aws

用户指南

AWS IoT TwinMaker

AWS IoT TwinMaker: 用户指南

Table of Contents

什么是 AWS IoT TwinMaker？

AWS IoT TwinMaker 是一项可用于构建物理和数字系统的可操作数字双胞胎的 AWS IoT 服务。 AWS IoT TwinMaker 使用来自各种现实世界传感器、摄像头和企业应用程序的测量和分析来创建数字可视化效果，以帮助您跟踪实际工厂、建筑物或工业厂房。您可以使用此真实数据监控运营、诊断和纠正错误以及优化运营。

数字孪生是系统及其所有物理和数字组件的实时数字表示。它根据数据动态更新，以模仿系统的真实结构、状态和行为。您可将其用于推动实现业务成果。

终端用户通过用户界面应用，与您的数字孪生数据交互。

工作方式

若要满足最基本的数字孪生创建要求，您必须执行以下操作。

- 对物理位置中的装置、设备、空间和流程进行建模。
- 将此模型与存储重要上下文信息（如传感器数据摄像头馈送）的数据来源连接。
- 创建可视化效果，帮助用户理解数据和见解，从而更有效地做出业务决策。
- 向终端用户提供数字孪生，推动其实现业务成果。

AWS IoT TwinMaker 通过提供以下功能来解决这些难题。

- 实体组件系统知识图谱： AWS IoT TwinMaker 提供用于在知识图中对设备、设备、空间和流程进行建模的工具。

  此知识图包含有关系统的元数据，可以连接到不同位置的数据。 AWS IoT TwinMaker 为存储在 AWS IoT SiteWise 和 Kinesis Video Streams 中的数据提供了内置连接器。您也可以为存储在其他位置的数据创建自定义连接器。

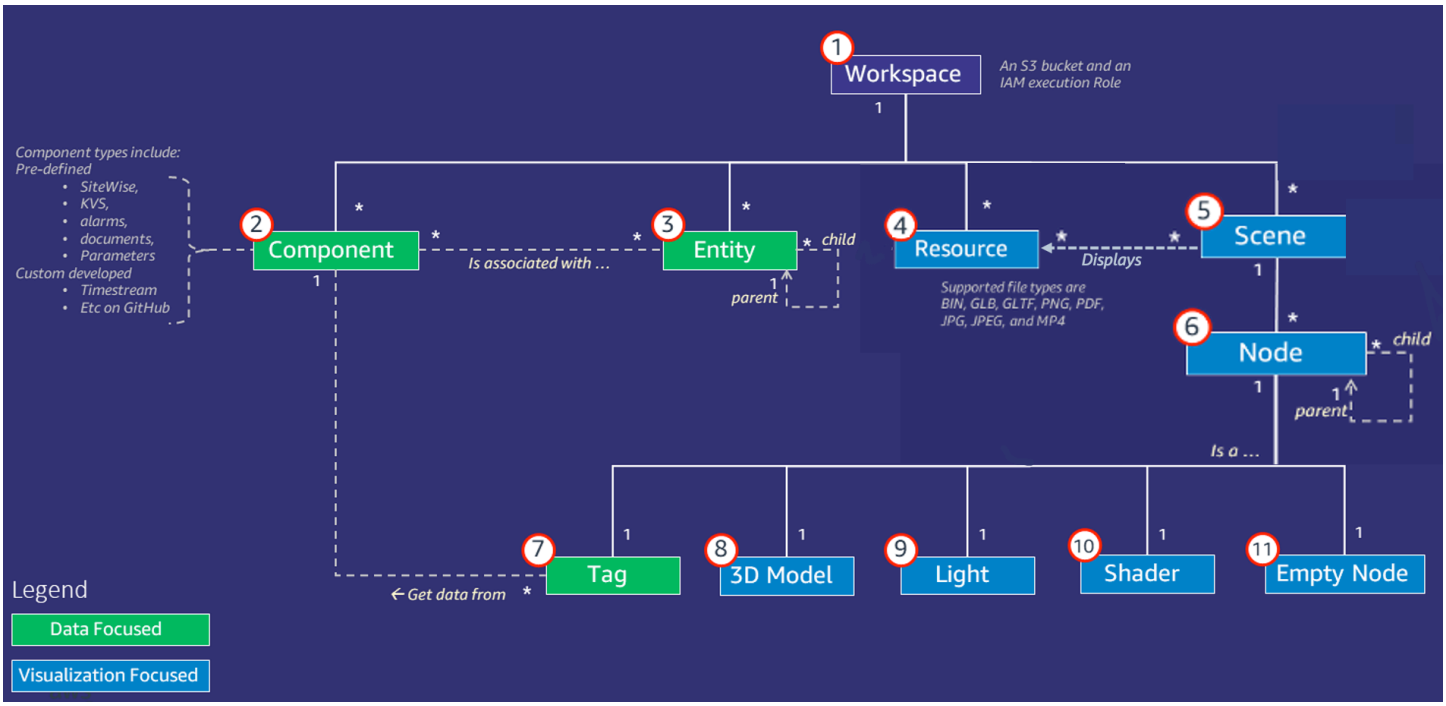
  知识图和连接器结合，提供了用于查询不同位置数据的界面。
- 场景编辑器： AWS IoT TwinMaker 控制台提供了用于创建 3D 场景的场景合成工具。您可以上传之前构建的 3D/CAD 模型，针对网页显示进行优化，并转换为 .gltf 或 .glb 格式。然后，您可以使用场景编辑器将多个模型置于单个场景中，从而以可视化方式呈现其操作。

  您也可以在场景内叠加数据。例如，您可以在场景位置创建标签，将其连接至传感器的温度数据。这将数据与位置相关联。

- 应用程序：为 Grafana 和 Amazon Managed Grafana AWS IoT TwinMaker 提供插件，您可以使用该插件为最终用户构建控制面板应用程序。
- 第三方工具：Mendix 与 Mendix 合作 AWS IoT TwinMaker ，为工业物联网提供完整的解决方案。参阅 Mendix 精益日常管理应用程序研讨会，并 AWS IoT TwinMaker开始使用 Mendi x 低代码应用程序开发平台 (LCAP) 和 Kinesis Video Streams 等 AWS AWS IoT TwinMaker服务。 AWS IoT SiteWise

关键概念与组件

下图说明了关键概念是如何 AWS IoT TwinMaker 组合在一起的。



> **Note**
>
> 图中的星号 (*) 表示 one-to-many 关系。有关每种关系的配额，请参见 AWS IoT TwinMaker 端点和配额。

下述为图中阐述的概念。

工作区

工作区是数字孪生应用程序的顶级容器。您可以在此工作区内为数字孪生创建一组逻辑实体、组件、场景资源和其他资源。它还充当安全边界，用于管理对数字孪生应用程序及其所含资源的访问权限。每个工作区都与存储工作区数据的 Amazon S3 桶关联。您可以通过 IAM 角色限制对工作区的访问。

一个工作区可以包含多个组件、实体、场景与资源。组件类型、实体、场景或资源仅存在于一个工作区内。

实体组件模型

AWS IoT TwinMaker 提供了使用 entity-component-based 知识图谱对系统进行建模的工具。您可以使用实体组件架构表示物理系统。此实体组件模型由实实体、组件和关系组成。更多关于实体组件系统的信息，请参见实体组件系统。

实体

实体采集数字元素功能，是以数字形式表示的数字孪生元素 此元素可以是物理设备、概念或者过程。实体包含与之关联的组件。这些组件为关联实体提供数据与上下文。

使用 AWS IoT TwinMaker，您可以将实体组织到自定义层次结构中，以实现更高效的管理。实体和组件系统默认为分层结构。

组件

组件为场景内的实体提供上下文和数据。您可向实体添加组件。组件的使用周期与实体的使用周期息息相关。

组件可添加静态数据，例如文档列表或地理位置坐标。它们还可以具有连接到其他系统的功能，包括包含时间序列数据的系统，例如 AWS IoT SiteWise 其他时间序列云历史学家。

组件由 JSON 文档定义，这些文档描述了数据来源与 AWS IoT TwinMaker之间的连接。组件可以描述外部数据源或内置的数据源 AWS IoT TwinMaker。组件通过使用 JSON 文档中指定的 Lambda 函数访问外部数据来源。工作区可能包含很多组件。组件通过关联实体向标签提供数据。

AWS IoT TwinMaker 提供了几个内置组件，您可以从控制台添加这些组件。您也可创建自定义组件，以连接时间流遥测和地理空间坐标等数据来源。例如， TimeStream遥测、地理空间组件以及指向 Snowflake 等第三方数据源的连接器。

AWS IoT TwinMaker 为常见用例提供了以下类型的内置组件：

- 文档，例如指定 URL 的用户手册或图片。
- 时间序列，例如来自 AWS IoT SiteWise的传感器数据。
- 警报，如来自外部数据来源的时间序列警报。
- 视频，来自连接至 Kinesis Video Streams 的 IP 摄像机。
- 自定义组件用于连接至其他数据来源。例如，您可以创建自定义连接器，将您的 AWS IoT TwinMaker 实体与外部存储的时间序列数据连接起来。

数据来源

数据源是数字双胞胎源数据的位置。 AWS IoT TwinMaker 支持两种类型的数据源：

- 层级结构连接器，允许您将外部模型持续同步至 AWS IoT TwinMaker。
- 时间序列连接器，允许您连接至时间序列数据库，例如 AWS IoT SiteWise

属性

属性是组件中包含的值，包括静态值与时间序列值。向实体添加组件时，组件中的属性描述了有关当前实体状态的详细信息。

AWS IoT TwinMaker 支持三种属性：

- 单值、 non-time-series属性 — 这些属性通常是静态键值对，直接存储在 AWS IoT TwinMaker 关联实体的元数据中。
- 时间序列属性 — AWS IoT TwinMaker 存储对这些属性的时间序列存储的引用。默认为最新值。
- 关系属性 - 这些属性存储其他实体或组件的引用。例如，`seen_by` 是一个关系组件，它可能将摄像机实体与另一个由该摄像机直接可视的实体关联起来。

您可以使用统一的数据查询接口，跨异构数据来源查询属性值。

可视化

您可以使用 AWS IoT TwinMaker 增强数字双胞胎的三维表示，然后在 Grafana 中进行查看。若要创建场景，请使用现有的 CAD 或其他 3D 文件类型。然后，您可以使用数据叠加层为数字孪生添加相关数据。

场景数

场景是三维表示，可为连接的数据提供视觉背景 AWS IoT TwinMaker。场景的创建方式包括：对整个环境使用单独的 gltf（GL 传输格式）或 glb 3D 模型，或使用多个模型组合。场景还包括用于表示场景兴趣点的标签。

场景是可视化的顶级容器。场景由一个或多个节点组成。

工作区可包含多个场景。例如，工作区设施的每一层可包含一个场景。

资源

场景显示资源，这些资源在 AWS IoT TwinMaker 控制台中显示为节点。一个场景可能包含多种资源。

资源是用于创建场景的、基于图像和 glTF 的三维模型。资源可以表示一台设备，也可以表示一个完整的场地。

若要将资源放入场景内，您可是将 .gltf 或 .glb 文件上传至工作区资源库，然后将其添加至场景内。

增强用户界面

借 AWS IoT TwinMaker 助，您可以使用数据叠加来增强场景，从而向场景中的位置添加重要的上下文和信息（例如传感器数据）。

节点：节点是标签、灯光和三维模型实例。它们可以为空，以向场景层次添加结构。例如，您可以将多个节点分组至同一空节点。

标签：标签是一种表示组件（通过实体）数据的节点类型。一个标签只能与一个组件关联。标签是添加到特定场景 x,y,z 坐标位置的注释。标签通过实体属性将此场景部分连接至知识图。您可以使用标签来配置场景中项目的行为或视觉外观，如警报。

灯光：您可以向场景中添加灯光，以使某些对象聚焦，或者在对象上投射阴影以指示其物理位置。

三维模型：三维模型是以视觉形式表示导入为资源的 .gltf 或 .glb 文件。

> **Note**
>
> AWS IoT TwinMaker 不打算用于任何可能导致严重人身伤害或死亡或造成环境或财产损失的危险环境或关键系统的运行，或与之相关联。
> AWS IoT TwinMaker 应根据您的用例对通过使用收集的数据进行准确性评估。 AWS IoT TwinMaker 不应取代人为评估物理系统是否安全运行而对这些系统进行监测.

入门 AWS IoT TwinMaker

本节中的主题描述了如何执行以下操作。

- 创建并设置一个新工作区。
- 创建实体并向其添加组件。

先决条件：

要创建您的第一个工作空间和场景，您需要以下 AWS 资源。

- AWS 账户。
- 的 IAM 服务角色 AWS IoT TwinMaker。默认情况下，当您在AWS IoT TwinMaker 控制台中创建新 AWS IoT TwinMaker 工作区时，会自动生成此角色。

  如果您不选择允许 AWS IoT TwinMaker 自动创建新的 IAM 服务角色，则必须指定一个您已经创建的角色。

  有关如何创建和管理此服务角色的说明，请参阅 ???。

  有关 IAM 服务角色的更多信息，请参阅创建角色以向 AWS 服务委派权限。

  > **Important**
  >
  > 此服务角色必须附加策略，该策略允许该服务读取和写入 Amazon S3 存储桶。 AWS IoT TwinMaker 使用此角色代表您访问其他服务。您还需要在此角色和之间分配信任关系，AWS IoT TwinMaker 以便服务可以代入该角色。如果您的双胞胎与其他 AWS 服务互动，请同时为这些服务添加必要的权限。

主题

- 创建和管理 AWS IoT TwinMaker的服务角色
- 创建工作区
- 创建您的第一个实体
- 设置 AWS 账户

创建和管理 AWS IoT TwinMaker的服务角色

AWS IoT TwinMaker 要求您使用服务角色来允许该服务代表您访问其他服务中的资源。此角色必须与建立信任关系 AWS IoT TwinMaker。创建工作区时，必须将此角色分配给该工作区。此主题包含说明如何为常见场景配置权限的策略示例。

分配信任

以下策略在您的角色和之间建立了信任关系 AWS IoT TwinMaker。将此信任关系分配给用于工作区的角色。

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

Amazon S3 权限

以下策略允许角色对 Amazon S3 存储桶执行读取、删除和写入操作。工作区将资源存储在 Amazon S3 中，因此所有工作区都需要 Amazon S3 权限。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetBucket*",
        "s3:GetObject",
        "s3:ListBucket",
```

```
        "s3:PutObject"
      ],
      "Resource": [
        "arn:aws:s3:::*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:DeleteObject"
      ],
      "Resource": [
        "arn:aws:s3:::*/DO_NOT_DELETE_WORKSPACE_*"
      ]
    }
  ]
}
```

> **Note**
>
> 创建工作空间时，AWS IoT TwinMaker 会在您的 Amazon S3 存储桶中创建一个文件，表明工作空间正在使用该文件。此策略 AWS IoT TwinMaker 允许您在删除工作区时删除该文件。AWS IoT TwinMaker 放置与您的工作区相关的其他对象。删除工作区时，您有责任删除这些对象。

为特定 Amazon S3 存储桶分配权限

在 AWS IoT TwinMaker 控制台中创建工作空间时，您可以选择为您 AWS IoT TwinMaker 创建 Amazon S3 存储桶。您可以使用以下 AWS CLI 命令查找有关此存储桶的信息。

```
aws iottwinmaker get-workspace --workspace-id workspace name
```

下面的示例显示此命令的输出格式：

```
{
```

```
    "arn": "arn:aws:iottwinmaker:region:account Id:workspace/workspace name",
    "creationDateTime": "2021-11-30T11:30:00.000000-08:00",
    "description": "",
    "role": "arn:aws:iam::account Id:role/service role name",
    "s3Location": "arn:aws:s3:::bucket name",
    "updateDateTime": "2021-11-30T11:30:00.000000-08:00",
    "workspaceId": "workspace name"
}
```

要对策略进行更新，使其为特定 Amazon S3 存储桶分配权限，请使用*#####*的值。

以下策略允许角色对特定 Amazon S3 存储桶进行读取、删除和写入。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetBucket*",
        "s3:GetObject",
        "s3:ListBucket",
        "s3:PutObject"
      ],
      "Resource": [
        "arn:aws:s3:::bucket name",
        "arn:aws:s3:::bucket name/*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:DeleteObject"
      ],
      "Resource": [
        "arn:aws:s3:::iottwinmakerbucket/DO_NOT_DELETE_WORKSPACE_*"
      ]
    }
  ]
}
```

内置连接器的权限

如果您的工作空间使用内置连接器与其他 AWS 服务进行交互，则必须在此策略中包含这些服务的权限。如果使用 com.amazon.iotsitewise.connector 组件类型，则必须包括对 AWS IoT SiteWise的权限。有关组件类型的更多信息，请参阅 ???。

> **Note**
>
> 如果您使用自定义组件类型与其他 AWS 服务进行交互，则必须向该角色授予运行在您的组件类型中实现该函数的 Lambda 函数的权限。有关更多信息，请参阅 ???。

以下示例显示了如何在您的政策 AWS IoT SiteWise 中包含内容。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetBucket*",
        "s3:GetObject",
        "s3:ListBucket",
        "s3:PutObject"
      ],
      "Resource": [
        "arn:aws:s3:::bucket name",
        "arn:aws:s3:::bucket name/*"
      ]
    },
    {
        "Effect": "Allow",
        "Action": [
            "iotsitewise:DescribeAsset"
        ],
        "Resource": "asset ARN"
        },
    {
        "Effect": "Allow",
        "Action": [
            "iotsitewise:DescribeAssetModel"
```

```
        ],
        "Resource": "asset model ARN"
        },
    {
      "Effect": "Allow",
      "Action": [
        "s3:DeleteObject"
      ],
      "Resource": [
        "arn:aws:s3:::*/DO_NOT_DELETE_WORKSPACE_*"
      ]
    }
  ]
}
```

如果您使用 com.amazon.iotsitewise.connector 组件类型并且需要从中读取属性数据 AWS IoT SiteWise，则必须在策略中包含以下权限。

```
...
{
    "Action": [
        "iotsitewise:GetPropertyValueHistory",
    ],
    "Resource": [
        "AWS IoT SiteWise asset resource ARN"
    ],
    "Effect": "Allow"
},
...
```

如果您使用 com.amazon.iotsitewise.connector 组件类型并且需要向其写入属性数据 AWS IoT SiteWise，则必须在策略中包含以下权限。

```
...
{
    "Action": [
        "iotsitewise:BatchPutPropertyValues",
    ],
```

```
    "Resource": [
        "AWS IoT SiteWise asset resource ARN"
    ],
    "Effect": "Allow"
},
...
```

如果你使用 com.amazon.iotsitewise.connector.edgevideo 组件类型，则必须包括和 Kinesi s Video Streams 的权限。 AWS IoT SiteWise 以下示例策略说明如何在您的策略中包含 AWS IoT SiteWise 和 Kinesis Video Streams 权限。

```
...
{
    "Action": [
        "iotsitewise:DescribeAsset",
        "iotsitewise:GetAssetPropertyValue"
    ],
    "Resource": [
        "AWS IoT SiteWise asset resource ARN for the Edge Connector for Kinesis Video
 Streams"
    ],
    "Effect": "Allow"
},
{
    "Action": [
        "iotsitewise:DescribeAssetModel"
    ],
    "Resource": [
        "AWS IoT SiteWise model resource ARN for the Edge Connector for Kinesis Video
 Streams"
    ],
    "Effect": "Allow"
},
{
    "Action": [
        "kinesisvideo:DescribeStream"
    ],
    "Resource": [
        "Kinesis Video Streams stream ARN"
    ],
    "Effect": "Allow"
```

```
},
...
```

对外部数据来源连接器的权限

如果创建的组件类型使用连接到外部数据来源的函数，则必须向服务角色授予使用实现该函数的 Lambda 函数的权限。有关创建组件类型的更多信息，请参阅 ???。

以下示例向服务角色授予使用 Lambda 函数的权限。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetBucket*",
        "s3:GetObject",
        "s3:ListBucket",
        "s3:PutObject"
      ],
      "Resource": [
        "arn:aws:s3:::bucket name",
        "arn:aws:s3:::bucket name/*"
      ]
    },
    {
        "Action": [
            "lambda:invokeFunction"
        ],
        "Resource": [
            "Lambda function ARN"
        ],
        "Effect": "Allow"
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:DeleteObject"
      ],
      "Resource": [
```

```
        "arn:aws:s3:::*/DO_NOT_DELETE_WORKSPACE_*"
      ]
    }
  ]
}
```

有关使用 IAM 控制台、和 IAM API 创建角色以及为角色分配策略和信任关系的更多信息，请参阅创建向委派权限的角色 AWS 服务。 AWS CLI

修改工作区 IAM 角色以使用 Athena 数据连接器

要使用 AWS IoT TwinMaker Athena 表格数据连接器，您必须更新 AWS IoT TwinMaker 您的工作空间 IAM 角色。向工作区 IAM 角色添加以下权限：

> **Note**
>
> 此 IAM 更改仅适用于存储在 Amazon S3 中的 Athena 表格数据。 AWS Glue 要将 Athena 与其他数据来源一起使用，必须为 Athena 配置 IAM 角色，请参阅 Athena 中的身份和访问管理。

```
{
    "Effect": "Allow",
    "Action": [
        "athena:GetQueryExecution",
        "athena:GetQueryResults",
        "athena:GetTableMetadata",
        "athena:GetWorkGroup",
        "athena:StartQueryExecution",
        "athena:StopQueryExecution"
    ],
    "Resource": [
        "athena resouces arn"
    ]
},// Athena permission
{
    "Effect": "Allow",
    "Action": [
        "glue:GetTable",
        "glue:GetTables",
```

```
            "glue:GetDatabase",
            "glue:GetDatabases"
        ],
        "Resource": [
            "glue resouces arn"
        ]
},// This is an example for accessing aws glue
{
        "Effect": "Allow",
        "Action": [
            "s3:ListBucket",
            "s3:GetObject"
        ],
        "Resource": [
            "Amazon S3 data source bucket resources arn"
        ]
}, // S3 bucket for storing the tabular data.
{
        "Effect": "Allow",
        "Action": [
            "s3:GetBucketLocation",
            "s3:GetObject",
            "s3:ListBucket",
            "s3:ListBucketMultipartUploads",
            "s3:ListMultipartUploadParts",
            "s3:AbortMultipartUpload",
            "s3:CreateBucket",
            "s3:PutObject",
            "s3:PutBucketPublicAccessBlock"
        ],
        "Resource": [
            "S3 query result bucket resources arn"
        ]
} // Storing the query results
```

有关 Athena IAM 配置的更多信息，请阅读 Athena 中的身份和访问管理。

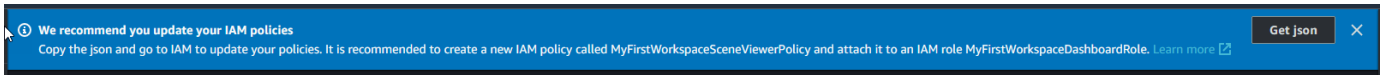
创建工作区

要创建和配置第一个工作区，请按以下步骤操作。

> **Note**
>
> 此主题介绍如何使用单个资源创建简单的工作区。要获得具有多种资源的功能齐全的工作空间，请尝试示例 Github 存储库中的AWS IoT TwinMaker 示例设置。

1. 在 AWS IoT TwinMaker 控制台主页上，选择左侧导航窗格中的工作区。
2. 在 Workspaces（工作区）页面中，选择 Create workspaces（创建工作区）。
3. 在 Create a Workspace（创建工作区）页面上，输入工作区的名称。
4. （可选）为工作区添加描述。
5. 在 S3 资源下，选择创建 S3 存储桶。此选项创建一个 Amazon S3 AWS IoT TwinMaker 存储桶，用于存储与工作空间相关的信息和资源。每个工作区都需要自己的存储桶。
6. 在执行角色下，选择自动生成新角色或为此工作区创建的自定义 IAM 角色。

   如果您选择自动生成新角色，则会向该角色 AWS IoT TwinMaker 附加一个策略，该策略向新服务角色授予访问其他 AWS 服务的权限，包括读取和写入您在上一步中指定的 Amazon S3 存储桶的权限。有关向该角色分配权限的更多信息，请参阅 ???。
7. 选择 Create Workspace（创建工作区）。以下横幅显示在工作区页面的顶部。



8. 选择获取 JSON。我们建议您将看到的 IAM 策略添加到为查看 Grafana AWS IoT TwinMaker 控制面板的用户和账户创建的 IAM 角色中。此角色的名称遵循以下模式：*workspace-name*DashboardRole，有关如何创建策略并将其附加到角色的说明，请参阅修改角色权限策略（控制台）。

   以下示例包含要添加到控制面板角色的策略。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject"
      ],
      "Resource": [
        "arn:aws:s3:::iottwinmaker-workspace-workspace-name-lower-case-account-id",
```

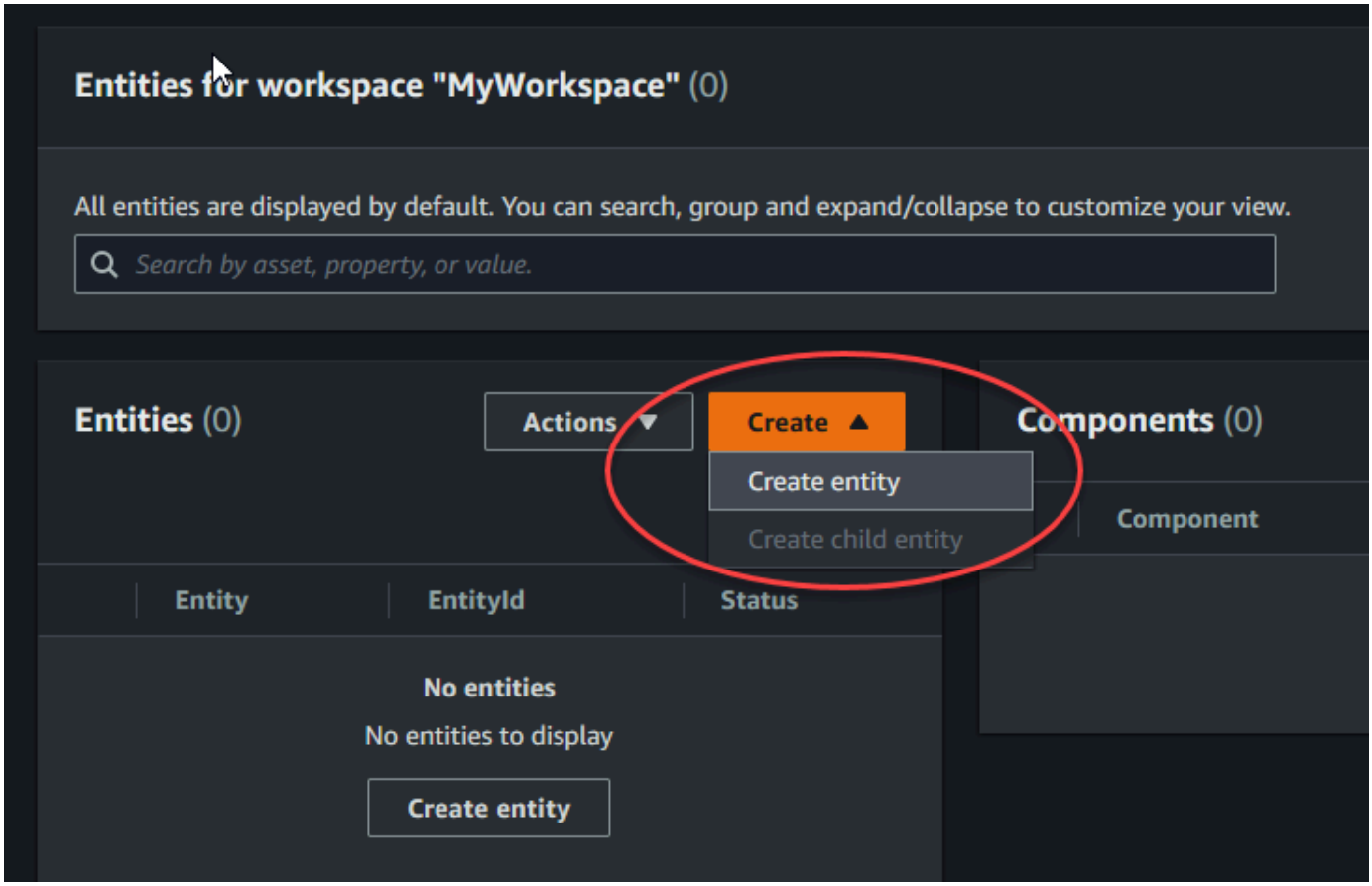
```
        "arn:aws:s3:::iottwinmaker-workspace-workspace-name-lower-case-account-id/
*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "iottwinmaker:Get*",
        "iottwinmaker:List*"
      ],
      "Resource": [
        "arn:aws:iottwinmaker:us-east-1:account-id:workspace/workspace-name",
        "arn:aws:iottwinmaker:us-east-1:account-id:workspace/workspace-name/*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": "iottwinmaker:ListWorkspaces",
      "Resource": "*"
    }
  ]
}
```

您现在可以开始使用第一个实体为工作区创建数据模型。有关如何执行此操作的说明，请参阅 创建您的第一个实体。

创建您的第一个实体

要创建第一个实体，请使用以下步骤。

1. 在工作区页面，选择您的工作区，然后在左侧窗格中选择实体。
2. 在实体页面，选择创建，然后选择创建实体。



3. 在创建实体窗口中，输入实体的名称。此示例使用 **`CookieMixer`** 实体。
4. （可选）为实体输入描述。
5. 选择 创建实体，

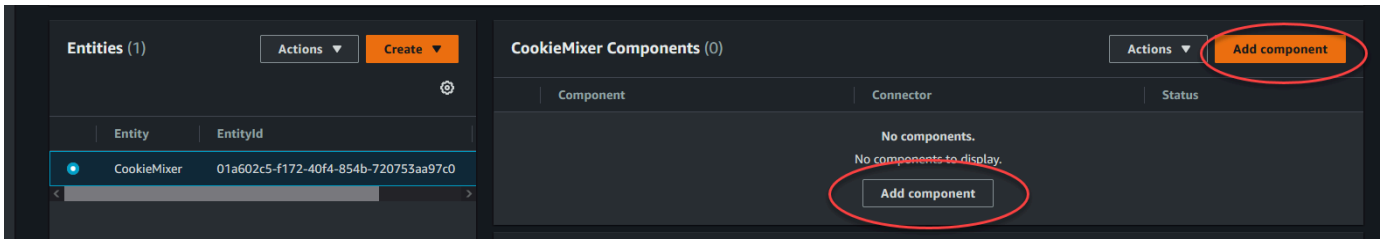
实体包含有关工作区中每个项目的数据。您可以通过添加组件将数据放入实体中。 AWS IoT TwinMaker 提供了以下内置组件类型。

- 参数：添加一组键值属性。
- 文档：为包含实体相关信息的文档添加名称和 URL。
- 警报：连接到警报时间序列数据来源。
- SiteWise 连接器：提取资产中定义的时间序列属性。 AWS IoT SiteWise
- Kinesis Video AWS IoT Greengrass Streams 的边缘连接器：从 KVS 的边缘连接器提取视频数据。AWS IoT Greengrass有关更多信息，请参阅 AWS IoT TwinMaker 视频集成。

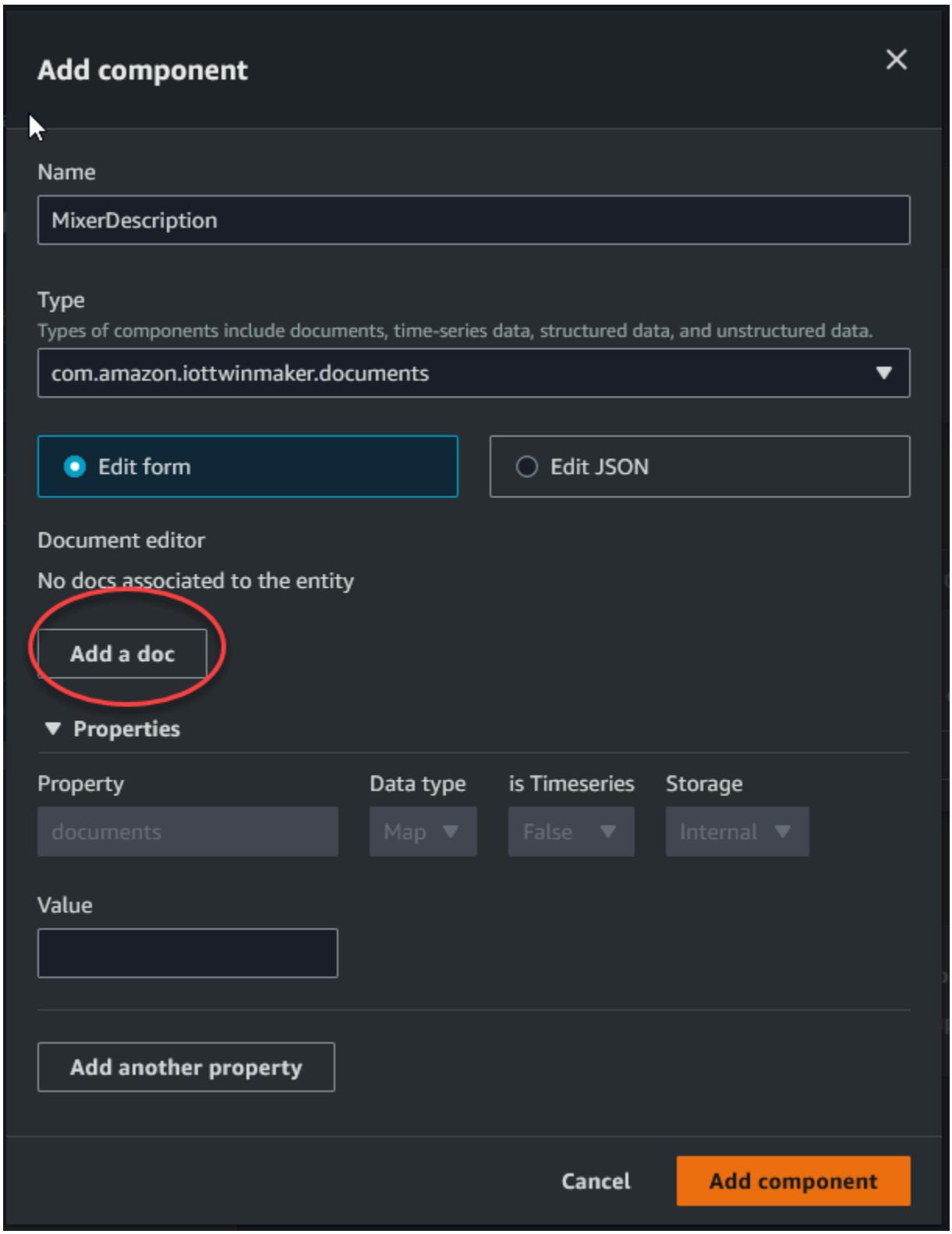
在左窗格中选择组件类型可查看这些组件类型及其定义。还可以在组件类型页面创建新的组件类型。有关创建组件类型的更多信息，请参阅 使用和创建组件类型。

在此示例中，我们创建了一个简单的文档组件，用于添加有关实体的描述性信息。

1. 在实体页面上，选择实体，然后选择添加组件。



2. 在添加组件窗口中，为组件输入一个名称。由于此示例使用饼干搅拌器实体，因此我们在名称字段输入 **`MixerDescription`**。



3. 选择 “添加文档”，然后输入文档名称和外部 URL 的值。使用文档组件，您可以存储包含实体重要信息的外部 URL 列表。
4. 选择 添加组件。

您现在可以创建第一个场景。有关如何执行此操作的说明，请参阅 创建和编辑 AWS IoT TwinMaker 场景。

设置 AWS 账户

如果您没有 AWS 账户，请完成以下步骤来创建一个。

要注册 AWS 账户

1. 打开 https://portal.aws.amazon.com/billing/signup。
2. 按照屏幕上的说明进行操作。

   在注册时，将接到一通电话，要求使用电话键盘输入一个验证码。

   当您注册时 AWS 账户，就会创建AWS 账户根用户一个。根用户有权访问该账户中的所有 AWS 服务 和资源。作为安全最佳实践，请为用户分配管理访问权限，并且只使用根用户来执行需要根用户访问权限的任务。

注册获取 AWS 账户

如果您没有 AWS 账户，请完成以下步骤来创建一个。

要注册 AWS 账户

1. 打开 https://portal.aws.amazon.com/billing/signup。
2. 按照屏幕上的说明进行操作。

   在注册时，将接到一通电话，要求使用电话键盘输入一个验证码。

   当您注册时 AWS 账户，就会创建AWS 账户根用户一个。根用户有权访问该账户中的所有 AWS 服务 和资源。作为安全最佳实践，请为用户分配管理访问权限，并且只使用根用户来执行需要根用户访问权限的任务。

AWS 注册过程完成后会向您发送一封确认电子邮件。在任何时候，您都可以通过转至 https://aws.amazon.com/ 并选择我的账户来查看当前的账户活动并管理您的账户。

创建具有管理访问权限的用户

注册后，请保护您的安全 AWS 账户 AWS 账户根用户 AWS IAM Identity Center，启用并创建管理用户，这样您就不会使用 root 用户执行日常任务。

保护你的 AWS 账户根用户

1. 选择 Root 用户并输入您的 AWS 账户 电子邮件地址，以账户所有者的身份登录。AWS Management Console在下一页上，输入您的密码。

   要获取使用根用户登录方面的帮助，请参阅《AWS 登录 用户指南》中的以根用户身份登录。
2. 为您的根用户启用多重身份验证 (MFA)。

   有关说明，请参阅 I A M 用户指南中的为 AWS 账户 根用户启用虚拟 MFA 设备（控制台）。

创建具有管理访问权限的用户

1. 启用 IAM Identity Center

   有关说明，请参阅《AWS IAM Identity Center 用户指南》中的启用 AWS IAM Identity Center。
2. 在 IAM Identity Center 中，为用户授予管理访问权限。

   有关使用 IAM Identity Center 目录 作为身份源的教程，请参阅《用户指南》 IAM Identity Center 目录中的使用默认设置配置AWS IAM Identity Center 用户访问权限。

以具有管理访问权限的用户身份登录

- 要使用您的 IAM Identity Center 用户身份登录，请使用您在创建 IAM Identity Center 用户时发送到您的电子邮件地址的登录网址。

  有关使用 IAM Identity Center 用户登录的帮助，请参阅AWS 登录 用户指南中的登录 AWS 访问门户。

将访问权限分配给其他用户

1. 在 IAM Identity Center 中，创建一个权限集，该权限集遵循应用最低权限的最佳做法。

   有关说明，请参阅《AWS IAM Identity Center 用户指南》中的创建权限集。
2. 将用户分配到一个组，然后为该组分配单点登录访问权限。

有关说明，请参阅《AWS IAM Identity Center 用户指南》中的添加组。

使用和创建组件类型

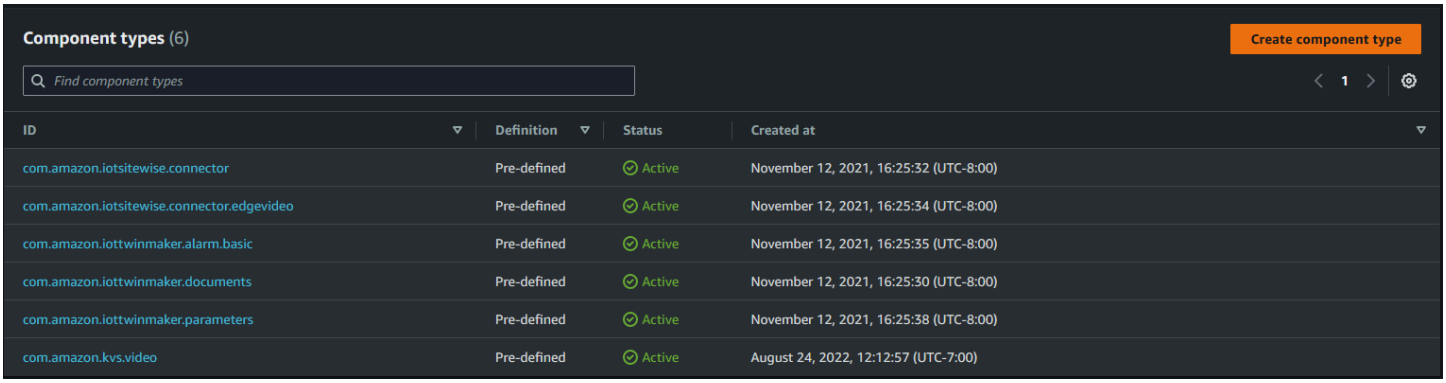
本主题将引导您了解用于创建 AWS IoT TwinMaker 组件类型的值和结构。它向您展示了如何创建可以传递给 CreateComponentTypeAPI 的请求对象，也可以使用 AWS IoT TwinMaker 控制台中的组件类型编辑器来创建请求对象。

组件为其关联实体的属性和数据提供上下文。

内置组件类型

在 AWS IoT TwinMaker 控制台中，当您选择工作区，然后在左侧窗格中选择组件类型时，您会看到以下组件类型。

- com.amazon.iotsitewise.resourcesync：一种组件类型，可自动同步您的 AWS IoT SiteWise 资产和资产模型并将其转换为实体、组件和组件类型。 AWS IoT TwinMaker 有关使用 AWS IoT SiteWise 资产同步的更多信息，请参阅与资产同步 AWS IoT SiteWise。
- com.amazon.iottwinmaker.alarm.basic：一个基本的警报组件，可将警报数据从外部来源提取到实体。此组件未包含连接到特定数据来源的函数。这意味着警报组件是抽象的，可以由另一种组件类型继承，该组件类型指定了数据来源和从该数据来源读取的函数。
- com.amazon.iottwinmaker.documents：包含实体相关信息的文档的标题到 URL 的简单映射。
- com.amazon.iotsitewise.connector.edgevideo：一种使用边缘连接器 for Kinesis Video Streams 组件将视频从物联网设备提取到实体的组件。 AWS IoT Greengrass Kinesis Video AWS IoT Greengrass Streams 的 Edge Con AWS IoT TwinMaker nec tor 组件不是组件，而是一个部署在物联网设备上的本地 AWS IoT Greengrass 预构建组件。
- com.amazon.iotsitewise.connector：一种将 AWS IoT SiteWise 数据提取到实体的组件。
- com.amazon.iottwinmaker.parameters：一种向实体添加静态键值对的组件。
- com.amazon.kvs.video：将视频从 K inesis Video Streams 提取到实体的组件。 AWS IoT TwinMaker



AWS IoT TwinMaker 组件类型的核心特征

下面的列表介绍了组件类型的核心特征。

- 属性定义：该PropertyDefinitionRequest对象定义了一个可以在场景编辑器中填充的属性，也可以使用从外部数据源提取的数据进行填充。您设置的静态属性存储在中 AWS IoT TwinMaker。从数据来源提取的时间序列属性和其他属性存储在外部。

  可以在 `PropertyDefinitionRequest` 映射的字符串中指定属性定义。每个字符串对于映射必须是唯一的。

- 函数：该FunctionRequest对象指定一个 Lambda 函数，该函数从外部数据源读取并可能写入外部数据源。

  如果组件类型包含一个属性，其值存储在外部但没有相应的函数来检索这些值，则它属于抽象组件类型。您可以从抽象组件类型扩展具体的组件类型。您无法向实体添加抽象组件类型。它们不会出现在场景编辑器中。

  您可以在 `FunctionRequest` 映射的字符串中指定函数。该字符串必须指定以下一个预定义函数类型。

  - `dataReader`：一个从外部来源提取数据的函数。
  - `dataReaderByEntity`：一个从外部来源提取数据的函数。

    当您使用这种类型的数据读取器时，GetPropertyValueHistoryAPI 操作仅支持针对该组件类型的属性的实体特定查询。（只能请求 `componentName` + `entityId` 的属性值历史记录。）
  - `dataReaderByComponentType`：一个从外部来源提取数据的函数。

    当您使用这种类型的数据读取器时，GetPropertyValueHistoryAPI 操作仅支持对该组件类型中的属性的跨实体查询。（只能请求 `componentTypeId` 的属性值历史记录。）

- `dataWriter`：一种将数据写入外部来源的函数。
- `schemaInitializer`：一个在每次创建包含组件类型的实体时都会自动初始化属性值的函数。

非抽象组件类型需要三种类型的数据读取器函数之一。

有关实现时间流遥测组件（包括警报）的 Lambda 函数的示例，请参阅AWS IoT TwinMaker 示例中的数据读取器。

> **Note**
>
> 由于警报连接器继承自抽象警报组件类型，因此 Lambda 函数必须返回 `alarm_key` 值。如果没有返回此值，Grafana 将无法将其识别为警报。这对于所有返回警报的组件都是必需的。

- 继承：组件类型通过继承来提高代码的可重用性。一个组件类型可以继承最多 10 个父组件类型。

  使用 `extendsFrom` 参数指定组件类型，您的组件类型将从那里继承属性和函数。
- isSingleton：某些组件包含属性，如位置坐标，这些属性在一个实体中不能包含多次。将 `isSingleton` 参数的值设置为 `true`，表示组件类型在一个实体中只能包含一次。

创建属性定义

下表描述 `PropertyDefinitionRequest` 的参数。

参数	描述
`isExternalId`	一个布尔值，它指定该属性是否是外部存储的属性值的唯一标识符（例如 AWS IoT SiteWise 资产 ID）。 此属性的默认值为 `false`。
`isStoredExternally`	指定是否在外部存储属性的布尔值。 此属性的默认值为 `false`。
`isTimeSeries`	指定属性是否存储时间序列数据的布尔值。

参数	描述
	此属性的默认值为 false
isRequiredInEntity	一个布尔值，用于指定属性在使用该组件类型的实体中是否必须要有一个值。
dataType	一个DataType对象，用于指定属性的数据类型（例如字符串、地图、列表和度量单位）。
defaultValue	一个指定属性的默认值的DataValue对象。
configuration	一种 string-to-string 地图，用于指定连接到外部数据源所需的其他信息。

创建函数

下表描述 FunctionRequest 的参数。

参数	描述
implementedBy	一个DataConnector对象，它指定连接到外部数据源的 Lambda 函数。
requiredProperties	该函数从外部数据来源读取和写入外部数据来源所需属性的列表。
scope	函数的范围。Workspace 用于范围跨越整个工作区的函数。Entity 用于范围仅限于包含该组件的实体的函数。

有关说明如何创建和扩展组件类型的示例，请参阅 ???。

示例组件类型

此主题包含的示例说明了如何实现组件类型的关键概念。

警报（抽象）

以下示例是 AWS IoT TwinMaker 控制台中显示的抽象警报组件类型。它包含一个由没有 `implementedBy` 值的 `dataReader` 组成的 `functions` 列表。

```
{
  "componentTypeId": "com.example.alarm.basic:1",
  "workspaceId": "MyWorkspace",
  "description": "Abstract alarm component type",
  "functions": {
    "dataReader": {
         "isInherited": false
    }
  },
  "isSingleton": false,
  "propertyDefinitions": {
    "alarm_key": {
      "dataType": { "type": "STRING" },
      "isExternalId": true,
      "isRequiredInEntity": true,
      "isStoredExternally": false,
      "isTimeSeries": false
    },
    "alarm_status": {
      "dataType": {
        "allowedValues": [
          {
            "stringValue": "ACTIVE"
          },
          {
            "stringValue": "SNOOZE_DISABLED"
          },
          {
            "stringValue": "ACKNOWLEDGED"
          },
          {
            "stringValue": "NORMAL"
          }
        ],
        "type": "STRING"
      },
      "isRequiredInEntity": false,
```

```
      "isStoredExternally": true,
      "isTimeSeries": true
    }
  }
}
```

备注：

componentTypeId 和 workspaceID 的值为必需值。componentTypeId 的值对工作区必须是唯一性的。alarm_key 的值是函数可用于从外部来源检索警报数据的唯一标识符。密钥的值是必需的，并存储在中 AWS IoT TwinMaker。alarm_status 时间序列值存储在外部来源中。

更多示例参见 AWS IoT TwinMaker 示例。

时间流遥测

以下示例是一个简单的组件类型，它从外部来源检索有关特定类型组件（例如警报或 Cookie 混合器）的遥测数据。它指定了组件类型继承的 Lambda 函数。

```
{
    "componentTypeId": "com.example.timestream-telemetry",
    "workspaceId": "MyWorkspace",
    "functions": {
        "dataReader": {
            "implementedBy": {
                "lambda": {
                    "arn": "lambdaArn"
                }
            }
        }
    },
    "propertyDefinitions": {
        "telemetryType": {
            "dataType": { "type": "STRING" },
            "isExternalId": false,
            "isStoredExternally": false,
            "isTimeSeries": false,
            "isRequiredInEntity": true
        },
        "telemetryId": {
```

```
            "dataType": { "type": "STRING" },
            "isExternalId": false,
            "isStoredExternally": false,
            "isTimeSeries": false,
            "isRequiredInEntity": true
        }
    }
}
```

警报（继承自抽象警报）

以下示例继承自抽象警报和时间流遥测组件类型。它指定了自己的 Lambda 函数，用于检索警报数据。

```
{
    "componentTypeId": "com.example.cookiefactory.alarm",
    "workspaceId": "MyWorkspace",
    "extendsFrom": [
        "com.example.timestream-telemetry",
        "com.amazon.iottwinmaker.alarm.basic"
    ],
    "propertyDefinitions": {
        "telemetryType": {
            "defaultValue": {
                "stringValue": "Alarm"
            }
        }
    },
    "functions": {
        "dataReader": {
            "implementedBy": {
                "lambda": {
                    "arn": "lambdaArn"
                }
            }
        }
    }
}
```

> **Note**
>
> 由于警报连接器继承自抽象警报组件类型，因此 Lambda 函数必须返回 `alarm_key` 值。如果没有返回此值，Grafana 将无法将其识别为警报。这对于所有返回警报的组件都是必需的。

设备示例

本部分的示例说明如何对潜在的设备进行建模。可以使用这些示例来了解如何在自己的流程中对设备进行建模。

饼干搅拌机

以下示例继承自时间流遥测组件类型。它为饼干搅拌机的旋转速度和温度指定了额外的时间序列属性。

```
{
    "componentTypeId": "com.example.cookiefactory.mixer",
    "workspaceId": "MyWorkspace",
    "extendsFrom": [
        "com.example.timestream-telemetry"
    ],
    "propertyDefinitions": {
        "telemetryType": {
            "defaultValue" : { "stringValue": "Mixer" }
        },
        "RPM": {
            "dataType": { "type": "DOUBLE" },
            "isTimeSeries": true,
            "isStoredExternally": true
        },
        "Temperature": {
            "dataType": { "type": "DOUBLE" },
            "isTimeSeries": true,
            "isStoredExternally": true
        }
    }
}
```

水箱

以下示例继承自时间流遥测组件类型。它为水箱的体积和流速指定了额外的时间序列属性。

```
{
    "componentTypeId": "com.example.cookiefactory.watertank",
    "workspaceId": "MyWorkspace",
    "extendsFrom": [
        "com.example.timestream-telemetry"
    ],
    "propertyDefinitions": {
        "telemetryType": {
            "defaultValue" : { "stringValue": "WaterTank" }
        },
        "tankVolume1": {
            "dataType": { "type": "DOUBLE" },
            "isTimeSeries": true,
            "isStoredExternally": true
        },
        "tankVolume2": {
            "dataType": { "type": "DOUBLE" },
            "isTimeSeries": true,
            "isStoredExternally": true
        },
        "flowRate1": {
            "dataType": { "type": "DOUBLE" },
            "isTimeSeries": true,
            "isStoredExternally": true
        },
        "flowrate2": {
            "dataType": { "type": "DOUBLE" },
            "isTimeSeries": true,
            "isStoredExternally": true
        }
    }
}
```

空间位置

以下示例包含属性，其值存储在中 AWS IoT TwinMaker。由于这些值由用户指定并存储在内部，因此无需任何函数即可检索它们。该示例还使用 RELATIONSHIP 数据类型来指定与另一组件类型的关系。

该组件提供了一种用于向数字孪生添加上下文的轻量级机制。可以用它来添加指出某物所在位置的元数据。还可以在逻辑中使用这些信息，用于确定哪些摄像头可以看到设备或空间，或者用于知道如何派人到某个地点。

```
{
    "componentTypeId": "com.example.cookiefactory.space",
    "workspaceId": "MyWorkspace",
    "propertyDefinitions": {
        "position":  {"dataType": {"nestedType": {"type": "DOUBLE"},"type": "LIST"}},
        "rotation":  {"dataType": {"nestedType": {"type": "DOUBLE"},"type": "LIST"}},
        "bounds":  {"dataType": {"nestedType": {"type": "DOUBLE"},"type": "LIST"}},
        "parent_space" : { "dataType": {"type": "RELATIONSHIP"}}
    }
}
```

AWS IoT TwinMaker 批量操作

使用 metadataTransferJob 可以大规模转移和管理您的 AWS IoT TwinMaker 资源。A metadataTransferJob 允许您在和和 Amazon S3 之间执行批量操作 AWS IoT TwinMaker AWS IoT SiteWise 和传输资源。

您可以在以下场景中使用批量操作：

- 在账户之间大规模迁移资产和数据，例如从开发账户迁移到生产账户。
- 大规模资产管理，例如大规模上传和编辑 AWS IoT 资产。
- 将您的资产批量导入 AWS IoT TwinMaker 和 AWS IoT SiteWise。
- 从现有本体文件（例如`revit`或`BIM`文件）中批量导入 AWS IoT TwinMaker 实体。

主题

关键概念和术语

AWS IoT TwinMaker 批量操作使用以下概念和术语：

- 导入：将资源移入 AWS IoT TwinMaker 工作区的操作。例如，从本地文件、Amazon S3 存储桶中的文件或从 AWS IoT SiteWise 到 AWS IoT TwinMaker 工作空间。
- 导出：将资源从 AWS IoT TwinMaker 工作空间移动到本地计算机或 Amazon S3 存储桶的操作。
- 来源：您要移动资源的起始位置。

  例如，Amazon S3 存储桶是导入源， AWS IoT TwinMaker 工作空间是导出源。
- 目的地：您要将资源转移到的所需位置。

  例如，Amazon S3 存储桶是导出目标，而 AWS IoT TwinMaker 工作空间是导入目标。
- AWS IoT SiteWise 架构：用于导入和导出资源的架构 AWS IoT SiteWise。
- AWS IoT TwinMaker 架构：用于导入和导出资源的架构 AWS IoT TwinMaker。
- AWS IoT TwinMaker 顶级资源：现有 API 中使用的资源。具体而言，一个实体或 ComponentType.

- AWS IoT TwinMaker 子级资源：元数据定义中使用的嵌套资源类型。具体而言，是组件。
- 元数据：成功导入或导出所需的关键 AWS IoT SiteWise 信息和 AWS IoT TwinMaker 资源。
- metadataTransferJob：运行时创建的对象`CreateMetadataTransferJob`。

AWS IoT TwinMaker metadataTransferJob 功能

本主题说明了运行批量操作时 AWS IoT TwinMaker 所遵循的行为， metadataTransferJob 即如何处理。它还说明了如何使用传输资源所需的元数据来定义架构。 AWS IoT TwinMaker 批量操作支持以下功能：

- 顶级资源创建或替换： AWS IoT TwinMaker 将创建新资源或替换所有由资源 ID 唯一标识的现有资源。

  例如，如果系统中存在一个实体，则该实体定义将被该`Entity`密钥下模板中定义的新实体定义所取代。
- 子资源创建或替换：

  在 EntityComponent 关卡中，您只能创建或替换组件。该实体必须已经存在，否则，该操作将生成 ValidationException.

  在属性或关系级别上，您只能创建或替换属性或关系，并且包含的属性或关系 EntityComponent 必须已经存在。
- 子资源删除：

  AWS IoT TwinMaker 还支持删除子资源。子资源可以是组件、属性或关系。

  如果要删除组件，则必须从实体级别进行删除。

  如果要删除属性或关系，则必须从 “实体” 或 “ EntityComponent 关卡” 中删除。

  要删除子资源，请更新更高级别的资源并省略子资源的定义。
- 不删除顶级资源： AWS IoT TwinMaker 永远不会删除顶级资源。顶级资源是指实体或 ComponentType。
- 在一个模板中没有针对同一顶级资源的子资源定义：

  您不能在同一个模板中提供同一实体的完整实体定义和子资源（如属性）定义。

  如果实体中使用了 EntityID，则不能在实体 EntityComponent、属性或关系中使用相同的 ID。

如果在中使用了 EntityID 或 componentName 组合 EntityComponent，则不能在 EntityComponent、属性或关系中使用相同的组合。

如果在属性或关系中使用了 EntityID、componentName、PropertyName 组合，则不能在属性或关系中使用相同的组合。

- ExternalId 是可选的 AWS IoT TwinMaker： ExternalId 可用于帮助您识别资源。

执行批量导入和导出操作

本主题介绍如何执行批量导入和导出操作以及如何处理传输任务中的错误。它提供了使用 CLI 命令传输任务的示例。

AWS IoT TwinMaker API 参考包含有关 CreateMetadataTransferJob和其他 API 操作的信息。

主题

metadataTransferJob 先决条件

在运行之前，请完成以下先决条件 metadataTransferJob：

- 创建 AWS IoT TwinMaker 工作空间。工作区可以是的导入目标或导出源 metadataTransferJob。有关创建工作空间的信息，请参阅创建工作区。
- 创建 Amazon S3 存储桶来存储资源。有关使用 Amazon S3 的更多信息，请参阅 “什么是亚马逊 S3？

IAM 权限

执行批量操作时，您需要创建一个 IAM 策略，该策略具有允许 Amazon S3、 AWS IoT TwinMaker AWS IoT SiteWise、和您的本地计算机之间交换 AWS 资源的权限。有关创建 IAM 策略的更多信息，请参阅创建 IAM 策略。

以下列出了 AWS IoT TwinMaker、 AWS IoT SiteWise 和 Amazon S3 的政策声明：

- AWS IoT TwinMaker 政策：

```
{
    "Version": "2012-10-17",
    "Statement": [{
        "Effect": "Allow",
        "Action": [
            "s3:PutObject",
            "s3:GetObject",
            "s3:GetBucketLocation",
            "s3:ListBucket",
            "s3:AbortMultipartUpload",
            "s3:ListBucketMultipartUploads",
            "s3:ListMultipartUploadParts"
        ],
        "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "iottwinmaker:GetWorkspace",
                "iottwinmaker:CreateEntity",
                "iottwinmaker:GetEntity",
                "iottwinmaker:UpdateEntity",
                "iottwinmaker:GetComponentType",
                "iottwinmaker:CreateComponentType",
                "iottwinmaker:UpdateComponentType",
                "iottwinmaker:ListEntities",
                "iottwinmaker:ListComponentTypes",
                "iottwinmaker:ListTagsForResource",
                "iottwinmaker:TagResource",
                "iottwinmaker:UntagResource"
            ],
            "Resource": "*"
        }
```

```
    ]
}
```

- AWS IoT SiteWise 政策：

```
{
    "Version": "2012-10-17",
    "Statement": [{
        "Effect": "Allow",
        "Action": [
            "s3:PutObject",
            "s3:GetObject",
            "s3:GetBucketLocation",
            "s3:ListBucket",
            "s3:AbortMultipartUpload",
            "s3:ListBucketMultipartUploads",
            "s3:ListMultipartUploadParts"
        ],
        "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "iotsitewise:CreateAsset",
                "iotsitewise:CreateAssetModel",
                "iotsitewise:UpdateAsset",
                "iotsitewise:UpdateAssetModel",
                "iotsitewise:UpdateAssetProperty",
                "iotsitewise:ListAssets",
                "iotsitewise:ListAssetModels",
                "iotsitewise:ListAssetProperties",
                "iotsitewise:ListAssetModelProperties",
                "iotsitewise:ListAssociatedAssets",
                "iotsitewise:DescribeAsset",
                "iotsitewise:DescribeAssetModel",
                "iotsitewise:DescribeAssetProperty",
                "iotsitewise:AssociateAssets",
                "iotsitewise:DisassociateAssets",
                "iotsitewise:AssociateTimeSeriesToAssetProperty",
                "iotsitewise:DisassociateTimeSeriesFromAssetProperty",
                "iotsitewise:BatchPutAssetPropertyValue",
                "iotsitewise:BatchGetAssetPropertyValue",
                "iotsitewise:TagResource",
                "iotsitewise:UntagResource",
```

```
                "iotsitewise:ListTagsForResource"
            ],
            "Resource": "*"
        }
    ]
}
```

- 亚马逊 S3 政策：

```
{
    "Effect": "Allow",
    "Action": [
        "s3:PutObject",
        "s3:GetObject",
        "s3:GetBucketLocation",
        "s3:ListBucket",
        "s3:AbortMultipartUpload",
        "s3:ListBucketMultipartUploads",
        "s3:ListMultipartUploadParts"
    ],
    "Resource": "*"
}
```

或者，您可以将您的 Amazon S3 策略范围限制为仅访问单个 Amazon S3 存储桶，请参阅以下政策。

Amazon S3 单存储桶范围政策

```
{
    "Effect": "Allow",
    "Action": [
        "s3:PutObject",
        "s3:GetObject",
        "s3:GetBucketLocation",
        "s3:ListBucket",
        "s3:AbortMultipartUpload",
        "s3:ListBucketMultipartUploads",
        "s3:ListMultipartUploadParts"
    ],
    "Resource": [
        "arn:aws:s3:::bucket name",
        "arn:aws:s3:::bucket name/*"
    ]
```

```
}
```

为设置访问控制 metadataTransferJob

要控制用户可以访问的任务类型，请将以下 IAM 策略添加到用于调用的角色中 AWS IoT TwinMaker。

> **Note**
>
> 此策略仅允许访问在 Amazon S3 之间转移资源的导 AWS IoT TwinMaker 入和导出任务。

```
{
    "Effect": "Allow",
    "Action": [
        "iottwinmaker:*DataTransferJob*"
    ],
    "Resource": "*",
    "Condition": {
        "StringLikeIfExists": {
            "iottwinmaker:sourceType": [
                "s3",
                "iottwinmaker"
            ],
            "iottwinmaker:destinationType": [
                "iottwinmaker",
                "s3"
            ]
        }
    }
}
```

运行批量操作

本节介绍如何执行批量导入和导出操作。

将数据从 Amazon S3 导入到 AWS IoT TwinMaker

1. 使用 AWS IoT TwinMaker metadataTransferJob 架构指定要传输的资源。创建您的架构文件并将其存储在您的 Amazon S3 存储桶中。

   有关架构示例，请参阅导入元数据模板。

2. 创建请求正文并将其另存为 JSON 文件。请求正文指定传输任务的来源和目的地。请务必将您的 Amazon S3 存储桶指定为源，并将您的 AWS IoT TwinMaker 工作空间指定为目标。

   以下是请求正文的示例：

```
{
    "metadataTransferJobId": "your-transfer-job-Id",
    "sources": [{
        "type": "s3",
        "s3Configuration": {
            "location": "arn:aws:s3:::your-S3-bucket-name/your_import_data.json"
        }
    }],
    "destination": {
        "type": "iottwinmaker",
        "iotTwinMakerConfiguration": {
            "workspace": "arn:aws:iottwinmaker:us-
east-1:111122223333:workspace/your-worksapce-name"
        }
    }
}
```

   记录下您为请求正文提供的文件名，下一步将需要它。在此示例中，请求正文被命名为`createMetadataTransferJobImport.json`。

3. 运行以下 CLI 命令进行调用`CreateMetadataTransferJob`（将 input-json 文件名替换为你为请求正文提供的名称）：

```
aws iottwinmaker create-metadata-transfer-job --region us-east-1 \
--cli-input-json file://createMetadataTransferJobImport.json
```

   这将创建 metadataTransferJob 并开始转移所选资源的过程。

将数据从导出 AWS IoT TwinMaker 到 Amazon S3

1. 创建带有相应过滤器的 JSON 请求正文以选择要导出的资源。在这个例子中，我们使用：

```
{
    "metadataTransferJobId": "your-transfer-job-Id",
    "sources": [{
        "type": "iottwinmaker",
```

```
        "iotTwinMakerConfiguration": {
            "workspace": "arn:aws:iottwinmaker:us-
east-1:111122223333:workspace/your-workspace-name",
            "filters": [{
                "filterByEntity": {
                    "entityId": "parent"
                }},
                {
                "filterByEntity": {
                    "entityId": "child"
                }},
                {
                "filterByComponentType": {
                    "componentTypeId": "component.type.minimal"
                }}
            ]
        }
    }],
    "destination": {
        "type": "s3",
        "s3Configuration": {
            "location": "arn:aws:s3:::your-S3-bucket-location"
        }
    }
}
```

该filters数组允许您指定要导出的资源。在此示例中，我们按entity、和进行筛选componentType。

请务必将您的 AWS IoT TwinMaker 工作空间指定为源，将 Amazon S3 存储桶指定为元数据传输任务的目标。

保存您的请求正文并记录文件名，下一步将需要它。在这个例子中，我们命名了请求正文createMetadataTransferJobExport.json。

2. 运行以下 CLI 命令进行调用CreateMetadataTransferJob（将 input-json 文件名替换为你为请求正文提供的名称）：

```
aws iottwinmaker create-metadata-transfer-job --region us-east-1 \
--cli-input-json file://createMetadataTransferJobExport.json
```

这将创建 metadataTransferJob 并开始转移所选资源的过程。

要检查或更新传输任务的状态，请使用以下命令：

- 要取消任务，请使用 CancelMetadataTransferJobAPI 操作。当您调用时 CancelMetadataTransferJob，API 只会取消正在运行 metadataTransferJob，并且任何已导出或导入的资源都不会受到此 API 调用的影响。
- 要检索有关特定任务的信息，请使用 GetMetadataTransferJobAPI 操作。

  或者，您可以使用以下 CLI 命令调 GetMetadataTransferJob 用现有传输任务：

  ```
  aws iottwinmaker get-metadata-transfer-job --job-id ExistingJobId
  ```

  如果您调用不存在 GetMetadataTransferJob 的 AWS IoT TwinMaker 导入或导出任务，则会收到`ResourceNotFoundException`错误的响应。
- 要列出当前作业，请使用 ListMetadataTransferJobsAPI 操作。

  以下是以 destinationType 和 SourceType AWS IoT TwinMaker `s3` 作为源类型调 ListMetadataTransferJobs 用的 CLI 示例：

  ```
  aws iottwinmaker list-metadata-transfer-jobs --destination-type iottwinmaker --
  source-type s3
  ```

  > **Note**
  >
  > 您可以更改 SourceType 和 destinationType 参数的值，使其与导入或导出任务的源和目标相匹配。

有关调用这些 API 操作的 CLI 命令的更多示例，请参阅AWS IoT TwinMaker metadataTransferJob 例子。

如果您在传输任务期间遇到任何错误，请参阅错误处理。

错误处理

创建并运行传输任务后，您可以 GetMetadataTransferJob致电诊断出现的任何错误：

```
aws iottwinmaker get-metadata-transfer-job \
--metadata-transfer-job-id your_metadata_transfer_job_id \
```

```
--region us-east-1
```

看到作业的状态变为后COMPLETED，就可以验证作业的结果了。 GetMetadataTransferJob 返回一个名为的对象 MetadataTransferJobProgress，其中包含以下字段：

- failedCount：表示在转移过程中出现故障的资源数量。
- skippedCount：表示在转移过程中跳过的资源数量。
- succededCount：表示在转移过程中成功的资源数量。
- TotalCount：表示转移过程中涉及的资源总数。

此外，还会返回一个包含预签名网址的 reportUrl 元素。如果您的转移任务存在错误，您想进一步调查，则可以使用此网址下载完整的错误报告。

导入元数据模板

您可以通过一次批量导入操作导入多个组件、组件类型或实体。本节中的示例说明了如何执行此操作。

template: Importing entities

对于导入实体的作业，请使用以下模板格式：

```
{
  "entities": [
    {
      "description": "string",
      "entityId": "string",
      "entityName": "string",
      "parentEntityId": "string",
      "tags": {
        "string": "string"
      },
      "components": {
        "string": {
          "componentTypeId": "string",
          "description": "string",
          "properties": {
            "string": {
              "definition": {
                "configuration": {
                  "string": "string"
                },
```

```
                "dataType": "DataType",
                "defaultValue": "DataValue",
                "displayName": "string",
                "isExternalId": "boolean",
                "isRequiredInEntity": "boolean",
                "isStoredExternally": "boolean",
                "isTimeSeries": "boolean"
              },
              "value": "DataValue"
            }
          },
          "propertyGroups": {
            "string": {
              "groupType": "string",
              "propertyNames": [
                "string"
              ]
            }
          }
        }
      }
    }
  ]
}
```

template: Importing componentTypes

对于导入 ComponentTypes 的作业，请使用以下模板格式：

```
{
  "componentTypes": [
    {
      "componentTypeId": "string",
      "componentTypeName": "string",
      "description": "string",
      "extendsFrom": [
        "string"
      ],
      "functions": {
        "string": {
          "implementedBy": {
            "isNative": "boolean",
            "lambda": {
              "functionName": "Telemetry-tsDataReader",
```

```
              "arn": "Telemetry-tsDataReaderARN"
            }
          },
          "requiredProperties": [
            "string"
          ],
          "scope": "string"
        }
      },
      "isSingleton": "boolean",
      "propertyDefinitions": {
        "string": {
          "configuration": {
            "string": "string"
          },
          "dataType": "DataType",
          "defaultValue": "DataValue",
          "displayName": "string",
          "isExternalId": "boolean",
          "isRequiredInEntity": "boolean",
          "isStoredExternally": "boolean",
          "isTimeSeries": "boolean"
        }
      },
      "propertyGroups": {
        "string": {
          "groupType": "string",
          "propertyNames": [
            "string"
          ]
        }
      },
      "tags": {
        "string": "string"
      }
    }
  ]
}
```

template: Importing components

对于导入组件的作业，请使用以下模板格式：

```
{
```

```
  "entityComponents": [
    {
      "entityId": "string",
      "componentName": "string",
      "componentTypeId": "string",
      "description": "string",
      "properties": {
        "string": {
          "definition": {
            "configuration": {
              "string": "string"
            },
            "dataType": "DataType",
            "defaultValue": "DataValue",
            "displayName": "string",
            "isExternalId": "boolean",
            "isRequiredInEntity": "boolean",
            "isStoredExternally": "boolean",
            "isTimeSeries": "boolean"
          },
          "value": "DataValue"
        }
      },
      "propertyGroups": {
        "string": {
          "groupType": "string",
          "propertyNames": [
            "string"
          ]
        }
      }
    }
  ]
}
```

AWS IoT TwinMaker metadataTransferJob 例子

使用以下命令管理您的元数据传输：

- CreateMetadataTransferJobAPI 操作。

  CLI 命令示例：

```
aws iottwinmaker create-metadata-transfer-job --region us-east-1 \
--cli-input-json file://yourTransferFileName.json
```

- 要取消任务，请使用 CancelMetadataTransferJobAPI 操作。

  CLI 命令示例：

  ```
  aws iottwinmaker cancel-metadata-transfer-job
  --region us-east-1 \
  --metadata-transfer-job-id job-to-cancel-id
  ```

  当您调用时 CancelMetadataTransferJob，它只会取消特定的元数据传输任务，并且任何已导出或导入的资源都不会受到影响。

- 要检索有关特定任务的信息，请使用 GetMetadataTransferJobAPI 操作。

  CLI 命令示例：

  ```
  aws iottwinmaker get-metadata-transfer-job \
  --metadata-transfer-job-id your_metadata_transfer_job_id \
  --region us-east-1 \
  ```

- 要列出当前作业，请使用 ListMetadataTransferJobsAPI 操作。

  您可以使用 JSON 文件 ListMetadataTransferJobs 筛选返回的结果。使用 CLI 查看以下步骤：

  1. 创建 CLI 输入 JSON 文件以指定要使用的过滤器：

     ```
     {
         "sourceType": "s3",
         "destinationType": "iottwinmaker",
         "filters": [{
             "workspaceId": "workspaceforbulkimport"
         },
         {
             "state": "COMPLETED"
         }]
     }
     ```

     保存并记录文件名，输入 CLI 命令时将需要它。

  2. 使用 JSON 文件作为以下 CLI 命令的参数：

```
aws iottwinmaker list-metadata-transfer-job --region us-east-1 \
--cli-input-json file://ListMetadataTransferJobsExample.json
```

AWS IoT TwinMaker 元数据传输任务架构

metadataTransferJob 导入架构：在将数据上传到 Amazon S3 存储桶时，使用此 AWS IoT TwinMaker 元数据架构来验证您的数据：

```
{
  "$schema": "https://json-schema.org/draft/2020-12/schema",
  "title": "IoTTwinMaker",
  "description": "Metadata transfer job resource schema for IoTTwinMaker",
  "definitions": {
    "ExternalId": {
      "type": "string",
      "minLength": 1,
      "maxLength": 128,
      "pattern": "[a-zA-Z0-9][a-zA-Z_\\-0-9.:]*[a-zA-Z0-9]+"
    },
    "Description": {
      "type": "string",
      "minLength": 0,
      "maxLength": 512
    },
    "DescriptionWithDefault": {
      "type": "string",
      "minLength": 0,
      "maxLength": 512,
      "default": ""
    },
    "ComponentTypeName": {
      "description": "A friendly name for the component type.",
      "type": "string",
      "pattern": ".*[^\\u0000-\\u001F\\u007F]*.*",
      "minLength": 1,
      "maxLength": 256
    },
    "ComponentTypeId": {
      "description": "The ID of the component type.",
      "type": "string",
      "pattern": "[a-zA-Z_.\\-0-9:]+",
```

```
      "minLength": 1,
      "maxLength": 256
    },
    "ComponentName": {
      "description": "The name of the component.",
      "type": "string",
      "pattern": "[a-zA-Z_\\-0-9]+",
      "minLength": 1,
      "maxLength": 256
    },
    "EntityId": {
      "description": "The ID of the entity.",
      "type": "string",
      "minLength": 1,
      "maxLength": 128,
      "pattern": "[0-9a-f]{8}-[0-9a-f]{4}-[0-9a-f]{4}-[0-9a-f]{4}-[0-9a-f]{12}|^[a-zA-
Z0-9][a-zA-Z_\\-0-9.:]*[a-zA-Z0-9]+"
    },
    "EntityName": {
      "description": "The name of the entity.",
      "type": "string",
      "minLength": 1,
      "maxLength": 256,
      "pattern": "[a-zA-Z_0-9-.][a-zA-Z_0-9-. ]*[a-zA-Z0-9]+"
    },
    "ParentEntityId": {
      "description": "The ID of the parent entity.",
      "type": "string",
      "minLength": 1,
      "maxLength": 128,
      "pattern": "\\$ROOT|^[0-9a-f]{8}-[0-9a-f]{4}-[0-9a-f]{4}-[0-9a-f]{4}-[0-9a-f]
{12}|^[a-zA-Z0-9][a-zA-Z_\\-0-9.:]*[a-zA-Z0-9]+",
      "default": "$ROOT"
    },
    "DisplayName": {
      "description": "A friendly name for the property.",
      "type": "string",
      "pattern": ".*[^\\u0000-\\u001F\\u007F]*.*",
      "minLength": 0,
      "maxLength": 256
    },
    "Tags": {
      "description": "Metadata that you can use to manage the entity / componentType",
      "patternProperties": {
```

```
        "^([\\p{L}\\p{Z}\\p{N}_.:/=+\\-@]*)$": {
          "type": "string",
          "minLength": 1,
          "maxLength": 256
        }
      },
      "existingJavaType": "java.util.Map<String,String>",
      "minProperties": 0,
      "maxProperties": 50
    },
    "Relationship": {
      "description": "The type of the relationship.",
      "type": "object",
      "properties": {
        "relationshipType": {
          "description": "The type of the relationship.",
          "type": "string",
          "pattern": ".*",
          "minLength": 1,
          "maxLength": 256
        },
        "targetComponentTypeId": {
          "description": "The ID of the target component type associated with this
relationship.",
          "$ref": "#/definitions/ComponentTypeId"
        }
      },
      "additionalProperties": false
    },
    "DataValue": {
      "description": "An object that specifies a value for a property.",
      "type": "object",
      "properties": {
        "booleanValue": {
          "description": "A Boolean value.",
          "type": "boolean"
        },
        "doubleValue": {
          "description": "A double value.",
          "type": "number"
        },
        "expression": {
          "description": "An expression that produces the value.",
          "type": "string",
```

```
          "pattern": "(^\\$\\{Parameters\\.[a-zA-z]+([a-zA-z_0-9]*)}$)",
          "minLength": 1,
          "maxLength": 316
        },
        "integerValue": {
          "description": "An integer value.",
          "type": "integer"
        },
        "listValue": {
          "description": "A list of multiple values.",
          "type": "array",
          "minItems": 0,
          "maxItems": 50,
          "uniqueItems": false,
          "insertionOrder": false,
          "items": {
            "$ref": "#/definitions/DataValue"
          },
          "default": null
        },
        "longValue": {
          "description": "A long value.",
          "type": "integer",
          "existingJavaType": "java.lang.Long"
        },
        "stringValue": {
          "description": "A string value.",
          "type": "string",
          "pattern": ".*",
          "minLength": 1,
          "maxLength": 256
        },
        "mapValue": {
          "description": "An object that maps strings to multiple DataValue objects.",
          "type": "object",
          "patternProperties": {
            "[a-zA-Z_\\-0-9]+": {
              "$ref": "#/definitions/DataValue"
            }
          },
          "additionalProperties": {
            "$ref": "#/definitions/DataValue"
          }
        },
```

```
        "relationshipValue": {
          "description": "A value that relates a component to another component.",
          "type": "object",
          "properties": {
            "TargetComponentName": {
              "type": "string",
              "pattern": "[a-zA-Z_\\-0-9]+",
              "minLength": 1,
              "maxLength": 256
            },
            "TargetEntityId": {
              "type": "string",
              "pattern": "[0-9a-f]{8}-[0-9a-f]{4}-[0-9a-f]{4}-[0-9a-f]{4}-[0-9a-f]{12}|
^[a-zA-Z0-9][a-zA-Z_\\-0-9.:]*[a-zA-Z0-9]+",
              "minLength": 1,
              "maxLength": 128
            }
          },
          "additionalProperties": false
        }
      },
      "additionalProperties": false
    },
    "DataType": {
      "description": "An object that specifies the data type of a property.",
      "type": "object",
      "properties": {
        "allowedValues": {
          "description": "The allowed values for this data type.",
          "type": "array",
          "minItems": 0,
          "maxItems": 50,
          "uniqueItems": false,
          "insertionOrder": false,
          "items": {
            "$ref": "#/definitions/DataValue"
          },
          "default": null
        },
        "nestedType": {
          "description": "The nested type in the data type.",
          "$ref": "#/definitions/DataType"
        },
        "relationship": {
```

```
          "description": "A relationship that associates a component with another
 component.",
          "$ref": "#/definitions/Relationship"
        },
        "type": {
          "description": "The underlying type of the data type.",
          "type": "string",
          "enum": [
            "RELATIONSHIP",
            "STRING",
            "LONG",
            "BOOLEAN",
            "INTEGER",
            "DOUBLE",
            "LIST",
            "MAP"
          ]
        },
        "unitOfMeasure": {
          "description": "The unit of measure used in this data type.",
          "type": "string",
          "pattern": ".*",
          "minLength": 1,
          "maxLength": 256
        }
      },
      "required": [
        "type"
      ],
      "additionalProperties": false
    },
    "PropertyDefinition": {
      "description": "An object that specifies information about a property.",
      "type": "object",
      "properties": {
        "configuration": {
          "description": "An object that specifies information about a property.",
          "patternProperties": {
            "[a-zA-Z_\\-0-9]+": {
              "type": "string",
              "pattern": "[a-zA-Z_\\-0-9]+",
              "minLength": 1,
              "maxLength": 256
            }
```

```
          },
          "existingJavaType": "java.util.Map<String,String>"
        },
        "dataType": {
          "description": "An object that contains information about the data type.",
          "$ref": "#/definitions/DataType"
        },
        "defaultValue": {
          "description": "An object that contains the default value.",
          "$ref": "#/definitions/DataValue"
        },
        "displayName": {
          "description": "An object that contains the default value.",
          "$ref": "#/definitions/DisplayName"
        },
        "isExternalId": {
          "description": "A Boolean value that specifies whether the property ID comes
from an external data store.",
          "type": "boolean",
          "default": null
        },
        "isRequiredInEntity": {
          "description": "A Boolean value that specifies whether the property is
required.",
          "type": "boolean",
          "default": null
        },
        "isStoredExternally": {
          "description": "A Boolean value that specifies whether the property is stored
externally.",
          "type": "boolean",
          "default": null
        },
        "isTimeSeries": {
          "description": "A Boolean value that specifies whether the property consists
of time series data.",
          "type": "boolean",
          "default": null
        }
      },
      "additionalProperties": false
    },
    "PropertyDefinitions": {
      "type": "object",
```

```
      "patternProperties": {
        "[a-zA-Z_\\-0-9]+": {
          "$ref": "#/definitions/PropertyDefinition"
        }
      },
      "additionalProperties": {
        "$ref": "#/definitions/PropertyDefinition"
      }
    },
    "Property": {
      "type": "object",
      "properties": {
        "definition": {
          "description": "The definition of the property",
          "$ref": "#/definitions/PropertyDefinition"
        },
        "value": {
          "description": "The value of the property.",
          "$ref": "#/definitions/DataValue"
        }
      },
      "additionalProperties": false
    },
    "Properties": {
      "type": "object",
      "patternProperties": {
        "[a-zA-Z_\\-0-9]+": {
          "$ref": "#/definitions/Property"
        }
      },
      "additionalProperties": {
        "$ref": "#/definitions/Property"
      }
    },
    "PropertyName": {
      "type": "string",
      "pattern": "[a-zA-Z_\\-0-9]+"
    },
    "PropertyGroup": {
      "description": "An object that specifies information about a property group.",
      "type": "object",
      "properties": {
        "groupType": {
          "description": "The type of property group.",
```

```
          "type": "string",
          "enum": [
            "TABULAR"
          ]
        },
        "propertyNames": {
          "description": "The list of property names in the property group.",
          "type": "array",
          "minItems": 1,
          "maxItems": 256,
          "uniqueItems": true,
          "insertionOrder": false,
          "items": {
            "$ref": "#/definitions/PropertyName"
          },
          "default": null
        }
      },
      "additionalProperties": false
    },
    "PropertyGroups": {
      "type": "object",
      "patternProperties": {
        "[a-zA-Z_\\-0-9]+": {
          "$ref": "#/definitions/PropertyGroup"
        }
      },
      "additionalProperties": {
        "$ref": "#/definitions/PropertyGroup"
      }
    },
    "Component": {
      "type": "object",
      "properties": {
        "componentTypeId": {
          "$ref": "#/definitions/ComponentTypeId"
        },
        "description": {
          "$ref": "#/definitions/Description"
        },
        "properties": {
          "description": "An object that maps strings to the properties to set in the
component type. Each string in the mapping must be unique to this object.",
          "$ref": "#/definitions/Properties"
```

```
        },
        "propertyGroups": {
          "description": "An object that maps strings to the property groups to set in
 the entity component. Each string in the mapping must be unique to this object.",
          "$ref": "#/definitions/PropertyGroups"
        }
      },
      "required": [
        "componentTypeId"
      ],
      "additionalProperties": false
    },
    "RequiredProperty": {
      "type": "string",
      "pattern": "[a-zA-Z_\\-0-9]+"
    },
    "LambdaFunction": {
      "type": "object",
      "properties": {
        "arn": {
          "type": "string",
          "pattern": "arn:((aws)|(aws-cn)|(aws-us-gov)|(\\${partition})):lambda:(([a-
z0-9-]+)|(\\${region})):([0-9]{12}|(\\${accountId})):function:[/a-zA-Z0-9_-]+",
          "minLength": 1,
          "maxLength": 128
        }
      },
      "additionalProperties": false,
      "required": [
        "arn"
      ]
    },
    "DataConnector": {
      "description": "The data connector.",
      "type": "object",
      "properties": {
        "isNative": {
          "description": "A Boolean value that specifies whether the data connector is
 native to IoT TwinMaker.",
          "type": "boolean"
        },
        "lambda": {
          "description": "The Lambda function associated with this data connector.",
          "$ref": "#/definitions/LambdaFunction"
```

```
        }
      },
      "additionalProperties": false
    },
    "Function": {
      "description": "The function of component type.",
      "type": "object",
      "properties": {
        "implementedBy": {
          "description": "The data connector.",
          "$ref": "#/definitions/DataConnector"
        },
        "requiredProperties": {
          "description": "The required properties of the function.",
          "type": "array",
          "minItems": 1,
          "maxItems": 256,
          "uniqueItems": true,
          "insertionOrder": false,
          "items": {
            "$ref": "#/definitions/RequiredProperty"
          },
          "default": null
        },
        "scope": {
          "description": "The scope of the function.",
          "type": "string",
          "enum": [
            "ENTITY",
            "WORKSPACE"
          ]
        }
      },
      "additionalProperties": false
    },
    "Entity": {
      "type": "object",
      "properties": {
        "description": {
          "description": "The description of the entity.",
          "$ref": "#/definitions/DescriptionWithDefault"
        },
        "entityId": {
          "$ref": "#/definitions/EntityId"
```

```
        },
        "entityExternalId": {
          "description": "The external ID of the entity.",
          "$ref": "#/definitions/ExternalId"
        },
        "entityName": {
          "$ref": "#/definitions/EntityName"
        },
        "parentEntityId": {
          "$ref": "#/definitions/ParentEntityId"
        },
        "tags": {
          "$ref": "#/definitions/Tags"
        },
        "components": {
          "description": "A map that sets information about a component.",
          "type": "object",
          "patternProperties": {
            "[a-zA-Z_\\-0-9]+": {
              "$ref": "#/definitions/Component"
            }
          },
          "additionalProperties": {
            "$ref": "#/definitions/Component"
          }
        }
      },
      "required": [
        "entityId",
        "entityName"
      ],
      "additionalProperties": false
    },
    "ComponentType": {
      "type": "object",
      "properties": {
        "description": {
          "description": "The description of the component type.",
          "$ref": "#/definitions/DescriptionWithDefault"
        },
        "componentTypeId": {
          "$ref": "#/definitions/ComponentTypeId"
        },
        "componentTypeExternalId": {
```

```
          "description": "The external ID of the component type.",
          "$ref": "#/definitions/ExternalId"
        },
        "componentTypeName": {
          "$ref": "#/definitions/ComponentTypeName"
        },
        "extendsFrom": {
          "description": "Specifies the parent component type to extend.",
          "type": "array",
          "minItems": 1,
          "maxItems": 256,
          "uniqueItems": true,
          "insertionOrder": false,
          "items": {
            "$ref": "#/definitions/ComponentTypeId"
          },
          "default": null
        },
        "functions": {
          "description": "a Map of functions in the component type. Each function's key
must be unique to this map.",
          "type": "object",
          "patternProperties": {
            "[a-zA-Z_\\-0-9]+": {
              "$ref": "#/definitions/Function"
            }
          },
          "additionalProperties": {
            "$ref": "#/definitions/Function"
          }
        },
        "isSingleton": {
          "description": "A Boolean value that specifies whether an entity can have
more than one component of this type.",
          "type": "boolean",
          "default": false
        },
        "propertyDefinitions": {
          "description": "An map of the property definitions in the component type.
Each property definition's key must be unique to this map.",
          "$ref": "#/definitions/PropertyDefinitions"
        },
        "propertyGroups": {
```

```
          "description": "An object that maps strings to the property groups to set in
the component type. Each string in the mapping must be unique to this object.",
          "$ref": "#/definitions/PropertyGroups"
        },
        "tags": {
          "$ref": "#/definitions/Tags"
        }
      },
      "required": [
        "componentTypeId"
      ],
      "additionalProperties": false
    },
    "EntityComponent": {
      "type": "object",
      "properties": {
        "entityId": {
          "$ref": "#/definitions/EntityId"
        },
        "componentName": {
          "$ref": "#/definitions/ComponentName"
        },
        "componentExternalId": {
          "description": "The external ID of the component.",
          "$ref": "#/definitions/ExternalId"
        },
        "componentTypeId": {
          "$ref": "#/definitions/ComponentTypeId"
        },
        "description": {
          "description": "The description of the component.",
          "$ref": "#/definitions/Description"
        },
        "properties": {
          "description": "An object that maps strings to the properties to set in the
component. Each string in the mapping must be unique to this object.",
          "$ref": "#/definitions/Properties"
        },
        "propertyGroups": {
          "description": "An object that maps strings to the property groups to set in
the component. Each string in the mapping must be unique to this object.",
          "$ref": "#/definitions/PropertyGroups"
        }
      },
```

```
      "required": [
        "entityId",
        "componentTypeId",
        "componentName"
      ],
      "additionalProperties": false
    }
  },
  "additionalProperties": false,
  "properties": {
    "entities": {
      "type": "array",
      "uniqueItems": false,
      "items": {
        "$ref": "#/definitions/Entity"
      }
    },
    "componentTypes": {
      "type": "array",
      "uniqueItems": false,
      "items": {
        "$ref": "#/definitions/ComponentType"
      }
    },
    "entityComponents": {
      "type": "array",
      "uniqueItems": false,
      "items": {
        "$ref": "#/definitions/EntityComponent"
      },
      "default": null
    }
  }
}
```

以下是一个示例，它创建了一个名为的新 ComponentType，`component.type.intial`并创建了一个名为的实体：`initial`

```
{
  "componentTypes": [
    {
      "componentTypeId": "component.type.initial",
      "tags": {
```

```
        "key": "value"
      }
    }
  ],
  "entities": [
    {
      "entityName": "initial",
      "entityId": "initial"
    }
  ]
}
```

以下是更新现有实体的示例：

```
{
  "componentTypes": [
    {
      "componentTypeId": "component.type.initial",
      "description": "updated"
    }
  ],
  "entities": [
    {
      "entityName": "parent",
      "entityId": "parent"
    },
    {
      "entityName": "child",
      "entityId": "child",
      "components": {
        "testComponent": {
          "componentTypeId": "component.type.initial",
          "properties": {
            "testProperty": {
              "definition": {
                "configuration": {
                  "alias": "property"
                },
                "dataType": {
                  "relationship": {
                    "relationshipType": "parent",
                    "targetComponentTypeId": "test"
                  },
```

```
                  "type": "STRING",
                  "unitOfMeasure": "t"
                },
                "displayName": "displayName"
              }
            }
          }
        }
      },
      "parentEntityId": "parent"
    }
  ],
  "entityComponents": [
    {
      "entityId": "initial",
      "componentTypeId": "component.type.initial",
      "componentName": "entityComponent",
      "description": "additionalDescription",
      "properties": {
        "additionalProperty": {
          "definition": {
            "configuration": {
              "alias": "additionalProperty"
            },
            "dataType": {
              "type": "STRING"
            },
            "displayName": "additionalDisplayName"
          },
          "value": {
            "stringValue": "test"
          }
        }
      }
    }
  ]
}
```

AWS IoT TwinMaker 数据连接器

AWS IoT TwinMaker 使用基于连接器的架构，因此您可以将自己的数据存储中的数据连接到。AWS IoT TwinMaker这意味着您无需在使用之前迁移数据 AWS IoT TwinMaker。目前， AWS IoT TwinMaker 支持第一方连接器。 AWS IoT SiteWise如果您在中存储建模和属性数据 AWS IoT SiteWise，则无需实现自己的连接器。如果您将建模或属性数据存储在其他数据存储中，例如 Timestream、DynamoDB 或 Snowflake，则必须 AWS Lambda 实现带有数据连接器接口的连接器，AWS IoT TwinMaker 以便 AWS IoT TwinMaker 在必要时可以调用您的连接器。

主题

AWS IoT TwinMaker 数据连接器

连接器需要访问您的基础数据存储，才能解析已发送的查询并返回结果或错误。

要了解可用连接器、请求接口和响应接口信息，请参阅以下主题。

有关连接器接口中使用的属性的更多信息，请参阅 GetPropertyValueHistoryAPI 操作。

> **Note**
>
> 部分连接器在请求和响应接口有两个时间戳字段，分别表示起始时间和结束时间属性。`startDateTime` 和 `endDateTime` 使用长数字表示纪元秒，不再支持这种表示方式。为了保持向后兼容性，我们仍会向该字段发送时间戳值，但我们建议使用与 API 时间戳格式一致的 `startTime` 和 `endTime` 字段。

主题

- DataReader
- AttributePropertyValueReaderByEntity
- DataWriter
- 示例

架构初始值连接器

您可按组件类型或整个周期使用架构初始值设定项，或从基础数据来源获取组件类型或组件属性。架构初始值设定项自动导入组件类型或组件属性，无需显式调用 API 操作设置 properties。

SchemaInitializer 请求接口

```
{
  "workspaceId": "string",
  "entityId": "string",
  "componentName": "string",
  "properties": {
    // property name as key,
    // value is of type PropertyRequest
    "string": "PropertyRequest"
  }
}
```

> **Note**
>
> 此请求接口中的属性映射为 PropertyRequest。有关更多信息，请参阅PropertyRequest。

SchemaInitializer 响应接口

```
{
  "properties": {
    // property name as key,
    // value is of type PropertyResponse
    "string": "PropertyResponse"
  }
}
```

> **Note**
>
> 此请求接口中的属性映射为 PropertyResponse。有关更多信息，请参阅PropertyResponse。

DataReaderByEntity

DataReaderByEntity 是一个数据平面连接器，用于获取单个组件中属性的时间序列值。

有关此连接器的属性类型、语法和格式的信息，请参阅 GetPropertyValueHistoryAPI 操作。

DataReaderByEntity请求接口

```
{
  "startDateTime": long, // In epoch sec, deprecated
  "startTime": "string", // ISO-8601 timestamp format
  "endDateTime": long, // In epoch sec, deprecated
  "endTime": "string", // ISO-8601 timestamp format
  "properties": {
    // A map of properties as in the get-entity API response
    // property name as key,
    // value is of type PropertyResponse
    "string": "PropertyResponse"
   },
  "workspaceId": "string",
  "selectedProperties": List:"string",
  "propertyFilters": List:PropertyFilter,
  "entityId": "string",
  "componentName": "string",
  "componentTypeId": "string",
  "interpolation": InterpolationParameters,
  "nextToken": "string",
  "maxResults": int,
  "orderByTime": "string"
  }
```

DataReaderByEntity响应接口

```
{
  "propertyValues": [
```

```
    {
      "entityPropertyReference": EntityPropertyReference, // The same
 as EntityPropertyReference
      "values": [
        {
        "timestamp": long, // Epoch sec, deprecated
        "time": "string", // ISO-8601 timestamp format
        "value": DataValue // The same as DataValue
        }
      ]
    }
  ],
  "nextToken": "string"
}
```

DataReaderByComponentType

要获取来自相同组件类型的常用属性的时间序列值，请使用数据平面连接器 DataReaderByEntity。例如，如果您按组件类型定义时间序列属性，并且您有多个词类型组件，您可以查询给定时间范围内所有组件的属性。常见用例如：当您想要查询多个组件的警报状态以获取实体全局视图时。

有关此连接器的属性类型、语法和格式的信息，请参阅 GetPropertyValueHistoryAPI 操作。

DataReaderByComponentType 请求接口

```
{
  "startDateTime": long, // In epoch sec, deprecated
  "startTime": "string", // ISO-8601 timestamp format
  "endDateTime": long, // In epoch sec, deprecated
  "endTime": "string", // ISO-8601 timestamp format
  "properties": { // A map of properties as in the get-entity API response
    // property name as key,
    // value is of type PropertyResponse
    "string": "PropertyResponse"
   },
  "workspaceId": "string",
  "selectedProperties": List:"string",
  "propertyFilters": List:PropertyFilter,
  "componentTypeId": "string",
  "interpolation": InterpolationParameters,
  "nextToken": "string",
  "maxResults": int,
```

```
  "orderByTime": "string"
}
```

DataReaderByComponentType 响应接口

```
{
  "propertyValues": [
    {
      "entityPropertyReference": EntityPropertyReference, // The same
 as EntityPropertyReference
      "entityId": "string",
      "componentName": "string",
      "values": [
        {
        "timestamp": long, // Epoch sec, deprecated
        "time": "string", // ISO-8601 timestamp format
        "value": DataValue // The same as DataValue
        }
      ]
    }
  ],
  "nextToken": "string"
}
```

DataReader

DataReader 是一种数据平面连接器，可以同时处理 DataReaderByEntity 和的情况 DataReaderByComponentType。

有关此连接器的属性类型、语法和格式的信息，请参阅 GetPropertyValueHistoryAPI 操作。

DataReader 请求接口

`EntityId` 和 `componentName` 是可选的。

```
{
  "startDateTime": long, // In epoch sec, deprecated
  "startTime": "string", // ISO-8601 timestamp format
  "endDateTime": long, // In epoch sec, deprecated
  "endTime": "string", // ISO-8601 timestamp format
  "properties": { // A map of properties as in the get-entity API response
```

```
    // property name as key,
    // value is of type PropertyRequest
    "string": "PropertyRequest"
  },

  "workspaceId": "string",
  "selectedProperties": List:"string",
  "propertyFilters": List:PropertyFilter,
  "entityId": "string",
  "componentName": "string",
  "componentTypeId": "string",
  "interpolation": InterpolationParameters,
  "nextToken": "string",
  "maxResults": int,
  "orderByTime": "string"
}
```

DataReader 响应接口

```
{
  "propertyValues": [
    {
      "entityPropertyReference": EntityPropertyReference, // The same
 as EntityPropertyReference
      "values": [
        {
        "timestamp": long, // Epoch sec, deprecated
        "time": "string", // ISO-8601 timestamp format
        "value": DataValue // The same as DataValue
        }
      ]
    }
  ],
  "nextToken": "string"
}
```

AttributePropertyValueReaderByEntity

AttributePropertyValueReaderByEntity 是一个数据平面连接器，可用于获取单个实体中静态属性的值。

有关此连接器的属性类型、语法和格式的信息，请参阅 GetPropertyValueAPI 操作。

AttributePropertyValueReaderByEntity 请求接口

```
{
  "properties": {
    // property name as key,
    // value is of type PropertyResponse
    "string": "PropertyResponse"
  }

  "workspaceId": "string",
  "entityId": "string",
  "componentName": "string",
  "selectedProperties": List:"string",
}
```

AttributePropertyValueReaderByEntity 响应接口

```
{
  "propertyValues": {
    "string": { // property name as key
        "propertyReference": EntityPropertyReference, // The same
 as EntityPropertyReference
        "propertyValue": DataValue // The same as DataValue
    }
}
```

DataWriter

DataWriter 是一个数据平面连接器，可用于将时间序列数据点写回底层数据存储中，以获取单个组件中的属性。

有关此连接器的属性类型、语法和格式的信息，请参阅 BatchPutPropertyValuesAPI 操作。

DataWriter 请求接口

```
{
  "workspaceId": "string",
  "properties": {
    // entity id as key
    "String": {
      // property name as key,
      // value is of type PropertyResponse
```

```
      "string": PropertyResponse
    }
  },
  "entries": [
    {
      "entryId": "string",
      "entityPropertyReference": EntityPropertyReference, // The same
 as EntityPropertyReference
      "propertyValues": [
        {
        "timestamp": long, // Epoch sec, deprecated
        "time": "string", // ISO-8601 timestamp format
        "value": DataValue // The same as DataValue
        }
      ]
    }
  ]
}
```

DataWriter 响应接口

```
{
  "errorEntries": [
    {
      "errors": List:BatchPutPropertyError // The value is a list of
 type BatchPutPropertyError
    }
  ]
}
```

示例

以下 JSON 示例为多个连接器的响应和请求语法示例。

- SchemaInitializer:

  以下示例所示为组件类型使用周期中的架构初始值设定项。

  请求:

  ```
  {
    "workspaceId": "myWorkspace",
  ```

```
  "properties": {
    "modelId": {
      "definition": {
          "dataType": { "type": "STRING" },
          "isExternalId": true,
          "isFinal": true,
          "isImported": false,
          "isInherited": false,
          "isRequiredInEntity": true,
          "isStoredExternally": false,
          "isTimeSeries": false,
          "defaultValue": {
              "stringValue": "myModelId"
          }
      },
      "value": {
          "stringValue": "myModelId"
      }
    },
    "tableName": {
      "definition": {
          "dataType": { "type": "STRING" },
          "isExternalId": false,
          "isFinal": false,
          "isImported": false,
          "isInherited": false,
          "isRequiredInEntity": false,
          "isStoredExternally": false,
          "isTimeSeries": false,
          "defaultValue": {
              "stringValue": "myTableName"
          }
      },
      "value": {
          "stringValue": "myTableName"
      }
    }
  }
}
```

响应：

```
{
```

```
  "properties": {
    "myProperty1": {
      "definition": {
        "dataType": {
          "type": "DOUBLE",
          "unitOfMeasure": "%"
        },
        "configuration": {
          "myProperty1Id": "idValue"
        },
        "isTimeSeries": true
      }
    },
    "myProperty2": {
      "definition": {
        "dataType": { "type": "STRING" },
        "isTimeSeries": false,
        "defaultValue": {
          "stringValue": "property2Value"
        }
      }
    }
  }
}
```

- 实体使用周期中的架构初始值设定项：

  请求:

```
{
  "workspaceId": "myWorkspace",
  "entityId": "myEntity",
  "componentName": "myComponent",
  "properties": {
    "assetId": {
      "definition": {
          "dataType": { "type": "STRING" },
          "isExternalId": true,
          "isFinal": true,
          "isImported": false,
          "isInherited": false,
          "isRequiredInEntity": true,
          "isStoredExternally": false,
          "isTimeSeries": false
```

```
      },
      "value": {
          "stringValue": "myAssetId"
      }
    },
    "tableName": {
      "definition": {
          "dataType": { "type": "STRING" },
          "isExternalId": false,
          "isFinal": false,
          "isImported": false,
          "isInherited": false,
          "isRequiredInEntity": false,
          "isStoredExternally": false,
          "isTimeSeries": false
      },
      "value": {
          "stringValue": "myTableName"
      }
    }
  }
}
```

响应：

```
{
  "properties": {
    "myProperty1": {
      "definition": {
        "dataType": {
          "type": "DOUBLE",
          "unitOfMeasure": "%"
        },
        "configuration": {
          "myProperty1Id": "idValue"
        },
        "isTimeSeries": true
      }
    },
    "myProperty2": {
      "definition": {
        "dataType": { "type": "STRING" },
        "isTimeSeries": false
```

```
      },
      "value": {
        "stringValue": "property2Value"
      }
    }
  }
}
```

- DataReaderByEntity 和 DataReader：

  请求:

```
{
  "workspaceId": "myWorkspace",
  "entityId": "myEntity",
  "componentName": "myComponent",
  "selectedProperties": [
    "Temperature",
    "Pressure"
  ],
  "startTime": "2022-04-07T04:04:42Z",
  "endTime": "2022-04-07T04:04:45Z",
  "maxResults": 4,
  "orderByTime": "ASCENDING",
  "properties": {
      "assetId": {
          "definition": {
              "dataType": { "type": "STRING" },
              "isExternalId": true,
              "isFinal": true,
              "isImported": false,
              "isInherited": false,
              "isRequiredInEntity": true,
              "isStoredExternally": false,
              "isTimeSeries": false
          },
          "value": {
              "stringValue": "myAssetId"
          }
      },
      "Temperature": {
          "definition": {
              "configuration": {
                  "temperatureId": "xyz123"
```

```
            },
            "dataType": {
                "type": "DOUBLE",
                "unitOfMeasure": "DEGC"
            },
            "isExternalId": false,
            "isFinal": false,
            "isImported": true,
            "isInherited": false,
            "isRequiredInEntity": false,
            "isStoredExternally": false,
            "isTimeSeries": true
        }
    },
    "Pressure": {
        "definition": {
            "configuration": {
                "pressureId": "xyz456"
            },
            "dataType": {
                "type": "DOUBLE",
                "unitOfMeasure": "MPA"
            },
            "isExternalId": false,
            "isFinal": false,
            "isImported": true,
            "isInherited": false,
            "isRequiredInEntity": false,
            "isStoredExternally": false,
            "isTimeSeries": true
        }
    }
  }
}
```

响应：

```
{
  "propertyValues": [
    {
      "entityPropertyReference": {
        "entityId": "myEntity",
        "componentName": "myComponent",
```

```
        "propertyName": "Temperature"
      },
      "values": [
        {
          "time": "2022-04-07T04:04:42Z",
          "value": {
            "doubleValue": 588.168
          }
        },
        {
          "time": "2022-04-07T04:04:43Z",
          "value": {
            "doubleValue": 592.4224
          }
        }
      ]
    }
  ],
  "nextToken": "qwertyuiop"
}
```

- AttributePropertyValueReaderByEntity:

  请求:

```
{
  "workspaceId": "myWorkspace",
  "entityId": "myEntity",
  "componentName": "myComponent",
  "selectedProperties": [
    "manufacturer",
  ],
  "properties": {
    "assetId": {
      "definition": {
        "dataType": { "type": "STRING" },
        "isExternalId": true,
        "isFinal": true,
        "isImported": false,
        "isInherited": false,
        "isRequiredInEntity": true,
        "isStoredExternally": false,
        "isTimeSeries": false
      },
```

```
      "value": {
          "stringValue": "myAssetId"
      }
    },
    "manufacturer": {
      "definition": {
        "dataType": { "type": "STRING" },
        "configuration": {
            "manufacturerPropId": "M001"
        },
        "isExternalId": false,
        "isFinal": false,
        "isImported": false,
        "isInherited": false,
        "isRequiredInEntity": false,
        "isStoredExternally": true,
        "isTimeSeries": false
      }
    }
  }
}
```

响应：

```
{
  "propertyValues": {
    "manufacturer": {
      "propertyReference": {
        "propertyName": "manufacturer",
        "entityId": "myEntity",
        "componentName": "myComponent"
      },
      "propertyValue": {
        "stringValue": "Amazon"
      }
    }
  }
}
```

- DataWriter:

  请求:

```
{
  "workspaceId": "myWorkspaceId",
  "properties": {
    "myEntity": {
      "Temperature": {
          "definition": {
              "configuration": {
                  "temperatureId": "xyz123"
              },
              "dataType": {
                  "type": "DOUBLE",
                  "unitOfMeasure": "DEGC"
              },
              "isExternalId": false,
              "isFinal": false,
              "isImported": true,
              "isInherited": false,
              "isRequiredInEntity": false,
              "isStoredExternally": false,
              "isTimeSeries": true
          }
      }
    }
  },
  "entries": [
    {
      "entryId": "myEntity",
      "entityPropertyReference": {
        "entityId": "myEntity",
        "componentName": "myComponent",
        "propertyName": "Temperature"
      },
      "propertyValues": [
        {
          "timestamp": 1626201120,
          "value": {
            "doubleValue": 95.6958
          }
        },
        {
          "timestamp": 1626201132,
          "value": {
            "doubleValue": 80.6959
```

```
          }
        }
      ]
    }
  ]
}
```

响应：

```
{
    "errorEntries": [
        {
            "errors": [
                {
                    "errorCode": "409",
                    "errorMessage": "Conflict value at same timestamp",
                    "entry": {
                        "entryId": "myEntity",
                        "entityPropertyReference": {
                            "entityId": "myEntity",
                            "componentName": "myComponent",
                            "propertyName": "Temperature"
                        },
                        "propertyValues": [
                            "time": "2022-04-07T04:04:42Z",
                            "value": {
                                "doubleValue": 95.6958
                            }
                        ]
                    }
                }
            ]
        }
    ]
}
```

AWS IoT TwinMaker Athena 表格数据连接器

您可通过 Athena 表格数据连接器访问和使用 AWS IoT TwinMaker中存储的 Athena 数据。您可以使用 Athena 数据构建数字孪生，而无需大量迁移数据。您可以使用预先构建的连接器或创建自定义 Athena 连接器来访问来自 Athena 数据源的数据。

AWS IoT TwinMaker Athena 数据连接器先决条件

在使用 Athena 表格数据连接器之前，请完成以下先决条件：

- 创建托管 Athena 表及其关联的 Amazon S3 资源。有关使用 Athena 的信息，请参阅Athena 文档。
- 创建 AWS IoT TwinMaker 工作空间。您可以在 AWS IoT TwinMaker 控制台中创建工作区。
- 使用 Athena 权限更新您的工作空间 IAM 角色。有关更多信息，请参阅 修改工作区 IAM 角色以使用 Athena 数据连接器。
- 熟悉 AWS IoT TwinMaker的实体组件系统以及如何创建实体。有关更多信息，请参阅 创建您的第一个实体。
- 熟悉 AWS IoT TwinMaker的数据连接器。有关更多信息，请参阅 AWS IoT TwinMaker 数据连接器。

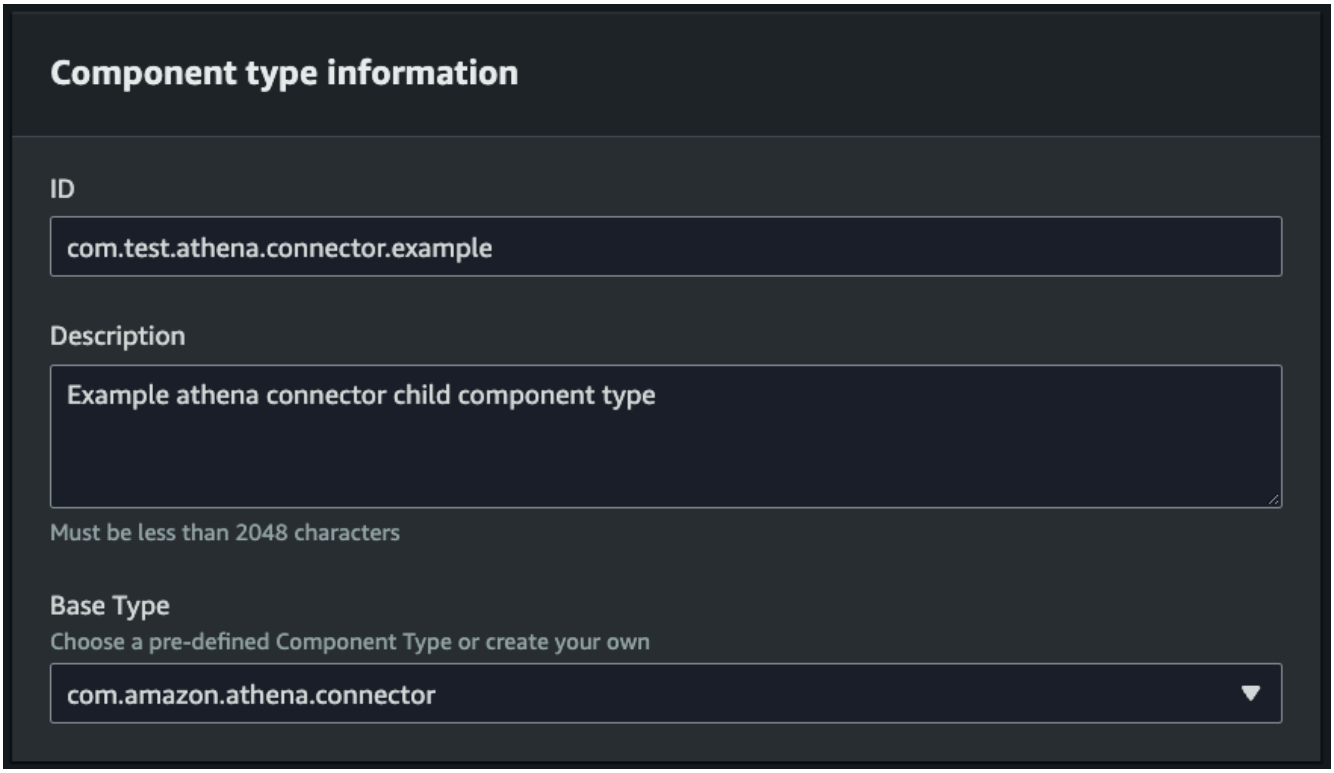
使用 Athena 数据连接器

若要使用 Athena 数据连接器，必须采用 Athena 连接器创建组件。然后将组件连接至 AWS IoT TwinMaker场景内使用的实体。

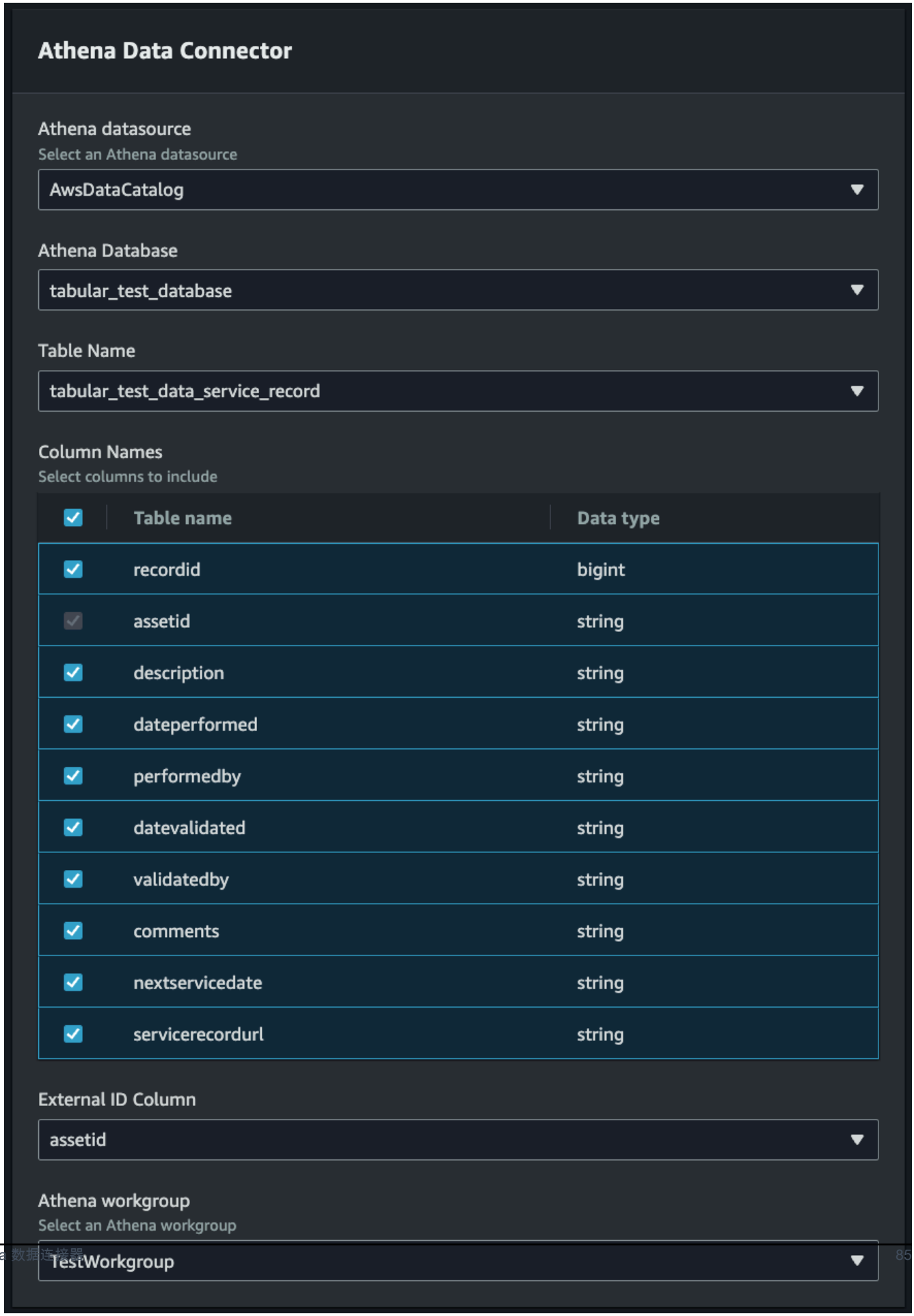
通过 Athena 数据连接器创建组件类型

使用以下步骤使用 Athena 表格数据连接器创建 AWS IoT TwinMaker 组件类型：

1. 导航到 AWS IoT TwinMaker 控制台。
2. 打开现有工作区或创建新工作区。
3. 在左侧导航菜单中选择 组件类型，然后选择创建组件类型以打开组件类型创建页面。
4. 在 创建组件类型 页面，填写匹配您用例的 ID。



5. 选择 基类型。从下拉列表中，选择标记为com.amazon.athena.connector 的 Athena 表格数据连接器。
6. 通过为以下字段选择 Athena 资源，以配置组件类型数据来源：
   - 选择 Athena 数据源。
   - 选择 Athena 数据库。
   - 选择 表格名称。
   - 选择 Athena 工作组。
7. 选择要用作数据来源的 Athena 资源后，从表格中选择要包含的列。
8. 选择外部 ID 列名称。从上一步中选择一列，作为外部 ID 列。外部 ID 是用于表示 Athena 资产并将其映射到实体的 ID。 AWS IoT TwinMaker


Athena Data Connector
Athena datasource
Select an Athena datasource
AwsDataCatalog
Athena Database
tabular_test_database
Table Name
tabular_test_data_service_record
Column Names
Select columns to include
Table name
Data type
recordid
bigint
assetid
string
description
string
dateperformed
string
performedby
string
datevalidated
string
validatedby
string
comments
string
nextservicedate
string
serviceRecordUrl
string
External ID Column
assetid
Athena workgroup
Select an Athena workgroup
TestWorkgroup

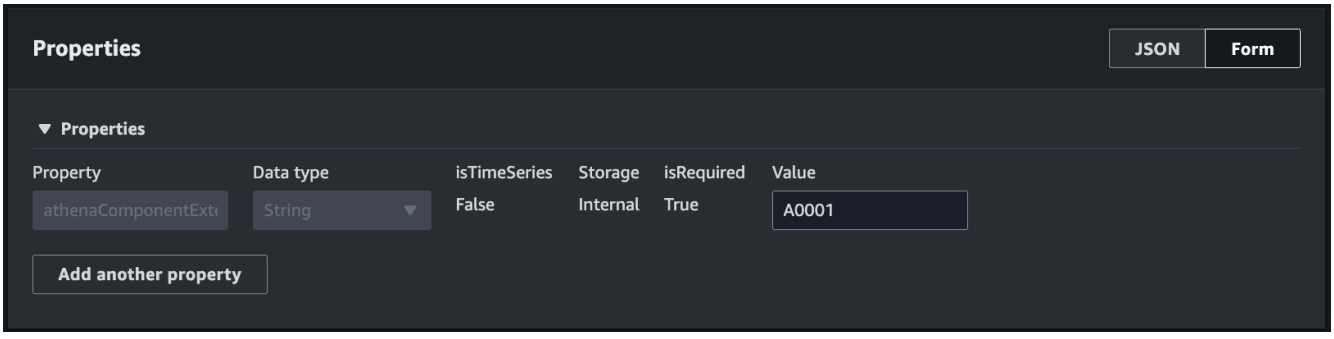
9. （可选）为这些资源添加 AWS 标签，以便您可以对它们进行分组和组织。
10. 选择 创建组件类型 以完成创建组件类型。

创建具有 Athena 数据连接器类型的组件，并将其附加至实体

使用以下步骤创建带有 Athena 表格数据连接器的 AWS IoT TwinMaker 组件并将其连接到实体：

> **Note**
>
> 若要完成此程序，您的现有组件类型必须将 Athena 表格数据连接器作为数据来源。查看前述程序开始本操作前，通过 Athena 数据连接器创建组件类型。

1. 导航到 AWS IoT TwinMaker 控制台。
2. 打开现有工作区或创建新工作区。
3. 从左侧导航菜单中选择 “实体”，然后选择要向其添加组件或创建新实体的实体。
4. 创建一个新实体。
5. 接下来选择添加组件，在组件名称字段中填写匹配您用例的名称。
6. 在 组件类型下拉菜单中，选择您在前述程序中创建的 组件类型 ID。
7. 输入组件信息、组件名称，然后选择之前 ComponentType 创建的子项。这是 ComponentType 您使用 Athena 数据连接器创建的。
8. 在 “属性” 部分中，输入组件的 athenaComponentExternalID。



9. 选择 添加组件以完成组件创建。

现在您已成功创建了带 Athena 数据连接器的组件，将其作为适当的组件类型，并连接至实体。

使用 Athena 表格数据连接器 JSON 参考资料

以下是 Athena 表格数据连接器的完整 JSON 参考资料示例。将其用作创建自定义数据连接器以及组件类型的资源。

```
{
    "componentTypeId": "com.amazon.athena.connector",
    "description": "Athena connector for syncing tabular data",
    "workspaceId":"AmazonOwnedTypesWorkspace",
    "propertyGroups": {
        "tabularPropertyGroup": {
            "groupType": "TABULAR",
            "propertyNames": []
        }
    },
    "propertyDefinitions": {
        "athenaDataSource": {
            "dataType": { "type": "STRING" },
            "isRequiredInEntity": true
        },
        "athenaDatabase": {
            "dataType": { "type": "STRING" },
            "isRequiredInEntity": true
        },
        "athenaTable": {
            "dataType": { "type": "STRING" },
            "isRequiredInEntity": true
        },
        "athenaWorkgroup": {
            "dataType": { "type": "STRING" },
            "isRequiredInEntity": true
        },
        "athenaExternalIdColumnName": {
            "dataType": { "type": "STRING" },
            "isRequiredInEntity": true,
            "isExternalId": false
        },
        "athenaComponentExternalId": {
            "dataType": { "type": "STRING" },
            "isStoredExternally": false,
            "isRequiredInEntity": true,
            "isExternalId": true
        }
```

```
    },
    "functions": {
        "tabularDataReaderByEntity": {
            "implementedBy": {
                "isNative": true
            }
        }
    }
}
```

使用 Athena 数据连接器

您可以在 Grafana 中显示使用 Athena 表的实体。有关更多信息，请参阅 AWS IoT TwinMaker Grafana 控制面板集成。

有关创建和使用 Athena 表存储数据的信息，请阅读 Athena 文档。

排除 Athena 数据连接器故障

本主题介绍您在配置 Athena 数据连接器时可能遇到的常见问题。

Athena 工作组位置：

当创建 Athena 连接器 ComponentType 时，Athena 工作组必须设置输出位置。查看 工作组的工作原理。

IAM 角色权限缺失：

在创建 ComponentType、向实体添加 Ca 组件或运行 API 时， AWS IoT TwinMaker; workspace 角色可能缺少 Athena API 访问权限。 GetPropertyValue 要更新 IAM 权限，请参阅创建和管理的服务角色 AWS IoT TwinMaker。

查看 Grafana 中的 Athena 表格数据

Grafana插件还可用于在Grafana上可视化您的表格数据，Grafana是一个仪表板面板，具有其他功能，例如根据所选属性进行排序和筛选，而无需对Athena进行API调用或与Athena进行交互。 AWS IoT TwinMaker本主题向您介绍如何配置 Grafana 以查看 Athena 表格数据。

先决条件

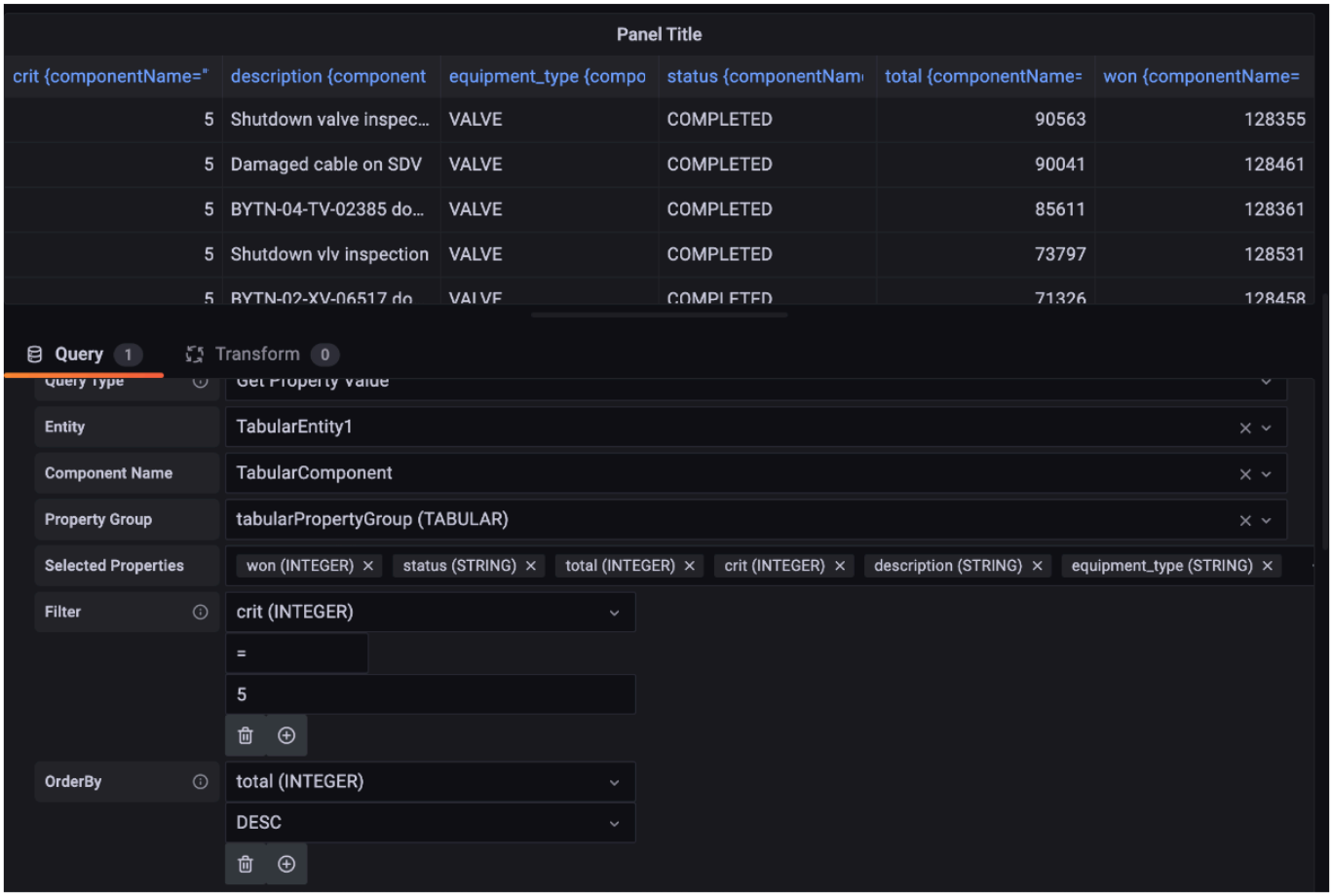
在配置 Grafana 平面以可查看 Athena 表格数据之前，请查看以下先决条件：

- 您已经设置了 Grafana 环境。有关更多信息，请参阅 AWS IoT TwinMaker Grafana 集成。
- 您可配置 Grafana 数据来源。欲了解更多信息，请参阅 Grafana AWS IoT TwinMaker。
- 您熟悉如何创建新控制面板和添加新面板。

查看 Grafana 中的 Athena 表格数据

此过程向您展示如何设置 Grafana 面板以查看 Athena 表格数据。

1. 打开你的 AWS IoT TwinMaker Grafana 控制面板。
2. 在面板设置中选择 表格面板。
3. 在查询配置中选择数据来源。
4. 选择 Get Property Value（获取属性值）查询。
5. 选择一个实体。
6. 选择包含扩展 Athena 基本组件类型的ComponentType 组件。
7. 选择 Athena 表格的属性群组。
8. 从属群组中选择任意数量的属性。
9. 通过筛选条件和属性顺序列表，配置表格条件。使用以下选项：
    - 筛选条件：定义数据筛选属性值的表达式。
    - OrderBy：指定应按升序还是降序返回属性的数据。



开发 AWS IoT TwinMaker 时间序列数据连接器

本节介绍如何在 step-by-step流程中开发时间序列数据连接器。此外，我们还提供了基于整个 cookie 出厂示例的时间序列数据连接器示例，其中包括 3D 模型、实体、组件、警报和连接器。Cookie 工厂示例源可在AWS IoT TwinMaker 示例 GitHub 存储库中找到。

主题

- AWS IoT TwinMaker 时间序列数据连接器先决条件
- 时间序列数据连接器背景
- 开发时间序列数据连接器
- 改进您的数据连接器
- 测试您的连接器
- 安全性

- 创建 AWS IoT TwinMaker 资源
- 接下来做什么
- AWS IoT TwinMaker cookie 出厂示例时间序列连接器

AWS IoT TwinMaker 时间序列数据连接器先决条件

开发时间序列数据连接器之前，建议您完成以下任务：

- 创建 AWS IoT TwinMaker 工作区。
- 创建 AWS IoT TwinMaker 组件类型。
- 创建 AWS IoT TwinMaker 实体。
- （可选）阅读 使用和创建组件类型。
- （可选）阅读AWS IoT TwinMaker 数据连接器接口以大致了解 AWS IoT TwinMaker 数据连接器。

> **Note**
>
> 有关完全实施的连接器示例，请参阅我们的 cookie 出厂示例实现介绍。

时间序列数据连接器背景

想象一下，您正在与一家工厂合作，那里有一套曲奇搅拌机和一个水箱。您想构建这些物理实体的 AWS IoT TwinMaker 数字双胞胎，以便您可以通过检查各种时间序列指标来监控它们的运行状态。

您已经在现场设置了传感器，并且已经在将测量数据流式传输至 Timestream 数据库。您希望能够以最小的成本查看和整理 AWS IoT TwinMaker 中的测量数据。您可以使用时间序列数据连接器完成此任务。下图所示为遥测表示例，该表通过时间序列连接器填充。

Rows returned (1000+)

Results are paginated. Scroll through the result pages to see more query results.

Filter

1 2 3 4 5 6 7 ... 100

TelemetryAssetId	TelemetryAssetType	measure_name	time	measure_value::varchar	measure_value::double
Mixer_22_680b5b8e-1afe-4a77-87ab-834fbe5ba01e	Mixer	Temperature	2022-04-19 00:28:00.241000000	-	99.1292877197266
Mixer_20_0568f25f-116c-429c-a974-5ceec065a6ac	Mixer	RPM	2022-04-19 00:28:00.241000000	-	59.4233207702637
Mixer_22_680b5b8e-1afe-4a77-87ab-834fbe5ba01e	Mixer	RPM	2022-04-19 00:28:00.241000000	-	59.9421195983887
Mixer_24_7ff0b75b-f0fa-43f0-bc89-b96337586d00	Mixer	Temperature	2022-04-19 00:28:00.241000000	-	99.1292877197266
Mixer_25_cf42effc-ba19-48ba-bbc3-d21d2508ce31	Mixer	RPM	2022-04-19 00:28:00.241000000	-	59.8453979492188
Mixer_20_0568f25f-116c-429c-a974-5ceec065a6ac	Mixer	Temperature	2022-04-19 00:28:00.241000000	-	99.1292877197266
Mixer_24_7ff0b75b-f0fa-43f0-bc89-b96337586d00	Mixer	RPM	2022-04-19 00:28:00.241000000	-	60.4532585144043
Mixer_15_0bb566cd-d6f3-4804-9fe1-7d2abcad82d0	Mixer	RPM	2022-04-19 00:28:00.241000000	-	58.397144317627
Mixer_2_d8e76844-e739-4845-a748-a83983279376	Mixer	RPM	2022-04-19 00:28:00.241000000	-	60.206958770752
Mixer_6_b66db3d3-c144-47b5-afb9-3a0150c53456	Mixer	RPM	2022-04-19 00:28:00.241000000	-	60.206958770752

此屏幕截图中使用的数据集和 Timestream 表可在AWS IoT TwinMaker 示例 GitHub 存储库中找到。另请参阅已经实施的 cookie 出厂示例连接器，其结果如前面的屏幕截图所示。

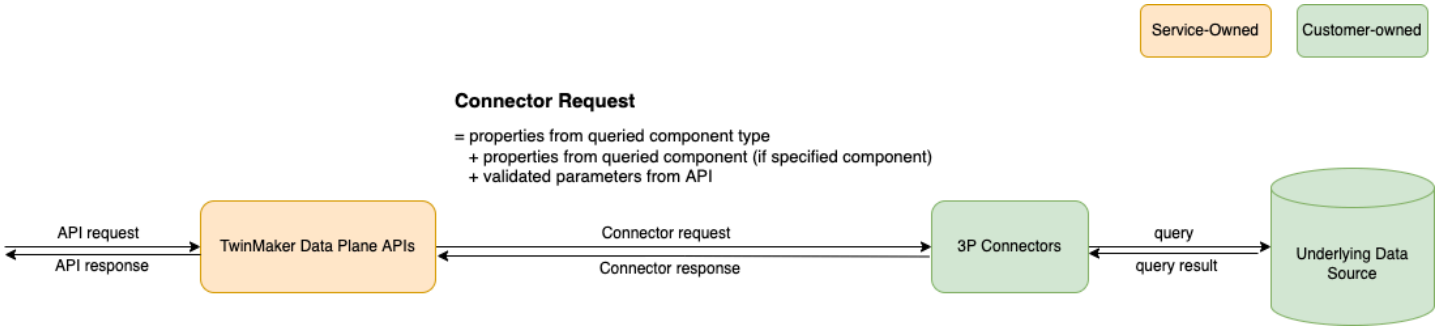
时间序列数据连接器数据流

对于数据平面查询，从组件和组件类型定义中 AWS IoT TwinMaker 获取组件和组件类型的相应属性。AWS IoT TwinMaker 将属性与查询中的任何 API 查询参数一起转发给 AWS Lambda 函数。

AWS IoT TwinMaker 使用 Lambda 函数访问和解析来自数据源的查询，并返回这些查询的结果。Lambda 函数使用数据平面中的组件和组件类型属性解析初始请求。

Lambda 查询的结果将映射至 API 响应并返回给您。

AWS IoT TwinMaker 定义数据连接器接口，并使用该接口与 Lambda 函数进行交互。您可通过数据连接器，从 AWS IoT TwinMaker API 查询数据来源，而无需进行任何数据迁移。下图概述了前几段描述的基本数据流。



开发时间序列数据连接器

以下程序概述了一种开发模型，该模型以增量方式构建功能性时间序列数据连接器。基本步骤如下所示：

1. 创建有效的基础组件类型

   在组件类型中，您可以定义在组件间共享的通用属性。要了解有关组件类型定义的更多信息，请参阅使用和创建组件类型。

   AWS IoT TwinMaker 使用实体-组件建模模式，因此每个组件都连接到一个实体。我们建议您将每个物理项目建模为一个实体，并使用不同的组件类型对不同的数据源进行建模。

   以下示例介绍了带一项目属性的 Timestream 模板组件类型：

```
{"componentTypeId": "com.example.timestream-telemetry",
    "workspaceId": "MyWorkspace",
```

```
    "functions": {
        "dataReader": {
            "implementedBy": {
                "lambda": {
                    "arn": "lambdaArn"
                }
            }
        }
    },
    "propertyDefinitions": {
        "telemetryType": {
            "dataType": { "type": "STRING" },
            "isExternalId": false,
            "isStoredExternally": false,
            "isTimeSeries": false,
            "isRequiredInEntity": true
        },
        "telemetryId": {
            "dataType": { "type": "STRING" },
            "isExternalId": true,
            "isStoredExternally": false,
            "isTimeSeries": false,
            "isRequiredInEntity": true
        },
        "Temperature": {
            "dataType": { "type": "DOUBLE" },
            "isExternalId": false,
            "isTimeSeries": true,
            "isStoredExternally": true,
            "isRequiredInEntity": false
        }
    }
}
```

此组件类型的关键元素如下：

- 该telemetryId属性标识相应数据源中物理项目的唯一密钥。数据连接器使用此属性作为筛选条件，仅查询与给定项目关联的值。此外，如果您将 telemetryId 属性值纳入数据平面 API 响应，则客户端会获取该 ID，必要时可以执行反向查找。
- 该 lambdaArn 字段标识组件类型所用的 Lambda 函数。
- 该 isRequiredInEntity 标志强制创建 ID。这个标志是必需的，这样在创建组件时，项目的 ID 也会被实例化。

- 将TelemetryId作为外部 ID 添加到组件类型中，以便可以在 Timestream 表中识别该项目。

2. 按组件类型创建组件

   要使用您创建的组件类型，必须创建组件并将其附加至数据检索目标实体。以下步骤详细介绍了此组件创建流程：

   a. 导航到 AWS IoT TwinMaker 控制台。
   b. 选择并打开您在其中创建组件类型的同一个工作区。
   c. 导航至实体页面。
   d. 创建新实体，或从表格中选择现有实体。
   e. 选择要使用的实体后，选择 添加组件以打开添加组件页面。
   f. 创建组件名称，对于组件类型，选择您要通过模板创建的组件类型。创建有效的基本组件类型。

3. 通过组建类型调用 Lambda 连接器

   Lambda 连接器需要访问数据来源，并根据输入生成查询语句并将其转发至数据来源。以下示例显示了执行此操作的 JSON 请求模板。

```
{
  "workspaceId": "MyWorkspace",
  "entityId": "MyEntity",
  "componentName": "TelemetryData",
  "selectedProperties": ["Temperature"],
  "startTime": "2022-08-25T00:00:00Z",
  "endTime": "2022-08-25T00:00:05Z",
  "maxResults": 3,
  "orderByTime": "ASCENDING",
  "properties": {
      "telemetryType": {
          "definition": {
              "dataType": { "type": "STRING" },
              "isExternalId": false,
              "isFinal": false,
              "isImported": false,
              "isInherited": false,
              "isRequiredInEntity": false,
              "isStoredExternally": false,
              "isTimeSeries": false
          },
          "value": {
```

```
                "stringValue": "Mixer"
            }
        },
        "telemetryId": {
            "definition": {
                "dataType": { "type": "STRING" },
                "isExternalId": true,
                "isFinal": true,
                "isImported": false,
                "isInherited": false,
                "isRequiredInEntity": true,
                "isStoredExternally": false,
                "isTimeSeries": false
            },
            "value": {
                "stringValue": "item_A001"
            }
        },
        "Temperature": {
            "definition": {
                "dataType": { "type": "DOUBLE", },
                "isExternalId": false,
                "isFinal": false,
                "isImported": true,
                "isInherited": false,
                "isRequiredInEntity": false,
                "isStoredExternally": false,
                "isTimeSeries": true
            }
        }
    }
}
```

请求关键要素：

- selectedProperties 是一种列表，您可填充想要 Timestream 测量的属性。
- startDateTime、startTime、EndDateTime 和 endTime 字段指定请求的时间范围。这决定了返回测量值的样本范围。
- entityId 是您要从中查询数据的实体名称。
- componentName 是您要从中查询数据的组件名称。
- 使用 orderByTime 字段来整理结果的显示顺序。

在前面的示例请求中，我们希望在给定时间窗口内按所选时间顺序获得给定项目的一系列样本。响应语句可概括如下：

```
{
  "propertyValues": [
    {
      "entityPropertyReference": {
        "entityId": "MyEntity",
        "componentName": "TelemetryData",
        "propertyName": "Temperature"
      },
      "values": [
        {
          "time": "2022-08-25T00:00:00Z",
          "value": {
            "doubleValue": 588.168
          }
        },
        {
          "time": "2022-08-25T00:00:01Z",
          "value": {
            "doubleValue": 592.4224
          }
        },
        {
          "time": "2022-08-25T00:00:02Z",
          "value": {
            "doubleValue": 594.9383
          }
        }
      ]
    }
  ],
  "nextToken": "..."
}
```

4. 将您的组件类型，使其包含两种属性

以下 JSON 模板显示了包含两个属性的有效组件类型：

```
{
```

```
    "componentTypeId": "com.example.timestream-telemetry",
    "workspaceId": "MyWorkspace",
    "functions": {
        "dataReader": {
            "implementedBy": {
                "lambda": {
                    "arn": "lambdaArn"
                }
            }
        }
    },
    "propertyDefinitions": {
        "telemetryType": {
            "dataType": { "type": "STRING" },
            "isExternalId": false,
            "isStoredExternally": false,
            "isTimeSeries": false,
            "isRequiredInEntity": true
        },
        "telemetryId": {
            "dataType": { "type": "STRING" },
            "isExternalId": true,
            "isStoredExternally": false,
            "isTimeSeries": false,
            "isRequiredInEntity": true
        },
        "Temperature": {
            "dataType": { "type": "DOUBLE" },
            "isExternalId": false,
            "isTimeSeries": true,
            "isStoredExternally": true,
            "isRequiredInEntity": false
        },
        "RPM": {
            "dataType": { "type": "DOUBLE" },
            "isExternalId": false,
            "isTimeSeries": true,
            "isStoredExternally": true,
            "isRequiredInEntity": false
        }
    }
}
```

5. 更新 Lambda 连接器，以处理第二属性

AWS IoT TwinMaker 数据平面 API 支持在单个请求中查询多个属性，并通过提供列表来 AWS IoT TwinMaker 跟踪对连接器的单个请求`selectedProperties`。

以下 JSON 请求所示为修改后的模板，该模板现在支持对两个属性的请求。

```
{
  "workspaceId": "MyWorkspace",
  "entityId": "MyEntity",
  "componentName": "TelemetryData",
  "selectedProperties": ["Temperature", "RPM"],
  "startTime": "2022-08-25T00:00:00Z",
  "endTime": "2022-08-25T00:00:05Z",
  "maxResults": 3,
  "orderByTime": "ASCENDING",
  "properties": {
      "telemetryType": {
          "definition": {
              "dataType": { "type": "STRING" },
              "isExternalId": false,
              "isFinal": false,
              "isImported": false,
              "isInherited": false,
              "isRequiredInEntity": false,
              "isStoredExternally": false,
              "isTimeSeries": false
          },
          "value": {
              "stringValue": "Mixer"
          }
      },
      "telemetryId": {
          "definition": {
              "dataType": { "type": "STRING" },
              "isExternalId": true,
              "isFinal": true,
              "isImported": false,
              "isInherited": false,
              "isRequiredInEntity": true,
              "isStoredExternally": false,
              "isTimeSeries": false
          },
          "value": {
```

```
                "stringValue": "item_A001"
            }
        },
        "Temperature": {
            "definition": {
                "dataType": { "type": "DOUBLE" },
                "isExternalId": false,
                "isFinal": false,
                "isImported": true,
                "isInherited": false,
                "isRequiredInEntity": false,
                "isStoredExternally": false,
                "isTimeSeries": true
            }
        },
        "RPM": {
            "definition": {
                "dataType": { "type": "DOUBLE" },
                "isExternalId": false,
                "isFinal": false,
                "isImported": true,
                "isInherited": false,
                "isRequiredInEntity": false,
                "isStoredExternally": false,
                "isTimeSeries": true
            }
        }
    }
}
```

同样，还按下例更新相应的响应：

```
{
  "propertyValues": [
    {
      "entityPropertyReference": {
        "entityId": "MyEntity",
        "componentName": "TelemetryData",
        "propertyName": "Temperature"
      },
      "values": [
        {
          "time": "2022-08-25T00:00:00Z",
```

```
          "value": {
            "doubleValue": 588.168
          }
        },
        {
          "time": "2022-08-25T00:00:01Z",
          "value": {
            "doubleValue": 592.4224
          }
        },
        {
          "time": "2022-08-25T00:00:02Z",
          "value": {
            "doubleValue": 594.9383
          }
        }
      ]
    },
    {
      "entityPropertyReference": {
        "entityId": "MyEntity",
        "componentName": "TelemetryData",
        "propertyName": "RPM"
      },
      "values": [
        {
          "time": "2022-08-25T00:00:00Z",
          "value": {
            "doubleValue": 59
          }
        },
        {
          "time": "2022-08-25T00:00:01Z",
          "value": {
            "doubleValue": 60
          }
        },
        {
          "time": "2022-08-25T00:00:02Z",
          "value": {
            "doubleValue": 60
          }
        }
      ]
```

```
    }
  ],
  "nextToken": "..."
}
```

> **Note**
>
> 就本例分页而言，请求中的页面大小适用于所有属性。这意味着查询中有五个属性，页面大小为 100，如果源中有足够的数据点，则每个属性应有 100 个数据点，总共有 500 个数据点。

有关实现示例，请参阅上的 Snowflake 连接器示例。 GitHub

改进您的数据连接器

处理异常

Lambda 连接器可以安全抛出异常。在数据层面 API 调用中， AWS IoT TwinMaker 服务等待 Lambda 函数返回响应。如果连接器实现引发异常，则将异常类型 AWS IoT TwinMaker 转换为`ConnectorFailure`，让 API 客户端意识到连接器内部发生了问题。

处理分页

在示例中，Timestream 提供了实用函数，可以帮助本地支持分页。但是，对于其他部分查询接口（例如 SQL），可能需要额外操作才能实现高效分页算法。Snowflake 连接器示例可以处理 SQL 接口中的分页。

当 AWS IoT TwinMaker 通过连接器响应接口返回新令牌时，该令牌在返回到 API 客户端之前会被加密。当令牌包含在另一个请求中时，先对其进行 AWS IoT TwinMaker 解密，然后再将其转发到 Lambda 连接器。我们建议您避免向令牌中添加敏感信息。

测试您的连接器

尽管在将连接器连接至组件类型之后，您仍然可以更新实施，但我们强烈建议您在集成 AWS IoT TwinMaker之前验证 Lambda 连接器。

有多种方法可以测试您的 Lambda 连接器：您可以在 Lambda 控制台中测试 Lambda 连接器，也可以在 AWS CDK中本地测试。

有关测试 Lambda 函数的更多信息，请参阅测试 Lambda 函数和本地测试应用程序。 AWS CDK

安全性

有关 Timestream 最佳安全方法文档，请参阅 Timestream 安全措施。

有关 SQL 注入防护的示例，请参阅 AWS IoT TwinMaker 示例 GitHub 存储库中的以下 Python 脚本。

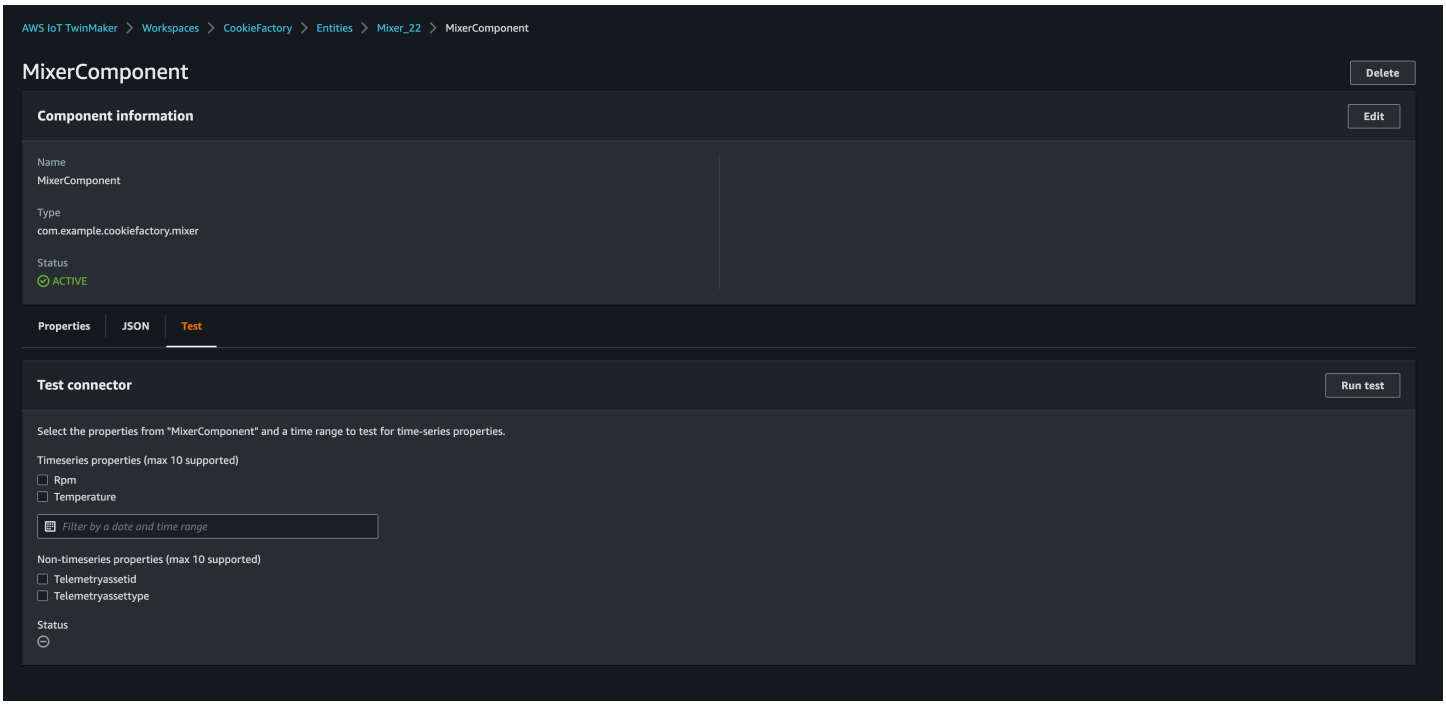
创建 AWS IoT TwinMaker 资源

实现 Lambda 函数后，您可以通过AWS IoT TwinMaker 控制台或 AWS IoT TwinMaker API 创建诸如组件类型、实体和组件之类的资源。

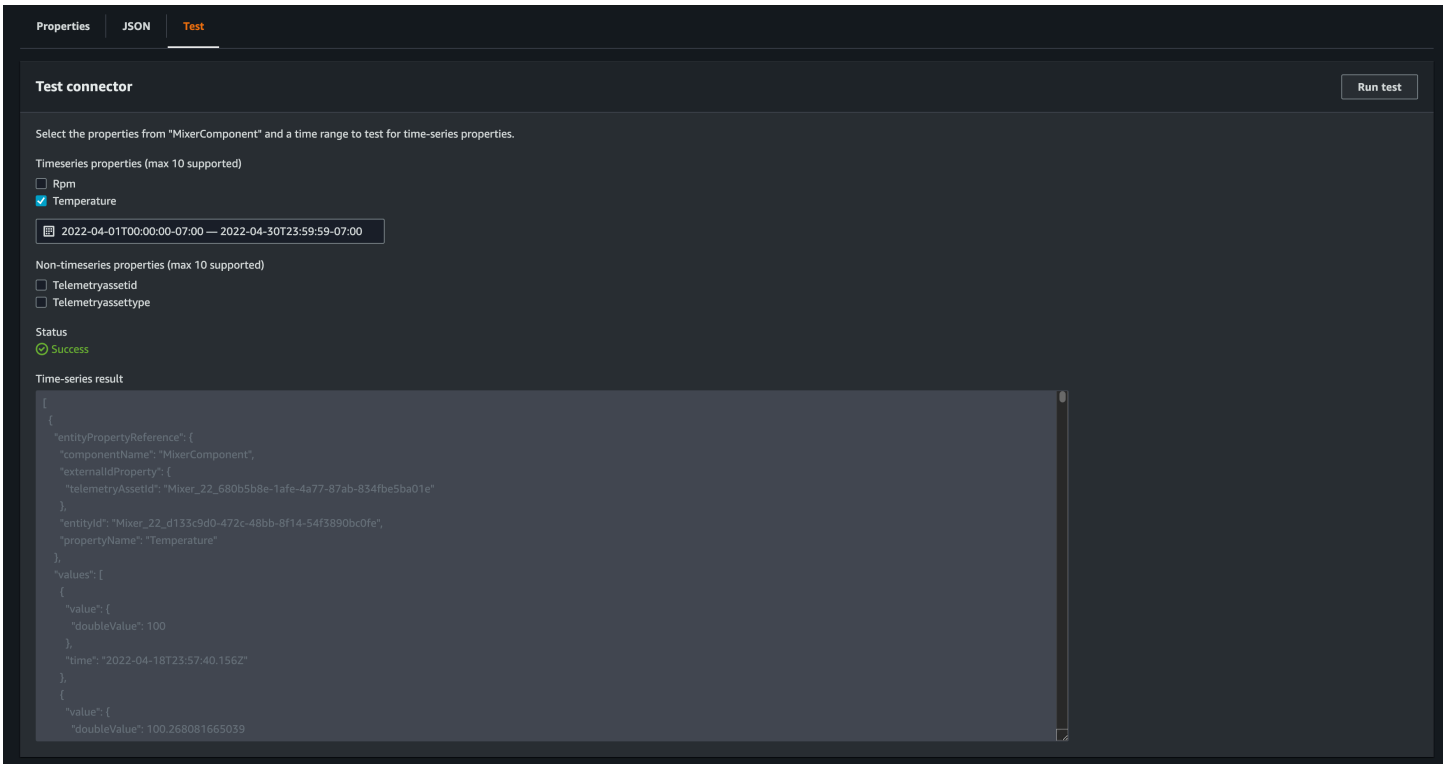
> **Note**
>
> 如果您按照 GitHub 示例中的设置说明进行操作，则所有 AWS IoT TwinMaker 资源都将自动可用。您可以在AWS IoT TwinMaker GitHub 样本中查看组件类型定义。一旦任何组件使用了此组件类型，就无法更新该组件类型的属性定义和函数。

集成测试

我们建议与进行集成测试 AWS IoT TwinMaker ，以验证数据平面查询是否有效 end-to-end。您可以通过 GetPropertyValueHistoryAPI 或在AWS IoT TwinMaker 控制台中轻松执行此操作。



在 AWS IoT TwinMaker 控制台中，转到组件详细信息，然后在 “测试” 下，您会看到组件中的所有属性都列在那里。控制台的测试区域允许您测试时间序列属性和 non-time-series 属性。对于时间序列属性，您也可以使用 GetPropertyValueHistoryAPI，对于 non-time-series 属性，也可以使用 GetPropertyValueAPI。如果您的 Lambda 连接器支持多属性查询，则可以选择多种属性。



接下来做什么

现在，您可以设置 AWS IoT TwinMaker Grafana 控制面板以查看指标。您还可以在示例 GitHub 存储库中浏览其他数据连接器AWS IoT TwinMaker 示例，以查看它们是否适合您的用例。

AWS IoT TwinMaker cookie 出厂示例时间序列连接器

cookie 工厂 Lambda 函数的完整代码可在上找到。 GitHub尽管在将连接器连接至组件类型之后，您仍然可以更新实施，但我们强烈建议您在集成 AWS IoT TwinMaker之前验证 Lambda 连接器。您可以在 Lambda 控制台测试 Lambda 函数，也可以在 AWS CDK本地测试。有关测试 Lambda 函数的更多信息，请参阅测试 Lambda 函数和本地测试应用程序。 AWS CDK

cookie 出厂组件类型示例

组件类型定义了跨组件共享的通用属性。对于 cookie 工厂示例，相同类型的物理组件共享相同的测量值，因此我们可以在组件类型中定义测量架构。例如，以下示例中定义了混合器类型。

```
{
    "componentTypeId": "com.example.cookiefactory.mixer"
    "propertyDefinitions": {
        "RPM": {
            "dataType": { "type": "DOUBLE" },
            "isTimeSeries": true,
            "isRequiredInEntity": false,
            "isExternalId": false,
            "isStoredExternally": true
        },
        "Temperature": {
            "dataType": { "type": "DOUBLE" },
            "isTimeSeries": true,
            "isRequiredInEntity": false,
            "isExternalId": false,
            "isStoredExternally": true
        }
    }
}
```

例如，物理组件可能在 Timestream 数据库中包含测量值，在 SQL 数据库中包含维护记录，或者在警报系统中包含警报数据。创建多个组件，并将它们与实体关联，将不同的数据来源连接至实体并填充实体组件图。在这种情况下，每个组件都需要一个`telemetryId`属性来标识相应数据源中该组件的唯一密钥。指定该`telemetryId`属性有两个好处：可以在数据连接器中将该属性用作筛选条件，仅查询给定组件的值；如果您在数据平面 API 响应中包含该`telemetryId`属性值，则客户端会获取 ID，并在必要时可以执行反向查找。

如果将作为外部 ID 添加到组件类型，则它会在`TimeStream`表格中标识该组件。`TelemetryId`

```
{
    "componentTypeId": "com.example.cookiefactory.mixer"
    "propertyDefinitions": {
        "telemetryId": {
            "dataType": { "type": "STRING" },
            "isTimeSeries": false,
            "isRequiredInEntity": true,
            "isExternalId": true,
            "isStoredExternally": false
        },
        "RPM": {
            "dataType": { "type": "DOUBLE" },
            "isTimeSeries": true,
```

```
            "isRequiredInEntity": false,
            "isExternalId": false,
            "isStoredExternally": true
        },
        "Temperature": {
            "dataType": { "type": "DOUBLE" },
            "isTimeSeries": true,
            "isRequiredInEntity": false,
            "isExternalId": false,
            "isStoredExternally": true
        }
    }
}
```

同样，如以下 JSON 示例所示，我们的组件类型为 `WaterTank`。

```
{
  "componentTypeId": "com.example.cookiefactory.watertank",
  "propertyDefinitions": {
    "flowRate1": {
      "dataType": { "type": "DOUBLE" },
      "isTimeSeries": true,
      "isRequiredInEntity": false,
      "isExternalId": false,
      "isStoredExternally": true
    },
    "flowrate2": {
      "dataType": { "type": "DOUBLE" },
      "isTimeSeries": true,
      "isRequiredInEntity": false,
      "isExternalId": false,
      "isStoredExternally": true
    },
    "tankVolume1": {
      "dataType": { "type": "DOUBLE" },
      "isTimeSeries": true,
      "isRequiredInEntity": false,
      "isExternalId": false,
      "isStoredExternally": true
    },
    "tankVolume2": {
      "dataType": { "type": "DOUBLE" },
      "isTimeSeries": true,
```

```
      "isRequiredInEntity": false,
      "isExternalId": false,
      "isStoredExternally": true
    },
    "telemetryId": {
      "dataType": { "type": "STRING" },
      "isTimeSeries": false,
      "isRequiredInEntity": true,
      "isExternalId": true,
      "isStoredExternally": false
    }
  }
}
```

如果在实体范围内查询属性值，TelemetryType 是可选的组件类型属性 有关示例，请参阅AWS IoT TwinMaker 示例 GitHub 存储库中定义的组件类型。同一个表中还嵌入了警报类型，因此可定义 TelemetryType，且您可将 TelemetryId 和 TelemetryType 等常见属性提取至父级组件类型，以供其他子类型共享。

Lambda 示例

Lambda 连接器需要访问数据来源，并根据输入生成查询语句并将其转发至数据来源。以下 JSON 示例显示了发送至 Lambda 的示例请求。

```
{
    'workspaceId': 'CookieFactory',
    'selectedProperties': ['Temperature'],
    'startDateTime': 1648796400,
    'startTime': '2022-04-01T07:00:00.000Z',
    'endDateTime': 1650610799,
    'endTime': '2022-04-22T06:59:59.000Z',
    'properties': {
        'telemetryId': {
            'definition': {
                'dataType': { 'type': 'STRING' },
                'isTimeSeries': False,
                'isRequiredInEntity': True,
                'isExternalId': True,
                'isStoredExternally': False,
                'isImported': False,
                'isFinal': False,
                'isInherited': True,
            },
```

```
            'value': {
                'stringValue': 'Mixer_22_680b5b8e-1afe-4a77-87ab-834fbe5ba01e'
            }
        }
        'Temperature': {
            'definition': {
                'dataType': { 'type': 'DOUBLE' },
                'isTimeSeries': True,
                'isRequiredInEntity': False,
                'isExternalId': False,
                'isStoredExternally': True,
                'isImported': False,
                'isFinal': False,
                'isInherited': False
            }
        }
        'RPM': {
            'definition': {
                'dataType': { 'type': 'DOUBLE' },
                'isTimeSeries': True,
                'isRequiredInEntity': False,
                'isExternalId': False,
                'isStoredExternally': True,
                'isImported': False,
                'isFinal':False,
                'isInherited': False
            }
        },
    'entityId': 'Mixer_22_d133c9d0-472c-48bb-8f14-54f3890bc0fe',
    'componentName': 'MixerComponent',
    'maxResults': 100,
    'orderByTime': 'ASCENDING'
}
```

Lambda 函数的目标是查询给定实体的历史测量数据。 AWS IoT TwinMaker 提供了组件属性映射，您应该为组件 ID 指定一个实例化的值。例如，要处理组件类型级别的查询（这在警报用例中很常见）并返回工作区中所有组件的警报状态，则属性图具有组件类型属性定义。

对于最直接的情况，就像在前面的请求中一样，我们希望在给定时间窗口内按时间升序为给定组件提供一系列温度样本。查询语句可概括如下：

```
...
SELECT measure_name, time, measure_value::double
```

```
    FROM {database_name}.{table_name}
    WHERE time < from_iso8601_timestamp('{request.start_time}')
    AND time >= from_iso8601_timestamp('{request.end_time}')
    AND TelemetryId = '{telemetry_id}'
    AND measure_name = '{selected_property}'
    ORDER BY time {request.orderByTime}
...
```

创建和编辑 AWS IoT TwinMaker 场景

场景是数字孪生的三维可视化。它们是编辑数字孪生的主要方式。了解如何将警报、时间序列数据、颜色叠加、标签和视觉规则添加到场景，以使数字孪生可视化与真实世界用例保持一致。

本节将介绍以下主题：

创建第一个场景前

场景依赖资源体现数字孪生。此资源由 3D 模型、数据或纹理文件组成。资源的大小和复杂性、场景中的元素（例如灯光）以及计算机硬件都会影响 AWS IoT TwinMaker 场景性能。使用本主题信息减少延迟、加载时间并提高场景帧速率。

在将资源导入之前对其进行优化 AWS IoT TwinMaker

您可以使用 AWS IoT TwinMaker 与您的数字双胞胎进行实时交互。为了获得最佳场景体验，我们建议您优化资源在实时环境中的使用。

3D 模型可能会对性能有显著影响。复杂模型几何形状和网格会降低性能。例如，工业 CAD 模型对细节要求较高。我们建议先压缩这些模型的网格并减少其多边形数量，然后再将其用于场 AWS IoT TwinMaker 景。如果要为创建新的 3D 模型 AWS IoT TwinMaker，则应确定细节水平并在所有模型中保持该水平。从模型中移除不影响用例可视化或解释说明的细节。

若要压缩模型并减小文件大小，请使用开源网格压缩工具，例如 DRACO 3D 数据压缩。

未优化纹理也会影响性能。如果您不需要纹理中有任何透明度，可以考虑选择 PEG 图像格式，不需要 PNG 格式。您可以使用开源纹理压缩工具（例如 Basis Universal 纹理压缩）压缩纹理文件。

AWS IoT TwinMaker性能的最佳方案

要获得最佳性能 AWS IoT TwinMaker，请注意以下限制和最佳实践。

- AWS IoT TwinMaker 场景渲染性能取决于硬件。该性能因不同的计算机硬件配置而异。
- 我们建议您的 AWS IoT TwinMaker所有对象的多边形总数应小于 100 万。
- 我们建议每个场景的对象总数为 200 个。将场景中的对象数量增加至 200 以上，会降低场景帧速率。
- 我们建议场景中唯一 3D 资源的总大小不超过 100 兆字节。否则，根据浏览器和硬件的不同，您可能会遇到加载速度慢或性能下降的问题。
- 默认情况下，场景配备环境光照。可以在场景中添加额外的灯光，以使某些对象聚焦，或者在对象上投射阴影。我们建议每个场景使用一盏照明。在需要的地方使用照明，并避免在场景中复制真实灯光。

了解更多信息

使用这些资源，了解更多用于提高场景性能的优化技巧。

- 如何将 OBJ 模型转换并压缩为 GLTF 以供使用 AWS IoT TwinMaker
- 针对网页内容优化 3D 模型
- 优化场景以获得更好的 WebGL 性能

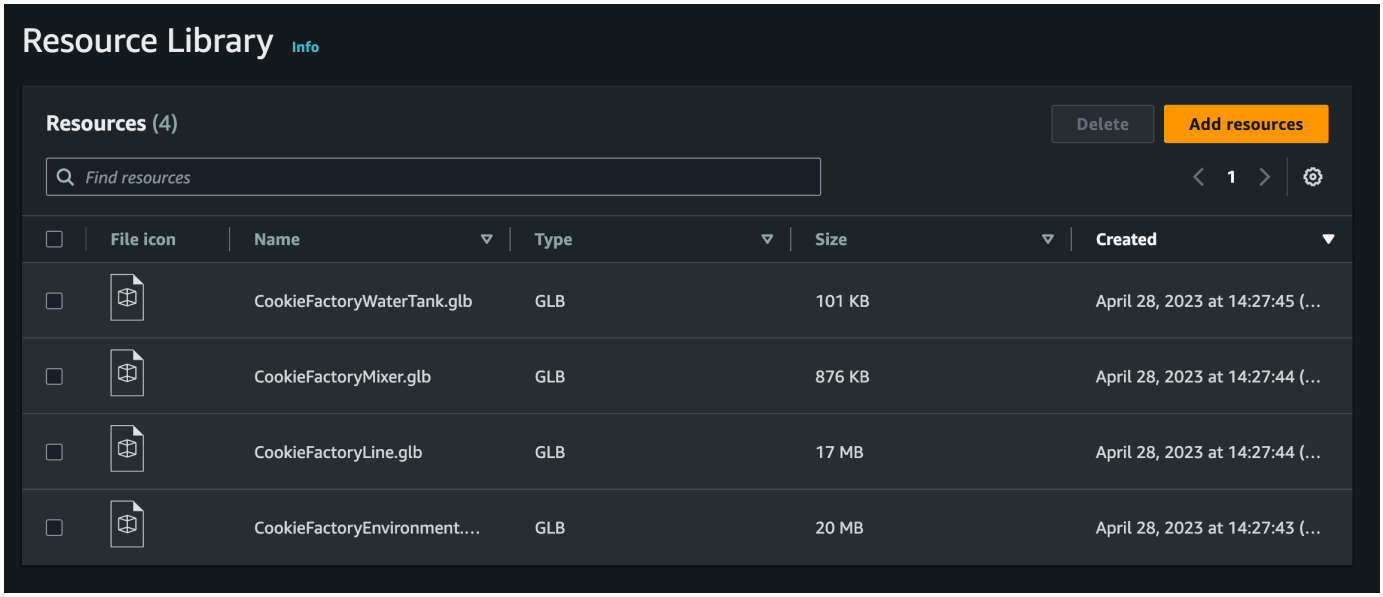
将资源上传到 AWS IoT TwinMaker 资源库

您可以使用资源库控制和管理您希望放入数字孪生应用场景中的任何资源。要 AWS IoT TwinMaker 了解这些资源，请使用资源库控制台页面上传这些资源。

使用控制台将文件上传到资源库

按照以下步骤使用 AWS IoT TwinMaker 控制台将文件添加到资源库中。

1. 在左侧导航菜单的 “工作空间” 下，选择 “资源库”。
2. 选择 “添加资源”，然后选择要上传的文件。



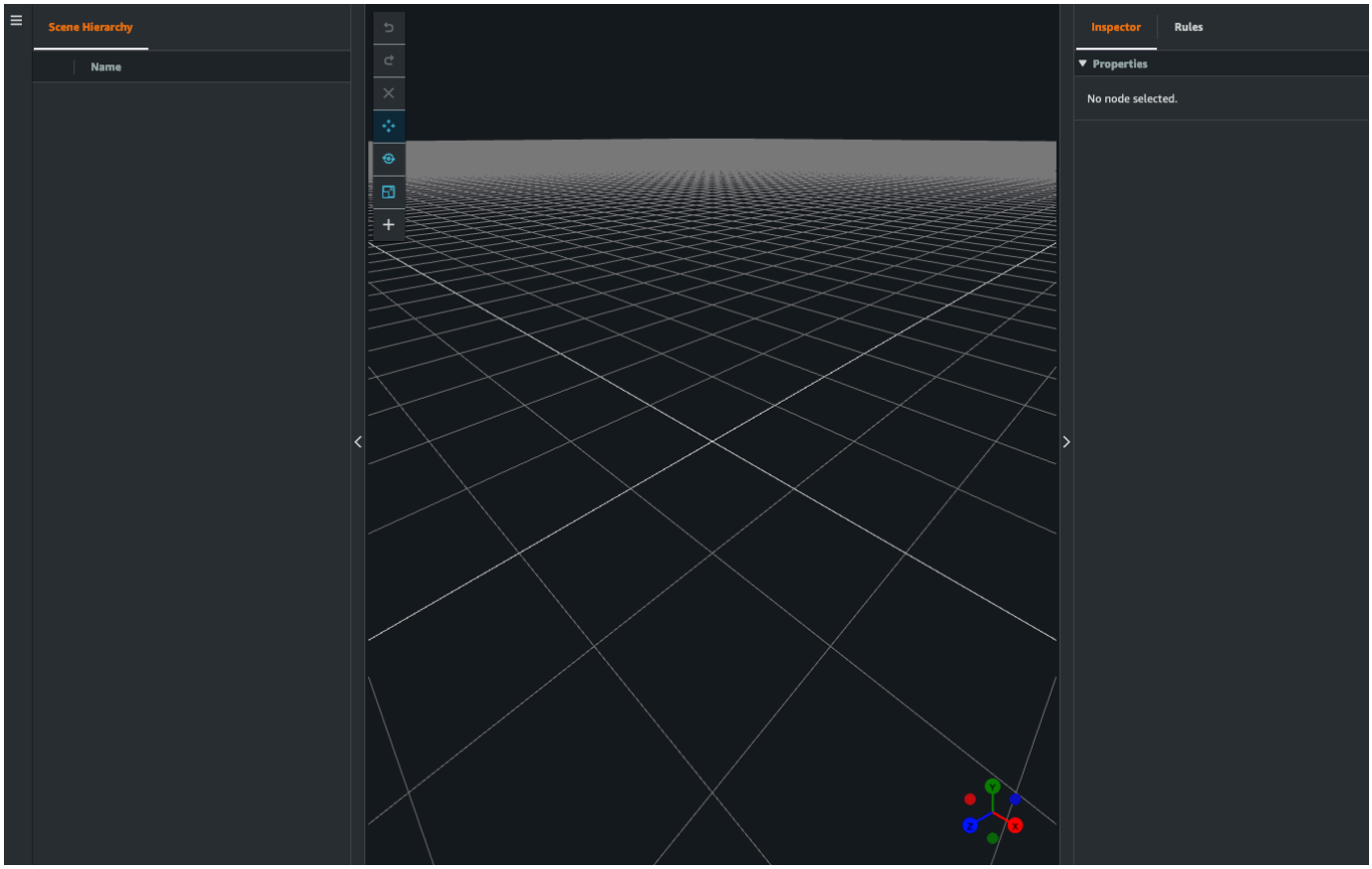
创建场景

在本部分，您将设置一个场景，以便能编辑数字孪生。您可以导入已上传到资源库的 3D 模型，然后添加控件并将属性数据绑定到对象以完成您的数字双胞胎。场景对象可以包括整个建筑物或空间，也可以包括位于其实际位置的单个设备。

> **Note**
>
> 在创建场景之前，必须创建一个工作空间。

使用以下步骤在中创建场景 AWS IoT TwinMaker。

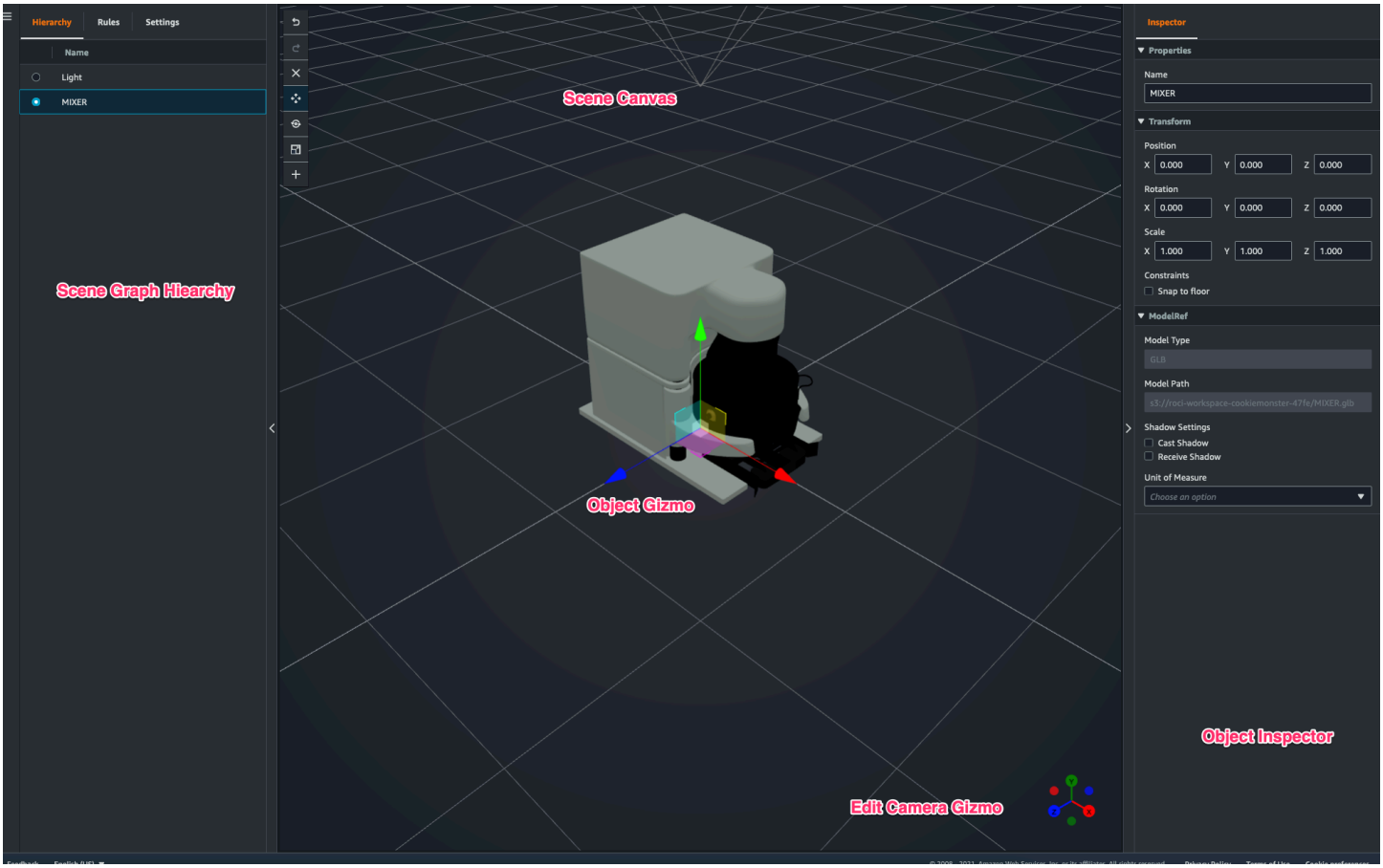
1. 要打开场景窗格，请在工作区的左侧导航栏中选择 “场景”。
2. 选择 Create scene（创建场景）。新的场景创建窗格随即打开。
3. 在场景创建窗格中，输入新场景的名称和描述。如果您有标准或分层套餐定价计划，则可以选择场景类型。建议使用动态场景。
4. 当您准备好创建用户时，请选择 Create scence（创建场景）。新场景随即打开，可以开始使用了。



在场景 AWS IoT TwinMaker 中使用 3D 导航

AWS IoT TwinMaker 场景中有一组导航控件，可用于在场景的 3D 空间中高效导航。要与场景所代表的 3D 空间和对象交互，可以使用以下小部件和菜单选项。

- 检查器：使用“检查器”窗口查看和编辑层次结构中选定实体或组件的属性和设置。
- 场景画布：场景画布是 3D 空间，可以在该空间定位和定向要使用的任何 3D 资源。
- 场景图层次结构：可以使用此面板来查看场景中存在的所有实体。它出现在窗口的左侧。
- 对象小工具：使用此小工具在画布上四处移动对象。它出现在场景画布中选定 3D 对象的中心。
- 编辑摄像头小工具：使用“编辑摄像头”小工具快速查看场景视图摄像头的当前方位并修改视角。可以在场景视图的右下角找到此小工具。
- 缩放控件：要在场景画布上导航，请点击右键并朝要移动的方向拖动。要旋转，请单击左键并拖动以旋转。要缩放，请使用鼠标上的滚轮，或者在笔记本电脑的触控板上捏合手指并分开。



层次结构窗格上的场景按钮按照按钮布局的顺序列出了以下功能：

- 撤销：撤销在场景中的最后一次更改。
- 重做：重做场景中的最后一次更改。
- 加号 (+)：使用此按钮可以访问以下操作：添加空节点、添加 3D 模型、添加标签、添加光源和添加模型着色器。
- 更改导航方法：访问场景摄像头导航选项，环绕和平移。
- 回收站（删除）：使用此按钮可删除场景中的选定对象。
- 对像操纵工具：使用此按钮可平移、旋转和缩放选定对象。

向实体添加固定摄像头

您可以将固定的摄像机视图附加到 AWS IoT TwinMaker 场景中的实体。这些摄像头可提供有关 3D 模型的固定视角，使得可以快速轻松地将场景中的视角转移到目标实体。

1. 在 AWS IoT TwinMaker 控制台中导航到您的场景。

2. 在场景层次结构菜单中，选择要将摄像头连接到的实体。
3. 按下 + 按钮，然后从下拉选项中选择从当前视图添加摄像头。将具有当前视角的摄像头应用于实体。
4. 在检查器中，可以配置摄像头并调整以下设置：

   - 摄像头名称
   - 摄像头位置和旋转
   - 摄像头焦距
   - 缩放级别
   - 近距离和远距离剪辑平面
5. 在放置摄像头后访问摄像头。在层次结构中选择已为其添加摄像头的实体。查找实体下方列出的摄像头名称。
6. 从实体中选择放置的摄像头后，场景摄像头视图将按已放置摄像头的设定视角捕捉画面。

场景增强编辑

AWS IoT TwinMaker 场景具有一组工具，用于增强、编辑和操作场景中存在的资源。

以下主题将教您如何在 AWS IoT TwinMaker 场景中使用增强的编辑功能。

- 针对性地放置场景对象
- 子模型选择
- 编辑场景层次结构中的实体

针对性地放置场景对象

AWS IoT TwinMaker 允许您在场景中精确放置和添加对象。这种增强编辑功能可以让您更好地控制在场景中放置标签、实体、光源和模型的位置。

1. 在 AWS IoT TwinMaker 控制台中导航到您的场景。
2. 按下 + 按钮，然后从下拉选项中选择其中一个选项。这可能是模型、光源、标签或 + 菜单中的任何东西。

   当您在场景的 3D 空间中移动光标时，应该会在光标周围看到一个靶框。
3. 使用目标精确地将元素放置在场景中。

子模型选择

AWS IoT TwinMaker 允许您在场景中选择 3D 模型的子模型并对其应用标准属性，例如标签、灯光或规则。

3D 模型文件格式包含元数据，这些元数据可以将模型的子区域指定为较大模型中的子模型。例如，模型可以是过滤系统，该系统的各个部分（如储罐、管道或电机）被标记为该过滤系统三维模型的子模型。

场景中受支持的 3D 文件格式：GLB 和 GLTF。

1. 在 AWS IoT TwinMaker 控制台中导航到您的场景。
2. 如果场景中没有模型，则确保通过从 + 菜单中进行选择来添加一个模型。
3. 选择场景层次结构中列出的模型，选择后，层次结构应显示模型下方的所有子模型。

> **Note**
>
> 如果没有看到列出的任何子模型，则该模型可能未配置为包含任何子模型。

4. 要切换下级模型的可见性，请按下位于层次结构中子模型名称右侧的眼睛图标。
5. 要编辑子模型数据（如名称或位置），场景检查器将在选择子模型时打开。使用检查器菜单更新或更改子模型数据。
6. 要向子模型添加标签、光源、规则或其他属性，请在层次结构中选择子模型时按 +。

编辑场景层次结构中的实体

AWS IoT TwinMaker 场景允许您直接编辑层次结构表中实体的属性。以下步骤说明您可以通过层次结构菜单对实体执行哪些操作。

1. 在 AWS IoT TwinMaker 控制台中导航到您的场景。
2. 打开场景层次结构，然后选择要操纵的实体的子元素。
3. 选择该元素后，按 + 按钮，然后从下拉列表中选择一个选项：

   - 添加空节点
   - 添加 3D 模型
   - 添加光源

- 从当前视图添加摄像头
- 添加标签
- 添加模型着色器
- 添加运动指示器

4. 从下拉列表中选择一个选项后，所选内容将作为步骤 2 中选定元素的子元素应用于场景。
5. 选择子元素并在层次结构中拖动到新的父元素，可以对子元素和重定父级元素进行重新排序。

为实体添加注释

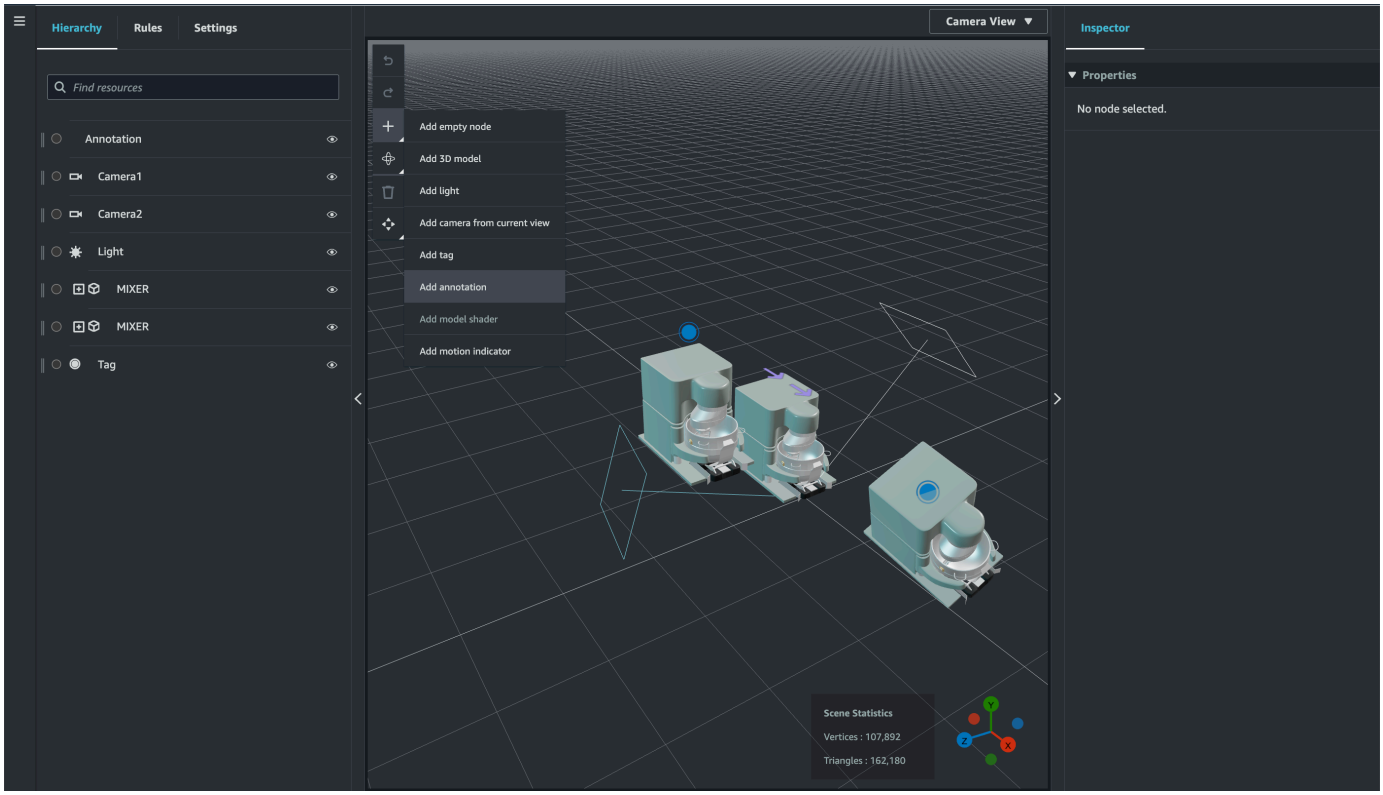
场 AWS IoT TwinMaker 景编辑器允许您为场景层次结构中的任何元素添加注释。注释是用 Markdown 创作的。

有关使用 Markdown 编写的更多信息，请参阅有关 Markdown 语法的官方文档基本语法。

> **Note**
>
> AWS IoT TwinMaker 仅限注释和叠加 Markdown 语法，而不是 HTML。

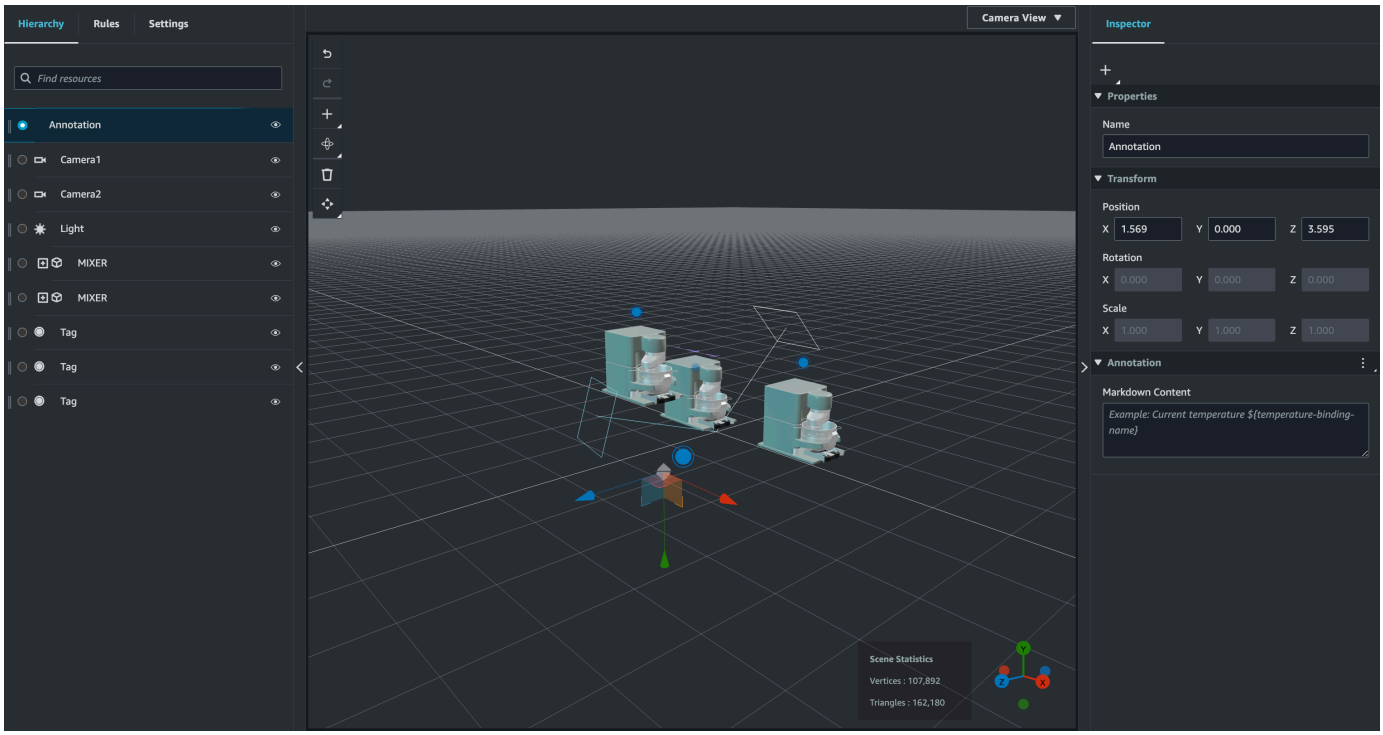
向实体添加注释

1. 在 AWS IoT TwinMaker 控制台中导航到您的场景。
2. 从场景层次结构中选择要注释的元素。如果没有选择层次结构中的任何元素，则可以向根添加注释。
3. 按加号 + 按钮，然后选择添加注释选项。

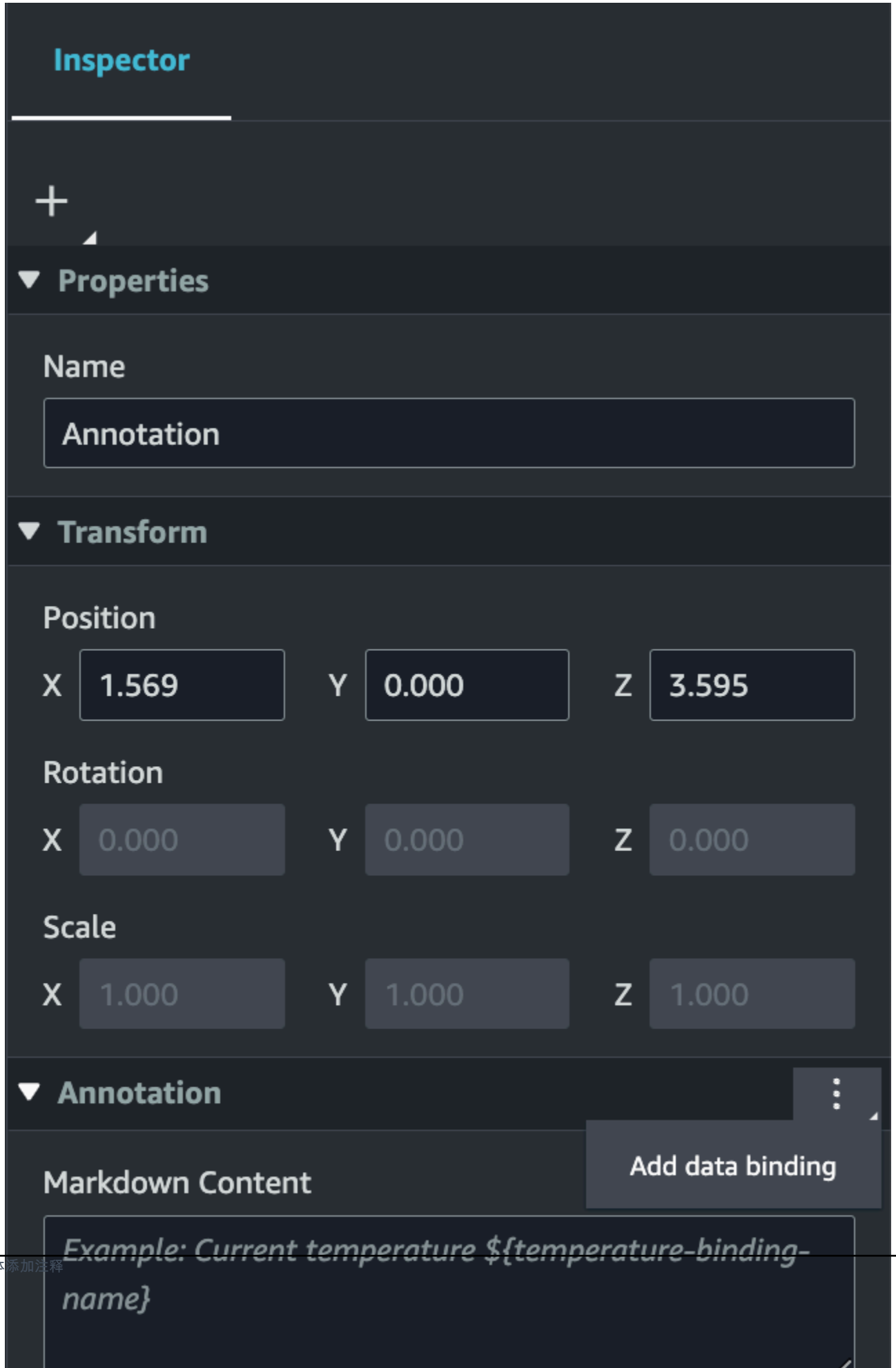


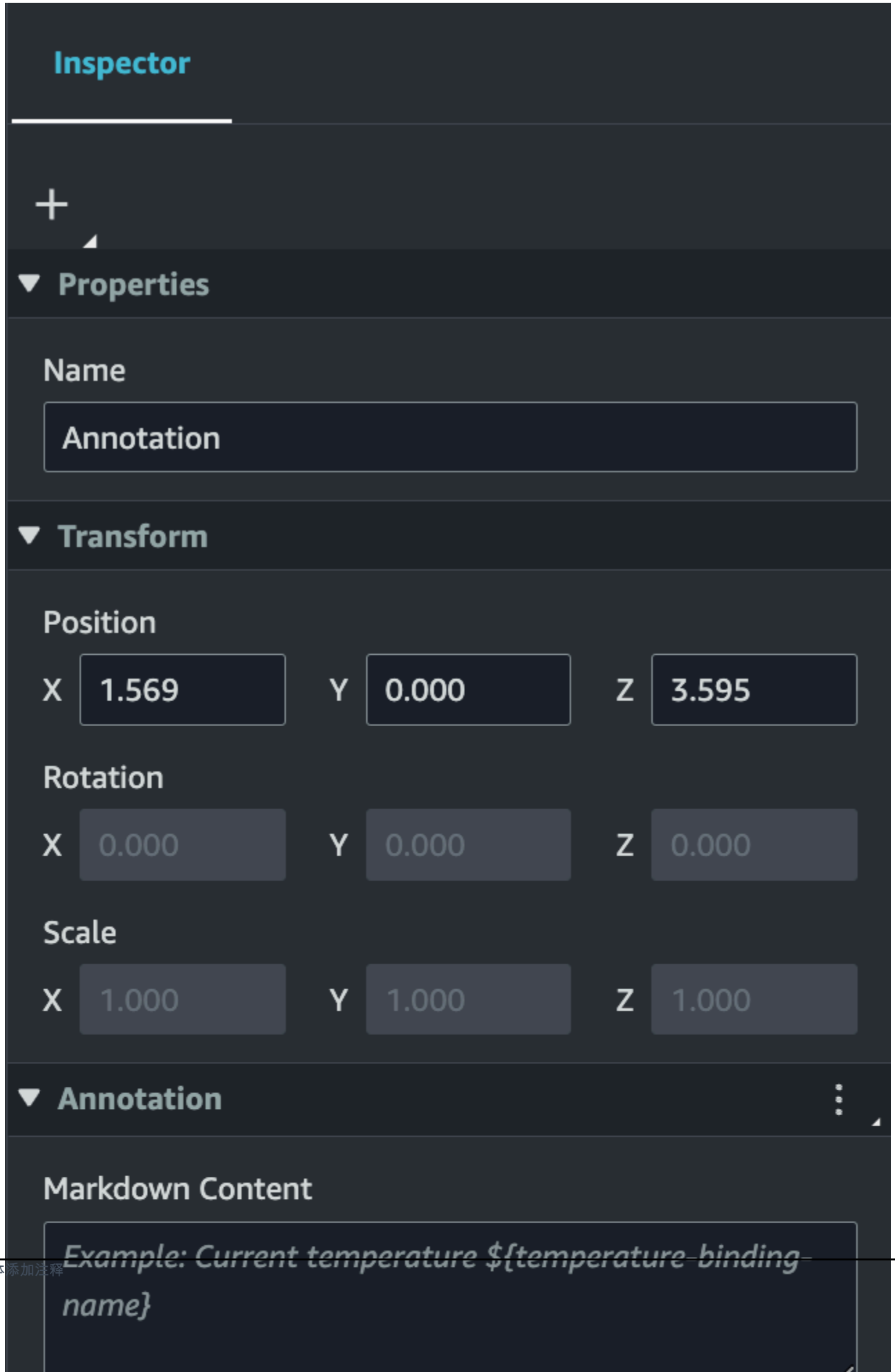
4. 在左侧的检查器窗口中，向下滚动到注释部分。使用 Markdown 语法，写下想让注释显示的文本。

   有关使用 Markdown 编写的更多信息，请参阅有关 Markdown 语法的官方文档基本语法。



5. 要将 AWS IoT TwinMaker 场景数据绑定到注释，请选择添加数据绑定，添加实体 ID，然后选择要显示数据的实体的组件名称和属性名称。您可以更新绑定名称以将其用作 Markdown 变量，并在注释中显示数据。

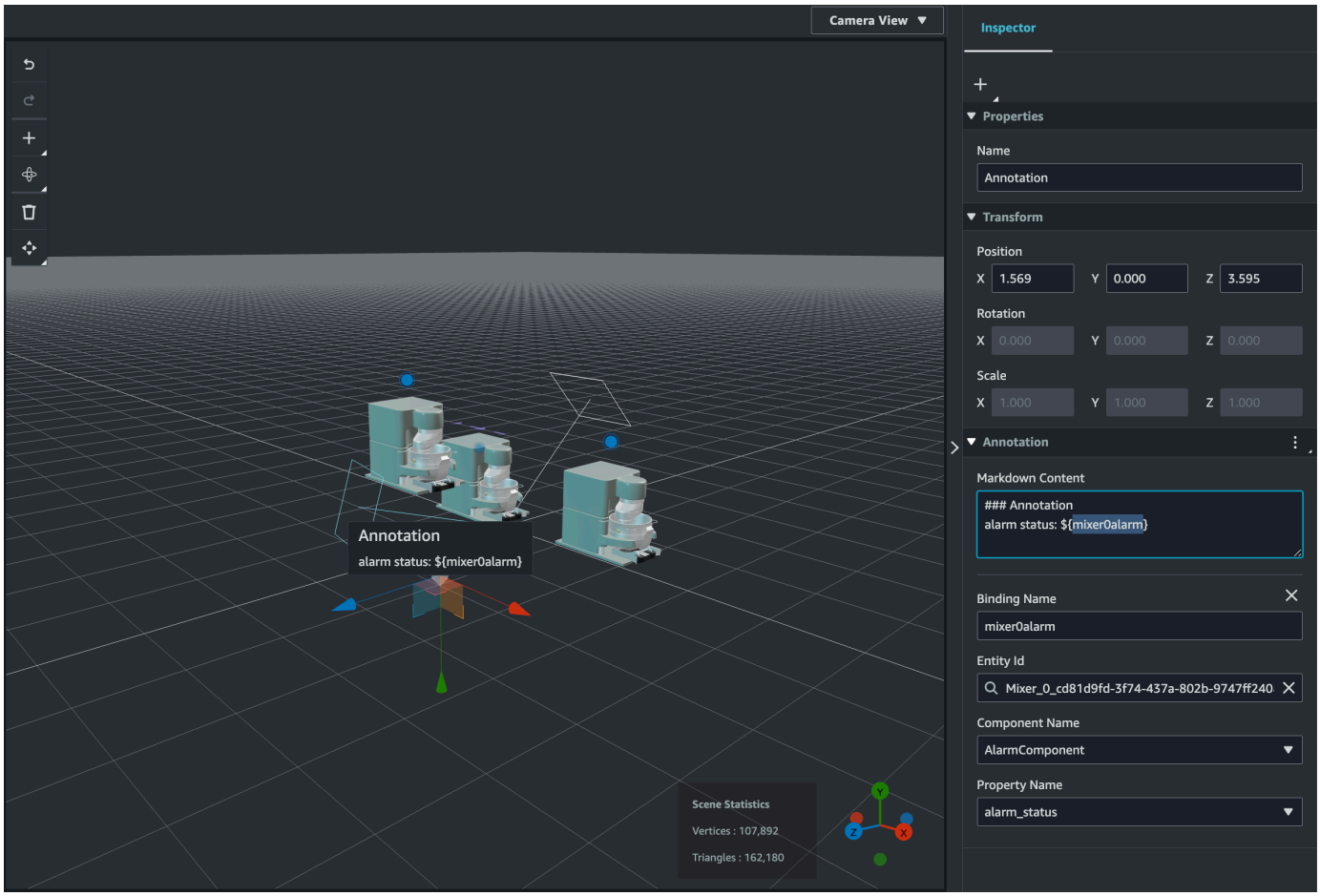

Inspector
Properties
Name
Annotation
Transform
Position
X 1.569
Y 0.000
Z 3.595
Rotation
X 0.000
Y 0.000
Z 0.000
Scale
X 1.000
Y 1.000
Z 1.000
Annotation
Add data binding
Markdown Content
Example: Current temperature ${temperature-binding-name}


Inspector
Properties
Name
Annotation
Transform
Position
X 1.569
Y 0.000
Z 3.595
Rotation
X 0.000
Y 0.000
Z 0.000
Scale
X 1.000
Y 1.000
Z 1.000
Annotation
Markdown Content
Example: Current temperature ${temperature-binding-name}

6. 绑定名称用于表示该注释的变量。

   输入绑定名称，通过 AWS IoT TwinMaker变量语法在注释中显示实体时间序列的最新历史值：${*variable-name*}

   例如，此叠加层在带有语法 ${mixer0alarm} 的注释中显示 mixer0alarm 的值。



为标签添加叠加层

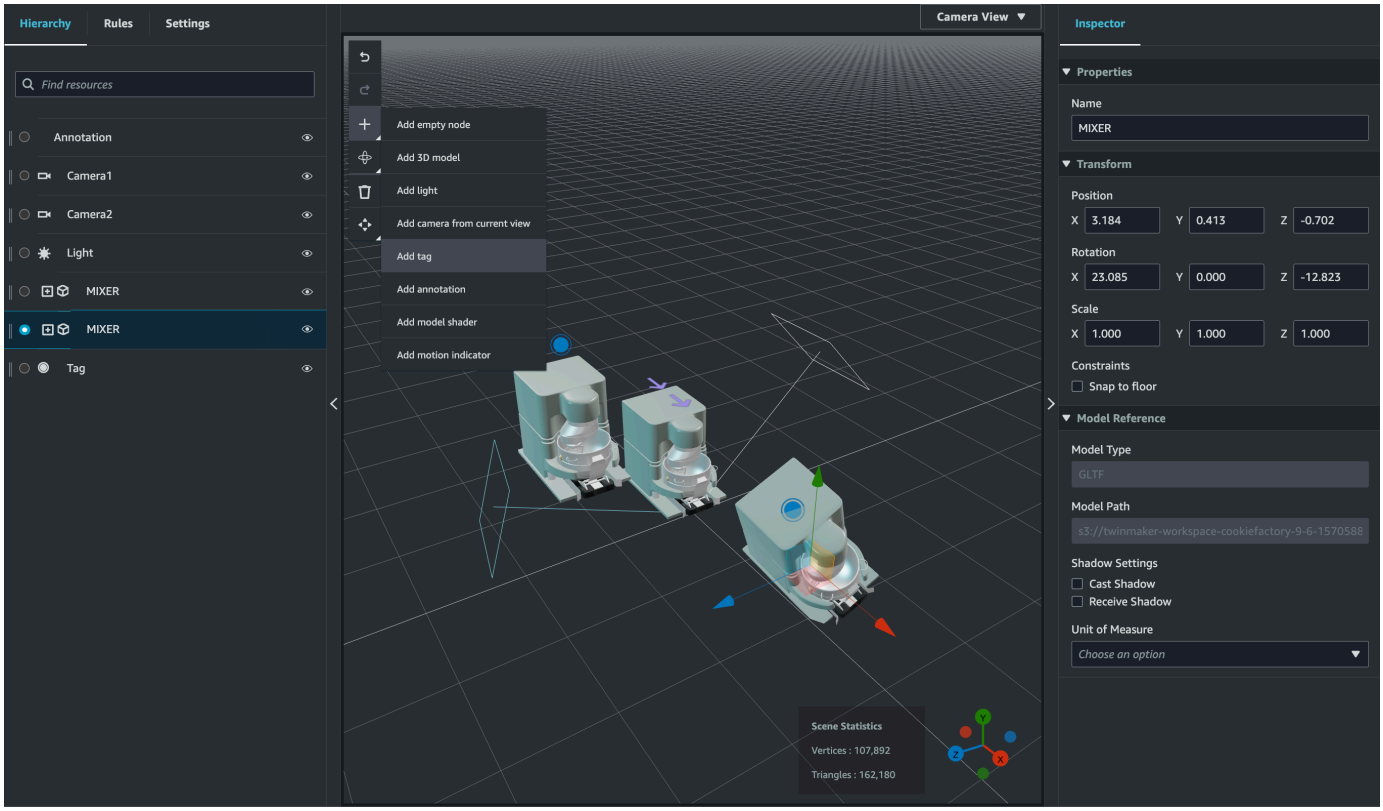
您可以为 AWS IoT TwinMaker 场景创建叠加层。场景叠加层与标签相关联，可用于显示与场景实体关联的关键数据。叠加层是用 Markdown 创作和渲染的。

有关使用 Markdown 编写的更多信息，请参阅有关 Markdown 语法的官方文档基本语法。

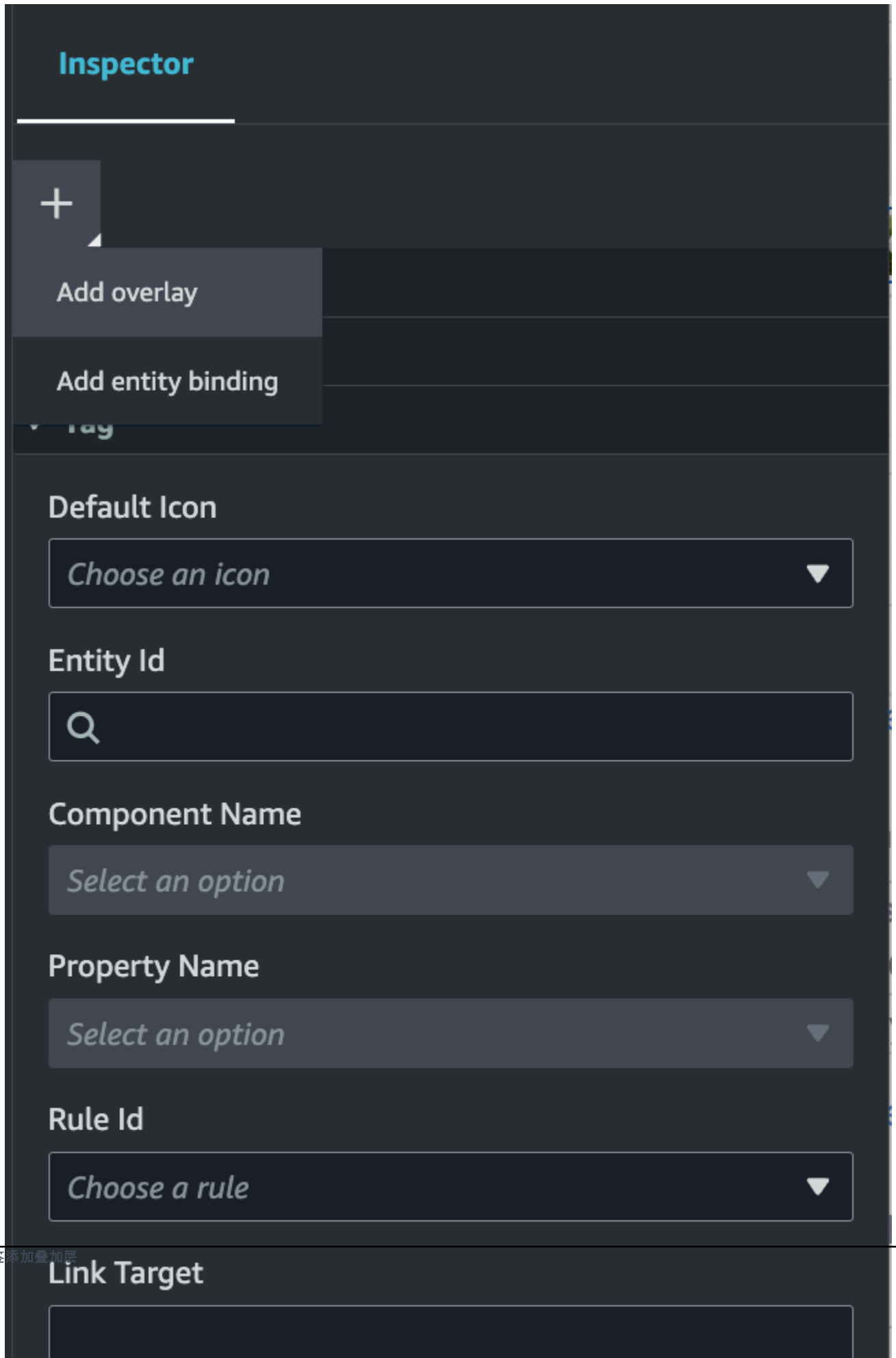
> **Note**
>
> 默认情况下，只有选中与叠加层关联的标签后，才会在场景中可见。你可以在场景设置中切换此设置，这样所有叠加层都可以同时看见。

1. 在 AWS IoT TwinMaker 控制台中导航到您的场景。
2. AWS IoT TwinMaker 叠加层与标签场景相关联，您可以更新现有标签或添加新标签。

   按加号 + 按钮，然后选择添加标签选项。



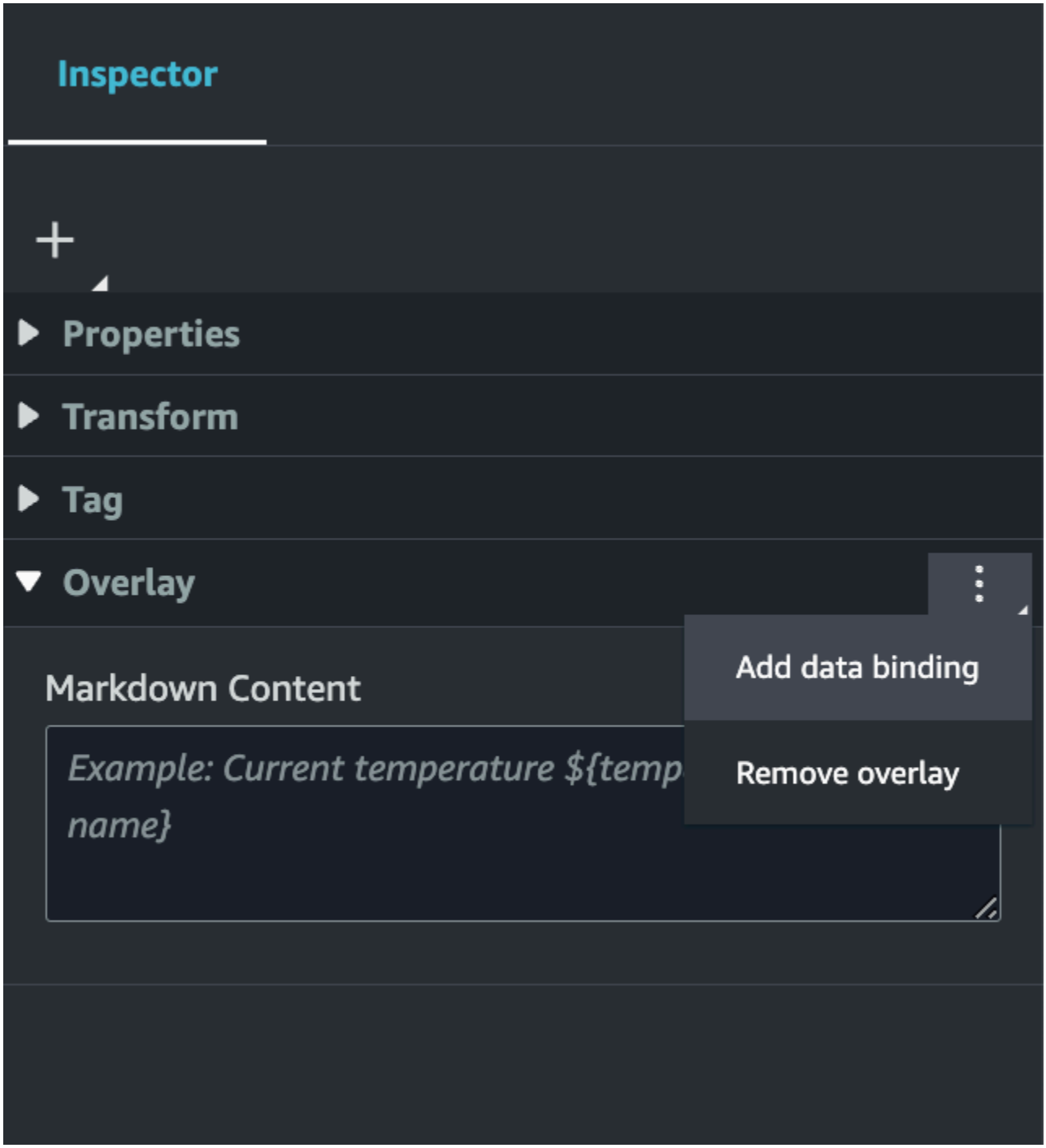
3. 在右侧的 Insp ector 面板中，选择 + (加号) 按钮，然后选择 “添加叠加”。


Inspector
Add overlay
Add entity binding
Default Icon
Choose an icon
Entity Id
Component Name
Select an option
Property Name
Select an option
Rule Id
Choose a rule
Link Target

4. 按照 Markdown 语法，写下想让叠加层显示的文本。

   有关使用 Markdown 编写的更多信息，请参阅有关 Markdown 语法的官方文档基本语法。

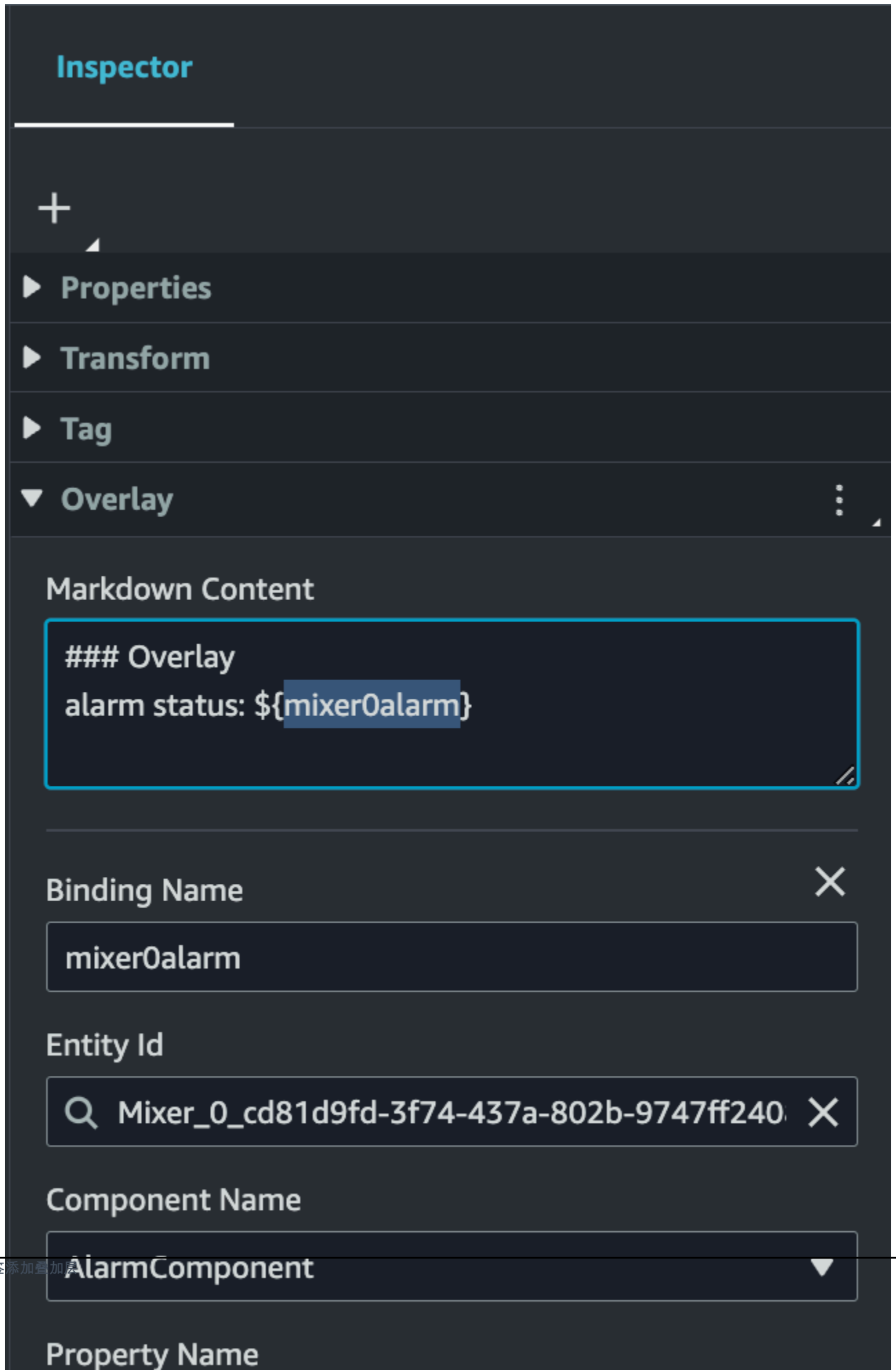
5. 要将 AWS IoT TwinMaker 场景数据绑定到叠加层，请选择添加数据绑定。



   添加绑定名称和实体 ID，然后选择要显示数据的实体的组件名称和属性名称。

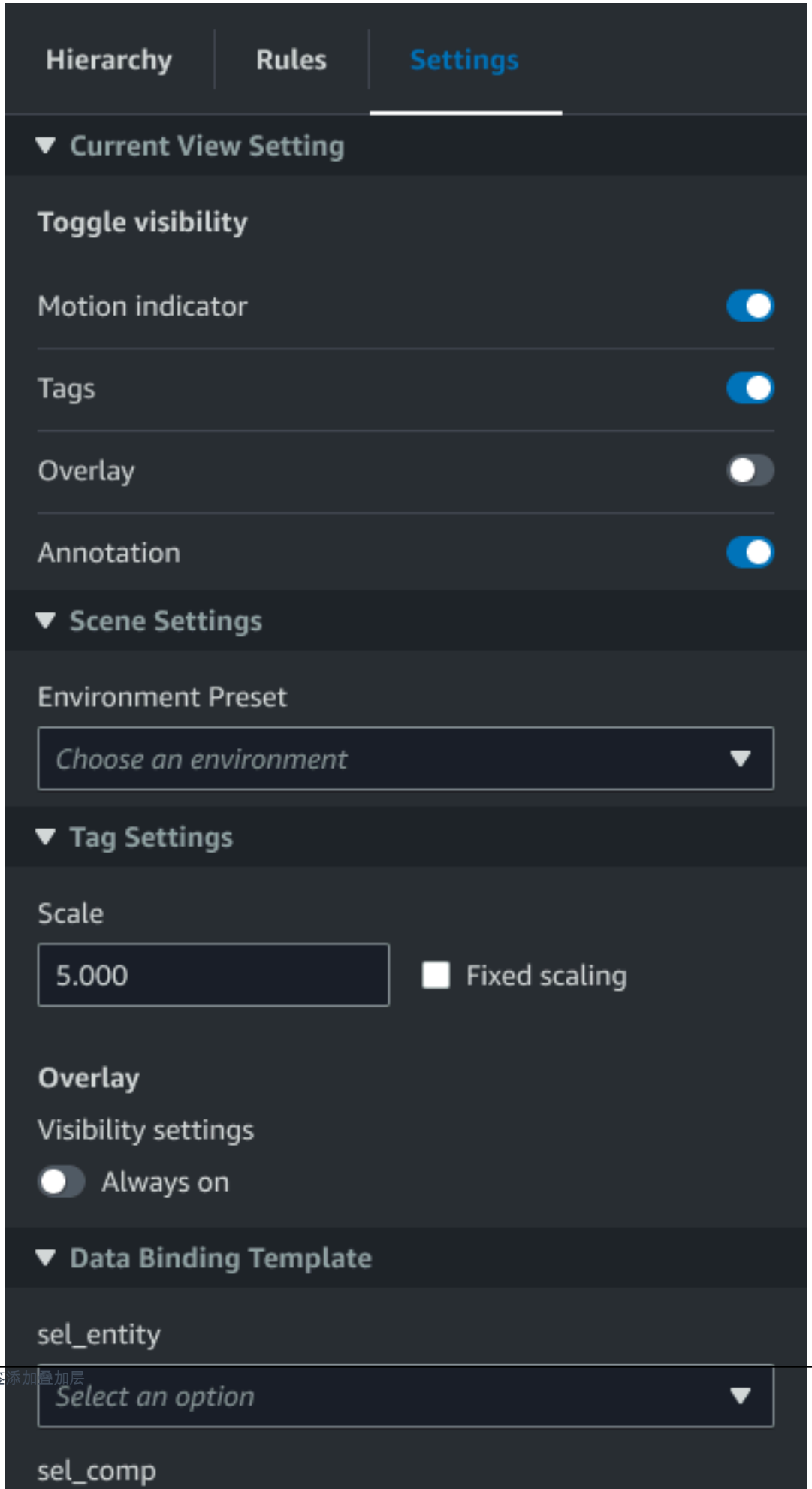
6. 您可以通过 AWS IoT TwinMaker变量语法在叠加层中显示实体时间序列数据的最新历史值:${*variable-name*}.

   例如，此叠加层在叠加层中使用语法 `mixer0alarm` 显示 `${mixer0alarm}` 的值。


Inspector
Properties
Transform
Tag
Overlay
Markdown Content
Overlay
alarm status: ${mixer0alarm}
Binding Name
mixer0alarm
Entity Id
Mixer_0_cd81d9fd-3f74-437a-802b-9747ff240
Component Name
AlarmComponent
Property Name

7. 要启用叠加层可见性，请打开左上角的 “设置” 选项卡，并确保 “叠加” 的切换开关已打开，以便所有叠加层同时可见。

> **Note**
>
> 默认情况下，只有选中与叠加层关联的标签后，才会在场景中可见。


Hierarchy
Rules
Settings
Current View Setting
Toggle visibility
Motion indicator
Tags
Overlay
Annotation
Scene Settings
Environment Preset
Choose an environment
Tag Settings
Scale
5.000
Fixed scaling
Overlay
Visibility settings
Always on
Data Binding Template
sel_entity
Select an option
sel_comp

编辑您的场景

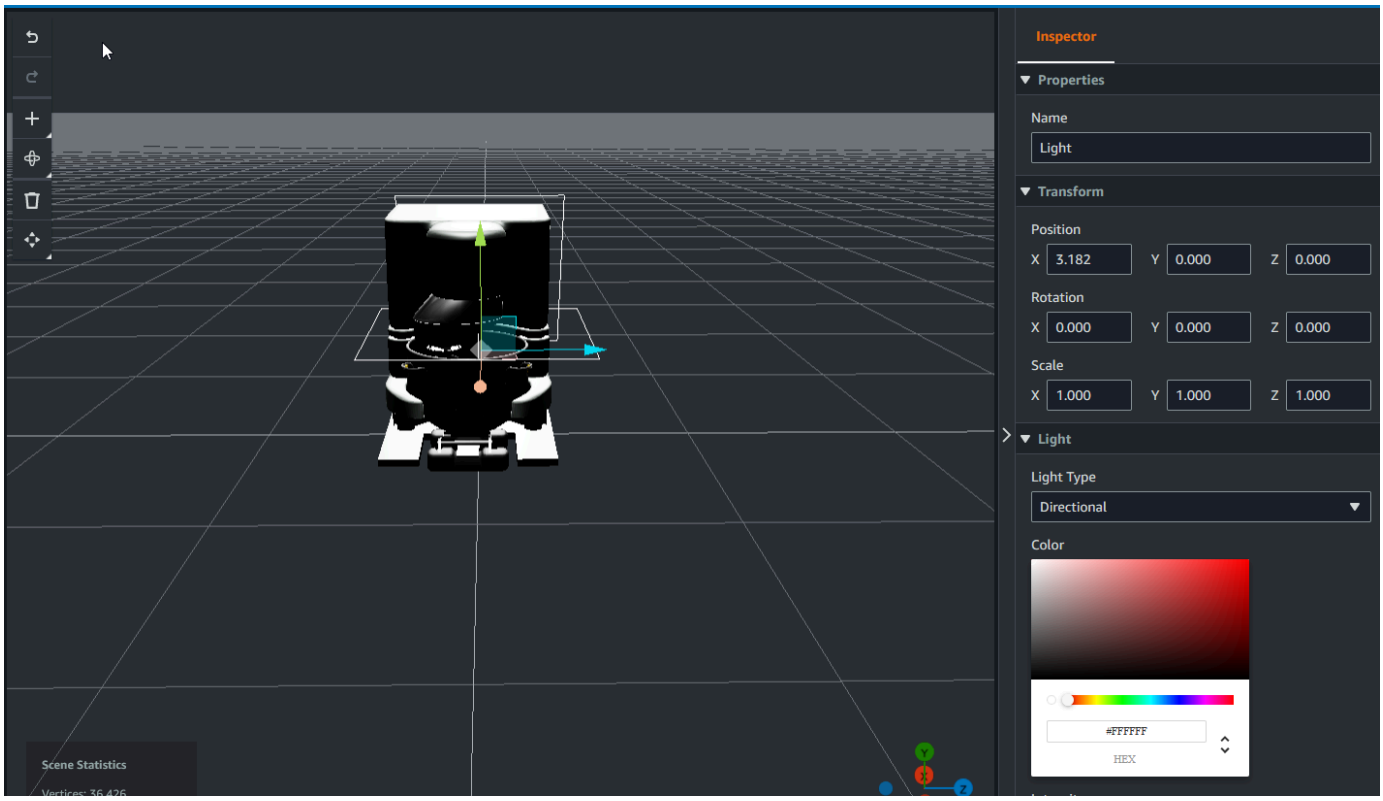
创建场景后，可以在场景中添加实体、组件和配置增强小组件。使用实体组件和小组件对您的数字孪生进行建模，并提供与您的用例相匹配的功能。

将模型添加至您的场景

若要将模型添加至场景，请使用以下程序。

> **Note**
>
> 若要将模型添加至场景，您必须先将模型上传至 AWS IoT TwinMaker 资源库。有关更多信息，请参阅 将资源上传到 AWS IoT TwinMaker 资源库。

1. 在场景编辑器页面，选择加号 (+)，然后选择添加 3D 模型。
2. 在 “从资源库添加资源” 窗口中，选择 CookieFactorMixer.glb 文件，然后选择 “添加”。打开场景编辑器。
3. 可选：选择加号(+)，然后选择添加光。
4. 选择每个灯光选项，以查看它们如何影响场景。



> **Note**
>
> 场景采用默认的环境光照。为避免帧速率损失，请考虑限制场景中放置的其他灯光数量。

将模型着色器增强的 UI 控件添加到场景中

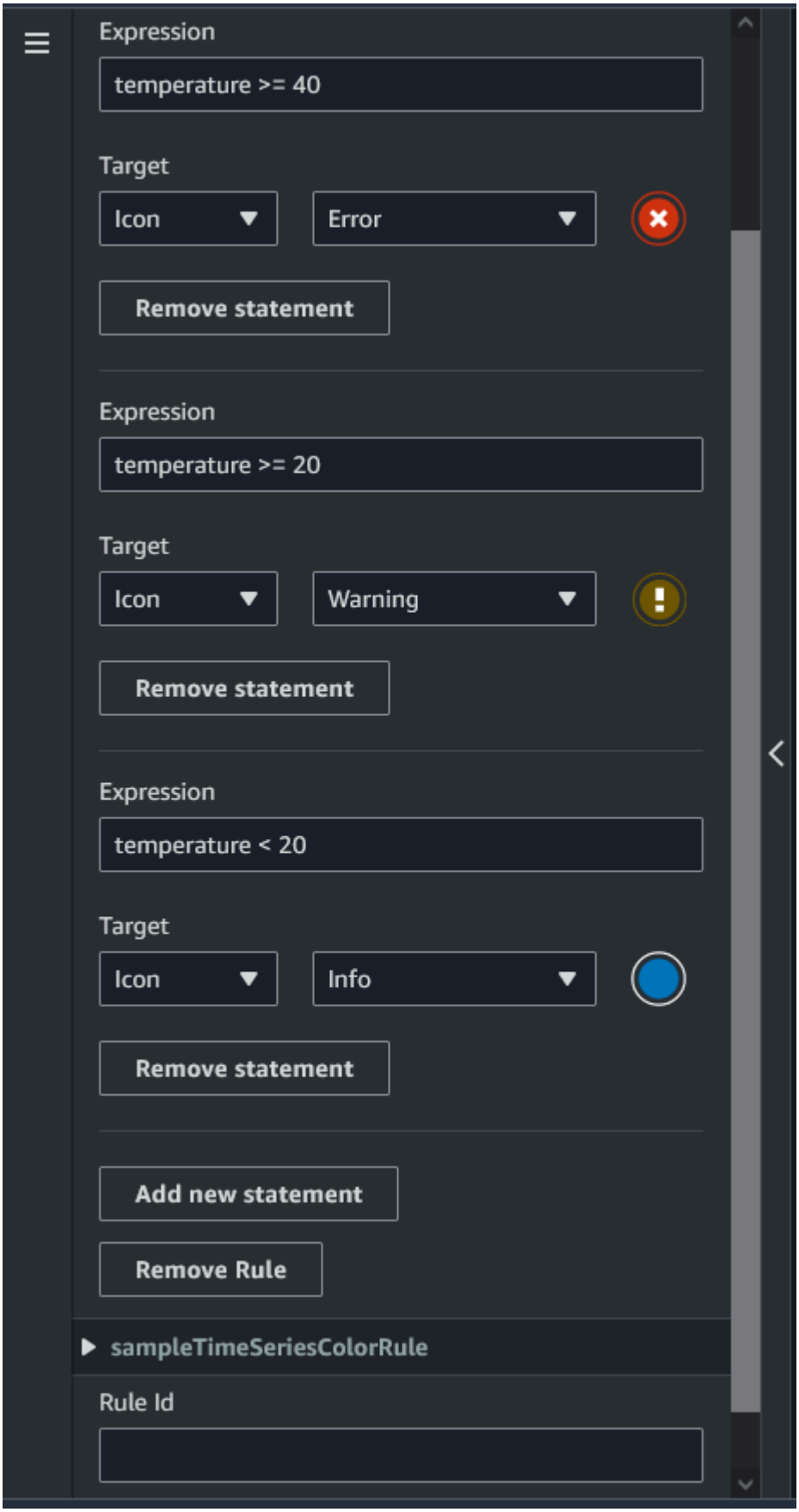
模型着色器控件可以在您定义的条件下更改对象的颜色。例如，您可以创建一个颜色小组件，该小组件可根据搅拌机的温度数据更改场景中饼干混合器的颜色。

使用以下步骤向选定对象添加模型着色器控件。

1. 在分层结构中选择您想要添加小组件的目标对象。按下 + 按钮，然后选择 “模型着色器”。
2. 要添加新的可视规则组，请先按照以下说明创建 ColorRule，然后在规则 ID 对象的 Inspector 面板中选择ColorRule。
3. 选择要将模型着色器绑定到的 E PropertyName ntityId ComponentName、、。

为场景创建视觉规则

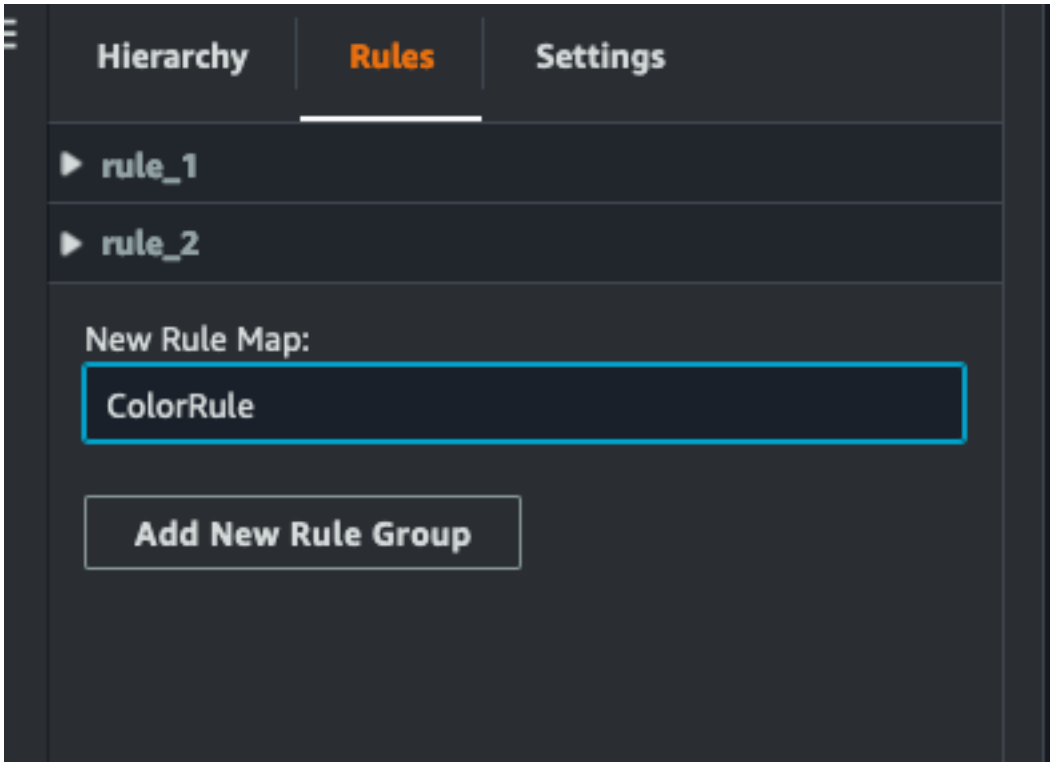
您可以使用可视化规则映射来指定数据驱动的条件，这些条件会更改增强用户界面控件（例如标签或模型着色器）的视觉外观。您可使用已提供的示例规则，获创建自定义规则。以下示例显示了一个可视化规则。


Expression
temperature >= 40
Target
Icon
Error
Remove statement
Expression
temperature >= 20
Target
Icon
Warning
Remove statement
Expression
temperature < 20
Target
Icon
Info
Remove statement
Add new statement
Remove Rule
sampleTimeSeriesColorRule
Rule Id

上图显示了何时根据特定值检查先前定义的 ID 为 “温度” 的数据属性的规则。例如，如果 “温度” 大于或等于 40，则该状态会将标签的外观更改为红色圆圈。在 Grafana 控制面板选择目标后，将填充配置为使用相同数据来源的详细信息面板。

以下程序介绍了如何为网格着色增强 UI 图层添加新的可视规则组。

1. 在控制台的规则选项卡下，输入名称，例如 ColorRule 在文本字段中，然后选择添加新规则组。



2. 为您的用例定义新规则。例如，您可以根据数据属性 “温度” 创建一个，其中报告的值小于 20。对规则表达式使用以下语法：小于等于 <，大于等于 >，小于或等于 <=，大于或等于 >=，等于 ==。（有关更多信息，请参阅 Apache Commons JEXL 语法。）
3. 将目标设置为一种颜色。要定义颜色，例如`#fcba03`，请使用十六进制值。（有关十六进制值的更多信息，请参阅十六进制。）

为您的场景创建标签

标签是添加至特定 `x,y,z` 场景坐标位置的注释。该标签使用实体属性，将场景部分连接至知识图谱。您可通过标签配置场景中某个项目（例如警报）的行为或视觉外观。

> **Note**
>
> 若要为标签添加功能，需要对标签应用视觉规则。

按以下程序将标签添加至您的场景。

1. 在层次结构中选择对象，选择 + 按钮，然后选择 Add Tag（添加标签）。
2. 为标签命名。然后，若要应用视觉规则，请选择视觉群组 ID。
3. 在下拉列表中，选择实体 ID、和。 ComponentName PropertyName
4. 要填充 “数据路径” 字段，请选择 “创建 DataFrameLabel”。

3D 图块模型格式

在场景中使用 3D 图块

如果您在加载 3D 场景时等待时间很长， AWS IoT TwinMaker 或者在浏览复杂的 3D 模型时渲染性能不佳，则可能需要将模型转换为 3D 图块。本节介绍了 3D 切片格式和可用的第三方工具。请继续阅读以确定 3D Tiles 是否适合您的用例并获取入门帮助。

复杂模型用例

如果您的 AWS IoT TwinMaker 场景中的 3D 模型符合以下条件，则可能会导致性能问题，例如加载时间缓慢和导航延迟：

- 大：其文件大小大于 100MB。
- 密集：它由成百上千个不同的网格组成。
- 复杂：网格几何体有数百万个三角形可以形成复杂的形状。

3D 图块格式

3D Tiles 格式是一种用于流式传输模型几何图形和提高 3D 渲染性能的解决方案。它支持在 AWS IoT TwinMaker 场景中即时加载 3D 模型，并通过根据摄像机视图中可见的内容分块加载模型来优化 3D 交互。

3D 图块格式由 Cesi um 创建。Cesium 有一项名为 Cesium Ion 的托管服务，可以将 3D 模型转换为 3D 图块。这是目前创建 3D 图块的最佳解决方案，我们建议您在支持格式的复杂模型中使用此解决方案。您可以在 Cesium 的定价页面上注册 Cesium 并根据您的业务需求选择合适的订阅计划。

要准备可以添加到 AWS IoT TwinMaker 场景中的 3D Tiles 模型，请按照 Cesium Ion 记录的说明进行操作：

- 将模型导入 Cesium Ion

将 Cesium 3D 方块上传到 AWS

将模型转换为 3D 切片后，下载模型文件，然后将其上传到您的 AWS IoT TwinMaker 工作空间 Amazon S3 存储桶：

1. 创建并下载您的 3D Tiles 模型档案。
2. 将存档解压缩到一个文件夹中。
3. 将整个 3D Tiles 文件夹上传到与您的 AWS IoT TwinMaker 工作空间关联的 Amazon S3 存储桶中。（请参阅 Amazon S3 用户指南中的上传对象。）
4. 如果您的 3D 图块模型已成功上传，您将在 AWS IoT TwinMaker 资源库中看到带有类型的 Amazon S3 文件夹路径`Tiles3D`。

> **Note**
>
> AWS IoT TwinMaker 资源库不支持直接上传 3D Tiles 模型。

在中使用 3D 图块 AWS IoT TwinMaker

AWS IoT TwinMaker 知道有任何 3D Tiles 模型上传到您的工作空间 S3 存储桶。模型必须有 a `tileset.json` 且所有依赖文件（.gltf、.b3dm、.i3dm、.i3dm、.cmpt、.pnts）都位于同一 Amazon S3 目录中。Amazon S3 目录路径将与类型一起出现在资源库中`Tiles3D`。

要将 3D Tiles 模型添加到场景中，请执行以下步骤：

1. 在场景编辑器页面，选择加号 (+)，然后选择添加 3D 模型。
2. 在从资源库添加资源窗口中，选择带有类型的 3D Tiles 模型的路径`Tiles3D`，然后选择添加。
3. 单击画布将模型放置在场景中。

3D 图块的区别

3D Tiles 目前不支持几何和语义元数据，这意味着原始模型的网格层次结构不适用于子模型选择功能。您仍然可以向 3D Tiles 模型添加控件，但不能使用针对子模型进行微调的功能：模型着色器、分离的 3D 变换或子模型网格的实体绑定。

对于用作场景背景背景的大型资产，建议使用 3D 图块转换。如果您想进一步分解子模型并添加注释，则应将其提取为单独的 gltF/GLB 资源并直接添加到场景中。这可以通过诸如 Blen der之类的免费和常见的3D工具来完成。

用例示例：

- 你有一个 1GB 的工厂模型，里面有精致的机房和地板、电箱和管道。当相关的属性数据超过阈值时，配电箱和管道需要发出红色光芒。
- 在模型中隔离盒体和管道网格，然后使用 Blender 将其导出到单独的 glTF 中。
- 您可以将没有电气和管道元件的工厂转换为 3D Tiles 模型，然后将其上传到 S3。
- 您可以将 3D Tiles 模型和 glTF 模型同时添加到原点 (0,0,0) 的 AWS IoT TwinMaker 场景中。
- 您可以将模型着色器组件添加到 glTF 的电箱和管道子模型中，以根据属性规则使网格变为红色。

动态场景

AWS IoT TwinMaker 场景通过将场景节点和设置存储在实体组件中来释放知识图谱的力量。使用 AWS IoT TwinMaker 控制台创建动态场景，以便更轻松地管理、构建和渲染 3D 场景。

主要特点：

- 所有 3D 场景节点对象、设置和数据绑定都基于知识图谱查询 “动态” 呈现。
- 如果您在 Grafana 或自定义应用程序中使用只读场景查看器，则可以每隔 30 秒获取场景的更新。

静态场景与动态场景

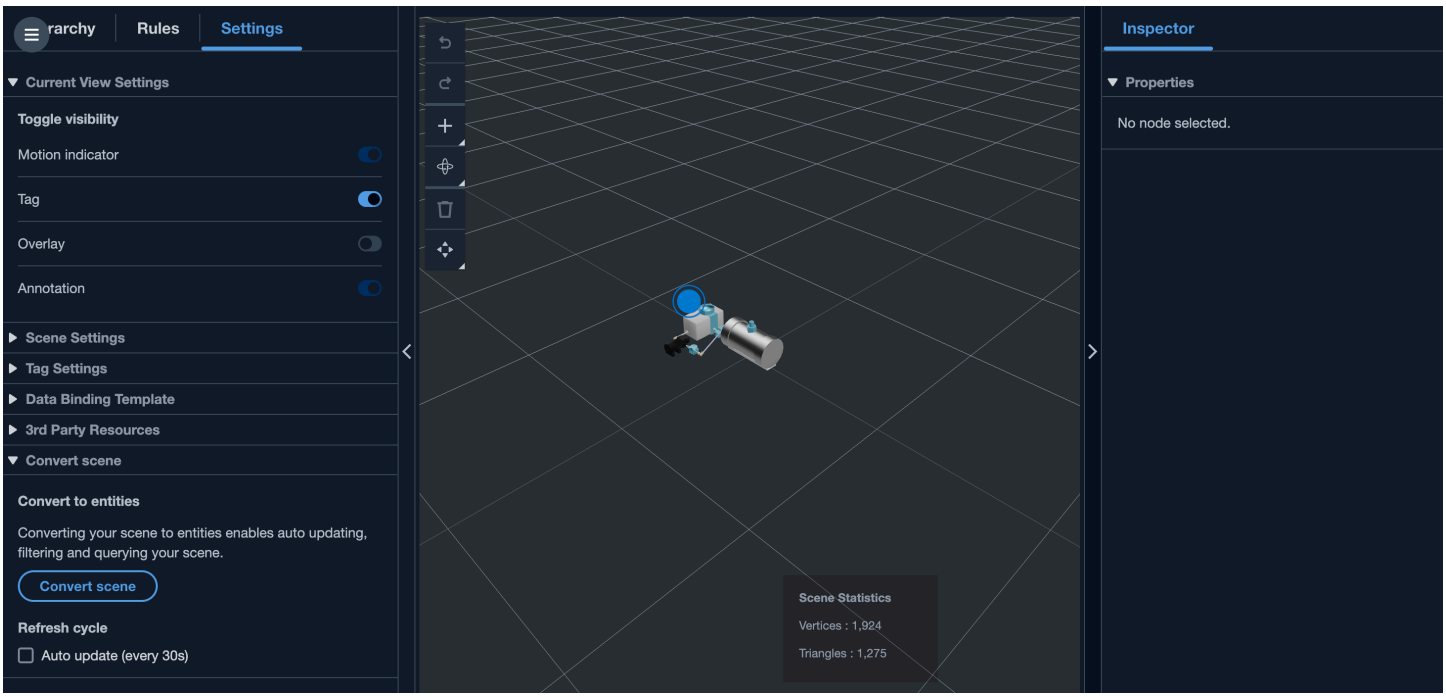
静态场景由存储在 S3 中的场景 JSON 文件组成，该文件包含所有场景节点和设置的详细信息。对场景的任何更改都必须对 JSON 文档进行更改并保存到 S3。如果您有基本的定价计划，则静态场景是唯一的选择。

动态场景由场景 JSON 文件组成，该文件具有场景的全局设置，而所有其他场景节点和节点设置则作为实体组件存储在知识图中。只有标准和分层套餐定价计划才支持动态场景。有关如何升级定价计划的信息，请参阅切换 AWS IoT TwinMaker 定价模式）。

您可以按照以下步骤将现有的静态场景转换为动态场景：

- 在 AWS IoT TwinMaker 控制台中导航到您的场景。
- 在左侧面板上，单击 “设置” 选项卡。
- 展开面板底部的转换场景部分。
- 单击 “转换场景” 按钮，然后单击 “确认”。

> **Warning**
>
> 从静态场景到动态场景的转换是不可逆的。



场景组件类型和实体

为了创建场景特定的实体组件，支持以下 1P 组件类型：

- com.amazon.iottwinmaker.3d.component.camera 一种存储相机小部件设置的组件类型。
- com.amazon.iottwinmaker.3d.component.dataovlay 一种存储注释或标签小部件叠加层设置的组件类型。
- com.amazon.iottwinmaker.3d.component. light 一种存储灯光控件设置的组件类型。

- com.amazon.iottwinmaker.3d.component.modelref 一种组件类型，用于存储场景中使用的 3D 模型的设置和 S 3 位置。
- com.amazon.iottwinmaker.3d.component.modelshader 一种在 3D 模型上存储模型着色器设置的组件类型。
- com.amazon.iottwinmaker.3d.component.motionindicator 一种存储运动指示器小部件设置的组件类型。
- com.amazon.iottwinmaker.3d.component.submodelref 一种存储 3D 模型子模型设置的组件类型。
- com.amazon.iottwinmaker.3d.component.tag 一种存储标签小部件设置的组件类型。
- com.amazon.iottwinmaker.3d.node 一种存储场景节点基本设置的组件类型，例如其 3D 变换、名称和通用属性。

动态场景概念

动态场景实体存储在标有标签$SCENES的全局实体下。每个场景都由根实体和与场景节点层次结构相匹配的子实体层次结构组成。根目录下的每个场景节点都有一个 com.amazon.iottwinmaker.3d.node 组件和一个用于该节点类型的组件（3D 模型、控件等）。

> **Warning**
>
> 请勿手动删除任何场景实体，否则您的场景可能处于损坏状态。如果要部分或全部删除场景，请使用场景编辑器页面添加和删除场景节点，并使用场景页面选择和删除场景。

使用 AWS IoT TwinMaker 用户界面组件创建自定义 Web 应用程序

AWS IoT TwinMaker 为 AWS IoT 应用程序开发人员提供开源 UI 组件。使用这些 UI 组件，开发人员可以构建为其数字双胞胎启用 AWS IoT TwinMaker 功能的自定义 Web 应用程序。

AWS IoT TwinMaker UI 组件是 App AWS IoT lication Kit 的一部分，Application Kit 是一个开源的客户端库，它使物联网应用开发人员能够简化复杂物联网应用的开发

AWS IoT TwinMaker 用户界面组件包括：

- AWS IoT TwinMaker 来源：

  一种数据连接器组件，使您能够检索数据并与 AWS IoT TwinMaker 数据和数字双胞胎进行交互。

  有关更多信息，请参阅 AWS IoT TwinMaker 源 文档。
- 场景查看器：

  通过 `@react-three/fiber` 构建的 3D 渲染组件，用于渲染您的数字孪生并使您与之交互。

  有关更多信息，请参阅场景查看器 文档。
- 视频播放器：

  一个视频播放器组件，允许你通过 Kinesis Video Streams 直播 AWS IoT TwinMaker视频。

  有关更多信息，请参阅视频播放器文档。

要了解有关使用 AWS IoT 应用程序套件的更多信息，请访问AWS IoT 应用程序套件 Github 页面。

有关如何使用 AWS IoT 应用套件启动新 Web 应用程序的说明，请访问官方 IoT App Kit 文档页面。

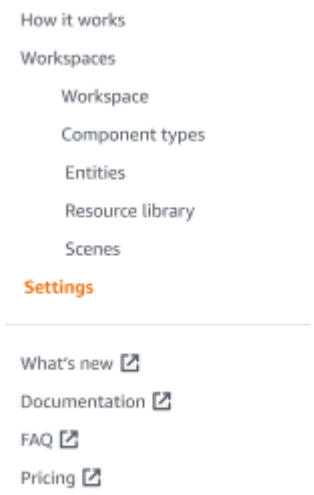
切换 AWS IoT TwinMaker 定价模式

AWS IoT TwinMaker 目前有三种定价模式，即基本、标准或分层套餐。默认为标准定价模式。

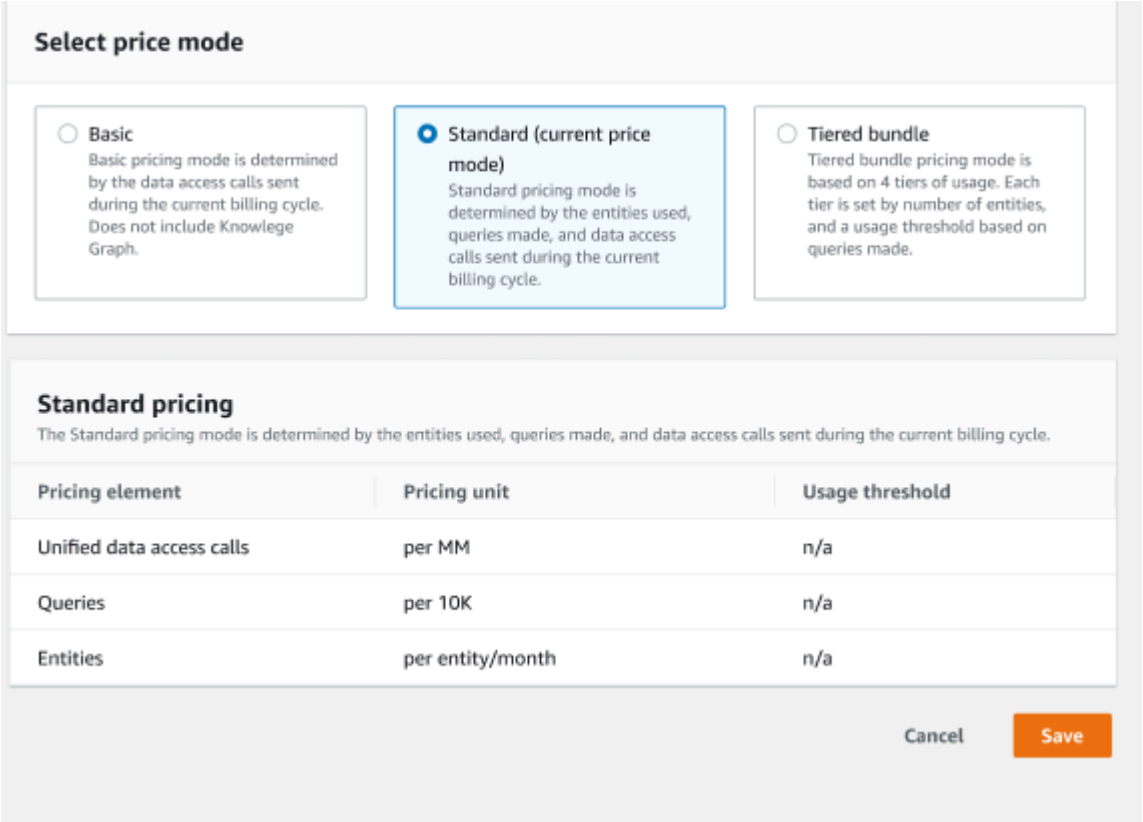
您可以随时从基于使用量的定价模式切换至基于分层的定价模式，但更改将在下一个计费周期开始时生效。从基于使用量的定价模式切换至分层定价模式后，您就无法在接下来的三个使用周期内切换回基于使用量的定价模式。如果您从基础版切换值标准版，则更改将立即生效。有关详情和费用信息，请参阅AWS IoT TwinMaker 定价

此过程向您展示如何在 AWS IoT TwinMaker 控制台切换定价模式：

1. 打开AWS IoT TwinMaker 控制台。
2. 在左侧导航窗格中，选择 “设置”。定价页面打开。



3. 选择 更改定价模式。
4. 选择 标准或 分层套餐模式，如以下屏幕截图所示。



5. 选择Save（保存）以确认您的新定价模式。
6. 您已更改定价模式。

> **Note**
>
> 您可以随时从基于使用量的定价模式切换至基于分层的定价模式，但更改将在下一个计费周期开始时生效。从基于使用量的定价模式切换至分层定价模式后，您就无法在接下来的三个使用周期内切换回基于使用量的定价模式。如果您从基础版切换值标准版，则更改将立即生效。

AWS IoT TwinMaker 知识图谱

AWS IoT TwinMaker 知识图将 AWS IoT TwinMaker 工作区中包含的所有信息组织起来，并以可视化图表的形式呈现。可以对实体、组件和组件类型运行查询，以生成可表明 AWS IoT TwinMaker 资源之间关系的可视化图表。

以下主题说明如何使用和集成知识图谱。

主题

AWS IoT TwinMaker 知识图谱核心概念

该主题涵盖知识图谱功能的关键概念和词汇。

知识图谱的工作原理：

知识图创建实体及其组件与现有CreateEntity或 UpdateEntityAPI 之间的关系。关系只是在实体组件上定义的特殊数据类型关系的属性。 AWS IoT TwinMaker 知识图调用 ExecuteQueryAPI，根据实体中的任何数据或实体之间的关系进行查询。Knowledge graph 使用灵活的 PartiQL 查询语言（许多 AWS 服务都使用），该语言具有新增的图表匹配语法支持，可帮助您编写查询。调用完成后，您可以以表格形式查看结果，也可以将其可视化为连接的节点和边的图表。

知识图谱关键术语：

- 实体图谱：工作区内节点和边缘的集合。
- 节点：工作区中的每个实体都将成为实体图谱中的一个节点。
- 边缘：在实体组件上定义的每个关系属性都将成为实体图谱中的一个边缘。此外，使用实体 parentEntityId 字段定义的分层父子关系也将成为实体图中具有 “isChildOf” 关系名称的边缘。所有边缘都是定向边缘。
- 关系： AWS IoT TwinMaker 关系是实体组件的一种特殊类型的属性。您可以使用 AWS IoT TwinMaker CreateEntity或 UpdateEntityAPI 来定义和编辑关系。在中 AWS IoT TwinMaker，必

须在实体的组件中定义关系。不能将关系定义为孤立的资源。关系从一个实体到另一个实体必须是定向的。

如何运行 AWS IoT TwinMaker 知识图谱查询

在使用 AWS IoT TwinMaker 知识图谱之前，请确保您已完成以下先决条件：

- 创建 AWS IoT TwinMaker 工作空间。您可以在 AWS IoT TwinMaker 控制台中创建工作区。
- 熟悉 AWS IoT TwinMaker的实体组件系统以及如何创建实体。有关更多信息，请参阅 创建您的第一个实体。
- 熟悉 AWS IoT TwinMaker的数据连接器。有关更多信息，请参阅 AWS IoT TwinMaker 数据连接器。

> **Note**
>
> 要使用 AWS IoT TwinMaker 知识图表，您需要处于标准或分层捆绑定价模式。有关更多信息，请参阅 切换 AWS IoT TwinMaker 定价模式。

以下步骤介绍如何编写、运行、保存和编辑查询。

打开查询编辑器

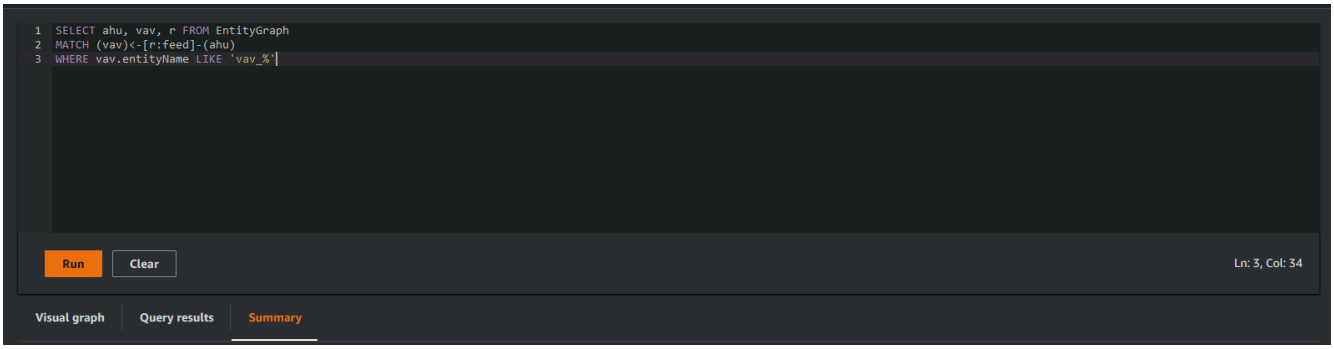
导航到知识图谱查询编辑器

1. 打开AWS IoT TwinMaker 控制台。
2. 打开要在其中使用知识图谱的工作区。
3. 在导航窗格中，选择 查询编辑器。
4. 查询编辑器打开。您现在可以对工作区资源运行查询。

运行查询

运行查询并生成图表

1. 在查询编辑器中，选择Editor（编辑器）选项卡以打开语法编辑器。
2. 在编辑器空间中，编写要对工作区资源运行的查询。



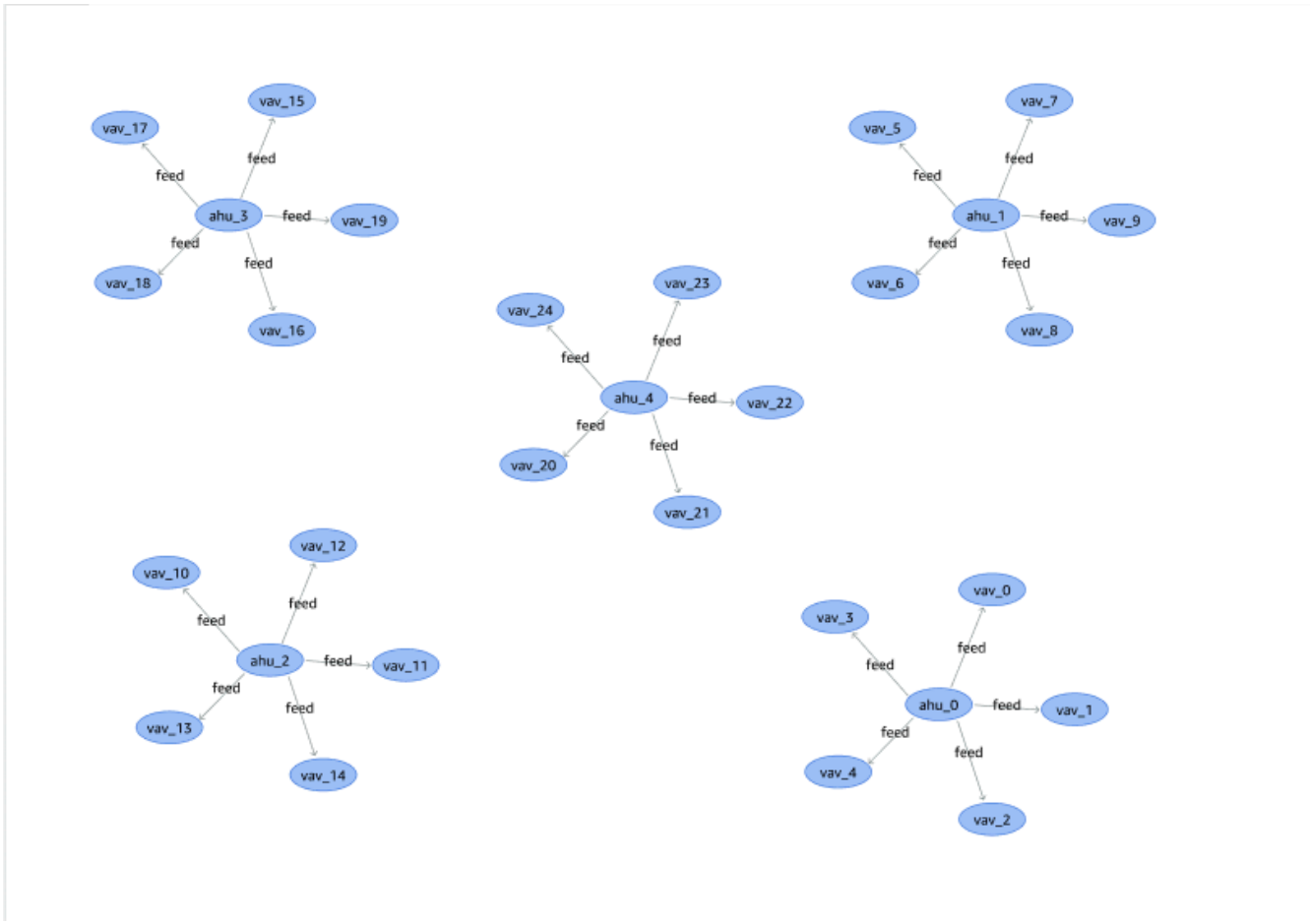
在所示的示例中，请求搜索名称vav_%中包含的实体，然后使用以下代码按它们之间的feed关系组织这些实体。

```
SELECT ahu, vav, r FROM EntityGraph
MATCH (vav)<-[r:feed]-(ahu)
WHERE vav.entityName LIKE 'vav_%'
```

> **Note**
>
> 知识图谱语法使用 P artiQL。有关此语法的信息，请参见AWS IoT TwinMaker 知识图谱其他资源。

3. 选择 “运行查询” 以运行您创建的请求。

   图表将根据您的请求生成。



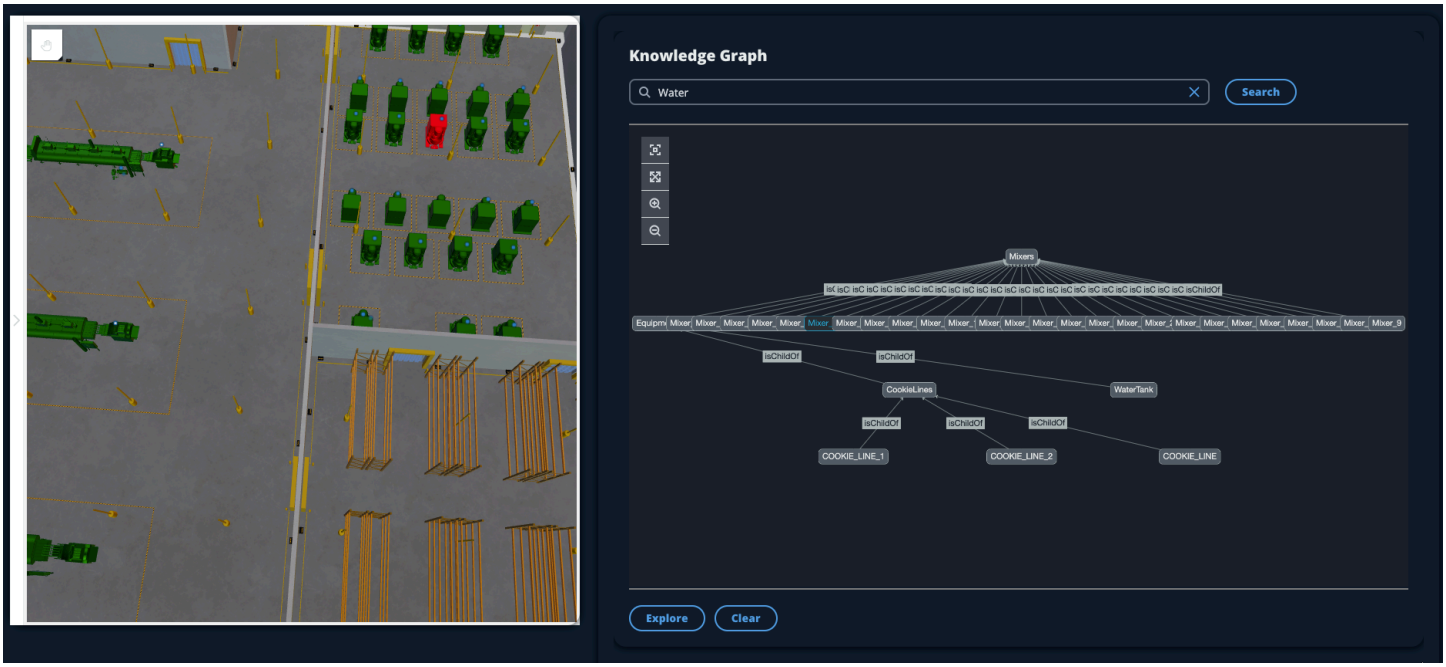
上面显示的示例图表基于步骤 2 中的查询示例。

4. 查询的结果也可以列表的形式呈现。选择结果以列表形式查看查询结果。
5. 或者，选择导出为，以 JSON 或 CSV 格式导出查询结果。

这涵盖了控制台中知识图谱的基本用法。有关展示知识图谱语法的更多信息和示例，请参阅 AWS IoT TwinMaker 知识图谱其他资源。

知识图谱场景集成

您可以使用 AWS IoT 应用套件组件来构建 Web 应用程序，将知识图谱集成到您的 AWS IoT TwinMaker 场景中。这允许您根据场景中存在的 3D 节点（代表您的设备或系统的 3D 模型）生成图表。要创建用于绘制场景中 3D 节点的应用程序，请先将 3D 节点绑定到工作空间中的实体。通过此映射，绘制场景中存在的 3D 模型与工作空间中实体之间的关系 AWS IoT TwinMaker 图。然后，您可以创建 Web 应用程序，在场景中选择 3D 模型，并以图表格式探索它们与其他实体的关系。



有关利用应用套件组件在 AWS IoT TwinMaker 场景中生成图表的运行 Web AWS IoT 应用程序的示例，请参阅 github 上的AWS IoT TwinMaker 示例反应应用程序。

AWS IoT TwinMaker 场景图先决条件

在创建在场景中使用 AWS IoT TwinMaker 知识图谱的 Web 应用程序之前，请完成以下先决条件：

- 创建 AWS IoT TwinMaker 工作空间。您可以在 AWS IoT TwinMaker 控制台中创建工作区。
- 熟悉 AWS IoT TwinMaker的实体组件系统以及如何创建实体。有关更多信息，请参阅 创建您的第一个实体。
- 创建填充有 3D 模型的 AWS IoT TwinMaker 场景。
- 熟悉 AWS IoT TwinMaker的 AWS IoT 应用程序套件组件。有关 AWS IoT TwinMaker 组件的更多信息，请参阅使用 AWS IoT TwinMaker 用户界面组件创建自定义 Web 应用程序。
- 熟悉知识图谱概念和关键术语。请参阅 AWS IoT TwinMaker 知识图谱核心概念。

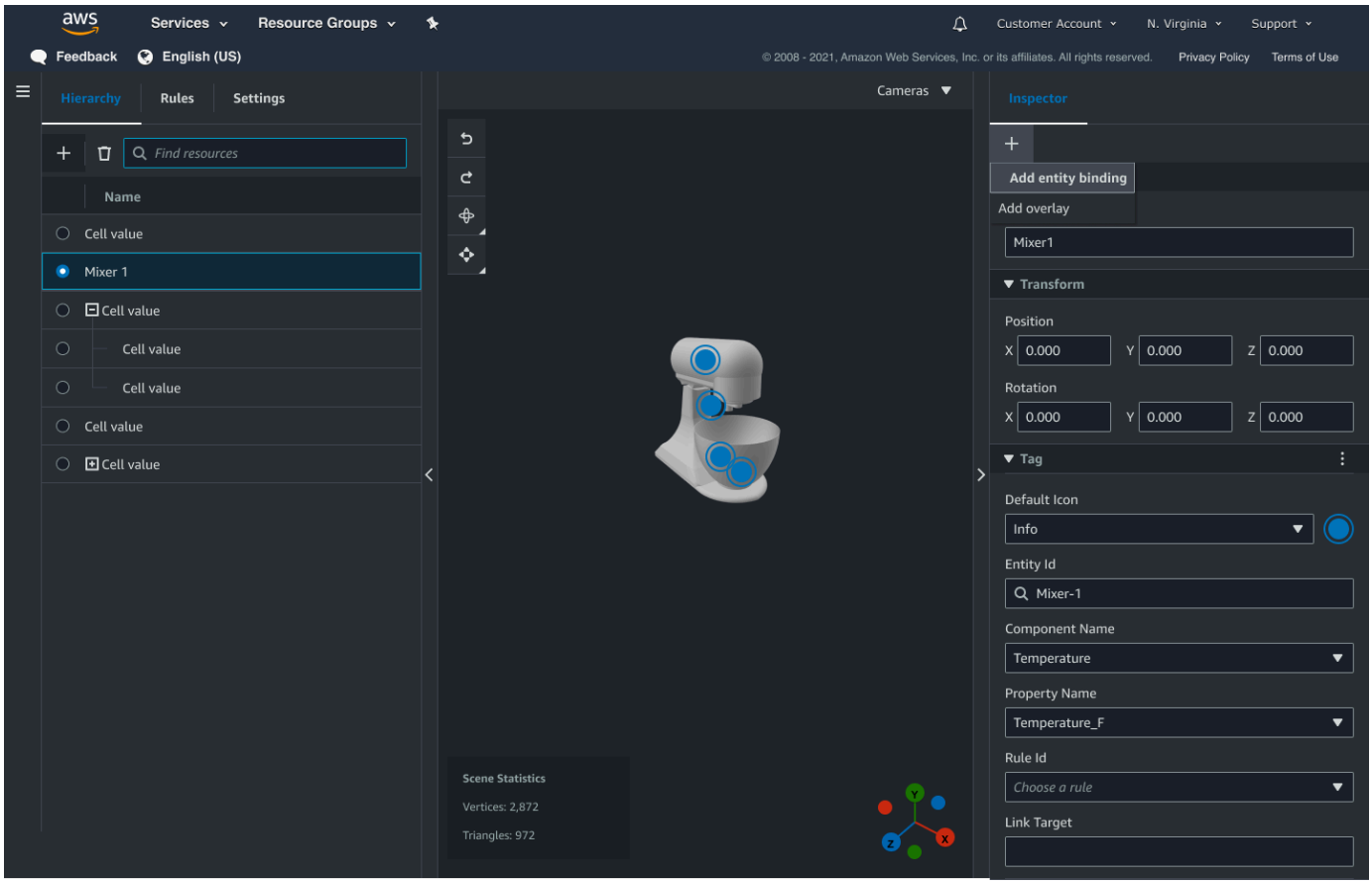
> **Note**
>
> 要使用 AWS IoT TwinMaker 知识图和任何相关功能，您需要处于标准或分层套装定价模式。有关 AWS IoT TwinMaker 定价的更多信息，请参阅切换 AWS IoT TwinMaker 定价模式。

绑定场景中的 3D 节点

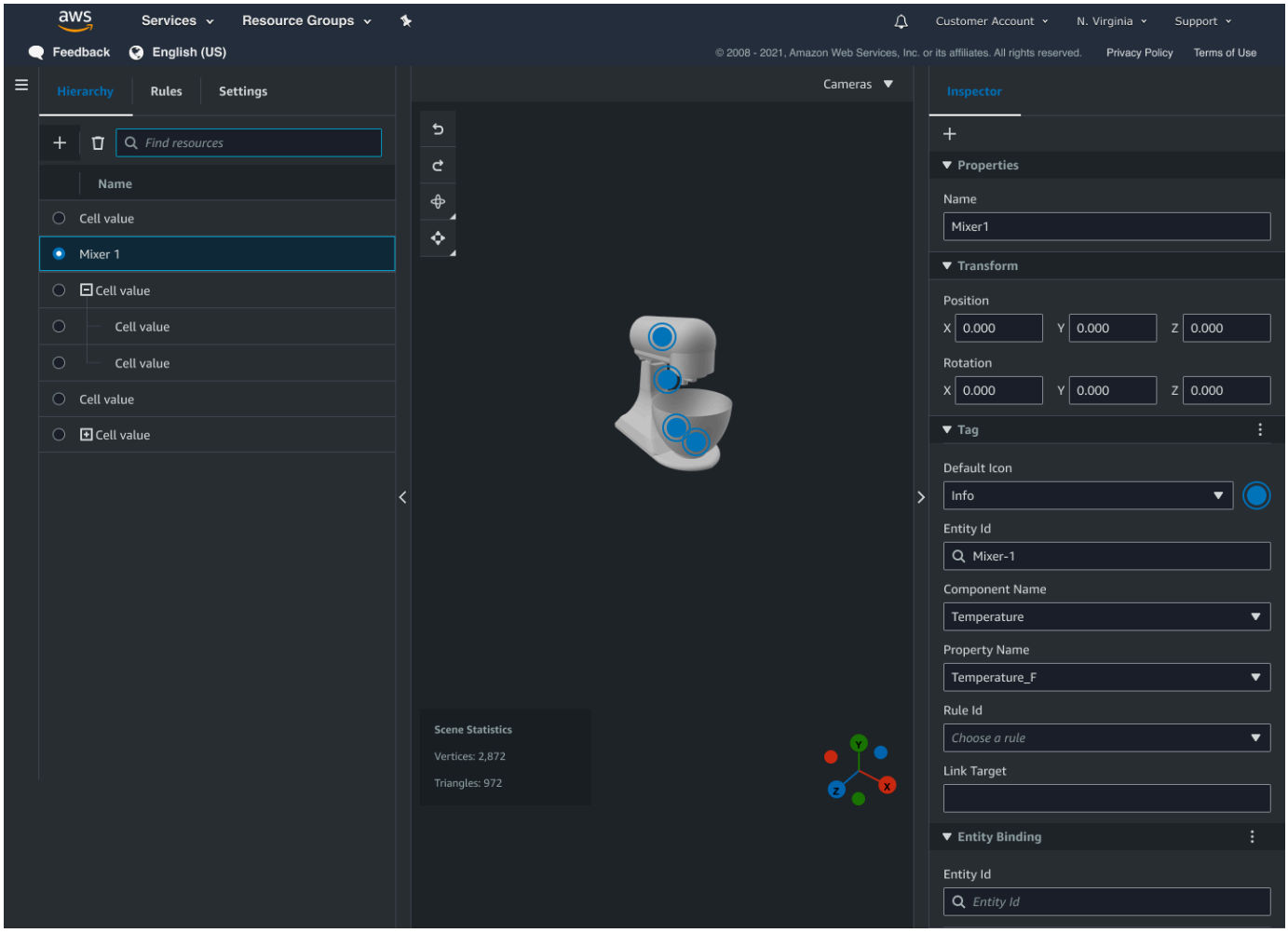
在创建将知识图谱与场景集成的 Web 应用程序之前，请将场景中存在的 3D 模型（称为 3D 节点）绑定到关联的工作区实体。例如，如果您在场景中有一个混音器设备模型，并且有一个名为的相应实体`mixer_0`，则在混音器模型和代表混音器的实体之间创建数据绑定，以便可以绘制模型和实体的图表。

执行数据绑定操作

1. 登录 AWS IoT TwinMaker 控制台。
2. 打开工作区并选择一个包含要绑定的 3D 节点的场景。
3. 在场景编辑器中选择一个节点（3D 模型）。当你选择一个节点时，它将在屏幕右侧打开一个检查器面板。
4. 在检查器面板中，导航到面板顶部，然后选择 + 按钮。然后选择添加实体绑定选项。这将打开一个下拉列表，您可以在其中选择要绑定到当前选定节点的实体。



5. 从数据绑定下拉菜单中，选择要映射到 3D 模型的实体 ID。在 “组件名称” 和 “属性名称” 字段中，选择要绑定的组件和属性。



选择实体 ID、组件名称和属性名称字段后，绑定就完成了。

6. 对要绘制图表的所有模型和实体重复此过程。

> **Note**
>
> 可以对场景标签执行相同的数据绑定操作，只需选择标签而不是实体，然后按照相同的过程将标签绑定到节点。

创建 Web 应用程序

绑定实体后，使用 AWS IoT 应用程序套件库构建一个带有知识图谱控件的 Web 应用程序，该控件允许您查看场景并探索场景节点和实体之间的关系。

使用以下资源创建自己的应用程序：

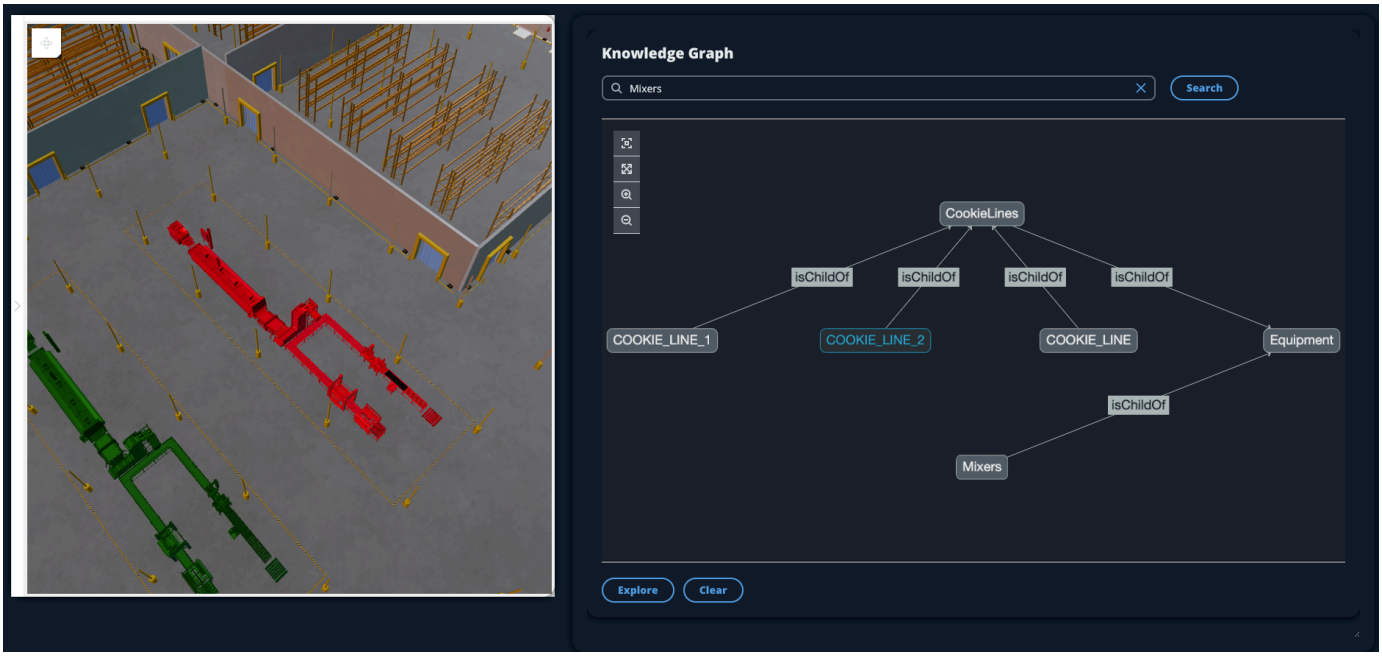
- AWS IoT TwinMaker 示例反应应用程序 github 自述文档。
- github 上的 react 应用程序源代码 AWS IoT TwinMaker示例。
- AWS IoT 应用程序套件入门文档。
- AWS IoT 应用程序套件视频播放器组件文档。
- AWS IoT 应用程序套件 Sc ene Viewer 组件文档。

以下过程演示了 Web 应用程序中场景查看器组件的功能。

> **Note**
>
> 此过程基于 AWS IoT TwinMaker 示例反应 AWS IoT 应用程序中应用套件场景查看器组件的实现。

1. 打开 AWS IoT TwinMaker 示例 React 应用程序的场景查看器组件。在搜索字段中键入实体名称或部分实体名称（区分大小写的搜索），然后选择搜索按钮。如果模型绑定到实体 ID，则场景中的模型将被突出显示，并且该实体的节点将显示在场景查看器面板中。

2. 要生成所有关系的图表，请在场景查看器控件中选择一个节点，然后选择 “浏览” 按钮。



3. 按下清除按钮可清除当前的图表选择并重新开始。

如何在 Grafana 中使用 AWS IoT TwinMaker 知识图谱

本节介绍如何在 AWS IoT TwinMaker Grafana 控制面板中添加查询编辑器面板以运行和显示查询。

AWS IoT TwinMaker 查询编辑器先决条件

在 Grafana 中使用 AWS IoT TwinMaker 知识图谱之前，请完成以下先决条件：

- 创建 AWS IoT TwinMaker 工作空间。您可以在 AWS IoT TwinMaker 控制台中创建工作区。
- 配置 AWS IoT TwinMaker 为与 Grafana 配合使用。有关说明，请参阅AWS IoT TwinMaker Grafana 控制面板集成。

> **Note**
>
> 要使用 AWS IoT TwinMaker 知识图表，您需要处于标准或分层捆绑定价模式。有关更多信息，请参阅 切换 AWS IoT TwinMaker 定价模式。

AWS IoT TwinMaker 查询编辑器权限

要在 Grafana 中使用 AWS IoT TwinMaker 查询编辑器，您必须拥有拥有操作权限的 IAM 角色。`iottwinmaker:ExecuteQuery`将该权限添加到您的工作区仪表板角色中，如以下示例所示：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject"
            ],
            "Resource": [
                "{s3Arn}",
                "{s3Arn}/"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iottwinmaker:Get",
                "iottwinmaker:List",
                "iottwinmaker:ExecuteQuery"
            ],
            "Resource": [
                "{workspaceArn}",
                "{workspaceArn}/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": "iottwinmaker:ListWorkspaces",
            "Resource": "*"
        }
    ]
}
```

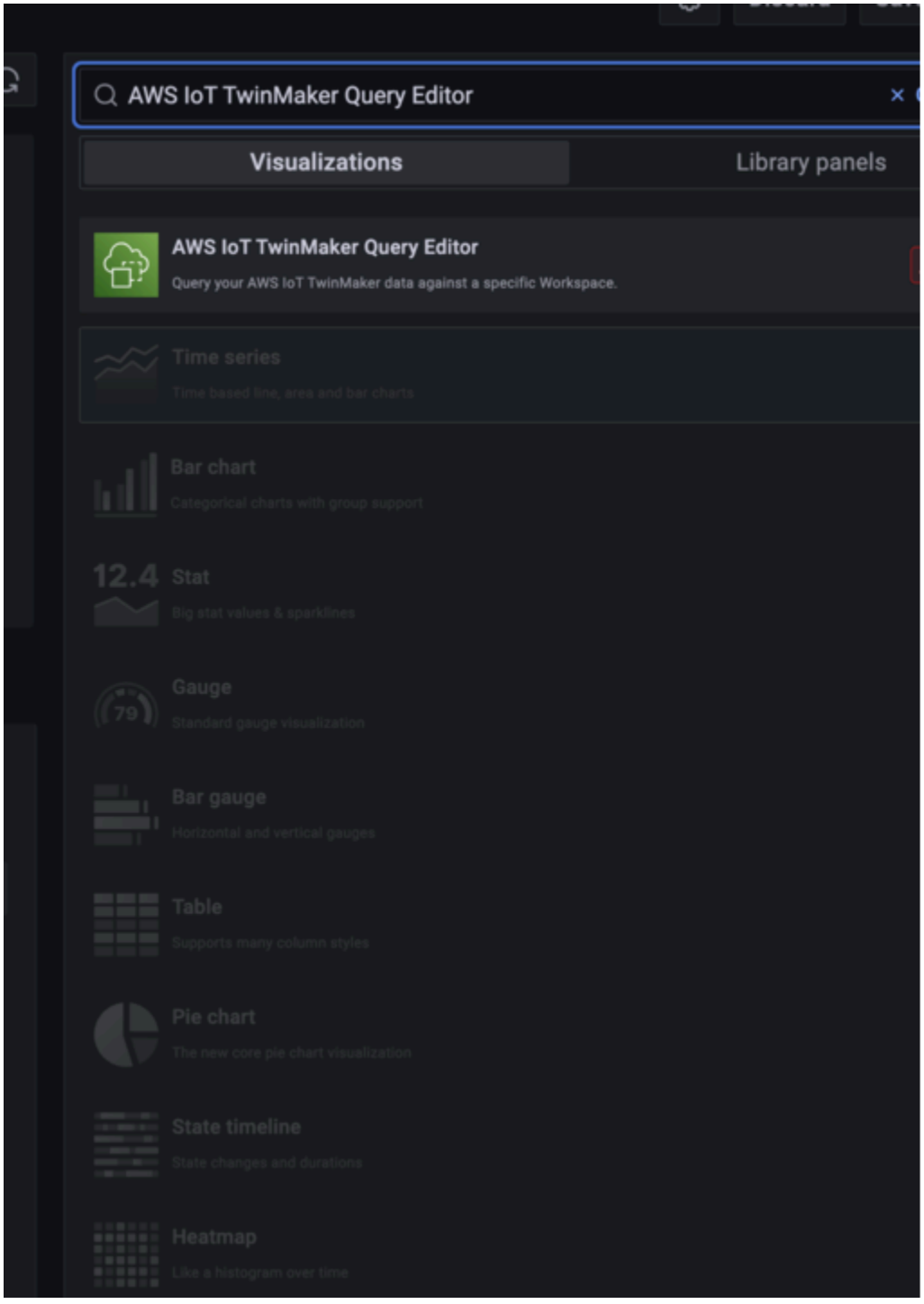
> **Note**
>
> 配置 AWS IoT TwinMaker Grafana 数据源时，请务必使用具有此权限的角色作为代入角色 ARN 字段。添加后，便可以从工作区旁边的下拉列表中选择工作区。

有关更多信息，请参阅 创建控制面板 IAM 角色。

设置 AWS IoT TwinMaker 查询编辑器面板

为知识图谱设置新的 Grafana 仪表板面板

1. 打开你的 AWS IoT TwinMaker Grafana 控制面板。
2. 新建一个控制面板。有关如何创建面板的详细步骤，请参阅 Grafan a 文档中的创建控制面板。
3. 从可视化列表中，选择AWS IoT TwinMaker 查询编辑器。



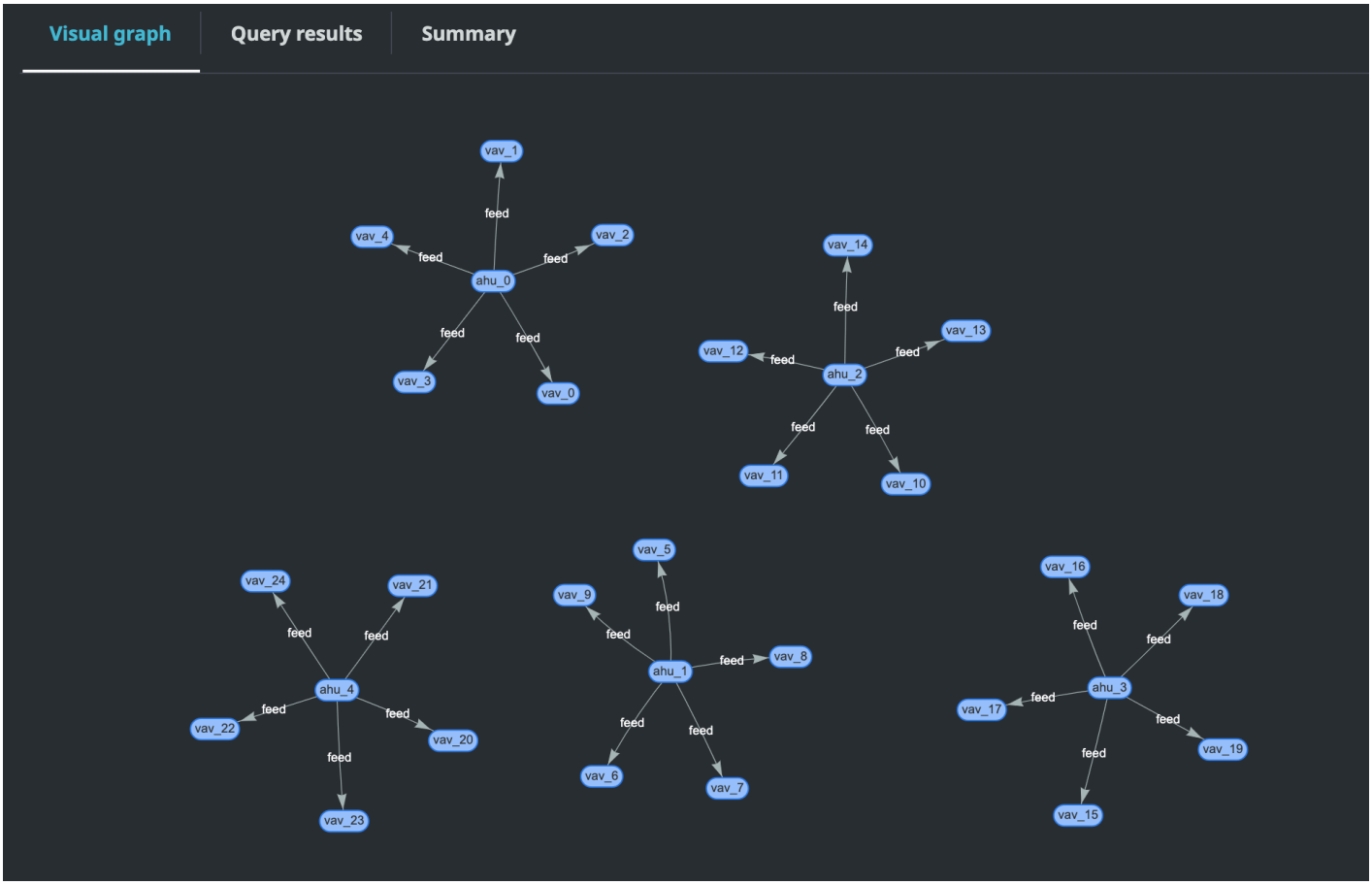
4. 选择要对其运行查询的数据来源。
5. （可选）在提供的字段中为新面板添加名称。
6. 选择“应用”以保存并确认您的新面板。

知识图谱面板的工作方式与 AWS IoT TwinMaker 控制台中提供的查询编辑器类似。您可以运行、编写和清除在面板中提出的查询。有关如何编写查询的更多信息，请参阅AWS IoT TwinMaker 知识图谱其他资源。

如何使用 AWS IoT TwinMaker 查询编辑器

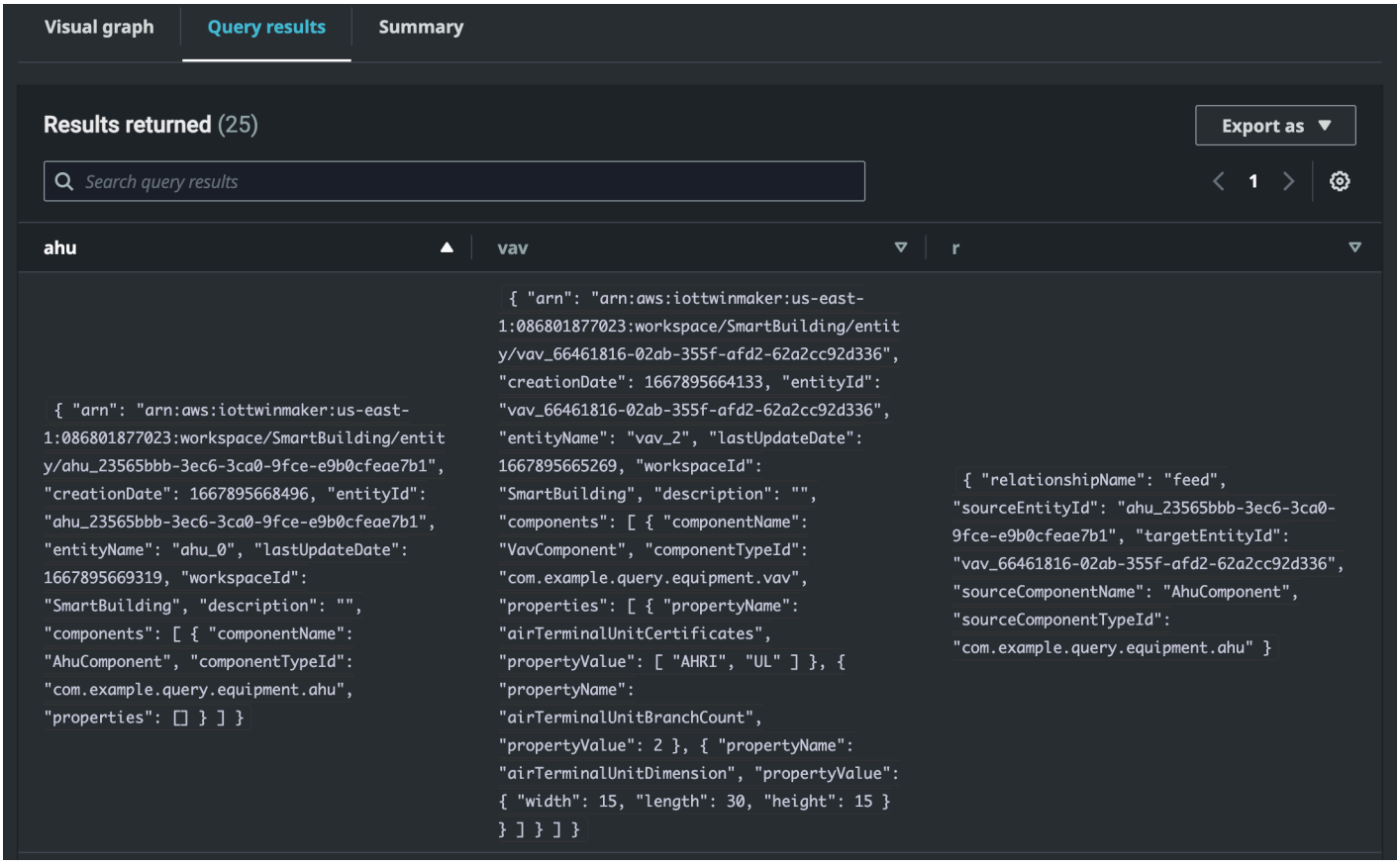
查询结果以三种方式显示，如下图所示：在图表中可视化、在表格中列出或以运行摘要的形式呈现。

- 图表可视化：



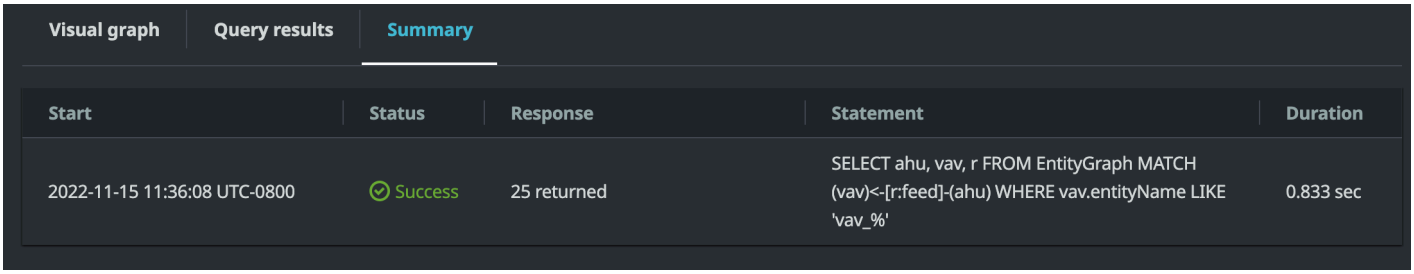
可视化图表仅显示结果中至少有一个关系的查询的数据。该图表将实体显示为节点，将关系显示为图形中的定向边。

- 表格数据：



表格数据格式显示所有查询的数据。可以在表格中搜索特定的结果或结果的子集。数据可以以 JSON 或 CSV 格式导出。

- 运行摘要



运行摘要显示查询和有关查询状态的元数据。

AWS IoT TwinMaker 知识图谱其他资源

本节提供了用于在知识图谱中编写查询的 PartiQL 语法的基本示例，以及提供知识图谱数据模型相关信息的 PartiQL 文档的链接。

- PartiQL 图数据模型文档
- PartiQL 图查询文档

这组示例显示了基本查询及其响应。以此作为参考来编写自己的查询。

基本查询

- 使用筛选器获取所有实体

```
SELECT entity
FROM EntityGraph MATCH (entity)
WHERE entity.entityName = 'room_0'
```

此查询返回工作空间中名称为的所有实体room_0。

FROM子句：EntityGraph是包含工作空间中所有实体及其关系的图表集合。此集合是 AWS IoT TwinMaker 根据工作区中的实体自动创建和管理的。

MATCH 子句：指定与图表一部分相匹配的模式。在这种情况下，该模式 (entity) 匹配图中的每个节点，并绑定到实体变量。FROM 子句后面必须是 MATCH 子句。

WHERE子句：在节点的entityName字段上指定一个过滤器，其中值必须匹配room_0。

SELECT子句：指定entity变量，以便返回整个实体节点。

响应：

```
{
  "columnDescriptions": [
    {
      "name": "entity",
      "type": "NODE"
    }
  ],
  "rows": [
    {
      "rowData": [
        {
          "arn": "arn:aws:iottwinmaker:us-east-1: 577476956029: workspace /
 SmartBuilding8292022 / entity / room_18f3ef90 - 7197 - 53 d1 - abab -
 db9c9ad02781 ",
          "creationDate": 1661811123914,
```

```
          "entityId": "room_18f3ef90-7197-53d1-abab-db9c9ad02781",
          "entityName": "room_0",
          "lastUpdateDate": 1661811125072,
          "workspaceId": "SmartBuilding8292022",
          "description": "",
          "components": [
            {
              "componentName": "RoomComponent",
              "componentTypeId": "com.example.query.construction.room",
              "properties": [
                {
                  "propertyName": "roomFunction",
                  "propertyValue": "meeting"
                },
                {
                  "propertyName": "roomNumber",
                  "propertyValue": 0
                }
              ]
            }
          ]
        }
      ]
    }
  ]
}
```

columnDescriptions返回有关该列的元数据，例如名称和类型。返回的类型是 NODE。这表示整个节点已返回。该类型的其他值可以是EDGE表示关系的值，也可以是表示标量值（例如整数或字符串）的值。VALUE

rows 返回行列表。由于只有一个实体匹配，因此会返回一个包含实体中所有字段的 rowData。

> **Note**
>
> 与只能返回标量值的 SQL 不同，可以使用 PartiQL 返回一个对象（作为 JSON）。

每个节点都包含所有实体级字段（例如entityId、arn和）components、组件级字段（例如和）以及属性级字段（例如componentTypeId和properties）componentName，全部作为嵌套的 JSO propertyName N propertyValue。

- 使用筛选器获取所有关系：

```
SELECT relationship
FROM EntityGraph MATCH (e1)-[relationship]->(e2)
WHERE relationship.relationshipName = 'isLocationOf'
```

此查询返回工作区中具有关系名称 isLocationOf 的所有关系。

MATCH子句：指定一种模式，该模式匹配两个节点（由指示()），这两个节点通过定向边（由指示-[]->）连接并绑定到名为的变量relationship。

子WHERE句：在边缘的relationshipName字段上指定一个过滤器，其中值为isLocationOf。

SELECT 子句：指定关系变量，以便返回整个边缘节点。

响应

```
{
    "columnDescriptions": [{
        "name": "relationship",
        "type": "EDGE"
    }],
    "rows": [{
        "rowData": [{
            "relationshipName": "isLocationOf",
            "sourceEntityId": "floor_83faea7a-ea3b-56b7-8e22-562f0cf90c5a",
            "targetEntityId": "building_4ec7f9e9-e67e-543f-9d1b- 235df7e3f6a8",
            "sourceComponentName": "FloorComponent",
            "sourceComponentTypeId": "com.example.query.construction.floor"
        }]
    },
        ... //rest of the rows are omitted
    ]
}
```

中的列的类型columnDescriptions是EDGE。

每个都rowData代表一条带有类似字段的边缘relationshipName。这与实体上定义的关系属性名称相同。sourceEntityId，sourceComponentName并sourceComponentTypeId提供有关在哪个实体和组件上定义关系属性的信息。targetEntityId指定此关系指向哪个实体。

- 获取与特定实体有特定关系的所有实体

```
SELECT e2.entityName
      FROM EntityGraph MATCH (e1)-[r]->(e2)
      WHERE relationship.relationshipName = 'isLocationOf'
      AND e1.entityName = 'room_0'
```

此查询返回与该实体有关isLocationOf系的所有实体的所有room_0实体名称。

MATCH子句：指定一种模式，该模式与具有定向边 (e2) 的任意两个节点 (e1,r) 相匹配。

WHERE 子句：指定对关系名称和来源实体名称的筛选器。

子SELECT句：返回e2节点中的entityName字段。

响应

```
{
  "columnDescriptions": [
    {
       "name": "entityName",
       "type": "VALUE"
    }
  ],
   "rows": [
    {
       "rowData": [
         "floor_0"
      ]
    }
  ]
}
```

在 ColumnDescriptions 中，列的类型是，VALUE因为entityName是字符串。

返回一个floor_0实体。

匹配

MATCH子句中支持以下模式：

- 匹配指向节点 'a' 的节点 'b'：

```
FROM EntityGraph MATCH (a)-[rel]-(b)
```

- 匹配指向节点 “b” 的节点 “a”：

```
FROM EntityGraph MATCH (a)-[]->(b)
```

假设不需要在关系上指定筛选器，则不存在绑定到关系的变量。

- 匹配指向节点 “b” 的节点 “a” 和指向节点 “a” 的节点 “b”：

```
FROM EntityGraph MATCH (a)-[rel]-(b)
```

这将返回两个匹配项：一个从 'a' 到 'b'，另一个从 'b' 到 'a'，因此建议尽可能使用定向边。

- 关系名称也是属性图的标签EntityGraph，因此您可以简单地在冒号 (:) 后面指定关系名称，而不必在WHERE子句rel.relationshipName中指定过滤器。

```
FROM EntityGraph MATCH (a)-[:isLocationOf]-(b)
```

- 链接：可以链接多个模式以在多个关系匹配。

```
FROM EntityGraph MATCH (a)-[rel1]->(b)-[rel2]-(c)
```

- 可变的跳跃模式也可以跨越多个节点和边缘：

```
FROM EntityGraph MATCH (a)-[]->{1,5}(b)
```

此查询匹配任何具有从节点 'a' 传出边缘在 1 到 5 个跳数内的模式。容许的限定符为：

{m,n} - 介于 m 和 n 次重复之间

{m,} - m 次或更多次重复。

FROM：

实体节点可以包含嵌套数据，例如其本身包含更多嵌套数据（例如属性）的组件。它们可以通过取消嵌套 MATCH 模式的结果来访问。

```
SELECT e
FROM EntityGraph MATCH (e), e.components AS c, c.properties AS p
WHERE c.componentTypeId = 'com.example.query.construction.room',
```

```
AND p.propertyName = 'roomFunction'
AND p.propertyValue = 'meeting'
```

通过点划 . 变量来访问嵌套的字段。逗号 (,) 用于取消嵌套（或连接）实体与内部的组件，然后解除这些组件内部的属性的嵌套（或连接）。 AS用于将变量绑定到未嵌套的变量，以便它们可以在 o WHERE r SELECT 子句中使用。此查询返回在组件类型为 ID com.example.query.construction.room 的组件中包含名为 roomFunction、值为 meeting 的属性的所有实体

要访问一个字段的多个嵌套字段，例如实体中的多个组件，请使用逗号表示法进行连接。

```
SELECT e
FROM EntityGraph MATCH (e), e.components AS c1, e.components AS c2
```

SELECT：

- 返回一个节点：

  ```
  SELECT e
  FROM EntityGraph MATCH (e)
  ```

- 返回一个边缘：

  ```
  SELECT r
  FROM EntityGraph MATCH (e1)-[r]->(e2)
  ```

- 返回标量值：

  ```
  SELECT floor.entityName, room.description, p.propertyValue AS roomfunction
  FROM EntityGraph MATCH (floor)-[:isLocationOf]-(room),
  room.components AS c, c.properties AS p
  ```

  通过使用 AS 对输出字段名称进行别名化来设置其格式。此处返回的是 roomfunction，则不是响应中作为列名称的 propertyValue。

- 返回别名：

  ```
  SELECT floor.entityName AS floorName, luminaire.entityName as luminaireName
  FROM EntityGraph MATCH (floor)-[:isLocationOf]-(room)-[:hasPart]-
  (lightingZone)-[:feed]-(luminaire)
  WHERE floor.entityName = 'floor_0'
  ```

```
AND luminaire.entityName like 'lumin%'
```

强烈建议使用别名，以保持明确性，提高可读性，并避免查询中出现任何歧义。

WHERE：

- 支持的逻辑运算符有ANDNOT、和OR。
- 支持的比较运算符是 <、<=、>、=>、= 和 !=。
- 如果要在同一个字段上指定多个OR条件，请使用IN关键字。
- 根据实体、组件或属性字段进行筛选：

```
FROM EntityGraph MATCH (e), e.components AS c, c.properties AS p
WHERE e.entityName = 'room_0'
AND c.componentTypeId = 'com.example.query.construction.room',
AND p.propertyName = 'roomFunction'
AND NOT p.propertyValue = 'meeting'
OR p.propertyValue = 'office'
```

- 对configuration属性进行筛选。unit这是配置图中的键，Celsius也是值。

```
WHERE p.definition.configuration.unit = 'Celsius'
```

- 检查地图属性是否包含给定的键和值：

```
WHERE p.propertyValue.length = 20.0
```

- 检查地图属性是否包含给定的键：

```
WHERE NOT p.propertyValue.length IS MISSING
```

- 检查列表属性是否包含给定值：

```
WHERE 10.0 IN p.propertyValue
```

- 使用 lower() 函数进行不区分大小写的比较。默认情况下，所有比较都区分大小写。

```
WHERE lower(p.propertyValue) = 'meeting'
```

LIKE：

在不知道某个字段的确切值但可以对指定字段执行全文搜索时很有用。% 代表零或多个。

```
WHERE e.entityName LIKE '%room%'
```

- 中缀搜索：%room%
- 前缀搜索：room%
- 后缀搜索：%room
- 如果你的值中有 '%'，则在中输入一个转义字符，LIKE并使用指定转义字符ESCAPE。

```
WHERE e.entityName LIKE 'room\%' ESCAPE '\'
```

DISTINCT：

```
SELECT DISTINCT c.componentTypeId
FROM EntityGraph MATCH (e), e.components AS c
```

- DISTINCT 关键字可消除最终结果中的重复项。

  DISTINCT 对复杂数据类型不支持。

COUNT

```
SELECT COUNT(e), COUNT(c.componentTypeId)
FROM EntityGraph MATCH (e), e.components AS c
```

- COUNT关键字计算查询结果中的项目数。
- COUNT不支持嵌套复杂字段和图表模式字段。
- COUNTDISTINCT和嵌套查询不支持聚合。

  例如，不支持 COUNT(DISTINCT e.entityId)。

路径

使用路径投影进行查询时支持以下模式投影：

- 可变跳查询

  ```
  SELECT p FROM EntityGraph MATCH p = (a)-[]->{1, 3}(b)
  ```

  此查询匹配并投影任何模式的节点元数据，这些模式的出站边缘在 1 到 3 跳之间。

- 固定跳跃查询

```
SELECT p FROM EntityGraph MATCH p = (a)-[]->(b)<-[]-(c)
```

此查询将实体和传入边的元数据匹配并投影到 b。

- 非定向查询

```
SELECT p FROM EntityGraph MATCH p = (a)-[]-(b)-[]-(c)
```

此查询以 1 跳模式匹配并投影通过 b 连接 a 和 c 的节点的元数据。

```
{
    "columnDescriptions": [
        {
            "name": "path",
            "type": "PATH"
        }
    ],
    "rows": [
        {
            "rowData": [
                {
                    "path": [
                        {
                            "entityId": "a",
                            "entityName": "a"
                        },
                        {
                            "relationshipName": "a-to-b-relation",
                            "sourceEntityId": "a",
                            "targetEntityId": "b"
                        },
                        {
                            "entityId": "b",
                            "entityName": "b"
                        }
                    ]
                }
            ]
        },
        {
            "rowData": [
                {
```

```
                    "path": [
                        {
                            "entityId": "b",
                            "entityName": "b"
                        },
                        {
                            "relationshipName": "b-to-c-relation",
                            "sourceEntityId": "b",
                            "targetEntityId": "c"
                        },
                        {
                            "entityId": "c",
                            "entityName": "c"
                        }
                    ]
                }
            ]
        }
    ]
}
```

此*PATH*查询响应仅包含标识通过 b 在 a 和 c 之间每条路径/模式的所有节点和边缘的元数据。

限制和抵消：

```
SELECT e.entityName
FROM EntityGraph MATCH (e)
WHERE e.entityName LIKE 'room_%'
LIMIT 10
OFFSET 5
```

LIMIT 指定要在查询中返回的结果数，OFFSET 指定要跳过的结果数。

限制和最大结果：

以下示例显示了一个查询，该查询总共返回 500 个结果，但每次 API 调用一次仅显示 50 个结果。此模式可用于需要限制显示结果数量的地方，例如，当用户界面中只能显示 50 个结果时。

```
aws iottwinmaker execute-query \
--workspace-id exampleWorkspace \
--query-statement "SELECT e FROM EntityGraph MATCH (e) LIMIT 500"\
--max-results 50
```

- LIMIT关键字会影响查询并限制结果行。如果您需要在不限制返回结果总数的情况下控制每次 API 调用返回的结果数，请使用LIMIT。
- max-results是 ExecuteQuery API 操作的可选参数。 max-results仅适用于 API 以及如何在上述查询的范围内读取结果。

  max-results在查询中使用允许您在不限制返回结果的实际数量的情况下减少显示结果的数量。

下面的查询会遍历下一页的结果。此查询使用 ExecuteQuery API 调用返回第 51-100 行，其中下一页结果由指定，在本例中标记为:。next-token "H7kyGmvK376L"

```
aws iottwinmaker execute-query \
--workspace-id exampleWorkspace \
--query-statement "SELECT e FROM EntityGraph MATCH (e) LIMIT 500"\
--max-results 50
--next-token "H7kyGmvK376L"
```

- 该next-token字符串指定下一页的结果。有关更多信息，请参阅 ExecuteQueryAPI 操作。

AWS IoT TwinMaker 知识图谱查询有以下限制：

限制名称	限额	可调整
查询执行超时值	10 秒	不支持
最大跃点数	10	是
自我JOIN的最大数量	20	是
投影字段最大数量	20	是
条件表达式的最大数量 (AND,OR,NOT)	10	是
LIKE表达式模式的最大长度（包括通配符和转义符）	20	是
一个IN子句中可以指定的最大项目数	10	是

限制名称	限额	可调整
的最大值 OFFSET	3000	支持
的最大值 LIMIT	3000	支持
遍历的最大值 (+) OFFSET LIMIT	3000	支持

使用 AWS IoT SiteWise实现资产同步

AWS IoT TwinMaker 支持资产和资产模型的资产同步（ AWS IoT SiteWise 资产同步）。使用 AWS IoT SiteWise 组件类型，资产同步获取现有 AWS IoT SiteWise 资产和资产模型，并将这些资源转换为 AWS IoT TwinMaker 实体、组件和组件类型。以下各节将向您介绍如何配置资产同步，以及哪些 AWS IoT SiteWise 资产和资产模型可以同步到您的 AWS IoT TwinMaker 工作空间。

主题

通过 AWS IoT SiteWise使用资产同步

本主题向您展示如何开启和配置 AWS IoT SiteWise 资产同步。根据您使用的工作空间类型，按照相应的步骤进行操作。

> **Important**
>
> 有关自定义工作空间和默认工作区之间的区别的信息，请参阅the section called “自定义工作空间和默认工作空间之间的区别”。

主题

使用自定义工作区

在开启资产同步前，请查看这些先决条件。

先决条件

在使用之前 AWS IoT SiteWise，请确保您已完成以下操作：

- 你有一个 AWS IoT TwinMaker 工作空间。
- 您已在 AWS IoT SiteWise中创建部分资产和资产模型。有关更多信息，请参阅创建资产模型。
- 您已创建一个 IAM 角色，该角色具有以下 AWS IoT SiteWise 操作的读取权限：`ListAssetsListAssetModels`、`DescribeAsset`、和`DescribeAssetModel`。

  IAM 角色还必须具有以下 AWS IoT TwinMaker写入权限：`CreateEntity`、`UpdateEntity`、`DeleteEntity`、`CreateComponentType`、`UpdateComponer`和 `DeleteComponentType`。

  您可以使用以下 IAM 角色作为所需角色模板：

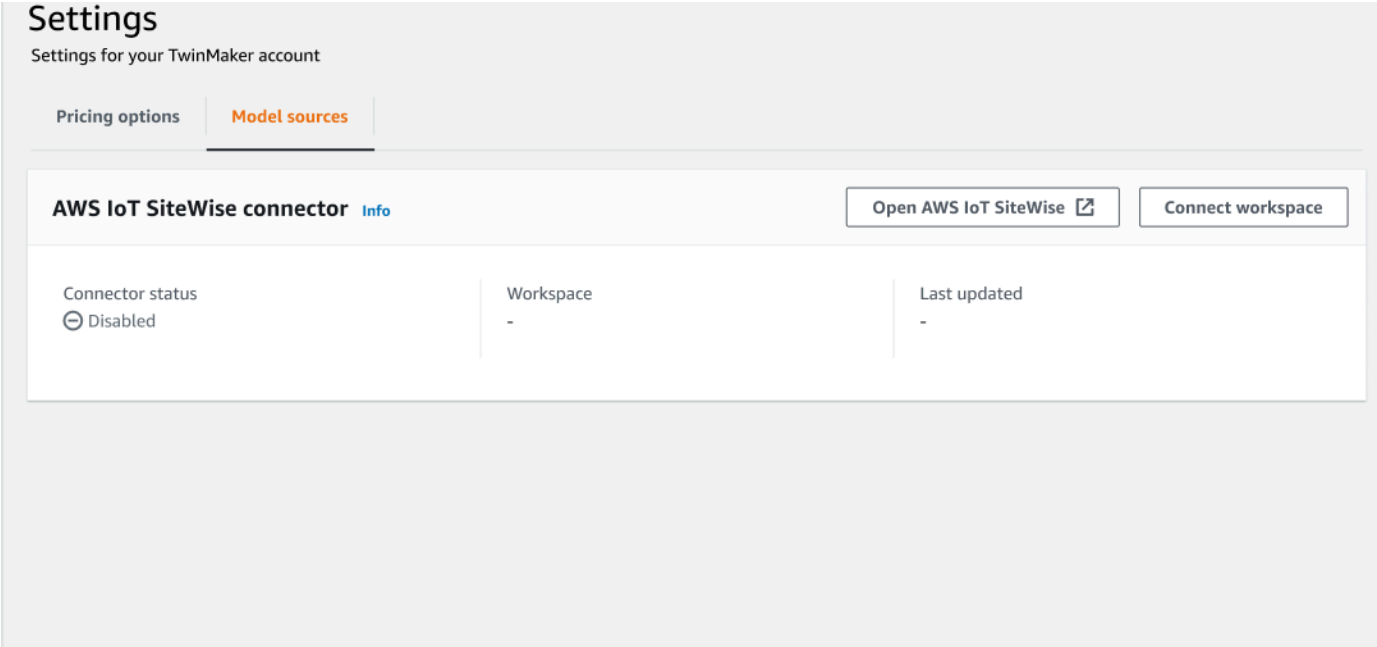
```
// trust relationships
                        {
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                 "Service": [
                    "iottwinmaker.amazonaws.com",
                    "iotsitewise.amazonaws.com"
                ]
            },
            "Action": "sts:AssumeRole"
        }
    ]
}

// permissions
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "iottwinmaker:*",
            "Resource": "*"
        },
```

```
        {
            "Effect": "Allow",
            "Action": [
                "iotsitewise:Describe*",
                "iotsitewise:List*"
            ],
            "Resource": "*"
        }
    ]
}
```

按照以下程序开启和配置 AWS IoT SiteWise 资产同步。

1. 在 AWS IoT TwinMaker 控制台，导航至 Settings（设置）页面。
2. 打开 模型来源选项卡。

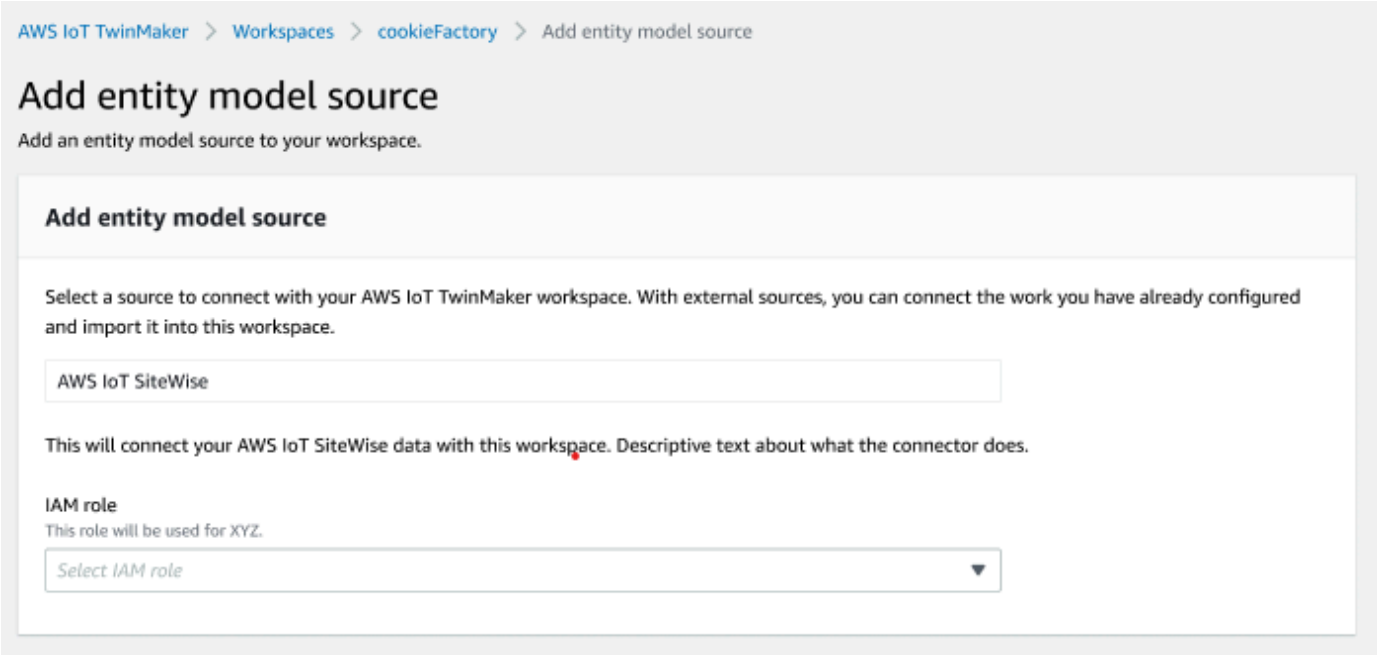


3. 选择 Connect 工作空间，将您的 AWS IoT TwinMaker 工作空间链接到您的 AWS IoT SiteWise 资产。

> **Note**
>
> 您只能在单个 AWS IoT TwinMaker 工作区中使用资产同步。如果要在其他工作区内同步，必须断开原有工作区连接，然后连接至另一工作区。

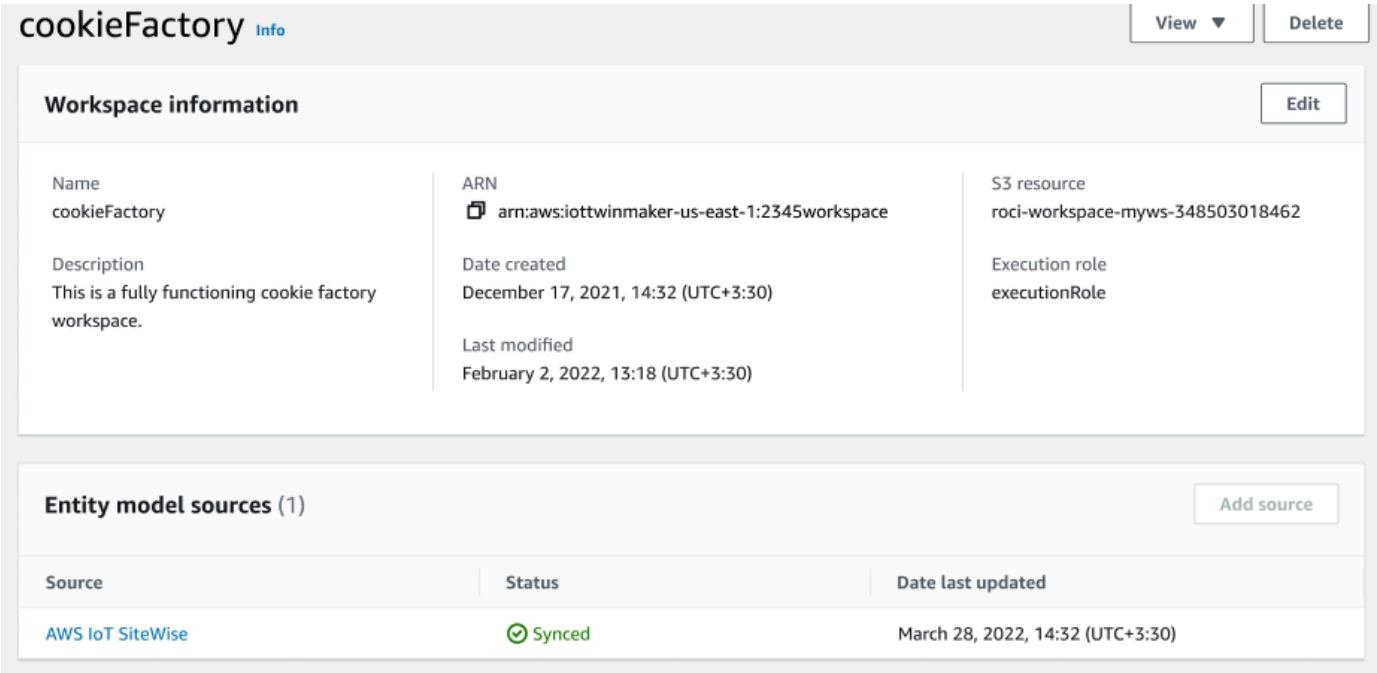
4. 接下来，导航至要在其中使用资产同步的工作区。

5. 选择 Add sources (添加源)。打开添加试题模型来源页面。



6. 在添加实体模型来源页面，确认源字段显示 AWS IoT SiteWise。选择您所创建的 IAM 角色作为IAM 角色的先决条件。

7. 现在，您已开启 AWS IoT SiteWise 资产同步。您应该会在选定工作区 页面顶部看到确认横幅，确认资产同步处于活动状态。现在，您还应该看到实体模型来源部分列出同步来源。



使用物联网 SiteWiseDefaultWorkspace

当您选择加入AWS IoT SiteWiseAWS IoT TwinMaker 集成时，系统会创建一个名IoTSiteWiseDefaultWorkspace为的默认工作区并自动与其 AWS IoT SiteWise同步。

您也可以使用 AWS IoT TwinMaker CreateWorkspace API 创建名为的工作空间IoTSiteWiseDefaultWorkspace。

先决条件

在创建之前IoTSiteWiseDefaultWorkspace，请确保您已完成以下操作：

- 创建 AWS IoT TwinMaker 服务相关角色。请参阅将服务相关角色用于 AWS IoT TwinMaker了解更多信息。
- 通过 https://console.aws.amazon.com/iam/ 打开 IAM 控制台。

  查看该角色或用户，并验证其是否有权这样做iotsitewise:EnableSiteWiseIntegration。

  如果需要，为角色或用户添加权限：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "iotsitewise:EnableSiteWiseIntegration",
            "Resource": "*"
        }
    ]
}
```

自定义工作空间和默认工作空间之间的区别

> **Important**
>
> 新 AWS IoT SiteWise 功能（例如）CompositionModel仅在中可用IoTSiteWiseDefaultWorkspace。我们建议您使用默认工作区而不是自定义工作区。

使用时IoTSiteWiseDefaultWorkspace，与使用带资产同步功能的自定义工作区有一些显著的区别。

- 创建默认工作空间时，Amazon S3 位置和 IAM 角色是可选的。

> **Note**
>
> 您可以使用UpdateWorkspace提供 Amazon S3 位置和 IAM 角色。

- IoTSiteWiseDefaultWorkspace没有资源数量限制，无法同步 AWS IoT SiteWise 资源 AWS IoT TwinMaker。
- 当你同步来自的资源时 AWS IoT SiteWise，它们SyncSource将是SITEWISE_MANAGED。这包括Entities和ComponentTypes。
- 新 AWS IoT SiteWise 功能（例如）CompositionModel仅在中可用IoTSiteWiseDefaultWorkspace。

有一些特定限制IoTSiteWiseDefaultWorkspace，它们是：

- 无法删除默认工作空间。
- 要删除资源，必须先删除 AWS IoT SiteWise 资源，然后删除中的 AWS IoT TwinMaker 相应资源。

通过 AWS IoT SiteWise同步资源

本主题列出了您可以从哪些资源同步 AWS IoT SiteWise 到 AWS IoT TwinMaker 工作区。

> **Important**
>
> 有关自定义工作空间和默认工作区之间的区别的信息，请参阅自定义工作空间和默认工作空间之间的区别。

自定义和默认工作空间

以下资源已同步，可在自定义工作空间和默认工作空间中使用：

资产模型

AWS IoT TwinMaker 为中的每个资产模型创建新的组件类型 AWS IoT SiteWise。

- 资产模型TypeId的组件将使用以下模式之一：
  - 自定义工作区- iotsitewise.assetmodel:*assetModelId*
  - 默认工作空间- *assetModelId*
- 资产模型中的每个属性都是该组件类型中的一个新属性，具有以下命名模式之一：
  - 自定义工作区- Property_*propertyId*
  - 默认工作空间- *propertyId*

  中的属性名称存储 AWS IoT SiteWise displayName在属性定义中。
- 资产模型中的每个层次结构都是一个新的属性LIST，类型nestedTypeRELATIONSHIP为组件类型。层次结构映射到以以下名称之一为前缀的属性：
  - 自定义工作区- *Hierarchy_hierarchyId*
  - 默认工作空间- *hierarchyId*

资产

AWS IoT TwinMaker 为中的每项资产创建一个新实体 AWS IoT SiteWise。

- entityId与 assetId in 相同 AWS IoT SiteWise。
- 这些实体包含一个名为 sitewiseBase 的组件，其组件类型与此资产的资产模型相对应。
- 任何资产级别的替代（例如设置属性别名或计量单位）都将反映在 AWS IoT TwinMaker中的实体中。

仅限默认工作空间

以下资源已同步，且仅在默认工作区中可用。IoTSiteWiseDefaultWorkspace

AssetModelComponents

AWS IoT TwinMaker 为AssetModelComponents中的每个组件创建一个新的组件类型 AWS IoT SiteWise。

- 资产模型TypeId的组件使用以下模式:assetModelId.
- 资产模型中的每个属性都是组件类型的新属性，其属性名称为 propertyId。中的属性名称存储 AWS IoT SiteWise displayName在属性定义中。
- 资产模型中的每个层次结构都是一个新的属性LIST，类型nestedTypeRELATIONSHIP为组件类型。层级结构映射至名称前缀为 hierarchyId 的属性。

AssetModelCompositeModel

AWS IoT TwinMaker 为`AssetModelCompositeModel`中的每个组件创建一个新的组件类型 AWS IoT SiteWise。

- 资产模型`TypeId`的组件使用以下模式:`assetModelId_assetModelCompositeModelId`.
- 资产模型中的每个属性都是组件类型的新属性，其属性名称为 `propertyId`。中的属性名称存储 AWS IoT SiteWise `displayName`在属性定义中。

AssetCompositeModels

AWS IoT TwinMaker 为`AssetCompositeModel`中的每个组件创建一个新的复合组件 AWS IoT SiteWise。

- `componentName`与 `assetModelCompositeModelId` in 相同 AWS IoT SiteWise。

资源未同步

未同步以下字段：

非同步资产与资产模型

- 警报模型将作为 CompositeModels 同步，但资产中与警报相关的相应数据不会同步。
- AWS IoT SiteWise 数据流未同步。仅同步资产模型的建模属性。
- 测量、变换、聚合和元数据计算（例如公式和窗口）等属性值不同步。仅同步有关属性的元数据，例如别名、度量单位和数据类型。可以使用常规 AWS IoT TwinMaker 数据连接器 API 查询这些值。 GetPropertyValueHistory

在中使用同步的实体和组件类型 AWS IoT TwinMaker

从中同步资源后 AWS IoT SiteWise，同步的组件类型只能在中读取。 AWS IoT TwinMaker任何更新或删除操作都必须在中完成 AWS IoT SiteWise，如果 SyncJob 仍处于活动状态， AWS IoT TwinMaker则这些更改会同步到。

同步的实体和 AWS IoT SiteWise 基础组件也是只读的。 AWS IoT TwinMaker当没有实体级别的属性（例如描述或 `entityName` 更新），您就可以向同步实体添加其他非同步组件。

部分限制适用于您与同步实体交互的方式。您无法在同步实体的层次结构中的同步实体下创建子实体。此外，您不能创建从同步组件类型扩展的非同步组件类型。

> **Note**
>
> 如果在中删除资源 AWS IoT SiteWise 或删除同步作业，则会删除其他组件和实体。

您可以在 Grafana 控制面板中使用这些同步实体，并将它们以标签形式添加至场景编辑器，就像普通实体一样。您还可以为这些同步实体发布知识图谱查询。

> **Note**
>
> 未经修改的同步实体为免费，但如果在 AWS IoT TwinMaker中进行了更改，则需要为这些实体付费。例如，如果您向已同步的实体添加非同步组件，则该实体现在会被收费。 AWS IoT TwinMaker有关更多信息，请参阅AWS IoT TwinMaker 定价。

分析同步状态与错误

本主题介绍了有关如何分析同步错误及状态。

> **Important**
>
> 有关自定义工作空间和默认工作区之间的区别的信息，请参阅the section called “自定义工作空间和默认工作空间之间的区别”。

同步任务状态

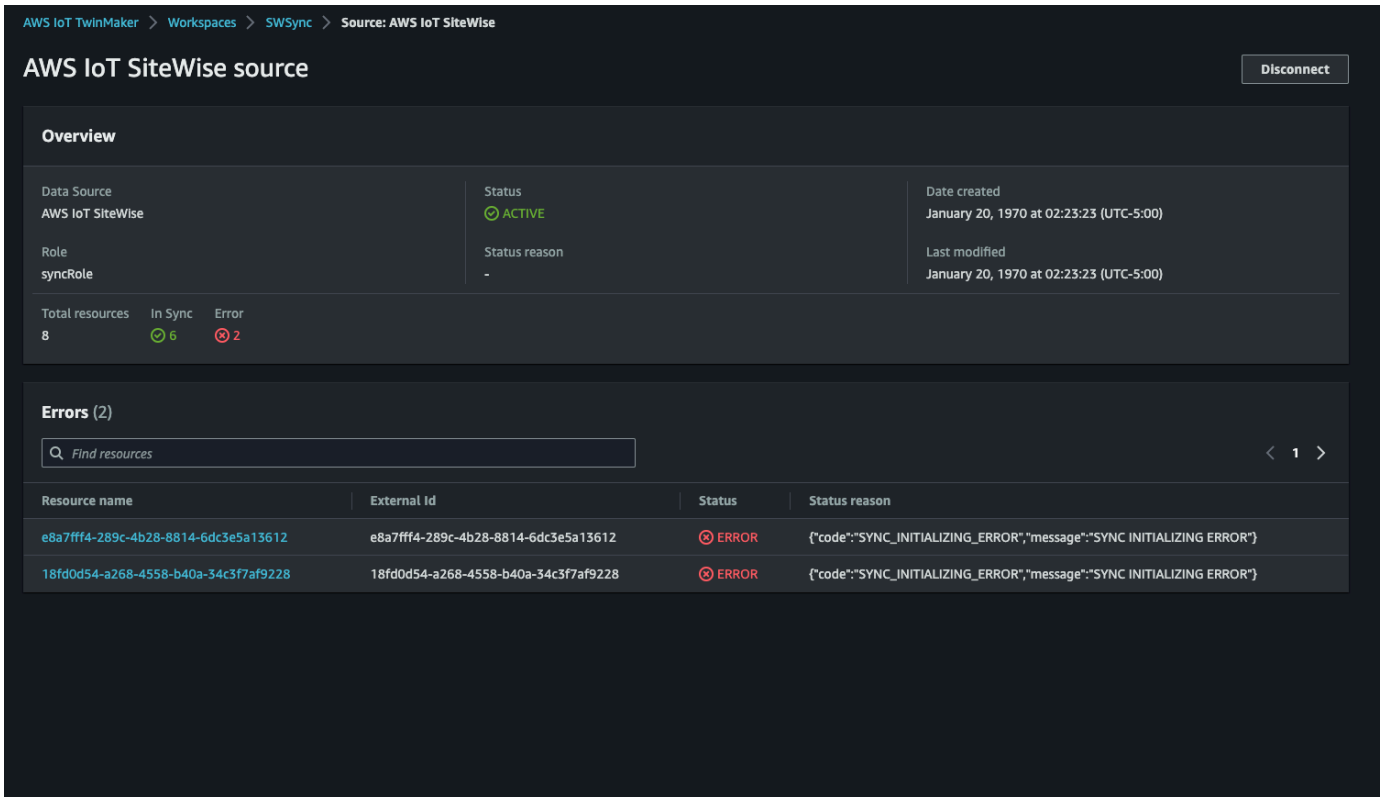
同步作业博山以下状态之一，具体取决于其状态。

- 同步作业`CREATING`状态表示该作业正在检查权限并从中加载数据 AWS IoT SiteWise 以准备同步。
- 同步任务`INITIALIZING`状态意味着中的所有现有资源 AWS IoT SiteWise 都已同步到。AWS IoT TwinMaker如果用户在 AWS IoT SiteWise中有大量资产和资产模型，则此步骤可能需要更长的时间才能完成。您可以通过在 AWS IoT TwinMaker 控制台中查看同步作业或调用 `ListSyncResources` API 监控已同步的资源数量。
- 同步作业 `ACTIVE` 状态表示初始化步骤已完成。该任务现已准备就绪，可以同步来自 AWS IoT SiteWise的任何新更新。
- 同步作业 `ERROR` 状态表示上述任何状态都存在错误。查看错误消息。IAM 角色设置可能存在问题。如果您想使用新的 IAM 角色，请删除出现错误的同步任务，然后使用新角色创建一个新任务。

同步错误在模型源页面中显示，可通过工作区中的实体模型来源表格访问该页面。模型来源页面显示同步失败的资源清单。同步作业会自动重试大多数错误，但是如果资源需要操作，则该资源将保持 ERROR 状态。您还可以使用 ListSyncResourcesAPI 获取错误列表。

要查看当前源的所有列出的错误，请使用以下程序。

1. 在 AWS IoT TwinMaker 控制台中导航至工作区。
2. 选择 “实体模型 AWS IoT SiteWise 来源” 模式中列出的来源，打开资产同步详细信息页面。



3. 如前述屏幕截图所示，错误表中列出了所有存在持续错误的资源。您可以使用此表跟踪和修复与特定资源相关的错误。

可能出现错误包括：

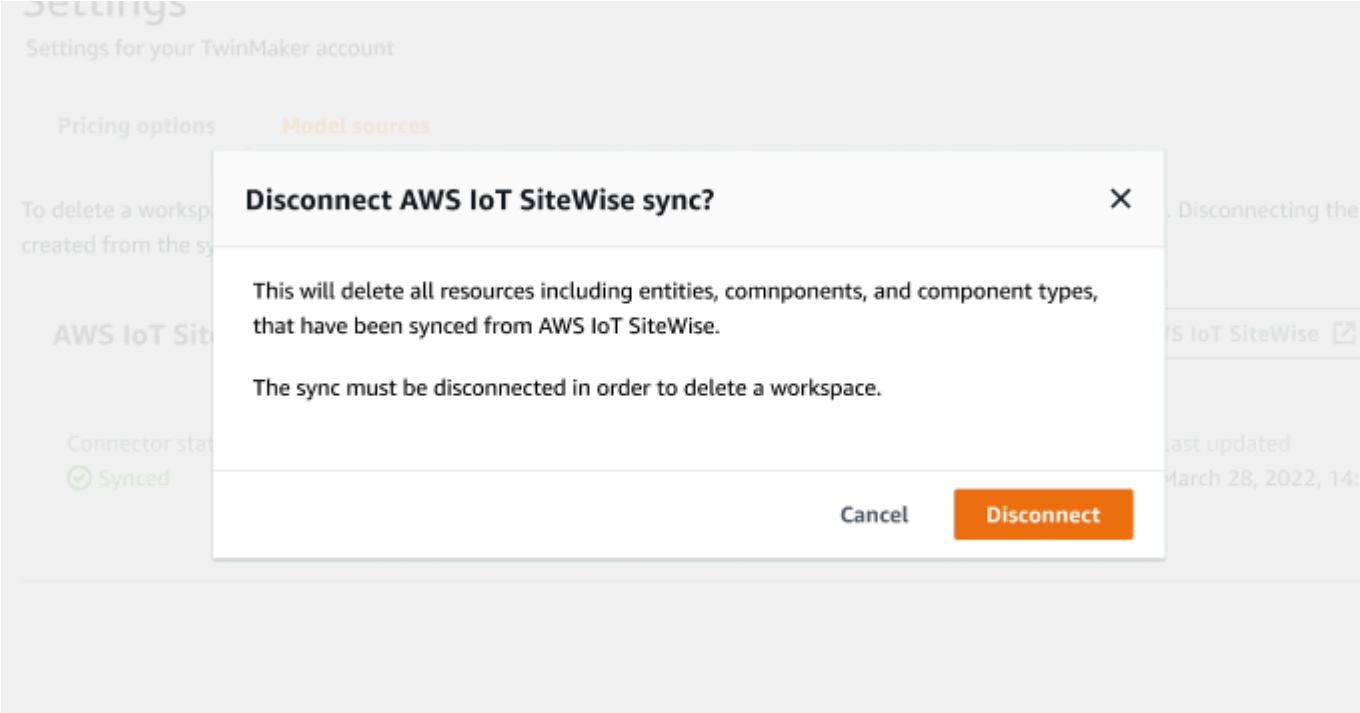
- 虽然 AWS IoT SiteWise 支持重复的资源名称，但 AWS IoT TwinMaker 仅在R00T关卡上支持重复的资源名称，而不支持在同一个父实体下。如果您在中的父实体下有两个同名的资产 AWS IoT SiteWise，则其中一个无法同步。要修复此错误，请在同步 AWS IoT SiteWise 之前删除其中一个资源或将一个资源移到另一个父资源下。
- 如果您已经有一个与 AWS IoT SiteWise 资产 ID 相同的 ID 的实体，则在您删除现有实体之前，该资产将无法同步。

删除同步作业

按照以下程序删除同步作业。

> **Important**
>
> 有关自定义工作空间和默认工作区之间的区别的信息，请参阅the section called “自定义工作空间和默认工作空间之间的区别”。

1. 导航到 AWS IoT TwinMaker 控制台。
2. 打开您想删除同步作业的工作区。
3. 在实体模型来源下，选择 AWS IoT SiteWise 源，以打开源详细信息页面。
4. 若要停止同步作业，请选择 Disconnect（断开连接）。确认您选择完全删除同步作业。



同步作业删除后，您可在相同或不同的工作区再次创建同步作业。

如果工作空间中有任何同步作业，则无法删除该工作空间。在删除工作区前，请先删除同步作业。

如果在删除同步作业过程中出现任何错误，则同步作业将保持 DELETING 状态并自动重试。如果在删除资源时出现任何错误，则可以手动删除任何已同步的实体或组件类型。

> **Note**
>
> 先删除从 AWS IoT SiteWise 中同步的所有资源，然后删除同步作业本身。

资产同步限制条件

> **Important**
>
> 有关自定义工作空间和默认工作区之间的区别的信息，请参阅the section called “自定义工作空间和默认工作空间之间的区别”。

由于此 AWS IoT SiteWise 配额高于默认 AWS IoT TwinMaker 配额，因此我们将对从 AWS IoT SiteWise中同步的实体和组件类型限制增加以下限制。

- 工作区中有 1000 个已同步的组件类型，因为它只能从 AWS IoT SiteWise中同步 1000 个资产模型。
- 一个工作空间中有 100,000 个已同步的实体，因为它只能同步来自 AWS IoT SiteWise 100,000 个资产。
- 每个父级实体最大 2,000 项子实体。它可以为每个单独的父级资产同步 2,000 个子资产。

  > **Note**
  >
  > GetEntityAPI 仅返回层次结构属性的前 50 个子实体，但您可以使用 GetPropertyValueAPI 对所有子实体列表进行分页和检索。

- 每个同步组件有 600 个属性 AWS IoT SiteWise，可以将资产模型与总共 600 个属性和层次结构同步。

> **Note**
>
> 这些限制仅适用于已同步实体。如果您需要为未同步资源提高配额，请申请增加配额。

AWS IoT TwinMaker Grafana 控制面板集成

AWS IoT TwinMaker通过应用程序插件支持 Grafana 集成。您可以使用 8.2.0 版及更高版本的 Grafana 与您的数字孪生应用程序交互。AWS IoT TwinMaker 插件提供了自定义面板、控制面板模板和用于连接到数字孪生数据的数据源。

有关如何使用 Grafana 搭载并为控制面板设置权限的更多信息，请参阅以下主题：

主题

- Grafana 场景查看器的 CORS 配置
- 设置 Grafana 环境
- 创建控制面板 IAM 角色
- 创建 AWS IoT TwinMaker 视频播放器策略

> **Note**
>
> 您需要修改 Amazon S3 存储桶的 CORS（跨源资源共享）配置，以便 Grafana 用户界面能从存储桶加载资源。有关说明，请参阅 Grafana 场景查看器的 CORS 配置。

有关 AWS IoT TwinMaker Grafana 插件的更多信息，请参阅AWS IoT TwinMaker应用程序文档。

有关 Grafana 插件关键组件的更多信息，请参阅以下内容：

- AWS IoT TwinMaker数据源
- 控制面板模板
- 场景查看器面板
- 视频播放器面板

Grafana 场景查看器的 CORS 配置

AWS IoT TwinMaker Grafana 插件需要 CORS（跨源资源共享）配置，这让 Grafana 用户界面可以从 Amazon S3 存储桶加载资源。如果没有 CORS 配置，您将在场景查看器上收到一条错误消息，即“由于网络故障，加载 3D 场景失败”，因为 Grafana 域无法访问 Amazon S3 存储桶中的资源。

要使用 CORS 配置 Amazon S3 存储桶，请按以下步骤操作：

1. 登录 IAM 控制台并打开 Amazon S3 控制台。
2. 在存储桶列表中，选择要用作AWS IoT TwinMaker工作空间资源存储桶的存储桶名称。
3. 请选择Permissions（权限）。
4. 在 跨源资源共享 (CORS) 部分中，请选择 编辑。
5. 在 CORS configuration editor（CORS 配置编辑器）文本框中，键入或复制并粘贴以下 JSON CORS 配置，方法是将 Grafana 工作空间域 *GRAFANA-WORKSPACE-DOMAIN* 替换为您的域。

> **Note**
>
> 您需要在 "AllowedOrigins": JSON 元素的开头保留星号 * 字符。

```
    [
{
    "AllowedHeaders": [
        "*"
    ],
    "AllowedMethods": [
        "GET",
        "PUT",
        "POST",
        "DELETE",
        "HEAD"
    ],
    "AllowedOrigins": [
        "*GRAFANA-WORKSPACE-DOMAIN"
    ],
    "ExposeHeaders": [
        "ETag"
    ]
}
]
```

6. 选择保存更改以完成 CORS 配置。

有关使用 Amazon S3 存储桶的 CORS 的更多信息，请参阅使用跨源资源共享 (CORS)。

设置 Grafana 环境

您可以将 Amazon Managed Grafana 用于完全托管的服务，也可以设置由您自己管理的 Grafana 环境。借助 Amazon Managed Grafana，您可以根据自己的需求快速部署、操作和扩展开源 Grafana。或者，您也可以设置自己的基础设施来管理 Grafana 服务器。

有关 Grafana 环境选项的更多信息，请参阅以下主题：

- Amazon Managed Grafana
- 自行管理的 Grafana

Amazon Managed Grafana

Amazon Managed Grafana 提供了一个 AWS IoT TwinMaker 插件，以便您可以快速将 AWS IoT TwinMaker 与 Grafana 集成。由于 Amazon Managed Grafana 为您管理 Grafana 服务器，因此您无需构建、封装或部署任何硬件或任何其他 Grafana 基础设施即可实现数据可视化。有关 Amazon Managed Grafana 的更多信息，请参阅什么是 Amazon Managed Grafana？

> **Note**
>
> Amazon Managed Grafana 目前支持 1.3.1 版的 AWS IoT TwinMaker Grafana 插件。

Amazon Managed Grafana 的先决条件

要在 Amazon Managed Grafana 控制面板中使用 AWS IoT TwinMaker，请先满足以下先决条件：

- 创建 AWS IoT TwinMaker 工作区。有关创建工作区的更多信息，请参阅使用 AWS IoT TwinMaker 入门。

> **Note**
>
> 当首次在 AWS 管理控制台创建 Amazon Managed Grafana 工作区时，AWS IoT TwinMaker 未列出。但该插件已安装在所有工作区中。您可以在开源 Grafana 插件列表中找到 AWS IoT TwinMaker 插件。您可以找到 AWS IoT TwinMaker 数据源，方法是在“数据源”页面上选择添加数据源。

在创建 Amazon Managed Grafana 工作区时，系统会自动创建一个 IAM 角色来管理 Grafana 实例的权限。这称为工作区 IAM 角色。这是身份验证提供程序选项，将用于为 Grafana 配置所有 AWS IoT TwinMaker 数据源。Amazon Managed Grafana 不支持自动为 AWS IoT TwinMaker 添加权限，因此您必须手动设置这些权限。有关设置手动权限的更多信息，请参阅 创建控制面板 IAM 角色。

自行管理的 Grafana

您可以选择托管自己的基础架构来运行 Grafana。有关在计算机上本地运行 Grafana 的信息，请参阅安装 Grafana。AWS IoT TwinMaker 插件可在公开 Grafana 目录中找到。有关在 Grafana 环境中安装此插件的信息，请参阅AWS IoT TwinMaker应用程序。

当您在本地运行 Grafana 时，您无法轻松共享控制面板或向多个用户提供访问权限。有关使用本地 Grafana 共享控制面板的脚本式快速入门指南，请参阅AWS IoT TwinMaker示例存储库。此资源将引导您完成在 Cloud9 上托管 Grafana 环境和在公共端点上托管 Amazon EC2 的过程。

您必须确定要使用哪个身份验证提供程序来配置 TwinMaker 数据源。您可以根据默认凭证链为环境配置凭证（请参阅使用默认凭证提供程序链）。默认凭证可以是任何用户或角色的永久凭证。例如，如果您在 Amazon EC2 上运行 Grafana，则默认凭证链可以访问 Amazon EC2 执行角色，该角色届时将是您的身份验证提供程序。在 创建控制面板 IAM 角色 的步骤中，需要身份验证提供程序的 IAM Amazon 资源名称（ARN）。

创建控制面板 IAM 角色

借助 AWS IoT TwinMaker，您可以控制 Grafana 控制面板上的数据访问权限。Grafana 控制面板用户应具有不同的权限范围来查看数据，某些情况下还要有写入数据的权限。例如，警报操作员可能没有查看视频的权限，而管理员拥有所有资源的权限。Grafana 通过数据源定义权限，而凭证和 IAM 角色在数据源中提供。AWS IoT TwinMaker 数据源获取具有该角色权限的 AWS 凭证。如果未提供 IAM 角色，Grafana 将使用凭证的范围，该范围不会被 AWS IoT TwinMaker 缩小。

要在 Grafana 中使用 AWS IoT TwinMaker 控制面板，您需要创建一个 IAM 角色并附加策略。您可以使用以下模板来帮助创建这些策略。

创建 IAM 策略

在 IAM 控制台创建一个名为 *YourWorkspaceId*DashboardPolicy 的 IAM policy。此策略允许您的工作区访问 Amazon S3 存储桶和 AWS IoT TwinMaker 资源。您也可以决定使用AWS IoT Greengrass Amazon Kinesis Video Streams 边缘连接器，这需要 Kinesis Video Streams 的权限和为该组件配置的 AWS IoT SiteWise 资产。要适合您的用例，请选择以下策略模板之一。

1。无视频权限策略

如果您不想使用 Grafana 视频播放器面板，请使用以下模板创建策略。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject"
      ],
      "Resource": [
        "arn:aws:s3:::bucketName/*",
        "arn:aws:s3:::bucketName"
        ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "iottwinmaker:Get*",
        "iottwinmaker:List*"
      ],
      "Resource": [
        "arn:aws:iottwinmaker:region:accountId:workspace/workspaceId",
        "arn:aws:iottwinmaker:region:accountId:workspace/workspaceId/*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": "iottwinmaker:ListWorkspaces",
      "Resource": "*"
    }
  ]
}
```

将为每个工作区创建 Amazon S3 存储桶。它包含要在控制面板上查看的 3D 模型和场景。SceneViewer面板会从这个桶中装入物品。

2。范围缩小的视频权限策略

要限制对 Grafana 中视频播放器面板的访问权限，请按标签对 AWS IoT GreengrassAmazon Kinesis Video Streams 边缘连接器资源进行分组。有关缩小视频资源权限范围的更多信息，请参阅创建 AWS IoT TwinMaker 视频播放器策略。

3。所有视频权限

如果您不想对视频进行分组，则可以通过 Grafana 视频播放器将其全部设置为可访问。任何有权访问 Grafana 工作区的人都可以播放您账户中任何流媒体的视频，并对任何 AWS IoT SiteWise 资产拥有只读权限。这包括将来创建的所有资源。

使用以下模板创建策略：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject"
      ],
      "Resource": [
        "arn:aws:s3:::bucketName/*",
        "arn:aws:s3:::bucketName"
        ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "iottwinmaker:Get*",
        "iottwinmaker:List*"
      ],
      "Resource": [
        "arn:aws:iottwinmaker:region:accountId:workspace/workspaceId",
        "arn:aws:iottwinmaker:region:accountId:workspace/workspaceId/*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": "iottwinmaker:ListWorkspaces",
      "Resource": "*"
    },
```

```
    {
      "Effect": "Allow",
      "Action": [
        "kinesisvideo:GetDataEndpoint",
        "kinesisvideo:GetHLSStreamingSessionURL"
      ],
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "iotsitewise:GetAssetPropertyValue",
        "iotsitewise:GetInterpolatedAssetPropertyValues"
      ],
      "Resource": "*"
    },
    {
       "Effect": "Allow",
       "Action": [
        "iotsitewise:BatchPutAssetPropertyValue"
      ],
      "Resource": "*",
      "Condition": {
        "StringLike": {
          "aws:ResourceTag/EdgeConnectorForKVS": "*workspaceId*"
        }
      }
    }
  ]
}
```

此策略模板提供以下权限：

- 对 S3 存储桶的只读访问权限以加载场景。
- 工作区中所有实体和组件的 AWS IoT TwinMaker 只读访问权限。
- 流媒体播放您账户中所有 Kinesis Video Streams 视频的只读访问权限。
- 您账户中所有 AWS IoT SiteWise 资产的财产价值历史记录只读访问权限。
- 将数据摄取到标有键 `EdgeConnectorForKVS` 和值 `workspaceId` 的 AWS IoT SiteWise 资产的任何属性中。

标记您的摄像头 AWS IoT SiteWise 资产请求从边缘上传视频

使用 Grafana 中的视频播放器，用户可以手动请求将视频从边缘缓存上传到 Kinesis Video Streams。您可以为任何与 AWS IoT GreengrassAmazon Kinesis Video Streams 边缘连接器关联且标有键 `EdgeConnectorForKVS` 的 AWS IoT SiteWise 资产启用此功能。

标签值可以是由以下任意字符分隔的 WorkspaceID 列表：`. : + = @ _ / -`。例如，如果要跨 AWS IoT TwinMaker 工作区使用与AWS IoT Greengrass Amazon Kinesis Video Streams 边缘连接器关联的 AWS IoT SiteWise 资产，则可以使用遵循以下模式的标签：`WorkspaceA/WorkspaceB/WorkspaceC`。Grafana 插件强制使用 AWS IoT TwinMakerWorkspaceID 对 AWS IoT SiteWise 资产数据摄取进行分组。

向控制面板策略添加更多权限

AWS IoT TwinMakerGrafana 插件使用您的身份验证提供程序来 AssumeRole 调用您创建的控制面板角色。在内部，该插件通过在 AssumeRole 调用中使用会话策略来限制您有权访问的最大权限范围。有关会话策略的更多信息，请参阅会话策略。

这是您可以为 AWS IoT TwinMaker 工作区控制面板角色设置的最大允许策略：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject"
      ],
      "Resource": [
        "arn:aws:s3:::bucketName/*",
        "arn:aws:s3:::bucketName"
        ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "iottwinmaker:Get*",
        "iottwinmaker:List*"
      ],
      "Resource": [
        "arn:aws:iottwinmaker:region:accountId:workspace/workspaceId",
```

```
        "arn:aws:iottwinmaker:region:accountId:workspace/workspaceId/*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": "iottwinmaker:ListWorkspaces",
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "kinesisvideo:GetDataEndpoint",
        "kinesisvideo:GetHLSStreamingSessionURL"
      ],
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "iotsitewise:GetAssetPropertyValue",
        "iotsitewise:GetInterpolatedAssetPropertyValues"
      ],
      "Resource": "*"
    },
    {
       "Effect": "Allow",
       "Action": [
        "iotsitewise:BatchPutAssetPropertyValue"
      ],
      "Resource": "*",
      "Condition": {
        "StringLike": {
          "aws:ResourceTag/EdgeConnectorForKVS": "*workspaceId*"
        }
      }
    }
  ]
}
```

如果您添加 `Allow` 更多权限的语句，它们将无法在 AWS IoT TwinMaker 插件上运行。这是为了确保该插件使用最低必要权限而设计的。

但您可以进一步缩小权限范围。有关信息，请参阅 创建 AWS IoT TwinMaker 视频播放器策略。

创建 Grafana 控制面板 IAM 角色

在 IAM 控制台中，创建名为 *YourWorkspaceId*DashboardRole 的 IAM 角色。将 *YourWorkspaceId*DashboardPolicy 附加到角色上。

要编辑控制面板角色的信任策略，您必须授予 Grafana 身份验证提供程序调用控制面板角色上 AssumeRole 的权限。使用以下模板更新信任策略：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "ARN of Grafana authentication provider"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

有关如何创建 Grafana 环境和查找身份验证提供程序的更多信息，请参阅 设置 Grafana 环境。

创建 AWS IoT TwinMaker 视频播放器策略

以下是一个策略模板，其中包含在 Grafana 中使用 AWS IoT TwinMaker 插件所需的所有视频权限：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject"
      ],
      "Resource": [
        "arn:aws:s3:::bucketName/*",
        "arn:aws:s3:::bucketName"
        ]
    },
    {
      "Effect": "Allow",
```

```
      "Action": [
        "iottwinmaker:Get*",
        "iottwinmaker:List*"
      ],
      "Resource": [
        "arn:aws:iottwinmaker:region:accountId:workspace/workspaceId",
        "arn:aws:iottwinmaker:region:accountId:workspace/workspaceId/*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": "iottwinmaker:ListWorkspaces",
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "kinesisvideo:GetDataEndpoint",
        "kinesisvideo:GetHLSStreamingSessionURL"
      ],
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "iotsitewise:GetAssetPropertyValue",
        "iotsitewise:GetInterpolatedAssetPropertyValues"
      ],
      "Resource": "*"
    },
    {
       "Effect": "Allow",
       "Action": [
        "iotsitewise:BatchPutAssetPropertyValue"
      ],
      "Resource": "*",
      "Condition": {
        "StringLike": {
          "aws:ResourceTag/EdgeConnectorForKVS": "*workspaceId*"
        }
      }
    }
  ]
```

```
}
```

有关完整策略的更多信息，请参阅 创建 IAM 策略 主题中的所有视频权限策略模板。

缩小访问资源的权限范围

Grafana 中的视频播放器面板直接调用 Kinesis Video Streams SiteWise 和 IoT，以提供完整的视频播放体验。为避免未经授权访问与您的 AWS IoT TwinMaker 工作区无关的资源，请为您工作区控制面板角色的 IAM policy 添加条件。

缩小 GET 权限范围

您可以通过标记资源来缩小 Amazon Kinesis Video Streams 和 AWS IoT SiteWise 资产的访问权限范围。您可能已根据 AWS IoT TwinMaker WorkspaceID 为 AWS IoT SiteWise 摄像头资产添加了标签，以启用视频上传请求功能，请参阅从边缘上传视频主题。您可以使用相同的标签键值对来限制对 AWS IoT SiteWise 资产的 GET 访问权限，也可以用同样的方式标记您的 Kinesis Video Streams。

然后，您可以将此条件添加到 *YourWorkspaceId*DashboardPolicy 中的 kinesisvideo 和 iotsitewise 语句：

```
"Condition": {
  "StringLike": {
    "aws:ResourceTag/EdgeConnectorForKVS": "*workspaceId*"
  }
}
```

真实生活用例：对摄像头进行分组

在此场景中，您有大量的摄像头监视工厂烘焙饼干的过程。成批的饼干面糊在面糊室制作，面糊在冷冻室冷冻，饼干在烘焙室烘烤。每个房间都有摄像头，不同的操作员团队分别监视每个过程。您希望每组操作员都获得各自工作间的授权。在为饼干厂构建数字孪生时，使用的是单个工作区，但摄像头权限需要按工作间划分范围。

您可以通过根据摄像头组的 GroupingId 标记摄像头组来实现这种权限分离。在这种情况下，GroupingID 为 BatterRoom FreezerRoom、和。 BakingRoom每个房间的摄像头都连接到 Kinesis Video Streams，并且应该有一个键为 EdgeConnectorForKVS、值为 BatterRoom 的标签。该值可以是由以下任何字符分隔的分组列表：. : + = @ _ / -

要修改 *YourWorkspaceId*DashboardPolicy，请使用以下策略语句：

```
...,
{
  "Effect": "Allow",
  "Action": [
    "kinesisvideo:GetDataEndpoint",
    "kinesisvideo:GetHLSStreamingSessionURL"
  ],
  "Resource": "*",
  "Condition": {
    "StringLike": {
      "aws:ResourceTag/EdgeConnectorForKVS": "*groupingId*"
    }
  }
},
{
  "Effect": "Allow",
  "Action": [
    "iotsitewise:GetAssetPropertyValue",
    "iotsitewise:GetInterpolatedAssetPropertyValues"
  ],
  "Resource": "*",
  "Condition": {
    "StringLike": {
      "aws:ResourceTag/EdgeConnectorForKVS": "*groupingId*"
    }
  }
},
...
```

这些语句限制对分组中特定资源的流媒体视频播放和 AWS IoT SiteWise 属性历史记录访问权限。*groupingId* 由您的用例定义。在我们的场景中，它将是 RoomID。

范围缩小AWS IoT SiteWise BatchPutAssetPropertyValue 权限

提供此权限可开启视频播放器中的视频上传请求功能。上传视频时，您可以指定一个时间范围，并在 Grafana 控制面板上选择提交来提交请求。

要授予 iotsitewise: BatchPutAssetPropertyValue 权限，请使用默认策略：

```
...,
{
  "Effect": "Allow",
```

```
  "Action": [
    "iotsitewise:BatchPutAssetPropertyValue"
  ],
  "Resource": "*",
  "Condition": {
    "StringLike": {
      "aws:ResourceTag/EdgeConnectorForKVS": "*workspaceId*"
    }
  }
},
...
```

通过使用此政策，用户可以调 BatchPutAssetPropertyValue 用AWS IoT SiteWise相机资产上的任何属性。您可以通过在语句的条件中指定特定 PropertyID 来限制对它的授权。

```
{
  ...
  "Condition": {
    "StringEquals": {
      "iotsitewise:propertyId": "propertyId"
    }
  }
  ...
}
```

Grafana 中的 “视频播放器” 面板将数据提取到 VideoUploadRequest名为的测量属性中，以启动将视频从边缘缓存上传到 Kinesis Video Streams 的操作。在 AWS IoT SiteWise 控制台找到该属性的 PropertyID。要修改 *YourWorkspaceId*DashboardPolicy，请使用以下策略语句：

```
...,
{
  "Effect": "Allow",
  "Action": [
    "iotsitewise:BatchPutAssetPropertyValue"
  ],
  "Resource": "*",
  "Condition": {
    "StringLike": {
      "aws:ResourceTag/EdgeConnectorForKVS": "*workspaceId*"
    },
    "StringEquals": {
      "iotsitewise:propertyId": "VideoUploadRequestPropertyId"
```

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

此语句将摄取数据限制在已标记的 AWS IoT SiteWise 摄像头资产的特定属性。有关更多信息，请参阅AWS IoT SiteWise 如何与 IAM 结合使用。

将 AWS IoT SiteWise 警报连接到 AWS IoT TwinMaker Grafana 控制面板

> **Note**
>
> 该功能为公共预览版，可能会发生变化。

AWS IoT TwinMaker 能够将事件警 AWS IoT SiteWise 报导入到 AWS IoT TwinMaker 组件中。这使您无需为数据迁移实现自定义数据连接器即可查询警报状态和配置警报阈值。 AWS IoT SiteWise 您可以使用 AWS IoT TwinMaker Grafana 插件在 Grafana 中可视化警报状态并配置警报阈值，无需对警报进行 API 调用或直接与警报交互。 AWS IoT TwinMaker AWS IoT SiteWise

AWS IoT SiteWise 警报配置先决条件

创建警报并将其集成至 Grafana 控制面板之前，请确保您已查看以下先决条件：

- 熟悉 AWS IoT SiteWise的模型和资产系统。有关更多信息，请参阅用户指南中的创建资产模型和创建资产。
- 熟悉 IoT Events 警报模型以及如何将它们连接到 AWS IoT SiteWise 模型。有关更多信息，请参阅《用户指南》中的定义 AWS IoT 事件警报。
- AWS IoT TwinMaker 与 Grafana 集成，这样你就可以在 Grafana 中访问自己的 AWS IoT TwinMaker 资源了。有关更多信息，请参阅AWS IoT TwinMaker Grafana 控制面板集成。

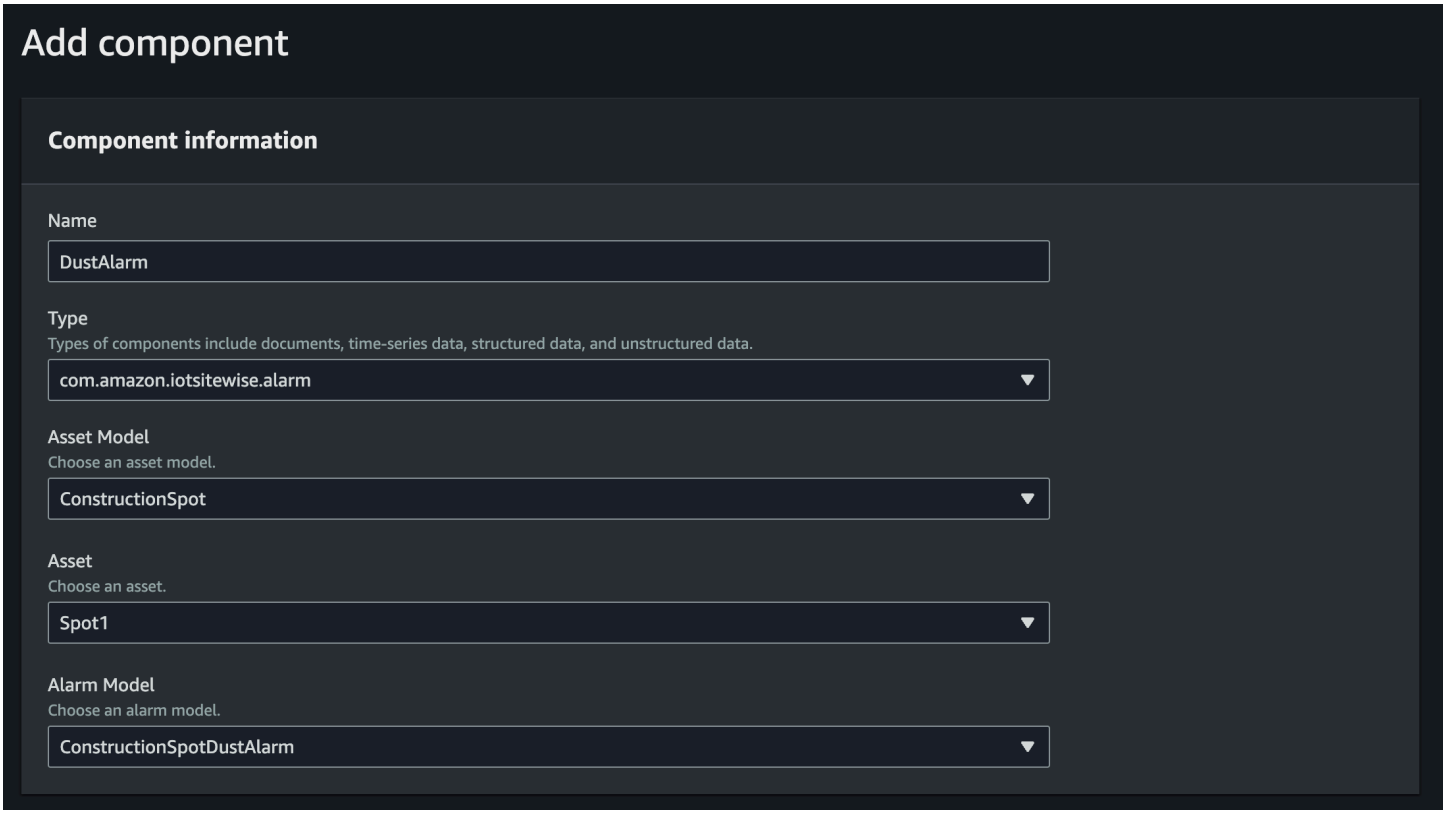
定义 AWS IoT SiteWise 警报组件 IAM 角色

AWS IoT TwinMaker 使用工作空间 IAM 角色在 Grafana 中查询和配置警报阈值。要在 Grafana 中与 AWS IoT SiteWise 警报进行交互， AWS IoT TwinMaker 工作空间角色需要以下权限：

```
{
"Effect": "Allow",
    "Action": [
        "iotevents:DescribeAlarmModel",
    ],
    "Resource": ["{IoTEventsAlarmModelArn}"]
},{
```

```
"Effect": "Allow",
    "Action": [
        "iotsitewise:BatchPutAssetPropertyValue"
    ],
    "Resource": ["{IoTSitewiseAssetArn}"]
}
```

在AWS IoT TwinMaker 控制台中，创建一个代表您的 AWS IoT SiteWise 资产的实体。确保使用com.amazon.iotsitewise.alarm组件类型为该实体添加组件，然后选择相应的资产和警报模型。



上面的屏幕截图是使用类型创建此实体的示例com.amazon.iotsitewise.alarm。

创建此组件时， AWS IoT TwinMaker 会自动从 AWS IoT SiteWise 和导入相关的警报属性 AWS IoT Events。您可以重复此警报组件类型模式，为工作区中所需的所有资产创建警报组件。

通过 AWS IoT TwinMaker API 进行查询和更新

创建警报组件后，您可以通过 AWS IoT TwinMaker API 查询警报状态、阈值和更新警报阈值。

以下是警报状态查询请求示例：

```
aws iottwinmaker get-property-value-history --cli-input-json \
'{
    "workspaceId": "{workspaceId}",
    "entityId": "{entityId}",
    "componentName": "{componentName}",
    "selectedProperties": ["alarm_status"],
    "startTime": "{startTimeIsoString}",
    "endTime": "{endTimeIsoString}"
}'
```

以下是警报阈值查询请求示例。

```
aws iottwinmaker get-property-value-history --cli-input-json \
'{
    "workspaceId": "{workspaceId}",
    "entityId": "{entityId}",
    "componentName": "{componentName}",
    "selectedProperties": ["alarm_threshold"],
    "startTime": "{startTimeIsoString}",
    "endTime": "{endTimeIsoString}"
}'
```

以下是警报阈值更新请求示例：

```
aws iottwinmaker batch-put-property-values --cli-input-json \
'{
    "workspaceId": "{workspaceId}",
    "entries": [
        {
            "entityPropertyReference": {
                "entityId": "{entityId}",
                "componentName": "{componentName}",
                "propertyName": "alarm_threshold"
            },
            "propertyValues": [
                {
                    "value": {
                        "doubleValue": "{newThreshold}"
                    },
                    "time": "{effectiveTimeIsoString}"
                }
            ]
```

```
        }
    ]
}'
```

配置您的 Grafana 控制面板以获取警报

需要创建第二个启用写入功能的仪表板 IAM 角色，该角色是一个普通角色，但具有将操作`iottwinmaker:BatchPutPropertyValues`添加到 TwinMaker 工作区 arn 的权限，如下例所示。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iottwinmaker:Get*",
               "iottwinmaker:List*",
                "iottwinmaker:BatchPutPropertyValues"
            ],
            "Resource": [
                "{workspaceArn}",
                "{workspaceArn}/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": "iottwinmaker:ListWorkspaces",
            "Resource": "*"
        }
    ]
}
```

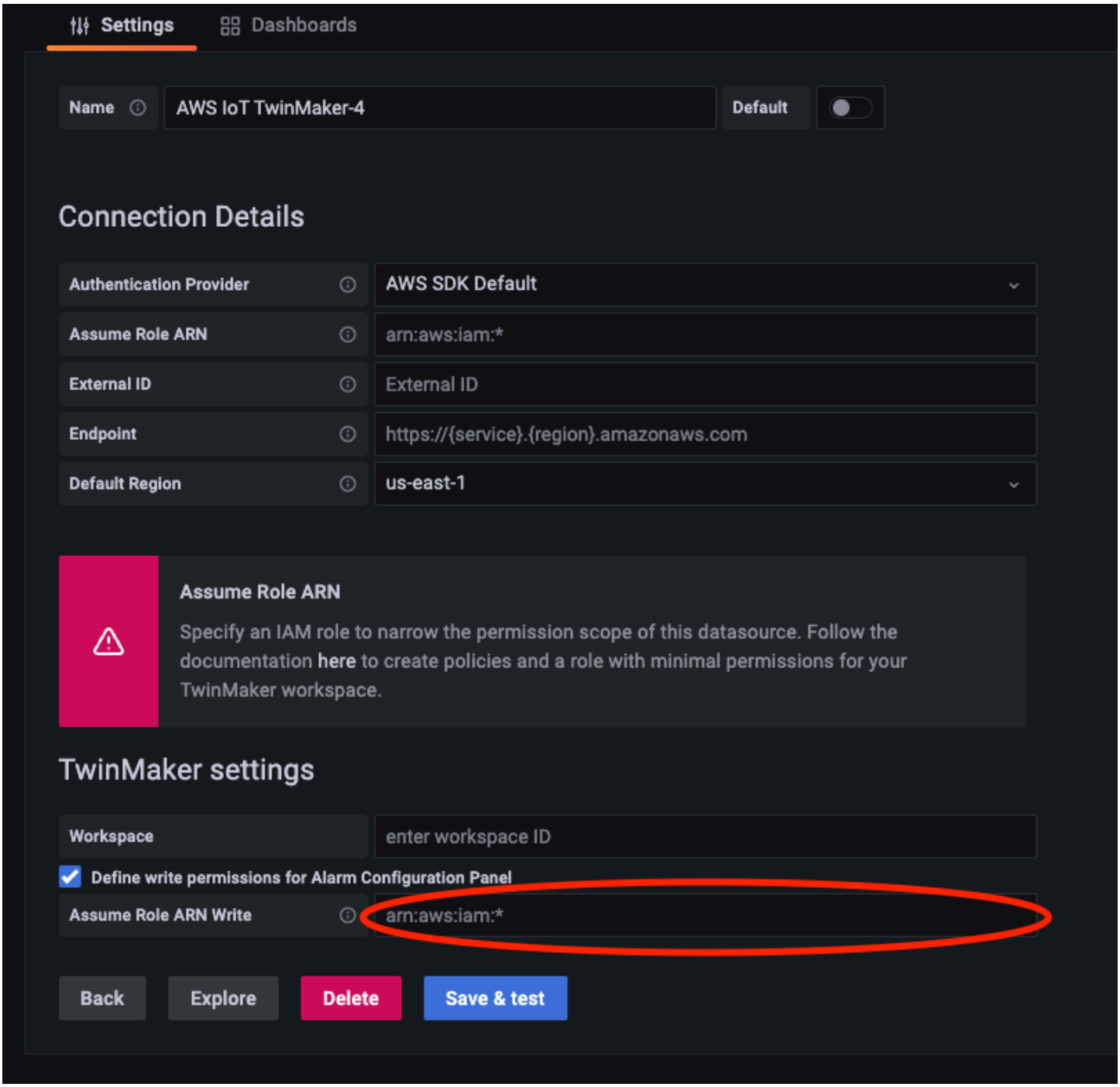
或者，您可以改为在 IAM 角色末尾添加以下语句：

```
{
    "Effect": "Allow",
    "Action": [
        "iottwinmaker:BatchPutPropertyValues"
    ],
    "Resource": [
        "{workspaceArn}",
```

```
            "{workspaceArn}/*"
        ]
}
```

通过您创建的写入角色，数据源写入 ARN 设置。

修改您的 IAM 角色后，登录您的 Grafana 控制面板以代入更新的角色 arn。选中 “为警报配置面板定义写入权限” 复选框，然后复选写入角色的 arn。



通过 Grafana 控制面板实现警报可视化

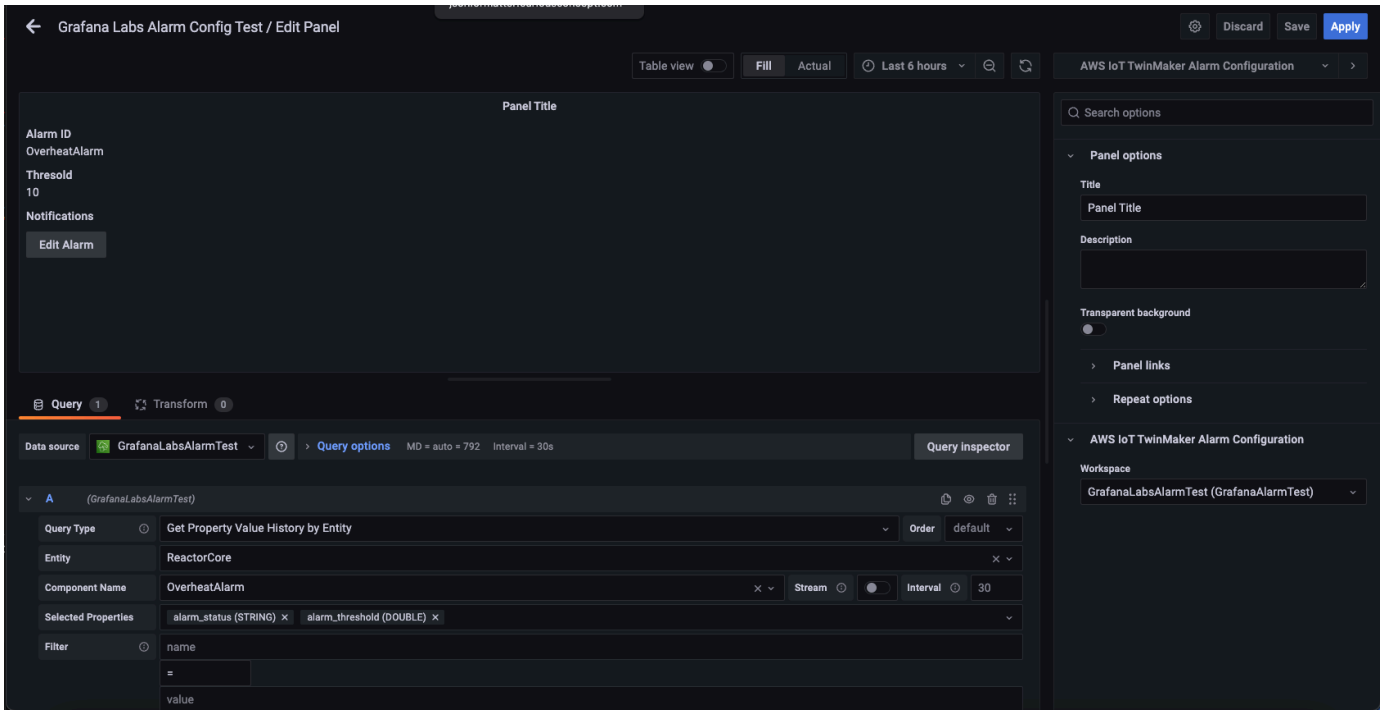
按以下程序，将警报配置平面添加至您的控制面板，并对其进行配置。

1. 在面板选项中选择工作区。
2. 在查询配置中设置数据源。
3. 使用以下查询类型：`Get Property Value History by Entity`。
4. 选择您想要添加警报的目标实体变量。
5. 选定实体后，选择组件或组件变量，以应用属性。
6. 对于该属性，选择 `alarm_status` 和 `alarm_threshold`。

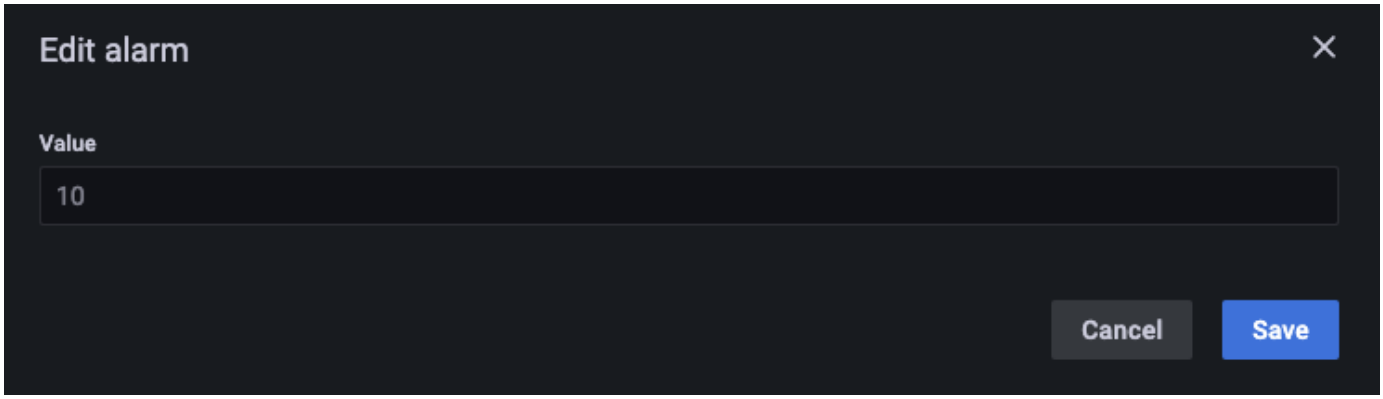
   连接后，您应该会看到警报 ID 及其当前阈值。

> **Note**
>
> 公开预览版不会显示任何通知。您应该查看警报状态和阈值，以确保属性正确应用。

7. 应使用默认的升序查询，以便显示最新值。
8. 查询的筛选器部分可留空。完整配置如下图所示：



9. 使用Edit Alarm（编辑警报）按钮，可以弹出对话框，以更改当前警报阈值。
10. 选择Save（保存）以设置新阈值。



> **Note**
>
> 此平面只能用于含当前在内的实时时间范围内。将警报阈值编辑为当前阈值时，将其与过去结束和开始的时间范围一起使用可能会导致显示意外值。

AWS IoT TwinMaker Matterport 集成

Matterport 提供多种拍摄选项，用于扫描现实世界环境和创建身临其境的 3D 模型，也称为 Matterport 数字孪生。这些模型被称为 Matterport 空间。 AWS IoT TwinMaker 支持 Matterport 集成，可以将 Matterport 数字孪生导入 AWS IoT TwinMaker 场景。通过将 Matterport 数字双胞胎与 Matterport 配对 AWS IoT TwinMaker，您可以在虚拟环境中可视化和监控您的数字双胞胎系统。

有关使用 Matterport 的更多信息，请阅读 AWS IoT TwinMaker 和 Matterport 页面上的 Matterport 文档。

集成主题

- 集成概述
- Matterport 集成的先决条件
- 生成并记录 Matterport 凭证
- 将你的 Matterport 凭证存储在 AWS Secrets Manager
- 将 Matterport 空间导入场景 AWS IoT TwinMaker
- 在你的 Gra AWS IoT TwinMaker fana 控制面板中使用 Matterport 空间
- 在你 AWS IoT TwinMaker 的 Web 应用程序中使用 Matterport 空格

集成概述

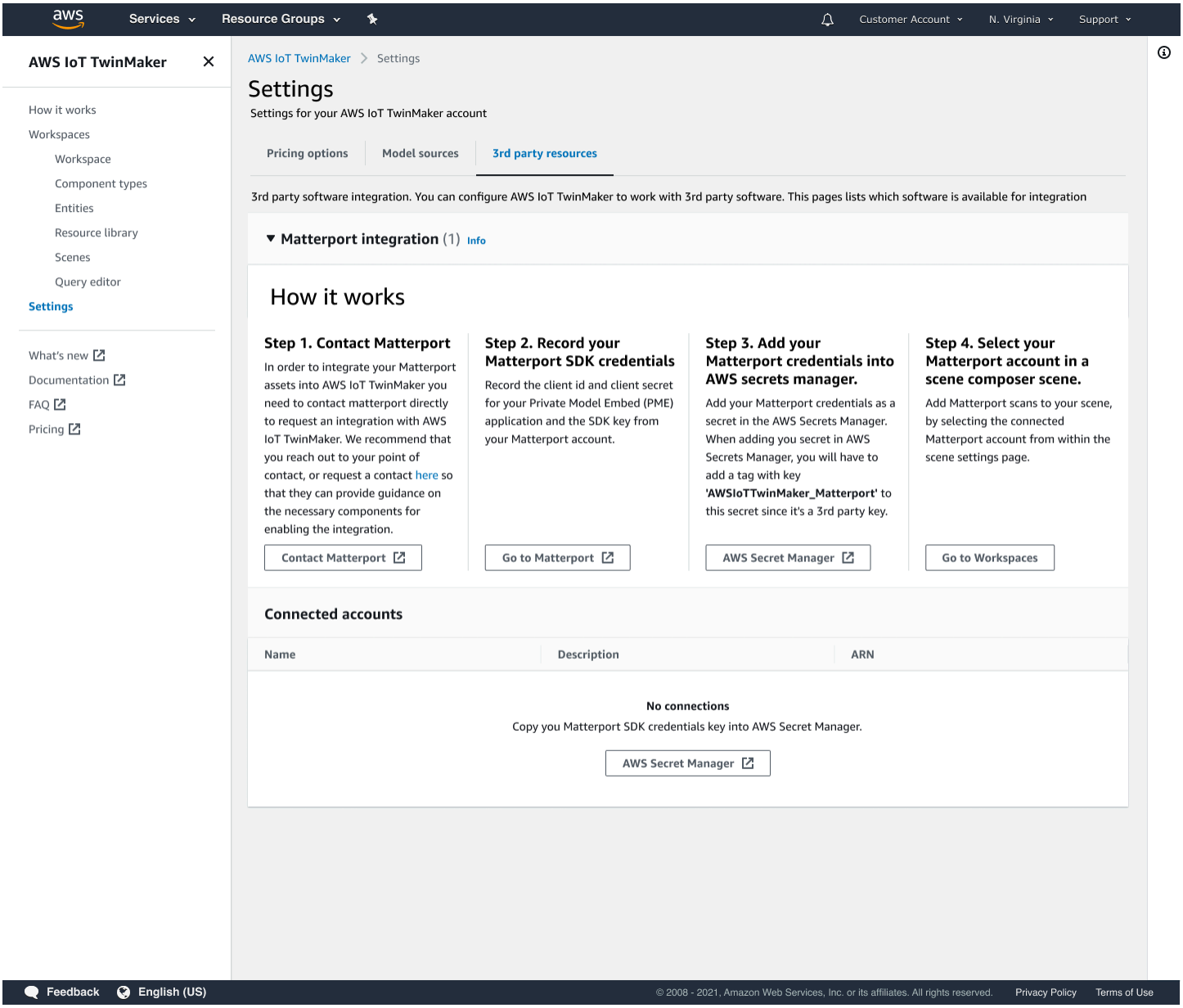
此集成让您可以执行以下操作：

- 在 AWS IoT TwinMaker 应用程序套件中使用你的 Matterport 标签和空格。
- 在 Graf AWS IoT TwinMaker ana 控制面板中查看您导入的 matterport 数据。有关使用 AWS IoT TwinMaker 和 Grafana 的更多信息，请阅读 Grafana 控制面板集成文档。
- 将你的 Matterport 空间导入你的 AWS IoT TwinMaker 场景中。
- 选择并导入要绑定到场景中数据的 Matterport 标签。 AWS IoT TwinMaker
- 在 AWS IoT TwinMaker 场景中自动显示您的 Matterport 空间和标记更改，并批准要同步哪些更改。

集成过程由 3 个关键步骤组成。

1. 生成并记录 Matterport 凭证
2. 将你的 Matterport 凭证存储在 AWS Secrets Manager
3. 将 Matterport 空间导入场景 AWS IoT TwinMaker

可以在 AWS IoT TwinMaker 控制台开始集成。在控制台的设置页面的第三方资源下，打开 Matterport 集成以浏览集成所需的不同资源。



Matterport 集成的先决条件

在与 Matterport 集成 Matterport 之前， AWS IoT TwinMaker 请确保满足以下先决条件：

- 您已经购买了企业级 Matterport 账户和集成所需的 Matterport 产品。 AWS IoT TwinMaker
- 你有一个 AWS IoT TwinMaker 工作空间。有关更多信息，请参阅入门 AWS IoT TwinMaker。
- 您已经更新了 AWS IoT TwinMaker 工作空间角色。有关创建工作区角色的更多信息，请参阅创建和管理 AWS IoT TwinMaker的服务角色。

  为工作区角色添加以下内容：

```
{
  "Effect": "Allow",
  "Action": "secretsmanager:GetSecretValue",
  "Resource": [
   "AWS Secrets Manager secret ARN"
  ]
}
```

- 必须联系 Matterport 以配置启用集成的必要许可。Matterport 还将为集成启用私有模型嵌入 (PME)。

  如果您已经有 Matterport 客户经理，请直接与他们联系。

  如果您没有 Matterport 联系人，请按照以下步骤联系 Matterport 并申请进行集成：

  1. 打开 Matterport 和 AWS IoT TwinMaker页面。
  2. 按联系我们按钮以打开联系表单。
  3. 在表格上填写所需信息。
  4. 准备就绪后，选择 SAY HELLO 将请求发送到 Matterport。

  请求集成后，您可以生成继续集成过程所需的 Matterport SDK 和私有模型嵌入 (PME) 凭证。

  > **Note**
  >
  > 这可能会因购买新产品或服务而发生费用。

生成并记录 Matterport 凭证

要将 Matterport 与 Matterport 集成 AWS IoT TwinMaker，您必须提供 Matterport AWS Secrets Manager 凭证。使用以下程序生成 Matterport SDK 凭证。

1. 登录您的 Matterport 账户。
2. 导航到您的账户设置页面。
3. 进入设置页面后，选择开发者工具选项。
4. 在开发者工具页面，前往 SDK 密钥管理部分。
5. 进入 SDK 密钥管理部分后，选择添加新 SDK 密钥的选项。

6. 获得 Matterport SDK 密钥后，在 Grafana 服务器的 AWS IoT TwinMaker 密钥中添加域名。如果您使用的是 AWS IoT TwinMaker 应用程序套件，请务必同时添加您的自定义域名。
7. 接下来，找到 应用程序集成管理部分，您应该会看到列出的 PME 应用程序。请注意以下信息：
   - 客户端 ID
   - 客户端密钥

   > **Note**
   >
   > 由于客户端密钥仅提供给您一次，因此我们强烈建议您记下客户端密钥。必须在 AWS Secrets Manager 控制台出示客户端密钥才能继续进行 Matterport 集成。

   当您购买了必要的组件并且 Matterport 已为您的账户启用了 PME 后，这些凭证就会自动创建。如果这些凭证未出现，请联系 Matterport。要想请求联系，请参阅 Matterport 和 AWS IoT TwinMaker 联系表单。

有关 Matterport SDK 凭证的更多信息，请参阅 Matterport 的官方 SDK 文档 SDK 文档概述。

将你的 Matterport 凭证存储在 AWS Secrets Manager

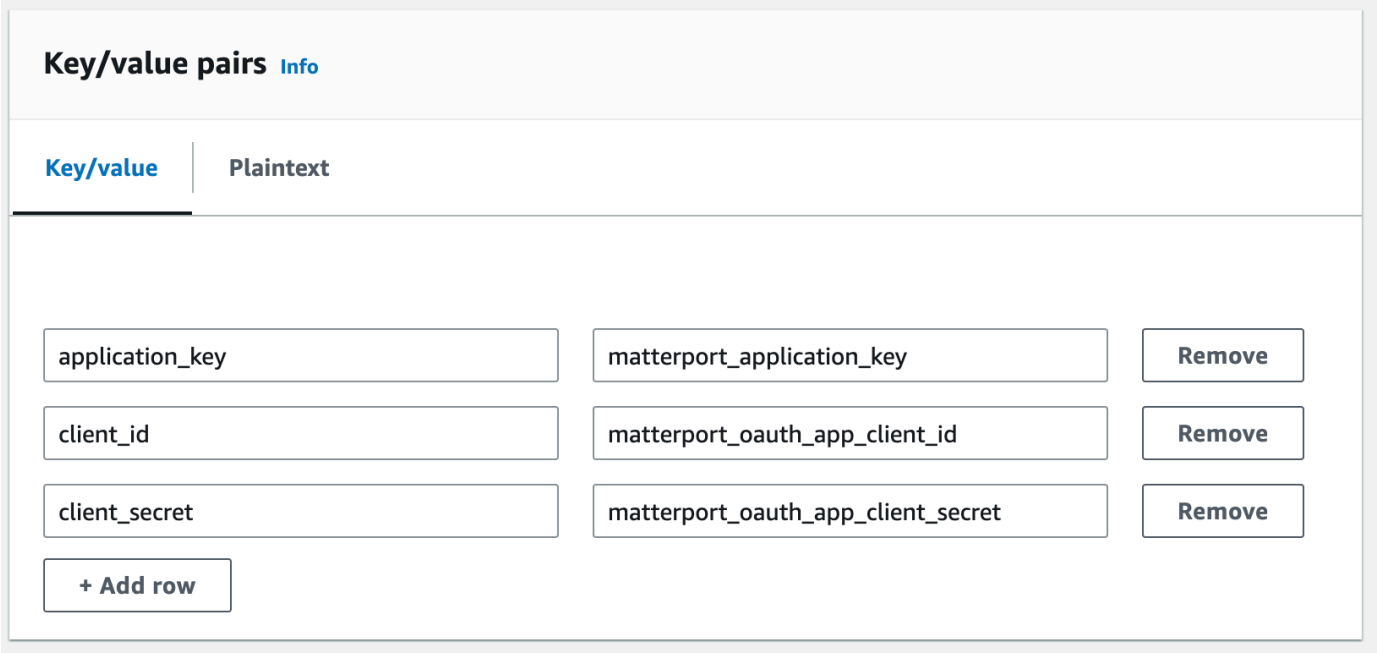
使用以下步骤将您的 Matterport 凭据存储在中。 AWS Secrets Manager

> **Note**
>
> 您需要有根据 生成并记录 Matterport 凭证 主题中的步骤创建的客户端密钥才能继续进行 Matterport 集成。

1. 登录 AWS Secrets Manager 控制台。
2. 导航到密钥页面，然后选择存储新密钥。
3. 对于密钥类型，请选择其他密钥类型。
4. 在键/值对部分，添加以下键值对，并将 Matterport 凭证作为值：
   - 创建一个键值对，其中键为 `application_key`，值为 *`<## Matterport ##>`*。

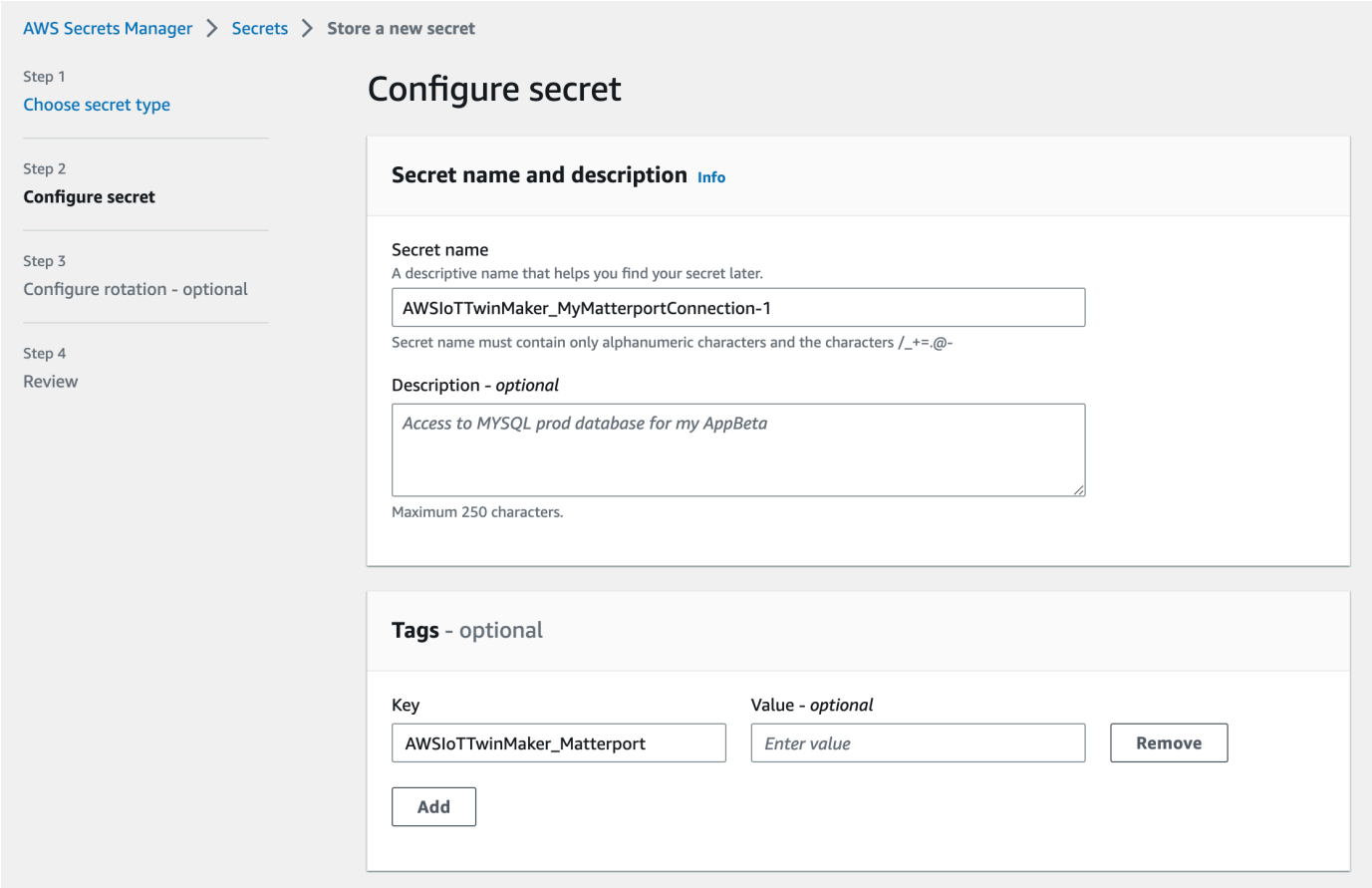
- 创建一个键值对，其中键为 `client_id`，值为 *<## Matterport ##>*。
- 创建一个键值对，其中键为 `client_secret`，值为 *<## Matterport ##>*。

完成后，您应当会有似类于以下示例的配置：



5. 对于加密键，仍选择默认的加密键 `aws/secretsmanager`。
6. 选择下一步转至配置密钥页面。
7. 填写密钥名称和描述字段。
8. 在标签部分为该密钥添加一个标签。

   创建标签时，按以下屏幕截图AWSIoTTwinMaker_Matterport所示分配密钥：



> **Note**
>
> 必须添加一个标签。尽管标签被列为可选，但向 AWS Secrets Manager添加第三方密钥时仍需要标签。

值字段为可选项。提供一个键后，您可以选择添加继续下一步。

9. 在配置旋转页面，选择 Next（下一步）。设置密钥轮换为可选操作。如果您希望完成密钥的添加且不需要轮换，请再次选择 Next（下一步）。有关密钥轮换的更多信息，请参阅轮换 AWS Secrets Manager 密钥。
10. 在查看页面上确认密钥配置。准备好添加密钥后，请选择存储。

有关使用的更多信息 AWS Secrets Manager，请参阅以下 AWS Secrets Manager 文档：

- 使用创建和管理密钥 AWS Secrets Manager
- 什么是 AWS Secrets Manager？

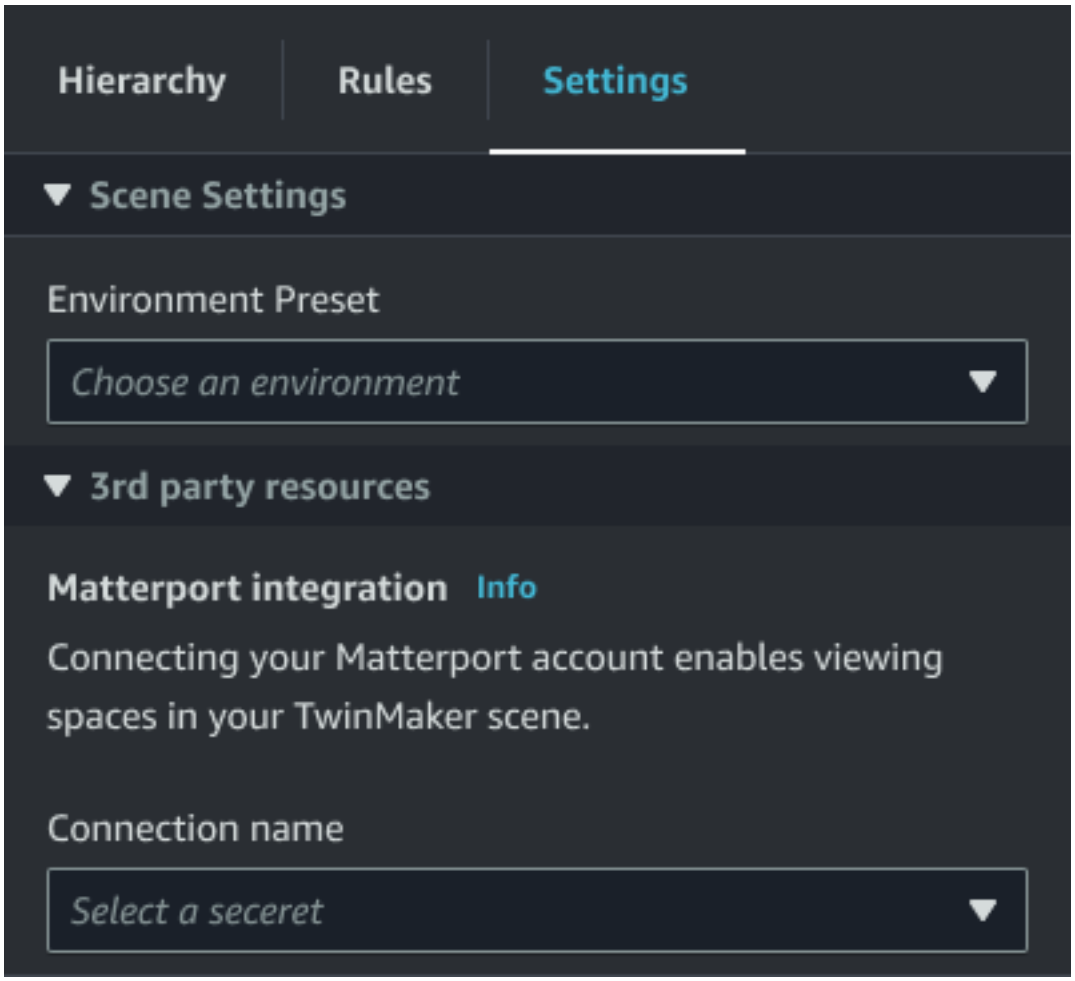
- 轮换 AWS Secrets Manager 秘密

现在，您可以将 AWS IoT TwinMaker Matterport 资源导入场景了。请参阅下一节中的步骤，将 Matterport 空间导入场景 AWS IoT TwinMaker

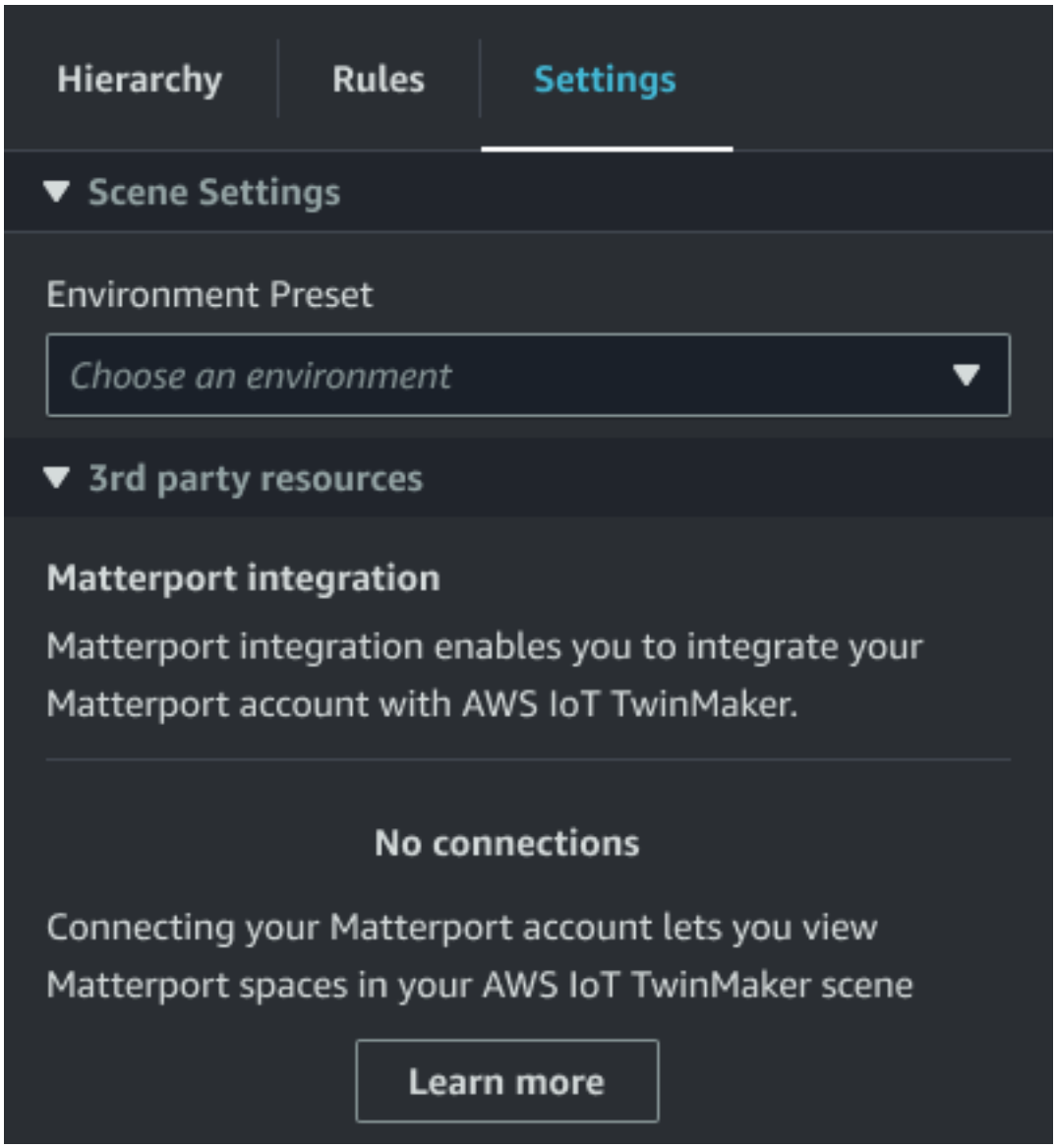
将 Matterport 空间导入场景 AWS IoT TwinMaker

从场景设置页面中选择关联的 Matterport 账户，将 Matterport 扫描添加到场景中。使用以下步骤导入 Matterport 扫描和标签：

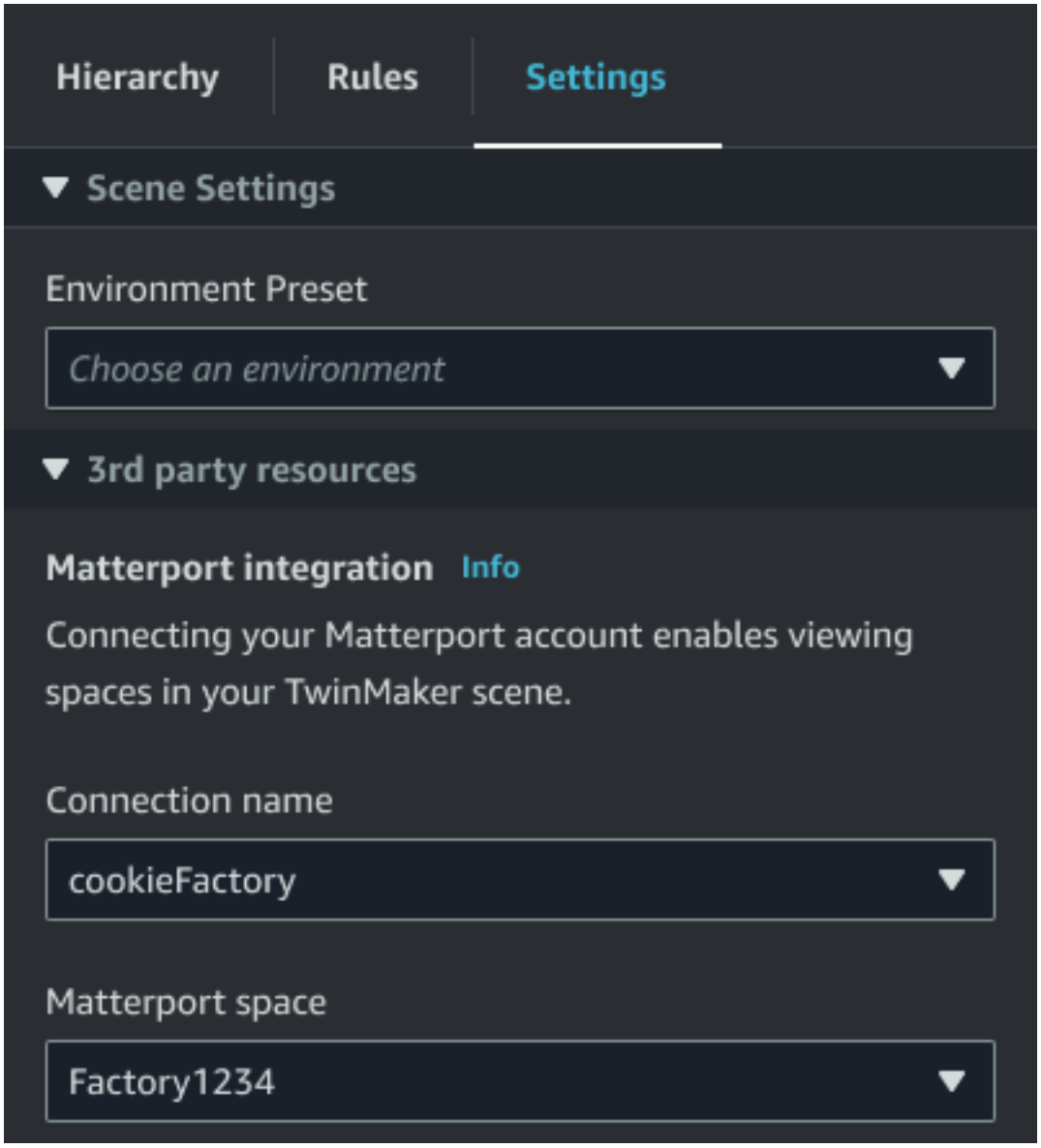
1. 登录 AWS IoT TwinMaker 控制台。
2. 创建或打开要在其中使用 Matterport 空间的现有 AWS IoT TwinMaker 场景。
3. 场景打开后，导航到设置选项卡。
4. 在设置的第三方资源下，找到连接名称，然后输入在以下过程中从 将你的 Matterport 凭证存储在 AWS Secrets Manager 创建的密钥。



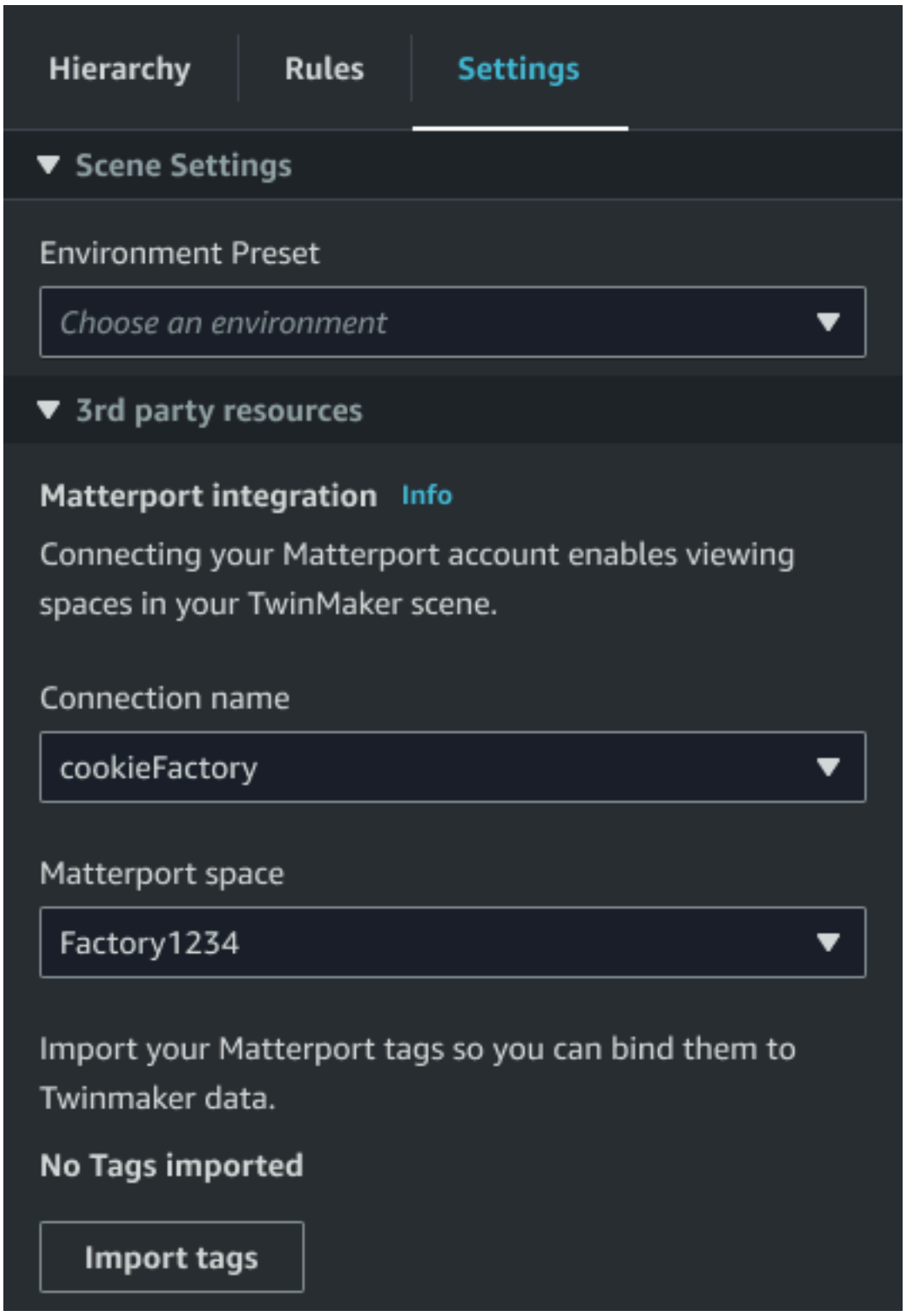
> **Note**
>
> 如果看到一条显示无连接的消息，请导航到 AWS IoT TwinMaker 控制台设置页面，以开始 Matterport 集成的过程。



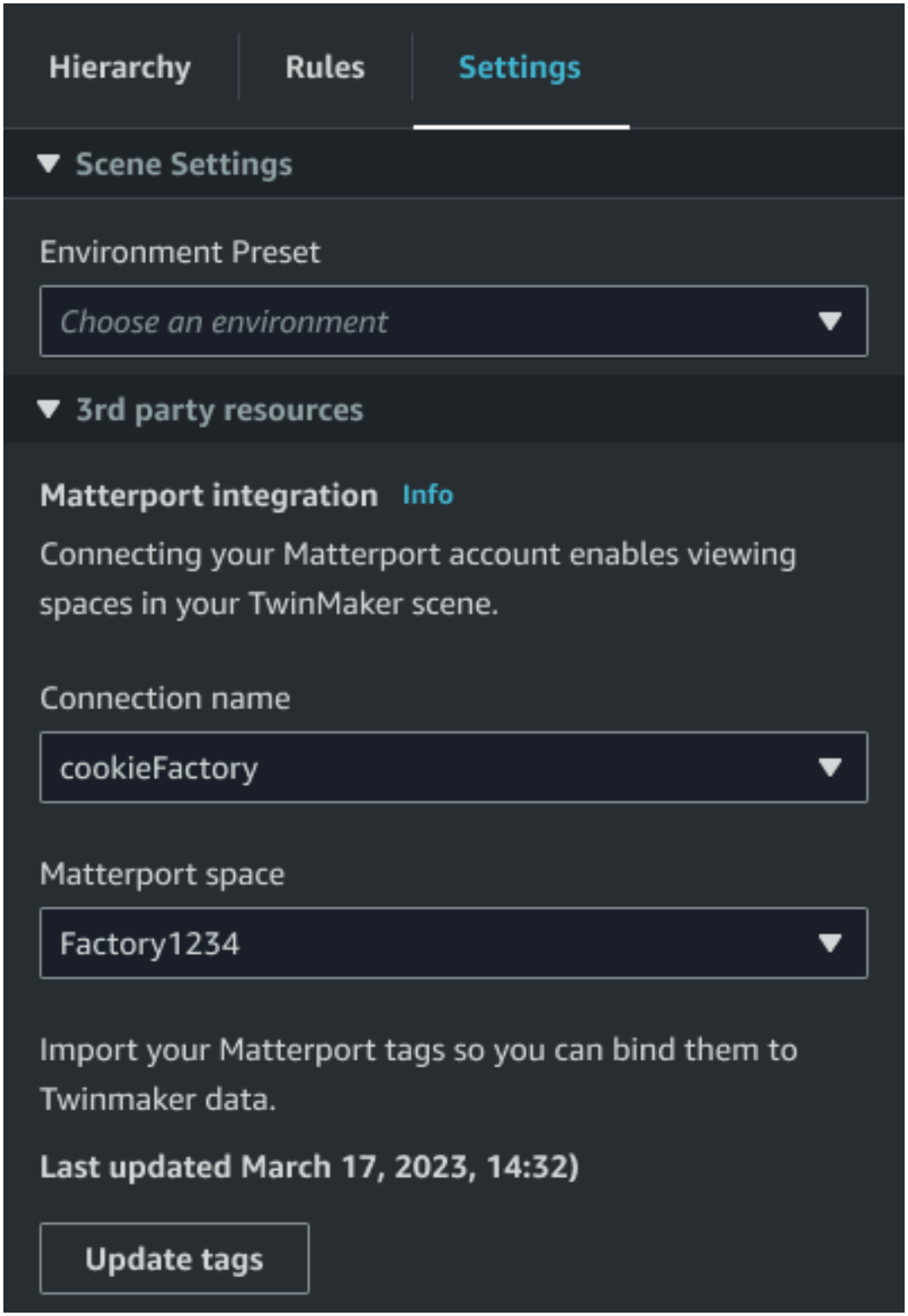
5. 接下来，在 Matterport 空间下拉列表中选择要在场景中使用的 Matterport 空间。



6. 选择空间后，您可以导入 Matterport 标签并将其转换为 AWS IoT TwinMaker 场景标签，方法是按下导入标签按钮。



导入 Matterport 标签后，该按钮将被更新标签按钮所取代。您可以不断更新您的 Matterport 标签， AWS IoT TwinMaker 以便它们始终反映您的 Matterport 账户中的最新更改。



7. 你已经成功 AWS IoT TwinMaker 与 Matterport 集成，现在你的 AWS IoT TwinMaker 场景既有导入的 Matterport 空间又有标签。你可以像处理任何其他场景一样在这个 AWS IoT TwinMaker 场景中工作。

有关使用 AWS IoT TwinMaker 场景的更多信息，请参阅创建和编辑 AWS IoT TwinMaker 场景。

在你的 Gra AWS IoT TwinMaker fana 控制面板中使用 Matterport 空间

将 Matterport 空间导入 AWS IoT TwinMaker 场景后，你可以使用 Grafana 仪表板中的 Matterport 空间查看该场景。如果你已经为 Grafana 配置了，那么你只需打开 Grafana 仪表板，即可 AWS IoT TwinMaker使用导入的 Matterport 空间查看场景。

如果您尚未 AWS IoT TwinMaker 使用 Grafana 进行配置，请先完成 Grafana 集成流程。在 AWS IoT TwinMaker 与 Grafana 集成时，你有两种选择。您可以使用自行管理的 Grafana 实例，也可以使用 Amazon Managed Grafana。

如需详细了解 Grafana 选项和集成过程，请参阅以下文档：

- AWS IoT TwinMaker Grafana 控制面板集成。
- Amazon Managed Grafana。
- 自行管理的 Grafana。

在你 AWS IoT TwinMaker 的 Web 应用程序中使用 Matterport 空格

将 Matterport 空间导入 AWS IoT TwinMaker 场景后，即可在应用程序套件 Web AWS IoT 应用程序中使用 Matterport 空间查看该场景。

要了解有关使用该 AWS IoT 应用套件的更多信息，请参阅以下文档：

- 要了解有关与 AWS IoT 应用套件 AWS IoT TwinMaker 配合使用的更多信息，请参阅使用 AWS IoT TwinMaker 用户界面组件创建自定义 Web 应用程序。
- 要了解有关使用 AWS IoT 应用程序套件的更多信息，请访问AWS IoT 应用程序套件 Github 页面。
- 有关如何使用 AWS IoT 应用套件启动新 Web 应用程序的说明，请访问官方 IoT App Kit 文档页面。

AWS IoT TwinMaker 视频集成

摄像机为数字孪生仿真提供了好机会。您可以使用 AWS IoT TwinMaker 模拟相机的位置和物理条件。在AWS IoT TwinMaker中为您的现场摄像机创建实体，然后使用视频组件将实时视频和元数据从您的站点流式传输至AWS IoT TwinMaker场景或 Grafana 控制面板。

AWS IoT TwinMaker 可通过两种方式通过边缘设备采集视频。您可以使用 Kinesis 视频流的边缘连接器流式传输边缘设备视频，也可以将视频保存在边缘设备上，并使用 MQTT 消息启动视频上传。使用此组件流式传输来自您设备的视频数据，以用于AWS IoT服务。要为 Kinesis Video Streams 生成所需资源并部署边缘连接器，请参阅 Kinesis 视频流边缘连接器入门。 GitHub有关该AWS IoT Greengrass组件的更多信息，请参阅有关Kinesis Video Streams 边缘连接器文档AWS IoT Greengrass。

创建所需 AWS IoT SiteWise 模型并配置 Kinesis Video Streams Greengrass 组件后，您可以通过控制台AWS IoT TwinMaker将边缘视频流式传输或录制到数字孪生应用程序。您还可以在 Grafana 控制面板中查看设备上的实时数据和元数据。有关 Grafana 和 AWS IoT TwinMaker 集成的更多信息，请参阅 AWS IoT TwinMaker Grafana 控制面板集成。

适用 Kinesis 视频流的边缘连接器将流式传输AWS IoT TwinMaker中的视频。

借助 Kinesis 视频流的边缘连接器，您可以将视频和数据流式传输至 AWS IoT TwinMaker场景中的实体。您可使用视频组件完成此操作。要创建用于场景的视频组件，请完成以下程序。

先决条件

在 AWS IoT TwinMaker 场景创建视频组件之前，请确保您已完成了以下先决条件。

- 为 Kinesis 视频流的边缘连接器创建所需的 AWS IoT SiteWise 模型和资产。有关为连接器创建AWS IoT SiteWise资产的更多信息，请参阅 Kinesis 视频流边缘连接器入门。
- 已在您的 AWS IoT Greengrass 设备上部署 Kinesis 视频流边缘连接器。有关部署 Kinesis 视频流边缘连接器组件的更多信息，请参阅部署 自述文件。

为 AWS IoT TwinMaker 场景创建视频组件

完成以下步骤，以为场景的 Kinesis 视频流组件创建边缘连接器。

1. 在AWS IoT TwinMaker 控制台，打开视频组件的目标添加场景。
2. 场景打开后，选择现有实体或创建组件添加目标实体，然后选择 Add component（添加组件)。
3. 在 Add component（添加组件）窗格中，输入组件名称，在 Type（类型）栏选择 com.amazon.iotsitewise.connector.edgevideo。
4. 通过指定您创建的AWS IoT SiteWise摄像机模型，选择资产模型。此名称应采用以下格式：`EdgeConnectorForKVSCameraModel-0abc`，其中末尾的字母和数字字符串与您自己的资产名称匹配。
5. 在 Asset（资产）窗格，选择您想要流失传播的视频的起始 AWS IoT SiteWise 摄像机资产。显示小窗口，可预览当前视频流。

   > **Note**
   >
   > 若要测试您的视频流，请选择test（测试）。此测试发出 MQTT 活动，以启动视频直播。稍等片刻，视频将会出现在播放器中。

6. 若要将视频组件添加至实体，请选择 Add component（添加组件）。

将 Kinesis 视频流中的视频和元数据添加至 Grafana 控制面板

在AWS IoT TwinMaker场景中为实体创建视频组件后，可在 Grafana 中配置视频面板以查看直播。请确保您已将AWS IoT TwinMaker与 Grafana 正确集成。有关更多信息，请参阅 AWS IoT TwinMaker Grafana 控制面板集成。

> **Important**
>
> 要在 Grafana 控制面板中观看视频，必须确保 Grafana 数据源具有适当的 IAM 权限。若要创建所需角色和政策，请参见创建控制面板 IAM 角色。

完成以下步骤，即可在 Grafana 控制面板中查看 Kinesis 视频流和元数据。

1. 打开 AWS IoT TwinMaker 控制面板。
2. 选择 “添加面板”，然后选择 “添加空面板”。
3. 在面板列表中，选择AWS IoT TwinMaker视频播放器面板。
4. 在AWS IoT TwinMaker视频播放器面板中，输入的直播名称 KinesisVideoStreamName，以及您要从中流式传输视频的 Kinesis 视频流的名称。

> **Note**
>
> 要将元数据流式传输至 Grafana 视频面板，必须先创建带有视频流组件的实体。

5. 可选：要将AWS IoT SiteWise资产中的元数据流式传输至视频播放器，在 Entity（实体）窗格，请选择您在AWS IoT TwinMaker场景中创建的AWS IoT TwinMaker实体。在 Component name（组件名称）窗格，选择您为AWS IoT TwinMaker场景中的实体创建的视频组件。

使用 AWS IoT TwinMaker Flink 库

AWS IoT TwinMaker 提供了一个 Flink 库，您可以使用该库向用于数字孪生的外部数据存储读取和写入数据。

若要使用 AWS IoT TwinMaker Flink 库，您可以将其作为自定义连接器安装至 Apache Flink 服务中，然后在 Managed Service for Apache Flink 的 Zeppelin 笔记本电脑中执行 Flink SQL 查询。可升级升级笔记本电脑，以连续运行流处理应用程序。该库利用 AWS IoT TwinMaker 组件从您的工作区检索数据。

AWS IoT TwinMaker Flink 库需要以下内容。

先决条件

1. 包含场景和组件的完全填充工作区。使用内置组件获取AWS 服务（AWS IoT SiteWise 和 Kinesis Video Streams）数据。为第三方来源的数据创建自定义组件类型。有关更多信息，请参阅 ???。
2. 了解带有 Apache Flink 托管服务的 Studio Notebook。这些笔记本电脑由Apache Zeppelin提供支持，采用Apache Flink框架。有关更多信息，请参阅对 Apache Flink for Apache Flink 使用带托管服务的 Studio 笔记本电脑。

有关该库的使用说明，请参阅 AWS IoT TwinMakerFlink 库用户指南。

有关在示例中使用快速入门AWS IoT TwinMaker设置的AWS IoT TwinMaker说明，请参阅示例见解应用程序自述文件。

AWS IoT TwinMaker 中的日志记录和监控

监控是保持 AWS IoT TwinMaker 和您的其他 AWS 解决方案的可靠性、可用性和性能的重要方面。AWS IoT TwinMaker 支持通过以下监控工具来监控服务、在出现错误时进行报告并适时自动采取措施：

- Amazon CloudWatch 实时监控您的 AWS 资源以及在 AWS 上运行的应用程序。您可以收集和跟踪指标，创建自定义的控制平面，以及设置警报以在指定的指标达到您指定的阈值时通知您或采取措施。例如，您可以具有 Amazon EC2 实例的 CloudWatch 跟踪 CPU 使用率或其它指标，并且在需要时自动启动新实例。有关更多信息，请参阅 Amazon CloudWatch 用户指南。
- Amazon CloudWatch Logs 监控、存储和访问来自 AWS IoT TwinMaker 网关、CloudTrail 或其他来源的日志文件。CloudWatch Logs 可以监控日志文件中的信息，并在达到特定阈值时通知您。您还可以在高持久性存储中检索您的日志数据。有关更多信息，请参阅 Amazon CloudWatch Logs 用户指南。
- AWS CloudTrail 捕获由您的 AWS 账户或代表该账户发出的 API 调用和相关事件，并将日志文件传输到您指定的 Simple Storage Service（Amazon S3）存储桶。您可以标识哪些用户和账户调用了 AWS、从中发出调用的源 IP 地址以及调用的发生时间。有关更多信息，请参阅 AWS CloudTrail 用户指南。

主题

使用 Amazon CloudWatch 指标监控 AWS IoT TwinMaker

您可以使用 CloudWatch 监控 AWS IoT TwinMaker。Amazon CloudWatch 会收集原始数据并将其处理为易读且近乎实时的指标。这些统计数据会保存 15 个月，从而使您能够访问历史信息，并能够更好地了解您的 Web 应用程序或服务的执行情况。此外，可以设置用于监测特定阈值的警报，并在达到相应阈值时发送通知或执行操作。有关更多信息，请参阅 Amazon CloudWatch 用户指南。

AWS IoT TwinMaker 将以下部分中列出的指标和维度发布至 `AWS/IoTTwinMaker` 命名空间。

> **Tip**
>
> AWS IoT TwinMaker 每隔一分钟发布一次指标。当您在 CloudWatch 控制台中以图表形式查看这些指标，我们建议您选择 1 分钟时间段，以最高分辨率查看指标数据。

目录

指标

AWS IoT TwinMaker 发布以下指标。

指标

指标	描述
ComponentTypeCreationFailure	此指标报告组件类型是否创建成功。 当组件类型处于 CREATING 状态时，就会发布该指标。当使用架构初始值设定项中所需的属性创建组件类型，并且通过默认值将这些属性设置为实例时，就会发生这种情况。 0表示指标值合格，1表示指标值不合格。 维度：ComponentTypeId、WorkspaceId。 单位：计数
ComponentTypeUpdateFailure	此指标报告组件类型更新是否成功。 当组件类型处于 UPDATING 状态时，就会发布该指标。当使用架构初始值设定项中所需的属性更新组件类型，并且通过默认值将这些属性设置为实例时，就会发生这种情况。 0表示指标值合格，1表示指标值不合格。 维度：ComponentTypeId、WorkspaceId。

指标	描述
	单位：计数
EntityCreationFailure	此指标报告实体创建是否成功。当组件类型处于 CREATING 状态时，就会发布该指标。当通过组件创建实体时，就会发生这种情况。 0表示指标值合格，1表示指标值不合格。 维度：EntityName、EntityId、WorkspaceId。 单位：计数
EntityUpdateFailure	此指标报告实体更新是否成功。当组件类型处于 UPDATING 状态时，就会发布该指标。实体更新时会发生这种情况。 0表示指标值合格，1表示指标值不合格。 维度：EntityName、EntityId、WorkspaceId。 单位：计数
EntityDeletionFailure	此指标报告实体删除是否成功。当组件类型处于 DELETING 状态时，就会发布该指标。删除实体时会发生这种情况。 0表示指标值合格，1表示指标值不合格。 维度：EntityName、EntityId、WorkspaceId。 单位：计数

> **Tip**
>
> 所有指标均已发布至 AWS/IoTTwinMaker 命名空间。

使用 AWS IoT TwinMaker 记录 AWS CloudTrail API 调用

AWS IoT TwinMaker 与 AWS CloudTrail 集成，后者是在 AWS 中记录用户、角色或 AWS IoT TwinMaker 服务所执行操作的服务。CloudTrail 将 AWS IoT TwinMaker 的API 调用作为事件捕获。捕获的调用包含来自 AWS IoT TwinMaker 控制台和代码的 AWS IoT TwinMaker API 操作调用。如果您创建跟踪，则可以使 CloudTrail 事件持续传送到 Amazon S3 存储桶（包括 AWS IoT TwinMaker 的事件）。如果您不配置跟踪记录，则仍可在 CloudTrail 控制台中的 Event history（事件历史记录）中查看最新事件。使用 CloudTrail 收集的信息，您可以确定向 AWS IoT TwinMaker 发出了什么请求、发出请求的 IP 地址、何人发出的请求、请求的发出时间以及其他详细信息。

有关 CloudTrail 的更多信息，请参阅《AWS CloudTrail 用户指南》。

CloudTrail 中的 AWS IoT TwinMaker 信息

在您创建AWS账户时，CloudTrail 自动启用。CloudTrail 记录AWS IoT TwinMaker中发生的支持活动，以及活动记录中的其他AWS服务活动。您可以在 AWS 账户中查看、搜索和下载最新事件。有关更多信息，请参阅使用 CloudTrail 事件历史记录查看事件。

要持续记录 AWS 账户中的事件（包括 AWS IoT TwinMaker 的事件），请创建跟踪。通过跟踪记录，CloudTrail 可将日志文件传送至 Amazon S3 存储桶。预设情况下，在控制台中创建跟踪时，此跟踪应用于所有 AWS 区域。CloudTrail 记录所有 AWS 分区的事件，并将日志文件传送至您指定的 Amazon S3 桶。此外，您可以配置其他 AWS 服务，进一步分析在 CloudTrail 日志中收集的事件数据并采取行动。有关更多信息，请参阅下列内容：

- 创建跟踪概览
- CloudTrail 支持的服务和集成
- 为 CloudTrail 配置 Amazon SNS 通知
- 从多个区域接收 CloudTrail 日志文件和从多个账户接收 CloudTrail 日志文件

CloudTrail 记录多数 AWS IoT TwinMaker 操作，AWS IoT TwinMaker API 参考中介绍了这些操作。

CloudTrail 不记录以下数据平面操作：

- GetPropertyValue
- GetPropertyValueHistory
- BatchPutPropertyValues

每个事件或日志条目都包含有关生成请求的人员信息。身份信息可帮助您确定以下内容：

- 请求是使用根用户凭证还是 用户凭证发出的。
- 请求是使用角色还是联合身份用户的临时安全凭证发出的。
- 请求是否由其它 AWS 服务发出。

有关更多信息，请参阅 CloudTrail userIdentity 元素。

安全性 AWS IoT TwinMaker

云安全 AWS 是重中之重。作为 AWS 客户，您可以受益于专为满足大多数安全敏感型组织的要求而构建的数据中心和网络架构。

安全是双方共同承担 AWS 的责任。责任共担模式将其描述为云的安全性和云中的安全性：

- 云安全 — AWS 负责保护在云中运行 AWS 服务的基础架构 AWS Cloud。 AWS 还为您提供可以安全使用的服务。作为AWS 合规计划合规计划合规计划合的一部分，第三方审计师定期测试和验证我们安全的有效性。要了解适用的合规计划 AWS IoT TwinMaker，请参阅按合规计划划分的范围内的 AWSAWS 服务按合规计划。
- 云端安全-您的责任由您使用的 AWS 服务决定。您还需要对其他因素负责，包括您的数据的敏感性、您公司的要求以及适用的法律法规。

本文档可帮助您了解在使用时如何应用分担责任模型 AWS IoT TwinMaker。以下主题向您介绍如何进行配置 AWS IoT TwinMaker 以满足您的安全和合规性目标。您还将学习如何使用其他 AWS 服务来帮助您监控和保护您的 AWS IoT TwinMaker 资源。

主题

- 中的数据保护 AWS IoT TwinMaker
- Identity and Access Management AWS IoT TwinMaker
- AWS IoT TwinMaker 和接口 VPC 终端节点 (AWS PrivateLink)
- 的合规性验证 AWS IoT TwinMaker
- 韧性在 AWS IoT TwinMaker
- 中的基础设施安全 AWS IoT TwinMaker

中的数据保护 AWS IoT TwinMaker

分 AWS 担责任模型适用于中的数据保护 AWS IoT TwinMaker。如本模型所述 AWS ，负责保护运行所有内容的全球基础架构 AWS Cloud。您负责维护对托管在此基础设施上的内容的控制。您还负责您所使用的 AWS 服务 的安全配置和管理任务。有关数据隐私的更多信息，请参阅数据隐私常见问题。有关欧洲数据保护的信息，请参阅 AWS 安全性博客 上的 AWS 责任共担模式和 GDPR 博客文章。

出于数据保护目的，我们建议您保护 AWS 账户 凭证并使用 AWS IAM Identity Center 或 AWS Identity and Access Management (IAM) 设置个人用户。这样，每个用户只获得履行其工作职责所需的权限。我们还建议您通过以下方式保护数据：

- 对每个账户使用多重身份验证 (MFA)。
- 使用 SSL/TLS 与资源通信。 AWS 我们要求使用 TLS 1.2，建议使用 TLS 1.3。
- 使用设置 API 和用户活动日志 AWS CloudTrail。
- 使用 AWS 加密解决方案以及其中的所有默认安全控件 AWS 服务。
- 使用高级托管安全服务（例如 Amazon Macie），它有助于发现和保护存储在 Amazon S3 中的敏感数据。
- 如果您在 AWS 通过命令行界面或 API 进行访问时需要经过 FIPS 140-2 验证的加密模块，请使用 FIPS 端点。有关可用的 FIPS 端点的更多信息，请参阅《美国联邦信息处理标准（FIPS）第 140-2 版》。

我们强烈建议您切勿将机密信息或敏感信息（如您客户的电子邮件地址）放入标签或自由格式文本字段（如名称字段）。这包括您使用控制台、API AWS IoT TwinMaker 或 SDK 或以其他 AWS 服务 方式使用控制台 AWS CLI、API 或 AWS SDK 的情况。在用于名称的标签或自由格式文本字段中输入的任何数据都可能会用于计费或诊断日志。如果您向外部服务器提供网址，强烈建议您不要在网址中包含凭证信息来验证对该服务器的请求。

静态加密

AWS IoT TwinMaker 如果您愿意，可以将您的工作空间信息存储在该服务为您创建的 Amazon S3 存储桶中。该为您创建的存储桶支持默认服务器加密功能。创建新工作区时，如您选择自己的 Amazon S3 桶，我们建议您启用默认服务器加密。有关 Amazon S3 默认加密的更多信息，请参阅 Amazon S3 存储桶默认服务器加密行为设置。

传输中加密

发送到 AWS IoT TwinMaker 的所有数据均使用 HTTPS 协议通过 TLS 连接发送，因此在传输过程中默认是安全的。

> **Note**
>
> 我们建议您在与 Amazon S3 存储桶地址上使用 HTTPS 作为控件，以便在与 Amazon S3 存储桶 AWS IoT TwinMaker 交互时在传输中强制加密。有关 Amazon S3 存储桶的详细信息，请参阅 创建、配置和使用 Amazon S3 存储桶。

Identity and Access Management AWS IoT TwinMaker

AWS Identity and Access Management (IAM) AWS 服务 可帮助管理员安全地控制对 AWS 资源的访问权限。IAM 管理员控制谁可以进行身份验证（登录）和授权（拥有权限）使用 AWS IoT TwinMaker 资源。您可以使用 IAM AWS 服务 ，无需支付额外费用。

主题

受众

您的使用方式 AWS Identity and Access Management (IAM) 会有所不同，具体取决于您所做的工作 AWS IoT TwinMaker。

服务用户-如果您使用该 AWS IoT TwinMaker 服务完成工作，则您的管理员会为您提供所需的凭证和权限。当你使用更多 AWS IoT TwinMaker 功能来完成工作时，你可能需要额外的权限。了解如何管理访问权限有助于您向管理员请求适合的权限。如果您无法访问 AWS IoT TwinMaker中的特征，请参阅 对 AWS IoT TwinMaker 身份和访问进行故障排除。

服务管理员-如果您负责公司的 AWS IoT TwinMaker 资源，则可能拥有完全访问权限 AWS IoT TwinMaker。您的工作是确定您的服务用户应访问哪些 AWS IoT TwinMaker 功能和资源。然后，您必

须向 IAM 管理员提交请求以更改服务用户的权限。请查看该页面上的信息以了解 IAM 的基本概念。要详细了解您的公司如何将 IAM 与配合使用 AWS IoT TwinMaker，请参阅如何 AWS IoT TwinMaker 与 IAM 配合使用。

IAM 管理员：如果您是 IAM 管理员，您可能希望了解如何编写策略以管理对 AWS IoT TwinMaker的访问权限的详细信息。要查看您可以在 IAM 中使用的 AWS IoT TwinMaker 基于身份的策略示例，请参阅。基于身份的策略示例 AWS IoT TwinMaker

使用身份进行身份验证

身份验证是您 AWS 使用身份凭证登录的方式。您必须以 IAM 用户身份或通过担 AWS 账户根用户任 IAM 角色进行身份验证（登录 AWS）。

您可以使用通过身份源提供的凭据以 AWS 联合身份登录。 AWS IAM Identity Center （IAM Identity Center）用户、贵公司的单点登录身份验证以及您的 Google 或 Facebook 凭据就是联合身份的示例。当您以联合身份登录时，您的管理员以前使用 IAM 角色设置了身份联合验证。当你使用联合访问 AWS 时，你就是在间接扮演一个角色。

根据您的用户类型，您可以登录 AWS Management Console 或 AWS 访问门户。有关登录的更多信息 AWS，请参阅《AWS 登录 用户指南》中的如何登录到您 AWS 账户的。

如果您 AWS 以编程方式访问，则会 AWS 提供软件开发套件 (SDK) 和命令行接口 (CLI)，以便使用您的凭据对请求进行加密签名。如果您不使用 AWS 工具，则必须自己签署请求。有关使用推荐的方法自行签署请求的更多信息，请参阅 IAM 用户指南中的签署 AWS API 请求。

无论使用何种身份验证方法，您可能需要提供其他安全信息。例如， AWS 建议您使用多重身份验证 (MFA) 来提高账户的安全性。要了解更多信息，请参阅《AWS IAM Identity Center 用户指南》中的多重身份验证和《IAM 用户指南》中的在 AWS中使用多重身份验证（MFA）。

AWS 账户 root 用户

创建时 AWS 账户，首先要有一个登录身份，该身份可以完全访问账户中的所有资源 AWS 服务 和资源。此身份被称为 AWS 账户 root 用户，使用您创建账户时使用的电子邮件地址和密码登录即可访问该身份。强烈建议您不要使用根用户执行日常任务。保护好根用户凭证，并使用这些凭证来执行仅根用户可以执行的任务。有关要求您以根用户身份登录的任务的完整列表，请参阅《IAM 用户指南》中的需要根用户凭证的任务。

联合身份

作为最佳实践，要求人类用户（包括需要管理员访问权限的用户）使用与身份提供商的联合身份验证 AWS 服务 通过临时证书进行访问。

联合身份是指您的企业用户目录、Web 身份提供商、Identity C enter 目录中的用户，或者任何使用 AWS 服务 通过身份源提供的凭据进行访问的用户。 AWS Directory Service当联合身份访问时 AWS 账户，他们将扮演角色，角色提供临时证书。

要集中管理访问权限，建议您使用 AWS IAM Identity Center。您可以在 IAM Identity Center 中创建用户和群组，也可以连接并同步到您自己的身份源中的一组用户和群组，以便在您的所有 AWS 账户 和应用程序中使用。有关 IAM Identity Center 的信息，请参阅《AWS IAM Identity Center 用户指南》中的什么是 IAM Identity Center？。

IAM 用户和群组

I AM 用户是您 AWS 账户 内部对个人或应用程序具有特定权限的身份。在可能的情况下，我们建议使用临时凭证，而不是创建具有长期凭证（如密码和访问密钥）的 IAM 用户。但是，如果您有一些特定的使用场景需要长期凭证以及 IAM 用户，建议您轮换访问密钥。有关更多信息，请参阅《IAM 用户指南》中的对于需要长期凭证的使用场景定期轮换访问密钥。

IAM 组是一个指定一组 IAM 用户的身份。您不能使用组的身份登录。您可以使用组来一次性为多个用户指定权限。如果有大量用户，使用组可以更轻松地管理用户权限。例如，您可能具有一个名为 IAMAdmins 的组，并为该组授予权限以管理 IAM 资源。

用户与角色不同。用户唯一地与某个人员或应用程序关联，而角色旨在让需要它的任何人代入。用户具有永久的长期凭证，而角色提供临时凭证。要了解更多信息，请参阅《IAM 用户指南》中的何时创建 IAM 用户（而不是角色）。

IAM 角色

I AM 角色是您内部具有特定权限 AWS 账户 的身份。它类似于 IAM 用户，但与特定人员不关联。您可以 AWS Management Console 通过切换角色在中临时担任 IAM 角色。您可以通过调用 AWS CLI 或 AWS API 操作或使用自定义 URL 来代入角色。有关使用角色的方法的更多信息，请参阅《IAM 用户指南》中的 使用 IAM 角色。

具有临时凭证的 IAM 角色在以下情况下很有用：

- 联合用户访问 – 要向联合身份分配权限，请创建角色并为角色定义权限。当联合身份进行身份验证时，该身份将与角色相关联并被授予由此角色定义的权限。有关联合身份验证的角色的信息，请参阅《IAM 用户指南》中的 为第三方身份提供商创建角色。如果您使用 IAM Identity Center，则需要配置权限集。为控制您的身份在进行身份验证后可以访问的内容，IAM Identity Center 将权限集与 IAM 中的角色相关联。有关权限集的信息，请参阅《AWS IAM Identity Center 用户指南》中的权限集。
- 临时 IAM 用户权限 – IAM 用户可代入 IAM 用户或角色，以暂时获得针对特定任务的不同权限。

- 跨账户存取 – 您可以使用 IAM 角色以允许不同账户中的某个人（可信主体）访问您的账户中的资源。角色是授予跨账户访问权限的主要方式。但是，对于某些资源 AWS 服务，您可以将策略直接附加到资源（而不是使用角色作为代理）。要了解用于跨账户访问的角色和基于资源的策略之间的差别，请参阅《IAM 用户指南》中的 IAM 角色与基于资源的策略有何不同。
- 跨服务访问 — 有些 AWS 服务 使用其他 AWS 服务服务中的功能。例如，当您在某个服务中进行调用时，该服务通常会在 Amazon EC2 中运行应用程序或在 Amazon S3 中存储对象。服务可能会使用发出调用的主体的权限、使用服务角色或使用服务相关角色来执行此操作。
  - 转发访问会话 (FAS) — 当您使用 IAM 用户或角色在中执行操作时 AWS，您被视为委托人。使用某些服务时，您可能会执行一个操作，然后此操作在其他服务中启动另一个操作。FAS 使用调用委托人的权限以及 AWS 服务 向下游服务发出请求的请求。 AWS 服务只有当服务收到需要与其他 AWS 服务 或资源交互才能完成的请求时，才会发出 FAS 请求。在这种情况下，您必须具有执行这两个操作的权限。有关发出 FAS 请求时的策略详情，请参阅转发访问会话。
  - 服务角色 - 服务角色是服务代表您在您的账户中执行操作而分派的 IAM 角色。IAM 管理员可以在 IAM 中创建、修改和删除服务角色。有关更多信息，请参阅《IAM 用户指南》中的创建向 AWS 服务委派权限的角色。
  - 服务相关角色-服务相关角色是一种与服务相关联的服务角色。 AWS 服务服务可以代入代表您执行操作的角色。服务相关角色出现在您的中 AWS 账户 ，并且归服务所有。IAM 管理员可以查看但不能编辑服务相关角色的权限。
- 在 A@@ mazon EC2 上运行的应用程序 — 您可以使用 IAM 角色管理在 EC2 实例上运行并发出 AWS CLI 或 AWS API 请求的应用程序的临时证书。这优先于在 EC2 实例中存储访问密钥。要向 EC2 实例分配 AWS 角色并使其可供其所有应用程序使用，您需要创建附加到该实例的实例配置文件。实例配置文件包含角色，并使 EC2 实例上运行的程序能够获得临时凭证。有关更多信息，请参阅《IAM 用户指南》中的 使用 IAM 角色为 Amazon EC2 实例上运行的应用程序授予权限。

要了解是使用 IAM 角色还是 IAM 用户，请参阅《IAM 用户指南》中的何时创建 IAM 角色（而不是用户）。

使用策略管理访问

您可以 AWS 通过创建策略并将其附加到 AWS 身份或资源来控制中的访问权限。策略是其中的一个对象 AWS ，当与身份或资源关联时，它会定义其权限。 AWS 在委托人（用户、root 用户或角色会话）发出请求时评估这些策略。策略中的权限确定是允许还是拒绝请求。大多数策略都以 JSON 文档的 AWS 形式存储在中。有关 JSON 策略文档的结构和内容的更多信息，请参阅《IAM 用户指南》中的 JSON 策略概览。

管理员可以使用 AWS JSON 策略来指定谁有权访问什么。也就是说，哪个主体 可以对什么资源执行操作，以及在什么条件下执行。

默认情况下，用户和角色没有权限。要授予用户对所需资源执行操作的权限，IAM 管理员可以创建 IAM 策略。管理员随后可以向角色添加 IAM 策略，用户可以代入角色。

IAM 策略定义操作的权限，无关乎您使用哪种方法执行操作。例如，假设您有一个允许 `iam:GetRole` 操作的策略。拥有该策略的用户可以从 AWS Management Console AWS CLI、或 AWS API 获取角色信息。

基于身份的策略

基于身份的策略是可附加到身份（如 IAM 用户、用户组或角色）的 JSON 权限策略文档。这些策略控制用户和角色可在何种条件下对哪些资源执行哪些操作。要了解如何创建基于身份的策略，请参阅《IAM 用户指南》中的创建 IAM 策略。

基于身份的策略可以进一步归类为内联策略或托管式策略。内联策略直接嵌入单个用户、组或角色中。托管策略是独立的策略，您可以将其附加到中的多个用户、群组和角色 AWS 账户。托管策略包括 AWS 托管策略和客户托管策略。要了解如何在托管式策略和内联策略之间进行选择，请参阅《IAM 用户指南》中的在托管式策略与内联策略之间进行选择。

基于资源的策略

基于资源的策略是附加到资源的 JSON 策略文档。基于资源的策略的示例包括 IAM 角色信任策略和 Simple Storage Service（Amazon S3）存储桶策略。在支持基于资源的策略的服务中，服务管理员可以使用它们来控制对特定资源的访问。对于在其中附加策略的资源，策略定义指定主体可以对该资源执行哪些操作以及在什么条件下执行。您必须在基于资源的策略中指定主体。委托人可以包括账户、用户、角色、联合用户或 AWS 服务。

基于资源的策略是位于该服务中的内联策略。您不能在基于资源的策略中使用 IAM 中的 AWS 托管策略。

访问控制列表 (ACL)

访问控制列表（ACL）控制哪些主体（账户成员、用户或角色）有权访问资源。ACL 与基于资源的策略类似，尽管它们不使用 JSON 策略文档格式。

Amazon S3 和 Amazon VPC 就是支持 ACL 的服务示例。 AWS WAF要了解有关 ACL 的更多信息，请参阅《Amazon Simple Storage Service 开发人员指南》 中的访问控制列表（ACL）概览。

其他策略类型

AWS 支持其他不太常见的策略类型。这些策略类型可以设置更常用的策略类型向您授予的最大权限。

- 权限边界 - 权限边界是一个高级功能，用于设置基于身份的策略可以为 IAM 实体（IAM 用户或角色）授予的最大权限。您可为实体设置权限边界。这些结果权限是实体基于身份的策略及其权限边界的交集。在 `Principal` 中指定用户或角色的基于资源的策略不受权限边界限制。任一项策略中的显式拒绝将覆盖允许。有关权限边界的更多信息，请参阅《IAM 用户指南》中的 IAM 实体的权限边界。
- 服务控制策略 (SCP)-SCP 是 JSON 策略，用于指定组织或组织单位 (OU) 的最大权限。 AWS Organizations AWS Organizations 是一项用于对您的企业拥有的多 AWS 账户 项进行分组和集中管理的服务。如果在组织内启用了所有功能，则可对任意或全部账户应用服务控制策略 (SCP)。SCP 限制成员账户中的实体（包括每个 AWS 账户根用户实体）的权限。有关 Organizations 和 SCP 的更多信息，请参阅《AWS Organizations 用户指南》中的 SCP 的工作原理。
- 会话策略 – 会话策略是当您以编程方式为角色或联合用户创建临时会话时作为参数传递的高级策略。结果会话的权限是用户或角色的基于身份的策略和会话策略的交集。权限也可以来自基于资源的策略。任一项策略中的显式拒绝将覆盖允许。有关更多信息，请参阅《IAM 用户指南》中的会话策略。

多个策略类型

当多个类型的策略应用于一个请求时，生成的权限更加复杂和难以理解。要了解在涉及多种策略类型时如何 AWS 确定是否允许请求，请参阅 IAM 用户指南中的策略评估逻辑。

如何 AWS IoT TwinMaker 与 IAM 配合使用

在使用 IAM 管理访问权限之前 AWS IoT TwinMaker，请先了解有哪些 IAM 功能可供使用 AWS IoT TwinMaker。

您可以搭配使用的 IAM 功能 AWS IoT TwinMaker

IAM 功能	AWS IoT TwinMaker 支持
基于身份的策略	是
基于资源的策略	否

IAM 功能	AWS IoT TwinMaker 支持
策略操作	是
策略资源	是
策略条件键	是
ACL	否
ABAC（策略中的标签）	部分
临时凭证	是
主体权限	是
服务角色	是
服务相关角色	否

要全面了解大多数 IAM 功能 AWS IoT TwinMaker 以及其他 AWS 服务如何与大多数 IAM 功能配合使用，请参阅AWS IAM Identity Center 用户指南中的与 IAM 配合使用的AWS 服务。

基于身份的策略 AWS IoT TwinMaker

支持基于身份的策略	是

基于身份的策略是可附加到身份（如 IAM 用户、用户组或角色）的 JSON 权限策略文档。这些策略控制用户和角色可在何种条件下对哪些资源执行哪些操作。要了解如何创建基于身份的策略，请参阅 IAM 用户指南中的创建 IAM 策略。

通过使用 IAM 基于身份的策略，您可以指定允许或拒绝的操作和资源以及允许或拒绝操作的条件。您无法在基于身份的策略中指定主体，因为它适用于其附加的用户或角色。要了解可在 JSON 策略中使用的所有元素，请参阅《IAM 用户指南》中的 IAM JSON 策略元素引用。

基于身份的策略示例 AWS IoT TwinMaker

要查看 AWS IoT TwinMaker 基于身份的策略的示例，请参阅。基于身份的策略示例 AWS IoT TwinMaker

内部基于资源的政策 AWS IoT TwinMaker

支持基于资源的策略	否

基于资源的策略是附加到资源的 JSON 策略文档。基于资源的策略的示例包括 IAM 角色信任策略和 Simple Storage Service（Amazon S3）存储桶策略。在支持基于资源的策略的服务中，服务管理员可以使用它们来控制对特定资源的访问。对于在其中附加策略的资源，策略定义指定主体可以对该资源执行哪些操作以及在什么条件下执行。您必须在基于资源的策略中指定主体。委托人可以包括账户、用户、角色、联合用户或 AWS 服务。

要启用跨账户存取，您可以将整个账户或其他账户中的 IAM 实体指定为基于资源的策略中的主体。将跨账户主体添加到基于资源的策略只是建立信任关系工作的一半而已。当委托人和资源处于不同位置时 AWS 账户，可信账户中的 IAM 管理员还必须向委托人实体（用户或角色）授予访问资源的权限。他们通过将基于身份的策略附加到实体以授予权限。但是，如果基于资源的策略向同一个账户中的主体授予访问权限，则不需要额外的基于身份的策略。有关更多信息，请参阅《IAM 用户指南》中的 IAM 角色与基于资源的策略有何不同。

的政策行动 AWS IoT TwinMaker

支持策略操作	是

管理员可以使用 AWS JSON 策略来指定谁有权访问什么。也就是说，哪个主体 可以对什么资源 执行操作，以及在什么 条件 下执行。

JSON 策略的 `Action` 元素描述可用于在策略中允许或拒绝访问的操作。策略操作通常与关联的 AWS API 操作同名。有一些例外情况，例如没有匹配 API 操作的仅限权限 操作。还有一些操作需要在策略中执行多个操作。这些附加操作称为相关操作。

在策略中包含操作以授予执行关联操作的权限。

要查看 AWS IoT TwinMaker 操作列表，请参阅《服务授权参考》 AWS IoT TwinMaker中定义的操作。

正在执行的策略操作在操作前 AWS IoT TwinMaker 使用以下前缀：

```
iottwinmaker
```

要在单个语句中指定多项操作，请使用逗号将它们隔开。

```
"Action": [
   "iottwinmaker:action1",
   "iottwinmaker:action2"
         ]
```

要查看 AWS IoT TwinMaker 基于身份的策略的示例，请参阅。基于身份的策略示例 AWS IoT TwinMaker

的政策资源 AWS IoT TwinMaker

支持策略资源	是

管理员可以使用 AWS JSON 策略来指定谁有权访问什么。也就是说，哪个主体 可以对什么资源 执行操作，以及在什么条件 下执行。

`Resource` JSON 策略元素指定要向其应用操作的一个或多个对象。语句必须包含 `Resource` 或 `NotResource` 元素。作为最佳实践，请使用其 Amazon 资源名称（ARN）指定资源。对于支持特定资源类型（称为资源级权限）的操作，您可以执行此操作。

对于不支持资源级权限的操作（如列出操作），请使用通配符 (*) 指示语句应用于所有资源。

```
"Resource": "*"
```

要查看 AWS IoT TwinMaker 资源类型及其 ARN 的列表，请参阅《服务授权参考》 AWS IoT TwinMaker中定义的资源。要了解可以在哪些操作中指定每个资源的 ARN，请参阅 AWS IoT TwinMaker定义的操作。

要查看 AWS IoT TwinMaker 基于身份的策略的示例，请参阅。基于身份的策略示例 AWS IoT TwinMaker

的策略条件密钥 AWS IoT TwinMaker

支持特定于服务的策略条件键	是

管理员可以使用 AWS JSON 策略来指定谁有权访问什么。也就是说，哪个主体可以对什么资源执行操作，以及在什么条件下执行。

在 Condition 元素（或 Condition 块）中，可以指定语句生效的条件。Condition 元素是可选的。您可以创建使用条件运算符（例如，等于或小于）的条件表达式，以使策略中的条件与请求中的值相匹配。

如果您在一个语句中指定多个 Condition 元素，或在单个 Condition 元素中指定多个键，则 AWS 使用逻辑 AND 运算评估它们。如果您为单个条件键指定多个值，则使用逻辑OR运算来 AWS 评估条件。在授予语句的权限之前必须满足所有的条件。

在指定条件时，您也可以使用占位符变量。例如，只有在使用 IAM 用户名标记 IAM 用户时，您才能为其授予访问资源的权限。有关更多信息，请参阅《IAM 用户指南》中的 IAM 策略元素：变量和标签。

AWS 支持全局条件密钥和特定于服务的条件密钥。要查看所有 AWS 全局条件键，请参阅 IAM 用户指南中的AWS 全局条件上下文密钥。

要查看 AWS IoT TwinMaker 条件键列表，请参阅《服务授权参考》 AWS IoT TwinMaker中的条件密钥。要了解可以使用条件键的操作和资源，请参阅由定义的操作 AWS IoT TwinMaker。

要查看 AWS IoT TwinMaker 基于身份的策略的示例，请参阅。基于身份的策略示例 AWS IoT TwinMaker

AWS IoT TwinMaker中的访问控制列表 (ACL)

支持 ACL	否

访问控制列表（ACL）控制哪些主体（账户成员、用户或角色）有权访问资源。ACL 与基于资源的策略类似，尽管它们不使用 JSON 策略文档格式。

基于属性的访问控制 (ABAC) AWS IoT TwinMaker

支持 ABAC（策略中的标签）	部分

基于属性的访问权限控制（ABAC）是一种授权策略，该策略基于属性来定义权限。在中 AWS，这些属性称为标签。您可以向 IAM 实体（用户或角色）和许多 AWS 资源附加标签。标记实体和资源是 ABAC 的第一步。然后设计 ABAC 策略，以在主体的标签与他们尝试访问的资源标签匹配时允许操作。

ABAC 在快速增长的环境中非常有用，并在策略管理变得繁琐的情况下可以提供帮助。

要基于标签控制访问，您需要使用 aws:ResourceTag/*key-name*、aws:RequestTag/*key-name* 或 aws:TagKeys 条件键在策略的条件元素中提供标签信息。

如果某个服务对于每种资源类型都支持所有这三个条件键，则对于该服务，该值为是。如果某个服务仅对于部分资源类型支持所有这三个条件键，则该值为部分。

有关 ABAC 的更多信息,请参阅《IAM 用户指南》中的什么是 ABAC？。要查看设置 ABAC 步骤的教程，请参阅《IAM 用户指南》中的使用基于属性的访问权限控制（ABAC）。

将临时凭证与配合使用 AWS IoT TwinMaker

支持临时凭证	是

当你使用临时证书登录时，有些 AWS 服务 不起作用。有关更多信息，包括哪些 AWS 服务 适用于临时证书，请参阅 IAM 用户指南中的AWS 服务 与 IAM 配合使用的信息。

如果您使用除用户名和密码之外的任何方法登录，则 AWS Management Console 使用的是临时证书。例如，当您 AWS 使用公司的单点登录 (SSO) 链接进行访问时，该过程会自动创建临时证书。当您以用户身份登录控制台，然后切换角色时，您还会自动创建临时凭证。有关切换角色的更多信息，请参阅《IAM 用户指南》中的 切换到角色（控制台）。

您可以使用 AWS CLI 或 AWS API 手动创建临时证书。然后，您可以使用这些临时证书进行访问 AWS。 AWS 建议您动态生成临时证书，而不是使用长期访问密钥。有关更多信息，请参阅 IAM 中的临时安全凭证。

的跨服务主体权限 AWS IoT TwinMaker

支持转发访问会话 (FAS)	是

当您使用 IAM 用户或角色在中执行操作时 AWS，您被视为委托人。使用某些服务时，您可能会执行一个操作，然后此操作在其他服务中启动另一个操作。FAS 使用调用委托人的权限以及 AWS 服务 向下游服务发出请求的请求。 AWS 服务只有当服务收到需要与其他 AWS 服务 或资源交互才能完成的请求时，才会发出 FAS 请求。在这种情况下，您必须具有执行这两个操作的权限。有关发出 FAS 请求时的策略详细信息，请参阅转发访问会话。

AWS IoT TwinMaker的服务角色

支持服务角色	是

服务角色是由一项服务担任、代表您执行操作的 IAM 角色。IAM 管理员可以在 IAM 中创建、修改和删除服务角色。有关更多信息，请参阅《IAM 用户指南》中的创建向 AWS 服务委派权限的角色。

> **Warning**
>
> 更改服务角色的权限可能会中断 AWS IoT TwinMaker 功能。只有在 AWS IoT TwinMaker 提供操作指导时才编辑服务角色。

的服务相关角色 AWS IoT TwinMaker

支持服务相关角色	否

服务相关角色是一种与服务相关联的 AWS 服务服务角色。服务可以代入代表您执行操作的角色。服务相关角色出现在您的中 AWS 账户 ，并且归服务所有。IAM 管理员可以查看但不能编辑服务相关角色的权限。

有关创建或管理服务相关角色的详细信息，请参阅能够与 IAM 搭配使用的AWS 服务。在表中查找服务相关角色列中包含 Yes 的表。选择是链接以查看该服务的服务相关角色文档。

基于身份的策略示例 AWS IoT TwinMaker

默认情况下，用户和角色没有创建或修改 AWS IoT TwinMaker 资源的权限。他们也无法使用 AWS Management Console、 AWS Command Line Interface (AWS CLI) 或 AWS API 执行任务。要授予用户对所需资源执行操作的权限，IAM 管理员可以创建 IAM 策略。管理员随后可以向角色添加 IAM 策略，用户可以代入角色。

要了解如何使用这些示例 JSON 策略文档创建基于 IAM 身份的策略，请参阅 IAM 用户指南中的 创建 IAM 策略。

有关由定义的操作和资源类型的详细信息 AWS IoT TwinMaker，包括每种资源类型的 ARN 格式，请参阅《服务授权参考》 AWS IoT TwinMaker中的操作、资源和条件密钥。

主题

策略最佳实践

基于身份的策略决定了某人是否可以在您的账户中创建、访问或删除 AWS IoT TwinMaker 资源。这些操作可能会使 AWS 账户产生成本。创建或编辑基于身份的策略时，请遵循以下准则和建议：

- 开始使用 AWS 托管策略并转向最低权限权限 — 要开始向用户和工作负载授予权限，请使用为许多常见用例授予权限的AWS 托管策略。它们在你的版本中可用 AWS 账户。我们建议您通过定义针对您的用例的 AWS 客户托管策略来进一步减少权限。有关更多信息，请参阅《IAM 用户指南》中的AWS 托管策略或工作职能的AWS 托管策略。
- 应用最低权限 – 在使用 IAM 策略设置权限时，请仅授予执行任务所需的权限。为此，您可以定义在特定条件下可以对特定资源执行的操作，也称为最低权限许可。有关使用 IAM 应用权限的更多信息，请参阅《IAM 用户指南》中的 IAM 中的策略和权限。
- 使用 IAM 策略中的条件进一步限制访问权限 – 您可以向策略添加条件来限制对操作和资源的访问。例如，您可以编写策略条件来指定必须使用 SSL 发送所有请求。如果服务操作是通过特定的方式使用的，则也可以使用条件来授予对服务操作的访问权限 AWS 服务，例如 AWS CloudFormation。有关更多信息，请参阅《IAM 用户指南》中的 IAM JSON 策略元素：条件。
- 使用 IAM Access Analyzer 验证您的 IAM 策略，以确保权限的安全性和功能性 – IAM Access Analyzer 会验证新策略和现有策略，以确保策略符合 IAM 策略语言 (JSON) 和 IAM 最佳实践。IAM Access Analyzer 提供 100 多项策略检查和可操作的建议，以帮助您制定安全且功能性强的策略。有关更多信息，请参阅《IAM 用户指南》中的 IAM Acess Analyzer 策略验证。
- 需要多重身份验证 (MFA)-如果 AWS 账户您的场景需要 IAM 用户或根用户，请启用 MFA 以提高安全性。若要在调用 API 操作时需要 MFA，请将 MFA 条件添加到您的策略中。有关更多信息，请参阅《IAM 用户指南》中的 配置受 MFA 保护的 API 访问。

有关 IAM 中的最佳实操的更多信息，请参阅《IAM 用户指南》中的 IAM 中的安全最佳实操。

使用 AWS IoT TwinMaker 控制台

要访问 AWS IoT TwinMaker 控制台，您必须拥有一组最低权限。这些权限必须允许您列出和查看有关您的 AWS IoT TwinMaker 资源的详细信息 AWS 账户。如果创建比必需的最低权限更为严格的基于身份的策略，对于附加了该策略的实体（用户或角色），控制台将无法按预期正常运行。

对于仅调用 AWS CLI 或 AWS API 的用户，您无需为其设置最低控制台权限。相反，只允许访问与其尝试执行的 API 操作相匹配的操作。

为确保用户和角色仍然可以使用 AWS IoT TwinMaker 控制台，还需要将 AWS IoT TwinMaker `ConsoleAccess`或`ReadOnly` AWS 托管策略附加到实体。有关更多信息，请参阅 AWS IAM Identity Center 用户指南中的为用户添加权限。

允许用户查看他们自己的权限

该示例说明了您如何创建策略，以允许 IAM 用户查看附加到其用户身份的内联和托管式策略。此策略包括在控制台上或使用 AWS CLI 或 AWS API 以编程方式完成此操作的权限。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ViewOwnUserInfo",
            "Effect": "Allow",
            "Action": [
                "iam:GetUserPolicy",
                "iam:ListGroupsForUser",
                "iam:ListAttachedUserPolicies",
                "iam:ListUserPolicies",
                "iam:GetUser"
            ],
            "Resource": ["arn:aws:iam::*:user/${aws:username}"]
        },
        {
            "Sid": "NavigateInConsole",
            "Effect": "Allow",
            "Action": [
                "iam:GetGroupPolicy",
                "iam:GetPolicyVersion",
                "iam:GetPolicy",
                "iam:ListAttachedGroupPolicies",
                "iam:ListGroupPolicies",
                "iam:ListPolicyVersions",
                "iam:ListPolicies",
                "iam:ListUsers"
            ],
            "Resource": "*"
        }
    ]
```

```
}
```

对 AWS IoT TwinMaker 身份和访问进行故障排除

使用以下信息来帮助您诊断和修复在使用 AWS IoT TwinMaker 和 IAM 时可能遇到的常见问题。

主题

- 我无权在以下位置执行操作 AWS IoT TwinMaker
- 我无权执行 iam：PassRole
- 我想允许我以外的人 AWS 账户 访问我的 AWS IoT TwinMaker 资源

我无权在以下位置执行操作 AWS IoT TwinMaker

如果您收到错误提示，表明您无权执行某个操作，则您必须更新策略以允许执行该操作。

当 `mateojackson` IAM 用户尝试使用控制台查看有关虚构 *`my-example-widget`* 资源的详细信息，但不拥有虚构 `iottwinmaker:`*`GetWidget`* 权限时，会发生以下示例错误。

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 iottwinmaker:GetWidget on resource: my-example-widget
```

在此情况下，必须更新 `mateojackson` 用户的策略，以允许使用 `iottwinmaker:`*`GetWidget`* 操作访问 *`my-example-widget`* 资源。

如果您需要帮助，请联系您的 AWS 管理员。您的管理员是提供登录凭证的人。

我无权执行 iam：PassRole

如果您收到一个错误，表明您无权执行 `iam:PassRole` 操作，则必须更新策略以允许您将角色传递给 AWS IoT TwinMaker。

有些 AWS 服务 允许您将现有角色传递给该服务，而不是创建新的服务角色或服务相关角色。为此，您必须具有将角色传递到服务的权限。

当名为 `marymajor` 的 IAM 用户尝试使用控制台在 AWS IoT TwinMaker中执行操作时，会发生以下示例错误。但是，服务必须具有服务角色所授予的权限才可执行此操作。Mary 不具有将角色传递到服务的权限。

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

在这种情况下，必须更新 Mary 的策略以允许她执行 `iam:PassRole` 操作。

如果您需要帮助，请联系您的 AWS 管理员。您的管理员是提供登录凭证的人。

我想允许我以外的人 AWS 账户 访问我的 AWS IoT TwinMaker 资源

您可以创建一个角色，以便其他账户中的用户或您组织外的人员可以使用该角色来访问您的资源。您可以指定谁值得信赖，可以担任角色。对于支持基于资源的策略或访问控制列表（ACL）的服务，您可以使用这些策略向人员授予对您的资源的访问权。

要了解更多信息，请参阅以下内容：

- 要了解是否 AWS IoT TwinMaker 支持这些功能，请参阅如何 AWS IoT TwinMaker 与 IAM 配合使用。
- 要了解如何提供对您拥有的资源的访问权限 AWS 账户 ，请参阅 IAM 用户指南中的向您拥有 AWS 账户 的另一个 IAM 用户提供访问权限。
- 要了解如何向第三方提供对您的资源的访问权限 AWS 账户，请参阅 IAM 用户指南中的向第三方提供访问权限。 AWS 账户
- 要了解如何通过身份联合验证提供访问权限，请参阅《IAM 用户指南》中的为经过外部身份验证的用户（身份联合验证）提供访问权限。
- 要了解使用角色和基于资源的策略进行跨账户存取之间的差别，请参阅《IAM 用户指南》中的 IAM 角色与基于资源的策略有何不同。

将服务相关角色用于 AWS IoT TwinMaker

AWS IoT TwinMaker 使用 AWS Identity and Access Management (IAM) 服务相关角色。服务相关角色是一种与之直接关联的 IAM 角色的独特类型。 AWS IoT TwinMaker服务相关角色由服务预定义 AWS IoT TwinMaker ，包括该服务代表您调用其他 AWS 服务所需的所有权限。

服务相关角色使设置变得 AWS IoT TwinMaker 更加容易，因为您不必手动添加必要的权限。 AWS IoT TwinMaker 定义其服务相关角色的权限，除非另有定义，否则 AWS IoT TwinMaker 只能担任其角色。定义的权限包括信任策略和权限策略，以及不能附加到任何其他 IAM 实体的权限策略。

只有在首先删除相关资源后，您才能删除服务相关角色。这样可以保护您的 AWS IoT TwinMaker 资源，因为您不会无意中删除访问资源的权限。

有关支持服务相关角色的其他服务的信息，请参阅与 IAM 配合使用的AWS 服务，并查找服务相关角色列中显示为是的服务。选择是和链接，查看该服务的服务相关角色文档。

的服务相关角色权限 AWS IoT TwinMaker

AWS IoT TwinMaker 使用名为的服务相关角色 AWSServiceRoleForIoTTwinMaker— AWS IoT TwinMaker 允许代表您调用其他 AWS 服务并同步其资源。

AWSServiceRoleForIoTTwinMaker 服务相关角色信任以下服务来代入该角色：

- `iottwinmaker.amazonaws.com`

名为的角色权限策略 AWSIoTTwinMakerServiceRolePolicy AWS IoT TwinMaker 允许对指定资源完成以下操作：

- 操作：`all your iotsitewise asset and asset-model resources` 上的 `iotsitewise:DescribeAsset, iotsitewise:ListAssets, iotsitewise:DescribeAssetModel, and iotsitewise:ListAssetModels, iottwinmaker:GetEntity, iottwinmaker:CreateEntity, iottwinmaker:UpdateEntity, iottwinmaker:DeleteEntity, iottwinmaker:ListEntities, iottwinmaker:GetComponentType, iottwinmaker:CreateComponentType, iottwinmaker:UpdateComponentType, iottwinmaker:DeleteComponentType, iottwinmaker:ListComponentTypes`

您必须配置允许用户、组或角色创建、编辑或删除服务相关角色的权限。有关更多信息，请参阅《IAM 用户指南》中的服务相关角色权限。

为创建服务相关角色 AWS IoT TwinMaker

您无需手动创建服务相关角色。当您在 AWS Management Console、或 AWS API 中同步 AWS IoT SiteWise 资产和资产模型（资产同步）时， AWS IoT TwinMaker 会为您创建服务相关角色。 AWS CLI

如果您删除该服务相关角色，然后需要再次创建，您可以使用相同流程在账户中重新创建此角色。当您同步 AWS IoT SiteWise 资产和资产模型（资产同步）时， AWS IoT TwinMaker 会再次为您创建服务相关角色。

您还可以使用 IAM 控制台通过 “IoT TwinMaker -托管角色” 用例创建服务相关角色。在 AWS CLI 或 AWS API 中，使用服务名称创建服务相关角色。`iottwinmaker.amazonaws.com`有关更多信息，

请参阅 IAM 用户指南 中的创建服务相关角色。如果您删除了此服务相关角色，可以使用同样的过程再次创建角色。

编辑的服务相关角色 AWS IoT TwinMaker

AWS IoT TwinMaker 不允许您编辑 AWSServiceRoleForIoTTwinMaker 服务相关角色。创建服务相关角色后，您将无法更改角色的名称，因为可能有多种实体引用该角色。但是可以使用 IAM 编辑角色描述。有关更多信息，请参阅《IAM 用户指南》中的编辑服务相关角色。

删除的服务相关角色 AWS IoT TwinMaker

如果不再需要使用某个需要服务相关角色的功能或服务，我们建议您删除该角色。这样就没有未被主动监控或维护的未使用实体。但是，您必须先清理所有仍在使用您的服务相关角色的 ServiceLinked-Workspaces，然后才能手动删除该角色。

> **Note**
>
> 如果您尝试删除资源时 AWS IoT TwinMaker 服务正在使用该角色，则删除可能会失败。如果发生这种情况，请等待几分钟后重试。

使用 IAM 手动删除服务相关角色

使用 IAM 控制台 AWS CLI、或 AWS API 删除 AWSServiceRoleForIoTTwinMaker 服务相关角色。有关更多信息，请参阅《IAM 用户指南》中的删除服务相关角色。

AWS IoT TwinMaker 服务相关角色支持的区域

AWS IoT TwinMaker 支持在提供服务的所有区域中使用服务相关角色。有关更多信息，请参阅AWS 区域和端点。

AWS 的托管策略 AWS IoT TwinMaker

要向用户、群组和角色添加权限，使用 AWS 托管策略比自己编写策略要容易得多。创建仅为团队提供所需权限的 IAM 客户管理型策略需要时间和专业知识。要快速入门，您可以使用我们的 AWS 托管策略。这些策略涵盖常见使用案例，可在您的 AWS 账户中使用。有关 AWS 托管策略的更多信息，请参阅 IAM 用户指南中的AWS 托管策略。

AWS 服务维护和更新 AWS 托管策略。您无法更改 AWS 托管策略中的权限。服务偶尔会向 AWS 托管策略添加额外权限以支持新功能。此类更新会影响附加策略的所有身份（用户、组和角色）。当启动新功能或新操作可用时，服务最有可能会更新 AWS 托管式策略。服务不会从 AWS 托管策略中移除权限，因此策略更新不会破坏您的现有权限。

此外，还 AWS 支持跨多个服务的工作职能的托管策略。例如，ReadOnly访问 AWS 管理策略提供对所有 AWS 服务和资源的只读访问权限。当服务启动一项新功能时， AWS 会为新操作和资源添加只读权限。有关工作职能策略的列表和说明，请参阅《IAM 用户指南》中的适用于工作职能的AWS 托管式策略。

AWS 托管策略： AWSIoTTwinMakerServiceRolePolicy

您无法附加 AWSIoTTwinMakerServiceRolePolicy 到您的 IAM 实体。此策略授予参与者权限，允许 代表您执行操作。有关更多信息，请参阅 的服务相关角色权限 AWS IoT TwinMaker。

名为的角色权限策略 AWSIoTTwinMakerServiceRolePolicy AWS IoT TwinMaker 允许对指定资源完成以下操作：

- 操作：`all your iotsitewise asset and asset-model resources` 上的 `iotsitewise:DescribeAsset, iotsitewise:ListAssets, iotsitewise:DescribeAssetModel, and iotsitewise:ListAssetModels, iottwinmaker:GetEntity, iottwinmaker:CreateEntity, iottwinmaker:UpdateEntity, iottwinmaker:DeleteEntity, iottwinmaker:ListEntities, iottwinmaker:GetComponentType, iottwinmaker:CreateComponentType, iottwinmaker:UpdateComponentType, iottwinmaker:DeleteComponentType, iottwinmaker:ListComponentTypes`

```
{
    "Version": "2012-10-17",
    "Statement": [{
            "Sid": "SiteWiseAssetReadAccess",
            "Effect": "Allow",
```

```
            "Action": [
                "iotsitewise:DescribeAsset"
            ],
            "Resource": [
                "arn:aws:iotsitewise:*:*:asset/*"
            ]
        },
        {
            "Sid": "SiteWiseAssetModelReadAccess",
            "Effect": "Allow",
            "Action": [
                "iotsitewise:DescribeAssetModel"
            ],
            "Resource": [
                "arn:aws:iotsitewise:*:*:asset-model/*"
            ]
        },
        {
            "Sid": "SiteWiseAssetModelAndAssetListAccess",
            "Effect": "Allow",
            "Action": [
                "iotsitewise:ListAssets",
                "iotsitewise:ListAssetModels"
            ],
            "Resource": [
                "*"
            ]
        },
        {
            "Sid": "TwinMakerAccess",
            "Effect": "Allow",
            "Action": [
                "iottwinmaker:GetEntity",
                "iottwinmaker:CreateEntity",
                "iottwinmaker:UpdateEntity",
                "iottwinmaker:DeleteEntity",
                "iottwinmaker:ListEntities",
                "iottwinmaker:GetComponentType",
                "iottwinmaker:CreateComponentType",
                "iottwinmaker:UpdateComponentType",
                "iottwinmaker:DeleteComponentType",
                "iottwinmaker:ListComponentTypes"
            ],
            "Resource": [
```

```
                "arn:aws:iottwinmaker:*:*:workspace/*"
            ],
            "Condition": {
                "ForAnyValue:StringEquals": {
                    "iottwinmaker:linkedServices": [
                        "IOTSITEWISE"
                    ]
                }
            }
        }
    ]
}
```

AWS IoT TwinMakerAWS 托管策略的更新

查看自该服务开始跟踪这些更改以来 AWS 托管策略更新的详细信息。有关此页面更改的自动提示，请订阅 文档历史记录页面上的 RSS 信息源。

更改	描述	日期
AWSIoTTwinMakerServiceRolePolicy— 添加了政策	AWS IoT TwinMaker 添加了名为的角色权限策略，AWSIoTTwinMakerServiceRolePolicy 该策略 AWS IoT TwinMaker 允许对指定资源完成以下操作： • 操作：all your iotsitewise asset and asset-model resources 上的 iotsitewise:DescribeAsset, iotsitewise:ListAssets, iotsitewise:DescribeAssetModel, and iotsitewise:ListAs	2023 年 11 月 17 日

更改	描述	日期
	setModels, iottwinmaker:GetEntity, iottwinmaker:CreateEntity, iottwinmaker:UpdateEntity, iottwinmaker:DeleteEntity, iottwinmaker:ListEntities, iottwinmaker:GetComponentType, iottwinmaker:CreateComponentType, iottwinmaker:UpdateComponentType, iottwinmaker:DeleteComponentType, iottwinmaker:ListComponentTypes 有关更多信息，请参阅 的服务相关角色权限 AWS IoT TwinMaker。	
已开始跟踪更改	开始跟踪其 AWS 托管策略的更改。	2022 年 5 月 11 日

AWS IoT TwinMaker 和接口 VPC 终端节点 (AWS PrivateLink)

您可通过创建 VPC 接口端点在虚拟私有云 (VPC) 和 AWS IoT TwinMaker 之间创建私有连接。接口端点由其提供支持，您无需互联网网关AWS PrivateLink、网络地址转换 (NAT) 设备、VPN 连接或 Di AWS rect Connect 连接即可使用它私密访问 AWS IoT TwinMaker API。您的 VPC 中的实例不需要公有 IP 地址即可与 AWS IoT TwinMaker API 通信。您的 VPC 和 VPC 之间的流量 AWS IoT TwinMaker 不会离开 Amazon 网络。

每个接口端点均由子网中的一个或多个弹性网络接口表示。

有关更多信息，请参阅 Amazon VPC 用户指南中的接口 VPC 终端节点 (AWS PrivateLink)。

AWS IoT TwinMaker VPC 终端节点的注意事项

在为设置接口 VPC 终端节点之前 AWS IoT TwinMaker，请查看 Amazon VPC 用户指南中的接口终端节点属性和限制。

AWS IoT TwinMaker 支持从您的 VPC 调用其所有 API 操作。

- 对于数据平面 API 操作，请使用以下端点：

  ```
  data.iottwinmaker.region.amazonaws.com
  ```

  数据平面 API 操作包括以下内容：

  - GetProperty价值
  - GetPropertyValueHistory
  - BatchPutPropertyValues
- 针对控制平面 API 操作，请使用以下端点：

  ```
  api.iottwinmaker.region.amazonaws.com
  ```

  支持的控制平面 API 操作包含以下内容：

  - CreateComponent类型
  - CreateEntity
  - CreateScene
  - CreateWorkspace
  - DeleteComponent类型
  - DeleteEntity
  - DeleteScene
  - DeleteWorkspace
  - GetComponent类型
  - GetEntity
  - GetScene

- GetWorkspace
- ListComponent类型
- ListComponent类型
- ListEntities
- ListScenes
- ListTagsForResource
- ListWorkspaces
- TagResource
- UntagResource
- UpdateComponent类型
- UpdateEntity
- UpdateScene
- UpdateWorkspace

为 AWS IoT TwinMaker创建接口 VPC 端点

您可以使用 Amazon VPC 控制台或 AWS Command Line Interface (AWS CLI) 为 AWS IoT TwinMaker 服务创建 VPC 终端节点。有关更多信息，请参阅《Amazon VPC 用户指南》中的创建接口端点。

使用以下服务名称为 AWS IoT TwinMaker 其创建一个 VPC 终端节点。

- 对于数据平面 API 操作，请使用以下服务名称：

```
com.amazonaws.region.iottwinmaker.data
```

- 对于控制平面 API 操作，请使用以下服务名称：

```
com.amazonaws.region.iottwinmaker.api
```

例如，如果您为终端节点启用私有 DNS，则可以使用该终端节点的默认 DNS 名称向发 AWS IoT TwinMaker 出 API 请求`iottwinmaker.us-east-1.amazonaws.com`。

有关更多信息，请参阅《Amazon VPC 用户指南》中的通过接口端点访问服务。

AWS IoT TwinMaker PrivateLink 在以下区域受支持：

- us-east-1

  以下可用区支持该 ControlPlane 服务：`use1-az1`、`use1-az2`、和`use1-az6`。

  以下可用区支持该 DataPlane 服务：`use1-az1`、`use1-az2`、和`use1-az4`。
- us-west-2

  以下可用区支持 ControlPlane 和 DataPlane 服务：`usw2-az1`、`usw2-az2`、和`usw2-az3`。
- eu-west-1
- eu-central-1
- ap-southeast-1
- ap-southeast-2

有关可用区的更多信息，请参阅AWS 资源的可用区 ID-Res AWS ource Access Manager。

AWS IoT TwinMaker 通过接口 VPC 终端节点进行访问

创建接口终端节点时， AWS IoT TwinMaker 会生成可用于与之通信的终端节点特定的 DNS 主机名。AWS IoT TwinMaker默认情况下启用私有 DNS。有关更多信息，请参阅 Amazon VPC 用户指南中的使用私有托管区域。

如果为端点启用私有 DNS，则可以通过以下 VPC 端点向 AWS IoT TwinMaker 发出 API 请求。

- 对于数据平面 API 操作，请使用以下端点。将 *##* 更换为您的 AWS 区域。

  ```
  data.iottwinmaker.region.amazonaws.com
  ```
- 对于控制平面 API 操作，请使用以下端点。将 *##* 更换为您的 AWS 区域。

  ```
  api.iottwinmaker.region.amazonaws.com
  ```

如果您为端点禁用私有 DNS，则必须执行以下操作，才能通过端点访问 AWS IoT TwinMaker ：

- 在 API 请求中指定 VPC 端点 URL。
  - 对于数据平面 API 操作，请使用以下端点 URL。将 *vpc-endpoint-id* 和 *##* 更换为您的 VPC 端点 ID 和区域。

```
vpc-endpoint-id.data.iottwinmaker.region.vpce.amazonaws.com
```

- 对于控制平面 API 操作，请使用以下端点 URL。将 *vpc-endpoint-id* 和 *##* 更换为您的 VPC 端点 ID 和区域。

```
vpc-endpoint-id.api.iottwinmaker.region.vpce.amazonaws.com
```

- 禁用主机前缀注入功能。当您调用每个 API 操作时， AWS CLI 和 AWS SDK 会在服务端点前面加上各种主机前缀。这会导致 AWS CLI 和 AWS 软件开发工具包在您指定 VPC 终端节点 AWS IoT TwinMaker 时生成无效的 URL。

> **Important**
>
> 您无法在 AWS CLI 或 AWS Tools for PowerShell中禁用主机前缀注入。这意味着，如果您禁用了私有 DNS，您将无法使用 AWS CLI 或 AWS Tools for PowerShell AWS IoT TwinMaker 通过 VPC 终端节点进行访问。如果您想使用这些工具 AWS IoT TwinMaker 通过终端节点进行访问，请启用私有 DNS。

有关如何禁用 AWS SDK 中的主机前缀注入的更多信息，请参阅以下文档中关于 SDK 的部分：

- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Go v2
- AWS SDK for Java
- AWS SDK for Java 2.x
- AWS SDK for JavaScript
- AWS SDK for .NET
- AWS SDK for PHP
- AWS SDK for Python (Boto3)
- AWS SDK for Ruby

有关更多信息，请参阅《Amazon VPC 用户指南》中的通过接口端点访问服务。

为创建 VPC 终端节点策略 AWS IoT TwinMaker

您可以为 VPC 端点附加控制对 AWS IoT TwinMaker的访问的端点策略。该策略指定以下信息：

- 可执行操作的主体。
- 可执行的操作。
- 可对其执行操作的资源。

有关更多信息，请参阅《Amazon VPC 用户指南》中的使用 VPC 端点控制对服务的访问。

示例：用于 AWS IoT TwinMaker 操作的 VPC 终端节点策略

以下是的终端节点策略示例 AWS IoT TwinMaker。当关联到终端节点时，此策略授予 AWS 账户中的 IAM 用户`iottwinmakeradmin123456789012`对所有资源进行列出的 AWS IoT TwinMaker 操作的访问权限。

```
{
   "Statement":[
      {
        "Principal": {
            "AWS": "arn:aws:iam::123456789012:user/role"
                },
         "Resource": "*",
         "Effect":"Allow",
         "Action":[
            "iottwinmaker:CreateEntity",
            "iottwinmaker:GetScene",
            "iottwinmaker:ListEntities"
         ]
        }
    ]
}
```

的合规性验证 AWS IoT TwinMaker

要了解是否属于特定合规计划的范围，请参阅AWS 服务 “按合规计划划分的范围” ”，然后选择您感兴趣的合规计划。 AWS 服务 有关一般信息，请参阅AWS 合规计划AWS。

您可以使用下载第三方审计报告 AWS Artifact。有关更多信息，请参阅中的 “下载报告” 中的 “ AWS Artifact。

您在使用 AWS 服务 时的合规责任取决于您的数据的敏感性、贵公司的合规目标以及适用的法律和法规。 AWS 提供了以下资源来帮助实现合规性：

- 安全与合规性快速入门指南 — 这些部署指南讨论了架构注意事项，并提供了在这些基础上 AWS 部署以安全性和合规性为重点的基准环境的步骤。
- 在 A@@ mazon Web Services 上构建 HIPAA 安全与合规架构 — 本白皮书描述了各公司如何使用 AWS 来创建符合 HIPAA 资格的应用程序。

> **Note**
>
> 并非所有 AWS 服务 人都符合 HIPAA 资格。有关更多信息，请参阅符合 HIPAA 要求的服务参考。

- AWS 合规资源AWS — 此工作簿和指南集可能适用于您所在的行业和所在地区。
- AWS 客户合规指南 — 从合规角度了解责任共担模式。这些指南总结了保护的最佳实践， AWS 服务并将指南映射到跨多个框架（包括美国国家标准与技术研究院 (NIST)、支付卡行业安全标准委员会 (PCI) 和国际标准化组织 (ISO)）的安全控制。
- 使用AWS Config 开发人员指南中的规则评估资源 — 该 AWS Config 服务评估您的资源配置在多大程度上符合内部实践、行业准则和法规。
- AWS Security Hub— 这 AWS 服务 提供了您内部安全状态的全面视图 AWS。Security Hub 通过安全控件评估您的 AWS 资源并检查其是否符合安全行业标准和最佳实践。有关受支持服务及控件的列表，请参阅 Security Hub 控件参考。
- Amazon GuardDuty — 它通过监控您的 AWS 账户环境中是否存在可疑和恶意活动，来 AWS 服务检测您的工作负载、容器和数据面临的潜在威胁。 GuardDuty 通过满足某些合规性框架规定的入侵检测要求，可以帮助您满足各种合规性要求，例如 PCI DSS。
- AWS Audit Manager— 这 AWS 服务 可以帮助您持续审计 AWS 使用情况，从而简化风险管理以及对法规和行业标准的合规性。

韧性在 AWS IoT TwinMaker

AWS 全球基础设施是围绕 AWS 区域 可用区构建的。 AWS 区域 提供多个物理隔离和隔离的可用区，这些可用区通过低延迟、高吞吐量和高度冗余的网络连接。利用可用区，您可以设计和操作在可用区之间无中断地自动实现失效转移的应用程序和数据库。与传统的单个或多个数据中心基础设施相比，可用区具有更高的可用性、容错性和可扩展性。

有关 AWS 区域 和可用区的更多信息，请参阅AWS 全球基础设施。

除了 AWS 全球基础架构外，还 AWS IoT TwinMaker 提供多项功能来帮助支持您的数据弹性和备份需求。

中的基础设施安全 AWS IoT TwinMaker

作为一项托管服务， AWS IoT TwinMaker 受到《Amazon Web Services：安全流程概述》白皮书中描述的 AWS 全球网络安全程序的保护。

您可以使用 AWS 已发布的 API 调用 AWS IoT TwinMaker 通过网络进行访问。客户端必须支持传输层安全性（TLS）1.2 或更高版本。建议使用 TLS 1.3 或更高版本。客户端还必须支持具有完全向前保密（PFS）的密码套件，例如 Ephemeral Diffie-Hellman（DHE）或 Elliptic Curve Ephemeral Diffie-Hellman（ECDHE）。大多数现代系统（如 Java 7 及更高版本）都支持这些模式。

此外，必须使用访问密钥 ID 和与 IAM 委托人关联的秘密访问密钥来对请求进行签名。或者，您可以使用 AWS Security Token Service（AWS STS）生成临时安全凭证来对请求进行签名。

端点和配额

AWS IoT TwinMaker 端点和配额

您可以在AWS 一般参考中找到有关 AWS IoT TwinMaker 终端节点和配额的信息。

- 有关服务端点的信息，请参阅 AWS IoT TwinMaker 服务端点。
- 有关配额的信息，请参阅 AWS IoT TwinMaker 服务配额。
- 有关 API 节流限制的信息，请参阅 AWS IoT TwinMaker API 节流限制。

有关 AWS IoT TwinMaker 终端节点的其他信息

要以编程方式连接 AWS IoT TwinMaker，请使用端点。如您使用 HTTP 客户端，则需要按如下方式为控制面板和数据面板 API 添加前缀。但是，没有必要在 AWS SDK 和 AWS Command Line Interface 命令中添加前缀，因为它们会自动添加必要的前缀。

- 用于控制面板 API 的 `api` 前缀。例如，`api.iottwinmaker.us-west-1.amazonaws.com`。
- 用于数据面板 API 的 `data` 前缀。例如，`data.iottwinmaker.us-west-1.amazonaws.com`。

AWS IoT TwinMaker 用户指南的文档历史记录

下表介绍了 AWS IoT TwinMaker 的文档版本。

变更	说明	日期
新的服务相关角色和新的 IAM policy	AWS IoT TwinMaker添加了一个名为 AWSServiceRoleForIoTTwinMaker“” 的新服务相关角色。 AWS IoT TwinMaker 添加了这个新的服务相关角色AWS IoT TwinMaker，允许您调用其他AWS服务并同步其资源。新的 AWSIoTTwinMakerServiceRolePolicy IAM 策略已附加到此角色，该策略授予代表您调用其他AWS服务和同步其资源的权限。AWS IoT TwinMaker	2023 年 11 月 17 日
初始版本	首次发布 AWS IoT TwinMaker 用户指南	2021 年 11 月 30 日

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。