組都會擴展 `Construct`類別，但也會實作其對應的界面。因此 L2 建構會`Bucket`實作 `IBucket` 界面。

靜態方法

L2 建構的每個執行個體也是其對應界面的執行個體，但反向不是真的。這在查看結構時很重要，以查看需要哪些資料類型。如果結構具有稱為 的屬性bucket，其需要資料類型 IBucket，您可以傳遞包含IBucket介面中列出屬性的物件或 L2 的執行個體Bucket。其中一個都可以運作。不過，如果該bucket屬性針對 L2 呼叫 Bucket，則您只能在該欄位中傳遞Bucket執行個體。

當您將預先存在的資源匯入堆疊時，此區別變得非常重要。您可以為堆疊原生的任何資源建立 L2 建構，但如果您需要參考在堆疊外部建立的資源，則必須使用該 L2 建構的界面。這是因為如果 L2 建構模組不存在於該堆疊中，則建立該建構模組會建立新的資源。現有資源的參考必須是符合該 L2 建構的界面的純物件。

為了讓實務上更輕鬆，大多數 L2 建構都有一組與其相關聯的靜態方法，可傳回該 L2 建構的界面。這些靜態方法通常以 一詞開頭from。傳遞給這些方法的前兩個引數相同，scope且標準 L2 建構所需的id引數相同。不過，第三個引數不是props定義界面的一小部分屬性（有時只是一個屬性）。因此，當您傳遞 L2 建構時，在大多數情況下只需要界面的元素。這樣您就可以盡可能使用匯入的資源。

```
// Example of referencing an external S3 bucket
const preExistingBucket = Bucket.fromBucketName(this, "external-bucket", "name-of-
bucket-that-already-exists");
```

不過，您不應該高度依賴界面。您應該匯入資源並僅在絕對必要時直接使用界面，因為界面不提供許多屬性，例如協助程式方法，讓 L2 建構變得如此強大。

協助程式方法

L2 建構是程式設計類別，而不是簡單的物件，因此可以公開類別方法，允許您在執行個體化發生後操作資源組態。例如 AWS Identity and Access Management (IAM) L2 角色建構。下列程式碼片段顯示使用 L2 建構模組建立相同 IAM Role 角色的兩種方式。

沒有協助程式方法：

```
const role = new Role(this, "my-iam-role", {
   assumedBy: new FederatedPrincipal('my-identity-provider.com'),
   managedPolicies: [
      ManagedPolicy.fromAwsManagedPolicyName("ReadOnlyAccess")
   ],
   inlinePolicies: {
      lambdaPolicy: new PolicyDocument({
```

```
         statements: [
            new PolicyStatement({
               effect: Effect.ALLOW,
               actions: [ 'lambda:UpdateFunctionCode' ],
               resources: [ 'arn:aws:lambda:us-east-1:123456789012:function:my-
function' ]
            })
         ]
      })
   }
});
```

使用 協助程式方法：

```
const role = new Role(this, "my-iam-role", {
   assumedBy: new FederatedPrincipal('my-identity-provider.com')
});

role.addManagedPolicy(ManagedPolicy.fromAwsManagedPolicyName("ReadOnlyAccess"));
role.attachInlinePolicy(new Policy(this, "lambda-policy", {
   policyName: "lambdaPolicy",
   statements: [
      new PolicyStatement({
         effect: Effect.ALLOW,
         actions: [ 'lambda:UpdateFunctionCode' ],
         resources: [ 'arn:aws:lambda:us-east-1:123456789012:function:my-function' ]
      })
   ]
 }));
```

在執行個體化後使用執行個體方法來操作資源組態的功能，可讓 L2 建構比上一層更多的彈性。L1 建構也繼承一些資源方法（例如 addPropertyOverride），但直到第二層取得專為該資源及其屬性設計的方法，才會發生這種情況。

列舉

CloudFormation 語法通常需要您指定許多詳細資訊，才能正確佈建資源。不過，大多數使用案例通常只涵蓋少數組態。使用一系列列舉值來代表這些組態，可以大幅減少所需的程式碼數量。

例如，在本節稍早的 S3 儲存貯體 L2 程式碼範例中，您必須使用 CloudFormation 範本的 bucketEncryption 屬性來提供所有詳細資訊，包括要使用的加密演算法名稱。反之， AWS CDK

會提供BucketEncryption列舉，採用五種最常見的儲存貯體加密形式，並可讓您使用單一變數名稱來表達每個形式。

列舉未涵蓋的邊緣案例呢？ L2 建構的其中一個目標是簡化佈建第 1 層資源的任務，因此第 2 層可能不支援較不常用的特定邊緣案例。為了支援這些邊緣案例， AWS CDK 可讓您使用 addPropertyOverride 方法直接操作基礎 CloudFormation 資源屬性。如需屬性覆寫的詳細資訊，請參閱本指南的最佳實務一節，以及 AWS CDK 文件中的抽象和逃生艙一節。

協助程式類別

有時列舉無法完成為指定使用案例設定資源所需的程式設計邏輯。在這些情況下， AWS CDK 通常會改為提供協助程式類別。列舉是提供一系列鍵值對的簡單物件，而協助程式類別則提供 TypeScript 類別的完整功能。協助程式類別仍然可以透過公開靜態屬性來充當列舉，但這些屬性可以在內部使用協助程式類別建構函數或協助程式方法中的條件邏輯來設定其值。

因此，雖然列舉可以減少在 S3 BucketEncryption 儲存貯體上設定加密演算法所需的程式碼數量，但相同的策略無法用於設定時間持續時間，因為只有太多可能的值可供選擇。為每個值建立列舉會比它更麻煩。因此，協助程式類別用於 S3 儲存貯體的預設 S3 物件鎖定組態設定，如 ObjectLockRetention 類別所示。 ObjectLockRetention包含兩種靜態方法：一種用於合規保留，另一種用於控管保留。這兩種方法都需要持續時間協助程式類別的執行個體做為引數，以表示應該設定鎖定的時間量。

另一個範例是 AWS Lambda 協助程式類別執行期。乍看之下，與此類別相關聯的靜態屬性可能會由列舉處理。不過，在幕後，每個屬性值代表Runtime類別本身的執行個體，因此在類別建構函數中執行的邏輯無法在列舉內實現。

第 3 層建構

如果 L1 建構模組將 CloudFormation 資源轉譯為程式設計程式碼，而 L2 建構模組將大部分詳細 CloudFormation 語法取代為協助程式方法和自訂邏輯，L3 建構模組會做什麼？ 的答案僅受限於您的想像。您可以建立第 3 層，以符合任何特定使用案例。如果您的專案需要具有特定屬性子集的資源，您可以建立可重複使用的 L3 建構以滿足該需求。

L3 建構稱為 內的模式 AWS CDK。模式是延伸 `Construct`類別的任何物件 AWS CDK （或延伸延伸 `Construct` 類別的類別），以執行第 2 層以外的任何抽象邏輯。當您使用 AWS CDK CLI 執行 cdk init 來啟動新 AWS CDK 專案時，您必須從三種 AWS CDK 應用程式類型中選擇：`app`、 `lib`和 `sample-app`。

```
Available templates:
* app: Template for a CDK Application
   └─ cdk init app --language=[csharp|fsharp|go|java|javascript|python|typescript]
* lib: Template for a CDK Construct Library
   └─ cdk init lib --language=typescript
* sample-app: Example CDK Application with some constructs
   └─ cdk init sample-app --language=[csharp|fsharp|go|java|javascript|python|typescript]
```

`app` 和 `sample-app`都代表您在 AWS 環境中建置和部署 CloudFormation 堆疊的傳統 AWS CDK 應用程式。當您選擇 時`lib`，您選擇建置全新的 L3 建構。 `app` 並`sample-app`允許您選擇 AWS CDK 支援的任何語言，但您只能選擇 TypeScript 搭配 `lib`。這是因為 AWS CDK 以 TypeScript 原生寫入，並使用名為 JSii 的開放原始碼系統，將原始程式碼轉譯為其他支援的語言。當您選擇`lib`啟動專案時，您會選擇建置 的延伸 AWS CDK。

擴展類別的任何`Construct`類別都可以是 L3 建構，但第 3 層最常見的使用案例是資源互動、資源延伸和自訂資源。大多數 L3 建構會使用這三個案例的一或多個，以擴展 AWS CDK 功能。

資源互動

解決方案通常會採用數個可一起運作的 AWS 服務。例如，Amazon CloudFront 分佈通常會使用 S3 儲存貯體做為原始伺服器，並 AWS WAF 防止常見的入侵。 AWS AppSync 而 Amazon API Gateway 通常會使用 Amazon DynamoDB 資料表做為其 APIs的資料來源。中的管道 AWS CodePipeline 通常會使用 Amazon S3 做為其來源，並 AWS CodeBuild 用於其建置階段。在這些情況下，建立處理兩個或多個互連 L2 建構的單一 L3 建構通常很有用。 L2

以下是 L3 建構範例，其會佈建 CloudFront 分佈及其 S3 原始伺服器、 AWS WAF 要放在其前面的 、Amazon Route 53 記錄和 AWS Certificate Manager (ACM) 憑證，以新增具有傳輸中加密的自訂端點，全都在一個可重複使用的建構中：

```
// Define the properties passed to the L3 construct
export interface CloudFrontWebsiteProps {
     distributionProps: DistributionProps
     bucketProps: BucketProps
     wafProps: CfnWebAclProps
     zone: IHostedZone
}

// Define the L3 construct
export class CloudFrontWebsite extends Construct {
     public distribution: Distribution

     constructor(
        scope: Construct,
        id: string,
        props: CloudFrontWebsiteProps
     ) {
         super(scope, id);

         const certificate = new Certificate(this, "Certificate", {
            domainName: props.zone.zoneName,
            validation: CertificateValidation.fromDns(props.zone)
         });
         const defaultBehavior = {
            origin: new S3Origin(new Bucket(this, "bucket", props.bucketProps))
         }
         const waf = new CfnWebACL(this, "waf", props.wafProps);
         this.distribution = new Distribution(this, id, {
            ...props.distributionProps,
            defaultBehavior,
            certificate,
            domainNames: [this.domainName],
            webAclId: waf.attrArn,
          });
     }
}
```

請注意，CloudFront、Amazon S3、Route 53 和 ACM 都使用 L2 建構，但 Web ACL （定義處理 Web 請求的規則） 使用 L1 建構。這是因為 AWS CDK 是不斷發展的開放原始碼套件，尚未完全完成，而且`WebAcl`尚無 L2 建構。不過，任何人都可以 AWS CDK 透過建立新的 L2 建構來為 做出貢獻。因此，在 為 AWS CDK 提供 L2 建構之前`WebAcl`，您必須使用 L1 建構。若要使用 L3 建構 建立新的網站`CloudFrontWebsite`，請使用下列程式碼：

```
const siteADotCom = new CloudFrontWebsite(stack, "siteA", siteAProps);
const siteBDotCom = new CloudFrontWebsite(stack, "siteB", siteBProps);
const siteCDotCom = new CloudFrontWebsite(stack, "siteC", siteCProps);
```

在此範例中，CloudFront `Distribution` L2 建構會公開為 L3 建構的公有屬性。在某些情況下，您仍然需要公開這類 L3 屬性。事實上，我們將在稍後的自訂資源區段中`Distribution`再次看到 。

AWS CDK 包含一些資源互動模式的範例，例如此模式。除了包含 Amazon Elastic Container Service (Amazon ECS) L2 建構的`aws-ecs`套件之外， AWS CDK 還具有名為 aws-ecs-patterns 的套件。此套件包含數個 L3 建構模組，結合 Amazon ECS 與 Application Load Balancer、Network Load Balancer 和目標群組，同時提供 Amazon Elastic Compute Cloud (Amazon EC2) 和 的預設不同版本 AWS Fargate。由於許多無伺服器應用程式只搭配 Fargate 使用 Amazon ECS，因此這些 L3 建構提供便利性，可節省開發人員和客戶的時間。

資源擴充功能

有些使用案例需要資源具有非 L2 建構的原生特定預設設定。在堆疊層級，這可以透過使用 層面來處理，但另一個提供 L2 建構新預設值的便利方法是延伸第 2 層。由於建構是繼承 類別的任何`Construct`類別，而 L2 建構延伸該類別，您也可以直接延伸 L2 建構來建立 L2L3 建構。

這對於支援客戶單一需求的自訂商業邏輯特別有用。假設公司有一個儲存庫，將所有 AWS Lambda 函數程式碼存放在名為 的單一目錄中，`src/lambda`而且大多數 Lambda 函數每次都會重複使用相同的執行時間和處理常式名稱。您可以建立新的 L3 建構，而不是在每次設定新的 Lambda 函數時設定程式碼路徑：

```
export class MyCompanyLambdaFunction extends Function {
      constructor(
        scope: Construct,
        id: string,
        props: Partial<FunctionProps> = {}
     ) {
         super(scope, id, {
            handler: 'index.handler',
            runtime: Runtime.NODEJS_LATEST,
            code: Code.fromAsset(`src/lambda/${props.functionName || id}`),
            ...props
     });
}
```

然後，您可以在儲存庫中的任何位置取代 L2 `Function` 建構，如下所示：

```
new MyCompanyLambdaFunction(this, "MyFunction");
new MyCompanyLambdaFunction(this, "MyOtherFunction");
new MyCompanyLambdaFunction(this, "MyThirdFunction", {
   runtime: Runtime.PYTHON_3_11
});
```

預設值可讓您在單行上建立新的 Lambda 函數，並設定 L3 建構，因此您仍然可以視需要覆寫預設屬性。

當您只想將新的預設值新增至現有的 L2 建構時，擴充 L2 建構會直接運作最佳。如果您也需要其他自訂邏輯，最好擴展 Construct類別。原因來自 super方法，在建構函數中稱為 。在擴展其他類別的類別中， super方法用於呼叫父類別的建構函式，這必須是建構函式中發生的第一件事。這表示只有在建立原始 L2 建構模組之後，才能處理傳遞的引數或其他自訂邏輯。如果您需要在執行個體化 L2 建構之前執行任何此自訂邏輯，最好遵循先前在資源互動區段中概述的模式。

自訂資源

自訂資源是 CloudFormation 中的強大功能，可讓您從在堆疊部署期間啟用的 Lambda 函數執行自訂邏輯。每當您在部署期間需要 CloudFormation 未直接支援的任何程序時，您可以使用自訂資源來實現。AWS CDK 提供的類別也可讓您以程式設計方式建立自訂資源。透過在 L3 建構函數中使用自訂資源，您可以讓建構幾乎沒有任何效果。

使用 Amazon CloudFront 的優點之一是其強大的全域快取功能。如果您想要手動重設該快取，讓您的網站立即反映對原始伺服器所做的新變更，您可以使用 CloudFront 無效。不過，失效是在 CloudFront 分佈上執行的程序，而不是 CloudFront 分佈的屬性。它們可以隨時建立並套用到現有的 分佈，因此它們本質上不是佈建和部署程序的一部分。

在此案例中，您可能想要在每次更新分佈的原始伺服器後建立並執行失效。由於自訂資源，您可以建立看起來像這樣的 L3 建構：

```
export interface CloudFrontInvalidationProps {
     distribution: Distribution
     region?: string
     paths?: string[]
}

export class CloudFrontInvalidation extends Construct {
      constructor(
        scope: Construct,
        id: string,
```

```
        props: CloudFrontInvalidationProps
    ) {
        super(scope, id);
        const policy = AwsCustomResourcePolicy.fromSdkCalls({
            resources:AwsCustomResourcePolicy.ANY_RESOURCE
        });
        new AwsCustomResource(scope, `${id}Invalidation`, {
           policy,
           onUpdate: {
                service: 'CloudFront',
                action: 'createInvalidation',
                region: props.region || 'us-east-1',
                physicalResourceId:
 PhysicalResourceId.fromResponse('Invalidation.Id'),
                parameters: {
                   DistributionId: props.distribution.distributionId,
                   InvalidationBatch: {
                       Paths: {
                          Quantity: props.paths?.length || 1,
                          Items: props.paths || ['/*']
                       },
                       CallerReference: crypto.randomBytes(5).toString('hex')
                   }
                }
           }
        }
    }
}
```

使用先前在 `CloudFrontWebsite` L3 建構中建立的分佈，您可以非常輕鬆地執行此操作：

```
new CloudFrontInvalidation(this, 'MyInvalidation', {
       distribution: siteADotCom.distribution
});
```

此 L3 建構模組使用稱為 AwsCustomResource 的 an AWS CDK L3 建構模組來建立執行自訂邏輯的自訂資源。當您需要只進行一次 AWS 開發套件呼叫時， `AwsCustomResource` 非常方便，因為它可讓您執行此操作，而無需撰寫任何 Lambda 程式碼。如果您有更複雜的需求，並想要實作自己的邏輯，您可以直接使用基本的 CustomResource 類別。

AWS CDK 另一個使用自訂資源 L3 建構的良好範例是 S3 儲存貯體部署。此 L3 建構的建構函數中的自訂資源所建立的 Lambda 函數會新增 CloudFormation 無法以其他方式處理的功能：它會新增和更新

S3 儲存貯體中的物件。如果沒有 S3 儲存貯體部署，您將無法將內容放入剛在堆疊中建立的 S3 儲存貯體，這會非常不方便。

AWS CDK 消除需要寫入 CloudFormation 語法範圍的最佳範例是此基本 `S3BucketDeployment`：

```
new BucketDeployment(this, 'BucketObjects', {
  sources: [Source.asset('./path/to/amzn-s3-demo-bucket')],
  destinationBucket: amzn-s3-demo-bucket
});
```

將 與您必須寫入才能完成相同操作的 CloudFormation 程式碼進行比較：

```
"lambdapolicyA5E98E09": {
   "Type": "AWS::IAM::Policy",
   "Properties": {
    "PolicyDocument": {
     "Statement": [
      {
       "Action": "lambda:UpdateFunctionCode",
       "Effect": "Allow",
       "Resource": "arn:aws:lambda:us-east-1:123456789012:function:my-function"
      }
     ],
     "Version": "2012-10-17"
    },
    "PolicyName": "lambdaPolicy",
    "Roles": [
     {
      "Ref": "myiamroleF09C7974"
     }
    ]
   },
   "Metadata": {
    "aws:cdk:path": "CdkScratchStack/lambda-policy/Resource"
   }
  },
  "BucketObjectsAwsCliLayer8C081206": {
   "Type": "AWS::Lambda::LayerVersion",
   "Properties": {
    "Content": {
     "S3Bucket": {
      "Fn::Sub": "cdk-hnb659fds-assets-${AWS::AccountId}-${AWS::Region}"
     },
```

```
      "S3Key": "e2277687077a2abf9ae1af1cc9565e6715e2ebb62f79ec53aa75a1af9298f642.zip"
     },
     "Description": "/opt/awscli/aws"
    },
    "Metadata": {
     "aws:cdk:path": "CdkScratchStack/BucketObjects/AwsCliLayer/Resource",
     "aws:asset:path":
 "asset.e2277687077a2abf9ae1af1cc9565e6715e2ebb62f79ec53aa75a1af9298f642.zip",
     "aws:asset:is-bundled": false,
     "aws:asset:property": "Content"
    }
   },
   "BucketObjectsCustomResourceB12E6837": {
    "Type": "Custom::CDKBucketDeployment",
    "Properties": {
     "ServiceToken": {
      "Fn::GetAtt": [
       "CustomCDKBucketDeployment8693BB64968944B69AAFB0CC9EB8756C81C01536",
       "Arn"
      ]
     },
     "SourceBucketNames": [
      {
       "Fn::Sub": "cdk-hnb659fds-assets-${AWS::AccountId}-${AWS::Region}"
      }
     ],
     "SourceObjectKeys": [
      "f888a9d977f0b5bdbc04a1f8f07520ede6e00d4051b9a6a250860a1700924f26.zip"
     ],
     "DestinationBucketName": {
      "Ref": "amzn-s3-demo-bucket77F80CC0"
     },
     "Prune": true
    },
    "UpdateReplacePolicy": "Delete",
    "DeletionPolicy": "Delete",
    "Metadata": {
     "aws:cdk:path": "CdkScratchStack/BucketObjects/CustomResource/Default"
    }
   },
   "CustomCDKBucketDeployment8693BB64968944B69AAFB0CC9EB8756CServiceRole89A01265": {
    "Type": "AWS::IAM::Role",
    "Properties": {
     "AssumeRolePolicyDocument": {
```

```
     "Statement": [
      {
       "Action": "sts:AssumeRole",
       "Effect": "Allow",
       "Principal": {
        "Service": "lambda.amazonaws.com"
       }
      }
     ],
     "Version": "2012-10-17"
    },
    "ManagedPolicyArns": [
     {
      "Fn::Join": [
       "",
       [
        "arn:",
        {
         "Ref": "AWS::Partition"
        },
        ":iam::aws:policy/service-role/AWSLambdaBasicExecutionRole"
       ]
      ]
     }
    ]
   },
   "Metadata": {
    "aws:cdk:path": "CdkScratchStack/
Custom::CDKBucketDeployment8693BB64968944B69AAFB0CC9EB8756C/ServiceRole/Resource"
   }
  },

 "CustomCDKBucketDeployment8693BB64968944B69AAFB0CC9EB8756CServiceRoleDefaultPolicy88902FDF":
 {
   "Type": "AWS::IAM::Policy",
   "Properties": {
    "PolicyDocument": {
     "Statement": [
      {
       "Action": [
        "s3:GetBucket*",
        "s3:GetObject*",
        "s3:List*"
       ],
```

```
        "Effect": "Allow",
        "Resource": [
         {
          "Fn::Join": [
           "",
           [
            "arn:",
            {
             "Ref": "AWS::Partition"
            },
            ":s3:::",
            {
             "Fn::Sub": "cdk-hnb659fds-assets-${AWS::AccountId}-${AWS::Region}"
            },
            "/*"
           ]
          ]
         },
         {
          "Fn::Join": [
           "",
           [
            "arn:",
            {
             "Ref": "AWS::Partition"
            },
            ":s3:::",
            {
             "Fn::Sub": "cdk-hnb659fds-assets-${AWS::AccountId}-${AWS::Region}"
            }
           ]
          ]
         }
        ]
       },
       {
        "Action": [
         "s3:Abort*",
         "s3:DeleteObject*",
         "s3:GetBucket*",
         "s3:GetObject*",
         "s3:List*",
         "s3:PutObject",
         "s3:PutObjectLegalHold",
```

```
        "s3:PutObjectRetention",
        "s3:PutObjectTagging",
        "s3:PutObjectVersionTagging"
       ],
       "Effect": "Allow",
       "Resource": [
        {
         "Fn::GetAtt": [
          "amzns3demobucket77F80CC0",
          "Arn"
         ]
        },
        {
         "Fn::Join": [
          "",
          [
           {
            "Fn::GetAtt": [
             "amzns3demobucket77F80CC0",
             "Arn"
            ]
           },
           "/*"
          ]
         ]
        }
       ]
      }
     ],
     "Version": "2012-10-17"
    },
    "PolicyName":
 "CustomCDKBucketDeployment8693BB64968944B69AAFB0CC9EB8756CServiceRoleDefaultPolicy88902FDF",
    "Roles": [
     {
      "Ref":
 "CustomCDKBucketDeployment8693BB64968944B69AAFB0CC9EB8756CServiceRole89A01265"
     }
    ]
   },
   "Metadata": {
    "aws:cdk:path": "CdkScratchStack/
Custom::CDKBucketDeployment8693BB64968944B69AAFB0CC9EB8756C/ServiceRole/DefaultPolicy/
Resource"
```

```
    }
   },
   "CustomCDKBucketDeployment8693BB64968944B69AAFB0CC9EB8756C81C01536": {
    "Type": "AWS::Lambda::Function",
    "Properties": {
     "Code": {
      "S3Bucket": {
       "Fn::Sub": "cdk-hnb659fds-assets-${AWS::AccountId}-${AWS::Region}"
      },
      "S3Key": "9eb41a5505d37607ac419321497a4f8c21cf0ee1f9b4a6b29aa04301aea5c7fd.zip"
     },
     "Role": {
      "Fn::GetAtt": [
       "CustomCDKBucketDeployment8693BB64968944B69AAFB0CC9EB8756CServiceRole89A01265",
       "Arn"
      ]
     },
     "Environment": {
      "Variables": {
       "AWS_CA_BUNDLE": "/etc/pki/ca-trust/extracted/pem/tls-ca-bundle.pem"
      }
     },
     "Handler": "index.handler",
     "Layers": [
      {
       "Ref": "BucketObjectsAwsCliLayer8C081206"
      }
     ],
     "Runtime": "python3.9",
     "Timeout": 900
    },
    "DependsOn": [

  "CustomCDKBucketDeployment8693BB64968944B69AAFB0CC9EB8756CServiceRoleDefaultPolicy88902FDF",
     "CustomCDKBucketDeployment8693BB64968944B69AAFB0CC9EB8756CServiceRole89A01265"
    ],
    "Metadata": {
     "aws:cdk:path": "CdkScratchStack/
Custom::CDKBucketDeployment8693BB64968944B69AAFB0CC9EB8756C/Resource",
     "aws:asset:path":
  "asset.9eb41a5505d37607ac419321497a4f8c21cf0ee1f9b4a6b29aa04301aea5c7fd",
     "aws:asset:is-bundled": false,
     "aws:asset:property": "Code"
    }
```

```
}
```

4 行與 241 行是很大的差異！這只是您利用第 3 層自訂堆疊時可能的範例之一。

最佳實務

L1 建構

- 您不一定會直接避免使用 L1 建構，但您應該盡可能避免。如果特定 L2 建構不支援您的邊緣案例，您可以探索這兩個選項，而不是直接使用 L1 建構：
  - 存取 `defaultChild`： 如果 L2 建構模組中無法使用您需要的 CloudFormation 屬性，您可以使用 存取基礎 L1 建構模組`L2Construct.node.defaultChild`。您可以透過此屬性存取 L1 建構體的任何公有屬性，而不是自行建立 L1 建構體的麻煩。
  - 使用屬性覆寫：如果您要更新的屬性不是公開的？ 允許 AWS CDK 執行 CloudFormation 範本可執行的任何操作的最終逃生艙是使用每個 L1 建構中可用的方法：addPropertyOverride。您可以在 CloudFormation 範本層級操作堆疊，方法是直接將 CloudFormation 屬性名稱和值傳遞至此方法。

L2 建構

- 請記得利用 L2 建構常提供的協助程式方法。透過第 2 層，您不需要在執行個體化時傳遞每個屬性。L2 協助程式方法可以讓資源佈建更方便，特別是需要條件式邏輯時。最方便的協助程式方法之一衍生自授予類別。此類別不會直接使用，但許多 L2 建構會使用它來提供協助程式方法，讓許可更容易實作。例如，如果您想要授予 L2 Lambda 函數存取 L2 S3 儲存貯體的許可，您可以呼叫 ，`s3Bucket.grantReadWrite(lambdaFunction)`而不是建立新的角色和政策。

L3 建構

- 雖然當您想要讓堆疊更可重複使用且可自訂時，L3 建構非常方便，但建議您謹慎使用。考慮您需要哪種類型的 L3 建構，或是否需要 L3 建構：
  - 如果您不直接與 AWS 資源互動，通常更適合建立協助程式類別，而不是擴展`Construct`類別。這是因為 `Construct`類別預設會執行許多動作，只有在您直接與 AWS 資源互動時才需要這些動作。因此，如果您不需要執行這些動作，避免這些動作會更有效率。
  - 如果您判斷建立新的 L3 建構是適當的，在大多數情況下，您會想要直接擴展 `Construct`類別。只有當您想要更新建構的預設屬性時，才擴展其他 L2 建構。如果涉及其他 L2 建構模組或自訂邏輯，請`Construct`直接擴展並執行個體化建構模組中的所有資源。

常見問答集

如果 AWS CDK 不了解圖層，我無法使用 嗎？

您絕對可以。但是，與最強大的工具一樣， AWS CDK 變得越來越強大。了解 AWS CDK圖層互動如何釋放新的理解層級，這有助於簡化堆疊部署，遠遠超過您只需要基本 AWS CDK 知識就能完成的操作。

我可以使用與從 L2 建立 L3 建構相同的方式，從 L1 建立 L2 建構嗎？

如果資源已有 L2 建構，建議您使用該建構，並在第 3 層中進行自訂。這是因為許多研究已經開始找出為特定資源設定現有 L2 建構的最佳方法。不過，有數個 L1 建構的 L2 建構尚不存在。在這些情況下，我們鼓勵您建立自己的 L2 建構，並透過成為開放原始碼程式庫的 AWS CDK 貢獻者來與他人共用。您可以在 的貢獻準則中找到開始使用所需的一切 AWS CDK。

哪些 AWS 資源還沒有官方的 L2 建構？

沒有 L2 建構 AWS 的 資源數量依日減少，但如果您有興趣協助為其中一個資源建立 L2 建構，請造訪 AWS CDK API 參考。查看左側窗格中的資源清單。名稱旁具有上標 1 的資源沒有官方 L2 建構。

我可以使用 AWS CDK 支援的任何語言進行 L2 或 L3 建構嗎？

AWS CDK 支援多種程式設計語言，包括 TypeScript、JavaScript、Python、Java、C# 和 Go。您可以使用編譯為相關語言的 AWS CDK 程式碼來建立個人 L3 建構。不過，如果您想要對 做出貢獻 AWS CDK 或建立原生 AWS CDK 建構，則必須使用 TypeScript。這是因為 TypeScript 是唯一原生於 的語言 AWS CDK。其他語言的 AWS CDK 版本是使用名為 JSii 的 AWS 程式庫，從原生 TypeScript 程式碼建置而成。

哪裡可以找到 外部的現有 L3 建構 AWS CDK？

這裡有許多位置可以共用，但您可以在 AWS 解決方案建構網站和 Construct Hub 的 AWS CDK 區段中找到許多最熱門的建構。

資源

- AWS CDK API 參考
- AWS CloudFormation 資源規格
- AWS CDK 建構文件
- AWS CDK 抽象和逃生艙
- 利用 L2 建構來降低 AWS CDK 應用程式的複雜性 (AWS 部落格文章)
- AWS CloudFormation 自訂資源
- AWS 解決方案建構
- 建構中樞
- AWS CDK 範例 (GitHub 儲存庫)

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	—	2023 年 12 月 4 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：將內部部署 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，但只有來源資料庫會在資料複寫至目標資料庫時處理來自連線應用程式的交易。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

永久刪除資料集中個人資訊的程序。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

經常用於經常性問題的解決方案，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

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、低延遲網路連線能力。

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)

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的工具。 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，例如在網際網路上編製資訊索引的 Web 爬蟲程式。某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，讓使用者快速取得他們通常無權存取 AWS 帳戶 之 的存取權。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 在部落格文章 The Journey Toward Cloud-First 和企業策略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)

AI 欄位???，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。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

在整個資料生命週期中追蹤資料的來源和歷史記錄的程序，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 成員帳戶來管理組織的帳戶，並管理該服務的許可。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。

deployment

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，用於識別並優先考慮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。DVSM 擴展了最初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 上工作負載的災難復原 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

將人類可讀取的純文字資料轉換為加密文字的運算程序。

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 () 文件中的信封加密。 AWS Key Management Service AWS KMS

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

一種使用頻繁且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界限會限制故障的影響，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。少量的提示對於需要特定格式、推理或網域知識的任務來說非常有效。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，已在廣義和未標記資料的大量資料集上進行訓練。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 CSPM、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，以及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

請參閱基礎設施即程式碼。

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、特洛伊木馬程式、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。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，並請參閱調動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 中的操作整備審查 (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)

當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。

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

以預留位置值取代資料集中個人識別符的程序。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 Rs。

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 Rs。

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 Rs。

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 Rs。

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

定義所有涉及遷移活動和雲端操作之各方的角色和責任的矩陣。矩陣名稱衍生自矩陣中定義的責任類型：負責人 (R)、責任 (A)、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 管理主控台 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

由接收資料的 AWS 服務 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。

單一故障點 (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

做為中繼資料的鍵/值對，用於組織您的 AWS 資源。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，這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

危害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，通常可接受中等速度的查詢。

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，多次讀取。

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，並防止刪除或修改資料。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缺陷或漏洞。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。