1.17.0 版用户指南

AWS SimSpace Weaver

AWS SimSpace Weaver: 1.17.0 版用户指南

Table of Contents

终止支持通知：2026 年 5 月 20 日， AWS 将终止对的支持。 AWS SimSpace Weaver 2026 年 5 月 20 日之后，您将无法再访问 SimSpace Weaver 控制台或 SimSpace Weaver 资源。有关更多信息，请参阅AWS SimSpace Weaver 终止支持。

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。

什么是 AWS SimSpace Weaver？

AWS SimSpace Weaver 是一项服务，可用于在中构建和运行大规模空间模拟。 AWS Cloud例如，您可以创建人群模拟、大型真实环境以及沉浸式交互体验。

借助 SimSpace Weaver，您可以将模拟工作负载分配到多个亚马逊弹性计算云 (Amazon EC2) 实例。SimSpace Weaver 为您部署底层 AWS 基础设施，并处理模拟数据管理和运行您的模拟的 Amazon EC2 实例之间的网络通信。

的关键概念 SimSpace Weaver

模拟或游戏会受到运行它的计算机的限制。随着虚拟世界的规模不断扩大以及复杂性不断增加，处理性能会逐渐降低。计算时间会更长，系统内存更容易耗尽，客户端帧速率也会下降。对于不需要实时性能的模拟，这可能只会带来少许烦恼。但是，对于业务关键型应用场景，如果处理延迟增加，则会导致成本攀升。如果您的模拟或游戏需要实时性能，那么性能下降就必然会成为问题。

对于达到性能限制的模拟，常见的解决方案是采取简化措施。通常，用户数量众多的在线游戏会在不同的服务器上部署虚拟世界的副本并分散用户，从而解决扩展问题。

SimSpace Weaver 通过在空间上划分虚拟世界并将各个部分分布在中运行的计算实例集群来解决扩展问题。 AWS Cloud计算实例协同工作，并行处理整个模拟世界。对其中的所有对象以及与其连接的所有客户端而言，您的模拟世界依然是一个单一的集成空间。您不必再因硬件性能限制而简化模拟，而是可以在云中添加更多计算容量。

主题

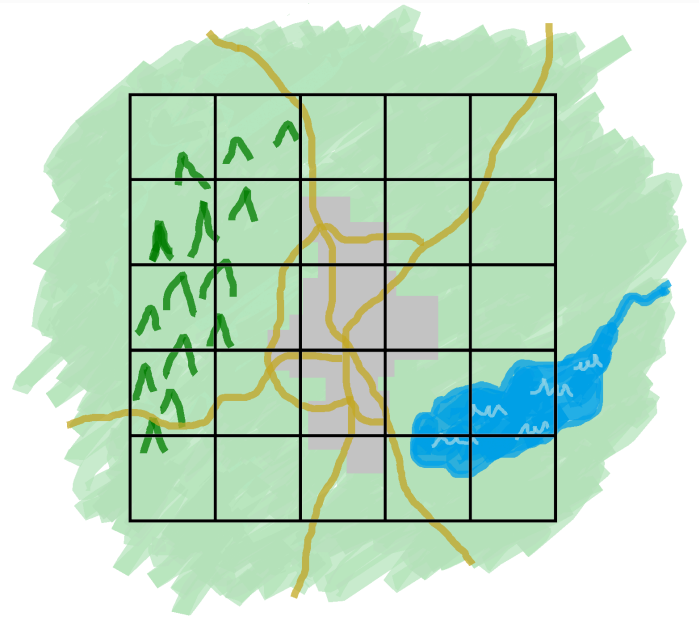
- 如何 SimSpace Weaver 运作
- 你怎么用 SimSpace Weaver
- 模拟架构
- 工作线程和资源单位
- 模拟时钟
- 分区
- State Fabric
- 实体
- 应用程序

如何 SimSpace Weaver 运作

您的模拟由一个包含许多对象的世界组成。有些对象（例如，人和车辆）会移动和做事情。其他对象（例如，树木和建筑物）是静态的。在中 SimSpace Weaver，实体是模拟世界中的一个对象。

您可以定义模拟世界的边界，然后将其划分为网格。您不需要创建在整个网格上运行的模拟逻辑，而是可以创建在网格中的一个单元格上运行的模拟逻辑。在中 SimSpace Weaver，空间应用程序是您编写的程序，用于实现网格单元的仿真逻辑。这包括单元格中所有实体的逻辑。空间应用程序的所有权区域是空间应用程序控制的网格单元格。

> **Note**
>
> 在中 SimSpace Weaver，“应用程序” 一词可以指应用程序的代码或该代码的运行实例。



您的模拟世界划分为一个网格

您将模拟世界划分为一个网格。每个空间应用程序都为该网格中的单个单元格实现模拟逻辑。

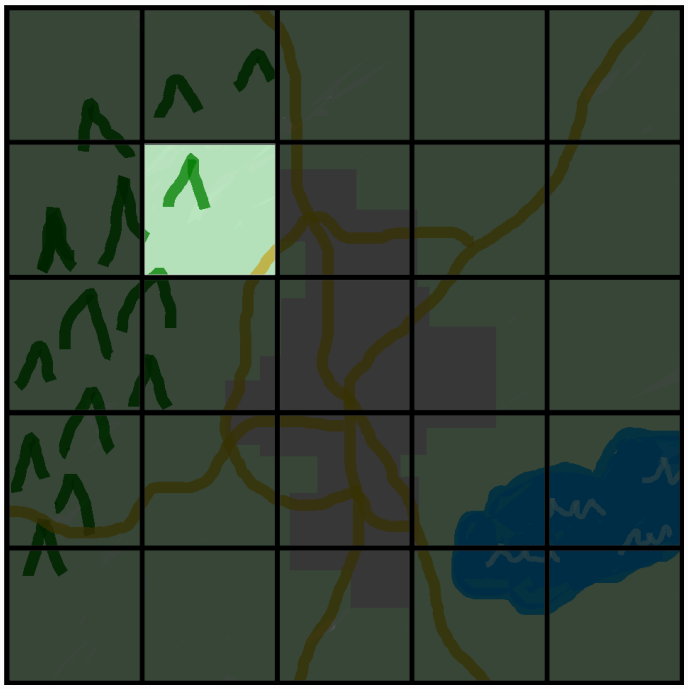
SimSpace Weaver 为格网的每个单元运行空间应用程序代码的实例。所有空间应用程序实例都并行运行。本质上， SimSpace Weaver 将您的整体仿真分为多个较小的模拟。每个较小的模拟都处理整个仿

真世界的一部分。 SimSpace Weaver 可以在中的多个亚马逊弹性计算云 (Amazon EC2) 实例（称为工作程序）上分发和运行这些较小的模拟。 AWS Cloud一个工作线程可运行多个空间应用程序。

实体可在模拟世界中移动。如果实体进入另一个空间应用程序的所有权区域（网格中的另一个单元格），则新区域的空间应用程序所有者将接管该实体的控制权。如果您的模拟在多个工作线程上运行，则实体可以从一个工作线程上的空间应用程序控制下移到另一个工作线程上的空间应用程序中。当一个实体转移到另一个工作线程时，会 SimSpace Weaver 处理底层网络通信。

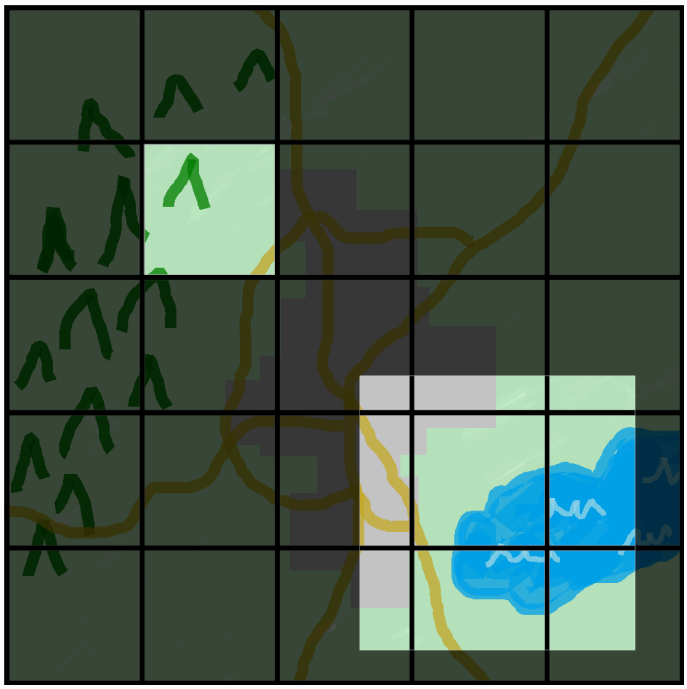
订阅

空间应用程序在世界中的视图就是它的所有权区域。为了了解模拟世界中另一个部分正在发生的事件，空间应用程序创建了订阅。订阅区域是整个模拟世界区域的子集。订阅区域可以包括多个所有权区域的一部分，包括空间应用程序自己的所有权区域。 SimSpace Weaver 将订阅区域内发生的所有实体事件（例如，进入、退出、创建、更新和删除）通知空间应用程序。



空间应用程序的世界视图

空间应用程序的世界视图就是其所有权区域，即世界网格中的一个单元格。



增加了订阅区域的空间应用程序视图

空间应用程序使用订阅来了解模拟世界中另一个部分正在发生的事件。订阅区域可包含多个网格单元格和单元格部分。

例如，模拟实体进行物理交互的应用程序可能需要了解其所有权区域空间边界之外的实体。为此，应用程序可订阅与其所有权区域边界相接的区域。创建订阅后，应用程序会收到有关这些区域中实体事件的通知，并且可以读取实体。另一个例子是自动驾驶车辆，它需要看到前方 200 米内的所有实体，无论哪个应用程序拥有该区域。车辆应用程序可创建具有筛选器的订阅，该筛选器用于覆盖可视区域的轴对齐边界框 (AABB)。

您可以创建不负责管理模拟空间方面的模拟逻辑。自定义应用程序是在单个工作线程上运行的可执行程序。您可以控制自定义应用程序的生命周期（启动和停止）。模拟客户端可以连接到自定义应用程序来查看模拟或与之交互。您还可以创建在每个工作人员上运行的服务应用程序。 SimSpace Weaver 在运行您的模拟的每个工作人员上启动您的服务应用程序的实例。

自定义应用程序和服务应用程序创建订阅来了解实体事件和读取实体。这些应用程序没有所有权区域，因为它们不是空间应用程序。使用订阅是这些应用程序了解模拟世界中正在发生的事件的唯一途径。

你怎么用 SimSpace Weaver

使用时 SimSpace Weaver，您需要遵循以下主要步骤：

1. 编写和构建集成C++应用程序 SDK 的 SimSpace Weaver 应用程序。
   a. 您的应用程序调用 API 来与模拟状态进行交互。
2. 编写客户端，通过某些应用程序查看您的模拟并与其交互。
3. 在文本文件中配置模拟。
4. 将您的应用程序包和模拟配置上传到服务。
5. 启动模拟。
6. 根据需要启动和停止您的自定义应用程序。
7. 将客户端连接到您的自定义或服务应用程序，以便查看模拟或与其交互。
8. 在 Amazon 日志中查看您的模拟 CloudWatch 日志。
9. 停止模拟。
10清理模拟。

模拟架构

模拟架构（或架构）是一个YAML格式的文本文件，其中包含模拟的配置信息。 SimSpace Weaver 在启动模拟时使用您的架构。A SimSpace Weaver pp SDK 可分发包包含示例项目的架构。在创建自己的架构时，您可以将其作为起点。有关模拟架构的更多信息，请参阅 SimSpace Weaver 仿真架构参考。

工作线程和资源单位

工作线程是运行模拟的 Amazon EC2 实例。您可以在模拟架构中指定工作器类型。 SimSpace Weaver 将您的工作人员类型映射到该服务使用的特定 Amazon EC2 实例类型。 SimSpace Weaver 为您启动和停止工作人员，并管理工作人员之间的网络通信。 SimSpace Weaver 为每个模拟启动一组工作程序。不同的模拟使用不同的工作线程。

工作线程上的可用计算（处理器和内存）能力划分为称为计算资源单位（或简称资源单位）的逻辑单元。资源单位代表着固定数量的处理器和内存容量。

> **Note**
>
> 我们之前将计算资源单位称为槽。在我们的文档中，您可能仍然会看到“槽”这个术语。

模拟时钟

每个模拟都有自己的时钟。您可以使用 API 调用或 SimSpace Weaver 控制台来启动和停止时钟。只有在时钟运行时，模拟才会更新。模拟中的所有操作都发生在称为刻度的时间段内。时钟会向所有工作线程公告每个刻度的开始时间。

时钟率（或刻度率）是时钟公告的每秒刻度数量（赫兹或 Hz）。模拟所需的时钟率是模拟架构的一部分。针对刻度的所有操作都必须先完成，然后才能开始下一个刻度。因此，生效的时钟率可能低于所需的时钟率。生效的时钟率不会高于所需的时钟率。

分区

分区是工作线程上的一段共享内存。每个分区都包含部分模拟状态数据。

空间应用程序的分区（也称为空间应用程序分区或空间分区）包含空间应用程序所有权区域中的所有实体。 SimSpace Weaver 根据每个实体的空间位置将实体放在空间应用程序分区中。这意味着 SimSpace Weaver 会尝试将空间上彼此邻近的实体放在同一个工作线程上。这样，在模拟应用程序拥有的实体时，可以最大限度减少应用程序对其不拥有的实体所需的知识量。

State Fabric

State Fabric 是所有工作线程上的共享内存（所有分区的集合）的系统。它会保存模拟的所有状态数据。

State Fabric 使用自定义二进制格式，将实体描述为该实体的每个数据字段的一组初始数据和更新日志。凭借这种格式，您可以访问实体在模拟时间前一点的状态，并将其映射回现实世界时间中的某个点。缓冲区的大小是有限的，不可能超出缓冲区中的时间倒流。 SimSpace Weaver 在更新日志中为每个字段使用指向当前偏移量的指针，并在字段更新过程中更新指针。 SimSpace Weaver 使用共享内存将这些更新日志映射到应用程序的进程空间。

这种对象格式可降低开销和无序列化成本。 SimSpace Weaver 还使用此对象格式来解析和识别索引字段（例如实体位置）。

实体

实体是模拟中的最小数据构建基块。实体的示例包括角色（例如，人和车辆）和静态对象（例如，建筑物和障碍物）。实体具有可作为永久数据存储在 SimSpace Weaver中的属性（例如，位置和方向）。实体存在于分区中。

应用程序

SimSpace Weaver 应用程序是您编写的软件，其中包含运行每个仿真滴答的自定义逻辑。大多数应用程序的目的是在模拟运行时更新实体。您的应用程序调用 APIs SimSpace Weaver App SDK 对模拟中的实体执行操作（例如读取和更新）。

您可以将应用程序及其所需资源（例如库）打包为.zip 文件，然后将其上传到。 SimSpace Weaver应用程序在工作线程上的 Docker 容器中运行。 SimSpace Weaver 为每个应用程序分配固定数量的资源单位。

SimSpace Weaver 为每个应用程序分配一个（且只有一个）分区的所有权。应用程序及其分区位于同一个工作线程上。每个分区只有一个应用程序所有者。应用程序可以在其分区中创建、读取、更新和删除实体。应用程序拥有其分区中的所有实体。

共有三种类型的应用程序：空间应用程序、自定义应用程序和服务应用程序。它们因使用案例和生命周期而异。

> **Note**
>
> 在中 SimSpace Weaver，“应用程序” 一词可以指应用程序的代码或该代码的运行实例。

空间应用程序

空间应用程序会更新模拟中空间上存在的实体的状态。例如，您可以定义一个 Physics 应用程序，该应用程序负责根据实体的速度、形状和大小在每个刻度中移动和碰撞实体。在这种情况下， SimSpace Weaver 会并行运行 Physics 应用程序的多个实例来处理工作负载的大小。

SimSpace Weaver 管理空间应用程序的生命周期。您可以在模拟架构中指定空间应用程序分区的排布。当您启动模拟时， SimSpace Weaver 会为每个空间应用程序分区启动一个空间应用程序。当您停止模拟时， SimSpace Weaver 会关闭您的空间应用程序。

其他类型的应用程序可以创建实体，但只有空间应用程序可以更新实体。其他类型的应用程序必须将其创建的实体传输到空间域。 SimSpace Weaver 使用实体的空间位置将实体移动到空间应用程序的分区。这会将实体的所有权转移给空间应用程序。

自定义应用程序

您可以使用自定义应用程序来与模拟进行交互。自定义应用程序使用订阅读取实体数据。自定义应用程序可以创建实体。但是，应用程序必须将实体转移给空间应用程序，才能将该实体包含在模拟中并进行更新。您可以为自定义应用程序 SimSpace Weaver 分配网络终端节点。模拟客户端可连接到网络端点来与模拟进行交互。您可以在模拟架构中定义自定义应用程序，但由您负责启动和停止它们（使用 SimSpace Weaver API 调用）。在工作人员上启动自定义应用程序实例后， SimSpace Weaver 不会将该实例转移给其他工作人员。

服务应用程序

当您需要在每个工作线程上运行只读进程时，您可以使用服务应用程序。例如，如果您有一个大型模拟，并且需要一个查看客户端，该客户端可在模拟中移动，并且仅向用户显示可见实体，则可以使用服务应用程序。在这种情况下，单个自定义应用程序实例无法处理模拟中的所有实体。您可以将服务应用程序配置为在每个工作线程上启动。然后，其中的每一个服务应用程序都可以筛选分配给它的工作线程上的实体，并且仅将相关实体发送给与其连接的客户端。然后，当它在模拟空间中移动时，您的查看客户端可以连接到不同的服务应用程序。您可以在模拟架构中配置服务应用程序。 SimSpace Weaver 为您启动和停止服务应用程序。

应用程序摘要

下表总结了不同类型的 SimSpace Weaver 应用程序的特征。

	空间应用程序	自定义应用程序	服务应用程序
读取实体	支持	是	是

	空间应用程序	自定义应用程序	服务应用程序
更新实体	是	否	否
创建实体	是	是*	是*
生命周期	托管（SimSpace Weaver 控制它。）	非托管（由您控制。）	托管（SimSpace Weaver 控制它。）
启动方法	SimSpace Weaver 为架构中指定的每个空间分区启动一个应用程序实例。	您启动每个应用程序实例。	SimSpace Weaver 按照架构中的指定，在每个 worker 上启动一个或多个应用程序实例。
客户端可以连接	否	是	是

* 当自定义应用程序或服务应用程序创建实体时，该应用程序必须将实体的所有权转移给空间应用程序，以便空间应用程序更新实体的状态。

Domains

SimSpace Weaver 域是运行相同可执行应用程序代码并具有相同启动选项和命令的应用程序实例的集合。我们按域所包含的应用程序类型来对域进行分类：空间域、自定义域和服务域。您可以在域中配置应用程序。

订阅和复制

应用程序会创建对空间区域的订阅，从而了解区域中的实体事件（例如，进入、退出、创建、更新和删除）。应用程序会先处理订阅中的实体事件，然后读取它不拥有的分区中实体的数据。

分区可以与应用程序位于同一个工作线程上（这称为本地分区），但该分区可以归另一个应用程序拥有。分区也可以存在于其他工作线程上（这称为远程分区）。如果订阅的是远程分区，则工作线程通过称为复制的过程来创建远程分区的本地副本。然后，工作线程会读取本地副本（复制的远程分区）。如果工作线程上的另一个应用程序需要在同一个刻度上从该分区读取，则该工作线程会读取相同的本地副本。

的示例用例 SimSpace Weaver

SimSpace Weaver 可用于基于代理的模型和带有空间组件的离散时间步长模拟。

创建大型人群模拟

您可以使用 SimSpace Weaver 模拟现实环境中的人群。 SimSpace Weaver 使您可以将仿真扩展到数百万个具有自身行为的动态对象。

创建城市级环境

SimSpace Weaver 用于创建整个城市的数字化双胞胎。可通过创建模拟协助开展城市规划、设计交通路线以及制定环境灾害响应计划。您可以将自己的地理空间数据来源作为环境的构建基块。

创造沉浸式交互体验

创建多个用户可参与和互动的模拟体验。使用流行的开发工具（如 Unreal Engine 和 Unity）来构建三维 (3D) 虚拟世界。使用自己的内容和行为自定义 3D 体验。

AWS SimSpace Weaver 支持终止

经过仔细考虑，我们决定终止对的支持 AWS SimSpace Weaver，自2026年5月20日起生效。 AWS SimSpace Weaver 从 2025 年 5 月 20 日起，将不再接受新客户。作为拥有账户在 2025 年 5 月 20 日之前注册该服务的现有客户，您可以继续使用这些 AWS SimSpace Weaver 功能。2026 年 5 月 20 日之后，您将无法再使用 AWS SimSpace Weaver。

有关过渡到 AWS Batch 以帮助运行容器化仿真的更多信息，请参阅此博客文章。

正在设置 SimSpace Weaver

要设置为首次使用 SimSpace Weaver ，必须设置您 AWS 账户 和您的本地环境。完成这些任务后，您将准备好学习入门教程。

设置任务

1. 设置你的 AWS 账户 “使用” SimSpace Weaver.
2. 设置您的本地环境用于 SimSpace Weaver.

设置你的 AWS 账户 “使用” SimSpace Weaver

完成以下任务以设置 AWS 账户 要使用 SimSpace Weaver。

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。

报名参加 AWS 账户

1. 打开https://portal.aws.amazon.com/billing/注册。
2. 按照屏幕上的说明操作。

   在注册时，将接到电话或收到短信，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。作为最佳安全实践，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。您可以随时前往 https://aws.amazon.com/并选择“我的账户”，查看您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 管理控制台在下一页上，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 Signing in as the root user。
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 URL。

  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 Create a permission set。
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 Add groups。

添加要使用的权限 SimSpace Weaver

要提供访问权限，请为您的用户、组或角色添加权限：

- 中的用户和群组 AWS IAM Identity Center：

  创建权限集合。按照《AWS IAM Identity Center 用户指南》中创建权限集的说明进行操作。

- 通过身份提供者在 IAM 中托管的用户：

  创建适用于身份联合验证的角色。按照《IAM 用户指南》中针对第三方身份提供者创建角色（联合身份验证）的说明进行操作。

- IAM 用户：
  - 创建您的用户可以担任的角色。按照《IAM 用户指南》中为 IAM 用户创建角色的说明进行操作。
  - （不推荐使用）将策略直接附加到用户或将用户添加到用户组。按照《IAM 用户指南》中向用户添加权限（控制台）中的说明进行操作。

Example授予使用权限的 IAM 政策 SimSpace Weaver

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "CreateAndRunSimulations",
            "Effect": "Allow",
            "Action": [
                "simspaceweaver:*",
                "iam:GetRole",
                "iam:ListRoles",
                "iam:CreateRole",
                "iam:DeleteRole",
                "iam:UpdateRole",
                "iam:CreatePolicy",
                "iam:AttachRolePolicy",
                "iam:PutRolePolicy",
                "iam:GetRolePolicy",
                "iam:DeleteRolePolicy",
                "s3:PutObject",
                "s3:GetObject",
                "s3:ListAllMyBuckets",
                "s3:PutBucketPolicy",
```

```
                "s3:CreateBucket",
                "s3:ListBucket",
                "s3:PutEncryptionConfiguration",
                "s3:DeleteBucket",
                "cloudformation:CreateStack",
                "cloudformation:UpdateStack",
                "cloudformation:DescribeStacks"
            ],
            "Resource": "*"
        },
        {
            "Sid": "PassAppRoleToSimSpaceWeaver",
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "iam:PassedToService": "simspaceweaver.amazonaws.com"
                }
            }
        }
    ]
}
```

设置您的本地环境用于 SimSpace Weaver

SimSpace Weaver 在容器化环境中运行模拟 Amazon Linux 2 (AL2) 环境。您必须有一个 AL2 环境才能编译您的应用程序并将其与 SimSpace Weaver 应用程序 SDK 关联起来。标准的本地开发环境是一个 AL2 容器 Docker。 如果你选择不使用 Docker，我们提供了在其中运行 AL2 环境的备用说明 Windows Subsystem for Linux (WSL)。 您也可以使用自己的方法来创建本地 AL2 环境。有关在 AL2 本地运行的其他方法，请参阅 Amazon EC2 文档。

> **Important**
>
> Docker on Microsoft Windows 是标准开发环境。 为方便起见，我们建议使用其他方法来设置本地开发环境，但它们不是标准的，也不受支持。

主题

- 为以下内容设置 SimSpace Weaver 分发包 Amazon Linux 2 (AL2) 在 Docker

- 为以下内容设置 SimSpace Weaver 分发包 Amazon Linux 2 (AL2) 在 Windows Subsystem for Linux (WSL)

为以下内容设置 SimSpace Weaver 分发包 Amazon Linux 2 (AL2) 在 Docker

本节提供有关在 AL2 环境中设置本地 SimSpace Weaver 分发 zip 的说明 Docker。 有关使用 AL2 中的设置说明 Windows Subsystem for Linux (WSL)，请参阅 为以下内容设置 SimSpace Weaver 分发包 Amazon Linux 2 (AL2) 在 Windows Subsystem for Linux (WSL)。

要求

- 微软 Windows 10 或更高版本，或者兼容的 Linux 系统
- Microsoft Visual Studio 2019或稍后，使用 Desktop development with C++已安装工作负载
- CMake3
- Git
- Docker Desktop
- AWS CLI
- Python 3.9

要在中设置 SimSpace Weaver 分发 z AL2 ip Docker

1. 如果您尚未为配置 AWS 证书 AWS CLI，请按照以下说明进行操作：配置 AWS CLI。
2. 下载 SimSpace Weaver 应用程序 SDK 可分发包。其中包含以下内容：
   - 用于 SimSpace Weaver 应用程序开发的二进制文件和库
   - 自动执行部分开发工作流程的帮助程序脚本
   - 演示 SimSpace Weaver 概念的示例应用程序
3. 将文件解压缩到您选择的 *sdk-folder*。
4. 转至 *sdk-folder*。
5. 输入以下命令来安装所需的 Python 软件包：

   ```
   pip install -r PackagingTools/python_requirements.txt
   ```

6. 输入以下命令以使用 Docker 镜像设置 SimSpace Weaver 发行版。

```
python setup.py
```

此命令执行以下操作：

- 创建一个 AL2 docker 镜像，其中安装了构建 SimSpace Weaver 项目的所有要求。
- 创建启动模拟所需的 CloudFormation 资源。
  - 示例 CloudFormation 堆栈模板可以在中找到 *sdk-folder*/PackagingTools/sample-stack-template.yaml
- 使用本地系统的正确路径配置所提供的示例项目。

故障排除

- Docker 似乎被卡住了
  - 如果在调用 Docker 命令后控制台输出卡住了，请尝试重新启动 Docker 引擎。如果这不起作用，请重新启动计算机。

为以下内容设置 SimSpace Weaver 分发包 Amazon Linux 2 (AL2) 在 Windows Subsystem for Linux (WSL)

本节提供有关在 AL2 环境中设置 SimSpace Weaver 分发 zip 的说明 Windows Subsystem for Linux (WSL)。 有关在中设置 AL2 的说明 Docker，请参阅 为以下内容设置 SimSpace Weaver 分发包 Amazon Linux 2 (AL2) 在 Docker。

> **Important**
>
> 本节介绍一种解决方案，该解决方案使用的版本并非由 Amazon 拥有、开发或支持。 AL2 如果您选择不使用，则提供此解决方案只是为了方便起见 Docker。 如果您选择使用此解决方案，Amazon 不 AWS 承担任何责任。

要求

- Hyper-V on Windows 10
- Windows Subsystem for Linux (WSL)
- 的第三方开源 AL2 发行版 WSL （下载版本 2.0.20200722.0- update.2 ）（参见说明书 ）

> **Important**
>
> 我们的 WSL 说明使用发行版的 2.0.20200722.0-update.2 版本进行 AL2 WSL。 如果您使用任何其他版本，则可能会遇到错误。

要设置 SimSpace Weaver 配送 zip，请使用 in AL2 WSL

1. 在 Windows 命令提示符下，启动您的 AL2 环境 WSL.

```
wsl -d Amazon2
```

> **Important**
>
> 当你跑进去的时候 WSL，在运行位于的其中一个 `quick-start.py` Python 帮助脚本时包含该`--al2`选项`sdky-folder/Samples/sample-name/tools/cloud/quick-start.py`。

2. 在 Linux Shell 提示符下，更新您的 yum 程序包管理器。

```
yum update -y
```

> **Important**
>
> 如果此步骤超时，您可能需要切换到 WSL1 然后重试这些过程。退出你的 WSL AL2 会话并在 Windows 命令提示符下输入以下内容：
>
> ```
> wsl --set-version Amazon2 1
> ```

3. 安装解压缩的工具。

```
yum install -y unzip
```

4. 删除所有yum已安装 AWS CLI 的内容。如果您不确定是否yum安装了，请尝试以下两个命令。AWS CLI

```
yum remove awscli
```

```
yum remove aws-cli
```

5. 创建一个临时目录并转到该目录。

```
mkdir ~/temp
cd ~/temp
```

6. 下载并安装 AWS CLI：

```
curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv2.zip"
unzip awscliv2.zip
./aws/install
```

7. 您可以删除该临时目录。

```
cd ~
rm -rf temp
```

8. 重新启动 Shell 会话以更新环境中的路径。

```
exec
```

9. 在您的 AL2 环境 AWS CLI 中为配置您的 AWS 证书。有关更多信息，请参阅配置 AWS CLI。如果您使用 AWS IAM Identity Center，请参阅《AWS Command Line Interface 用户指南》 AWS IAM Identity Center中的配置 AWS CLI 以使用。

```
aws configure
```

10. 安装 Git。

```
yum install -y git
```

11. 安装 wget。

```
yum install -y wget
```

12. 为 SimSpace Weaver 应用程序 SDK 创建文件夹。

```
mkdir sdk-folder
```

13. 转到您的 SDK 文件夹。

```
cd sdk-folder
```

14. 下载 SimSpace Weaver 应用程序 SDK 可分发包。其中包含以下内容：

    - 用于 SimSpace Weaver 应用程序开发的二进制文件和库
    - 自动执行部分开发工作流程的帮助程序脚本
    - 演示 SimSpace Weaver 概念的示例应用程序

```
wget https://artifacts.simspaceweaver.us-east-2.amazonaws.com/latest/
SimSpaceWeaverAppSdkDistributable.zip
```

15. 解压缩该文件。

```
unzip *.zip
```

16. 运行 WSL 安装脚本。

```
source ./setup-wsl-distro.sh
```

17. 输入以下命令来安装所需的 Python 软件包：

```
pip install -r PackagingTools/python_requirements.txt
```

18. 运行 SimSpace Weaver 发行版 zip 安装脚本：

```
python setup.py --samples --cloudformation
```

    此命令执行以下操作：

    - 创建启动模拟所需的 CloudFormation 资源。
        - 示例 CloudFormation 堆栈模板可以在中找到 *sdk-folder*/PackagingTools/sample-stack-template.yaml
    - 使用本地系统的正确路径配置所提供的示例项目。

> **Note**
>
> 在 WSL 中，您只需要为自己的 AL2 环境执行一次此操作即可。

将许可软件用于 AWS SimSpace Weaver

AWS SimSpace Weaver 允许您使用自己选择的仿真引擎和内容来构建模拟。在使用时 SimSpace Weaver，您有责任获取、维护和遵守您在模拟中使用的任何软件或内容的许可条款。确认您的许可协议允许在虚拟托管环境中部署软件和内容。

入门 SimSpace Weaver

本节提供的教程可帮助您入门 SimSpace Weaver。这些教程将向您介绍使用 SimSpace Weaver构建仿真的一般工作流程。这些教程演示了如何在中 SimSpace Weaver创建、部署和运行模拟。我们建议您从快速入门教程开始，以便在几分钟内运行模拟。之后请阅读其他教程以了解更多信息。

这些教程使用了您在安装过程中下载的 SimSpace Weaver 应用程序 SDK .zip 文件中包含的示例应用程序 (`PathfindingSample`)。示例应用程序演示了所有 SimSpace Weaver 仿真共享的概念，包括空间分区、跨分区实体切换、应用程序和订阅。

在教程中，您将创建一个包含四个空间分区的模拟。`PathfindingSample` 空间应用程序的一个单独实例管理各单一分区。空间应用程序在自己的分区中创建实体。实体移动到模拟世界中的特定位置，在移动时避开障碍物。您可以使用单独的客户端应用程序（包含在 SimSpace Weaver 应用程序 SDK 中）来查看模拟。

主题

- 的快速入门教程 SimSpace Weaver
- 详细教程：了解构建示例应用程序的详细信息

的快速入门教程 SimSpace Weaver

本教程将指导您在几分钟内在 SimSpace Weaver 上完成构建和运行模拟的过程。我们建议您从本教程开始，然后再阅读详细的教程。

要求

开始之前，请确保您已完成 正在设置 SimSpace Weaver 中的步骤。

> **Note**
>
> 此处提供的脚本是为了方便起见，不是必需的。有关如何手动完成这些步骤，请参阅详细教程。

步骤 1：启用日志记录（可选）

启用日志记录

1. 导航至：

```
sdk-folder/Samples/PathfindingSample/tools
```

2. 在文本编辑器中打开架构文件：

```
pathfinding-single-worker-schema.yaml
```

3. 找到文件开头的 `simulation_properties:` 部分：

```
simulation_properties:
  default_entity_index_key_type: "Vector3<f32>"
```

4. 在 `simulation_properties:` 一行之后插入以下 2 行：

```
  log_destination_service: "logs"
  log_destination_resource_name: "MySimulationLogs"
```

5. 确认 `simulation_properties:` 部分与以下内容相同：

```
simulation_properties:
  log_destination_service: "logs"
  log_destination_resource_name: "MySimulationLogs"
  default_entity_index_key_type: "Vector3<f32>"
```

6. 保存文件并退出文本编辑器。

步骤 2：使用控制台客户端快速入门（选项 1）

导航至：

```
sdk-folder/Samples/PathfindingSample/tools/cloud
```

运行以下命令之一：

- 码头工人：`python quick-start.py --consoleclient`

- WSL：`python quick-start.py –-consoleclient --al2`

默认情况下，这将启动一个在单个工作器上使用单个分区的模拟。其他配置可以通过传递`--schema {file name}.yaml`来自`/Samples/PathfindingSample/tools/`文件夹的来启动。

> **Note**
>
> 详细教程：了解构建示例应用程序的详细信息有关此脚本功能的深入说明，请参阅。

第 2 步：使用虚幻引擎客户端快速入门（选项 2）

请参阅 启动虚幻引擎视图客户端。

停止并删除您的模拟

导航至：

```
sdk-folder/Samples/PathfindingSample/tools/cloud
```

查找您的模拟名称：

```
aws simspaceweaver list-simulations
```

停止并删除模拟

```
python stop-and-delete.py --simulation simulation-name
```

故障排除

- FileNotFoundError: cmake

```
subprocess.run('cmake')
...
 FileNotFoundError: The system cannot find the file specified
```

  - 解决方案：脚本找不到命令`cmake`。请确保您安装了推荐的最低 CMake 版本，并且可以使用 PATH 中的`cmake`命令调用该版本。使用命令`cmake -version`进行验证。

- ImportError: 导入 libweaver_app_sdk_python_v1 时加载失败：找不到指定的模块。
  - 解决方法：当未使用 Python 3.9 启动 Weaver Python SDK 时，就会发生此错误。请确保与 “python” 命令关联的 python 版本为 Python 3.9。你可以通过运行`python --version`命令进行检查。
- 启动 Docker Build 后，快速启动脚本似乎停滞不前。
  - 解决方案：有时 Docker 需要几分钟来预热。如果此问题持续时间超过 5 分钟，请重启 Docker 或您的系统。
- target_compile_features CXX 编译器 “GNU” 没有已知的功能：
  - 解决方案：清除 Docker 缓存，删除 weaverappbuilder Docker 镜像，删除项目构建工件，然后重新运行。`setup.py`这应该会重置你的 Docker 环境并解决错误。

详细教程：了解构建示例应用程序的详细信息

快速入门教程介绍了如何使用`quick-start.py`和构建、启动、停止和删除示例模拟`stop-and-delete.py`。本教程将详细介绍这些脚本的工作原理，以及这些脚本可以采用哪些其他参数来最大限度地提高自定义 Weaver 仿真的灵活性。

要求

开始之前，请确保您已完成 正在设置 SimSpace Weaver 中的步骤。

步骤 1：启用日志记录（可选）

启用日志记录

1. 导航至：

   ```
   sdk-folder/Samples/PathfindingSample/tools
   ```

2. 在文本编辑器中打开架构文件：

   ```
   pathfinding-single-worker-schema.yaml
   ```

3. 找到文件开头的 `simulation_properties:` 部分：

   ```
   simulation_properties:
     default_entity_index_key_type: "Vector3<f32>"
   ```

4. 在 simulation_properties: 一行之后插入以下 2 行：

```
  log_destination_service: "logs"
  log_destination_resource_name: "MySimulationLogs"
```

5. 确认 simulation_properties: 部分与以下内容相同：

```
simulation_properties:
  log_destination_service: "logs"
  log_destination_resource_name: "MySimulationLogs"
  default_entity_index_key_type: "Vector3<f32>"
```

6. 保存文件并退出文本编辑器。

第 2 步：开始模拟

如快速入门教程所示，启动示例仿真的最基本步骤是：

1. 导航至：

```
sdk-folder/Samples/PathfindingSample/tools/cloud
```

2. 运行以下命令之一：

   - 码头工人：python quick-start.py
   - WSL：python quick-start.py --al2

此脚本可自动执行常见的终端命令，所有这些命令均可使用手动执行。 AWS CLI这些步骤是：

1. 将 Weaver 架构上传到 S3。

   - SimSpace Weaver 使用架构来配置您的模拟。架构是一个 YAML 格式的纯文本文件。有关更多信息，请参阅 配置模拟。

2. 构建并上传自定义容器（可选）。

   - 如果您的架构定义了自定义容器，则快速启动脚本将构建 docker 镜像并将其上传到 Amazon ECR。有关更多信息，请参阅 自定义容器。有关此功能的示例，请参阅PythonBubblesSample架构。

3. 构建项目。

- `quick-start.py`调用中定义的`build_project`函数`build.py`。此步骤将因项目而异。对 CMake 于 PathfindingSample，使用。 CMake 和 Docker 命令可以在中找到。`build.py`

4. 将构建项目上传到 S3。

   - 您可以检查您的 S3 存储桶，确保所有上传均成功。有关使用 Amazon S3 的更多信息，请参阅《Amazon Simple Storage Service 用户指南》中的创建、配置和使用 Amazon S3 存储桶。
   - 示例应用程序 zip 和 S3 存储桶使用以下名称格式：
     - `weaver-sample-bucket-account-number-region`
     - 空间应用程序：`ProjectNameSpatial.zip`
     - 查看（自定义）应用程序：`ProjectNameView.zip`

5. 开始模拟。

   - 这是`aws simspaceweaver start-simulation` AWS CLI 通话的包装。有关更多信息，请参阅的AWS CLI 命令参考 SimSpace Weaver。
   - 该脚本将循环运行，直到模拟状态变成 `STARTED` 或 `FAILED`。模拟可能需要几分钟才能启动。

6. 获取仿真详情。

   - DescribeSimulation API 提供有关模拟的详细信息，包括其状态。模拟可处于以下几种状态之一：

     模拟生命周期状态

     1. **`STARTING`** – 调用 StartSimulation 后的初始状态
     2. **`STARTED`** – 所有空间应用程序均已启动且运行状况正常
     3. **`STOPPING`** – 调用 StopSimulation 后的初始状态
     4. **`STOPPED`** – 所有计算资源均已停止
     5. **`DELETING`** – 调用 DeleteSimulation 后的初始状态
     6. **`DELETED`** – 所有分配给模拟的资源均已被删除
     7. **`FAILED`**— 仿真处于临界状态 error/failure 并已停止
     8. **`SNAPSHOT_IN_PROGRESS`** – 正在拍摄快照

     获取模拟详细信息

     1. 调用 ListSimulations API。

```
aws simspaceweaver list-simulations
```

该脚本应显示有关您的每个模拟的详细信息，类似于以下内容：

```
{
    "Status": "STARTED",
    "CreationTime": 1664921418.09,
    "Name": "MyProjectSimulation_22-10-04_22_10_15",
    "Arn": "arn:aws:simspaceweaver:us-west-2:111122223333:simulation/
MyProjectSimulation_22-10-04_22_10_15",
    "TargetStatus": "STARTED"
}
```

2. 调用 DescribeSimulation 以获取模拟详细信息。将 *simulation-name* 替换为上一步输出中模拟的 Name。

```
aws simspaceweaver describe-simulation --simulation simulation-name
```

该脚本应显示有关指定模拟的更多详细信息，类似于以下内容：

```
{
    "Status": "STARTED",
    "CreationTime": 1664921418.09,
    "Name": "MyProjectSimulation_22-10-04_22_10_15",
    "Arn": "arn:aws:simspaceweaver:us-west-2:111122223333:simulation/
MyProjectSimulation_22-10-04_22_10_15",
    "TargetStatus": "STARTED"
}
```

7. 启动定制化 App。

- SimSpace Weaver 不管理定制化 App 的生命周期。您必须启动自定义应用程序。最佳实践是在启动模拟时钟之前启动自定义应用程序，不过，您可以在启动时钟之后启动自定义应用程序。

  您可以调用 StartApp API 来启动您的定制化应用程序。

```
aws simspaceweaver start-app --simulation simulation-name --name app-name --
domain domain-name
```

StartApp API 调用将使用您提供的名称创建和启动自定义应用程序的新实例。如果您提供已存在的应用程序的名称，则会收到一条错误消息。如果要重新启动特定应用程序（实例），则必须先停止该应用程序并将其删除。

> **Note**
>
> 在启动自定义应用程序之前，模拟的状态必须为 STARTED。

示例应用程序提供了可查看模拟的 ViewApp 自定义应用程序。该应用程序为您提供用于连接模拟客户端的静态 IP 地址和端口号（您将在本教程后面的步骤中执行此操作）。您可以将 domain 看作一类具有相同可执行代码和启动选项的应用程序。app name 标识应用程序的实例。有关 SimSpace Weaver 概念的更多信息，请参阅的关键概念 SimSpace Weaver。

启动自定义应用程序后，您可以使用 DescribeApp API 检查其状态。

```
aws simspaceweaver describe-app --simulation simulation-name --app app-name --
domain domain-name
```

启动本教程中的查看应用程序

1. 呼叫StartAppViewApp。

   ```
   aws simspaceweaver start-app --simulation simulation-name --name ViewApp --
   domain MyViewDomain
   ```

2. 调用 DescribeApp 以查看自定义应用程序的状态。

   ```
   aws simspaceweaver describe-app --simulation simulation-name --app ViewApp --
   domain MyViewDomain
   ```

在自定义应用程序（实例）的状态变成 STARTED 后，DescribeApp 的输出将包括该自定义应用程序（实例）的 IP 地址和端口号。在以下示例输出中，IP 地址是 Address 的值，端口号是 EndpointInfo 数据块中 Actual 的值。

```
{
```

```
    "Status": "STARTED",
    "Domain": "MyViewDomain",
    "TargetStatus": "STARTED",
    "Simulation": "MyProjectSimulation_22-10-04_22_10_15",
    "LaunchOverrides": {
        "LaunchCommands": []
    },
    "EndpointInfo": {
        "IngressPortMappings": [
            {
                "Declared": 7000,
                "Actual": 4321
            }
        ],
        "Address": "198.51.100.135"
    },
    "Name": "ViewApp"
}
```

> **Note**
>
> `Declared` 的值是应用程序代码应绑定的端口号。的值`Actual`是向客户端 SimSpace Weaver 公开的用于连接您的应用程序的端口号。 SimSpace Weaver 将`Declared`端口映射到端`Actual`口。

> **Note**
>
> 您可以使用中描述的过程获取定制化应用程序的 IP 地址和端口号来获取任何已启动的自定义应用程序的 IP 地址和端口号。

8. 启动时钟。

   - 首次创建模拟时，它有一个时钟，但时钟不会运行。如果时钟未运行，模拟将不会更新状态。启动时钟后，它会开始向应用程序发送时钟周期。每次勾选，您的空间应用程序都会遍历它们拥有的实体，并将结果提交给 SimSpace Weaver

> **Note**
>
> 启动时钟可能需要 30-60 秒。

调用 StartClock API。

```
aws simspaceweaver start-clock --simulation simulation-name
```

> **Note**
>
> StartClock API 使用您的 *simulation-name*，您可以使用 ListSimulations API 找到该名称：
>
> ```
> aws simspaceweaver list-simulations
> ```

快速启动参数

- -h、--help
  - 列出这些参数。
- --干净
  - 在构建之前，请删除构建目录的内容。
- --al2
  - 直接在本机计算机上构建，而不是 Docker。只有在 Amazon Linux 2 环境（例如 WSL）中运行时才使用此选项。
- --仅上传
  - 仅将架构和应用程序 zip 上传到 Amazon S3，不要开始模拟。
- --nobuild
  - 跳过重建项目。
- --没有容器
  - 跳过重建架构中列出的模拟容器。
- --控制台客户端

  - 自动构建并连接 config.py 中列出的控制台客户端。
- --架构架构
  - 此调用将使用什么架构。config.py 中默认为 “架构” 的值。
- --name NAME
  - 模拟将使用什么名字。默认为 config.py 中'项目名称'的日期-时间的值。

步骤 3：检查日志（可选）

SimSpace Weaver 将模拟管理消息和应用程序的控制台输出写入 Amazon CloudWatch Logs。有关使用日志的更多信息，请参阅 Amazon Logs 用户指南中的使用日志组和 CloudWatch 日志流。

您创建的每个模拟在 Log CloudWatch s 中都有自己的日志组。日志组的名称在模拟架构中指定。在以下架构片段中，`log_destination_service` 的值为 `logs`。这意味着 `log_destination_resource_name` 的值是日志组的名称。在本例中，日志组是 `MySimulationLogs`。

```
simulation_properties:
  log_destination_service: "logs"
  log_destination_resource_name: "MySimulationLogs"
  default_entity_index_key_type: "Vector3<f32>"
```

在启动模拟日志组后，您还可以使用 DescribeSimulation API 来查找该日志组的名称。

```
aws simspaceweaver describe-simulation --simulation simulation-name
```

以下示例显示了 DescribeSimulation 的输出中描述日志配置的部分。日志组的名称显示在 `LogGroupArn` 的末尾。

```
    "LoggingConfiguration": {
        "Destinations": [
            {
                "CloudWatchLogsLogGroup": {
                    "LogGroupArn": "arn:aws:logs:us-west-2:111122223333:log-
group:MySimulationLogs"
```

```
                }
            }
        ]
    },
```

每个模拟日志组都包含多个日志流：

- 管理日志流- SimSpace Weaver 服务生成的模拟管理消息。

  ```
  /sim/management
  ```

- 错误日志流- SimSpace Weaver 服务生成的错误消息。只有在出现错误时，此日志流才会存在。SimSpace Weaver 将您的应用程序写入的错误存储在它们自己的应用程序日志流中（请参阅以下日志流）。

  ```
  /sim/errors
  ```

- 空间应用程序日志流（每个工作线程上的每个空间应用程序 1 个）– 空间应用程序生成的控制台输出。每个空间应用程序都会将日志写入其日志流。*spatial-app-id* 是 *worker-id* 末尾斜杠后的全部字符。

  ```
  /domain/spatial-domain-name/app/worker-worker-id/spatial-app-id
  ```

- 自定义应用程序日志流（每个自定义应用程序实例 1 个）– 自定义应用程序生成的控制台输出。每个自定义应用程序实例都会将日志写入其日志流。

  ```
  /domain/custom-domain-name/app/custom-app-name/random-id
  ```

- 服务应用程序日志流（每个服务应用程序实例 1 个）– 服务应用程序生成的控制台输出。每个服务应用程序都会将日志写入其日志流。*service-app-id* 是 *service-app-name* 末尾斜杠后的全部字符。

  ```
  /domain/service-domain-name/app/service-app-name/service-app-id
  ```

> **Note**
>
> 示例应用程序没有服务应用程序。

第 4 步：查看您的模拟

SimSpace Weaver 应用程序 SDK 提供了不同的选项来查看示例应用程序。如果你在本地不支持虚幻引擎开发，你可以使用示例控制台客户端。虚幻引擎客户端的说明假设你使用的是Windows。

当实体事件发生时，控制台客户端会显示一个事件列表。客户端从 ViewApp 中获取实体事件信息。如果您的控制台客户端显示了事件列表，则确认了模拟中与 ViewApp 和活动的网络连接。

PathfindingSample 模拟在二维平面上创建静止和移动的实体。移动的实体围绕静止实体移动。Unreal Engine 客户端提供实体事件的可视化效果。

控制台客户端

quick-start.py如果包含该--consoleclient选项，则在启动示例时可以自动构建和连接控制台客户端。要在调用后quick-start.py构建和连接控制台客户端，请执行以下操作：

导航至：

```
sdk-folder/Clients/TCP/CppConsoleClient
```

运行脚本来构建和连接客户端：

```
python start_client.py --host ip-address --port port-number
```

该脚本将执行以下操作：

1. 使用构建控制台客户端 CMake。
2. 使用给定的 IP 地址和端口号启动构建的可执行文件。

```
.\WeaverNngConsoleClient.exe --url tcp://ip-address:port-number
```

虚幻引擎客户端

请参阅启动虚幻引擎视图客户端。

第 5 步：停止并删除您的模拟

导航至：

```
sdk-folder/Samples/PathfindingSample/tools/cloud
```

查找您的模拟名称：

```
aws simspaceweaver list-simulations
```

停止并删除模拟：

```
python stop-and-delete.py --simulation simulation-name
```

该脚本`stop-and-delete.py`将执行以下操作：

1. 调用 AWS CLI 命令停止模拟。
   - `aws simspaceweaver stop-simulation`
   - 有关更多信息，请参阅的AWS CLI 命令参考 SimSpace Weaver。
2. 调用 AWS CLI 命令删除模拟。
   - `aws simpaceweaver delete-simulation`
   - 有关更多信息，请参阅的AWS CLI 命令参考 SimSpace Weaver。

stop-and-delete 参数

- -h、--help
  - 列出这些参数。
- --仿真模拟
  - 将模拟的名称设置为 stop-and-delete
- --停下来
  - 只停止模拟。不会将其删除。
- --删除
  - 仅删除模拟。只有当模拟为`STOPPED`或时才会起作用`FAILED`。

问题排查

参见快速故障排除入门教程。

与... 合作 SimSpace Weaver

本章提供帮助您在 SimSpace Weaver中构建应用程序的信息和指导。

主题

配置模拟

模拟架构（或架构）是 YAML格式的文本文件，用于指定模拟的配置。您可以使用相同的架构启动多个模拟。架构文件位于模拟的项目文件夹中。您可以使用任何文本编辑器来编辑文件。 SimSpace Weaver 仅在启动模拟时读取您的架构。您对架构文件所做的任何编辑只会影响在编辑后启动的新模拟。

要配置模拟，请编辑您的仿真架构文件（使用适合您的操作系统的路径分隔符）：

```
project-folder\tools\project-name-schema.yaml
```

在创建新模拟时上传模拟架构。在构建模拟的过程中，项目的快速入门帮助程序脚本将上传架构：

```
project-folder\tools\windows\quick-start.py
```

有关运行快速入门脚本的更多信息，请参阅本指南一入门章详细教程中的。

模拟配置参数

模拟架构包含引导信息，包括：

- 模拟属性 – SDK 版本和计算配置（工作线程的类型和数量）
- 时钟 – 刻度率和容限
- 空间分区策略 – 空间拓扑（例如网格）、边界和置放组（工作线程上的空间分区分组）
- 域及其应用程序 – 应用程序存储桶、路径和启动命令

SimSpace Weaver 使用您的架构配置来配置和排列空间分区、启动应用程序以及以您指定的滴答率推进模拟。

> **Note**
>
> SimSpace Weaver 应用程序 SDK 中的创建项目脚本将根据示例应用程序自动为您生成仿真架构。

以下主题介绍模拟架构中的参数。有关模拟架构的完整说明，请参阅SimSpace Weaver 仿真架构参考。

主题

- SDK 版本
- 模拟属性
- 工作线程
- 时钟
- 分区策略

- 域

SDK 版本

该sdk_version字段指定架构 SimSpace Weaver 的格式化版本。有效值：1.17、1.16、1.15、1.14、1.13、1.12

> **Important**
>
> sdk_version 的值仅包括主版本号和第一个次要版本号。例如，值 1.12 指定所有版本 1.12.x，例如 1.12.0、1.12.1、和 1.12.2。

模拟属性

架构的 simulation_properties 部分为实体的索引字段（通常是空间位置）指定日志配置和数据类型。

```
simulation_properties:
  log_destination_service: "logs"
  log_destination_resource_name: "MySimulationLogs"
  default_entity_index_key_type: "Vector3<f32>"
```

log_destination_service 的值决定对 log_destination_resource_name 的值的解释。目前，唯一支持的值是 logs。这意味着，的值log_destination_resource_name是 Amazon CloudWatch 日志中日志组的名称

> **Note**
>
> 日志记录是可选的。如果您未配置日志目标属性，则您的模拟将不会生成日志。

default_entity_index_key_type 属性为必需属性。唯一有效值为 Vector3<f32>。

工作线程

该workers部分指定了您想要进行模拟的工作人员的类型和数量。 SimSpace Weaver 使用自己的工作线程类型，这些类型映射到 Amazon EC2 实例类型。

```
workers:
  MyComputeWorkers:
    type: "sim.c5.24xlarge"
    desired: 1
```

启用多工作线程模拟

您可以创建使用多个工作线程的模拟。默认情况下，模拟使用 1 个工作线程。在开始模拟之前，您必须修改模拟架构。

> **Note**
>
> 您无法更改已经开始的模拟。如果要为正在运行的模拟启用多工作线程，必须先停止并删除模拟。

要使用多个工作线程，请将计算实例的 `desired` 数量设置为大于 1 的值。每个工作线程支持的应用程序数量有上限。有关更多信息，请参阅SimSpace Weaver 端点和配额。 SimSpace Weaver 只有当工作线程上的应用程序数量超过此限制时，才会使用 1 个以上的工作线程。 SimSpace Weaver 可以将应用程序放置在任何可用的工作程序上。无法保证将应用程序放置在特定工作线程上。

以下架构片段演示了请求 2 个工作线程的模拟的配置。如果应用程序的数量超过 1 个工作线程支持的最大应用程序数量， SimSpace Weaver 会尝试分配第二个工作线程。

```
workers:
  MyComputeWorkers:
    type: "sim.c5.24xlarge"
    desired: 2
```

时钟

`clock` 部分指定模拟时钟的属性。目前，您只能配置刻度率（时钟每秒发送给应用程序的时钟周期数）。刻度率是最大速率。有效刻度率可能会更低，因为在下一个时钟周期开始之前，所有客户的操作（如实体更新）都必须完成。刻度率也称为时钟频率。

`tick_rate` 的有效值取决于架构中 `sdk_version` 指定的值。

刻度率的有效值

- 低于 "1.14" 的版本：
  - 10
  - 15
  - 30
- 版本 "1.14" 或更高版本：
  - "10"
  - "15"
  - "30"
  - "unlimited"

    有关更多信息，请参阅 无限制刻度率。

> **Important**
>
> - 对于低于 "1.14" 的 sdk_version，tick_rate 的值为整数，例如 30。
> - 对于 sdk_version 为 "1.14" 或更高版本的架构，tick_rate 的值是一个字符串，例如 "30"。该值必须包含在双引号内。
>
>   如果将版本 "1.12" 或 "1.13" 的架构转换为版本 "1.14" 或更高版本，则必须将 tick_rate 的值包含在双引号中。

无限制刻度率

您可以将 tick_rate 设置为 "unlimited"，以便让模拟的运行速度与代码的执行速度一样快。使用无限的报价率，在所有应用程序完成当前报价的提交后立即 SimSpace Weaver 发送下一个报价。

> **Important**
>
> 1.14.0 之前的 SimSpace Weaver 版本不支持无限滴答率。架构中 sdk_version 的最小值为 "1.14"。

SimSpace Weaver Local 中的无限制刻度率

SimSpace Weaver Local 实现 "unlimited" 时就像架构指定 10 kHz (10000) 的刻度率一样。其效果与在 AWS Cloud中指定无限制刻度率相同。您仍然可以在架构指定 tick_rate: "unlimited"。有关 SimSpace Weaver Local的更多信息，请参阅当地发展 SimSpace Weaver。

有关时钟的常见问题解答

问题 1：我能否将 STARTED 模拟更改为使用不同的刻度率？

对于在生命周期的任何阶段已存在于 AWS Cloud 中的模拟，您都无法更改刻度率。您也无法更改正在 SimSpace Weaver Local 中运行的模拟的刻度率。您可以在架构中设置 tick_rate，然后从该架构启动新的模拟。

问题 2：我能否在 1.14 之前的版本中以无限制的刻度率运行模拟？

不能，1.14.0 之前的版本不支持无限制的刻度率。

排除时钟错误

如果您的模拟无法启动，则可以在 DescribeSimulationAPI 的输出"StartError"中检查的值。架构中的无效的 tick_rate 值将产生以下错误。

> **Note**
>
> 为了提高可读性，此处显示的错误输出以多行显示。实际错误输出只有一行。

- 低于 "1.14" 的 sdk_version 和 tick_rate 的值是无效整数。有效值：10、15、30

```
"[{\"errorType\":\"SchemaFormatInvalid\",\"errorMessage\":
    \"$.clock.tick_rate: does not have a value in the enumeration [10, 15, 30]\"}]"
```

- 低于 "1.14" 的 sdk_version 和 tick_rate 的值是字符串。有效值：10、15、30

```
"[{\"errorType\":\"SchemaFormatInvalid\",\"errorMessage\":
    \"$.clock.tick_rate: does not have a value in the enumeration [10, 15, 30]\"},
    {\"errorType\":\"SchemaFormatInvalid\",
    \"errorMessage\":\"$.clock.tick_rate: string found, integer expected\"}]"
```

- sdk_version 是 "1.14" 或更高版本，tick_rate 的值是无效字符串。有效值："10"、"15"、"30"、"unlimited"

```
"[{\"errorType\":\"SchemaFormatInvalid\",\"errorMessage\":
```

```
    \"$.clock.tick_rate: does not have a value in the enumeration [10, 15, 30,
 unlimited]\"}]"
```

- sdk_version 是 "1.14" 或更高版本，tick_rate 的值是整数。有效值："10"、"15"、"30"、"unlimited"

```
"[{\"errorType\":\"SchemaFormatInvalid\",\"errorMessage\":
    \"$.clock.tick_rate: does not have a value in the enumeration [10, 15, 30,
 unlimited]\"},
    {\"errorType\":\"SchemaFormatInvalid\",
    \"errorMessage\":\"$.clock.tick_rate: integer found, string expected\"}]"
```

分区策略

partitioning_strategies 部分指定了空间应用程序分区的配置属性。您可以为分区策略（本部分的一组属性）提供一个名称，并在空间应用程序配置中使用该名称。

```
partitioning_strategies:
  MyGridPartitioning:
    topology: "Grid"
    aabb_bounds:
      x: [0, 1000]
      y: [0, 1000]
    grid_placement_groups:
      x: 1
      y: 1
```

topology 属性指定模拟使用的坐标系类型。值 Grid 指定一个二维 (2D) 网格。

对于Grid拓扑，仿真空间被建模为轴对齐的边界框 (AABB)。 您可以在属性中指定 AABB 的每个轴的坐标边界。aabb_bounds模拟中空间上存在的所有实体都必须在 AABB 中一个位置。

网格置放群组

置放群组是您 SimSpace Weaver 要放置在同一个工作器上的空间应用程序分区的集合。您可以在grid_placement_groups属性中指定置放群组（在网格中）的数量和排列方式。 SimSpace Weaver 将尝试在置放群组之间均匀分配分区。同一置放群组中具有分区的空间应用程序的所有权区域在空间上是相邻的。

我们建议让 x * y 等于所需的工作线程数量。如果不相等，SimSpace Weaver 将尝试在可用工作人员之间平衡您的安置组。

如果您未指定置放群组配置，SimSpace Weaver 将为您计算一个置放群组配置。

域

您可以为域的一组配置属性提供一个名称。域中应用程序的启动设置决定域的类型：

- **launch_apps_via_start_app_call** – 自定义域
- **launch_apps_by_partitioning_strategy** – 空间域
- **launch_apps_per_worker**（未包含在示例应用程序中）– 服务域

> **Important**
>
> SimSpace Weaver 每次仿真最多支持 5 个域。这包括所有空间、自定义和服务域。

```
domains:
  MyViewDomain:
    launch_apps_via_start_app_call: {}
    app_config:
      package: "s3://weaver-myproject-111122223333-us-west-2/MyViewApp.zip"
      launch_command: ["MyViewApp"]
      required_resource_units:
        compute: 1
      endpoint_config:
        ingress_ports:
          - 7000
  MySpatialDomain:
    launch_apps_by_partitioning_strategy:
      partitioning_strategy: "MyGridPartitioning"
      grid_partition:
        x: 2
        y: 2
    app_config:
      package: "s3://weaver-myproject-111122223333-us-west-2/MySpatialApp.zip"
      launch_command: ["MySpatialApp"]
      required_resource_units:
```

```
        compute: 1
```

> **Note**
>
> SimSpace Weaver 应用程序 SDK 版本 1.12.x 项目对应用程序.zip 文件和架构使用单独的存储桶：
>
> - weaver-*lowercase-project-name*--ap *account-number* p-zips-*region*
> - weaver-*lowercase-project-name-account-number*-schemas-*region*

主题

应用程序配置

您可以将应用程序 (app_config) 的配置指定为其域配置的一部分。所有类型的域都使用这些相同的应用程序配置属性。

```
app_config:
  package: "s3://weaver-myproject-111122223333-us-west-2/MyViewApp.zip"
  launch_command: ["MyViewApp"]
  required_resource_units:
    compute: 1
```

> **Note**
>
> SimSpace Weaver 应用程序 SDK 版本 1.12.x 项目对应用程序.zip 文件和架构使用单独的存储桶：
>
> - weaver-*lowercase-project-name*--ap *account-number* p-zips-*region*
> - weaver-*lowercase-project-name-account-number*-schemas-*region*

package 属性指定 S3 存储桶中 zip 文件的 S3 URI。该 zip 文件包含应用程序可执行文件（也称为二进制文件）及其所需的任何其他资源（例如库）。应用程序可执行文件的每个实例都在工作线程上的 Docker 容器中运行。

launch_command 属性指定可执行文件的名称以及用于运行该应用程序的任何命令行选项。launch_command 的值是一个数组。整个启动命令字符串的每个令牌都是数组中的一个元素。

示例

- 对于启动命令：MyTestApp --option1 value1
- 指定：launch_command: ["MyTestApp", "-option1", "value1"]

该required_resource_units属性指定 SimSpace Weaver 应分配给此应用程序的计算资源单位数。计算资源单位是工作程序上固定数量的处理容量 (vCPU) 和内存 (RAM)。您可以增加该值来提高应用程序在工作线程上运行时可用的计算能力。每个工作线程上的计算资源单位数量有限。有关更多信息，请参阅 SimSpace Weaver 端点和配额。

配置空间域

对于空间域，必须指定 partitioning_strategy。此属性的值是您为分区策略指定的名称，该分区策略在架构的另一个部分中定义。

```
MySpatialDomain:
  launch_apps_by_partitioning_strategy:
    partitioning_strategy: "MyGridPartitioning"
    grid_partition:
      x: 2
      y: 2
  app_config:
    package: "s3://weaver-myproject-111122223333-us-west-2/MySpatialApp.zip"
    launch_command: ["MySpatialApp"]
    required_resource_units:
      compute: 1
```

> **Note**
>
> SimSpace Weaver 应用程序 SDK 版本 1.12.x 项目对应用程序.zip 文件和架构使用单独的存储桶：

- weaver-*lowercase-project-name*--ap *account-number* p-zips-*region*
- weaver-*lowercase-project-name*-*account-number*-schemas-*region*

使用Grid拓扑（此版本中唯一支持的拓扑）的分区策略指示 SimSpace Weaver 在网格中排列此域的空间应用程序分区。grid_partition 属性指定分区网格的行数和列数。

SimSpace Weaver 将为分区网格中的每个像元启动 1 个空间应用程序实例。例如，如果空间域具有grid_partition值x: 2和y: 2，则该空间域中有 2 * 2 = 4 个分区。 SimSpace Weaver 将启动在空间域中配置的 4 个应用程序实例，并为每个应用程序实例分配 1 个分区。

主题

空间域的资源需求

您可以为每个工作线程分配最多 17 个计算资源单位。您可以指定每个空间应用程序在空间域的 app_config 部分使用的计算资源单位的数量。

Example显示空间应用程序计算资源单位的示例架构片段

```
  MySpatialDomain:
    launch_apps_by_partitioning_strategy:
      partitioning_strategy: "MyGridPartitioning"
      grid_partition:
        x: 2
        y: 2
    app_config:
      package: "s3://weaver-myproject-111122223333-artifacts-us-west-2/
MySpatialApp.zip"
      launch_command: ["MySpatialApp"]
      required_resource_units:
        compute: 1
```

要计算域所需的计算资源单位数量，请将网格中的像元数量（在 grid_partition中，x * y）乘以分配给空间应用程序的计算资源单位数量。

在上述示例中，域 MySpatialDomain 指定：

- x: 2
- y: 2
- compute: 1

MySpatialDomain 的网格有 2 * 2 = 4 个像元。空间域需要 4 * 1 = 4 个计算资源单位。

架构中指定的所有域的计算资源单位总数必须小于或等于工作线程的数量 desired 乘以每个工作线程的最大计算资源单位数量 (17)。

多个空间域

您可以将模拟配置为使用多个的空间域。例如，您可以使用 1 个空间域来控制模拟中的主角色（例如，人和汽车），同时使用不同的空间域来控制环境。

您也可以使用多个空间域为模拟的不同部分分配不同的资源。例如，如果您的模拟中有一种类型的实体的实例数量比其他类型多 10 倍，则可以创建不同的域来处理各种实体类型，并为具有更多实体的域分配更多资源。

Important

SimSpace Weaver 1.14.0 之前的版本不支持多个空间域。

Important

AWS SimSpace Weaver Local目前不支持多个空间域。有关 SimSpace Weaver Local的更多信息，请参阅当地发展 SimSpace Weaver。

Important

SimSpace Weaver 每次仿真最多支持 5 个域。这包括所有空间、自定义和服务域。

配置多个空间域

要配置多个空间域，请将其他空间域定义作为单独的命名部分添加到架构中。每个域都必须指定 launch_apps_by_partitioning_strategy 密钥。请参阅以下示例架构。

```
sdk_version: "1.14"
workers:
  MyComputeWorkers:
    type: "sim.c5.24xlarge"
    desired: 1
clock:
  tick_rate: "30"
partitioning_strategies:
  MyGridPartitioning:
    topology: Grid
    aabb_bounds:
      x: [0, 1000]
      y: [0, 1000]
domains:
  MySpatialDomain:
    launch_apps_by_partitioning_strategy:
      partitioning_strategy: "MyGridPartitioning"
      grid_partition:
        x: 2
        y: 2
    app_config:
      package: "s3://weaver-myproject-111122223333-artifacts-us-west-2/
MySpatialApp.zip"
      launch_command: ["MySpatialApp"]
      required_resource_units:
        compute: 1
  MySecondSpatialDomain:
    launch_apps_by_partitioning_strategy:
      partitioning_strategy: "MyGridPartitioning"
      grid_partition:
        x: 2
        y: 2
    app_config:
      package: "s3://weaver-myproject-111122223333-artifacts-us-west-2/
MySpatialApp2.zip"
      launch_command: ["MySpatialApp2"]
      required_resource_units:
        compute: 1
```

将空间域放在一起

在某些情况下，您可能需要将空间域的分区放在另一个域的分区旁。如果这些分区相互创建跨域订阅，则可以改善性能特征。

将顶级密钥`placement_constraints`添加到架构中，以指定 SimSpace Weaver 应将哪些域放在一起。所需的 `on_workers` 键必须引用架构中名为 `workers` 的配置。

Example显示放置在一起的空间域的示例架构片段

```
workers:
  MyComputeWorkers:
    type: "sim.c5.24xlarge"
    desired: 2
placement_constraints:
  - placed_together: ["MySpatialDomain", "MySecondSpatialDomain"]
    on_workers: ["MyComputeWorkers"]
```

Important

- 如果您使用置放群组：
  - 确保 x * y 是工作线程数量的倍数。
  - 确保置放群组的值是放置在一起的域的网格维度的公因数。
- 如果您不使用置放群组：
  - 确保空间域网格的 1 个轴具有等于工作线数量的公因数。

有关置放群组的更多信息，请参阅分区策略。

有关空间域的常见问题解答

问题 1：如何向现有模拟添加另一个空间域？

- 对于正在运行的模拟 – 无法更改正在运行的模拟的配置。更改架构中的域配置，上传架构和应用程序 zip，然后启动新的模拟。
- 对于新的模拟 – 将域配置添加到架构，上传架构和应用程序 zip，然后启动新的模拟。

空间域问题排查

当您尝试启动模拟但域配置无效时，可能出现以下错误。

```
"StartError": "[{\"errorType\":\"SchemaFormatInvalid\",\"errorMessage\":
    \"We were unable to determine an arrangement of your domains that would fit
    within the provided set of workers. This can generally be resolved by
    increasing the number of workers if able, decreasing your domains\u0027
    [\u0027\u0027grid_partition\u0027\u0027] values, or adjusting the
    dimensions of your [\u0027\u0027grid_placement_groups\u0027\u0027].\"}]"
```

潜在原因

- 该架构为应用程序分配的计算资源单位数量多于工作线程上可用的计算资源单位数量。
- SimSpace Weaver 无法确定将域名放在工作人员身上的安排。当您指定多个空间域，但域网格之间没有公因数或倍数（例如 2x4 格网和 3x5 网格之间）时，就会发生这种情况。

网络端点

自定义应用程序和服务应用程序可具有外部客户端可连接的网络端点。您可以在 `endpoint_config` 中将端口号列表指定为 `ingress_ports` 的值。这些端口号都是 TCP 和 UDP 端口。自定义应用程序或服务应用程序应绑定到您在中指定的端口号`ingress_ports`。 SimSpace Weaver 在运行时动态分配端口号，并将这些端口映射到动态端口。您可以在应用程序开始查找动态（实际）端口号后调用 describe-app API。有关更多信息，请参阅快速入门教程中的获取定制化应用程序的 IP 地址和端口号。

```
domains:
  MyViewDomain:
    launch_apps_via_start_app_call: {}
    app_config:
      package: "s3://weaver-myproject-111122223333-us-west-2/MyViewApp.zip"
      launch_command: ["MyViewApp"]
      required_resource_units:
        compute: 1
      endpoint_config:
        ingress_ports:
          - 7000
```

> **Note**
>
> SimSpace Weaver 应用程序 SDK 版本 1.12.x 项目对应用程序.zip 文件和架构使用单独的存储桶：
>
> - weaver-*lowercase-project-name*--ap *account-number* p-zips-*region*
> - weaver-*lowercase-project-name*-*account-number*-schemas-*region*

> **Note**
>
> `endpoint_config` 是自定义应用程序和服务应用程序的可选属性。如果您未指定 `endpoint_config`，则应用程序将没有网络端点。

配置服务域

域配置`launch_apps_per_worker:`中存在表示它是一个包含服务应用程序的服务域。 SimSpace Weaver 为您启动和停止服务应用程序。 SimSpace Weaver 启动和停止应用程序时，该应用程序被视为具有托管生命周期。 SimSpace Weaver 目前支持在每个工作人员上启动 1 或 2 个服务应用程序。

Example配置为在每个工作线程上启动 1 个服务应用程序的域的示例

```
domains:
  MyServiceDomain:
    launch_apps_per_worker:
      count: 1
    app_config:
      package: "s3://weaver-myproject-111122223333-app-zips-us-west-2/
PlayerConnectionServiceApp.zip"
      launch_command: ["PlayerConnectionServiceApp"]
      required_resource_units:
        compute: 1
      endpoint_config:
        ingress_ports:
          - 9000
          - 9001
```

Example配置为在每个工作线程上启动 2 个服务应用程序的域的示例

```
domains:
  MyServiceDomain:
    launch_apps_per_worker:
      count: 2
    app_config:
      package: "s3://weaver-myproject-111122223333-app-zips-us-west-2/
PlayerConnectionServiceApp.zip"
      launch_command: ["PlayerConnectionServiceApp"]
      required_resource_units:
        compute: 1
      endpoint_config:
        ingress_ports:
          - 9000
          - 9001
```

模拟的最长持续时间

中的每个模拟都 AWS SimSpace Weaver 有一个最大持续时间设置，用于指定模拟可以运行的最大时间。启动模拟时，您可以将最长持续时间作为参数值提供。StartSimulation 应用程序编程接口 (API) 有一个可选参数 MaximumDuration。该参数的值是分钟数（m 或 M）、小时数（h 或 H）或天数（d 或 D）。例如，1h 或 1H 表示 1 小时。当达到此限制时， SimSpace Weaver 将停止模拟。

最大值

MaximumDuration 的最大有效值为 14D，或以小时 (336H) 或分钟 (20160M) 为单位的等效值。

默认值

MaximumDuration 参数是可选的。如果您不提供值，则 SimSpace Weaver 使用值14D。

最小值

MaximumDuration 的最小有效值是在数值上等效于 0 的值。例如，值 0M、0H 和 0D 在数值上都等效于 0。

如果您提供最长持续时间的最小值，则模拟在达到STOPPING状态后会立即过渡到该STARTED状态。

使用控制台启动模拟

当您在 SimSpace Weaver 控制台中启动模拟时，可以为最长持续时间提供一个值。选择启动模拟时，在模拟设置表单的最长持续时间字段中输入该值。

> **Important**
>
> 如果您没有为最长持续时间提供值，则 SimSpace Weaver 会使用默认值 (14D)。

模拟达到最长持续时间时的状态

当 SimSpace Weaver 自动停止达到最大持续时间的模拟时，模拟的状态为`STOPPING`（如果正在进行中）或`STOPPED`。在 SimSpace Weaver 控制台中，模拟的目标状态仍然是 `STARTED`，因为这是用户请求的最后一个状态。

开发应用程序

SimSpace Weaver 开发需要 Amazon Linux 2 (AL2) 用于构建应用程序的环境，因为您的模拟运行在上面 Amazon Linux 在 AWS Cloud。 如果你正在使用 Windows，您可以使用 SimSpace Weaver 应用程序 SDK 中的脚本来创建和启动 Docker 运行的容器 AL2 以及构建 SimSpace Weaver 应用程序所需的依赖关系。你也可以启动 AL2 环境使用 Windows Subsystem for Linux (WSL)，或者使用本机 AL2 系统。有关更多信息，请参阅 设置您的本地环境用于 SimSpace Weaver。

> **Note**
>
> 无论您如何配置本地开发环境，您的应用程序都可以在中运行 Docker 当您上传容器以在中运行时 AWS Cloud。您的应用程序对主机操作系统没有直接访问权限。

SimSpace Weaver 应用程序的一般流程

1. 创建应用程序。
2. 循环：
   a. 通过创建 `Transaction` 开始更新。
      - 如果模拟关闭，则退出循环。

b. 处理订阅和所有权实体事件。

c. 更新模拟。

d. 提交 Transaction 以结束更新。

3. 销毁应用程序。

空间应用程序

每个空间应用程序都有一个所有权区域，即模拟环境下的空间区域。位于空间应用程序所有权区域中的实体存储在应用程序的已分配分区中。对于已分配分区中的所有实体，该单一空间应用程序拥有完全所有权（读取和写入权限）。其他应用程序均无法对这些实体执行写入操作。空间应用程序可推进实体的状态。每个空间应用程序仅拥有 1 个分区。 SimSpace Weaver 使用实体的空间位置编制索引并将其分配给空间应用程序分区。

SimSpace Weaver 应用程序 SDK 提供了一个示例应用程序。您可以在以下文件夹中找到示例应用程序的空间应用程序的源代码（使用适用于您的操作系统的正确路径分隔符）：

```
sdk-folder\Samples\PathfindingSample\src\SpatialApp
```

自定义应用程序

您可以创建并使用自定义应用程序来与模拟进行交互。

自定义应用程序可以

- 创建实体
- 订阅其他分区
- 提交更改

自定义应用程序的一般流程

1. 创建应用程序。
2. 订阅模拟中的特定区域：

    a. 创建 Transaction 以开始首次更新。

    b. 为特定区域创建订阅。

    c. 提交 Transaction 以结束更新。

3. 循环：
    a. 创建 `Transaction` 以开始更新。
        - 如果模拟关闭，则退出循环。
    b. 处理状态更改。
    c. 提交 `Transaction` 以结束更新。
4. 销毁应用程序。

定制化 App 创建实体后，必须将实体转移到空间域，实体才能在模拟中以空间形式存在。 SimSpace Weaver 使用实体的空间位置将实体放置在相应的空间应用程序分区中。将实体转移到空间域中后，创建实体的自定义应用程序无法更新或删除实体。

SimSpace Weaver 应用程序 SDK 提供了一个示例应用程序。您可以将示例应用程序中包含的自定义应用程序用作自己的自定义应用程序的模型。您可以在以下文件夹中找到示例应用程序的 view 应用程序（自定义应用程序）的源代码（使用适用于您的操作系统的正确路径分隔符）：

```
sdk-folder\Samples\PathfindingSample\src\ViewApp
```

开发客户端应用程序

您可能想要将客户端连接到模拟的一些原因包括：

- 将实时交通信息注入城市级模拟中。
- 创建human-in-the-loop模拟，其中人工操作员控制仿真的某些方面。
- 使用户可以与模拟进行交互（例如，训练模拟）。

这些示例中的自定义应用程序充当模拟状态与外部环境之间的接口。客户端连接到自定义应用程序，以便与模拟进行交互。

SimSpace Weaver 不处理客户端应用程序及其与您的自定义应用程序的通信。您负责客户端应用程序的设计、创建、操作和安全，以及它们与您的自定义应用程序之间的通信。 SimSpace Weaver 仅公开每个自定义应用程序的 IP 地址和端口号，以便客户端可以连接到它们。

SimSpace Weaver 应用程序 SDK 为其示例应用程序提供客户端。您可以将这些客户端用作自己的客户端应用程序的模型。您可以在以下文件夹中找到示例应用程序客户端的源代码：

Docker

```
sdk-folder\packaging-tools\clients\PathfindingSampleClients
```

WSL

> **Important**
>
> 为方便起见，我们提供了这些说明。它们可用于 Windows Subsystem for Linux (WSL)，并且不受支持。有关更多信息，请参阅 设置您的本地环境用于 SimSpace Weaver。

```
sdk-folder/packaging-tools/clients/PathfindingSampleClients
```

有关构建和使用示例应用程序客户端的更多信息，请参阅中的教程入门 SimSpace Weaver。

获取定制化应用程序的 IP 地址和端口号

要查看您的模拟，您需要创建一个定制化 App 并通过客户端连接到该应用程序。有关更多信息，请参阅中的教程入门 SimSpace Weaver。您可以使用以下步骤获取定制化应用程序的 IP 地址和端口号。使用适合您的操作系统的路径分隔符（例如，\在 Windows 和 Linux / 中）。

获取 IP 地址和端口号

1. 使用 ListSimulationsAPI 获取模拟的名称。

   ```
   aws simspaceweaver list-simulations
   ```

   输出示例：

   ```
   {
       "Simulations": [
           {
               "Status": "STARTED",
               "CreationTime": 1664921418.09,
               "Name": "MyProjectSimulation_22-10-04_22_10_15",
   ```

```
            "Arn": "arn:aws:simspaceweaver:us-west-2: 111122223333:simulation/
MyProjectSimulation_22-10-04_22_10_15",
            "TargetStatus": "STARTED"
        }
    ]

}
```

2. 使用 DescribeSimulationAPI 获取模拟中的域名列表。

```
aws simspaceweaver describe-simulation --simulation simulation-name
```

在输出的 LiveSimulationState 部分中查找 Domains 部分。

输出示例：

```
    "LiveSimulationState": {
        "Domains": [
            {
                "Type": "",
                "Name": "MySpatialSimulation",
                "Lifecycle": "Unknown"
            },
            {
                "Type": "",
                "Name": "MyViewDomain",
                "Lifecycle": "ByRequest"
            }
        ],
```

3. 使用 ListAppsAPI 获取网域中的自定义应用程序列表。例如，示例项目中视图（自定义）应用程序的域名为MyViewDomain。在输出中查找应用程序名称。

```
aws simspaceweaver list-apps --simulation simulation-name --domain domain-name
```

输出示例：

```
{
    "Apps": [
        {
            "Status": "STARTED",
            "Domain": "MyViewDomain",
            "TargetStatus": "STARTED",
            "Name": "ViewApp",
            "Simulation": "MyProjectSimulation_22-10-04_22_10_15"
        }
    ]
}
```

4. 使用 DescribeAppAPI 获取 IP 地址和端口号。对于示例项目，域名称为 MyViewDomain，应用程序名称为 ViewApp。

```
aws simspaceweaver describe-app --simulation simulation-name --domain domain-name
 --app app-name
```

IP 地址和端口号位于输出的 EndpointInfo 块中。IP 地址是 Address 的值，端口号是 Actual 的值。

输出示例：

```
{
    "Status": "STARTED",
    "Domain": "MyViewDomain",
    "TargetStatus": "STARTED",
    "Simulation": "MyProjectSimulation_22-10-04_22_10_15",
    "LaunchOverrides": {
        "LaunchCommands": []
    },
    "EndpointInfo": {
        "IngressPortMappings": [
            {
                "Declared": 7000,
                "Actual": 4321
            }
        ],
        "Address": "198.51.100.135"
```

```
    },
    "Name": "ViewApp"
}
```

> **Note**
>
> Declared 的值是应用程序代码应绑定的端口号。的值Actual是向客户端 SimSpace Weaver 公开的用于连接您的应用程序的端口号。 SimSpace Weaver 将Declared端口映射到端Actual口。

启动虚幻引擎视图客户端

导航至：

```
sdk-folder/Samples/PathfindingSample/tools/cloud
```

1. 运行以下命令之一：
   - 码头工人：python quick-start.py
   - WSL：python quick-start.py --al2
2. 获取 IP 地址和 “实际” 端口号。它们将出现在运行 quick-start.py 的控制台输出中，或者按照中的步骤获取它们获取定制化应用程序的 IP 地址和端口号。
3. 导航至：

   ```
   sdk-folder/Clients/TCP/UnrealClient/lib
   ```
4. 运行以下命令来构建 NNG 库：

   ```
   cmake -S . -B build
   cmake --build build --config RelWithDebInfo
   cmake --install build
   ```
5. 在文本编辑器中，打开 view_app_url.txt。
6. 在查看应用程序中更新 URL 的 IP 地址和端口号：tcp://ip-address:actual-port-number（它应该类似于 tcp://198.51.100.135:1234）。
7. 在虚幻编辑器中，选择播放。

故障排除

- NNG CMake 安装步骤失败，并显示 “可能需要管理权限”：

```
CMake Error at build/_deps/nng-build/src/cmake_install.cmake:39 (file):
  file cannot create directory: C:/Program Files
  (x86)/ThirdPartyNngBuild/lib.  Maybe need administrative privileges.
Call Stack (most recent call first):
  build/_deps/nng-build/cmake_install.cmake:37 (include)
  build/cmake_install.cmake:73 (include)
```

  - 解决方案：如果`nng.lib`或`nng.so`存在于 UnrealClient /lib 目录中，则可以放心地忽略此错误。如果没有，请尝试在具有管理员权限的终端中运行 cmake build 命令。
- “CMake 查找 nng 提供的软件包配置文件”：

```
CMake Error at CMakeLists.txt:23 (find_package):
By not providing "Findnng.cmake" in CMAKE_MODULE_PATH this project has
 asked CMake to find a package configuration file provided by "nng", but
 CMake did not find one.
```

  - 解决方法： CMake 在查找`Findnng.cmake`文件时遇到问题。使用构建时 CMake，请添加参数`-DTHIRD_PARTY_LIB_PATH sdk-folder/ThirdParty`。在重新运行 CMake 构建之前，请确保`Findnng.cmake`文件仍在`ThirdParty`目录中。

```
cmake -S . -B build -DTHIRD_PARTY_LIB_PATH sdk-folder/ThirdParty
cmake --build build --config RelWithDebInfo
cmake --install build
```

当地发展 SimSpace Weaver

您可以在本地部署 SimSpace Weaver 应用程序，以便进行快速测试和调试。

要求

- 完成正在设置 SimSpace Weaver中的步骤。

主题

- 步骤 1：启动本地模拟

步骤 1：启动本地模拟

1. 导航到

```
cd sdk-folder/Samples/sample-name/tools/local
```

2. 运行以下命令在本地构建和启动模拟。

```
python quick-start.py
```

该脚本将执行以下操作：

1. 构建 项目。
   - `quick-start.py`调用 build.py 中定义的`build_project`函数。此步骤将因项目而异。对 CMake 于 PathfindingSample，使用。 CMake 和 Docker 命令可以在 build.py 中找到。
2. 启动本地模拟
   - 该脚本将为架构中定义的每个空间分区启动一个本地进程。
   - 该脚本将为架构中定义的每个自定义应用程序启动一个进程。
   - 空间应用程序将首先启动，然后是自定义应用程序，每个应用程序都按照它们在架构中的显示顺序启动。

Important

在不支持 GUI 的环境（例如控制台 SSH 会话）中启动时，请使用`--noappwindow`选项将所有输出重定向到当前终端。

Important

对于 Linux 用户，该脚本假设您的系统具有该`xterm`命令。如果您的 Linux 发行版没有该`xterm`命令，请使用该`--noappwindow`选项将所有输出重定向到当前终端。

- -h、--help
  - 列出这些参数。
- --干净
  - 在构建之前，请删除构建目录的内容。
- --nobuild
  - 跳过重建项目。
- --noappwindow
  - 不要为每个应用程序打开一个新窗口。而是将 stdout 重定向到当前终端。
- --日志文件
  - 将控制台输出写入日志文件。
- --控制台客户端
  - 自动连接配置中列出的控制台客户端。
- --架构架构
  - 此调用将使用什么架构。config.py 中默认为 “架构”。

第 2 步：查看您的本地模拟

要查看您的本地模拟，您可以使用随附的任何客户端 SimSpaceWeaverAppSdkDistributable。有关构建和使用示例客户端的更多信息，请参阅中的教程入门 SimSpace Weaver。

您必须更新客户端的 IP 地址和端口号，才能连接到查看应用程序以进行本地模拟。请务必使用以下值和 SimSpace Weaver Local:

```
tcp://127.0.0.1:7000
```

根据选择的客户端，您可以按如下方式更新 IP 地址和端口号：

- Unreal – 更改 `view_app_url.txt` 第 1 行中的 URL
- 控制台 – 使用 IP 地址和端口号 URL 作为参数，启动客户端

步骤 3：停止本地模拟（在 Windows 上是可选的）

> **Note**
>
> 此步骤在 Linux 上是必需的，但在 Windows 上是可选的。

1. 导航至：

   ```
   sdk-folder/Samples/sample-name/tools/local
   ```

2. 运行以下命令以停止本地模拟并删除所有共享内存资源。

   ```
   python stop-and-delete.py
   ```

   该脚本将执行以下操作：

   - 停止本地进程。
   - 删除共享内存对象（仅在 Linux 上需要）。

stop-and-delete.py 参数

- -h、--help
  - 列出这些参数。
- --停下来
  - 仅尝试停止进程。
- --删除
  - 仅尝试删除共享内存资源。
- --进程
  - 要停止的进程名称。如果您的进程名称与架构中的软件包名称不匹配，请使用此选项。
- --架构架构
  - 此调用将使用什么架构。config.py 中默认为 “架构” 的值。

对本地开发进行故障排除 SimSpace Weaver

- Linux：找不到 xterm 命令/无法打开
  - 本地脚本假设在 Linux 上运行时存在 xterm 命令。如果您没有 xterm 命令或者正在不支持 GUI 的环境中运行，请在运行快速启动脚本时使用该`--noappwindow`选项。
- 没有打开任何应用程序窗口！
  - 当本地模拟立即崩溃时，就会发生这种情况。要在崩溃后查看控制台输出，请在运行快速启动脚本时使用`--noappwindow`或`--logfile`选项。
- 在视图应用程序启动或视图客户端连接后，模拟没有启动！
  - 使用该–`noappwindow`选项运行通常可以解决这类问题。否则，重启几次也会成功（尽管速度要低得多）。

AWS SimSpace Weaver 应用程序 SDK

A SimSpace Weaver pp SDK 提供了 APIs 可用于控制模拟中的实体和响应 SimSpace Weaver 事件的功能。其中包括以下命名空间：

- API – API 的核心定义及其用途

链接到以下库：

- `libweaver_app_sdk_cxx_v1_full.so`

> **Important**
>
> 当您在 AWS Cloud中运行应用程序时，该库可用于动态链接。您无需将其与应用程序一起上传。

> **Note**
>
> SimSpace Weaver 应用程序 SDK APIs 控制仿真中的数据。 APIs它们与 SimSpace Weaver 服务是分开的 APIs，后者控制您的 SimSpace Weaver 服务资源（例如模拟、应用程序和时钟）。 AWS有关更多信息，请参阅 SimSpace Weaver API 参考资料。

主题

API 方法返回一个 Result

大多数 SimSpace Weaver API 函数都有返回类型`Aws::WeaverRuntime::Result<T>`。如果函数执行成功，则 `Result` 会包含 `T`。否则，将`ResultAws::WeaverRuntime::ErrorCode`包含表示错误代码的 Rust App SDK.

Example示例

```
Result<Transaction> BeginUpdate(Application& app)
```

此方法：

- 如果 `BeginUpdate()` 执行成功，则返回 `Transaction`。
- 如果 `BeginUpdate()` 失败，则返回 `Aws::WeaverRuntime::ErrorCode`。

在顶层与应用程序 SDK 交互

生命周期

- SimSpace Weaver 应用程序 SDK 管理应用程序的生命周期。您无需读取或写入应用程序的生命周期状态。

分区

- 使用 `Result <PartitionSet> AssignedPartitions(Transaction& txn);` 可获取拥有的分区。
- 使用 `Result <PartitionSet> AllPartitions(Transaction& txn);` 可获取模拟中的所有分区。

模拟管理

本节介绍适用于常见模拟管理任务的解决方案。

主题

- 启动模拟。
- 更新模拟
- 终止模拟

启动模拟。

使用 `CreateApplication()` 可创建应用程序。

Example示例

```
Result<Application> applicationResult = Api::CreateApplication();

if (!applicationResult)
{
    ErrorCode errorCode = WEAVERRUNTIME_EXPECT_ERROR(applicationResult);

    std::cout << "Failed to create application. Error code " <<
        static_cast<std::underlying_type_t<ErrorCode>>(errorCode) <<
        " Last error message "<< Api::LastErrorMessage() << ".";

    return 1;
}

/**
* Run simulation
*/
RunSimulation(std::move(applicationResult.assume_value()));
```

更新模拟

使用以下 BeginUpdate 函数可更新应用程序：

- Result<Transaction> BeginUpdate(Application& app)
- Result<bool> BeginUpdateWillBlock(Application& app) – 告诉您会不会阻止 BeginUpdate()。

使用 Result<void> Commit(Transaction& txn) 可提交更改：

Example示例

```
Result<void> AppDriver::RunSimulation(Api::Application app) noexcept
{
    while (true)
    {
        {
            bool willBlock;

            do
            {
                WEAVERRUNTIME_TRY(willBlock, Api::BeginUpdateWillBlock(m_app));
            } while (willBlock);
        }

        WEAVERRUNTIME_TRY(Transaction transaction, Api::BeginUpdate(app));

        /**
         * Simulate app.
         */
        WEAVERRUNTIME_TRY(Simulate(transaction));
        WEAVERRUNTIME_TRY(Api::Commit(std::move(transaction)));
    }

    return Success();
}
```

终止模拟

使用 Result<void> DestroyApplication(Application&& app) 可终止应用程序和模拟。

从对 BeginUpdateWillBlock() 或 BeginUpdate() 的调用收到 ErrorCode::ShuttingDown 时，其他应用程序发现模拟正在关闭。当应用程序收到时 ErrorCode::ShuttingDown 时，可以调用 Result<void> DestroyApplication(Application&& app) 来自行终止。

Example示例

```
Result<void> AppDriver::EncounteredAppError(Application&& application) noexcept
{
    const ErrorCode errorCode = WEAVERRUNTIME_EXPECT_ERROR(runAppResult);

    switch (errorCode)
    {
    case ErrorCode::ShuttingDown:
        {
            // insert custom shutdown process here.

            WEAVERRUNTIME_TRY(Api::DestroyApplication(std::move(application)));
            return Success();
        }
    default:
        {
            OnAppError(errorCode);
            return errorCode;
        }
    }
}
```

Important

只能在 Result<void> DestroyApplication(Application&& app) 之后调用 Api::Commit()。在更新过程中销毁应用程序可能会导致未定义的行为。

Important

您必须在程序退出之前调用 DestroyApplication()，以确保应用程序报告为成功终止。程序退出时未能调用 DestroyApplication() 将导致状态报告为 FATAL。

订阅

您可以创建具有订阅区域和域 ID 的订阅。域 ID 表示拥有该订阅区域的域。BoundingBox2F32 描述订阅区域。使用以下函数可创建订阅：

```
Result<SubscriptionHandle> CreateSubscriptionBoundingBox2F32(Transaction& txn, DomainId
 id, const BoundingBox2F32& boundingBox)
```

Example示例

```
Result<void> CreateSubscriptionInSpatialDomain(Transaction& transaction)
{
    WEAVERRUNTIME_TRY(Api::PartitionSet partitionSet, Api::AllPartitions(transaction));


    Api::DomainId spatialDomainId;

    for (const Api::Partition& partition : partitionSet.partitions)
    {
        if (partition.domain_type == Api::DomainType::Spatial)
        {
            /**
            * Get the spatial domain ID.
            */
            spatialDomainId = partition.domain_id;
            break;
        }
    }

    constexpr Api::BoundingBox2F32 subscriptionBounds {
        /* min */ { /* x */ 0, /* y */ 0 },
        /* max */ { /* x */ 1000, /* y */ 1000 } }

    WEAVERRUNTIME_TRY(
        Api::SubscriptionHandle subscriptionHandle,
        Api::CreateSubscriptionBoundingBox2F32(
        transaction,
        spatialDomainId,
        subscriptionBounds));

    return Success();
}
```

您可以使用 CreateSubscriptionBoundingBox2F32() 返回的 Api::SubscriptionHandle 来修改订阅。您可以将其作为参数传递给以下函数：

```
Result<void> ModifySubscriptionBoundingBox2F32(Transaction& txn, SubscriptionHandle
 handle, const BoundingBox2F32& boundingBox)
```

```
Result<void> DeleteSubscription(Transaction& txn, SubscriptionHandle handle)
```

实体

当实体进入应用程序Api:Entity的订阅区域时CreateEntity()，您可以使用从或从所有权变更事件Result<Api::Entity>返回的 and（有关更多信息，请参阅实体事件）。Store Load APIs 我们建议您跟踪您的Api::Entity对象，以便可以将它们与这些对象一起使用 APIs。

主题

- 创建实体
- 将实体转移到空间域
- 写入和读实体字段数据
- 存储实体的位置
- 存储实体的位置

创建实体

使用 CreateEntity() 可创建实体。您可以定义传递给此函数的 Api::TypeId 的含义。

```
Namespace
{
    constexpr Api::TypeId k_entityTypeId { /* value */ 512 };
}

Result<void> CreateEntity(Transaction& transaction)
{
    WEAVERRUNTIME_TRY(
        Api::Entity entity,
        Api::CreateEntity(
            transaction, Api::BuiltinTypeIdToTypeId(k_entityTypeId )));
}
```

> **Note**
>
> Api::BuiltinTypeId 的值 0-511 是保留值。你的实体 TypeID （ 在本例k_entityTypeId中 ）的值必须等于 512 或更高。

将实体转移到空间域

自定义应用程序或服务应用程序创建实体后，必须将实体转移到空间域中，实体才能在模拟中以空间形式存在。空间域中的实体可以由其他应用程序读取，并且可以由空间应用程序更新。使用 ModifyEntityDomain() API 可将实体转移到空间域中。

```
AWS_WEAVERRUNTIME_API Result<void> ModifyEntityDomain(Transaction& txn, const Entity&
 entity, DomainId domainId) noexcept;
```

如果 DomainId 与调用的应用程序分配的 Partition 不匹配，则 DomainId 必须为 DomainType::Spatial Domain。在 Commit(Transaction&&) 过程中，所有权会转移到新的 Domain。

参数

txn

当前 Transaction。

entity

更改 Entity 的目标 Domain。

domainId

Entity 的目标 Domain 的 DomainId。

如果已成功更改实体域，此 API 会返回 Success。

写入和读实体字段数据

所有实体数据字段都是 BLOB 类型。您最多可以向实体写入 1,024 字节数据。我们建议您尽量减小 BLOB，因为 BLOB 越大性能越低。当你写入 blob 时，你会传递 SimSpace Weaver 一个指向数据及其长度的指针。从 BLOB 中读取时， SimSpace Weaver 会向您提供一个指针和一个读取长度。在应

用程序调用 Commit() 之前，必须完成所有读取操作。当应用程序调用 Commit() 时，读取调用返回的指针会失效。

> **Important**
>
> - 不支持在 Commit() 之后从缓存的 BLOB 指针中读取，这可能会导致模拟失败。
> - 不支持写入读取调用返回的 BLOB 指针，这可能会导致模拟失败。

主题

存储实体的字段数据

以下示例演示如何存储（写入 State Fabric）应用程序拥有的实体的字段数据。这些示例使用以下函数：

```
AWS_WEAVERRUNTIME_API Result<void> StoreEntityField(
    Transaction& txn,
    const Entity& entity,
    TypeId keyTypeId,
    FieldIndex index,
    std::int8_t* src,
    std::size_t length) noexcept;
```

Api::TypeId keyTypeId 参数表示传入数据的数据类型。

Api::TypeId keyTypeId 参数应从 Api::BuiltinTypeId 接收相应的 Api::TypeId。如果没有适当的转换，则可以使用 Api::BuiltinTypeId::Dynamic。

对于复杂的数据类型，请使用 Api::BuiltInTypeId::Dynamic。

> **Note**
>
> FieldIndex index 的值必须大于 0。0 是为索引键保留的值（请参阅 StoreEntityIndexKey()）。

Example使用基元数据类型的示例

```
namespace
{
    constexpr Api::FieldIndex k_isTrueFieldId { /* value */ 1 };
}

Result<void> SetEntityFields(
    Api::Entity& entity,
    Transaction& transaction)
{
    bool value = true;

    auto* src = reinterpret_cast<std::int8_t*>(value);
    size_t length = sizeof(*value);

    WEAVERRUNTIME_TRY(Api::StoreEntityField(
        transaction,
        entity,
        Api::BuiltinTypeIdToTypeId(
            Aws::WeaverRuntime::Api::BuiltinTypeId::Bool),
        k_isTrueFieldId,
        src,
        length));
}
```

Example使用 a 的示例 struct 来保存数据

```
namespace
{
    constexpr Api::FieldIndex k_dataFieldId { /* value */ 1 };
}

struct Data
{
    bool boolData;
    float floatData;
};

Result<void> SetEntityFields(
    Api::Entity& entity,
    Transaction& transaction)
{
```

```
    Data data = { /* boolData */ false, /* floatData */ -25.93 };

    auto* src = reinterpret_cast<std::int8_t*>(data);
    size_t length = sizeof(*data);

    WEAVERRUNTIME_TRY(Api::StoreEntityField(
        transaction,
        entity,
        Api::BuiltinTypeIdToTypeId(
            Aws::WeaverRuntime::Api::BuiltinTypeId::Dynamic),
        k_dataFieldId,
        src,
        length));
}
```

加载实体的字段数据

以下示例演示如何加载（从 State Fabric 读取）实体的字段数据。这些示例使用以下函数：

```
Result<std::size_t> LoadEntityField(
    Transaction& txn,
    const Entity& entity,
    TypeId keyTypeId,
    FieldIndex index,
    std::int8_t** dest) noexcept;
```

`Api::TypeId keyTypeId` 参数应从 `Api::BuiltinTypeId` 接收相应的 `Api::TypeId`。如果没有适当的转换，则可以使用 `Api::BuiltinTypeId::Dynamic`。

> **Note**
>
> `FieldIndex` 索引的值必须大于 0。0 是为索引键保留的值（请参阅 `StoreEntityIndexKey()`）。

Example使用基元数据类型的示例

```
namespace
{
    constexpr Api::FieldIndex k_isTrueFieldId { /* value */ 1 };
}
```

```
Result<void> LoadEntityFields(
    Api::Entity& entity,
    Transaction& transaction)
{
    std::int8_t* dest = nullptr;

    WEAVERRUNTIME_TRY(Api::LoadEntityField(
        transaction,
        entity,
        Api::BuiltinTypeIdToTypeId(
            Aws::WeaverRuntime::Api::BuiltinTypeId::Bool),
        k_isTrueFieldId,
        &dest));

    bool isTrueValue = *reinterpret_cast<bool*>(dest);
}
```

Example使用 a 的示例 struct 来保存数据

```
namespace
{
    constexpr Api::FieldIndex k_dataFieldId { /* value */ 1 };
}

struct Data
{
    bool boolData;
    float floatData;
};

Result<void> LoadEntityFields(
    Api::Entity& entity,
    Transaction& transaction)
{
    std::int8_t* dest = nullptr;

    WEAVERRUNTIME_TRY(Api::LoadEntityField(
        transaction,
        entity,
        Api::BuiltinTypeIdToTypeId(
            Aws::WeaverRuntime::Api::BuiltinTypeId::Dynamic),
        k_dataFieldId,
```

```
        &dest));

    Data dataValue = *reinterpret_cast<Data*>(dest);
}
```

加载已移除实体的字段数据

对于已经从应用程序所有权和订阅区域中移除的实体，您无法加载（从 State Fabric 读取）实体字段数据。以下示例会导致错误，因为 `Api::ChangeListAction::Remove` 会导致它调用实体的 `Api::LoadIndexKey()`。第二个示例显示了直接在应用程序中存储和加载实体数据的正确方法。

Example错误代码示例

```
Result<void> ProcessSubscriptionChanges(Transaction& transaction)
{
    /* ... */

    WEAVERRUNTIME_TRY(Api::SubscriptionChangeList subscriptionChangeList,
        Api::AllSubscriptionEvents(transaction));

    for (const Api::SubscriptionEvent& event :
        subscriptionChangeList.changes)
    {
        switch (event.action)
        {
        case Api::ChangeListAction::Remove:
            {
                std::int8_t* dest = nullptr;

                /**
                 * Error!
                 * This calls LoadEntityIndexKey on an entity that
                 * has been removed from the subscription area.
                 */
                WEAVERRUNTIME_TRY(Api::LoadEntityIndexKey(
                    transaction,
                    event.entity,
                    Api::BuiltinTypeIdToTypeId(
                        Api::BuiltinTypeId::Vector3F32),
                    &dest));

                AZ::Vector3 position =
                    *reinterpret_cast<AZ::Vector3*>(dest);
```

```
                break;
            }
        }

    }

    /* ... */
}
```

Example在应用程序中存储和加载实体数据的正确方法示例

```
Result<void> ReadAndSaveSubscribedEntityPositions(Transaction& transaction)
{
    static std::unordered_map<Api::EntityId, AZ::Vector3>
        positionsBySubscribedEntity;

    WEAVERRUNTIME_TRY(Api::SubscriptionChangeList subscriptionChangeList,
        Api::AllSubscriptionEvents(transaction));

    for (const Api::SubscriptionEvent& event :
        subscriptionChangeList.changes)
    {
        switch (event.action)
        {
        case Api::ChangeListAction::Add:
            {
                std::int8_t* dest = nullptr;

                /**
                 * Add the position when the entity is added.
                 */
                WEAVERRUNTIME_TRY(Api::LoadEntityIndexKey(
                    transaction,
                    event.entity,
                    Api::BuiltinTypeIdToTypeId(
                        Api::BuiltinTypeId::Vector3F32),
                    &dest));

                AZ::Vector3 position =
                    *reinterpret_cast<AZ::Vector3*>(dest);
                positionsBySubscribedEntity.emplace(
                    event.entity.descriptor->id, position);
```

```
                break;
            }
        case Api::ChangeListAction::Update:
            {
                std::int8_t* dest = nullptr;

                /**
                 * Update the position when the entity is updated.
                 */
                WEAVERRUNTIME_TRY(Api::LoadEntityIndexKey(
                    transaction,
                    event.entity,
                    Api::BuiltinTypeIdToTypeId(
                        Api::BuiltinTypeId::Vector3F32),
                    &dest));

                AZ::Vector3 position =
                    *reinterpret_cast<AZ::Vector3*>(dest);
                positionsBySubscribedEntity[event.entity.descriptor->id] =
                    position;

                break;
            }
        case Api::ChangeListAction::Remove:
            {
                /**
                 * Load the position when the entity is removed.
                 */
                AZ::Vector3 position = positionsBySubscribedEntity[
                    event.entity.descriptor->id];

                /**
                 * Do something with position...
                 */
                break;
            }
        }
    }

    /* ... */
}
```

存储实体的位置

您可以使用整数数据结构存储（写入 State Fabric）实体的位置。这些示例使用以下函数：

```
Result<void> StoreEntityIndexKey(
    Transaction& txn,
    const Entity& entity,
    TypeId keyTypeId,
    std::int8_t* src,
    std::size_t length)
```

> **Note**
>
> 您必须将 `Api::BuiltinTypeId::Vector3F32` 提供给 `Api::StoreEntityIndexKey()`，如以下示例所示。

Example使用数组表示位置的示例

```
Result<void> SetEntityPositionByFloatArray(
    Api::Entity& entity,
    Transaction& transaction)
{
    std::array<float, 3> position = { /* x */ 25, /* y */ 21, /* z */ 0 };

    auto* src = reinterpret_cast<std::int8_t*>(position.data());
    std::size_t length = sizeof(position);

    WEAVERRUNTIME_TRY(Api::StoreEntityIndexKey(
        transaction,
        entity,
        Api::BuiltinTypeIdToTypeId(Api::BuiltinTypeId::Vector3F32),
        src,
        length));
}
```

Example使用 a 的示例 struct 来代表立场

```
struct Position
```

```
{
   float x;
   float y;
   float z;
};

Result<void> SetEntityPositionByStruct(
    Api::Entity& entity,
    Transaction& transaction)
{
    Position position = { /* x */ 25, /* y */ 21, /* z */ 0 };

    auto* src = reinterpret_cast<std::int8_t*>(&position);
    std::size_t length = sizeof(position);

    WEAVERRUNTIME_TRY(Api::StoreEntityIndexKey(
        transaction,
        entity,
        Api::BuiltinTypeIdToTypeId(Api::BuiltinTypeId::Vector3F32),
        src,
        length));
}
```

存储实体的位置

您可以使用整数数据结构加载（从 State Fabric 读取）实体的位置。这些示例使用以下函数：

> **Note**
>
> 您必须将 `Api::BuiltinTypeId::Vector3F32` 提供给 `Api::LoadEntityIndexKey()`，如以下示例所示。

Example使用数组表示位置的示例

```
Result<void> GetEntityPosition(Api::Entity& entity,
    Transaction& transaction)
{
    std::int8_t* dest = nullptr;

    WEAVERRUNTIME_TRY(Aws::WeaverRuntime::Api::LoadEntityIndexKey(
        transaction,
```

```
        entity,
        Api::BuiltinTypeIdToTypeId(
            Aws::WeaverRuntime::Api::BuiltinTypeId::Vector3F32),
        &dest));

    std::array<float, 3> position =
        *reinterpret_cast<std::array<float, 3>*>(dest);
}
```

Example使用 a 的示例 struct 来代表立场

```
struct Position
{struct
   float x;
   float y;
   float z;
};

Result<void> GetEntityPosition(Api::Entity& entity, Transaction& transaction)
{
    std::int8_t* dest = nullptr;

    WEAVERRUNTIME_TRY(Aws::WeaverRuntime::Api::LoadEntityIndexKey(
        transaction,
        entity,
        Api::BuiltinTypeIdToTypeId(
            Aws::WeaverRuntime::Api::BuiltinTypeId::Vector3F32),
        &dest));

    Position position = *reinterpret_cast<Position*>(dest);
}
```

实体事件

您可以使用 SimSpace Weaver 应用程序 SDK 中的以下函数来获取所有权和订阅事件：

- `Result<OwnershipChangeList> OwnershipChanges(Transaction& txn)`
- `Result<SubscriptionChangeList> AllSubscriptionEvents(Transaction& txn)`

如果您需要回调驱动的实体事件处理，则可以使用 SimSpace Weaver 演示框架。有关更多信息，请参阅以下网站头文件：

- *sdk-folder*/packaging-tools/samples/ext/DemoFramework/include/DemoFramework/EntityEventProcessor.h

您还可以创建自己的实体事件处理。

主题

遍历所拥有实体的事件

使用 OwnershipChanges() 可获取所拥有实体（应用程序所有权区域中的实体）的事件列表。该函数具有以下签名：

```
Result<OwnershipChangeList> OwnershipChanges(Transaction& txn)
```

然后使用循环遍历各个实体，如以下示例所示。

Example示例

```
WEAVERRUNTIME_TRY(Result<Api::OwnershipChangeList> ownershipChangesResult,
 Api::OwnershipChanges(transaction));

for (const Api::OwnershipChange& event : ownershipChangeList.changes)
{
    Api::Entity entity = event.entity;
    Api::ChangeListAction action = event.action;

    switch (action)
    {
    case Api::ChangeListAction::None:
        // insert code to handle the event
        break;
    case Api::ChangeListAction::Remove:
        // insert code to handle the event
        break;
    case Api::ChangeListAction::Add:
        // insert code to handle the event
```

```
        break;
    case Api::ChangeListAction::Update:
        // insert code to handle the event
        break;
    case Api::ChangeListAction::Reject:
        // insert code to handle the event
        break;
    }
}
```

事件类型

- None – 实体位于该区域内，未修改其位置和字段数据。
- Remove – 实体已从该区域内移除。
- Add – 实体已添加到该区域内。
- Update – 实体位于该区域内且已被修改。
- Reject – 应用程序未能将实体从该区域内移除。

> **Note**
>
> 如果发生 Reject 事件，应用程序将在下一个刻度再次尝试转移。

遍历所订阅实体的事件

使用 AllSubscriptionEvents() 可获取所订阅实体（应用程序订阅区域中的实体）的事件列表。该函数具有以下签名：

```
Result<SubscriptionChangeList> AllSubscriptionEvents(Transaction& txn)
```

然后使用循环遍历各个实体，如以下示例所示。

Example示例

```
WEAVERRUNTIME_TRY(Api::SubscriptionChangeList subscriptionChangeList,
 Api::AllSubscriptionEvents(transaction));

for (const Api::SubscriptionEvent& event : subscriptionChangeList.changes)
{
```

```
    Api::Entity entity = event.entity;
    Api::ChangeListAction action = event.action;

    switch (action)
    {
    case Api::ChangeListAction::None:
        // insert code to handle the event
        break;
    case Api::ChangeListAction::Remove:
        // insert code to handle the event
        break;
    case Api::ChangeListAction::Add:
        // insert code to handle the event
        break;
    case Api::ChangeListAction::Update:
        // insert code to handle the event
        break;
    case Api::ChangeListAction::Reject:
        // insert code to handle the event
        break;
    }
}
```

事件类型

- None – 实体位于该区域内，未修改其位置和字段数据。
- Remove – 实体已从该区域内移除。
- Add – 实体已添加到该区域内。
- Update – 实体位于该区域内且已被修改。
- Reject – 应用程序未能将实体从该区域内移除。

> **Note**
>
> 如果发生 Reject 事件，应用程序将在下一个刻度再次尝试转移。

遍历实体的所有权更改事件

要获取实体在所有权区域和订阅区域之间移动的事件，请比较实体所有权和订阅事件当前与之前的变化。

您可以通过阅读以下 API 来处理这些事件：

- `Api::SubscriptionChangeList`
- `Api::OwnershipEvents`

然后，您可以将更改与之前存储的数据进行比较。

以下示例说明了如何处理实体所有权更改事件。此示例假设，对于在已订阅实体和拥有的实体（任一方向）之间过渡的实体，所有权remove/add event occurs first followed by the subscription remove/add 事件将在下一个勾选中显示。

Example示例

```
Result<void> ProcessOwnershipEvents(Transaction& transaction)
{
    using EntityIdsByAction =
        std::unordered_map<Api::ChangeListAction,
        std::vector<Api::EntityId>>;
    using EntityIdSetByAction =
        std::unordered_map<Api::ChangeListAction,
        std::unordered_set<Api::EntityId>>;

    static EntityIdsByAction m_entityIdsByPreviousOwnershipAction;

    EntityIdSetByAction entityIdSetByAction;

    /**
     * Enumerate Api::SubscriptionChangeList items
     * and store Add and Remove events.
     */
    WEAVERRUNTIME_TRY(Api::SubscriptionChangeList subscriptionEvents,
        Api::AllSubscriptionEvents(transaction));

    for (const Api::SubscriptionEvent& event : subscriptionEvents.changes)
    {
        const Api::ChangeListAction action = event.action;

        switch (action)
        {
        case Api::ChangeListAction::Add:
        case Api::ChangeListAction::Remove:
```

```
            {
                entityIdSetByAction[action].insert(
                    event.entity.descriptor->id);
                break;
            }
        case Api::ChangeListAction::None:
        case Api::ChangeListAction::Update:
        case Api::ChangeListAction::Reject:
            {
                break;
            }
        }
    }

    EntityIdsByAction entityIdsByAction;

    /**
     * Enumerate Api::OwnershipChangeList items
     * and store Add and Remove events.
     */

    WEAVERRUNTIME_TRY(Api::OwnershipChangeList ownershipChangeList,
        Api::OwnershipChanges(transaction));

    for (const Api::OwnershipChange& event : ownershipChangeList.changes)
    {
        const Api::ChangeListAction action = event.action;

        switch (action)
        {
        case Api::ChangeListAction::Add:
        case Api::ChangeListAction::Remove:
            {
                entityIdsByAction[action].push_back(
                    event.entity.descriptor->id);
                break;
            }
        case Api::ChangeListAction::None:
        case Api::ChangeListAction::Update:
        case Api::ChangeListAction::Reject:
            {
                break;
            }
        }
```

```
    }

    std::vector<Api::EntityId> fromSubscribedToOwnedEntities;
    std::vector<Api::EntityId> fromOwnedToSubscribedEntities;

    /**
     * Enumerate the *previous* Api::OwnershipChangeList Remove items
     * and check if they are now in
     * the *current* Api::SubscriptionChangeList Add items.
     *
     * If true, then that means
     * OnEntityOwnershipChanged(bool isOwned = false)
     */
    for (const Api::EntityId& id : m_entityIdsByPreviousOwnershipAction[
        Api::ChangeListAction::Remove])
    {
        if (entityIdSetBySubscriptionAction[
            Api::ChangeListAction::Add].find(id) !=
                entityIdSetBySubscriptionAction[
                Api::ChangeListAction::Add].end())
        {
            fromOwnedToSubscribedEntities.push_back(id);
        }
    }


    /**
     * Enumerate the *previous* Api::OwnershipChangeList Add items
     * and check if they are now in
     * the *current* Api::SubscriptionChangeList Remove items.
     *
     * If true, then that means
     * OnEntityOwnershipChanged(bool isOwned = true)
     */
    for (const Api::EntityId& id : m_entityIdsByPreviousOwnershipAction[
        Api::ChangeListAction::Add])
    {
        if (entityIdSetBySubscriptionAction[
            Api::ChangeListAction::Remove].find(id) !=

                entityIdSetBySubscriptionAction[
                Api::ChangeListAction::Remove].end())
        {
```

```
            fromSubscribedToOwnedEntities.push_back(id);
        }
    }

    m_entityIdsByPreviousOwnershipAction = entityIdsByOwnershipAction;

    return Success();
}
```

Result 和错误处理

Aws::WeaverRuntime::Result<T> 类使用第三方 Outcome 库。您可以使用以下模式来检查 API 调用返回的 Result 和捕获错误。

```
void DoBeginUpdate(Application& app)
{
    Result<Transaction> transactionResult = Api::BeginUpdate(app);

    if (transactionResult)
    {
        Transaction transaction =
            std::move(transactionResult).assume_value();

        /**
         * Do things with transaction ...
         */
    }
    else
    {
        ErrorCode errorCode = WEAVERRUNTIME_EXPECT_ERROR(transactionResult);
        /**
         * Macro compiles to:
         * ErrorCode errorCode = transactionResult.assume_error();
         */
    }
}
```

Result 控制语句宏

在具有返回类型 Aws::WeaverRuntime::Result<T> 的函数中，您可以使用 WEAVERRUNTIME_TRY 宏来代替之前的代码模式。该宏将执行传递给它的函数。如果传递的函数失败，该宏将使用封闭函数返回错误。如果传递的函数执行成功，则会执行到下一行。以下示例显示对之

前 DoBeginUpdate() 函数的重写。此版本使用WEAVERRUNTIME_TRY宏而不是 if-else 控制结构。请注意，此函数的返回类型为 Aws::WeaverRuntime::Result<void>。

```
Aws::WeaverRuntime::Result<void> DoBeginUpdate(Application& app)
{
    /**
     * Execute Api::BeginUpdate()
     * and return from DoBeginUpdate() if BeginUpdate() fails.
     * The error is available as part of the Result.
     */
    WEAVERRUNTIME_TRY(Transaction transaction, Api::BeginUpdate(m_app));

    /**
     * Api::BeginUpdate executed successfully.
     *
     * Do things here.
     */

    return Aws::Success();
}
```

如果 BeginUpdate() 失败，该宏会在失败时提前返回 DoBeginUpdate()。
您可以使用 WEAVERRUNTIME_EXPECT_ERROR 宏从 BeginUpdate() 获取 Aws::WeaverRuntime::ErrorCode。以下示例显示了 Update() 函数在失败时如何调用 DoBeginUpdate() 和获取错误代码。

```
void Update(Application& app)
{
    Result<void> doBeginUpdateResult = DoBeginUpdate(app);

    if (doBeginUpdateResult)
    {
        /**
         * Successful.
         */
    }
    else
    {
        /**
         * Get the error from Api::BeginUpdate().
         */
        ErrorCode errorCode = WEAVERRUNTIME_EXPECT_ERROR(doBeginUpdateResult);
```

```
    }
}
```

通过将 Update() 的返回类型更改为 Aws::WeaverRuntime::Result<void>，您可以将错误代码从 BeginUpdate() 提供给调用 Update() 的函数。您可以重复此过程，继续将错误代码发送到调用堆栈以下。

泛型和域类型

SimSpace Weaver 应用程序 SDK 提供单精度数据类型Api::Vector2F32和Api::BoundingBox2F32，以及双精Api::Vector2F64度和。Api::BoundingBox2F64这些数据类型是被动数据结构，没有便捷的方法。请注意，API 仅使用 Api::Vector2F32 和 Api::BoundingBox2F32。您可以使用这些数据类型来创建和修改订阅。

演 SimSpace Weaver 示框架提供了最小版本的 AzCore 数学库，其中包含Vector3和Aabb。有关更多信息，请参阅以下路径下的头文件：

- *sdk-folder*/packaging-tools/samples/ext/DemoFramework/include/AzCore/Math

其他应用程序 SDK 操作

主题

- AllSubscriptionEvents 以及 OwnershipChanges 包含上次通话中的事件
- 处理后解除读取锁 SubscriptionChangeList
- 创建用于测试的独立应用程序实例

AllSubscriptionEvents 以及 OwnershipChanges 包含上次通话中的事件

对 Api::AllSubscriptionEvents() 和 Api::OwnershipChanges() 的调用的返回值包含上一个调用（不是上一个刻度）的事件。在以下示例中，secondSubscriptionEvents 和 secondOwnershipChangeList 为空，因为它们的函数会在第一个调用后立即调用。

如果您等待 10 个刻度，然后调用 Api::AllSubscriptionEvents() 和 Api::OwnershipChanges()，则它们的结果将既包含事件又包含最近 10 个刻度（不是最后一个刻度）内的变化。

Example示例

```
Result<void> ProcessOwnershipChanges(Transaction& transaction)
{
    WEAVERRUNTIME_TRY(
        Api::SubscriptionChangeList firstSubscriptionEvents,
        Api::AllSubscriptionEvents(transaction));
    WEAVERRUNTIME_TRY(
        Api::OwnershipChangeList firstOwnershipChangeList,
        Api::OwnershipChanges(transaction));

    WEAVERRUNTIME_TRY(
        Api::SubscriptionChangeList secondSubscriptionEvents,
        Api::AllSubscriptionEvents(transaction));
    WEAVERRUNTIME_TRY(
        Api::OwnershipChangeList secondOwnershipChangeList,
        Api::OwnershipChanges(transaction));

    /**
     * secondSubscriptionEvents and secondOwnershipChangeList are
     * both empty because there are no changes since the last call.
     */
}
```

> **Note**
>
> 函数 `AllSubscriptionEvents()` 已实施，但函数 `SubscriptionEvents()` 未实施。

处理后解除读取锁 SubscriptionChangeList

当您开始更新时，在其他分区中会有在前一个刻度提交的数据的共享内存段。这些共享内存段可能会被读取器锁定。在所有读取器都释放锁定之前，应用程序将无法完全提交。作为一项优化措施，应用程序应在处理 `Api::SubscriptionChangelist` 项目后调用 `Api::ReleaseReadLeases()` 来释放锁定。这样可以减少提交时的争用情况。在默认情况下，`Api::Commit()` 会释放读取租约，但最佳实践是在处理订阅更新后手动将其释放。

Example示例

```
Result<void> ProcessSubscriptionChanges(Transaction& transaction)
{
```

```
    WEAVERRUNTIME_TRY(ProcessSubscriptionChanges(transaction));

    /**
     * Done processing Api::SubscriptionChangeList items.
     * Release read locks.
     */

    WEAVERRUNTIME_EXPECT(Api::ReleaseReadLeases(transaction));

    ...
}
```

创建用于测试的独立应用程序实例

在实际模拟中运行代码之前，您可以使用 `Api::CreateStandaloneApplication()` 来创建独立应用程序，以便测试应用程序逻辑。

Example示例

```
int main(int argc, char* argv[])
{
    Api::StandaloneRuntimeConfig config = {
        /* run_for_seconds (the lifetime of the app) */ 3,
        /* tick_hertz (the app clock rate) */ 10 };

    Result<Application> applicationResult =
        Api::CreateStandaloneApplication(config);

    ...
}
```

AWS SimSpace Weaver 演示框架

AWS SimSpace Weaver 演示框架（演示框架）是一个可用于开发 SimSpace Weaver 应用程序的实用程序库。

该演示框架提供以下内容

- 供您使用和检查的代码示例及编程模式
- 抽象和实用程序函数，可简化简单应用程序的开发
- 测试 SimSpace Weaver 应用程序 SDK 实验功能的更简单方法

我们设计了具有低级访问权限的 SimSpace Weaver 应用程序 SDK， SimSpace Weaver APIs 以提供更高的性能。相比之下，我们设计的演示框架旨在提供更高级别的抽象和访问权限 APIs ，使其 SimSpace Weaver 更易于使用。与直接使用 SimSpace Weaver 应用程序 SDK 相比，易用性的代价是性能水平较低。可容忍较低性能的模拟（例如，不要求实时性能的模拟）可能适合使用演示框架。对于复杂的应用程序，我们建议您使用 SimSpace Weaver 应用程序 SDK 中的本机功能，因为演示框架不是完整工具包。

该演示框架包括以下内容

- 支持和演示以下功能的工作代码示例：
  - 应用程序流量管理
  - 回调驱动的实体事件处理
- 一组第三方实用程序库：
  - spdlog （一个日志库）
  - 最小版本的 AZCore （数学库）仅包含：
    - Vector3
    - Aabb
  - cxxopts （命令行选项解析器库）
- 特定于的实用函数 SimSpace Weaver

演示框架由一个库、源文件和 CMakeLists。 这些文件包含在 SimSpace Weaver 应用程序 SDK 可分发包中。

使用服务限额

本节介绍如何使用的服务配额 SimSpace Weaver。限额也称为限制。有关服务限额的列表，请参阅 SimSpace Weaver 端点和配额。本节 APIs中的来自应用程序集 APIs。应用程序 APIs与服务不同 APIs。该应用程序 APIs 是 SimSpace Weaver 应用程序 SDK 的一部分。你可以在本地系统上的 app SDK 文件夹 APIs 中找到该应用程序的文档：

```
sdk-folder\SimSpaceWeaverAppSdk-sdk-version\documentation\index.html
```

主题

- 获取应用程序的限制
- 获取应用程序使用的资源数量

- 重置指标
- 超出限制
- 内存不足
- 最佳实践

获取应用程序的限制

您可以使用 RuntimeLimits 应用程序 API 来查询应用程序的限制。

```
Result<Limit> RuntimeLimit(Application& app, LimitType type)
```

参数

Application& 应用程序

对应用程序的引用。

LimitType 类型

具有以下限制类型的枚举：

```
enum LimitType {
    Unset = 0,
    EntitiesPerPartition = 1,
    RemoteEntityTransfers = 2,
    LocalEntityTransfers = 3
};
```

以下示例查询实体计数限制。

```
WEAVERRUNTIME_TRY(auto entity_limit,
    Api::RuntimeLimit(m_app, Api::LimitType::EntitiesPerPartition))
Log::Info("Entity count limit", entity_limit.value);
```

获取应用程序使用的资源数量

您可以调用 RuntimeMetrics 应用程序 API 来获取应用程序使用的资源数量：

```
Result<std::reference_wrapper<const AppRuntimeMetrics>> RuntimeMetrics(Application&
 app) noexcept
```

参数

Application& 应用程序

对应用程序的引用。

API 会返回对包含指标的 struct 的引用。计数器指标包含一个运行总值，并且只会增加。量规指标包含一个可以增加或减少的值。每当有事件增加值时，应用程序运行时都会更新计数器。只有在您调用 API 时，运行时才会更新量规。 SimSpace Weaver 保证该引用在应用程序的生命周期内有效。重复调用 API 不会更改引用。

```
struct AppRuntimeMetrics {
    uint64_t total_committed_ticks_gauge,

    uint32_t active_entity_gauge,
    uint32_t ticks_since_reset_counter,

    uint32_t load_field_counter,
    uint32_t store_field_counter,

    uint32_t created_entity_counter,
    uint32_t deleted_entity_counter,

    uint32_t entered_entity_counter,
    uint32_t exited_entity_counter,

    uint32_t rejected_incoming_transfer_counter,
    uint32_t rejected_outgoing_transfer_counter
}
```

重置指标

ResetRuntimeMetrics 应用程序 API 会重置中的 `AppRuntimeMetrics` struct 中值。

```
Result<void> ResetRuntimeMetrics(Application& app) noexcept
```

以下示例演示如何在应用程序中调用 ResetRuntimeMetrics。

```
if (ticks_since_last_report > 100)
{
    auto metrics = WEAVERRUNTIME_EXPECT(Api::RuntimeMetrics(m_app));
    Log::Info(metrics);

    ticks_since_last_report = 0;

    WEAVERRUNTIME_EXPECT(Api::ResetRuntimeMetrics(m_app));
}
```

超出限制

超过限制的应用程序 API 调用将返回 `ErrorCode::CapacityExceeded`，但实体转移除外。作为提交和 BeginUpdate 应用程序 API 操作的一部分， SimSpace Weaver 会异步处理实体转移，因此，如果由于实体转移限制而导致转移失败，则不存在会返回错误的特定操作。要检测转移失败，可以将 `rejected_incoming_transfer_counter` 和 `rejected_outgoing_transfer_counter`（在 `AppRuntimeMetrics` struct 中）的当前值与其先前值进行比较。被拒绝的实体不会出现在分区中，但应用程序仍然可以模拟它们。

内存不足

SimSpace Weaver 使用垃圾收集器进程来清理和释放释放的内存。写入数据的速度可能超过垃圾收集器释放内存的速度。如果发生这种情况，写入操作可能会超过应用程序的预留内存限制。 SimSpace Weaver 将返回内部错误，其中包含一条具有 `OutOfMemory`（以及其他详细信息）的消息。有关更多信息，请参阅 跨时间分散写入。

最佳实践

以下最佳实践是设计应用程序以避免超出限制的一般指南。这些实践可能不适用于您的特定应用程序设计。

经常监视并减慢速度

您应该经常监控自己的指标，并减慢接近限制的操作。

避免超过订阅限制和转移限制

如果可能，设计模拟以减少远程订阅和实体转移的数量。您可以使用置放群组，以便在同一个工作线程上放置多个分区，从而减少在工作线程之间进行远程实体转移的需求。

跨时间分散写入

在一个时间周期中更新的数量和大小可能会对提交事务所需的时间和内存产生重大影响。内存需求过大可能会导致应用程序运行时内存不足。您可以跨时间分散写入次数，以降低每个时间周期内更新的平均总大小。这可以帮助提高性能并避免超出限制。我们建议您在每个时间周期内的平均写入数量不要超过 12 MB，或者每个实体的平均写入数量不要超过 1.5 KB。

调试模拟

您可以使用以下方法获取模拟的相关信息。

主题

- 使用 SimSpace Weaver Local 然后看看控制台的输出
- 在 Amazon 日志中查看你的 CloudWatch 日志
- 使用 描述 API 调用
- 连接客户端

使用 SimSpace Weaver Local 然后看看控制台的输出

我们建议您先在本地开发模拟，然后在 AWS Cloud中运行模拟。当你使用运行时，你可以直接查看控制台输出 SimSpace Weaver Local。 有关更多信息，请参阅当地发展 SimSpace Weaver。

在 Amazon 日志中查看你的 CloudWatch 日志

当您在控制台中运行模拟时，应用程序的输出 AWS Cloud 将发送到 Amazon Logs 中的 CloudWatch 日志流。您的模拟还会写入其他日志数据。如果您想让模拟写入日志数据，则必须在模拟架构中启用日志记录。有关更多信息，请参阅 SimSpace Weaver 在 Amazon CloudWatch 日志中登录。

> **Warning**
>
> 您的模拟可能生成大量日志数据。日志数据可能增长得非常快。您应该密切关注日志，并在不再需要运行模拟时停止模拟。日志可能会产生高昂的成本。

使用 描述 API 调用

您可以使用以下服务在中 APIs 获取有关您的模拟的信息。 AWS Cloud

- ListSimulations— 在中获取所有模拟的 AWS Cloud清单。

  Example示例

  ```
  aws simspaceweaver list-simulations
  ```

- DescribeSimulation – 获取有关模拟的详细信息。

  Example示例

  ```
  aws simspaceweaver describe-simulation --simulation MySimulation
  ```

- DescribeApp – 获取有关应用程序的详细信息。

  Example示例

  ```
  aws simspaceweaver describe-app --simulation MySimulation --domain MyCustomDomain --
  app MyCustomApp
  ```

有关更多信息 SimSpace Weaver APIs，请参阅SimSpace Weaver API 参考资料。

连接客户端

您可以将客户端连接到在模拟架构中使用 `endpoint_config` 定义的正在运行的自定义应用程序或服务应用程序。 SimSpace Weaver 应用程序 SDK 包含可用于查看示例应用程序的示例客户端。您可以查看这些示例客户端的源代码和示例应用程序，以便了解如何创建自己的客户端。有关如何构建和运行示例客户端的更多信息，请参阅中的教程入门 SimSpace Weaver。

您可以在以下文件夹中找到示例客户端的源代码：

- *`sdk-folder`*`\packaging-tools\clients\PathfindingSampleClients\`

调试本地模拟

您可以使用 Microsoft Visual Studio 调试 SimSpace Weaver Local 应用程序。有关如何使用 Visual Studio 进行调试的更多信息，请参阅 Microsoft Visual Studio documentation。。

调试本地模拟

1. 请确保您的 `schema.yaml` 位于工作目录中。

2. 在 Visual Studio 中，打开要调试的每个应用程序的上下文菜单（如 `PathfindingSampleLocalSpatial` 或 `PathfindingSampleLocalView`），然后在调试部分设置工作目录。
3. 打开要调试的应用程序的上下文菜单，然后选择设置为启动项目。
4. 选择 F5 以开始调试应用程序。

调试模拟的要求与正常运行模拟的要求相同。您必须启动在架构中指定的空间应用程序数量。例如，如果您的架构指定一个 2x2 网格，而您在调试模式下启动一个空间应用程序，则只有在您再启动 3 个空间应用程序后，模拟才会运行（无论是否在调试模式下）。

要调试自定义应用程序，您必须先启动空间应用程序，然后在调试程序中启动自定义应用程序。

请注意，您的模拟以锁定步进运行。当应用程序遇到断点时，所有其他应用程序都会暂停。从该断点继续运行后，其他应用程序也会继续运行。

自定义容器

AWS SimSpace Weaver 应用程序在容器化 Amazon Linux 2 (AL2) 环境中运行。在中 AWS Cloud，在根据亚马逊弹性容器注册表 (Amazon ECR) Container Registry 提供的`amazonlinux:2`映像构建的 Docker 容器中 SimSpace Weaver 运行您的模拟。您可以创建自定义 Docker 映像，将其存储在 Amazon ECR 中，然后使用该映像进行模拟，而不是使用我们提供的默认 Docker 映像。

您可以使用自定义容器来管理软件依赖项，并包含标准 Docker 映像中没有的其他软件组件。例如，您可以将应用程序使用的公开可用软件库添加到容器中，并且只将您的自定义代码包含在应用程序 zip 文件中。

> **Important**
>
> 我们仅支持托管在 Amazon ECR 存储库中的 AL2 Docker 镜像，无论是在亚马逊 ECR 公共库中还是在您的私有 Amazon ECR 注册表中。我们不支持在 Amazon ECR 外部托管的 Docker 映像。有关 Amazon ECR 的更多信息，请参阅 Amazon Elastic Container Registry 文档。

主题

- 有关自定义容器的常见问题解答
- 自定义容器故障排除

创建自定义容器

这些说明假设您知道如何使用 Docker 和 Amazon Elastic Container Registry (Amazon ECR)。有关 Amazon ECR 的更多信息，请参阅 Amazon ECR 用户指南。

先决条件

- 用于执行这些操作的 IAM 身份（用户或角色）具有使用 Amazon ECR 的正确权限
- 本地系统上已安装 Docker

创建自定义容器

1. 创建 `Dockerfile`。

   `Dockerfile`要运行的 AWS SimSpace Weaver 应用程序，请从 Amazon ECR 中的Amazon Linux 2图像开始。

   ```
   # parent image required to run AWS SimSpace Weaver apps
   FROM public.ecr.aws/amazonlinux/amazonlinux:2
   ```

2. 构建 `Dockerfile`。
3. 将容器映像上传到 Amazon ECR。
   - 使用 AWS 管理控制台。
   - 使用 AWS Command Line Interface。

   > **Note**
   >
   > 如果您在尝试将容器映像上传到 Amazon ECR 时出现 `AccessDeniedException` 错误，则说明您的 IAM 身份（用户或角色）可能没有使用 Amazon ECR 所需的权限。您可以将`AmazonEC2ContainerRegistryPowerUser` AWS 托管策略附加到您的 IAM 身份，然后重试。有关如何附加策略的更多信息，请参阅《AWS Identity and Access Management 用户指南》中的添加和删除 IAM 身份权限。

修改项目以使用自定义容器

这些说明假设您已经知道如何使用 ， AWS SimSpace Weaver 并希望提高应用程序存储和开发工作流程的 AWS Cloud 效率。

先决条件

- 在 Amazon Elastic Container Registry (Amazon ECR) 中有一个自定义容器。有关创建自定义容器的更多信息，请参阅创建自定义容器。

修改项目以使用自定义容器

1. 为项目的模拟应用程序角色添加使用 Amazon ECR 的权限。

   a. 如果您目前没有具有以下权限的 IAM 策略，请创建该策略。我们建议使用策略名称 `simspaceweaver-ecr`。有关如何创建 IAM 策略的更多信息，请参阅《AWS Identity and Access Management 用户指南》中的创建 IAM 策略。

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

   b. 查找项目的模拟应用程序角色的名称：

      i. 在文本编辑器中，打开 CloudFormation 模板：

```
sdk-folder\PackagingTools\sample-stack-template.yaml
```

      ii. 在 `WeaverAppRole` 下找到 `RoleName` 属性。该值是项目的模拟应用程序角色的名称。

Example

```
AWSTemplateFormatVersion: "2010-09-09"
Resources:
  WeaverAppRole:
    Type: 'AWS::IAM::Role'
    Properties:
      RoleName: 'weaver-MySimulation-app-role'
      AssumeRolePolicyDocument:
        Version: "2012-10-17"
        Statement:
        - Effect: Allow
          Principal:
            Service:
              - 'simspaceweaver.amazonaws.com'
```

c. 将 simspaceweaver-ecr 策略附加到项目的模拟应用程序角色。有关如何附加策略的更多信息，请参阅《AWS Identity and Access Management 用户指南》中的添加和删除 IAM 身份权限。

d. 导航到*sdk-folder*并运行以下命令以更新示例 SimSpace Weaver 堆栈：

```
python setup.py --cloudformation
```

2. 在项目的模拟架构中指定您的容器映像。

- 您可以在 simulation_properties 下添加可选 default_image 属性，为所有域指定默认的自定义容器映像。
- 在 app_config 中为要使用自定义容器映像的域添加 image 属性。指定 Amazon ECR 存储库 URI 作为值。您可以为每个域指定不同的映像。
  - 如果没有为域指定 image，但已指定 default_image，则该域中的应用程序将使用默认映像。
  - 如果image未为网域指定且default_image未指定，则该网域中的应用程序将在标准 SimSpace Weaver 容器中运行。

Example包含自定义容器设置的架构片段

```
sdk_version: "1.17.0"
simulation_properties:
```

```
  log_destination_service: "logs"
  log_destination_resource_name: "MySimulationLogs"
  default_entity_index_key_type: "Vector3<f32>"
  default_image: "111122223333.dkr.ecr.us-west-2.amazonaws.com/my-ecr-
repository:latest" # image to use if no image specified for a domain
domains:
  MyCustomDomain:
    launch_apps_via_start_app_call: {}
    app_config:
      package: "s3://weaver-myproject-111122223333-us-west-2/MyViewApp.zip"
      launch_command: ["MyViewApp"]
      required_resource_units:
        compute: 1
      endpoint_config:
        ingress_ports:
          - 7000
      image: "111122223333.dkr.ecr.us-west-2.amazonaws.com/my-ecr-
repository:latest" # custom container image to use for this domain
  MySpatialDomain:
    launch_apps_by_partitioning_strategy:
      partitioning_strategy: "MyGridPartitioning"
      grid_partition:
        x: 2
        y: 2
    app_config:
      package: "s3://weaver-myproject-111122223333-us-west-2/MySpatialApp.zip"
      launch_command: ["MySpatialApp"]
      required_resource_units:
        compute: 1
      image: "111122223333.dkr.ecr.us-west-2.amazonaws.com/my-ecr-
repository:latest" # custom container image to use for this domain
```

3. 照常构建和上传项目。

有关自定义容器的常见问题解答

问题 1：如果我想更改容器中的内容，该怎么办？

- 对于正在运行的模拟 – 无法更改正在运行的模拟的容器。您必须构建一个新容器并启动使用该容器的新模拟。
- 对于新模拟 – 构建一个新容器，将其上传到 Amazon Elastic Container Registry (Amazon ECR)，然后启动使用该容器的新模拟。

问题 2：如何更改模拟的容器映像？

- 对于正在运行的模拟 – 无法更改正在运行的模拟的容器。您必须启动一个使用该容器的新模拟。
- 对于新的模拟 – 在项目的模拟架构中指定新的容器映像。有关更多信息，请参阅 修改项目以使用自定义容器。

自定义容器故障排除

主题

- AccessDeniedException 将您的图片上传到亚马逊 Elastic Container Registry (Amazon ECR) 时
- 使用自定义容器的模拟无法启动

AccessDeniedException 将您的图片上传到亚马逊 Elastic Container Registry (Amazon ECR) 时

如果您在尝试将容器映像上传到 Amazon ECR 时出现 `AccessDeniedException` 错误，则说明您的 IAM 身份（用户或角色）可能没有使用 Amazon ECR 所需的权限。您可以将`AmazonEC2ContainerRegistryPowerUser` AWS 托管策略附加到您的 IAM 身份，然后重试。有关如何附加策略的更多信息，请参阅《AWS Identity and Access Management 用户指南》中的添加和删除 IAM 身份权限。

使用自定义容器的模拟无法启动

问题排查技巧

- 如果为模拟启用了日志记录，请查看错误日志。
- 在没有自定义容器的情况下测试模拟。
- 在本地测试模拟。有关更多信息，请参阅 当地发展 SimSpace Weaver。

使用 Python

你可以将 Python 用于你的 SimSpace Weaver 应用程序和客户端。Python 软件开发套件 (Python SDK) 作为标准 SimSpace Weaver 应用程序 SDK 可分发包的一部分包含在内。使用 Python 进行开发与使用其他支持的语言进行开发的方式类似。

> **Important**
>
> SimSpace Weaver 仅支持 Python 版本 3.9。

> **Important**
>
> SimSpace Weaver 支持 Python 需要 SimSpace Weaver 版本 1.15.0 或更高版本。

主题

创建 Python 项目

Python 自定义容器

要在中运行基于 Python 的 SimSpace Weaver 模拟 AWS Cloud，您可以创建一个包含必要依赖项的自定义容器。有关更多信息，请参阅 自定义容器。

Python 自定义容器必须包含以下内容：

- gcc
- openssl-devel
- bzip2-devel
- libffi-devel
- wget
- tar
- gzip
- make
- Python（版本 3.9）

如果您使用 PythonBubblesSample 模板创建项目，则可以运行 quick-start.py 脚本（位于项目的 tools 文件夹中）来创建具有必要依赖项的 Docker 映像。该脚本会将映像上传到 Amazon Elastic Container Registry (Amazon ECR)。

quick-start.py 脚本使用以下 Dockerfile：

```
FROM public.ecr.aws/amazonlinux/amazonlinux:2
RUN yum -y install gcc openssl-devel bzip2-devel libffi-devel
RUN yum -y install wget
RUN yum -y install tar
RUN yum -y install gzip
RUN yum -y install make
WORKDIR /opt
RUN wget https://www.python.org/ftp/python/3.9.0/Python-3.9.0.tgz
RUN tar xzf Python-3.9.0.tgz
WORKDIR /opt/Python-3.9.0
RUN ./configure --enable-optimizations
RUN make altinstall
COPY requirements.txt ./
RUN python3.9 -m pip install --upgrade pip
RUN pip3.9 install -r requirements.txt
```

您可以将自己的依赖项添加到 Dockerfile：

```
RUN yum -y install dependency-name
```

requirements.txt 文件包含 PythonBubblesSample 示例模拟所需的 Python 程序包列表：

```
Flask==2.1.1
```

您可以将自己的 Python 程序包依赖项添加到 requirements.txt：

```
package-name==version-number
```

Dockerfile 和 requirements.txt 位于项目的 tools 文件夹中。

> **Important**
>
> 从技术上而言，您不必在 Python 模拟中使用自定义容器，但我们强烈建议您使用。我们提供的标准亚马逊 Linux 2 (AL2) 容器没有 Python。因此，如果您不使用包含 Python 的自定

义容器，则必须在上传到的每个应用程序 zip 文件中包含 Python 和所需的依赖项 SimSpace Weaver。

启动 Python 模拟

你可以像普通 SimSpace Weaver 仿真一样启动基于 Python 的模拟，两者都在 SimSpace Weaver Local 然后 SimSpace Weaver 在 AWS Cloud. 有关更多信息，请参阅中的教程入门 SimSpace Weaver。

`PythonBubblesSample` 包含自己的 Python 示例客户端。有关更多信息，请参阅 示例 Python 客户端。

示例 Python 客户端

如果您使用 `PythonBubblesSample` 模板创建项目，则您的项目将包含一个 Python 示例客户端。您可以使用该示例客户端来查看 `PythonBubblesSample` 模拟。您也可以使用该示例客户端作为起点来创建自己的 Python 客户端。

以下过程假设您已创建 `PythonBubblesSample` 项目并启动了它的模拟。

启动 Python 客户端

1. 在命令提示符窗口中，转到`PyBubbleClient`示例项目文件夹。

   ```
   cd sdk-folder\Clients\HTTP\PyBubbleClient
   ```

2. 运行 Python 客户端。

   ```
   python tkinter_client.py --host ip-address --port port-number
   ```

   参数

   host

   模拟的 IP 地址。对于在中启动的模拟 AWS Cloud，可以在SimSpace Weaver 控制台中找到模拟的 IP 地址或使用快速入门教程中的获取定制化应用程序的 IP 地址和端口号步骤。对于本地模拟，请使用 `127.0.0.1` 作为 IP 地址。

port

模拟的端口号。对于在中启动的模拟 AWS Cloud，这是Actual端口号。您可以在 SimSpace Weaver 控制台中找到模拟的端口号，也可以按照快速入门教程中获取定制化应用程序的 IP 地址和端口号的过程操作。对于本地模拟，请使用 7000 作为端口号。

simsize

要在客户端中显示的最大实体数。

有关使用 Python 的常见问题解答

问题 1：哪些版本的 Python 受支持？

SimSpace Weaver 仅支持 Python 版本 3.9。

排除与 Python 相关的问题

主题

- 创建自定义容器时失败
- Python 模拟无法启动
- Python 模拟或视图客户端会引发错误 ModuleNotFound

创建自定义容器时失败

如果您在运行 `quick-start.py` 后出现错误 `no basic auth credentials`，则您的 Amazon ECR 临时凭证可能有问题。使用您的 AWS 区域 ID 和 AWS 账号运行以下命令：

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

Example

```
aws ecr get-login-password --region us-west-2 | docker login --username AWS --password-
stdin 111122223333.dkr.ecr.region.amazonaws.com
```

> **Important**
>
> 请确保 AWS 区域 您指定的与您用于模拟的相同。使用其中一个 SimSpace Weaver 支持 AWS 区域 的。有关更多信息，请参阅 SimSpace Weaver 端点和配额。

运行 `aws ecr` 命令后，再次运行 `quick-start.py`。

其他需要查看的故障排除资源

- 自定义容器故障排除
- 《Amazon ECR 用户指南》中的 Amazon ECR 故障排除
- 《Amazon ECR 用户指南》中的对 Amazon ECR 进行设置

Python 模拟无法启动

您可能会在模拟的管理日志中看到 `Unable to start app` 错误。如果您的自定义容器创建失败，则可能会发生这种情况。有关更多信息，请参阅 创建自定义容器时失败。有关日志的更多信息，请参阅SimSpace Weaver 在 Amazon CloudWatch 日志中登录。

如果您确定容器没有问题，请查看应用程序的 Python 源代码。您可以使用 … SimSpace Weaver Local 来测试你的应用程序。有关更多信息，请参阅 当地发展 SimSpace Weaver。

Python 模拟或视图客户端会引发错误 ModuleNotFound

当 Python 找不到所需的 Python 程序包时，则会出现 `ModuleNotFound` 错误。

如果您的模拟在中 AWS Cloud，请确保您的自定义容器中列出了所有必需的依赖项`requirements.txt`。如果您对 `requirements.txt` 进行了编辑，请记住再次运行 `quick-start.py`。

如果出现有关 `PythonBubblesSample` 客户端的错误，请使用 `pip` 安装指示的软件包：

```
pip install package-name==version-number
```

对其他引擎的支持

你可以使用自己的自定义 C++ 带有 SimSpace Weaver. 的发动机 目前，我们正在开发对以下引擎的支持。每个引擎都有单独的文档。

> **Important**
>
> 与此处所列引擎的集成是实验性的。它们可供预览。

引擎

- Unity（最低版本 2022.3.19.F1）
- Unreal Engine（最低版本 5.0）

Unity

你必须有 Unity 在使用 Unity 构建 SimSpace Weaver 模拟之前，已经安装了开发环境。有关更多信息，请参阅单独的说明：

```
sdk-folder\Unity-Guide.pdf
```

Unreal Engine

你必须建一个 Unreal Engine 源代码中的专用服务器。 SimSpaceWeaverAppSdkDistributable 包括 for PathfindingSample 的版本 Unreal Engine。 有关更多信息，请参阅单独的说明：

```
sdk-folder\Unreal-Engine-Guide.pdf
```

将许可软件用于 AWS SimSpace Weaver

AWS SimSpace Weaver 允许您使用自己选择的仿真引擎和内容来构建模拟。在使用时 SimSpace Weaver，您有责任获取、维护和遵守您在模拟中使用的任何软件或内容的许可条款。确认您的许可协议允许在虚拟托管环境中部署软件和内容。

使用管理您的资源 AWS CloudFormation

您可以使用 AWS CloudFormation 来管理您的 AWS SimSpace Weaver 资源。 CloudFormation 是一项单独的 AWS 服务，可帮助您以代码形式指定、配置和管理 AWS 基础架构。通过 CloudFormation 创建名为模板的 JSON 或 YAML 文件。您的模板指定了基础设施的详细信息。 CloudFormation 使用您的模板将基础架构配置为一个单元，称为堆栈。当您删除堆栈时，您可以同时 CloudFormation 删除

堆栈中的所有内容。您可以使用标准源代码管理流程来管理模板（例如，在像 Git 这样的版本控制系统中进行跟踪）。有关的更多信息 CloudFormation，请参阅《AWS CloudFormation 用户指南》。

模拟资源

在中 AWS，资源是您可以使用的实体。示例包括亚马逊 EC2 实例、Amazon S3 存储桶或 IAM 角色。您的 SimSpace Weaver 模拟是一种资源。在配置中，您通常在表单中指定 AWS 资源`AWS::service::resource`。对于 SimSpace Weaver，您可以将模拟资源指定为`AWS::SimSpaceWeaver::Simulation`。有关您的模拟资源的更多信息 CloudFormation，请参阅《AWS CloudFormation 用户指南》中的SimSpace Weaver部分。

我该如何 CloudFormation 搭配使用 SimSpace Weaver？

您可以创建一个 CloudFormation 模板来指定要置备的 AWS 资源。您的模板可以指定整个架构、架构的一部分或小型解决方案。例如，您可以为 SimSpace Weaver 解决方案指定一个架构，其中包括 Amazon S3 存储桶、IAM 权限、Amazon Relational Database Service 或 Amazon DynamoDB 中的支持数据库以及您的资源。`Simulation`然后，您可以使用 CloudFormation 将所有这些资源作为一个单元同时置备。

Example创建 IAM 资源并启动模拟的示例模板

以下示例模板创建 SimSpace Weaver 在您的账户中执行操作所需的 IAM 角色和权限。当你创建项目 AWS 区域 时， SimSpace Weaver App SDK 脚本会创建特定角色和权限，但你可以使用 CloudFormation 模板将模拟部署到另一个模拟， AWS 区域 而无需再次运行脚本。例如，您可以这样设置用于灾难恢复目的的备份模拟。

在本示例中，原始模拟名为 `MySimulation`。架构的存储桶已存在于构建堆栈 CloudFormation 的 AWS 区域 位置中。该存储桶包含已正确配置为在 AWS 区域中运行模拟的架构版本。回想一下，该架构指定了应用程序 zip 文件的位置，该文件是与模拟 AWS 区域 位于相同 中的 Amazon S3 存储桶。应用程序压缩存储桶和文件在 CloudFormation 构建堆栈 AWS 区域 时必须已经存在，否则您的模拟将无法启动。请注意，此示例中的存储桶名称包括 AWS 区域，但这并不能确定存储桶的实际位置。您必须确保存储桶实际上位于该存储桶中 AWS 区域（您可以在 Amazon S3 控制台、使用 Amazon S3 或使用中的 APIs Amazon S3 命令查看存储桶属性 AWS CLI）。

此示例使用中的一些内置函数和参数 CloudFormation 来执行变量替换。有关更多信息，请参阅《AWS CloudFormation 用户指南》中的内置函数参考和伪参数参考。

```
AWSTemplateFormatVersion: 2010-09-09
Resources:
  WeaverAppRole:
```

```
    Type: AWS::IAM::Role
    Properties:
      RoleName: SimSpaceWeaverAppRole
      AssumeRolePolicyDocument:
        Version: 2012-10-17
        Statement:
        - Effect: Allow
          Principal:
            Service:
              - simspaceweaver.amazonaws.com
          Action:
            - sts:AssumeRole
      Path: /
      Policies:
        - PolicyName: SimSpaceWeaverAppRolePolicy
          PolicyDocument:
            Version: 2012-10-17
            Statement:
            - Effect: Allow
              Action:
                - logs:PutLogEvents
                - logs:DescribeLogGroups
                - logs:DescribeLogStreams
                - logs:CreateLogGroup
                - logs:CreateLogStream
              Resource: *
            - Effect: Allow
              Action:
                - cloudwatch:PutMetricData
              Resource: *
            - Effect: Allow
              Action:
                - s3:ListBucket
                - s3:PutObject
                - s3:GetObject
              Resource: *
  MyBackupSimulation:
    Type: AWS::SimSpaceWeaver::Simulation
    Properties:
      Name: !Sub 'mySimulation-${AWS::Region}'
      RoleArn: !GetAtt WeaverAppRole.Arn
      SchemaS3Location:
        BucketName: !Sub 'weaver-mySimulation-${AWS::AccountId}-schemas-${AWS::Region}'
```

```
        ObjectKey: !Sub 'schema/mySimulation-${AWS::Region}-schema.yaml'
```

将快照用于 AWS CloudFormation

快照是模拟的备份。以下示例从快照启动新的模拟，而不是从架构启动。此示例中的快照是根据 SimSpace Weaver 应用程序 SDK 项目模拟创建的。 CloudFormation 创建新的仿真资源并使用快照中的数据对其进行初始化。新模拟可能与原始模拟的 `MaximumDuration` 不同。

我们建议您创建并使用原始模拟应用程序角色的副本。如果您删除原始模拟的 CloudFormation 堆栈，则可能会删除该模拟的应用程序角色。

```
Description: "Example - Start a simulation from a snapshot"
Resources:
  MyTestSimulation:
    Type: "AWS::SimSpaceWeaver::Simulation"
    Properties:
      MaximumDuration: "2D"
      Name: "MyTestSimulation_from_snapshot"
      RoleArn: "arn:aws:iam::111122223333:role/weaver-MyTestSimulation-app-role-copy"

      SnapshotS3Location:
        BucketName: "weaver-mytestsimulation-111122223333-artifacts-us-west-2"
        ObjectKey: "snapshot/MyTestSimulation_22-12-15_12_00_00-230428-1207-13.zip"
```

快照

您可以创建快照来随时备份模拟实体数据。 SimSpace Weaver 在 Amazon S3 存储桶中创建 .zip 文件。您可以使用快照创建新的模拟。 SimSpace Weaver 使用快照中存储的实体数据初始化新模拟的 State Fabric，启动创建快照时正在运行的空间和服务应用程序，并将时钟设置为相应的刻度。 SimSpace Weaver 从快照中获取模拟配置，而不是从架构文件中获取。您的应用程序 .zip 文件在 Amazon S3 中的位置必须与原始模拟中的位置相同。您必须单独启动任何自定义应用程序。

主题

- 快照的使用案例
- 使用 SimSpace Weaver 控制台处理快照
- 使用 AWS CLI 来处理快照

- 将快照用于 AWS CloudFormation
- 有关快照的常见问题解答

快照的使用案例

返回到之前的状态并探索分支方案

您可以创建模拟快照以将其保存在特定状态下。然后，您可以从该快照创建多个新的模拟，并探索可能从该状态分支的不同方案。

灾难恢复和安全最佳实践

我们建议您定期备份模拟，特别是对于运行时间超过 1 小时或使用多个工作线程的模拟。备份可以帮助您从灾难和安全事件中恢复。快照为您提供了一种备份模拟的方法。快照要求您的应用程序 .zip 文件与之前一样存在于 Amazon S3 中的相同位置。如果您需要能够将应用程序 .zip 文件移到其他位置，则必须使用自定义备份解决方案。

有关其他最佳实践的更多信息，请参阅使用时的最佳实践 SimSpace Weaver和以下方面的安全最佳实践 SimSpace Weaver。

延长模拟的持续时间

模拟资源是您的模拟在 SimSpace Weaver中的表示形式。所有模拟资源都有一个 `MaximumDuration` 设置。当模拟资源到达 `MaximumDuration` 时，它会自动停止。`MaximumDuration` 的最大值为 `14D`（14 天）。

如果您需要模拟的持续时间超过其模拟资源的 `MaximumDuration`，则可以在模拟资源达到 `MaximumDuration` 之前创建快照。您可以使用快照启动新的模拟（创建新的模拟资源）。SimSpace Weaver 从快照初始化实体数据，启动之前运行的相同空间和服务应用程序，并恢复时钟。您可以启动自定义应用程序并执行任何其他自定义初始化。启动新模拟资源时，可以将 `MaximumDuration` 设置为不同的值。

使用 SimSpace Weaver 控制台处理快照

您可以使用 SimSpace Weaver 控制台创建模拟的快照。

主题

- 使用控制台创建快照
- 使用控制台从快照启动模拟

使用控制台创建快照

创建快照

1. 登录 AWS 管理控制台 并连接到主SimSpace Weaver 机。
2. 从导航窗格中选择模拟。
3. 选定模拟名称旁的单选按钮。模拟的状态必须为已启动。
4. 在页面顶部，选择创建快照。
5. 在 “快照设置” 下的 “快照目标” 中，输入要 SimSpace Weaver 在其中创建快照的存储桶或存储桶和文件夹的 Amazon S3 URI。如果您更喜欢浏览可用的存储桶并选择位置，则可以选择浏览 S3。

   > **Important**
   >
   > Amazon S3 存储桶必须与模拟位于同一 AWS 区域 中。

   > **Note**
   >
   > SimSpace Weaver 在选定的快照目标位置内创建一个`snapshot`文件夹。 SimSpace Weaver 在该`snapshot`文件夹中创建快照.zip 文件。

6. 选择创建快照。

使用控制台从快照启动模拟

要从快照启动模拟，您的快照 .zip 文件必须存在于模拟可访问的 Amazon S3 存储桶中。您的模拟使用在启动模拟时选择的应用程序角色中定义的权限。原始模拟中的所有应用程序 .zip 文件必须与创建快照时位于相同的位置。

从快照启动模拟

1. 登录 AWS 管理控制台 并连接到主SimSpace Weaver 机。
2. 从导航窗格中选择模拟。
3. 在页面顶部，选择启动模拟。
4. 在模拟设置下，输入模拟的名称和可选描述。模拟名称在 AWS 账户中必须是唯一的。
5. 对于模拟启动方法，请选择使用 Amazon S3 中的快照。

6. 对于快照的 Amazon S3 URI，请输入快照文件的 Amazon S3 URI，或者选择浏览 S3，以便浏览并选择该文件。

> **Important**
>
> Amazon S3 存储桶必须与模拟位于同一 AWS 区域 中。

7. 对于 IAM 角色，请选择您的模拟将使用的应用程序角色。
8. 在最长持续时间中，请输入您的模拟资源应运行的最长时间。最大值为 14D。有关最长持续时间的更多信息，请参阅。
9. 如果要添加标签，请在标签 - 可选下，选择添加新标签。
10. 选择启动模拟。

使用 AWS CLI 来处理快照

您可以使用在 AWS CLI 命令提示符 SimSpace Weaver APIs 下调用。您必须正确 AWS CLI 安装和配置。有关更多信息，请参阅版本 2 AWS Command Line Interface 用户指南中的安装或更新最新版本的 AWS CLI。

主题

- 使用 AWS CLI 创建快照
- 使用 AWS CLI 从快照开始模拟

使用 AWS CLI 创建快照

创建快照

- 在命令提示符下，调用 `CreateSnapshot` API。

```
aws simspaceweaver create-snapshot --simulation simulation-name –destination s3-destination
```

参数

模拟

启动的模拟的名称。您可以使用 `aws simspaceweaver list-simulations` 来查看模拟的名称和状态。

destination

一个字符串，用于为您的快照文件指定目标 Amazon S3 存储桶和可选的对象键前缀。您的对象 key prefix 通常是存储桶中的一个文件夹。 SimSpace Weaver 在此目标的`snapshot`文件夹中创建您的快照。

> **Important**
>
> Amazon S3 存储桶必须与模拟位于同一 AWS 区域 中。

示例

```
aws simspaceweaver create-snapshot –simulation
 MyProjectSimulation_23-04-29_12_00_00 –destination BucketName=weaver-
myproject-111122223333-artifacts-us-west-2,ObjectKeyPrefix=myFolder
```

有关 `CreateSnapshot` API 的更多信息，请参阅 AWS SimSpace Weaver API 参考[CreateSnapshot](#)中的。

使用 AWS CLI 从快照开始模拟

从快照启动模拟

- 在命令提示符下，调用 `StartSimulation` API。

```
aws simspaceweaver start-simulation --name simulation-name --role-arn role-arn --
snapshot-s3-location s3-location
```

参数

name

新作业的名称。您的模拟名称必须是唯一的 AWS 账户。您可以使用 `aws simspaceweaver list-simulations` 来查看现有模拟的名称。

role-arn

您的模拟将使用的 Amazon 资源名称 (ARN)。

snapshot-s3-location

一个字符串，用于为您的快照文件指定 Amazon S3 存储桶和对象键。

> **Important**
>
> Amazon S3 存储桶必须与模拟位于同一 AWS 区域 中。

示例

```
aws simspaceweaver start-simulation –name MySimulation –role-arn
 arn:aws:iam::111122223333:role/weaver-MyProject-app-role –snapshot-s3-location
 BucketName=weaver-myproject-111122223333-artifacts-us-west-2,ObjectKey=myFolder/
snapshot/MyProjectSimulation_23-04-29_12_00_00-230429-1530-27.zip
```

有关 `StartSimulation` API 的更多信息，请参阅 AWS SimSpace Weaver API 参考[StartSimulation](#)中的。

有关快照的常见问题解答

在创建快照时，我的模拟会继续运行吗？

在创建快照时，您的模拟资源会继续运行，并且您需要继续为这段时间支付账单费用。这段时间也会计入模拟的最长持续时间。正在创建快照时，您的应用程序不会收到时间周期。开始创建快照后，如果您的时钟状态为 STARTED，则仍会显示 STARTED 状态。快照创建完成后，您的应用程序会再次收到时间周期。如果您的时钟状态为 STOPPED，则会保持为 STOPPED。请注意，即使时钟状态为 STOPPED，具有 STARTED 状态的模拟也仍在运行。

如果正在创建快照，并且我的模拟已达到最长持续时间，会怎样？

您的模拟将完成快照创建，然后在快照过程结束（无论是成功还是失败）后立即停止。我们建议您预先测试快照过程，了解需要多长时间、预计的快照文件大小以及能否成功完成。

如果我停止正在创建快照的模拟，会怎样？

停止模拟后，正在创建的快照会立即停止。它不会创建快照文件。

如何停止正在创建的快照？

要停止正在创建的快照，唯一方法是停止模拟。模拟一旦停止，将无法重新启动。

完成创建快照需要多长时间？

创建快照所需的时间取决于您的模拟。我们建议您预先测试快照过程，了解您的模拟需要多长时间。

我的快照文件会有多大？

快照文件的大小取决于您的模拟。我们建议您预先测试快照过程，了解您的模拟的快照文件可能有多大。

消息收发

消息传递 API 简化了仿真中应用程序与应用程序的通信。 APIs 发送和接收消息是 SimSpace Weaver 应用程序 SDK 的一部分。消息传递目前使用尽力发送和接收消息。 SimSpace Weaver 尝试在下一次模拟报价时 send/receive 发送消息，但没有送达、订购或到达时间保证。

主题

消息传递用例

在仿真应用程序之间进行通信

在模拟中，使用消息传递 API 在应用程序之间进行通信。使用它可以在一定距离内更改实体的状态、更改实体行为或向整个仿真广播信息。

确认收到消息

已发送的消息在邮件标题中包含有关发件人的信息。收到消息后，使用此信息发回确认回复。

将定制化 App 接收到的数据转发到仿真中的其他应用程序

消息传递不能取代客户端连接到中运行的自定义应用程序的方式 SimSpace Weaver。但是，消息传递确实允许用户将接收客户端数据的自定义应用程序中的数据转发到其他没有外部连接的应用程序。消息流也可以反向工作，允许没有外部连接的应用程序将数据转发到自定义应用程序，然后再转发到客户端。

使用消息 APIs

消息包含 APIs 在 SimSpace Weaver 应用程序 SDK（最低版本 1.16.0）中。C++、Python 以及我们与虚幻引擎 5 和 Unity 的集成都支持消息传递。

有两个函数可以处理消息事务：`SendMessage`和`ReceiveMessages`。所有发送的消息都包含目的地和有效负载。`ReceiveMessages`API 会返回应用程序入站消息队列中当前的消息列表。

C++

发送消息

```
AWS_WEAVERRUNTIME_API Result<void> SendMessage(
    Transaction& txn,
    const MessagePayload& payload,
    const MessageEndpoint& destination,
    MessageDeliveryType deliveryType = MessageDeliveryType::BestEffort
    ) noexcept;
```

接收消息

```
AWS_WEAVERRUNTIME_API Result<MessageList> ReceiveMessages(
    Transaction& txn) noexcept;
```

Python

发送消息

```
api.send_message(
 txn, # Transaction
 payload, # api.MessagePayload
```

```
 destination, # api.MessageDestination
 api.MessageDeliveryType.BestEffort # api.MessageDeliveryType
)
```

接收消息

```
api.receive_messages(
 txn, # Transaction
) -> api.MessageList
```

主题

发送消息

消息由交易（类似于其他 Weaver API 调用）、有效负载和目标组成。

消息负载

消息有效载荷是一种灵活的数据结构，最多 256 字节。我们建议将以下作为创建消息负载的最佳实践。

创建消息负载

1. 创建定义消息内容的数据结构（例如 C++ `struct` 中的）。
2. 创建包含要在消息中发送的值的消息负载。
3. 创建 `MessagePayload` 对象。

消息目的地

消息的目的地由`MessageEndpoint`对象定义。这包括端点类型和终端节点 ID。目前唯一支持的端点类型是`Partition`，它允许您在模拟中将消息发送到其他分区。终端节点 ID 是目标目标的分区 ID。

您只能在一封邮件中提供 1 个目标地址。如果您想同时向多个分区发送消息，请创建并发送多条消息。

有关如何从某个位置解析消息端点的指导，请参阅处理消息传递时的提示。

发送消息

您可以在创建目标和负载对象之后使用 SendMessage API。

C++

```
Api::SendMessage(transaction, payload, destination,
 MessageDeliveryType::BestEffort);
```

Python

```
api.send_message(txn, payload, destination, api.MessageDeliveryType.BestEffort)
```

发送消息的完整示例

以下示例演示了如何构造和发送通用消息。此示例发送了 16 条单独的消息。每条消息都包含一个值介于 0 和 15 之间的有效载荷以及当前的模拟滴答声。

Example

C++

```
// Message struct definition
struct MessageTickAndId
{
    uint32_t id;
    uint32_t tick;
};

Aws::WeaverRuntime::Result<void> SendMessages(Txn& txn) noexcept
{
     // Fetch the destination MessageEndpoint with the endpoint resolver
    WEAVERRUNTIME_TRY(
        Api::MessageEndpoint destination,
        Api::Utils::MessageEndpointResolver::ResolveFromPosition(
        txn,
            "MySpatialSimulation",
            Api::Vector2F32 {231.3, 654.0}
        )
    );
```

```
    Log::Info("destination: ", destination);

    WEAVERRUNTIME_TRY(auto tick, Api::CurrentTick(txn));

    uint16_t numSentMessages = 0;
    for (std::size_t i=0; i<16; i++)
    {
        // Create the message that'll be serialized into payload
        MessageTickAndId message {i, tick.value};

        // Create the payload out of the struct
        const Api::MessagePayload& payload = Api::Utils::CreateMessagePayload(
            reinterpret_cast<const std::uint8_t*>(&message),
            sizeof(MessageTickAndId)
        );

        // Send the payload to the destination
        Result<void> result = Api::SendMessage(txn, payload, destination);
        if (result.has_failure())
        {
            // SendMessage has failure modes, log them
            auto error = result.as_failure().error();
            std::cout<< "SendMessage failed, ErrorCode: " << error << std::endl;
            continue;
        }

        numSentMessages++;
    }

    std::cout << numSentMessages << " messages is sent to endpoint"
       << destination << std::endl;
    return Aws::WeaverRuntime::Success();
}
```

Python

```
# Message data class
@dataclasses.dataclass
class MessageTickAndId:
    tick: int = 0
    id: int = 0

# send messages
```

```
def _send_messages(self, txn):
    tick = api.current_tick(txn)
    num_messages_to_send = 16

    # Fetch the destination MessageEndpoint with the endpoint resolver
    destination = api.utils.resolve_endpoint_from_domain_name_position(
       txn,
       "MySpatialSimulation",
       pos
   )
    Log.debug("Destination_endpoint = %s", destination_endpoint)

   for id in range(num_messages_to_send):
       # Message struct that'll be serialized into payload
        message_tick_and_id = MessageTickAndId(id = id, tick = tick.value)

       # Create the payload out of the struct
        message_tick_and_id_data = struct.pack(
           '<ii',
           message_tick_and_id.id,
           message_tick_and_id.tick
       )
        payload = api.MessagePayload(list(message_tick_and_id_data))

        # Send the payload to the destination
        Log.debug("Sending message: %s, endpoint: %s",
           message_tick_and_id,
           destination
       )
        api.send_message(
           txn,
           payload,
           destination,
           api.MessageDeliveryType.BestEffort
       )

    Log.info("Sent %s messages to %s", num_messages_to_send, destination)
    return True
```

接收消息

SimSpace Weaver 将消息传送到分区的入站消息队列中。使用 `ReceiveMessages` API 获取包含队列消息的`MessageList`对象。使用 `ExtractMessage` API 处理每条消息以获取消息数据。

Example

C++

```
Result<void> ReceiveMessages(Txn& txn) noexcept
{
     // Fetch all the messages sent to the partition owned by the app
    WEAVERRUNTIME_TRY(auto messages, Api::ReceiveMessages(txn));
    std::cout << "Received" << messages.messages.size() << " messages" << std::endl;
    for (Api::Message& message : messages.messages)
    {
        std::cout << "Received message: " << message << std::endl;

         // Deserialize payload to the message struct
        const MessageTickAndId& receivedMessage
            = Api::Utils::ExtractMessage<MessageTickAndId>(message);
        std::cout << "Received MessageTickAndId, Id: " << receivedMessage.id
            <<", Tick: " << receivedMessage.tick << std::endl;
    }

    return Aws::WeaverRuntime::Success();
}
```

Python

```
# process incoming messages
def _process_incoming_messages(self, txn):
    messages = api.receive_messages(txn)
    for message in messages:
        payload_list = message.payload.data
        payload_bytes = bytes(payload_list)
        message_tick_and_id_data_struct
           = MessageTickAndId(*struct.unpack('<ii', payload_bytes))

        Log.debug("Received message. Header: %s, message: %s",
                    message.header, message_tick_and_id_data_struct)

    Log.info("Received %s messages", len(messages))
```

```
    return True
```

回复发件人

收到的每封邮件都包含一个邮件标头，其中包含有关邮件原始发件人的信息。你可以使用 message.header.source_endpoint 发送回复。

Example

C++

```
Result<void> ReceiveMessages(Txn& txn) noexcept
{
     // Fetch all the messages sent to the partition owned by the app
    WEAVERRUNTIME_TRY(auto messages, Api::ReceiveMessages(txn));
    std::cout << "Received" << messages.messages.size() << " messages" << std::endl;
    for (Api::Message& message : messages.messages)
    {
        std::cout << "Received message: " << message << std::endl;

         // Deserialize payload to the message struct
        const MessageTickAndId& receivedMessage
            = Api::Utils::ExtractMessage<MessageTickAndId>(message);
        std::cout << "Received MessageTickAndId, Id: " << receivedMessage.id
            <<", Tick: " << receivedMessage.tick << std::endl;

        // Get the sender endpoint and payload to bounce the message back
        Api::MessageEndpoint& sender = message.header.source_endpoint;
        Api::MessagePayload& payload = message.payload;
        Api::SendMessage(txn, payload, sender);
    }

    return Aws::WeaverRuntime::Success();
}
```

Python

```
# process incoming messages
def _process_incoming_messages(self, txn):
    messages = api.receive_messages(txn)
    for message in messages:
        payload_list = message.payload.data
```

```
        payload_bytes = bytes(payload_list)
        message_tick_and_id_data_struct
           = MessageTickAndId(*struct.unpack('<ii', payload_bytes))

        Log.debug("Received message. Header: %s, message: %s",
                    message.header, message_tick_and_id_data_struct)
      # Get the sender endpoint and payload
      # to bounce the message back
      sender = message.header.source_endpoint
      payload = payload_list
      api.send_message(
          txn,
          payload_list,
          sender,
          api.MessageDeliveryType.BestEffort

   Log.info("Received %s messages", len(messages))
   return True
```

何时使用消息传递

中的消息传递为仿真应用程序之间交换信息 SimSpace Weaver 提供了另一种模式。订阅提供了一种拉取机制，用于从仿真的特定应用程序或区域读取数据；消息提供了一种推送机制，用于将数据发送到仿真的特定应用程序或区域。

以下是两个用例，在这些用例中，使用消息传递推送数据比通过订阅提取或读取数据更有帮助。

Example 1：向其他应用程序发送命令以更改实体位置

```
// Message struct definition
struct MessageMoveEntity
{
     uint64_t entityId;
    std::array<float, 3> destinationPos;
};

// Create the message
MessageMoveEntity message {45, {236.67, 826.22, 0.0} };

// Create the payload out of the struct
const Api::MessagePayload& payload = Api::Utils::CreateMessagePayload(
    reinterpret_cast<const std::uint8_t*>(&message),
```

```
    sizeof(MessageTickAndId)
);

// Grab the MessageEndpoint of the recipient app.
Api::MessageEndpoint destination = ...

// One way is to resolve it from the domain name and position
WEAVERRUNTIME_TRY(
    Api::MessageEndpoint destination,
    Api::Utils::MessageEndpointResolver::ResolveFromPosition(
    txn,
        "MySpatialSimulation",
        Api::Vector2F32 {200.0, 100.0}
    )
);

// Then send the message
Api::SendMessage(txn, payload, destination);
```

在接收方，应用程序更新实体的位置并将其写入State Fabric。

```
Result<void> ReceiveMessages(Txn& txn) noexcept
{
    WEAVERRUNTIME_TRY(auto messages, Api::ReceiveMessages(txn));
    for (Api::Message& message : messages.messages)
    {
        std::cout << "Received message: " << message << std::endl;
         // Deserialize payload to the message struct
        const MessageMoveEntity& receivedMessage
            = Api::Utils::ExtractMessage<MessageMoveEntity>(message);

        ProcessMessage(txn, receivedMessage);
    }

    return Aws::WeaverRuntime::Success();
}

void ProcessMessage(Txn& txn, const MessageMoveEntity& receivedMessage)
{
     // Get the entity corresponding to the entityId
    Entity entity = EntityFromEntityId (receivedMessage.entityId);

    // Update the position and write to StateFabric
```

```
    WEAVERRUNTIME_TRY(Api::StoreEntityIndexKey(
            txn,
            entity,
            k_vector3f32TypeId, // type id of the entity
            reinterpret_cast<std::int8_t*>(&receivedMessage.destinationPos),
            sizeof(receivedMessage.destinationPos)));

}
```

Example 2：向空间应用程序发送创建实体消息

```
struct WeaverMessage
{
    const Aws::WeaverRuntime::Api::TypeId messageTypeId;
};

const Aws::WeaverRuntime::Api::TypeId k_createEntityMessageTypeId = { 1 };

struct CreateEntityMessage : WeaverMessage
{
    const Vector3 position;
   const Aws::WeaverRuntime::Api::TypeId typeId;
};


CreateEntityMessage messageData {
    k_createEntityMessageTypeId,
    Vector3{ position.GetX(), position.GetY(), position.GetZ() },
    Api::TypeId { 0 }
}

WEAVERRUNTIME_TRY(Api::MessageEndpoint destination,
 Api::Utils::MessageEndpointResolver::ResolveFromPosition(
    transaction, "MySpatialDomain", DemoFramework::ToVector2F32(position)
));

Api::MessagePayload payload = Api::Utils::CreateMessagePayload(
    reinterpret_cast<const uint8_t*>(&messageData),
    sizeof(CreateEntityMessage));

Api::SendMessage(transaction, payload, destination);
```

在接收方，该应用程序在 State Fabric 中创建一个新实体并更新其位置。

```
Result<void> ReceiveMessages(Txn& txn) noexcept
{
    WEAVERRUNTIME_TRY(auto messageList, Api::ReceiveMessages(transaction));
    WEAVERRUNTIME_TRY(auto tick, Api::CurrentTick(transaction));
    for (auto& message : messageList.messages)
    {
        // cast to base WeaverMessage type to determine MessageTypeId
        WeaverMessage weaverMessageBase =
 Api::Utils::ExtractMessage<WeaverMessage>(message);
        if (weaverMessageBase.messageTypeId == k_createEntityMessageTypeId)
        {
            CreateEntityMessage createEntityMessageData =
                Api::Utils::ExtractMessage<CreateEntityMessage>(message);
        CreateActorFromMessage(transaction, createEntityMessageData));
        }
        else if (weaverMessageBase.messageTypeId == k_tickAndIdMessageTypeId)
        {
            ...
        }
    }
}

void ProcessMessage(Txn& txn, const CreateEntityMessage& receivedMessage)
{
    // Create entity
    WEAVERRUNTIME_TRY(
        Api::Entity entity,
        Api::CreateEntity(transaction, receivedMessage.typeId)
    );

    // Update the position and write to StateFabric
    WEAVERRUNTIME_TRY(Api::StoreEntityIndexKey(
        transaction,
        entity,
        receivedMessage.typeId,
        reinterpret_cast<std::int8_t*>(&receivedMessage.position),
        sizeof(receivedMessage.position)));
}
```

处理消息传递时的提示

根据位置或应用程序名称解析端点

您可以使用该AllPartitions函数获取空间边界和域 ID，以确定消息分 IDs区和消息目的地。但是，如果您知道要发送消息的位置，但不知道其分区 ID，则可以使用该 MessageEndpointResolver 函数。

```
/**
* Resolves MessageEndpoint's from various inputs
**/
class MessageEndpointResolver
{
    public:
    /**
    * Resolves MessageEndpoint from position information
    **/
    Result<MessageEndpoint> ResolveEndpointFromPosition(
        const DomainId& domainId,
        const weaver_vec3_f32_t& pos);

    /**
    * Resolves MessageEndpoint from custom app name
    **/
    Result<MessageEndpoint> ResolveEndpointFromCustomAppName(
        const DomainId& domainId,
        const char* agentName);
};
```

序列化和反序列化消息负载

您可以使用以下函数来创建和读取消息负载。有关更多信息，请参阅本地系统上应用程序 SDK 库中的 MessagingUtils .h。

```
/**
* Utility function to create MessagePayload from a custom type
*
* @return The @c MessagePayload.
*/
template <class T>
AWS_WEAVERRUNTIME_API MessagePayload CreateMessagePayload(const T& message) noexcept
{
```

```
    const std::uint8_t* raw_data = reinterpret_cast<const std::uint8_t*>(&message);

    MessagePayload payload;
    std::move(raw_data, raw_data + sizeof(T), std::back_inserter(payload.data));

    return payload;
}

/**
* Utility function to convert MessagePayload to custom type
*/
template <class T>
AWS_WEAVERRUNTIME_API T ExtractMessage(const MessagePayload& payload) noexcept
{
    return *reinterpret_cast<const T*>(payload.data.data());
}
```

消息错误和疑难解答

使用消息时，您可能会遇到以下错误 APIs。

端点解析错误

这些错误可能发生在应用程序发送消息之前。

域名检查

向无效端点发送消息会导致以下错误：

```
ManifoldError::InvalidArgument {"No DomainId found for the given domain name" }
```

当您尝试向定制化 App 发送消息，但该定制化 App 尚未加入模拟时，就会发生这种情况。使用该 `DescribeSimulation` API 确保您的定制化应用程序已启动，然后再向其发送消息。这种行为在 SimSpace Weaver Local和中是一样的 AWS Cloud。

位置检查

尝试解析具有有效域名但位置无效的端点会导致以下错误。

```
ManifoldError::InvalidArgument {"Could not resolve endpoint from domain : DomainId
 { value: domain-id } and position: Vector2F32 { x: x-position, y: y-position}" }
```

我们建议使用 SimSpace Weaver 应用程序 SDK 中包含的`MessageUtils`库中的。`MessageEndpointResolver`

消息发送错误

应用程序发送消息时可能会出现以下错误。

已超过每个应用程序、每次点击的消息发送限制

目前，每个应用程序每次仿真跳动可以发送的消息数量限制为 128。同一报价的后续调用将失败，并出现以下错误：

```
ManifoldError::CapacityExceeded {"At Max Outgoing Message capacity: {}", 128}
```

SimSpace Weaver 尝试在下次勾选时发送未发送的消息。降低发送频率以解决此问题。合并小于 256 字节限制的消息负载，以减少出站消息的数量。

这种行为在中SimSpace Weaver Local和之中都是一样的 AWS Cloud。

已超出消息有效负载大小限制

当前消息有效载荷大小限制为 256 字节，两者SimSpace Weaver Local兼而有之 AWS Cloud。发送有效载荷大于 256 字节的消息会导致以下错误：

```
ManifoldError::CapacityExceeded {"Message data too large! Max size: {}", 256}
```

SimSpace Weaver 会检查每条消息，只拒绝超过限制的消息。例如，如果您的应用程序尝试发送 10 条消息，但有 1 条未通过检查，则只有这 1 条消息被拒绝。 SimSpace Weaver 发送其他 9 条消息。

这种行为在SimSpace Weaver Local和中是一样的 AWS Cloud。

目的地与源相同

应用程序无法向其拥有的分区发送消息。如果应用程序向其拥有的分区发送消息，则会出现以下错误。

```
ManifoldError::InvalidArgument { "Destination is the same as source" }
```

这种行为在SimSpace Weaver Local和中是一样的 AWS Cloud。

尽力发送消息

SimSpace Weaver 不能保证消息的传送。该服务将尝试在随后的模拟滴答中完成消息的传送，但消息可能会丢失或延迟。

使用时的最佳实践 SimSpace Weaver

在使用时，我们建议采用以下最佳实践 SimSpace Weaver。

主题

设置账单警报

即使不再需要这些资源，也可以很容易地调配资源 AWS 并让它们一直运行。这可能会导致成本失控，您在收到账单时可能会感到惊讶。您可以在 Amazon 中配置警报 CloudWatch ，当您的费用超过您设定的阈值时，该警报将触发并通知您。您可以使用成本管理工具检查成本。有关更多信息，请参阅:

- 创建账单警报以监控您的预估 AWS 费用
- 什么是 AWS Cost Management

使用 SimSpace Weaver Local

我们建议您先使用SimSpace Weaver Local来开发和测试您的模拟，然后再将其上传到中的 SimSpace Weaver AWS Cloud服务。使用 SimSpace Weaver Local 进行开发的好处包括：

- 无需等待大量上传
- 可创建的本地模拟数量没有限制
- 无需为本地计算机上的计算时间付费
- 可直接访问应用程序的控制台输出
- 修改、重建和重新启动您的本地模拟，无需在中重新创建它 AWS Cloud

可停止不需要的模拟

当模拟运行时，系统将会计费。您必须停止模拟，系统才会停止计费。运行的模拟还会计入最大模拟次数的配额。如果配置了日志记录，运行的模拟还会生成大量日志，您也需要为此支付账单费用。为了避免额外付费，您应该停止所有不需要的模拟。

> **Important**
>
> 停止模拟时钟并不会停止模拟，时钟只是停止向应用程序发布时钟周期。模拟一旦停止，将无法重新启动。

删除不需要的资源

您在中创建的每个模拟 SimSpace Weaver 还会在其他 AWS 服务中创建资源。您需要为这些其他服务中的资源和数据支付账单费用。运行和失败的模拟还会计入最大模拟次数的配额。您应该删除不需要的失败模拟，以便开始新的模拟。删除模拟时，可能不会删除其他 AWS 服务中存在的模拟资源。例如，Amazon Logs 中的任何模拟 CloudWatch 日志数据都将保留在那里，直到您将其删除。您需要为这些日志数据支付账单费用。如果您不再需要这些模拟，则应清理这些模拟的所有相关资源。

备份

最好为所有数据创建备份和备份计划。你不应该仅仅因为你的数据就假 AWS 设你不必对其进行备份。如果需要备份模拟状态，您必须创建自己的系统。考虑使用多个， AWS 区域 并制定计划，以便在需要时能够快速将生产工作负载切换到另一个 AWS 区域 工作负载。有关 AWS 区域 该支持的更多信息 SimSpace Weaver，请参阅SimSpace Weaver 端点和配额。

安全性 AWS SimSpace Weaver

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 SimSpace Weaver，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 SimSpace Weaver。以下主题向您介绍如何进行配置 SimSpace Weaver 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 SimSpace Weaver 资源。

主题

- 中的数据保护 AWS SimSpace Weaver
- Identity and Access Management AWS SimSpace Weaver
- 中的安全事件记录和监控 AWS SimSpace Weaver
- 的合规性验证 AWS SimSpace Weaver
- 韧性在 AWS SimSpace Weaver
- 中的基础设施安全 AWS SimSpace Weaver
- 中的配置和漏洞分析 AWS SimSpace Weaver
- 以下方面的安全最佳实践 SimSpace Weaver

中的数据保护 AWS SimSpace Weaver

分 AWS 担责任模型适用于中的数据保护 AWS SimSpace Weaver。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结构上的内容的控制。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 Security Blog 上的 AWS Shared Responsibility Model and 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。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）。
- 用于 SSL/TLS 与 AWS 资源通信。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。有关使用 CloudTrail 跟踪捕获 AWS 活动的信息，请参阅《AWS CloudTrail 用户指南》中的使用跟 CloudTrail 踪。
- 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 140-3 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 140-3 版》https://aws.amazon.com/compliance/fips/。

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 SimSpace Weaver 或以其他 AWS 服务 方式使用控制台 AWS CLI、API 或时 AWS SDKs。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 URL，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

当数据位于非易失性（永久）数据存储器（例如磁盘）中时，将其视为静态数据。位于易失性数据存储器中的数据（例如内存和寄存器）不是静态数据。

使用时 SimSpace Weaver，唯一的静态数据是：

- 上传到 Amazon Simple Storage Service (Amazon S3) 的应用程序和架构
- 存储在 Amazon 中的模拟日志数据 CloudWatch

停止仿真后，内部 SimSpace Weaver 使用的其他数据不会保留。

要了解如何加密静态数据，请参阅：

- 在 Amazon S3 中加密数据
- 加密日志数据

传输中加密

您通过 AWS Command Line Interface (AWS CLI)、 AWS SDK 和 SimSpace Weaver 应用程序 SDK 与 SimSpace Weaver API 的连接在签名版本 4 签名过程中使用 TLS 加密。 AWS 使用 IAM 为用于连接的安全凭证定义的访问策略管理身份验证。

在内部， SimSpace Weaver 使用 TLS 连接到其使用的其他 AWS 服务。

> **Important**
>
> 您的应用程序与其客户之间的通信不涉及 SimSpace Weaver。如果需要，您有责任对与模拟客户端的通信进行加密。我们建议您创建一种解决方案，对通过客户端连接传输的所有数据进行加密。

要详细了解支持您的加密解决方案的 AWS 服务，请参阅AWS 安全博客。

互联网络流量隐私

SimSpace Weaver 计算资源位于所有 SimSpace Weaver 客户共享的 1 个 Amazon VPC 内。所有内部 SimSpace Weaver 服务流量都保留在 AWS 网络内，不会通过互联网传输。模拟客户端和您的应用程序之间的通信通过互联网传输。

Identity and Access Management AWS SimSpace Weaver

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 SimSpace Weaver 资源。您可以使用 IAM AWS 服务 ，无需支付额外费用。

主题

- 受众

- 使用身份进行身份验证
- 使用策略管理访问
- 如何 AWS SimSpace Weaver 与 IAM 配合使用
- 基于身份的策略示例 AWS SimSpace Weaver
- SimSpace Weaver 为您创建的权限
- 防止跨服务混淆代理
- 对 AWS SimSpace Weaver 身份和访问进行故障排除

受众

您的使用方式 AWS Identity and Access Management (IAM) 因您的角色而异：

- 服务用户：如果您无法访问功能，请从管理员处请求权限（请参阅对 AWS SimSpace Weaver 身份和访问进行故障排除）
- 服务管理员：确定用户访问权限并提交权限请求（请参阅如何 AWS SimSpace Weaver 与 IAM 配合使用）
- IAM 管理员：编写用于管理访问权限的策略（请参阅基于身份的策略示例 AWS SimSpace Weaver）

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 用户身份进行身份验证 AWS 账户根用户，或者通过担任 IAM 角色进行身份验证。

您可以使用来自身份源的证书 AWS IAM Identity Center（例如（IAM Identity Center）、单点登录身份验证或 Google/Facebook 证书，以联合身份登录。有关登录的更多信息，请参阅《AWS 登录 用户指南》中的如何登录您的 AWS 账户。

对于编程访问， AWS 提供 SDK 和 CLI 来对请求进行加密签名。有关更多信息，请参阅《IAM 用户指南》中的适用于 API 请求的AWS 签名版本 4。

AWS 账户 root 用户

创建时 AWS 账户，首先会有一个名为 AWS 账户 root 用户的登录身份，该身份可以完全访问所有资源 AWS 服务 和资源。我们强烈建议不要使用根用户进行日常任务。有关需要根用户凭证的任务，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，要求人类用户使用与身份提供商的联合身份验证才能 AWS 服务 使用临时证书进行访问。

联合身份是指来自您的企业目录、Web 身份提供商的用户 Directory Service ，或者 AWS 服务 使用来自身份源的凭据进行访问的用户。联合身份代入可提供临时凭证的角色。

要集中管理访问权限，建议使用。 AWS IAM Identity Center有关更多信息，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

IAM 用户是对某个人员或应用程序具有特定权限的一个身份。建议使用临时凭证，而非具有长期凭证的 IAM 用户。有关更多信息，请参阅 IAM 用户指南中的要求人类用户使用身份提供商的联合身份验证才能 AWS 使用临时证书进行访问。

IAM 组指定一组 IAM 用户，便于更轻松地对大量用户进行权限管理。有关更多信息，请参阅《IAM 用户指南》中的 IAM 用户使用案例。

IAM 角色

IAM 角色是具有特定权限的身份，可提供临时凭证。您可以通过从用户切换到 IAM 角色（控制台）或调用 AWS CLI 或 AWS API 操作来代入角色。有关更多信息，请参阅《IAM 用户指南》中的担任角色的方法。

IAM 角色对于联合用户访问、临时 IAM 用户权限、跨账户访问、跨服务访问以及在 Amazon EC2 上运行的应用程序非常有用。有关更多信息，请参阅《IAM 用户指南》中的 IAM 中的跨账户资源访问。

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。策略定义了与身份或资源关联时的权限。 AWS 在委托人提出请求时评估这些政策。大多数策略都以 JSON 文档的 AWS 形式存储在中。有关 JSON 策略文档的更多信息，请参阅《IAM 用户指南》中的 JSON 策略概述。

管理员使用策略，通过定义哪个主体可以在什么条件下对哪些资源执行哪些操作来指定谁有权访问什么。

默认情况下，用户和角色没有权限。IAM 管理员创建 IAM 策略并将其添加到角色中，然后用户可以担任这些角色。IAM 策略定义权限，与执行操作所用的方法无关。

基于身份的策略

基于身份的策略是您附加到身份（用户、组或角色）的 JSON 权限策略文档。这些策略控制身份可以执行什么操作、对哪些资源执行以及在什么条件下执行。要了解如何创建基于身份的策略，请参阅《IAM 用户指南》中的使用客户管理型策略定义自定义 IAM 权限。

基于身份的策略可以是内联策略（直接嵌入到单个身份中）或托管策略（附加到多个身份的独立策略）。要了解如何在托管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。示例包括 IAM 角色信任策略和 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。您必须在基于资源的策略中指定主体。

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。

其他策略类型

AWS 支持其他策略类型，这些策略类型可以设置更常见的策略类型授予的最大权限：

- 权限边界 – 设置基于身份的策略可以授予 IAM 实体的最大权限。有关更多信息，请参阅《 IAM 用户指南》中的 IAM 实体的权限边界。
- 服务控制策略 (SCPs)-在中指定组织或组织单位的最大权限 AWS Organizations。有关更多信息，请参阅《AWS Organizations 用户指南》中的服务控制策略。
- 资源控制策略 (RCPs)-设置账户中资源的最大可用权限。有关更多信息，请参阅《AWS Organizations 用户指南》中的资源控制策略 (RCPs)。
- 会话策略 – 在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。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 SimSpace Weaver 与 IAM 配合使用

在使用 IAM 管理访问权限之前 SimSpace Weaver，请先了解有哪些 IAM 功能可供使用 SimSpace Weaver。

您可以搭配使用的 IAM 功能 AWS SimSpace Weaver

IAM 功能	SimSpace Weaver 支持
基于身份的策略	是
基于资源的策略	否
策略操作	是
策略资源	是
策略条件键（特定于服务）	是
ACLs	否
ABAC（策略中的标签）	是
临时凭证	是
主体权限	是
服务角色	是
服务关联角色	否

要全面了解 SimSpace Weaver 以及其他 AWS 服务如何与大多数 IAM 功能配合使用，请参阅 IAM 用户指南中的与 IAM 配合使用的AWS 服务。

基于身份的策略 SimSpace Weaver

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》中的使用客户管理型策略定义自定义 IAM 权限。

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。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 SimSpace Weaver

要查看 SimSpace Weaver 基于身份的策略的示例，请参阅。基于身份的策略示例 AWS SimSpace Weaver

内部基于资源的政策 SimSpace Weaver

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 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。

要启用跨账户访问，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。有关更多信息，请参阅《IAM 用户指南》中的 IAM 中的跨账户资源访问。

的政策行动 SimSpace Weaver

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。在策略中包含操作以授予执行关联操作的权限。

要查看 SimSpace Weaver 操作列表，请参阅《服务授权参考》 AWS SimSpace Weaver中定义的操作。

正在执行的策略操作在操作前 SimSpace Weaver 使用以下前缀：

```
simspaceweaver
```

要在单个语句中指定多项操作，请使用逗号将它们隔开。

```
"Action": [
      "simspaceweaver:action1",
      "simspaceweaver:action2"
         ]
```

要查看 SimSpace Weaver 基于身份的策略的示例，请参阅。基于身份的策略示例 AWS SimSpace Weaver

的政策资源 SimSpace Weaver

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。作为最佳实践，请使用其 Amazon 资源名称（ARN）指定资源。对于不支持资源级权限的操作，请使用通配符 (*) 指示语句应用于所有资源。

```
"Resource": "*"
```

要查看 SimSpace Weaver 资源类型及其列表 ARNs，请参阅《服务授权参考》 AWS SimSpace Weaver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 SimSpace Weaver定义的操作。

要查看 SimSpace Weaver 基于身份的策略的示例，请参阅。基于身份的策略示例 AWS SimSpace Weaver

的策略条件密钥 SimSpace Weaver

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。

`Condition` 元素根据定义的条件指定语句何时执行。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。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 SimSpace Weaver 条件键列表，请参阅《服务授权参考》 AWS SimSpace Weaver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 SimSpace Weaver。

要查看 SimSpace Weaver 基于身份的策略的示例，请参阅。基于身份的策略示例 AWS SimSpace Weaver

中的访问控制列表 (ACLs) SimSpace Weaver

支持 ACLs：否

访问控制列表 (ACLs) 控制哪些委托人（账户成员、用户或角色）有权访问资源。 ACLs 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) SimSpace Weaver

支持 ABAC（策略中的标签）：是

基于属性的访问权限控制（ABAC）是一种授权策略，该策略基于称为标签的属性来定义权限。您可以将标签附加到 IAM 实体和 AWS 资源，然后设计 ABAC 策略以允许在委托人的标签与资源上的标签匹配时进行操作。

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》中的使用 ABAC 授权定义权限。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 SimSpace Weaver

支持临时凭证：是

临时证书提供对 AWS 资源的短期访问权限，并且是在您使用联合身份或切换角色时自动创建的。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 用户指南》中的 IAM 中的临时安全凭证和使用 IAM 的。AWS 服务

的跨服务主体权限 SimSpace Weaver

支持转发访问会话（FAS）：是

转发访问会话 (FAS) 使用调用主体的权限 AWS 服务，再加上 AWS 服务 向下游服务发出请求的请求。有关发出 FAS 请求时的策略详情，请参阅转发访问会话。

的服务角色 SimSpace Weaver

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 SimSpace Weaver 功能。只有在 SimSpace Weaver 提供操作指导时才编辑服务角色。

A SimSpace Weaver pp SDK 脚本使用 CloudFormation 模板在其他 AWS 服务中创建资源以支持您的模拟。其中一个资源是模拟的应用程序角色。 SimSpace Weaver 代入应用程序角色代表您 AWS 账户执行操作，例如将日志数据写入 CloudWatch 日志。有关应用程序角色的更多信息，请参阅 SimSpace Weaver 为您创建的权限。

的服务相关角色 SimSpace Weaver

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关联角色的权限。

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 SimSpace Weaver

默认情况下，用户和角色没有创建或修改 SimSpace Weaver 资源的权限。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。

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（控制台）。

有关由 SimSpace Weaver定义的操作和资源类型（包括每种资源类型的格式）的详细信息，请参阅《服务授权参考》 AWS SimSpace Weaver中的操作、资源和条件密钥。 ARNs

主题

- 策略最佳实践
- 使用控制 SimSpace Weaver 台
- 允许用户查看他们自己的权限
- 允许用户创建和运行模拟

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 SimSpace Weaver 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，请遵循以下指南和建议：

- 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 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》中的使用 IAM Access Analyzer 验证策略。
- 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》中的使用 MFA 保护 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 中的安全最佳实践。

使用控制 SimSpace Weaver 台

要访问 AWS SimSpace Weaver 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 SimSpace Weaver 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 SimSpace Weaver 控制台，还需要将 SimSpace Weaver *ConsoleAccess*或*ReadOnly* AWS 托管策略附加到实体。有关更多信息，请参阅《IAM 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": "*"
        }
    ]
}
```

允许用户创建和运行模拟

该示例 IAM 策略提供了在 SimSpace Weaver中创建和运行模拟所需的基本权限。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "CreateAndRunSimulations",
            "Effect": "Allow",
            "Action": [
                "simspaceweaver:*",
                "iam:GetRole",
                "iam:ListRoles",
                "iam:CreateRole",
                "iam:DeleteRole",
                "iam:UpdateRole",
                "iam:CreatePolicy",
                "iam:AttachRolePolicy",
                "iam:PutRolePolicy",
                "iam:GetRolePolicy",
                "iam:DeleteRolePolicy",
                "s3:PutObject",
                "s3:GetObject",
                "s3:ListAllMyBuckets",
                "s3:PutBucketPolicy",
                "s3:CreateBucket",
                "s3:ListBucket",
                "s3:PutEncryptionConfiguration",
                "s3:DeleteBucket",
                "cloudformation:CreateStack",
                "cloudformation:UpdateStack",
```

```
                "cloudformation:DescribeStacks"
            ],
            "Resource": "*"
        },
        {
            "Sid": "PassAppRoleToSimSpaceWeaver",
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "iam:PassedToService": "simspaceweaver.amazonaws.com"
                }
            }
        }
    ]
}
```

SimSpace Weaver 为您创建的权限

当您创建 SimSpace Weaver 项目时，该服务将使用名称`weaver-`*`project-name`*`-app-role`和 IAM 信任策略创建一个 AWS Identity and Access Management (IAM) 角色。信任策略 SimSpace Weaver 允许代入该角色，以便它可以为您执行操作。

应用程序角色权限策略

该模拟应用程序角色具有以下权限策略。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "logs:PutLogEvents",
                "logs:DescribeLogGroups",
                "logs:DescribeLogStreams",
                "logs:CreateLogGroup",
                "logs:CreateLogStream"
            ],
            "Resource": "*"
        },
```

```
        {
            "Effect": "Allow",
            "Action": [
                "cloudwatch:PutMetricData"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:ListBucket",
                "s3:PutObject",
                "s3:GetObject"
            ],
            "Resource": "*"
        }
    ]
}
```

应用程序角色信任策略

SimSpace Weaver 将信任关系作为信任策略添加到模拟应用程序角色中。 SimSpace Weaver 为每个模拟创建信任策略，类似于以下示例。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "simspaceweaver.amazonaws.com"
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "ArnLike": {
          "aws:SourceArn":
     "arn:aws:simspaceweaver:us-west-2:111122223333:simulation/MySimName*"
        }
      }
    }
  ]
}
```

> **Note**
>
> 在此示例中，账号为 111122223333，模拟名称为 MySimName。这些值在您的信任策略中有所不同。

防止跨服务混淆代理

混淆代理问题是一个安全问题，即没有执行操作权限的实体可能会诱使更具权限的实体执行该操作。在中 AWS，跨服务模仿可能会导致混乱的副手问题。一个服务（呼叫服务）调用另一项服务（所谓的服务)时，可能会发生跨服务模拟。可以操纵调用服务以使用其权限对另一个客户的资源进行操作，否则该服务不应有访问权限。为了防止这种情况， AWS 提供可帮助您保护所有服务的服务委托人数据的工具，这些服务委托人有权限访问账户中的资源。

我们建议在资源策略中使用aws:SourceArn和aws:SourceAccount全局条件上下文密钥来限制为资源 AWS SimSpace Weaver 提供其他服务的权限。如果 aws:SourceArn 值不包含账户 ID，例如 Amazon S3 存储桶 Amazon 资源名称（ARN），您必须使用两个全局条件上下文键来限制权限。如果同时使用全局条件上下文密钥和包含账户 ID 的 aws:SourceArn 值，则 aws:SourceAccount 值和 aws:SourceArn 值中的账户在同一策略语句中使用时，必须使用相同的账户 ID。如果您只希望将一个资源与跨服务访问相关联，请使用 aws:SourceArn。如果您想允许该账户中的任何资源与跨服务使用操作相关联，请使用。aws:SourceAccount

aws:SourceArn 的值必须使用扩展的 ARN。

防范混淆代理问题最有效的方法是使用 aws:SourceArn 全局条件上下文键和资源的完整 ARN。如果不知道扩展的完整 ARN，或者正在指定多个扩展，请针对 ARN 未知部分使用带有通配符（*）的 aws:SourceArn 全局上下文条件键。例如 arn:aws:*simspaceweaver*:*:*111122223333*:*。

以下示例显示了如何在中使用aws:SourceArn和aws:SourceAccount全局条件上下文键 SimSpace Weaver 来防止出现混淆的副手问题。只有当请求来自指定源账户并提供指定的 ARN 时，此策略才允许 SimSpace Weaver 代入该角色。在这种情况下， SimSpace Weaver 只能在请求者自己的账户 (*111122223333*) 中为来自模拟的请求担任角色，并且只能在指定的区域 (*us-west-2*) 中扮演角色。

JSON

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

```
    {
      "Effect": "Allow",
      "Principal": {
        "Service": [
          "simspaceweaver.amazonaws.com"
        ]
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "StringEquals": {
          "aws:SourceAccount": "111122223333"
        },
        "ArnLike": {
          "aws:SourceArn": "arn:aws:simspaceweaver:us-
west-2:111122223333:simulation/*"
        }
      }
    }
  ]
}
```

编写此策略的一种更安全的方法是，将模拟名称包含在 aws:SourceArn 中，如以下示例所示，这会将策略限制为名为 MyProjectSimulation_22-10-04_22_10_15 的模拟：

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": [
          "simspaceweaver.amazonaws.com"
        ]
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "StringEquals": {
          "aws:SourceAccount": "111122223333"
        },
```

```
        "StringLike": {
          "aws:SourceArn": "arn:aws:simspaceweaver:us-
west-2:111122223333:simulation/MySimulation"
        }
      }
    }
  ]
}
```

如果您的 aws:SourceArn 明确包含账号，则可以省略 aws:SourceAccount 的 Condition 元素测试（有关更多信息，请参阅《IAM 用户指南》），例如，在以下简化政策中：

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": [
          "simspaceweaver.amazonaws.com"
        ]
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "StringLike": {
          "aws:SourceArn": "arn:aws:simspaceweaver:us-
west-2:111122223333:simulation/MySimulation"
        }
      }
    }
  ]
}
```

对 AWS SimSpace Weaver 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 SimSpace Weaver 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 SimSpace Weaver
- 我无权执行 iam：PassRole
- 我想要查看我的访问密钥
- 我是一名管理员，想允许其他人访问 SimSpace Weaver
- 我想允许我以外的人 AWS 账户 访问我的 SimSpace Weaver 资源

我无权在以下位置执行操作 SimSpace Weaver

如果 AWS 管理控制台 告诉您您无权执行某项操作，则必须联系管理员寻求帮助。管理员是指提供用户名和密码的人员。

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 `simspaceweaver:`*`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:
 simspaceweaver:GetWidget on resource: my-example-widget
```

在这种情况下，Mateo 请求他的管理员更新其策略，以允许他使用 `simspaceweaver:`*`GetWidget`* 操作访问 *`my-example-widget`* 资源。

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。 SimSpace Weaver

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 SimSpace Weaver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。

我想要查看我的访问密钥

在创建 IAM 用户访问密钥后，您可以随时查看您的访问密钥 ID。但是，您无法再查看您的秘密访问密钥。如果您丢失了私有密钥，则必须创建一个新的访问密钥对。

访问密钥包含两部分：访问密钥 ID（例如 `AKIAIOSFODNN7EXAMPLE`）和秘密访问密钥（例如 `wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY`）。与用户名和密码一样，您必须同时使用访问密钥 ID 和秘密访问密钥对请求执行身份验证。像对用户名和密码一样，安全地管理访问密钥。

> **Important**
>
> 请不要向第三方提供访问密钥，即便是为了帮助找到您的规范用户 ID 也不行。通过这样做，您可以授予他人永久访问您的权限 AWS 账户。

当您创建访问密钥对时，系统会提示您将访问密钥 ID 和秘密访问密钥保存在一个安全位置。秘密访问密钥仅在您创建它时可用。如果丢失了您的秘密访问密钥，您必须为 IAM 用户添加新的访问密钥。您最多可拥有两个访问密钥。如果您已有两个密钥，则必须删除一个密钥对，然后再创建新的密钥。要查看说明，请参阅 IAM 用户指南中的管理访问密钥。

我是一名管理员，想允许其他人访问 SimSpace Weaver

要允许其他人访问 SimSpace Weaver，您必须向需要访问的人员或应用程序授予权限。如果使用 AWS IAM Identity Center 管理人员和应用程序，则可以向用户或组分配权限集来定义其访问权限级别。权限集会自动创建 IAM 策略并将其分配给与人员或应用程序关联的 IAM 角色。有关更多信息，请参阅《AWS IAM Identity Center 用户指南》中的权限集。

如果未使用 IAM Identity Center，则必须为需要访问的人员或应用程序创建 IAM 实体（用户或角色）。然后，您必须将策略附加到实体，以便在 SimSpace Weaver中向其授予正确的权限。授予权限后，向用户或应用程序开发人员提供凭证。他们将使用这些凭证访问 AWS。要了解有关创建 IAM 用户、组、策略和权限的更多信息，请参阅《IAM 用户指南》中的 IAM 身份和 IAM 中的策略和权限。

我想允许我以外的人 AWS 账户 访问我的 SimSpace Weaver 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，可以代入角色。对于支持基于资源的策略或访问控制列表 (ACLs) 的服务，您可以使用这些策略向人们授予访问您的资源的权限。

要了解更多信息，请参阅以下内容：

- 要了解是否 SimSpace Weaver 支持这些功能，请参阅如何 AWS SimSpace Weaver 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访问之间的差别，请参阅《IAM 用户指南》中的 IAM 中的跨账户资源访问。

中的安全事件记录和监控 AWS SimSpace Weaver

监控是维护 AWS 解决方案的可靠性、可用性和性能的重要组成部分。 SimSpace Weaver 您应该从 AWS 解决方案的所有部分收集监控数据，以便在出现多点故障时可以更轻松地对其进行调试。

AWS 并 SimSpace Weaver 提供了多种用于监控您的仿真资源和应对潜在事件的工具。

登录 Amazon CloudWatch

SimSpace Weaver 将其日志存储在中 CloudWatch。您可以使用这些日志来监控模拟中的事件（如启动和停止应用程序）以及进行调试。有关更多信息，请参阅 SimSpace Weaver 在 Amazon CloudWatch 日志中登录。

亚马逊 CloudWatch 警报

使用 Amazon CloudWatch 警报，您可以监控您指定的时间段内的单个指标。如果该指标超过给定的阈值，则会向 Amazon SNS 主题或 A AWS uto Scaling 策略发送通知。 CloudWatch 警报在状态发生变化时触发，并在指定的时间段内维护，而不是通过处于特定状态来维持。有关更多信息，请参阅 SimSpace Weaver 使用 Amazon 进行监控 CloudWatch。

AWS CloudTrail 日志

CloudTrail 提供了用户、角色或 AWS 服务在中执行的操作的记录 SimSpace Weaver。使用收集的信息 CloudTrail，您可以确定向哪个请求发出 SimSpace Weaver、发出请求的 IP 地址、谁发出了请求、何时发出请求以及其他详细信息。有关更多信息，请参阅 使用记录 AWS SimSpace Weaver API 调用 AWS CloudTrail。

的合规性验证 AWS SimSpace Weaver

SimSpace Weaver 不在任何 AWS 合规计划的范围内。

作为多个合规计划的一部分，第三方审计师评估其他 AWS 服务的安全 AWS 性和合规性。其中包括 SOC、PCI、FedRAMP、HIPAA 及其他。

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。有关您在使用时的合规责任的更多信息 AWS 服务，请参阅AWS 安全文档。

韧性在 AWS SimSpace Weaver

AWS 全球基础设施是围绕 AWS 区域 可用区构建的。 AWS 区域 提供多个物理分隔和隔离的可用区，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、容错能力和可扩展性。

有关 AWS 区域 和可用区的更多信息，请参阅AWS 全球基础设施。

中的基础设施安全 AWS SimSpace Weaver

作为一项托管服务 AWS SimSpace Weaver ，受 AWS 全球网络安全的保护。有关 AWS 安全服务以及如何 AWS 保护基础设施的信息，请参阅AWS 云安全。要使用基础设施安全的最佳实践来设计您的 AWS 环境，请参阅 S AWS ecurity Pillar Well-Architected Fram ework 中的基础设施保护。

您可以使用 AWS 已发布的 API 调用 SimSpace Weaver 通过网络进行访问。客户端必须支持以下内容：

- 传输层安全性协议（TLS）。我们要求使用 TLS 1.2，建议使用 TLS 1.3。
- 具有完全向前保密（PFS）的密码套件，例如 DHE（临时 Diffie-Hellman）或 ECDHE（临时椭圆曲线 Diffie-Hellman）。大多数现代系统（如 Java 7 及更高版本）都支持这些模式。

网络连接安全模型

您的模拟在位于您选择的 AWS 区域内的 Amazon VPC 内的计算实例上运行。Amazon VPC 是 AWS 云中的虚拟网络，它按工作负载或组织实体隔离基础设施。Amazon VPC 内计算实例之间的通信保持在 AWS 网络内，不会通过互联网传输。一些内部服务通信会通过互联网进行并经过加密。在同一 AWS 地区运行的所有客户的模拟共享相同的 Amazon VPC。在同一 Amazon VPC 中，不同客户的模拟使用不同的计算实例。

您的仿真客户端与在 SimSpace Weaver 旅行中运行的仿真之间通过互联网进行通信。 SimSpace Weaver 不处理这些连接。您负责保护您的客户端连接。

您与该 SimSpace Weaver 服务的连接是通过互联网进行的，并且已被加密。这包括使用 AWS 管理控制台、 AWS Command Line Interface (AWS CLI)、 AWS 软件开发套件 (SDK) 和 SimSpace Weaver 应用程序 SDK 进行连接。

中的配置和漏洞分析 AWS SimSpace Weaver

配置和 IT 控制是您 AWS 和您的共同责任。有关更多信息，请参阅责任 AWS 共担模型共。 AWS 处理底层基础设施的基本安全任务，例如在计算实例上修补操作系统、防火墙配置和 AWS 基础设施灾难恢复。这些流程已通过相应第三方审核和认证。有关更多详细信息，请参阅安全性、身份和合规性最佳实践。

您负责模拟软件的安全性：

- 维护您的应用程序代码，包括更新和安全补丁。
- 对您的模拟客户端以及与其连接的应用程序之间的通信进行身份验证和加密。
- 更新您的模拟以使用最新的 SDK 版本，包括 S AWS DK 和 SimSpace Weaver 应用程序 SDK。

> **Note**
>
> SimSpace Weaver 不支持在运行模拟中更新应用程序。如果您需要更新应用程序，则必须停止并删除模拟，然后使用更新的应用程序代码创建新的模拟。我们建议您将模拟状态保存在外部数据存储中，以便在需要重新创建模拟时进行恢复。

以下方面的安全最佳实践 SimSpace Weaver

本节介绍特定于的安全最佳实践 SimSpace Weaver。要详细了解中的安全最佳实践 AWS，请参阅安全、身份和合规性最佳实践。

主题

您的应用程序与其客户端之间的加密通信

SimSpace Weaver 不管理您的应用程序与其客户端之间的通信。您应该为客户端会话实现某种形式的身份验证和加密。

定期备份模拟状态

SimSpace Weaver 不会保存您的模拟状态。停止的模拟（由于 API 调用、控制台选项或系统崩溃导致）不会保存其状态，也没有固有的恢复方法。停止的模拟无法重新启动。执行等同于重新启动操作的唯一方法是使用相同的配置和数据重新创建模拟。您可以使用模拟状态的备份来初始化新的模拟。AWS 提供高度可靠且可用的云存储和数据库服务，可用于保存模拟状态。

维护您的应用程序和 SDKs

维护您的应用程序、本地安装的 AWS 软件开发套件 (SDKs) 和 SimSpace Weaver 应用程序 SDK。您可以下载并安装新版本的 AWS SDKs。使用非生产 SimSpace Weaver 应用程序版本测试新版本的应用程序 SDK，以确保您的应用程序继续按预期运行。您无法在运行的模拟中更新应用程序。要更新您的应用程序，请执行以下操作：

1. 在本地（或在测试环境中）更新和测试应用程序代码。
2. 停止更改模拟状态并保存（如有必要）。
3. 停止模拟（一旦停止，将无法重新启动）。
4. 删除模拟（已停止但未删除的模拟仍会计入您的服务限额）。
5. 使用相同的配置和更新的应用程序代码重新创建模拟。
6. 使用保存的状态数据（如果有）初始化模拟。
7. 启动新的模拟。

> **Note**
>
> 使用相同配置创建的新模拟与旧模拟是分开的。它将有一个新的模拟 ID，并将日志发送到 Amazon 中的新日志流 CloudWatch。

登录和监控 SimSpace Weaver

监控是维护和其他 AWS 解决方案的可靠性、可用性和性能的重要组成部分。 SimSpace Weaver AWS 提供以下监控工具 SimSpace Weaver，供您监视、报告问题并在适当时自动采取措施：

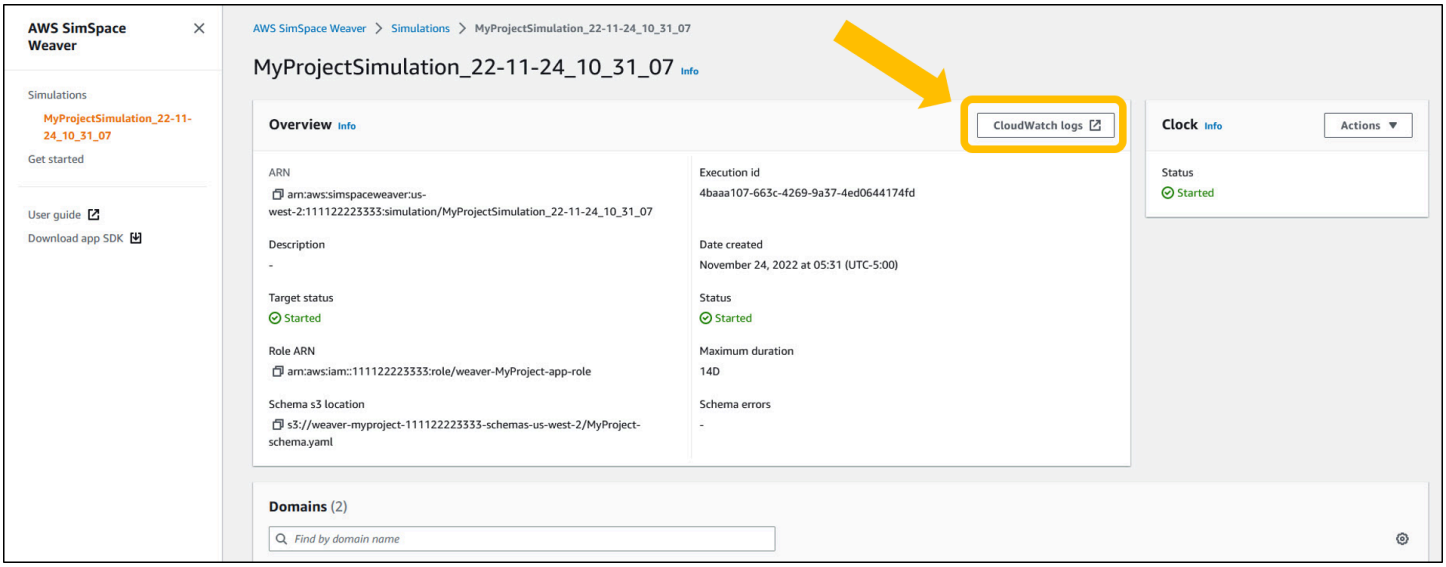
- Amazon 会实时 CloudWatch监控您的 AWS 资源和您运行 AWS 的应用程序。您可以收集和跟踪指标，创建自定义的控制面板，以及设置警报以在指定的指标达到您指定的阈值时通知您或采取措施。有关更多信息，请参阅 Amazon CloudWatch 用户指南。
- Amazon Lo CloudWatch gs 使您能够监控、存储和访问来自 SimSpace Weaver 工作人员和其他来源的日志数据。 CloudTrail CloudWatch 日志可以监控日志数据中的信息，并在达到特定阈值时通知您。您还可以在高持久性存储中检索您的日志数据。有关更多信息，请参阅 Amazon CloudWatch 日志用户指南。
- 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 Amazon S3 桶。您可以标识哪些用户和账户调用了 AWS、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

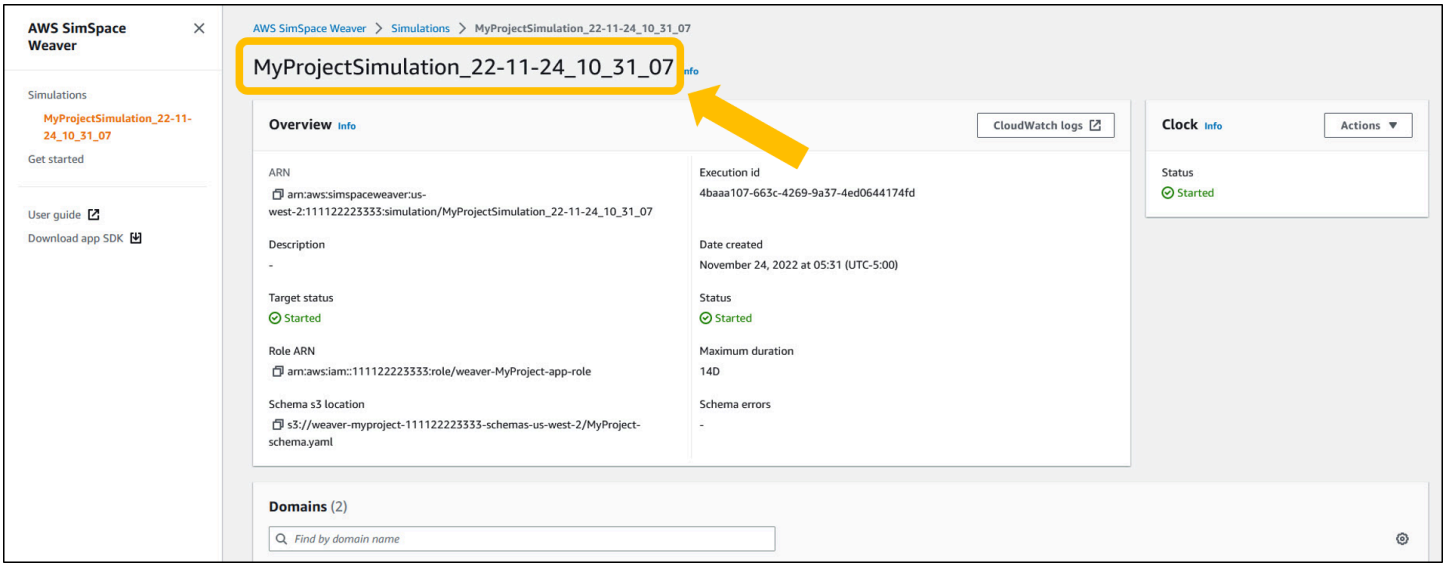
SimSpace Weaver 在 Amazon CloudWatch 日志中登录

访问您的 SimSpace Weaver 日志

您的 SimSpace Weaver 模拟生成的所有日志都存储在 Amazon CloudWatch 日志中。要访问日志，您可以使用 SimSpace Weaver 控制台中模拟概述窗格中的CloudWatch 日志按钮，该按钮将直接带您进入该特定模拟的日志。



您也可以通过 CloudWatch 控制台访问日志。您需要模拟的名称才能搜索其日志。



SimSpace Weaver 日志

SimSpace Weaver 将模拟管理消息和应用程序的控制台输出写入 Amazon CloudWatch Logs。有关使用日志的更多信息，请参阅 Amazon Logs 用户指南中的使用日志组和 CloudWatch 日志流。

您创建的每个模拟在 Log CloudWatch s 中都有自己的日志组。日志组的名称在模拟架构中指定。在以下架构片段中，`log_destination_service` 的值为 `logs`。这意味着 `log_destination_resource_name` 的值是日志组的名称。在本例中，日志组是 `MySimulationLogs`。

```
simulation_properties:
  log_destination_service: "logs"
  log_destination_resource_name: "MySimulationLogs"
  default_entity_index_key_type: "Vector3<f32>"
```

在启动模拟日志组后，您还可以使用 DescribeSimulation API 来查找该日志组的名称。

```
aws simspaceweaver describe-simulation --simulation simulation-name
```

以下示例显示了 DescribeSimulation 的输出中描述日志配置的部分。日志组的名称显示在 LogGroupArn 的末尾。

```
    "LoggingConfiguration": {
        "Destinations": [
            {
                "CloudWatchLogsLogGroup": {
                    "LogGroupArn": "arn:aws:logs:us-west-2:111122223333:log-
group:MySimulationLogs"
                }
            }
        ]
    },
```

每个模拟日志组都包含多个日志流：

- 管理日志流- SimSpace Weaver 服务生成的模拟管理消息。

  ```
  /sim/management
  ```

- 错误日志流- SimSpace Weaver 服务生成的错误消息。只有在出现错误时，此日志流才会存在。SimSpace Weaver 将您的应用程序写入的错误存储在它们自己的应用程序日志流中（参见以下日志流）。

  ```
  /sim/errors
  ```

- 空间应用程序日志流（每个工作线程上的每个空间应用程序 1 个）– 空间应用程序生成的控制台输出。每个空间应用程序都会将日志写入其日志流。*spatial-app-id* 是 *worker-id* 末尾斜杠后的全部字符。

  ```
  /domain/spatial-domain-name/app/worker-worker-id/spatial-app-id
  ```

- 自定义应用程序日志流（每个自定义应用程序实例 1 个）– 自定义应用程序生成的控制台输出。每个自定义应用程序实例都会将日志写入其日志流。

  ```
  /domain/custom-domain-name/app/custom-app-name/random-id
  ```

- 服务应用程序日志流（每个服务应用程序实例 1 个）– 服务应用程序生成的控制台输出。每个服务应用程序都会将日志写入其日志流。*service-app-id* 是 *service-app-name* 末尾斜杠后的全部字符。

  ```
  /domain/service-domain-name/app/service-app-name/service-app-id
  ```

SimSpace Weaver 使用 Amazon 进行监控 CloudWatch

您可以 SimSpace Weaver 使用 Amazon 进行监控 CloudWatch，Amazon 会收集原始数据并将其处理为可读的近乎实时的指标。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。

该 SimSpace Weaver 服务在`AWS/simspaceweaver`命名空间中报告以下指标。

SimSpace Weaver 账户层面的指标

SimSpace Weaver 命名空间包括以下与 AWS 账户级别的活动相关的指标。

指标	说明
`SimulationCount`	当前账户的模拟次数。 单位：计数 维度：无 统计数据：Average、Minimum、Maximum

使用记录 AWS SimSpace Weaver API 调用 AWS CloudTrail

AWS SimSpace Weaver 与 AWS CloudTrail一项服务集成，该服务提供用户、角色或 AWS 服务在中执行的操作的记录 SimSpace Weaver。 CloudTrail 将所有 API 调用捕获 SimSpace Weaver 为事件。捕获的调用包括来自 SimSpace Weaver 控制台的调用和对 SimSpace Weaver API 操作的代码调用。如果您创建了跟踪，则可以允许将 CloudTrail 事件持续传输到 Amazon S3 存储桶，包括的事件 SimSpace Weaver。如果您未配置跟踪，您仍然可以在 CloudTrail 控制台中查看最新事件Event history。使用收集的信息 CloudTrail，您可以确定向哪个请求发出 SimSpace Weaver、发出请求的 IP 地址、谁发出了请求、何时发出请求以及其他详细信息。

要了解更多信息 CloudTrail，请参阅《AWS CloudTrail 用户指南》。

SimSpace Weaver 信息在 CloudTrail

CloudTrail 在您创建账户 AWS 账户 时已在您的账户上启用。当活动发生在中时 SimSpace Weaver，该活动会与其他 AWS 服务 CloudTrail 事件一起记录在事件中Event history。您可以在中查看、搜索和下载最近发生的事件 AWS 账户。有关更多信息，请参阅使用事件历史记录查看 CloudTrail 事件。

要持续记录您的 AWS 账户事件（包括的事件） SimSpace Weaver，请创建跟踪。跟踪允许 CloudTrail 将日志文件传输到 Amazon S3 存储桶。默认情况下，在控制台中创建跟踪记录时，此跟踪记录应用于所有 AWS 区域。跟踪记录 AWS 分区中所有区域的事件，并将日志文件传送到您指定的 Amazon S3 存储桶。此外，您可以配置其他 AWS 服务，以进一步分析和处理 CloudTrail 日志中收集的事件数据。有关更多信息，请参阅下列内容：

- 创建跟踪记录概述
- CloudTrail 支持的服务和集成
- 配置 Amazon SNS 通知 CloudTrail
- 接收来自多个地区的 CloudTrail 日志文件和接收来自多个账户的 CloudTrail 日志文件

所有 SimSpace Weaver 操作均由《API 参考》记录 CloudTrail 并记录在《AWS SimSpace Weaver API 参考》中。例如，调用`DescribeSimulation`和`DeleteSimulation`操作会在 CloudTrail 日志文件中生成条目。`ListSimulations`

每个事件或日志条目都包含有关生成请求的人员信息。身份信息有助于您确定以下内容：

- 请求是使用根证书还是 AWS Identity and Access Management (IAM) 用户凭证发出。
- 请求是使用角色还是联合用户的临时安全凭证发出的。

- 请求是否由其他 AWS 服务发出。

有关更多信息，请参阅 CloudTrail userIdentity 元素。

了解 SimSpace Weaver 日志文件条目

跟踪是一种配置，允许将事件作为日志文件传输到您指定的 Amazon S3 存储桶。 CloudTrail 日志文件包含一个或多个日志条目。事件代表来自任何来源的单个请求，包括有关所请求操作的信息，例如操作的日期和时间、请求参数和其他详细信息。 CloudTrail 日志文件不是公共 API 调用的有序堆栈跟踪，因此它们不会按任何特定的顺序出现。

以下示例显示了演示该`ListSimulations`操作的 CloudTrail 日志条目。

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "AIDACKCEVSQ6C2EXAMPLE:aws-console-signin-utils",
        "arn": "arn:aws:sts::111122223333:assumed-role/ConsoleSigninRole/aws-console-
signin-utils",
        "accountId": "111122223333",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "AIDACKCEVSQ6C2EXAMPLE",
                "arn": "arn:aws:iam::111122223333:role/ConsoleSigninRole",
                "accountId": "111122223333",
                "userName": "ConsoleSigninRole"
            },
            "webIdFederationData": {},
            "attributes": {
                "creationDate": "2022-02-14T15:57:02Z",
                "mfaAuthenticated": "false"
            }
        }
    },
    "eventTime": "2022-02-14T15:57:08Z",
    "eventSource": "simspaceweaver.amazonaws.com",
    "eventName": "ListSimulations",
    "awsRegion": "us-west-2",
```

```
    "sourceIPAddress": "192.0.2.10",
    "userAgent": "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like
 Gecko) Chrome/86.0.4240.0 Safari/537.36",
    "requestParameters": null,
    "responseElements": null,
    "requestID": "1234abcd-1234-5678-abcd-12345abcd123",
    "eventID": "5678abcd-5678-1234-ab12-123abc123abc",
    "readOnly": true,
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333",
    "eventCategory": "Management"
}
```

SimSpace Weaver 端点和配额

下表描述了 SimSpace Weaver 的服务端点和服务配额。服务限额（也称为限制）是 AWS 账户使用的服务资源或操作的最大数量。有关更多信息，请参阅《AWS 一般参考》中的 AWS 服务限额。

服务端点

区域名称	区域	端点	协议
美国东部 (弗吉尼亚北部)	us-east-1	simspaceweaver.us-east-1.amazonaws.com	HTTPS
美国东部（俄亥俄州）	us-east-2	simspaceweaver.us-east-2.amazonaws.com	HTTPS
美国西部（俄勒冈州）	us-west-2	simspaceweaver.us-west-2.amazonaws.com	HTTPS
亚太地区（新加坡）	ap-southeast-1	simspaceweaver.ap-southeast-1.amazonaws.com	HTTPS
亚太地区（悉尼）	ap-southeast-2	simspaceweaver.ap-southeast-2.amazonaws.com	HTTPS
欧洲地区（斯德哥尔摩）	eu-north-1	simspaceweaver.eu-north-1.amazonaws.com	HTTPS
欧洲地区（法兰克福）	eu-central-1	simspaceweaver.eu-central-1.amazonaws.com	HTTPS

区域名称	区域	端点	协议
欧洲地区（爱尔兰）	eu-west-1	simspaceweaver.eu-west-1.amazonaws.com	HTTPS
AWS GovCloud（美国东部）	us-gov-east-1	simspaceweaver.us-gov-east-1.amazonaws.com	HTTPS
AWS GovCloud（美国西部）	us-gov-west-1	simspaceweaver.us-gov-west-1.amazonaws.com	HTTPS

服务限额

Name	默认值	可调整	描述
每个应用程序的计算资源单位数	每个受支持的区域：4 个	否	可分配给每个应用程序的计算资源单位的最大数量。
每个工作线程可用的计算资源单位数	每个受支持的区域：17 个	否	每个工作线程可用的计算资源单位数。
每个实体的数据字段数	每个受支持的区域：7 个	否	每个实体可拥有的数据（非索引）字段的最大数量。
分区中的实体	每个受支持的区域：8,192 个	否	每个分区的最大实体数。
实体数据字段大小	每个受支持的区域：1024 字节	否	实体数据（非索引）字段的最大大小。

Name	默认值	可调整	描述
工作线程之间的实体转移	每个受支持的区域：25 个	否	对于每个分区和每个刻度，工作线程之间的实体转移的最大数量。
同一个工作线程的实体转移	每个受支持的区域：500 个	否	对于每个分区和每个刻度，同一个工作线程的实体转移的最大数量。
每个实体的索引字段数	每个受支持的区域：1 个	否	每个实体可拥有的索引字段的最大数量。
模拟的最大最长持续时间（以天为单位）	每个受支持的区域：14 个	否	您可以指定为模拟的最长持续时间的最大天数。所有的模拟都有一个最大持续时间，即使您没有指定数值。模拟达到最大持续时间后自动停止。
每个计算资源单位的内存量	每个受支持的区域：1 GB	否	对于每个计算资源单位，应用程序获取的随机访问内存（RAM）量。
每个工作线程可用的远程订阅数	每个受支持的区域：24 个	否	每个工作线程可用的远程订阅的最大数量。
模拟计数	每个受支持的区域：2 个	是	账户中目标状态为“已开始”的模拟最大次数。您可以请求将限额提高到 10。
执行模拟作业的工作线程	每个受支持的区域：2 个	是	可为每个模拟分配的工作人员的最大数量。您可以请求将限额提高到 10。

Name	默认值	可调整	描述
v CPUs 代表每个计算资源单元	每个受支持的区域：2 个	否	应用程序为每个计算资源单元获得的虚拟中央处理单元 (vCPUs) 的数量。

消息配额

以下配额适用于SimSpace Weaver Local和中的应用程序到应用程序的消息传递 AWS Cloud。

Name	默认值	可调整	描述
最大消息大小	每个支持的区域：256 字节	否	消息负载的最大大小。
最大消息发送速率	每个受支持的区域：128 个	否	每个应用程序每次报价可以发送的最大消息数。

时钟率

模拟架构指定了模拟的时钟率（也称为刻度率）。下表指定了您可以使用的有效时钟率。

Name	有效值	描述
时钟率	每个支持的区域：“10”、“15”、“30”、“无限制”	模拟的有效时钟率。
时钟频率（版本 1.13 和 1.12）	每个支持的区域：10、15、30	模拟的有效时钟率。

SimSpace Weaver Local 的服务限额

以下服务限额仅适用于 SimSpace Weaver Local。所有其他限额也仅适用于 SimSpace Weaver Local。

Name	默认值	可调整	描述
最大分区数	SimSpace Weaver Local：24	否	模拟的最大分区数。
最大应用程序数	SimSpace Weaver Local：24	否	模拟的最大应用程序总数（任何类型）。
最大域数	SimSpace Weaver Local：24	否	模拟的最大域总数（任何类型）。
每个分区的实体数	SimSpace Weaver Local：4096	否	每个分区的最大实体数。
每个实体的字段数	SimSpace Weaver Local：8	否	每个实体的最大字段数。
字段大小	SimSpace Weaver Local：1024 字节	否	实体字段的最大大小。

中的疑难解答 SimSpace Weaver

主题

AssumeRoleAccessDenied

如果您的模拟无法启动，可能会收到以下错误：

```
Unable to assume role arn:aws:iam::111122223333:role/weaver-project-name-app-role;
 verify the role exists and has trust policy on SimSpace Weaver
```

如果您的模拟的 AWS Identity and Access Management (IAM) 角色存在以下任一情况，则可能会收到此错误：

- Amazon 资源名称 (ARN) 是指不存在的 IAM 角色。
- IAM 角色的信任策略，该策略不允许使用此名称的新模拟担任此角色。

检查以确保该角色存在。如果该角色存在，请查看角色的信任策略。以下示例信任策略中的 `aws:SourceArn` 仅允许名称以 `MySimulation` 开头的模拟（账户 111122223333 中）代入该角色。

JSON

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

```
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "simspaceweaver.amazonaws.com"
            },
            "Action": "sts:AssumeRole",
            "Condition": {
                "ArnLike": {
                    "aws:SourceArn": "arn:aws:simspaceweaver:us-
west-2:111122223333:simulation/MySimulation*"
                }
            }
        }
    ]
}
```

要允许名称以 MyOtherSimulation 开头的另一个模拟担任该角色，必须修改信任策略，如以下编辑后的示例所示：

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "simspaceweaver.amazonaws.com"
            },
            "Action": "sts:AssumeRole",
            "Condition": {
                "ArnLike": {
                    "aws:SourceArn": [
                        "arn:aws:simspaceweaver:us-
west-2:111122223333:simulation/MySimulation*",
                        "arn:aws:simspaceweaver:us-
west-2:111122223333:simulation/MyOtherSimulation*"
                        ]
                }
            }
```

```
        }
    ]
}
```

InvalidBucketName

创建项目时，您可能会收到以下错误：

```
An error occurred (InvalidBucketName) when calling the CreateBucket operation: The
 specified bucket is not valid.
```

您之所以收到此错误，是因为 SimSpace Weaver 传递给亚马逊简单存储服务 (Amazon S3) 的名称违反了存储桶命名规则（有关更多信息，请参阅亚马逊简单存储服务用户指南中的存储桶命名规则）。

SimSpace Weaver 应用程序 SDK 中的`create-project`脚本使用您为脚本提供的项目名称创建存储桶名称。存储桶名称采用以下格式。

- 版本 1.13.x 或更高版本
  - `weaver-`*`lowercase-project-name`*`-`*`account-number`*`-`*`region`*
- 版本 1.12.x
  - `weaver-`*`lowercase-project-name`*`-`*`account-number`*`-app-zips-`*`region`*
  - `weaver-`*`lowercase-project-name`*`-`*`account-number`*`-schemas-`*`region`*

例如，指定以下项目属性：

- 项目名称：`MyProject`
- AWS 账户 数字:`111122223333`
- AWS 区域: `us-west-2`

项目可能包含以下存储桶：

- 版本 1.13.x 或更高版本
  - `weaver-myproject-111122223333-us-west-2`
- 版本 1.12.x

- `weaver-myproject-111122223333-app-zips-us-west-2`
- `weaver-myproject-111122223333-schemas-us-west-2`

您的项目名称不得违反 Amazon S3 命名规则。您还必须使用足够短的项目名称，以便 `create-project` 脚本创建的存储桶名称不会超过 Amazon S3 存储桶的名称长度限制。

ServiceQuotaExceededException

当您启动模拟时，可能会收到以下错误：

```
An error occurred (ServiceQuotaExceededException) when calling the StartSimulation
 operation: Failed to start simulation due to: simulation quota has already been
 reached.
```

如果您尝试启动新的模拟，但您的账户中目标状态为“已启动”的当前模拟数量已达到上限，则会收到此错误。这些模拟包括正在运行的模拟、失败的模拟以及由于达到最长运行时间而被停止的模拟。您可以删除已停止或失败的模拟，以便能够启动新的模拟。如果所有模拟都在运行，则可以停止和删除一个正在运行的模拟。您也可以请求提高服务限额（如果尚未达到请求限制）。

TooManyBuckets

创建项目时，您可能会收到以下错误：

```
An error occurred (TooManyBuckets) when calling the CreateBucket operation: You have
 attempted to create more buckets than allowed.
```

亚马逊简单存储服务 (Amazon S3) Simple Service 对 AWS 您的账户中可以拥有的存储桶数量有限制（有关更多信息，请参阅亚马逊简单存储服务用户指南中的存储桶限制和限制）。

要继续，您必须执行以下操作之一：

- 删除 2 个或更多不需要的现有 Amazon S3 存储桶。
- 请求提高 Amazon S3 限制（有关更多信息，请参阅《Amazon Simple Storage Service 用户指南》中的存储桶限制。）
- 使用其他 AWS 账户。

> **Note**
>
> 中的 `DeleteSimulation` API SimSpace Weaver 不会删除与您的模拟关联的 Amazon S3 资源。如果不再需要，我们建议您删除与您的模拟相关联的所有资源。

启动模拟时权限被拒绝

当您启动模拟时，可能会收到一条错误消息，指出权限被拒绝或访问您的应用程序构件时出错。如果您为模拟指定 Amazon S3 存储桶，但该存储桶 SimSpace Weaver 不是为您创建的（通过控制台或 SimSpace Weaver 应用程序 SDK 脚本），则可能会出现此问题。

最可能的根本原因如下：

- 该服务无权访问您在模拟架构中指定的一个或多个 Amazon S3 存储桶 – 请检查您的应用程序角色权限策略、Amazon S3 存储桶策略和 Amazon S3 存储桶权限，确保 `simspaceweaver.amazonaws.com` 具有访问您的存储桶的正确权限。有关应用程序角色权限策略的更多信息，请参阅SimSpace Weaver 为您创建的权限。
- 您的 Amazon S3 存储桶可能与您的模拟 AWS 区域 不同 — 您的模拟项目的 Amazon S3 存储桶必须与您的模拟 AWS 区域 相同。请查看您的 Amazon S3 控制台以查看 AWS 区域 您的存储桶在哪里。如果您的 Amazon S3 存储桶位于其他存储桶中 AWS 区域，请选择与您的模拟 AWS 区域 相同的存储桶。

使用 Docker 时与时间相关的问题

如果您正在使用，Docker并且在运行来自 SimSpace Weaver 应用程序 SDK 的脚本时收到与时间相关的错误，则原因可能是您的Docker虚拟机时钟不正确。如果您的计算机此前正在运行 Docker，之后从睡眠或休眠状态恢复，则可能会发生这种情况。

可尝试的解决方案

- 重新启动 Docker。
- 在 Windows PowerShell 中禁用并重新启用时间同步：

```
Get-VMIntegrationService -VMName DockerDesktopVM -Name "Time Synchronization" |
 Disable-VMIntegrationService
Get-VMIntegrationService -VMName DockerDesktopVM -Name "Time Synchronization" |
 Enable-VMIntegrationService
```

PathfindingSample 控制台客户端无法连接

当您连接到中教程中所述的PathfindingSample模拟时，您可能会从控制台客户端收到以下错误入门 SimSpace Weaver。之所以出现此错误，是因为客户端无法通过您提供的 IP 地址和端口号组合打开与 ViewApp 的网络连接。

```
Fatal error in function nng_dial. Error code: 268435577. Error message: no link
```

要在中进行模拟 AWS Cloud

- 您的网络连接能否正常工作？ 确认您可以连接到其他应该正常运行的 IP 地址或网站。确保您的网络浏览器不是从其缓存中加载的网站。
- 您的模拟在运行吗？ 您可以使用 ListSimulationsAPI 来获取仿真状态。有关更多信息，请参阅 获取定制化应用程序的 IP 地址和端口号。您也可以使用 SimSpace Weaver 控制台来查看模拟的状态。
- 您的应用程序在运行吗？ 您可以使用 DescribeAppAPI 来获取应用程序的状态。有关更多信息，请参阅 获取定制化应用程序的 IP 地址和端口号。您也可以使用 SimSpace Weaver 控制台来查看模拟的状态。
- 您的应用程序在运行吗？ 您可以使用 DescribeAppAPI 来获取应用程序的状态。有关更多信息，请参阅 获取定制化应用程序的 IP 地址和端口号。您也可以使用 SimSpace Weaver 控制台来查看模拟的状态。
- 您使用的 IP 地址和端口号是否正确？ 通过互联网连接时，您必须使用 ViewApp 的 IP 地址和 Actual 端口号。您可以在 DescribeAppAPI 输出EndpointInfo区块中找到 IP Address和Actual端口号。您也可以使用 SimSpace Weaver 控制台在 MyViewDomain 详细信息页面中查找 ViewApp 的 IP 地址 (URI) 和端口号（入口端口）。
- 您的网络连接是否有防火墙保护？ 防火墙可能会阻止您与 IP 地址或端口号（或两者）的连接。请检查您的防火墙设置或咨询您的防火墙管理员。

对于本地模拟

- 您可以连接到环回地址 (127.0.0.1) 吗？ 如果您在 Windows 中使用 ping 命令行工具，则可以打开命令提示符窗口并尝试 Ping 127.0.0.1。按 Ctrl-C 以结束 Ping 操作。

```
ping 127.0.0.1
```

Example Ping 输出

```
C:\>ping 127.0.0.1

Pinging 127.0.0.1 with 32 bytes of data:
Reply from 127.0.0.1: bytes=32 time<1ms TTL=128
Reply from 127.0.0.1: bytes=32 time<1ms TTL=128
Reply from 127.0.0.1: bytes=32 time=1ms TTL=128

Ping statistics for 127.0.0.1:
    Packets: Sent = 3, Received = 3, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 1ms, Average = 0ms
Control-C
^C
C:\>
```

如果 Ping 显示有数据包丢失，则可能有其他软件（如本地防火墙、安全设置或反恶意软件程序）阻止了您的连接。

- 您的应用程序在运行吗？ 对于每个应用程序，您的本地模拟在单独的窗口中运行。确保空间应用程序和 ViewApp 的窗口处于打开状态。有关更多信息，请参阅 当地发展 SimSpace Weaver。
- 您使用的 IP 地址和端口号是否正确？ 在连接到本地模拟时必须使用 `tcp://127.0.0.1:7000`。有关更多信息，请参阅 当地发展 SimSpace Weaver。
- 您有可能阻止连接的本地安全软件吗？ 检查您的安全设置、本地防火墙或反恶意软件程序，看看它们是否阻止了您与 TCP 端口 `7000` 上 `127.0.0.1` 的连接。

他们认 AWS CLI 不出来 **`simspaceweaver`**

如果 AWS CLI 给出的错误提示它不知道 SimSpace Weaver，请运行以下命令。

```
aws simspaceweaver help
```

如果您收到以以下几行开头并列出所有可用选项的错误，则 AWS CLI 可能是旧版本。

```
usage: aws [options] <command> <subcommand> [<subcommand> ...] [parameters]
To see help text, you can run:

  aws help
```

```
  aws <command> help
  aws <command> <subcommand> help

aws: error: argument command: Invalid choice, valid choices are:
```

运行以下命令以检查您的版本 AWS CLI。

```
aws --version
```

如果版本低于 2.9.19，则必须更新您的 AWS CLI。请注意，的当前版本晚 AWS CLI 于 2.9.19。

要更新您的 AWS CLI，请参阅版本 2 AWS Command Line Interface 用户指南 AWS CLI中的安装或更新最新版本的。

SimSpace Weaver 仿真架构参考

SimSpace Weaver 使用 YAML 文件配置模拟的属性。此文件称为模拟架构（或简称架构）。SimSpace Weaver App SDK 中包含的示例模拟包括一个架构，您可以复制和编辑该架构以进行自己的模拟。

主题

完整架构示例

以下示例显示了 YAML描述 SimSpace Weaver 模拟的格式文本文件。此示例包括属性的虚拟值。文件格式因文件中指定的 `sdk_version` 值而异。有关这些属性及其有效值的完整说明，请参阅架构格式。

```
sdk_version: "1.17"
simulation_properties:
  log_destination_resource_name: "MySimulationLogs"
  log_destination_service: "logs"
  default_entity_index_key_type: "Vector3<f32>"
  default_image: "111122223333.dkr.ecr.us-west-2.amazonaws.com/my-ecr-
repository:latest"
workers:
  MyComputeWorkers:
    type: "sim.c5.24xlarge"
    desired: 3
clock:
  tick_rate: "30"
partitioning_strategies:
  MyGridPartitioning:
    topology: "Grid"
    aabb_bounds:
      x: [-1000, 1000]
      y: [-1000, 1000]
    grid_placement_groups:
      x: 3
      y: 3
domains:
```

```
  MyCustomDomain:
    launch_apps_via_start_app_call: {}
    app_config:
      package: "s3://weaver-myproject-111122223333-us-west-2/MyViewApp.zip"
      launch_command: ["MyViewApp"]
      required_resource_units:
        compute: 1
      endpoint_config:
        ingress_ports: [9000, 9001]
  MyServiceDomain:
    launch_apps_per_worker:
      count: 1
    app_config:
      package: "s3://weaver-myproject-111122223333-us-west-2/
MyConnectionServiceApp.zip"
      launch_command: ["MyConnectionServiceApp"]
      required_resource_units:
        compute: 1
      endpoint_config:
        ingress_ports:
          - 9000
          - 9001
  MySpatialDomain:
    launch_apps_by_partitioning_strategy:
      partitioning_strategy: "MyGridPartitioning"
      grid_partition:
        x: 6
        y: 6
    app_config:
      package: "s3://weaver-myproject-111122223333-us-west-2/MySpatialApp.zip"
      launch_command: ["MySpatialApp"]
      required_resource_units:
        compute: 1
  MySpatialDomainWithCustomContainer:
    launch_apps_by_partitioning_strategy:
      partitioning_strategy: "MyGridPartitioning"
      grid_partition:
        x: 6
        y: 6
    app_config:
      package: "s3://weaver-myproject-111122223333-us-west-2/MySpatialApp2.zip"
      launch_command: ["MySpatialApp2"]
      required_resource_units:
        compute: 1
```

```
      image: "111122223333.dkr.ecr.us-west-2.amazonaws.com/my-ecr-repository:latest"
placement_constraints:
  - placed_together: ["MySpatialDomain", "MySpatialDomainWithCustomContainer"]
    on_workers: ["MyComputeWorkers"]
```

架构格式

下面的示例显示架构的整体结构。只要父子关系相同，架构在每个级别的属性顺序就无关紧要。顺序对数组中的元素很重要。

```
sdk_version: "sdk-version-number"
simulation_properties:
  simulation-properties
workers:
  worker-group-configurations
clock:
  tick_rate: tick-rate
partitioning_strategies:
  partitioning-strategy-configurations
domains:
  domain-configurations
placement_constraints:
  placement-constraints-configuration
```

各个部分

- SDK 版本
- 模拟属性
- 工作线程
- 时钟
- 分区策略
- 域
- 放置约束

SDK 版本

`sdk_version`部分（必填）标识了支持此架构的 SimSpace Weaver 应用程序 SDK 的版本。有效值：`1.17`、`1.16`、`1.15`、`1.14`、`1.13`、`1.12`

> **Important**
>
> sdk_version 的值仅包括主版本号和第一个次要版本号。例如，值 1.12 指定所有版本 1.12.x，例如 1.12.0、1.12.1、和 1.12.2。

```
sdk_version: "1.17"
```

模拟属性

simulation_properties 部分（必需）指定模拟的各种属性。使用此部分可以配置日志记录并指定默认容器映像。即使您未配置日志记录或选择指定默认容器映像，也需要此部分。

```
simulation_properties:
  log_destination_resource_name: "log-destination-resource-name"
  log_destination_service: "log-destination-service"
  default_entity_index_key_type: "Vector3<f32>"
  default_image: "ecr-repository-uri"
```

属性

log_destination_resource_name

指定 SimSpace Weaver 将写入日志的资源。

必需：否。如果不包括此属性，则 SimSpace Weaver 不会为模拟写入日志。

类型：字符串

有效值：

- CloudWatch 日志日志组的名称（例如，MySimulationLogs）
- CloudWatch 日志组的 Amazon 资源名称 (ARN)（例如，）arn:aws:logs:us-west-2:111122223333:log-group/MySimulationLogs

> **Note**
>
> SimSpace Weaver 仅支持在同一个账户和 AWS 区域 模拟中使用日志目标。

log_destination_service

如果您指定不是 ARN 的 logging_destination resource_name，则表示日志目标资源的类型。

必需：如果指定了 log_destination_resource_name 且不是 ARN，则必须指定此属性。如果未指定 log_destination_resource_name，或者是 ARN，则无法指定此属性。

类型：字符串

有效值：

- logs：日志目标资源是一个日志组。

default_entity_index_key_type

为模拟实体的索引键字段指定数据类型。

必需：是

类型：字符串

有效值：Vector3<f32>

default_image

为您的模拟指定默认容器映像（版本 1.13 和 1.12 不支持）。如果指定了此属性，则未指定 image 的域将使用 default_image。

必需：否

类型：字符串

有效值：

- Amazon Elastic Container Registry (Amazon ECR) 中存储库的 URI（例如，111122223333.dkr.ecr.us-west-2.amazonaws.com/my-ecr-repository:latest）

工作线程

workers 部分（必需）指定工作线程组的配置。 SimSpace Weaver 将这些信息与 placement_constraints 一起用于配置模拟的底层基础架构。目前仅支持 1 个工作线程组。

要指定工作线程组的属性，请将 *worker-group-name* 替换为所选名称。该名称的长度必须为 3-64 个字符，可以包含以下字符：A -Z、a -z、0 -9 和 _ -（短横线）。在名称后指定工作线程组的属性。

```
workers:
  worker-group-name:
    type: "sim.c5.24xlarge"
    desired: number-of-workers
```

属性

type

指定工作线程类型。

必需：是

类型：字符串

有效值：sim.c5.24xlarge

desired

为该工作线程组指定所需的工作线程数量。

必需：是

类型：整数

有效值：1-3。您的模拟工作线程数量的服务限额（限制）决定此属性的最大值。例如，如果您的服务限额为 2，则此属性的最大值为 2。您也可以请求增加服务限额。有关更多信息，请参阅 SimSpace Weaver 端点和配额。

时钟

clock 部分（必需）指定模拟时钟的属性。

```
clock:
  tick_rate: tick-rate
```

属性

`tick_rate`

指定时钟每秒向应用程序发布的刻度数量。

必需：是

类型：

- 版本 *1.14* 和 *1.15*：字符串
- 版本 *1.13* 和 *1.12*：整数

有效值：

- 版本 *1.14* 和 *1.15*："10" | "15" | "30" | "unlimited"
  - "unlimited"：当所有应用程序完成当前刻度的提交操作时，时钟就会发送下一个刻度。
- 版本 *1.13* 和 *1.12*：10 | 15 | 30

分区策略

`partitioning_strategies` 部分（必需）指定空间域分区的组织方式。

> **Note**
>
> SimSpace Weaver 仅支持 1 个分区策略。

要指定分区策略的属性，请使用您选择*partitioning-strategy-name*的名称替换。该名称的长度必须为 3-64 个字符，可以包含以下字符：A -Z、a -z、0 -9 和 _ -（短横线）。在名称后指定分区策略的属性。

```
partitioning_strategies:
  partitioning-strategy-name:
    topology: "Grid"
    aabb_bounds:
      x: [aabb-min-x, aabb-max-x]
      y: [aabb-min-y, aabb-max-y]
    grid_placement_groups:
      x: number-of-placement-groups-along-x-axis
      y: number-of-placement-groups-along-y-axis
```

属性

`topology`

指定此分区策略的拓扑（分区排列方案）。

必需：是

类型：字符串

有效值："Grid"

`aabb_bounds`

指定主轴对齐的边界框的边界 (AABB) 用于您的模拟。您可以将边界指定为 2 元素有序数组，这些数组描述每个轴的最小值和最大值（按该顺序排列）(x 以及 y).

必填：条件性。如果拓扑设置为 "Grid"，则此属性为必需项（且只能指定）。

类型：Float 数组（用于各个轴）

有效值：-3.4028235e38-3.4028235e38

`grid_placement_groups`

指定网格拓扑中沿每个轴（x 和 y）的置放群组数量。置放群组是空间上相邻分区（在同一个域中）的集合。

必填：条件性。如果拓扑设置为 "Grid"，则此属性为必需项（且只能指定）。如果您未指定置放群组配置， SimSpace Weaver 将为您计算一个。任何使用分区策略但没有置放群组配置的域都必须指定 `grid_partition`（请参阅空间域分区策略）。

类型：整数（对于各个轴）

有效值：1-20。我们建议让 x * y 等于所需的工作线程数量。否则， SimSpace Weaver 将尝试在可用的工作人员之间平衡您的安置组。

域

`domains` 部分（必需）指定每个域的属性。所有模拟都必须具有至少一个空间域。对于其他域，您可以创建多个部分。每种类型的域都有自己的配置格式。

> **Important**
>
> 版本 1.13 和版本 1.12 不支持多个空间域。

> **Important**
>
> SimSpace Weaver 每次仿真最多支持 5 个域。这包括所有空间、自定义和服务域。

```
domains:
  domain-name:
    domain-configuration
  domain-name:
    domain-configuration
  ...
```

域配置

- 空间域配置
- 自定义域配置
- 服务域配置

空间域配置

要指定空间域的属性，请*spatial-domain-name*使用您选择的名称替换。该名称的长度必须为 3-64 个字符，可以包含以下字符：A -Z、a -z、0 -9 和 _ -（短横线）。在名称后指定空间域的属性。

```
  spatial-domain-name:
    launch_apps_by_partitioning_strategy:
      partitioning_strategy: "partitioning-strategy-name"
      grid_partition:
        x: number-of-partitions-along-x-axis
        y: number-of-partitions-along-y-axis
    app_config:
      package: "app-package-s3-uri"
      launch_command: ["app-launch-command", "parameter1", ...]
      required_resource_units:
        compute: app-resource-units
    image: "ecr-repository-uri"
```

空间域分区策略

launch_apps_by_partitioning_strategy 部分（必需）指定模拟空间的分区策略和维度（以分区数量的形式）。

```
    launch_apps_by_partitioning_strategy:
      partitioning_strategy: "partitioning-strategy-name"
      grid_partition:
        x: number-of-partitions-along-x-axis
        y: number-of-partitions-along-y-axis
```

属性

partitioning_strategy

指定此空间域的分区策略。

必需：是

类型：字符串

有效值：此属性的值必须与 partitioning_strategies 部分中定义的分区策略的名称匹配。有关更多信息，请参阅 分区策略。

grid_partition

指定网格拓扑中沿每个轴（x 和 y）的分区数量。这些维度描述该域的总模拟空间。

必填：条件性。只有拓扑设置为 "Grid" 时，才能指定此属性。此属性取决于该域的指定分区策略的 grid_placement_groups 属性：

- 如果此域的分区策略未指定 grid_placement_groups 配置，则此属性是必需的。
- 如果有 grid_placement_groups 配置，但您未指定 grid_partition，则 SimSpace Weaver 将使用与 grid_placment_groups 配置相同的维度。
- 如果同时指定 grid_placement_groups 和 grid_partition，则 grid_partition 维度必须是 grid_placement_groups 的维度的倍数（例如，如果您的 grid_placement_groups 维度是 2x2，则对 grid_partition 有效的一些维度是 2x2、4x4、6x6、8x8、10x10）。

类型：整数（对于各个轴）

有效值：1-20

空间应用程序配置

app_config 部分（必需）指定该域中应用程序的程序包、启动配置和资源要求。

```
  app_config:
    package: "app-package-s3-uri"
    launch_command: ["app-launch-command", "parameter1", ...]
    required_resource_units:
      compute: app-resource-units
```

属性

package

指定包含应用程序可执行文件/二进制文件的程序包（zip 文件）。程序包必须存储在 Amazon S3 存储桶中。仅支持 zip 文件格式。

必需：是

类型：字符串

有效值：Amazon S3 存储桶中程序包的 Amazon S3 URI。例如，s3://weaver-myproject-111122223333-app-zips-us-west-2/MySpatialApp.zip。

launch_command

指定用于启动应用程序的可执行文件/二进制文件名和命令行参数。每个命令行字符串标记都是数组中的一个元素。

必需：是

类型：字符串数组

required_resource_units

指定 SimSpace Weaver 应分配给此应用程序的每个实例的资源单位数量。资源单位是固定数量的虚拟中央处理单元 (vCPUs) 和随机存取存储器 (RAM) 在工人身上。有关资源单位的更多信息，请参阅端点和服务限额。compute 属性为工作线程的 compute 系列指定资源单位分配，并且是目前唯一有效的分配类型。

必需：是

类型：整数

有效值：1-4

自定义容器映像

image属性（可选）指定用于在此域中运行应用程序的容器镜像的位置（版本1.13和不支持1.12）。SimSpace Weaver 将 URI 提供给 Amazon Elastic Container Registry (Amazon ECR) 中包含映像的存储库。如果未指定此属性，但 default_image 是在顶级 simulation_properties 部分指定的，则该域中的应用程序使用 default_image。有关更多信息，请参阅 自定义容器。

```
    image: "ecr-repository-uri"
```

属性

image

指定容器映像的位置，以便在此域中运行应用程序。

必需：否

类型：字符串

有效值：

- Amazon Elastic Container Registry (Amazon ECR) 中存储库的 URI（例如，111122223333.dkr.ecr.us-west-2.amazonaws.com/my-ecr-repository:latest）

自定义域配置

要指定自定义域的属性，请将 *custom-domain-name* 替换为所选名称。该名称的长度必须为 3-64 个字符，可以包含以下字符：A -Z、a -z、0 -9 和 _ -（短横线）。在名称后指定自定义域的属性。为每个自定义域重复这一过程。

```
custom-domain-name:
  launch_apps_via_start_app_call: {}
  app_config:
    package: "app-package-s3-uri"
    launch_command: ["app-launch-command", "parameter1", ...]
    required_resource_units:
      compute: app-resource-units
    endpoint_config:
      ingress_ports: [port1, port2, ...]
  image: "ecr-repository-uri"
```

属性

launch_apps_via_start_app_call

使用此属性启动您的定制化应用程序时必须使用该属性 StartApp API。

必需：是

类型：不适用

有效值：{}

自定义应用程序配置

app_config section（必需）指定该自定义域中应用程序的程序包、启动配置、资源要求和网络端口。

```
app_config:
  package: "app-package-s3-uri"
  launch_command: ["app-launch-command", "parameter1", ...]
  required_resource_units:
    compute: app-resource-units
  endpoint_config:
    ingress_ports: [port1, port2, ...]
```

属性

package

指定包含应用程序可执行文件/二进制文件的程序包（zip 文件）。程序包必须存储在 Amazon S3 存储桶中。仅支持 zip 文件格式。

必需：是

类型：字符串

有效值：Amazon S3 存储桶中程序包的 Amazon S3 URI。例如，s3://weaver-myproject-111122223333-app-zips-us-west-2/MyCustomApp.zip。

launch_command

指定用于启动应用程序的可执行文件/二进制文件名和命令行参数。每个命令行字符串标记都是数组中的一个元素。

必需：是

类型：字符串数组

`required_resource_units`

指定 SimSpace Weaver 应分配给此应用程序的每个实例的资源单位数量。资源单位是固定数量的虚拟中央处理单元 (vCPUs) 和随机存取存储器 (RAM) 在工人身上。有关资源单位的更多信息，请参阅端点和服务限额。`compute` 属性为工作线程的 `compute` 系列指定资源单位分配，并且是目前唯一有效的分配类型。

必需：是

类型：整数

有效值：1-4

`endpoint_config`

指定此域中应用程序的网络端点。的值`ingress_ports`指定您的自定义应用程序为传入的客户端连接绑定到的端口。 SimSpace Weaver 将动态分配的端口映射到您指定的入口端口。入口端口为 TCP 和 UDP 端口。使用 DescribeApp 用于查找用于连接客户端的实际端口号的 API。

必需：否。如果您未指定端点配置，则该域中的自定义应用程序将没有网络端点。

类型：整数数组

有效值：`1024-49152`。值必须是唯一的。

自定义容器映像

`image`属性（可选）指定用于在此域中运行应用程序的容器镜像的位置（版本`1.13`和不支持`1.12`）。SimSpace Weaver 将 URI 提供给 Amazon Elastic Container Registry (Amazon ECR) 中包含映像的存储库。如果未指定此属性，但 `default_image` 是在顶级 `simulation_properties` 部分指定的，则该域中的应用程序使用 `default_image`。有关更多信息，请参阅 自定义容器。

```
    image: "ecr-repository-uri"
```

属性

`image`

指定容器映像的位置，以便在此域中运行应用程序。

必需：否

类型：字符串

有效值：

- Amazon Elastic Container Registry (Amazon ECR) 中存储库的 URI（例如，`111122223333.dkr.ecr.us-west-2.amazonaws.com/my-ecr-repository:latest`）

服务域配置

要指定服务域的属性，请*service-domain-name*使用您选择的名称替换。该名称的长度必须为 3-64 个字符，可以包含以下字符：A -Z、a -z、0 -9 和 _ -（短横线）。在名称后指定服务域的属性。为每个服务域重复这一过程。

```
service-domain-name:
  launch_apps_per_worker:
    count: number-of-apps-to-launch
  app_config:
    package: "app-package-s3-uri"
    launch_command: ["app-launch-command", "parameter1", ...]
    required_resource_units:
      compute: app-resource-units
    endpoint_config:
      ingress_ports: [port1, port2, ...]
  image: "ecr-repository-uri"
```

按工作线程启动应用程序

`launch_apps_per_worker` 部分（必需）表示这是服务域配置，并指定每个工作线程要启动的服务应用程序的数量。

```
  launch_apps_per_worker:
    count: number-of-apps-to-launch
```

属性

`count`

此属性指定每个工作线程要启动的服务应用程序的数量。

必需：是

类型：整数

有效值：{} | 1 | 2. {} 的值指定 1 的默认值。

服务应用程序配置

app_config section（必需）指定该服务域中应用程序的程序包、启动配置、资源要求和网络端口。

```
app_config:
  package: "app-package-s3-uri"
  launch_command: ["app-launch-command", "parameter1", ...]
  required_resource_units:
    compute: app-resource-units
  endpoint_config:
    ingress_ports: [port1, port2, ...]
```

属性

package

指定包含应用程序可执行文件/二进制文件的程序包（zip 文件）。程序包必须存储在 Amazon S3 存储桶中。仅支持 zip 文件格式。

必需：是

类型：字符串

有效值：Amazon S3 存储桶中程序包的 Amazon S3 URI。例如，s3://weaver-myproject-111122223333-app-zips-us-west-2/MyServiceApp.zip。

launch_command

指定用于启动应用程序的可执行文件/二进制文件名和命令行参数。每个命令行字符串标记都是数组中的一个元素。

必需：是

类型：字符串数组

`required_resource_units`

指定 SimSpace Weaver 应分配给此应用程序的每个实例的资源单位数量。资源单位是固定数量的虚拟中央处理单元 (vCPUs) 和随机存取存储器 (RAM) 在工人身上。有关资源单位的更多信息，请参阅端点和服务限额。`compute` 属性为工作线程的 `compute` 系列指定资源单位分配，并且是目前唯一有效的分配类型。

必需：是

类型：整数

有效值：1-4

`endpoint_config`

指定此域中应用程序的网络端点。`ingress_ports` 的值指定服务应用程序为传入客户端连接绑定的端口。 SimSpace Weaver 将动态分配的端口映射到您指定的入口端口。入口端口为 TCP 和 UDP 端口。使用 DescribeApp 用于查找用于连接客户端的实际端口号的 API。

必需：否。如果您未指定端点配置，则该域中的服务应用程序将没有网络端点。

类型：整数数组

有效值：`1024-49152`。值必须是唯一的。

自定义容器映像

`image`属性（可选）指定用于在此域中运行应用程序的容器镜像的位置（版本`1.13`和不支持`1.12`）。SimSpace Weaver 将 URI 提供给 Amazon Elastic Container Registry (Amazon ECR) 中包含映像的存储库。如果未指定此属性，但 `default_image` 是在顶级 `simulation_properties` 部分指定的，则该域中的应用程序使用 `default_image`。有关更多信息，请参阅 自定义容器。

```
    image: "ecr-repository-uri"
```

属性

`image`

指定容器映像的位置，以便在此域中运行应用程序。

必需：否

类型：字符串

有效值：

- Amazon Elastic Container Registry (Amazon ECR) 中存储库的 URI（例如，`111122223333.dkr.ecr.us-west-2.amazonaws.com/my-ecr-repository:latest`）

放置约束

`placement_constraints` 部分（可选）指定 SimSpace Weaver 应将哪些空间域一起放在同一个工作线程上。有关更多信息，请参阅 配置空间域。

> **Important**
>
> 版本 `1.13` 和 `1.12` 不支持 `placement_constraints`。

```
placement_constraints:
  - placed_together: ["spatial-domain-name", "spatial-domain-name", ...]
    on_workers: ["worker-group-name"]
```

属性

`placed_together`

指定 SimSpace Weaver 应放置在一起的空间域。

必需：是

类型：字符串数组

有效值：架构中指定的空间域的名称

`on_workers`

指定 SimSpace Weaver 应在其上放置域的工作组。

必需：是

类型：1 元素字符串数组

有效值：架构中指定的工作线程组的名称

SimSpace Weaver API 参考资料

SimSpace Weaver 有 2 组不同的应用程序编程接口 (APIs)：

- 服务 APIs-它们 APIs 控制服务和服务资源，例如您的模拟、时钟和应用程序。它们是主 AWS 软件开发套件 (SDK) 的一部分，您可以使用 AWS 命令行界面 (CLI) 来调用它们。有关该服务的更多信息 APIs，请参阅 SimSpace Weaver API 参考。
- App SDK APIs — 它们 APIs 控制仿真中的数据。您可以在应用程序代码中使用这些 API 来执行读取和写入实体字段数据、使用订阅以及监控模拟中的事件等活动。有关更多信息，请参阅 SimSpace Weaver 应用程序 SDK 文件夹中的应用程序 SDK 文档：*sdk-folder*\SimSpaceWeaverAppSdk\documentation

> **Note**
>
> *sdk-folder*是您解压缩包的文件夹。 SimSpaceWeaverAppSdkDistributable

AWS SimSpace Weaver 版本

我们不断改进 AWS SimSpace Weaver。如果您想利用新功能和功能更新，则必须在我们发布新版本时下载最新的 SimSpace Weaver 应用程序 SDK。要使用更新的版本运行现有模拟，可能需要更新其架构和代码，然后启动新的模拟实例。您无需升级，并且可以继续使用以前的版本运行现有模拟。您可以查看此页面，了解不同版本之间的区别。目前支持所有版本。

> **Important**
>
> 最新版本的《AWS SimSpace Weaver 用户指南》仅涵盖最新版本的服务。您可以在AWS SimSpace Weaver 指南目录中找到以前版本的文档，该目录可从主文档登录页面获得。

最新版本

最新版本是：1.17.0

如何查找您的当前版本

如果您使用 SimSpace Weaver 应用程序 SDK 创建了模拟，则`create-project`脚本会将 SDK 库版本下载到您的子目录中。*`sdk-folder`*包含该开发工具包库的子目录的名称包含 SDK 版本号：`SimSpaceWeaverAppSdk-`*`sdk-version`*。例如，1.16.0 版本的库已在。`SimSpaceWeaverAppSdk-1.16.0`

您还可以在的文本文件`app_sdk_distributable_version.txt`中找到 SimSpace Weaver 应用程序 SDK 可分发包的版本。*`sdk-folder`*

下载最新版本

使用以下链接可下载最新版本。

- 完整的应用程序 SDK 可分发程序包
- 仅应用程序 SDK 库

您也可以从SimSpace Weaver 控制台中下载完整的 SimSpace Weaver App SDK 可分发软件包。AWS 管理控制台从导航窗格中，选择下载应用程序 SDK。

> **Warning**
>
> 请勿使用下载任何看似 SimSpace Weaver 应用程序 SDK 可分发包的内容。 AWS CLI 仅使用此页面上的下载链接或控制台中的下载链接。不支持任何其他下载方法或位置，因为其他位置可能包含过时、不正确或有恶意的代码。

应用程序 SDK 下载故障排除

我们使用亚马逊 CloudFront (CloudFront) 来分发应用程序 SDK .zip 文件。您可能会遇到以下几种情况。

- 下载的程序包不是最新版本
  - 如果您下载的.zip 文件不包含最新版本，则可能是您的 CloudFront 边缘站点的缓存尚未更新。在 24 小时后重新下载。
- 您在使用下载链接时收到 HTTP 4xx 或 5xx 错误
  - 在 24 小时后重试。如果您遇到同样的错误，请使用 SimSpace Weaver 控制台底部的反馈链接报告问题。选择报告问题作为反馈类型。
- 您的浏览器报告无法加载该页面
  - 您可能遇到了本地网络或浏览器配置问题。确认您可以加载其他页面。清除您的浏览器缓存，然后重试。确保您没有设置可能阻止下载网址的防火墙规则。
- 尝试保存文件时出现错误
  - 检查您的本地文件系统权限，确保您拥有保存文件的正确权限。
- 将显示您的浏览器 AccessDenied
  - 如果您在浏览器中手动输入网址，请检查拼写是否正确。如果您使用下载链接，请确保浏览器中的网址没有受到任何干扰；请再次使用该链接。

安装最新版本

安装最新版本

1. 下载最新版本。
2. 将 SimSpaceWeaverAppSdkDistributable .zip 解压缩到一个文件夹。
3. `python setup.py`从解压缩的最新版本 SimSpace Weaver 应用程序 SDK 文件夹中运行。

4. 使用已解压缩的最新版本 SimSpace Weaver 应用程序 SDK 文件夹，而不是以前的版本。

服务版本

版本	备注	发行日期	文档	下载应用程序 SDK
1.17.0	对 SimSpace Weaver 应用程序 SDK 可分发包进行了重大更改 • 我们用基于 Python 的脚本替换了 Windows 批处理和 Linux Bash 脚本。因此，即使你不使用（或打算使用）Python SDK，现在也需要使用 Python 3.9 才能使用脚本和示例。 • 此版本增加了对亚马逊 Linux 2 的支持。 • 我们修复了以下内容中的几个错误 SimSpace Weaver Local. 有关更多信息，请参阅版本注释。	2024 年 4 月 17 日	本指南	• 完整程序包 • 仅库 参阅故障排除。

版本	备注	发行日期	文档	下载应用程序 SDK
	错误修复 • 我们修复了一个错误，该错误会导致实体在没有完成远程工作人员之间的转移时变为无主。			
1.16.0	新特征： • 现在，您可以使用 SimSpace Weaver 应用程序 SDK APIs 中的消息在应用程序之间发送和接收消息。此功能可用于 C++、Python 以及 Unity 和虚幻引擎 5 集成。	2024 年 2 月 12 日	参见AWS SimSpace Weaver 指南目录。	• 完整程序包 • 仅库 参阅故障排除。

版本	备注	发行日期	文档	下载应用程序 SDK
1.15.3	SimSpace Weaver Local 更新： • 我们变了 SimSpace Weaver Local 使其与开发更紧密地保持一致 AWS Cloud。这些变化影响了 C++、Python、Unity 和虚幻引擎项目和工作流程 SimSpace Weaver Local.	2023 年 12 月 4 日	参见AWS SimSpace Weaver 指南目录。	不可供下载
1.15.2	应用程序 SDK 可分发程序包更新： • 我们更新了`Dockerfile` ，使用了所需的特定版本 cmake。 如果不进行此更改，Docker 容器构建可能会失败。	2023 年 11 月 2 日	参见AWS SimSpace Weaver 指南目录。	不可供下载
1.15.1	功能更新： • Python SDK：此版本修复了导致基于 Python 的模拟在 AWS Cloud 中失败的问题。	2023 年 9 月 22 日	参见AWS SimSpace Weaver 指南目录。	不可供下载

版本	备注	发行日期	文档	下载应用程序 SDK
1.15.0	新特征： • Python 开发工具包：您现在可以在 Python 中开发模拟。 SimSpace Weaver 应用程序 SDK 可分发程序包中有一个用于示例 Python 项目及其 Python 查看客户端的模板。	2023 年 8 月 31 日	参见AWS SimSpace Weaver 指南目录。	不可供下载

版本	备注	发行日期	文档	下载应用程序 SDK
1.14.0	新功能： • 自定义容器：创建你自己的基于 Amazon Linux 2 (AL2) 的容器镜像，将其存储在亚马逊弹性容器注册表 (Amazon ECR) Container Registry 中，然后用它在中运行 SimSpace Weaver 你的应用程序。 AWS Cloud • 多个空间域：在模拟中创建多个空间域。将模拟逻辑分开，而不是将其全部组合到一个空间应用程序中。根据空间域的要求为其分配不同的资源。 • 无限制的刻度率：让模拟的运行速度与代码的运行速度一样快。设置模拟的时钟，使其在所有应用程序完成当前刻度的提交	2023 年 7 月 26 日	参见AWS SimSpace Weaver 指南目录。	不可供下载

版本	备注	发行日期	文档	下载应用程序 SDK
	操作后立即发送下一个刻度。 SimSpace Weaver 应用程序 SDK： • 现在，`tick_rate` 的值是一个字符串。该值必须包含在双引号内 (")。早期版本的刻度率仍然是整数。			
1.13.1	SimSpace Weaver 应用程序 SDK： • 功能更新：现在，使用 `PathfindingSampleUnreal` 模板可正常创建项目。	2023 年 6 月 7 日	参见AWS SimSpace Weaver 指南目录。	不可供下载

版本	备注	发行日期	文档	下载应用程序 SDK
1.13.0	SimSpace Weaver 服务 APIs: • • 新的 CreateSnapshot 操作 • 对 StartSimulation 操作的更改： • 添加了一个 SnapshotS3Location 参数，以便从快照启动。 • 现在，SchemaS3Location 参数是可选的。 • 对 DescribeSimulation 输出的更改： • SchemaError 已弃用。 • 添加了 StartError 字段。 • 添加了 SnapshotS3Location 字段。	2023 年 4 月 29 日	参见AWS SimSpace Weaver 指南目录。	不可供下载

版本	备注	发行日期	文档	下载应用程序 SDK
	• 添加了 SNAPSHOT_IN_PROGRESS 模拟状态。 • 新的 S3Destination 数据类型 SimSpace Weaver 控制台： • 创建快照的新功能。 • 从快照启动模拟的新功能。 SimSpace Weaver 应用程序 SDK： • 支持快照的新脚本 • create-snapshot- *project-name* .bat • start-from-snapshot- *project-name* .bat • quick-start-from-snapshot- *pro*			

版本	备注	发行日期	文档	下载应用程序 SDK
	name -cli.bat • list-snapshots- *projec* *name* .bat • 现在，每个项目都使用一个 Amazon S3 存储桶：weaver-*lowercase-project-name* --*account-number* *region*			

版本	备注	发行日期	文档	下载应用程序 SDK
1.12.3	SimSpace Weaver 应用程序 SDK： • 现在，以下脚本支持 `--maximum-duration` 参数： • `quick-start-`*`project-name`*`-cli.bat` • `quick-start-`*`project-name`*`-cli.sh` • `start-simulation-`*`pro`*`*name*`.bat` • `start-simulation-`*`pro`*`*name*`.sh` • `run-`*`project-name`*`.bat` • `run-`*`project-name`*`.sh`	2023 年 3 月 27 日	参见AWS SimSpace Weaver 指南目录。	不可供下载
1.12.2	SimSpace Weaver 应用程序 SDK： • 错误修复：`docker-create-image.bat` 现在可以正常运行。	2023 年 3 月 1 日	参见AWS SimSpace Weaver 指南目录。	不可供下载

版本	备注	发行日期	文档	下载应用程序 SDK
1.12.1	SimSpace Weaver 应用程序 SDK： • 这些脚本现在接受用于 AWS 身份验证的 AWS CLI 配置文件。 • 这些脚本现在 AWS IAM Identity Center 支持 AWS 身份验证。 SimSpace Weaver Local: • 错误修复：现在，如果所有空间应用程序都未加入模拟，`Api::BeginUpdateWillBlock` 可正确返回 `true`。	2023 年 2 月 28 日	参见AWS SimSpace Weaver 指南目录。	不可供下载
1.12.0	公开发行版本 (GA)	2022 年 11 月 29 日	参见AWS SimSpace Weaver 指南目录。	不可供下载

AWS SimSpace Weaver 版本 1.17.0

此版本是对 A SimSpace Weaver pp SDK 可分发包的全面改版。我们用基于 Python 的脚本替换了过时的 Windows 批处理和 Linux Bash 脚本。

> **Important**
>
> 现在，Python 3.9 是使用脚本和示例的必要条件，而不仅仅是用于 Python SDK。

目录

1.17.0 的主要变化

- 简化了项目创建
  - 运行后`setup.py`，您只需复制粘贴示例即可创建自己的项目。
- 一键式示例
  - 分发 zip 文件现在包含在设置发行版后可以使用的 ready-to-use示例。
- 现在，每个 SDK 都存在于自己的目录中：`cpp`、`python`、`unreal`、`unity`。根据您使用的 SDK，您可能需要更新路径。
- 对帮助脚本的改进。
  - 脚本现在包含多个 AWS CLI 选项，以最大限度地提高其灵活性。
  - 集成的控制台客户端启动和连接是快速入门的一部分。
  - 改进了控制台输出。
  - Unreal 和 Unity 示例构建现在可以使用`quick-start`，无需再手动执行任何步骤。
  - SimSpace Weaver Local 现在只需调用即可工作`quick-start`，不再需要手动构建和启动。
  - SimSpace Weaver Local `quick-start`集成了对记录应用程序输出的支持。
  - SimSpace Weaver Local 现在可以在非 GUI 环境中启动，例如在 ssh 会话中。
  - “自定义容器” 功能现已集成到`quick-start`脚本中。
  - 增加了对亚马逊 Linux 2 (AL2) 的支持：适用于 Windows 的脚本工作流程， AL2 现在可以与之媲美。以前， AL2 项目需要更多的手动步骤，SimSpace Weaver Local 不支持 AL2。
- 虚幻引擎和Unity插件现已包含在 SimSpace Weaver 应用程序SDK可发行包中。
- 的错误修复 SimSpace Weaver Local
  - 修复了可以为实体分配相同实体 ID 的错误。

- 修复了两个分区可以分配相同分区 ID 的错误。
- 修复了与应用程序尝试写入其不拥有的实体相关的错误。
- 解决了内存泄漏问题。

将项目更新到 1.17.0

1. 设置 1.17.0 发行版：请再次完成安装过程，因为我们在 1.17.0 中对其进行了更改。有关更多信息，请参阅 正在设置 SimSpace Weaver。
2. 现在，每个 Weaver App SDK 都存在于自己的目录中。更新您的构建路径以反映这一点。

   a. C++ 目录：`SimSpaceWeaverAppSdk/cpp`

      - C SimSpace Weaver ++ 应用程序 SDK 现在使用`FindSimSpaceWeaverAppSdk.cmake`文件。此文件设置了一个链接到的`weaver`目标，并包含了在中为 Weaver 构建时的重要错误修复。 AWS Cloud你应该使用它而不是直接链接到二进制文件。

   b. Python 目录：`SimSpaceWeaverAppSdk/python`

   c. Unity 插件：`SimSpaceWeaverAppSdk/unity`

   d. 虚幻引擎插件：`SimSpaceWeaverAppSdk/unreal`

3. 之前的`tools`脚本不适用于新的 SimSpace Weaver 发行版。要在项目中使用新`tools`脚本，请执行以下操作：

   a. 删除您的旧`tools/windows`、`tools/linux`、和`tools/local`目录。

   b. 复制与您的项目使用相同 SimSpace Weaver 应用程序 SDK 的示例项目的`tools`目录。在复制此目录`setup.py`之前，请确保已运行。

> **Important**
>
> 仅保证这些工具脚本可以与示例项目配合使用。您可能需要编辑这些脚本，尤其是脚本，才能与您的项目配合使用。`build.py`任何编辑都是您的项目所独有的，因此我们无法提供任何指导。

关于版本 1.17.0 的常见问题

我必须更新到 1.17.0 版本吗？

这不是必需的更新，因为 SimSpace Weaver API 或 SimSpace Weaver 应用程序 SDK 没有变化。如果要使用 1.17.0，则必须更新到 1.17.0 SimSpace Weaver Local，其中包含多个错误修复。

所需的最低 Python 版本是多少？

Python 3.9 是最低版本。

所需的最低 CMake 版本是多少？

CMake 3.13 版本是最低版本。

需要的最低版本的虚幻引擎是多少？

虚幻引擎5.0是最低限度。

要求的 Unity 的最低版本是多少？

Unity 版本 2022.3.19.F1 是最低版本。

AWS SimSpace Weaver 版本 1.15.1

该版本是最初在 SimSpace Weaver 版本 1.15.0 中发布的 Python SDK 的必需更新。它修复了 AWS Cloud中导致基于 Python 的模拟失败的版本不匹配问题。请使用此版本，而不是 1.15.0。

将现有 Python 项目更新到 1.15.1

如果您已有使用 Python SDK 版本 1.15.0 创建的现有 Python 项目，则必须执行以下步骤，以便将其更新到 1.15.1，从而使其可在 AWS Cloud中运行。

您还可以使用 Python SDK 1.15.1 创建一个新的 Python 项目，然后将您的自定义代码转移到新项目中，这样就不必执行此过程。

将 1.15.0 Python 项目更新到 1.15.1

1. 转到您的 Python 项目的文件夹。
2. 在 `src/PythonBubblesSample/bin/run-python` 中更改以下行：

```
export PYTHONPATH=$PYTHONPATH:/roapp/lib
```

更改为以下来源：

```
export PYTHONPATH=$PYTHONPATH:$LD_LIBRARY_PATH:/roapp/lib
```

3. 在 CMakeLists.txt 中删除以下几行：

- ```
  file(COPY "${SDK_PATH}/libweaver_app_sdk_python_v1_$ENV{PYTHON_VERSION}.so"
   DESTINATION "${ZIP_FILES_DIR}/lib/weaver_app_sdk_v1")
  ```
- ```
  file(RENAME "${ZIP_FILES_DIR}/lib/weaver_app_sdk_v1/libweaver_app_sdk_python_v1_
  $ENV{PYTHON_VERSION}.so" "${ZIP_FILES_DIR}/lib/weaver_app_sdk_v1/
  libweaver_app_sdk_python_v1.so")
  ```
- ```
  message("    * COPYING WEAVER PYTHON SDK TO BUILD DIR ${ZIP_FILES_DIR}....")
  ```
- ```
  file(COPY ${SDK_DIR} DESTINATION ${ZIP_FILES_DIR}/lib/weaver_app_sdk_v1)
  ```

有关版本 1.15.1 的故障排除

更新 1.15.0 Python 模拟后，它无法在中启动 AWS Cloud

征兆：启动模拟后大约过了 5-10 分钟，模拟管理日志报告 `internal error` 且模拟状态为 `FAILED`。

如果应用程序 zip 文件中包含 1.15.0 Python SDK 的库文件，则可能会发生这种情况。请确保您已完成更新项目的步骤，并确保您的 zip 文件中没有 `libweaver_app_sdk_python_v1.so`，也没有以任何方式对其进行引用。

有关版本 1.15.1 的常见问题解答

除了 Python SDK 之外，该版本还会影响其他方面吗？

否。

我必须更新为版本 1.15.1 吗？

如果您不打算在空间应用程序中使用 Python，则不必更新到 1.15.1。如果你更新到 1.15.0，则基于 Python 的模拟将无法在中运行。 AWS Cloud如果您使用的是 1.15.0，我们建议您更新到 1.15.1。

什么是 **$LD_LIBRARY_PATH**？

当您的模拟在 AWS Cloud中运行时，这是 Python SDK 的位置。这是 1.15.1 的新特征。我们进行此更改是为了避免将来出现 Python 版本问题。从功能上而言，链接到该目录与在 1.15.0 中链接到 `libweaver_app_sdk_python_v1.so` 相同。

的文档历史记录 AWS SimSpace Weaver

下表描述了 SimSpace Weaver 文档的重要更改。

日期	更改	文档更新	更新的 API 版本
2025 年 5 月 20 日	终止支持通知	终止支持通知：2026 年 5 月 20 日，AWS 将终止对的支持。 AWS SimSpace Weaver 2026 年 5 月 20 日之后，您将无法再访问 SimSpace Weaver 控制台或 SimSpace Weaver 资源。有关更多信息，请参阅AWS SimSpace Weaver 终止支持。	不适用
2024 年 4 月 17 日	更新了内容	在 1.17.0 版本的用户指南中进行了更新。对本设置章和入门教程进行了重大更改。有关更多信息，请参阅发行说明。	不适用
2024 年 2 月 12 日	更新了内容	更新了 1.16.0 版本的AWS SimSpace Weaver 版本章节。	不适用
2024 年 2 月 12 日	新增内容	在 消息收发 1.16.0 版本中添加了该部分。本节介绍 APIs 添加到 SimSpace Weaver 应用程序 SDK 中的消息。您可以使用它们 APIs 在应用程序之间发送和接收消息。	不适用

日期	更改	文档更新	更新的 API 版本
2024 年 2 月 12 日	更新了内容	更新了 1.16.0 版的SimSpace Weaver 仿真架构参考章节。	不适用
2024 年 2 月 12 日	更新了内容	在本SimSpace Weaver 端点和配额章中增加了消息传递的服务配额。	不适用
2024 年 2 月 12 日	新指南	将 1.16.0 之前版本的内容拆分为单独的指南。添加了AWS SimSpace Weaver 指南目录（可从主文档登录页面获得）以访问先前版本的指南。	不适用
2023 年 12 月 4 日	更新了内容	更新了版本 1.15.3 发行版的AWS SimSpace Weaver 版本章节。	不适用
2023 年 12 月 4 日	更新了内容	更新了AWS SimSpace Weaver 版本章节，以加入最新版本的安装说明。	不适用
2023 年 12 月 4 日	更新了内容	更新了SimSpace Weaver Local 的服务限额。	不适用
2023 年 12 月 4 日	新增和更新的内容	重组了当地发展 SimSpace Weaver部分，并添加了一个新页面，其中描述了 1.15.3 版本中引入的 SimSpace Weaver Local 的差异。	不适用

日期	更改	文档更新	更新的 API 版本
2023 年 11 月 7 日	更新了内容	更新了设置 Docker 和 WSL 以使用应用程序 SDK 的直接下载链接/URL 的说明。有关更多信息，请参阅 设置您的本地环境用于 SimSpace Weaver。	不适用
2023 年 11 月 2 日	更新了内容	更新了版本 1.15.2 的服务版本页面。有关更多信息，请参阅 服务版本。	不适用
2023 年 10 月 23 日	更新了内容	更新了服务版本页面，添加了下载应用程序 SDK 可分发程序包的新说明。现在，客户只能使用我们批准的直接下载链接之一，而不能使用下载应用程序 SDK 可分发包。AWS CLI 有关更多信息，请参阅 下载最新版本。	不适用
2023 年 9 月 22 日	新增内容	添加了版本说明页面，其中包括版本 1.15.1 的更新说明。有关更多信息，请参阅 AWS SimSpace Weaver 版本 1.15.1。	不适用
2023 年 9 月 10 日	新增内容	为 AWS CLI 无法识别的情况添加了疑难解答部分 SimSpace Weaver。有关更多信息，请参阅 他们认 AWS CLI 不出来 `simspaceweaver` 。	不适用

日期	更改	文档更新	更新的 API 版本
2023 年 9 月 10 日	更新了内容	更新了 WSL AWS CLI 中的安装说明。有关更多信息，请参阅 为以下内容设置 SimSpace Weaver 分发包 Amazon Linux 2 (AL2) 在 Windows Subsystem for Linux (WSL)。	不适用
2023 年 9 月 7 日	API 更新	BucketName 现在 ObjectKey 是 S3Location 数据类型所必需的。BucketName 现在是 S3Destination 数据类型所必需的。	AWS SDK：2023-09-07
2023 年 8 月 31 日	新增内容	为版本 1.15.0 添加了一个新部分：使用 Python。	不适用
2023 年 8 月 15 日	更新了内容	更新了 AWS SimSpace Weaver 版本 中的下载说明，现在仅列出官方 SimSpace Weaver Amazon S3 存储桶。其他下载位置不受控制 AWS 且可能包含恶意代码。	不适用
2023 年 7 月 26 日	更新了内容	已更新 时钟。	不适用
2023 年 7 月 26 日	更新了内容	已更新 配置空间域。	不适用
2023 年 7 月 26 日	新增内容	增加了新的部分：自定义容器。	不适用

日期	更改	文档更新	更新的 API 版本
2023 年 7 月 26 日	更新了内容	已将 AWS SimSpace Weaver 版本 更新为版本 1.14.0。	不适用
2023 年 7 月 6 日	新增内容	增加了新的部分：PathfindingSample 控制台客户端无法连接。	不适用
2023 年 6 月 7 日	更新了内容	已将 AWS SimSpace Weaver 版本 更新为版本 1.13.1。	不适用
2023 年 5 月 15 日	新增内容	增加了新的部分：将快照用于 AWS CloudFormation。	不适用
2023 年 4 月 29 日	新增内容	添加了版本 1.13.0 的内容。有关更多信息，请参阅 AWS SimSpace Weaver 版本。	AWS SDK：2023-04-28
2023 年 3 月 27 日	新增内容	添加了介绍模拟最长持续时间的部分。在 1.12.3 版本的教程中添加了说明，该版本为 SimSpace Weaver 应用程序 SDK 脚本添加了对 `--maximum-duration` 参数的支持。	不适用

日期	更改	文档更新	更新的 API 版本
2023 年 3 月 9 日	更改的内容	澄清了我们仅提供适用于 Windows 上的 Docker 和适用于 Windows Subsystem for Linux (WSL) 的说明，不支持 WSL（以及任何其他 Linux 环境）。	不适用
2023 年 2 月 28 日	新增内容	添加了描述 SimSpace Weaver 版本的章节。	不适用
2023 年 2 月 28 日	更改的内容	更改了有关身份验证的内容，以包括对 AWS Command Line Interface (AWS CLI) 的使用 AWS IAM Identity Center 和命名配置文件。	不适用
2023 年 2 月 17 日	新增内容	添加了有关使用管理资源的部分 AWS CloudFormation。	不适用
2023 年 1 月 23 日	新增内容	添加了调试本地模拟的说明。	不适用
2022 年 11 月 29 日	服务启动	发布了 SimSpace Weaver 的用户指南和 API 参考。	AWS SDK：2022-11-29

术语表

此词汇表定义了特定于的术语 AWS SimSpace Weaver。

有关最新 AWS 术语，请参阅《AWS 通用参考》中的AWS 词汇表。

A

应用程序	您创建的可执行代码（也称为二进制文件）。应用程序一词可以指代码或代码的运行实例。应用程序封装了模拟行为。应用程序可创建、删除、读取和更新实体。
应用程序 SDK	用于将应用程序与 SimSpace Weaver进行集成的软件开发工具包 (SDK)。SDK APIs 提供了读取和写入实体数据以及跟踪仿真时间的功能。有关更多信息，请参阅 SimSpace Weaver 应用程序 SDK。

C

客户端	存在于仿真之外 SimSpace Weaver 并通过自定义应用程序或服务应用程序与之交互的进程（或其定义）。您可以使用客户端来查看或更改模拟状态。
时钟	的内部调度流程 SimSpace Weaver的抽象。时钟向应用程序发布刻度以保持时间同步。每个模拟都有自己的时钟。
时钟率	时钟每秒向应用程序发布的刻度数量。有关支持的时钟率的更多信息，请参阅SimSpace Weaver 端点和配额。
时钟刻度率	参阅时钟频率。
计算资源单位	工作线程上的计算资源（处理器和内存）单位。通常为应用程序的单个实例分配 1 个计算资源单位。您可以为每个应用程序分配多个计算资源单位。
自定义应用程序	一种用于读取模拟状态并与之交互的应用程序。自定义应用程序可以在模拟中创建实体，但不拥有这些实体。自定义应用程序创建实体时，必须将该实体转移到空间域中。您可以使用该应用程序控制自定义应用程序的生命周期 APIs。有关更多信息 SimSpace Weaver APIs，请参阅SimSpace Weaver API 参考资料。

自定义域	包含自定义应用程序的域。
自定义分区	自定义应用程序的分区。

D

截止时间	操作（例如，一个刻度内的处理）应完成的实际时间。
域	一组运行相同可执行代码（应用程序二进制文件）且具有相同启动选项的应用程序实例。

E

端点（服务）	程序（例如）用于连接到服务的完全限定域名 (FQDN AWS Command Line Interface)。 SimSpace Weaver
端点（模拟）	客户端用于连接到模拟的 IP 地址和端口号。您可以在自定义应用程序和服务应用程序上配置端点。
实体	客户数据对象（或其定义）。实体可以是静态的（保持在一个位置），也可以是动态的（在模拟空间中移动）。例如，模拟中的人和建筑物。

我

索引（模拟）	对模拟空间属性的描述，包括其空间边界和坐标系。

L

生命周期（应用程序）	对应用程序在模拟过程中预期要经历的逻辑步骤的描述。生命周期要么是托管的（SimSpace Weaver 启动和停止应用程序），要么是非托管的（由你启动和停止应用程序）。
加载（实体字段数据）	从 State Fabric 中读取实体字段数据。

P

分区	工作线程上的一段共享内存。每个分区都包含域内实体的一个离散子集。每个应用程序都分配有一个分区。应用程序拥有其分区中的所有实体。当应用

	程序创建实体时，它会在其分区中创建。当实体从一个分区移动到另一个分区时，所有权会从源分区的应用程序转移到目标分区的应用程序。

R

资源单位	请参阅???。

S

架构	描述模拟配置的 YAML 或 JSON 文档。 SimSpace Weaver 使用架构来创建模拟资源。
服务应用程序	一种用于读取模拟状态并与之交互的应用程序。服务应用程序可以在模拟中创建实体，但必须将其传输到空间域。 SimSpace Weaver 管理服务应用程序的生命周期，并在模拟中的每个工作线程上启动 1 个（也可以是多个，具体在模拟架构中指定）。
服务域	包含服务应用程序的域。
服务分区	服务应用程序的分区。
模拟（资源）	运行模拟虚拟空间的计算集群的抽象。您可以有多个模拟。您可以使用架构配置模拟。
空间应用程序	一种封装核心模拟逻辑的应用程序。每个空间应用程序有且只有 1 个分区。
空间域	包含空间应用程序的域。
空间分区	空间应用程序的分区。
State Fabric	SimSpace Weaver的内存数据库。State Fabric存储仿真状态，包括实体和内部 SimSpace Weaver 数据。
存储（实体字段数据）	将实体字段数据写入 State Fabric。
订阅	要求特定应用程序实例从订阅区域接收数据的长期请求。订阅应用程序使用订阅来发现订阅区域内实体的更改。
订阅区域	模拟空间的二维区域。订阅是指订阅区域。订阅区域可以跨越多个分区，也可以包含部分分区。订阅区域在其定义的范围内是连续的。

T

刻度	时间的离散值（挂钟时间或模拟时间）。应用程序的迭代速度可能快于刻度持续时间，但应在特定的截止时间内写入指定的刻度。所有应用程序针对指定刻度的所有操作都必须先完成，然后才能开始下一个刻度。
刻度率	参阅时钟频率。
时间（实际）	从现实的角度看当前时间。 SimSpace Weaver 使用 64 位 POSIX 时间戳，即自纪元以来的纳秒数 Unix (January 1, 1970, 00:00:00 UTC).
时间（模拟）	从仿真角度看当前时间。 SimSpace Weaver 使用 64 位整数逻辑刻度计数器，该计数器可能与实际时间不直接对应。

W

工作线程	运行模拟代码的亚马逊弹性计算云 (Amazon EC2) 实例。