aws

버전 1.17.0 사용 설명서

AWS SimSpace Weaver

AWS SimSpace Weaver: 버전 1.17.0 사용 설명서

Table of Contents

지원 종료 공지: 2026 AWS 년 5월 20일에에 대한 지원이 종료됩니다 AWS SimSpace Weaver. 2026 년 5월 20일 이후에는 SimSpace Weaver 콘솔 또는 SimSpace Weaver 리소스에 더 이상 액세스할 수 없습니다. 자세한 내용은 AWS SimSpace Weaver 지원 종료를 참조하세요.

기계 번역으로 제공되는 번역입니다. 제공된 번역과 원본 영어의 내용이 상충하는 경우에는 영어 버전이 우선합니다.

AWS SimSpace Weaver란 무엇인가요?

AWS SimSpace Weaver 는에서 대규모 공간 시뮬레이션을 빌드하고 실행하는 데 사용할 수 있는 서비스입니다 AWS 클라우드. 예를 들어 군중 시뮬레이션, 대규모 실제 환경, 몰입형 대화형 환경을 만들 수 있습니다.

를 사용하면 여러 Amazon Elastic Compute Cloud(Amazon EC2) 인스턴스에 시뮬레이션 워크로드를 배포할 SimSpace Weaver수 있습니다.는 기본 AWS 인프라를 SimSpace Weaver 배포하고 시뮬레이션을 실행하는 Amazon EC2 인스턴스 간의 시뮬레이션 데이터 관리 및 네트워크 통신을 처리합니다.

의 주요 개념 SimSpace Weaver

시뮬레이션 또는 게임은 이를 실행하는 컴퓨터에 따라 제한됩니다. 가상 세계의 크기와 복잡성이 커지면 처리 성능이 저하되기 시작합니다. 계산 시간이 더 오래 걸리고, 시스템 메모리가 부족하고, 클라이언트 프레임 속도가 떨어집니다. 실시간 성능이 필요하지 않은 시뮬레이션의 경우 이는 성가신 일일 수 있습니다. 또는 처리 지연이 증가하여 비용이 증가하는 비즈니스 크리티컬 상황일 수도 있습니다. 시뮬레이션이나 게임에 실시간 성능이 필요한 경우 성능 저하가 분명히 문제입니다.

성능 한계에 도달하는 시뮬레이션의 일반적인 해결 방법은 시뮬레이션을 단순화하는 것입니다. 사용자가 많은 온라인 게임에서는 가상 세계의 사본을 여러 서버에 만들어 여러 서버에 분산시켜 크기 조정 문제를 해결하는 경우가 많습니다.

SimSpace Weaver 는 가상 세계를 공간적으로 나누고에서 실행되는 컴퓨팅 인스턴스 클러스터에 조각을 배포하여 조정 문제를 해결합니다 AWS 클라우드. 컴퓨팅 인스턴스는 함께 작동하여 전체 시뮬레이션 세계를 병렬로 처리합니다. 시뮬레이션 세계는 그 안에 있는 모든 것은 물론 여기에 연결된 모든 클라이언트에게 하나의 통합된 공간으로 나타납니다. 하드웨어 성능 제한 때문에 더 이상 시뮬레이션을 단순화할 필요가 없습니다. 대신 클라우드에서 컴퓨팅 용량을 추가할 수 있습니다.

주제

- SimSpace Weaver 작동 방식
- 사용 방법 SimSpace Weaver
- 시뮬레이션 스키마
- 작업자 및 리소스 단위
- 시뮬레이션 클럭
- 파티션
- 상태 패브릭

- 개체
- 앱

SimSpace Weaver 작동 방식

시뮬레이션은 그 안에 객체가 있는 세계로 구성됩니다. 일부 객체(예: 사람, 차량)는 움직이고 작업을 수행합니다. 다른 객체(예: 나무, 건물)는 정적입니다. 에서 SimSpace Weaver개체는 시뮬레이션 세계의 객체입니다.

시뮬레이션 세계의 경계를 정의하고 그리드로 나눕니다. 전체 그리드에서 작동하는 시뮬레이션 로직을 생성하는 대신 그리드의 한 셀에서 작동하는 시뮬레이션 로직을 생성합니다. 에서 SimSpace Weaver공간 앱은 그리드의 셀에 대한 시뮬레이션 로직을 구현하는 사용자가 작성하는 프로그램입니다. 여기에는 해당 셀의 모든 객체에 대한 로직이 포함됩니다. 공간 앱의 소유권 영역은 공간 앱이 제어하는 그리드 셀입니다.

> **Note**
>
> 에서 "app" SimSpace Weaver이라는 용어는 앱의 코드 또는 해당 코드의 실행 중인 인스턴스를 지칭할 수 있습니다.

그리드로 분할된 시뮬레이션 세계

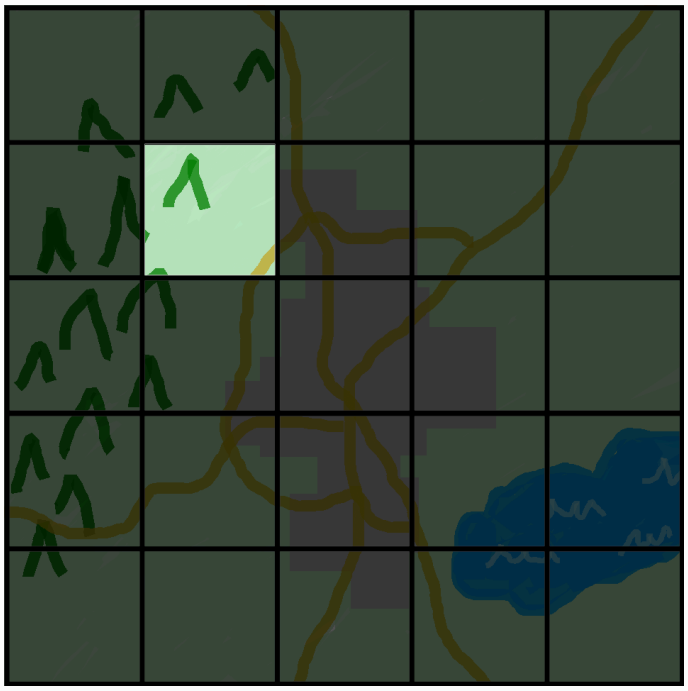
시뮬레이션 세계를 그리드로 나눕니다. 각 공간 앱은 해
당 그리드의 단일 셀에 대한 시뮬레이션 로직을 구현합니다.

SimSpace Weaver 는 그리드의 각 셀에 대해 공간 앱 코드의 인스턴스를 실행합니다. 모든 공간 앱 인스턴스는 병렬로 실행됩니다. 기본적으로는 전체 시뮬레이션을 여러 개의 더 작은 시뮬레이션으로 SimSpace Weaver 나눕니다. 각 소형 시뮬레이션은 전체 시뮬레이션 세계의 일부를 처리합니다. SimSpace Weaver 는의 여러 Amazon Elastic Compute Cloud(Amazon EC2) 인스턴스(작업자라고 함)에 이러한 소형 시뮬레이션을 배포하고 실행할 수 있습니다 AWS 클라우드. 한 작업자가 여러 개의 공간 앱을 실행할 수 있습니다.

엔터티는 시뮬레이션 세계에서 이동할 수 있습니다. 엔터티가 다른 공간 앱의 소유권 영역(그리드의 다른 셀)에 들어가면 새 영역의 공간 앱 소유자가 해당 엔터티에 대한 제어권을 넘겨받습니다. 시뮬레이션을 여러 작업자에서 실행하는 경우 엔터티가 한 작업자의 공간 앱 제어에서 다른 작업자의 공간 앱으로 이동할 수 있습니다. 개체가 다른 작업자로 이동하면는 기본 네트워크 통신을 SimSpace Weaver 처리합니다.

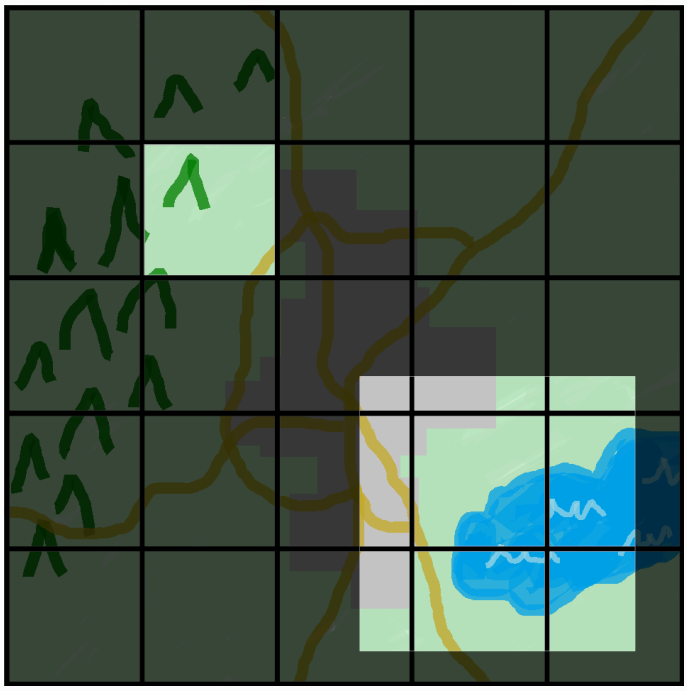
구독

공간 앱의 세계 보기는 자체 소유 영역입니다. 시뮬레이션 세계의 다른 부분에서 어떤 일이 벌어지고 있는지 알아보기 위해 공간 앱은 구독을 생성합니다. 구독 영역은 전체 시뮬레이션 세계 영역의 하위 집합입니다. 구독 영역에는 공간 앱의 자체 소유권 영역을 포함하여 여러 소유권 영역의 일부가 포함될 수 있습니다.는 구독 영역 내에서 발생하는 모든 엔터티 이벤트(예: 입력, 종료, 생성, 업데이트 및 삭제)를 공간 앱에 SimSpace Weaver 알립니다.



공간 앱의 세계 보기

공간 앱의 세계 보기는 소유권 영역이며, 이는 세계 그리드의 한 셀입니다.



구독 영역이 추가된 공간 앱의 보기

공간 앱은 구독을 사용하여 시뮬레이션 세계의 다른 부분에서 어떤 일이 벌어지고 있는지 알아봅니다. 구독 영역에는 여러 개의 그리드 셀과 셀의 일부가 포함될 수 있습니다.

예를 들어 물리적으로 상호 작용하는 엔터티를 시뮬레이션하는 앱은 소유권 영역의 공간적 경계 바로 너머에 있는 엔터티에 대해 알아야 할 수 있습니다. 이를 위해 앱은 소유권 영역과 경계를 이루는 영역을 구독할 수 있습니다. 구독을 생성한 후 앱은 해당 영역의 엔터티 이벤트에 대한 알림을 수신하고 해당 엔터티를 읽을 수 있습니다. 또 다른 예로 자율주행차를 들 수 있는데, 어떤 앱이 해당 영역을 소유하고 있든 관계없이 전방 200미터 앞에 있는 모든 엔터티를 볼 수 있어야 합니다. 차량용 앱은 가시 영역을 포함하는 축 정렬된 경계 상자 (AABB)로 필터를 사용하여 구독을 생성할 수 있습니다.

시뮬레이션의 공간적 측면을 관리하는 역할을 하지 않는 시뮬레이션 로직을 만들 수 있습니다. 사용자 지정 앱은 단일 작업자에서 실행되는 실행 가능한 프로그램입니다. 사용자 지정 앱의 수명 주기(시작 및 중지)를 제어할 수 있습니다. 시뮬레이션 클라이언트는 사용자 지정 앱에 연결하여 시뮬레이션을 보거나 상호 작용할 수 있습니다. 또한 모든 작업자에서 실행되는 서비스 앱을 생성할 수 있습니다.는 시뮬레이션을 실행하는 모든 작업자에서 서비스 앱의 인스턴스를 SimSpace Weaver 시작합니다.

사용자 지정 앱과 서비스 앱은 구독을 생성하여 엔터티 이벤트에 대해 알아보고 엔터티를 읽을 수 있습니다. 이러한 앱은 공간적이지 않기 때문에 소유권 영역이 없습니다. 구독 사용만이 시뮬레이션 세계에서 일어나는 일을 알아낼 수 있는 유일한 방법입니다.

사용 방법 SimSpace Weaver

를 사용할 때 따라야 할 주요 단계는 다음과 SimSpace Weaver같습니다.

1. C++ 앱 SDK를 통합하는 SimSpace Weaver 앱을 작성하고 빌드합니다.
   a. 앱은 API 직접 호출을 통해 시뮬레이션 상태와 상호작용합니다.
2. 일부 앱을 통해 시뮬레이션을 보고 상호 작용하는 클라이언트를 작성합니다.
3. 시뮬레이션을 텍스트 파일로 구성합니다.
4. 앱 패키지와 시뮬레이션 구성을 서비스에 업로드합니다.
5. 시뮬레이션을 시작합니다.
6. 필요에 따라 사용자 지정 앱을 시작하고 중지합니다.
7. 클라이언트를 사용자 지정 또는 서비스 앱에 연결하여 시뮬레이션을 보거나 상호 작용합니다.
8. Amazon CloudWatch Logs에서 시뮬레이션 로그를 확인합니다.
9. 시뮬레이션을 중지합니다.
10. 시뮬레이션을 정리합니다.

시뮬레이션 스키마

시뮬레이션 스키마(또는 스키마)는 시뮬레이션에 대한 구성 정보가 포함된 YAML형식의 텍스트 파일입니다.는 시뮬레이션을 시작할 때 스키마를 SimSpace Weaver 사용합니다. SimSpace Weaver 앱 SDK 배포 가능 패키지에는 샘플 프로젝트의 스키마가 포함되어 있습니다. 이를 자체 스키마를 위한 시작점으로 사용할 수 있습니다. 시뮬레이션 스키마에 대한 자세한 내용은 SimSpace Weaver 시뮬레이션 스키마 참조 섹션을 참조하세요.

작업자 및 리소스 단위

작업자는 시뮬레이션을 실행하는 Amazon EC2 인스턴스입니다. 시뮬레이션 스키마에서 작업자 유형을 지정합니다.는 서비스가 사용하는 특정 Amazon EC2 인스턴스 유형에 작업자 유형을 SimSpace Weaver 매핑합니다.는 작업자를 SimSpace Weaver 시작 및 중지하고 작업자 간의 네트워크 통신을 관리합니다.는 각 시뮬레이션에 대해 작업자 세트를 SimSpace Weaver 시작합니다. 시뮬레이션마다 다른 작업자를 사용합니다.

작업자의 가용 컴퓨팅(프로세서 및 메모리) 용량은 컴퓨팅 리소스 단위(또는 리소스 단위)라는 논리적 단위로 구분됩니다. 리소스 단위는 고정된 양의 프로세서 및 메모리 용량을 나타냅니다.

> **Note**
>
> 이전에는 컴퓨팅 리소스 단위를 슬롯이라고 했습니다. 이 이전 용어는 설명서에서 계속 볼 수 있습니다.

시뮬레이션 클럭

각 시뮬레이션에는 자체 클럭이 있습니다. API 직접 호출 또는 SimSpace Weaver 콘솔을 사용하여 클럭을 시작하고 중지합니다. 시뮬레이션은 클럭이 실행 중일 때만 업데이트됩니다. 시뮬레이션의 모든 작업은 틱이라는 시간 세그먼트 내에서 이루어집니다. 클럭은 모든 작업자에게 각 틱의 시작 시간을 알려줍니다.

클럭 속도(또는 틱 속도)는 클럭이 알려주는 초당 틱 수(헤르츠 또는 Hz)입니다. 시뮬레이션에 필요한 클럭 속도는 시뮬레이션 스키마의 일부입니다. 틱에 대한 모든 작업은 다음 틱이 시작되기 전에 완료되어야 합니다. 이러한 이유로 유효 클럭 속도는 원하는 클럭 속도보다 낮을 수 있습니다. 유효 클럭 속도는 원하는 클럭 속도보다 높지 않습니다.

파티션

파티션은 작업자의 공유 메모리 세그먼트입니다. 각 파티션에는 시뮬레이션 상태 데이터의 일부가 들어 있습니다.

공간 앱의 파티션(공간 앱 파티션 또는 공간 파티션이라고도 함)에는 공간 앱의 소유권 영역에 있는 모든 개체가 포함됩니다.는 각 개체의 공간 위치를 기반으로 공간 앱 파티션에 개체를 SimSpace Weaver 입력합니다. 즉, SimSpace Weaver 는 공간적으로 서로 가까이 있는 엔터티를 동일한 작업자에 배치하려고 합니다. 이렇게 하면 앱이 소유한 엔터티를 시뮬레이션하기 위해 소유하지 않은 엔터티에 대해 필요한 지식의 양이 최소화됩니다.

상태 패브릭

상태 패브릭은 모든 작업자의 공유 메모리(모든 파티션 모음) 시스템입니다. 여기에는 시뮬레이션에 필요한 모든 상태 데이터가 들어 있습니다.

상태 패브릭은 엔터티를 해당 엔터티의 각 데이터 필드에 대한 초기 데이터 및 업데이트 로그 세트로 설명하는 사용자 지정 바이너리 형식을 사용합니다. 이 형식을 사용하면 시뮬레이션 시간의 이전 시점

의 엔터티 상태에 액세스하고 이를 실제 시간의 특정 시점으로 다시 매핑할 수 있습니다. 버퍼의 크기는 유한하며 버퍼에 있는 것 이상으로 시간을 되돌릴 수 없습니다.는 각 필드에 대한 업데이트 로그의 현재 오프셋에 포인터를 SimSpace Weaver 사용하고 필드 업데이트의 일부로 포인터를 업데이트합니다.는 공유 메모리를 사용하여 이러한 업데이트 로그를 앱의 프로세스 공간에 SimSpace Weaver 매핑합니다.

이 객체 형식은 오버헤드가 낮고 직렬화 비용이 발생하지 않습니다. SimSpace Weaver 또한는이 객체 형식을 사용하여 인덱스 필드(예: 개체 위치)를 구문 분석하고 식별합니다.

개체

엔터티는 시뮬레이션에서 가장 작은 데이터 구성 요소입니다. 엔터티의 예로는 작업자(예: 사람, 차량) 및 정적 객체(예: 건물 및 장애물)가 있습니다. 엔터티에는 SimSpace Weaver에 영구 데이터로 저장할 수 있는 속성(예: 위치 및 방향)이 있습니다. 엔터티는 파티션 내에 존재합니다.

앱

A SimSpace Weaver app은 각 시뮬레이션 틱을 실행하는 사용자 지정 로직이 포함된 작성하는 소프트웨어입니다. 대부분의 앱의 목적은 시뮬레이션이 실행될 때 엔터티를 업데이트하는 것입니다. 앱은 SimSpace Weaver 앱 SDK에서 APIs를 호출하여 시뮬레이션의 개체에 대한 작업(예: 읽기 및 업데이트)을 수행합니다.

앱과 필요한 리소스(예: 라이브러리)를 .zip 파일로 패키징하고에 업로드합니다 SimSpace Weaver. 작업자의 Docker 컨테이너에서 앱이 실행됩니다. SimSpace Weaver 는 작업자에 고정된 수의 리소스 단위를 각 앱에 할당합니다.

SimSpace Weaver 는 각 앱에 파티션 하나(및 하나만)의 소유권을 할당합니다. 앱과 해당 파티션은 동일한 작업자에 있습니다. 각 파티션에는 앱 소유자가 한 명만 있습니다. 앱은 파티션에서 항목을 만들고, 읽고, 업데이트하고, 삭제할 수 있습니다. 앱은 해당 파티션의 모든 엔터티를 소유합니다.

앱에는 공간 앱, 사용자 지정 앱, 서비스 앱의 세 가지 유형이 있습니다. 이는 사용 사례와 수명 주기에 따라 다릅니다.

> **Note**
>
> 에서 "app" SimSpace Weaver이라는 용어는 앱의 코드 또는 해당 코드의 실행 중인 인스턴스를 참조할 수 있습니다.

공간 앱

공간 앱은 시뮬레이션에 공간적으로 존재하는 엔터티의 상태를 업데이트합니다. 예를 들어 속도, 모양, 크기를 기반으로 모든 틱에 대해 엔터티를 이동하고 충돌시키는 역할을 하는 Physics 앱을 정의할 수 있습니다. 이 경우 SimSpace Weaver 는 Physics 앱의 여러 인스턴스를 병렬로 실행하여 워크로드 크기를 처리합니다.

SimSpace Weaver 는 공간 앱의 수명 주기를 관리합니다. 시뮬레이션 스키마에서 공간 앱 파티션의 배열을 지정합니다. 시뮬레이션을 시작하면 SimSpace Weaver 는 각 공간 앱 파티션에 대한 공간 앱을 시작합니다. 시뮬레이션을 중지하면가 공간 앱을 SimSpace Weaver 종료합니다.

다른 유형의 앱에서는 엔터티를 생성할 수 있지만 공간 앱에서만 엔터티를 업데이트할 수 있습니다. 다른 유형의 앱은 자신이 생성한 개체를 공간 도메인으로 전송해야 합니다.는 개체의 공간 위치를 SimSpace Weaver 사용하여 개체를 공간 앱의 파티션으로 이동합니다. 이렇게 하면 엔터티의 소유권이 공간 앱으로 이전됩니다.

사용자 지정 앱

사용자 지정 앱을 사용하여 시뮬레이션과 상호 작용합니다. 사용자 지정 앱은 구독을 사용하여 엔터티 데이터를 읽습니다. 사용자 지정 앱은 엔터티를 만들 수 있습니다. 하지만 시뮬레이션에 엔터티를 포함하고 업데이트하려면 앱이 엔터티를 공간 앱으로 전송해야 합니다. 사용자 지정 앱에 네트워크 엔드포인트를 SimSpace Weaver 할당할 수 있습니다. 시뮬레이션 클라이언트는 네트워크 엔드포인트에 연결하여 시뮬레이션과 상호 작용할 수 있습니다. 시뮬레이션 스키마에서 사용자 지정 앱을 정의하지만 SimSpace Weaver API 직접 호출을 사용하여 앱을 시작하고 중지할 책임은 사용자에게 있습니다. 작업자에서 사용자 지정 앱 인스턴스를 시작한 후에는 인스턴스를 다른 작업자에게 전송 SimSpace Weaver 하지 않습니다.

서비스 앱

모든 작업자에서 읽기 전용 프로세스를 실행해야 하는 경우 서비스 앱을 사용할 수 있습니다. 예를 들어 대규모 시뮬레이션이 있고 시뮬레이션을 진행하면서 보이는 엔터티만 사용자에게 보여주는 표시 중 클라이언트가 필요한 경우 서비스 앱을 사용할 수 있습니다. 이 경우 단일 사용자 지정 앱 인스턴스로는 시뮬레이션의 모든 엔터티를 처리할 수 없습니다. 서비스 앱을 모든 작업자에서 시작되도록 구성할 수 있습니다. 그러면 각 서비스 앱이 할당된 작업자의 엔터티를 필터링하고 관련 엔터티만 연결된 클라이언트에 보낼 수 있습니다. 그러면 표시 중 클라이언트가 시뮬레이션 공간을 통과하면서 다른 서비스 앱에 연결할 수 있습니다. 시뮬레이션 schema. SimSpace Weaver starts에서 서비스 앱을 구성합니다.

앱 요약

다음 표에는 SimSpace Weaver 앱의 각 유형에 대한 특성이 요약되어 있습니다.

	공간 앱	사용자 지정 앱	서비스 앱
엔터티 읽기	예	예	예
엔터티 업데이트	예	아니요	아니요
엔터티 생성	예	예*	예*
수명 주기	관리형(가SimSpace Weaver 제어합니다.)	비관리형(사용자 제어)	관리형(가SimSpace Weaver 제어합니다.)
시작 방법	SimSpace Weaver 는 스키마에 지정된 대로 각 공간 파티션에 대해 하나의 앱 인스턴스를 시작합니다.	각 앱 인스턴스를 시작합니다.	SimSpace Weaver 는 스키마에 지정된 대로 각 작업자에서 하나 이상의 앱 인스턴스를 시작합니다.
클라이언트가 연결 가능	아니요	예	예

* 사용자 지정 앱 또는 서비스 앱이 엔터티를 생성할 때 공간 앱이 엔터티의 상태를 업데이트할 수 있도록 앱은 엔터티의 소유권을 공간 앱으로 이전해야 합니다.

도메인

SimSpace Weaver 도메인은 동일한 실행 앱 코드를 실행하고 동일한 시작 옵션 및 명령을 갖는 앱 인스턴스의 모음입니다. 도메인은 포함된 앱 유형(공간 도메인, 사용자 지정 도메인, 서비스 도메인 등)별로 참조합니다. 도메인 내에서 앱을 구성합니다.

구독 및 복제

앱은 공간 영역에 대한 구독을 생성하여 해당 리전의 엔터티 이벤트(예: 입력, 종료, 생성, 업데이트, 삭제)를 학습합니다. 앱은 자신이 소유하지 않은 파티션에 있는 엔터티의 데이터를 읽기 전에 구독의 엔터티 이벤트를 처리합니다.

파티션은 앱과 동일한 작업자에 존재할 수 있지만(이를 로컬 파티션이라고 함) 다른 앱이 해당 파티션을 소유할 수 있습니다. 파티션은 다른 작업자(이를 원격 파티션이라고 함)에 존재할 수도 있습니다. 원격 파티션을 구독하는 경우 작업자는 복제라는 프로세스를 통해 원격 파티션의 로컬 복사본을 만듭니다. 그러면 작업자가 로컬 복사본(복제된 원격 파티션) 을 읽습니다. 작업자의 다른 앱이 동일한 틱으로 해당 파티션에서 데이터를 읽어야 하는 경우 작업자는 동일한 로컬 복사본을 읽습니다.

에 대한 사용 사례 예 SimSpace Weaver

공간 구성 요소를 사용하여 에이전트 기반 모델 및 개별 시간 단계 시뮬레이션 SimSpace Weaver 에를 사용할 수 있습니다.

대규모 군중 시뮬레이션 생성

SimSpace Weaver 를 사용하여 실제 환경에서 군중을 시뮬레이션할 수 있습니다. SimSpace Weaver 를 사용하면 고유한 동작으로 수백만 개의 동적 객체로 시뮬레이션을 확장할 수 있습니다.

도시 규모 환경 생성

SimSpace Weaver 를 사용하여 도시 전체의 디지털 트윈을 생성합니다. 도시 계획, 교통 경로 설계, 환경 위험 대응 계획을 위한 시뮬레이션을 생성합니다. 자체 지리공간 데이터 소스를 환경의 구성 요소로 사용할 수 있습니다.

몰입형 및 대화형 환경 생성

여러 사용자가 참여하고 상호 작용할 수 있는 시뮬레이션 환경을 생성합니다. Unreal Engine 및 Unity 와 같은 인기 개발 툴을 사용하여 3차원(3D) 가상 세계를 구축합니다. 자신만의 콘텐츠와 동작으로 3D 환경을 사용자 정의합니다.

AWS SimSpace Weaver 지원 종료

신중한 고려 후 2026년 5월 20 AWS SimSpace Weaver일부터 지원을 종료하기로 결정했습니다. AWS SimSpace Weaver 는 2025년 5월 20일부터 더 이상 신규 고객을 받지 않습니다. 2025년 5월 20일 이전에 서비스에 가입한 계정이 있는 기존 고객은 AWS SimSpace Weaver 기능을 계속 사용할 수 있습니다. 2026년 5월 20일 이후에는 더 이상를 사용할 수 없습니다 AWS SimSpace Weaver.

컨테이너화된 시뮬레이션을 실행 AWS Batch 하기 위해 로 전환하는 방법에 대한 자세한 내용은이 블로그 게시물을 참조하세요.

에 대한 설정 SimSpace Weaver

를 SimSpace Weaver 처음 사용하도록 설정하려면 AWS 계정 및 로컬 환경을 설정해야 합니다. 이 작업을 마치면 시작하기 자습서를 사용할 수 있습니다.

설정 작업

1. AWS 계정 를 사용하도록 설정 SimSpace Weaver.
2. 에 대한 로컬 환경 설정 SimSpace Weaver.

AWS 계정 를 사용하도록 설정 SimSpace Weaver

다음 작업을 완료하여를 AWS 계정 사용하도록 설정합니다 SimSpace Weaver.

에 가입 AWS 계정

이 없는 경우 다음 단계를 AWS 계정완료하여 생성합니다.

에 가입하려면 AWS 계정

1. https://portal.aws.amazon.com/billing/signup을 엽니다.
2. 온라인 지시 사항을 따르세요.

   등록 절차 중 전화 또는 텍스트 메시지를 받고 전화 키패드로 확인 코드를 입력하는 과정이 있습니다.

   에 가입하면 AWS 계정AWS 계정 루트 사용자이 생성됩니다. 루트 사용자에게는 계정의 모든 AWS 서비스 및 리소스에 액세스할 권한이 있습니다. 보안 모범 사례는 사용자에게 관리 액세스 권한을 할당하고, 루트 사용자만 사용하여 루트 사용자 액세스 권한이 필요한 작업을 수행하는 것입니다.

AWS 는 가입 프로세스가 완료된 후 확인 이메일을 보냅니다. 언제든지 https://aws.amazon.com/으로 이동하고 내 계정을 선택하여 현재 계정 활동을 확인하고 계정을 관리할 수 있습니다.

관리자 액세스 권한이 있는 사용자 생성

에 가입한 후 일상적인 작업에 루트 사용자를 사용하지 않도록 관리 사용자를 AWS 계정보호 AWS IAM Identity Center, AWS 계정 루트 사용자활성화 및 생성합니다.

보안 AWS 계정 루트 사용자

1. 루트 사용자를 선택하고 AWS 계정 이메일 주소를 입력하여 계정 소유자AWS Management Console로에 로그인합니다. 다음 페이지에서 비밀번호를 입력합니다.

   루트 사용자를 사용하여 로그인하는 데 도움이 필요하면 AWS 로그인 사용 설명서의 루트 사용자로 로그인을 참조하세요.
2. 루트 사용자의 다중 인증(MFA)을 활성화합니다.

   지침은 IAM 사용 설명서의 AWS 계정 루트 사용자(콘솔)에 대한 가상 MFA 디바이스 활성화를 참조하세요.

관리자 액세스 권한이 있는 사용자 생성

1. IAM Identity Center를 활성화합니다.

   지침은 AWS IAM Identity Center 사용 설명서의 AWS IAM Identity Center설정을 참조하세요.
2. IAM Identity Center에서 사용자에게 관리 액세스 권한을 부여합니다.

   를 자격 증명 소스 IAM Identity Center 디렉터리 로 사용하는 방법에 대한 자습서는 사용 AWS IAM Identity Center 설명서의 기본값으로 사용자 액세스 구성을 IAM Identity Center 디렉터리 참조하세요.

관리 액세스 권한이 있는 사용자로 로그인

- IAM IDentity Center 사용자로 로그인하려면 IAM Identity Center 사용자를 생성할 때 이메일 주소로 전송된 로그인 URL을 사용합니다.

  IAM Identity Center 사용자를 사용하여 로그인하는 데 도움이 필요하면 AWS 로그인 사용 설명서의 AWS 액세스 포털에 로그인을 참조하세요.

추가 사용자에게 액세스 권한 할당

1. IAM Identity Center에서 최소 권한 적용 모범 사례를 따르는 권한 세트를 생성합니다.

   지침은 AWS IAM Identity Center 사용 설명서의 Create a permission set를 참조하세요.
2. 사용자를 그룹에 할당하고, 그룹에 Single Sign-On 액세스 권한을 할당합니다.

   지침은 AWS IAM Identity Center 사용 설명서의 그룹 추가를 참조하세요.

사용할 권한 추가 SimSpace Weaver

액세스 권한을 제공하려면 사용자, 그룹 또는 역할에 권한을 추가하세요.

- 의 사용자 및 그룹 AWS IAM Identity Center:

  권한 세트를 생성합니다. AWS IAM Identity Center 사용자 안내서에서 권한 세트 생성의 지침을 따릅니다.
- ID 제공업체를 통해 IAM에서 관리되는 사용자:

  ID 페더레이션을 위한 역할을 생성합니다. IAM 사용자 설명서의 Create a role for a third-party identity provider (federation)의 지침을 따릅니다.
- IAM 사용자:
  - 사용자가 맡을 수 있는 역할을 생성합니다. IAM 사용자 설명서에서 Create a role for an IAM user의 지침을 따릅니다.
  - (권장되지 않음) 정책을 사용자에게 직접 연결하거나 사용자를 사용자 그룹에 추가합니다. IAM 사용 설명서에서 사용자(콘솔)에 권한 추가의 지침을 따릅니다.

Example사용 권한을 부여하는 IAM 정책 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"
                }
            }
        }
    ]
}
```

에 대한 로컬 환경 설정 SimSpace Weaver

SimSpace Weaver 시뮬레이션은 컨테이너화된 Amazon Linux 2 (AL2) 환경에서 실행됩니다. 앱을 컴파일하고 SimSpace Weaver 앱 SDK와 연결하려면 AL2 환경이 있어야 합니다. 표준 로컬 개발 환경은 Docker의 AL2 컨테이너입니다. Docker를 사용하지 않기로 선택한 경우 Windows Subsystem for Linux (WSL)에 AL2 환경을 실행하기 위한 대체 지침을 제공합니다. 자체 방법을 사용하여 로컬 AL2 환경을 만들 수도 있습니다. AL2를 로컬에서 실행하는 몇 가지 추가 방법은 Amazon EC2 설명서를 참조하세요.

> **Important**
>
> Microsoft Windows용 Docker는 표준 개발 환경입니다. 편의를 위해 로컬 개발 환경을 설정하는 다른 방법을 제안하지만 이러한 방법은 표준이 아니며 지원되지 않습니다.

주제

에서 Amazon Linux 2 (AL2)에 대한 SimSpace Weaver 배포 패키지 설정 Docker

이 섹션에서는에서 AL2 환경을 사용하여 로컬 SimSpace Weaver 배포 zip을 설정하는 지침을 제공합니다Docker. 에서 AL2로 설정하는 지침은 단원을 Windows Subsystem for Linux (WSL)참조하십시오에서 Amazon Linux 2 (AL2)에 대한 SimSpace Weaver 배포 패키지 설정 Windows Subsystem for Linux (WSL).

요구 사항

- Microsoft Windows 10 이상 또는 호환되는 Linux 시스템
- Microsoft Visual Studio 2019 이후(Desktop development with C++ 워크로드 설치)
- CMake3
- Git
- Docker Desktop
- AWS CLI
- Python 3.9

에서 AL2를 사용하여 SimSpace Weaver 배포 zip을 설정하려면 Docker

1. 에 대한 AWS 자격 증명을 아직 구성하지 않은 경우 AWS CLI 구성 지침을 AWS CLI따릅니다.
2. SimSpace Weaver 앱 SDK 배포 가능 패키지를 다운로드합니다. 이는 다음을 포함합니다.
   - SimSpace Weaver 앱 개발을 위한 바이너리 및 라이브러리
   - 개발 워크플로의 일부를 자동화하는 도우미 스크립트
   - SimSpace Weaver 개념을 보여주는 샘플 애플리케이션
3. 원하는 *sdk-folder*에 파일의 압축을 풉니다.
4. *sdk-folder*로 이동합니다.
5. 다음 명령을 입력하여 필요한 Python 패키지를 설치합니다.

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

6. 다음 명령을 입력하여 Docker 이미지로 SimSpace Weaver 배포를 설정합니다.

```
python setup.py
```

이 명령은 다음 작업을 수행합니다.

- 설치된 SimSpace Weaver 프로젝트 빌드에 대한 모든 요구 사항이 포함된 AL2 도커 이미지를 생성합니다.
- 시뮬레이션을 시작하는 데 필요한 CloudFormation 리소스를 생성합니다.
  - 샘플 CloudFormation 스택 템플릿은에서 찾을 수 있습니다. *sdk-folder*/PackagingTools/sample-stack-template.yaml
- 제공된 샘플 프로젝트를 로컬 시스템의 올바른 경로로 구성합니다.

문제 해결

- 도커가 멈춘 것처럼 보임
  - Docker 명령을 호출한 후 콘솔 출력이 멈춘 것처럼 보이면 Docker 엔진을 다시 시작해 보세요. 작동하지 않으면 컴퓨터를 다시 시작합니다.

에서 Amazon Linux 2 (AL2)에 대한 SimSpace Weaver 배포 패키지 설정 Windows Subsystem for Linux (WSL)

이 섹션에서는에서 AL2 환경을 사용하여 SimSpace Weaver 배포 zip을 설정하는 지침을 제공합니다 Windows Subsystem for Linux (WSL). Docker에 AL2를 설정하는 자세한 방법은 에서 Amazon Linux 2 (AL2)에 대한 SimSpace Weaver 배포 패키지 설정 Docker 섹션을 참조하세요.

> **Important**
>
> 이 섹션에서는 Amazon에서 소유, 개발 또는 지원하지 않는 AL2 버전을 사용하는 솔루션을 설명합니다. 이 솔루션은 Docker를 사용하지 않기로 선택한 경우에만 편의를 위해 제공됩니다. 이 솔루션을 사용하기로 선택한 경우 Amazon 및 AWS 는 책임을 지지 않습니다.

요구 사항

- Windows 10용 Hyper-V
- Windows Subsystem for Linux (WSL)
- WSL(다운로드 버전 2.0.20200722.0-update.2)에 대한 타사 오픈 소스 AL2 배포(지침 참조)

> **Important**
>
> WSL 지침에서는 WSL용 AL2 배포에 2.0.20200722.0-update.2 버전을 사용합니다. 다른 버전을 사용하는 경우 오류가 발생할 수 있습니다.

에서 AL2를 사용하여 SimSpace Weaver 배포 zip을 설정하려면 WSL

1. Windows 명령 프롬프트의 WSL에서 AL2 환경을 시작합니다.

   ```
   wsl -d Amazon2
   ```

   > **Important**
   >
   > 에서를 실행하는 동안에 있는 `quick-start.py` Python 헬퍼 스크립트 중 하나를 실행할 때 `--al2` 옵션을 WSL포함합니다`sdky-folder/Samples/sample-name/tools/cloud/quick-start.py`.

2. Linux 쉘 프롬프트에서 yum 패키지 관리자를 업데이트합니다.

   ```
   yum update -y
   ```

   > **Important**
   >
   > 이 단계의 제한 시간이 초과되면 WSL1로 전환하여 해당 절차를 다시 시도해야 할 수 있습니다. WSL AL2 세션을 종료하고 Windows 명령 프롬프트에 다음을 입력합니다.
   >
   > ```
   > wsl --set-version Amazon2 1
   > ```

3. 압축 해제 도구를 설치합니다.

```
yum install -y unzip
```

4. yum 설치된 AWS CLI 를 제거합니다. 를 yum 설치했는지 확실하지 않은 경우 다음 명령을 모두 시도하세요 AWS CLI.

```
yum remove awscli
```

```
yum remove aws-cli
```

5. 임시 디렉터리를 만들어 해당 디렉터리로 이동합니다.

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

6. AWS CLI다음을 다운로드하여 설치합니다.

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

7. 임시 디렉터리를 제거할 수 있습니다.

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

8. 쉘 세션을 다시 시작하여 환경의 경로를 업데이트합니다.

```
exec
```

9. AL2 환경에서 AWS CLI 에 대한 AWS 자격 증명을 구성합니다. 자세한 내용은 AWS CLI 구성을 참조하세요. 를 사용하는 경우 AWS Command Line Interface 사용 설명서의 AWS CLI 를 사용하도록 구성을 AWS IAM Identity Center AWS IAM Identity Center참조하세요.

```
aws configure
```

10. Git을 설치합니다.

```
yum install -y git
```

11. wget을 설치합니다.

```
yum install -y wget
```

12. SimSpace Weaver 앱 SDK용 폴더를 생성합니다.

```
mkdir sdk-folder
```

13. SDK 폴더로 이동합니다.

```
cd sdk-folder
```

14. SimSpace Weaver 앱 SDK 배포 가능 패키지를 다운로드합니다. 이는 다음을 포함합니다.

    - SimSpace Weaver 앱 개발을 위한 바이너리 및 라이브러리
    - 개발 워크플로의 일부를 자동화하는 도우미 스크립트
    - SimSpace Weaver 개념을 보여주는 샘플 애플리케이션

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

15. 파일 압축을 풉니다.

```
unzip *.zip
```

16. WSL 설정 스크립트를 실행합니다.

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

17. 다음 명령을 입력하여 필요한 Python 패키지를 설치합니다.

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

18. SimSpace Weaver 배포 zip 설정 스크립트를 실행합니다.

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

    이 명령은 다음 작업을 수행합니다.

    - 시뮬레이션을 시작하는 데 필요한 CloudFormation 리소스를 생성합니다.

  - 샘플 CloudFormation 스택 템플릿은에서 찾을 수 있습니다. *sdk-folder*/PackagingTools/sample-stack-template.yaml
- 제공된 샘플 프로젝트를 로컬 시스템의 올바른 경로로 구성합니다.

> **Note**
>
> WSL의 AL2 환경에 대해이 작업을 한 번만 수행하면 됩니다.

에서 라이선스 소프트웨어 사용 AWS SimSpace Weaver

AWS SimSpace Weaver 를 사용하면 선택한 시뮬레이션 엔진 및 콘텐츠로 시뮬레이션을 빌드할 수 있습니다. 사용과 관련하여 시뮬레이션에 사용하는 소프트웨어 또는 콘텐츠의 라이선스 조건을 획득, 유지 및 준수할 SimSpace Weaver책임은 사용자에게 있습니다. 라이선싱 계약에서 소프트웨어와 콘텐츠를 가상 호스팅 환경에서 배포하도록 허용하는지 확인합니다.

시작하기 SimSpace Weaver

이 섹션에서는 시작하는 데 도움이 되는 자습서를 제공합니다 SimSpace Weaver. 이 자습서에서는를 사용하여 시뮬레이션을 빌드하기 위한 일반적인 워크플로를 소개합니다 SimSpace Weaver. 이 자습서에서는에서 시뮬레이션을 생성, 배포 및 실행하는 방법을 보여줍니다 SimSpace Weaver. 몇 분 안에 시뮬레이션을 실행하려면 빠른 시작 자습서로 시작하는 것이 좋습니다. 그 후 다른 자습서를 살펴보고 자세히 알아보세요.

이 자습서에서는 설정 절차 중에 다운로드한 SimSpace Weaver 앱 SDK .zip 파일에 포함된 샘플 애플리케이션(`PathfindingSample`)을 사용합니다. 샘플 애플리케이션은 공간 파티셔닝, 파티션 간 개체 핸드오프, 앱 및 구독을 포함하여 모든 SimSpace Weaver 시뮬레이션이 공유하는 개념을 보여줍니다.

자습서에서는 네 개의 공간 파티션을 포함하는 시뮬레이션을 만들어 보겠습니다. `PathfindingSample` 공간 앱의 개별 인스턴스가 각 개별 파티션을 관리합니다. 공간 앱은 자체 파티션에 엔터티를 만듭니다. 엔터티는 이동 시 장애물을 피하면서 시뮬레이션 세계의 특정 위치로 이동합니다. 별도의 클라이언트 애플리케이션(SimSpace Weaver 앱 SDK에 포함됨)을 사용하여 시뮬레이션을 볼 수 있습니다.

주제

- 에 대한 빠른 시작 자습서 SimSpace Weaver
- 상세 자습서: 샘플 애플리케이션을 빌드하면서 세부 정보를 학습합니다.

에 대한 빠른 시작 자습서 SimSpace Weaver

이 자습서는 몇 분 안에 SimSpace Weaver 에서 시뮬레이션을 빌드하고 실행하는 프로세스를 안내합니다. 이 자습서로 시작한 다음 나중에 자세한 자습서를 진행하는 것이 좋습니다.

요구 사항

시작하기 전에 먼저 에 대한 설정 SimSpace Weaver의 단계를 완료해야 합니다.

> **Note**
>
> 여기에 사용된 스크립트는 사용자의 편의를 위해 제공되며 필수는 아닙니다. 이러한 단계를 수동으로 수행하는 방법은 자세한 자습서를 참조하세요.

1단계: 로깅 활성화(선택 사항)

로깅 켜기

1. 다음으로 이동합니다.

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

2. 텍스트 편집기에서 스키마 파일을 엽니다.

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

3. 파일 시작 부분에 있는 `simulation_properties:` 섹션을 찾습니다.

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

4. `simulation_properties:` 줄 뒤에 다음 두 줄을 삽입합니다.

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

5. `simulation_properties:` 섹션이 다음과 같은지 확인합니다.

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

6. 파일을 저장하고 텍스트 편집기를 종료합니다.

2단계: 콘솔 클라이언트로 빠른 시작(옵션 1)

다음으로 이동합니다.

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

다음 명령 중 하나를 실행합니다.

- Docker: `python quick-start.py --consoleclient`

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

기본적으로 단일 작업자에서 단일 파티션으로 시뮬레이션을 시작합니다. `/Samples/PathfindingSample/tools/` 폴더`--schema {file name}.yaml`에서를 전달하여 다른 구성을 시작할 수 있습니다.

> **Note**
>
> 이 스크립트의 기능에 상세 자습서: 샘플 애플리케이션을 빌드하면서 세부 정보를 학습합니다. 대한 자세한 설명은 섹션을 참조하세요.

2단계: Unreal Engine 클라이언트로 빠른 시작(옵션 2)

Unreal Engine 뷰 클라이언트 시작을 참조하세요.

시뮬레이션 중지 및 삭제

다음으로 이동합니다.

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

시뮬레이션의 이름을 찾습니다.

```
aws simspaceweaver list-simulations
```

시뮬레이션 중지 및 삭제

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

문제 해결

- FileNotFoundError: cmake

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

  - 해결 방법: 스크립트가 명령을 찾을 수 없습니다cmake. 최소 권장 CMake 버전이 설치되어 있고 PATH의 cmake 명령으로 호출할 수 있는지 확인하세요. 명령을 사용하여 cmake -version 확인합니다.
- ImportError: libweaver_app_sdk_python_v1을 가져오는 동안 DLL 로드가 실패했습니다. 지정된 모듈을 찾을 수 없습니다.
  - 해결 방법:이 오류는 Python 3.9를 사용하여 Weaver Python SDK를 시작하지 않을 때 발생합니다. “python” 명령과 연결된 python 버전이 Python 3.9인지 확인하세요. python --version 명령을 실행하여 확인할 수 있습니다.
- Docker 빌드를 시작한 후 빠른 시작 스크립트가에 멈춰 있습니다.
  - 해결 방법: Docker가 워밍업하는 데 몇 분 정도 걸리는 경우가 있습니다. 이 문제가 약 5분 이상 지속되면 Docker 또는 시스템을 다시 시작하십시오.
- target_compile_features CXX 컴파일러 "GNU"에 대해 알려진 기능이 없습니다.
  - 해결 방법: 도커 캐시를 지우고, Weaverappbuilder 도커 이미지를 삭제하고, 프로젝트 빌드 아티팩트를 삭제하고,를 다시 실행합니다setup.py. 이렇게 하면 Docker 환경이 재설정되고 오류가 해결됩니다.

상세 자습서: 샘플 애플리케이션을 빌드하면서 세부 정보를 학습합니다.

빠른 시작 자습서에서는 quick-start.py 및를 사용하여 샘플 시뮬레이션을 빌드, 시작, 중지 및 삭제하는 방법을 다루었습니다stop-and-delete.py. 이 자습서에서는 이러한 스크립트의 작동 방식과 이러한 스크립트가 사용자 지정 Weaver 시뮬레이션의 유연성을 극대화하는 데 사용할 수 있는 추가 파라미터를 자세히 살펴봅니다.

요구 사항

시작하기 전에 먼저 에 대한 설정 SimSpace Weaver의 단계를 완료해야 합니다.

1단계: 로깅 활성화(선택 사항)

로깅 켜기

1. 다음으로 이동합니다.

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

2. 텍스트 편집기에서 스키마 파일을 엽니다.

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

3. 파일 시작 부분에 있는 `simulation_properties:` 섹션을 찾습니다.

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

4. `simulation_properties:` 줄 뒤에 다음 두 줄을 삽입합니다.

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

5. `simulation_properties:` 섹션이 다음과 같은지 확인합니다.

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

6. 파일을 저장하고 텍스트 편집기를 종료합니다.

2단계: 시뮬레이션 시작

빠른 시작 자습서에서 볼 수 있듯이 샘플 시뮬레이션을 시작하는 가장 기본적인 단계는 다음과 같습니다.

1. 다음으로 이동합니다.

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

2. 다음 명령 중 하나를 실행합니다.

   - Docker: `python quick-start.py`
   - WSL: `python quick-start.py --al2`

이 스크립트는 일반적인 터미널 명령을 자동화하며,이 모든 명령은를 사용하여 수동으로 실행할 수 있습니다 AWS CLI. 이 단계는 다음과 같습니다.

1. Weaver 스키마를 S3에 업로드합니다.

   - SimSpace Weaver 는 스키마를 사용하여 시뮬레이션을 구성합니다. 스키마는 YAML 형식의 일반 텍스트 파일입니다. 자세한 내용은 시뮬레이션 구성 섹션을 참조하세요.

2. 사용자 지정 컨테이너를 빌드하고 업로드합니다(선택 사항).

   - 스키마가 사용자 지정 컨테이너를 정의하는 경우 빠른 시작 스크립트는 도커 이미지를 빌드하여 Amazon ECR에 업로드합니다. 자세한 내용은 사용자 지정 컨테이너 단원을 참조하십시오. 이 기능의 예는 `PythonBubblesSample` 스키마를 참조하십시오.

3. 프로젝트를 빌드합니다.

   - `quick-start.py`는에 정의된 `build_project` 함수를 호출합니다`build.py`. 이 단계는 프로젝트에 따라 달라집니다. PathfindingSample의 경우 CMake가 사용됩니다. 에서 찾을 수 있는 CMake 및 Docker 명령입니다`build.py`.

4. 빌드 아티팩트를 S3에 업로드합니다.

   - S3 버킷을 확인하여 모든 업로드가 성공했는지 확인할 수 있습니다. Amazon S3 사용에 대한 자세한 내용은 Amazon Simple Storage Service 사용 설명서의 Amazon S3 버킷 생성, 구성, 작업을 참조하세요.
   - 샘플 애플리케이션 zip 및 S3 버킷은 다음 이름 형식을 사용합니다.
     - `weaver-sample-bucket-account-number-region`
     - 공간 앱: `ProjectNameSpatial.zip`
     - 보기(사용자 지정) 앱: `ProjectNameView.zip`

5. 시뮬레이션을 시작합니다.

   - 통화 주변의 래퍼입니다`aws simspaceweaver start-simulation` AWS CLI . 자세한 내용은 AWS CLI 명령 참조를 참조하세요 SimSpace Weaver.
   - 시뮬레이션 상태가 `STARTED` 또는 `FAILED` 상태가 될 때까지 스크립트가 반복됩니다. 시뮬레이션을 시작하는 데 몇 분 정도 걸릴 수 있습니다.

6. 시뮬레이션 세부 정보를 가져옵니다.

   - DescribeSimulation API는 상태를 포함하여 시뮬레이션에 대한 세부 정보를 제공합니다. 시뮬레이션은 다음 상태 중 하나일 수 있습니다.

     시뮬레이션 수명 주기 상태

     1. **`STARTING`** - StartSimulation 호출 후 초기 상태

2. **STARTED** - 모든 공간 앱이 실행되고 정상임
3. **STOPPING** - StopSimulation 호출 후 초기 상태
4. **STOPPED** - 모든 컴퓨팅 리소스가 중지됨
5. **DELETING** - DeleteSimulation 호출 후 초기 상태
6. **DELETED** - 시뮬레이션에 할당된 모든 리소스가 삭제됨
7. **FAILED** - 시뮬레이션에 심각한 오류/장애가 발생하여 중지됨
8. **SNAPSHOT_IN_PROGRESS** - 스냅샷이 진행 중임

시뮬레이션 세부 정보 가져오기

1. ListSimulations API를 직접적으로 호출합니다.

   ```
   aws simspaceweaver list-simulations
   ```

   스크립트에는 다음과 같이 각 시뮬레이션에 대한 세부 정보가 표시되어야 합니다.

   ```
   {
       "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. DescribeSimulation을 호출하여 시뮬레이션 세부 정보를 가져옵니다. *simulation-name*을 이전 단계의 출력에서 시뮬레이션의 Name으로 바꿉니다.

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

   스크립트에는 다음과 같이 지정된 시뮬레이션에 대한 세부 정보가 표시되어야 합니다.

   ```
   {
       "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. 사용자 지정 앱을 시작합니다.

   - SimSpace Weaver 는 사용자 지정 앱의 수명 주기를 관리하지 않습니다. 사용자 지정 앱을 시작해야 합니다. 시뮬레이션 클럭을 시작하기 전에 사용자 지정 앱을 시작하는 것이 가장 좋지만, 클럭을 시작한 후에 사용자 지정 앱을 시작할 수도 있습니다.

     StartApp API를 호출하여 사용자 지정 앱을 시작할 수 있습니다.

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

     StartApp API 직접 호출은 사용자가 제공한 이름을 사용하여 사용자 지정 앱의 새 인스턴스를 만들고 시작합니다. 이미 존재하는 앱의 이름을 입력하면 오류가 발생합니다. 특정 앱(인스턴스)을 다시 시작하려면 먼저 해당 앱을 중지하고 삭제해야 합니다.

     > **Note**
     >
     > 시뮬레이션 상태가 STARTED여야 사용자 지정 앱을 시작할 수 있습니다.

     샘플 애플리케이션은 시뮬레이션을 볼 수 있는 ViewApp 사용자 지정 앱을 제공합니다. 이 앱은 시뮬레이션 클라이언트를 연결하는 데 필요한 고정 IP 주소와 포트 번호를 제공합니다(이 방법은 이 자습서의 이후 단계에서 설명). domain은 실행 코드와 시작 옵션이 동일한 앱 클래스라고 생각할 수 있습니다. app name은 앱의 인스턴스를 식별합니다. SimSpace Weaver 개념에 대한 자세한 내용은 단원을 참조하십시오의 주요 개념 SimSpace Weaver.

     DescribeApp API를 사용하여 사용자 지정 앱을 시작한 후 상태를 확인할 수 있습니다.

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

     이 자습서에서 보기 앱 시작

     1. StartApp에 대해를 호출합니다ViewApp.

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

2. DescribeApp을 호출하여 사용자 지정 앱의 상태를 확인합니다.

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

사용자 지정 앱(인스턴스)의 상태가 STARTED이면 DescribeApp의 출력에는 해당 사용자 지정 앱(인스턴스)의 IP 주소 및 포트 번호가 포함됩니다. 다음 예제 출력에서 IP 주소는 Address 값이고 포트 번호는 EndpointInfo 블록의 Actual 값입니다.

```
{
    "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`의 값은 앱 코드가 바인딩되어야 하는 포트 번호입니다. 의 값은가 클라이언트에 SimSpace Weaver 노출하여 앱에 연결하는 포트 번호`Actual`입니다.는 `Declared`포트를 `Actual` 포트에 SimSpace Weaver 매핑합니다.

> **Note**
>
> 에 설명된 절차를 사용하여 시작된 사용자 지정 앱의 IP 주소와 포트 번호를 사용자 지정 앱의 IP 주소 및 포트 번호 가져오기 가져올 수 있습니다.

8. 시계를 시작합니다.

- 시뮬레이션을 처음 만들면 클럭이 있지만 실행되지는 않습니다. 클럭이 실행되지 않을 때는 시뮬레이션이 클럭의 상태를 업데이트하지 않습니다. 클럭을 시작하면 앱에 틱이 전송되기 시작합니다. 각 틱마다 공간 앱은 자신이 소유한 엔터티를 단계별로 살펴보고 결과를에 커밋합니다. SimSpace Weaver

> **Note**
>
> 클럭을 시작하는 데 30~60초가 걸릴 수 있습니다.

StartClock API를 직접적으로 호출합니다.

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

> **Note**
>
> StartClock API는 ListSimulations API를 사용하여 찾을 수 있는 *simulation-name*을 사용합니다.
>
> ```
> aws simspaceweaver list-simulations
> ```

빠른 시작 파라미터

- -h, --help
  - 이러한 파라미터를 나열합니다.
- --정리
  - 빌드하기 전에 빌드 디렉터리의 내용을 삭제합니다.
- --al2
  - Docker 대신 네이티브 머신에 직접 빌드됩니다. WSL과 같은 Amazon Linux 2 환경에서 실행되는 경우에만이 옵션을 사용합니다.
- --uploadonly
  - 스키마와 앱 zip만 Amazon S3에 업로드하고 시뮬레이션을 시작하지 마세요.
- --nobuild
  - 프로젝트 재구축을 건너뜁니다.
- --nocontainer
  - 스키마에 나열된 시뮬레이션 컨테이너 재구축을 건너뜁니다.
- --콘솔리언트
  - config.py 나열된 콘솔 클라이언트를 자동으로 빌드하고 연결합니다.
- --스키마 SCHEMA
  - 이 호출에서 사용할 스키마입니다. config.py 기본값은 'SCHEMA'입니다.
- --name NAME
  - 시뮬레이션의 이름입니다. 기본값은 config.py 'PROJECT_NAME'-date-time입니다.

3단계: 로그 확인(선택 사항)

SimSpace Weaver 는 앱의 시뮬레이션 관리 메시지와 콘솔 출력을 Amazon CloudWatch Logs에 기록합니다. 로그 작업에 대한 자세한 내용은 Amazon CloudWatch Logs 사용 설명서의 로그 그룹 및 로그 스트림 작업을 참조하세요.

CloudWatch Logs에서 생성하는 각 시뮬레이션에는 자체 로그 그룹이 있습니다. 로그 그룹의 이름은 시뮬레이션 스키마에 지정됩니다. 다음 스키마 스니펫에서 `log_destination_service`의 값은 `logs`입니다. 즉, `log_destination_resource_name`의 값은 로그 그룹의 이름입니다. 이 경우 로그 그룹은 `MySimulationLogs`입니다.

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

시뮬레이션을 시작한 후 DescribeSimulation API를 사용하여 시뮬레이션에 사용할 로그 그룹의 이름을 찾을 수도 있습니다.

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

다음 예제는 로깅 구성을 설명하는 DescribeSimulation의 출력 부분을 보여줍니다. 로그 그룹의 이름은 LogGroupArn 끝에 표시됩니다.

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

각 시뮬레이션 로그 그룹에는 여러 로그 스트림이 포함되어 있습니다.

- 관리 로그 스트림 - SimSpace Weaver 서비스에서 생성된 시뮬레이션 관리 메시지입니다.

  ```
  /sim/management
  ```

- 오류 로그 스트림 - SimSpace Weaver 서비스에서 생성된 오류 메시지입니다. 이 로그 스트림은 오류가 있는 경우에만 존재합니다.는 앱에서 작성한 오류를 자체 앱 로그 스트림에 SimSpace Weaver 저장합니다(다음 로그 스트림 참조).

  ```
  /sim/errors
  ```

- 공간 앱 로그 스트림(각 작업자의 공간 앱당 1개) - 공간 앱에서 생성된 콘솔 출력입니다. 각 공간 앱은 자체 로그 스트림에 씁니다. *spatial-app-id*는 *worker-id* 끝의 후행 슬래시 뒤에 오는 모든 문자입니다.

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

- 사용자 지정 앱 로그 스트림(각 사용자 지정 앱 인스턴스당 1개) - 사용자 지정 앱에서 생성된 콘솔 출력입니다. 각 사용자 지정 앱 인스턴스는 자체 로그 스트림에 씁니다.

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

- 서비스 앱 로그 스트림(각 서비스 앱 인스턴스당 1개) - 서비스 앱에서 생성된 콘솔 출력입니다. 각 공간 앱은 자체 로그 스트림에 씁니다. *service-app-id*는 *service-app-name* 끝의 후행 슬래시 뒤에 오는 모든 문자입니다.

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

> **Note**
>
> 샘플 애플리케이션에는 서비스 앱이 없습니다.

4단계: 시뮬레이션 보기

SimSpace Weaver 앱 SDK는 샘플 애플리케이션을 볼 수 있는 다양한 옵션을 제공합니다. Unreal Engine 개발에 대한 로컬 지원이 없는 경우 샘플 콘솔 클라이언트를 사용할 수 있습니다. Unreal Engine 클라이언트에 대한 지침은 Windows를 사용한다고 가정합니다.

콘솔 클라이언트는 발생하는 엔터티 이벤트 목록을 표시합니다. 클라이언트는 `ViewApp`에서 엔터티 이벤트 정보를 가져옵니다. 콘솔 클라이언트가 이벤트 목록을 표시하면 시뮬레이션의 `ViewApp` 및 활동을 통해 네트워크 연결을 확인합니다.

`PathfindingSample` 시뮬레이션은 2차원 평면에 고정된 엔터티와 움직이는 엔터티를 만듭니다. 움직이는 엔터티는 고정된 엔터티 주위를 이동합니다. Unreal Engine 클라이언트는 엔터티 이벤트를 시각화합니다.

콘솔 클라이언트

`--consoleclient` 옵션을 포함`quick-start.py`하면에서 샘플을 시작할 때 콘솔 클라이언트를 자동으로 빌드하고 연결할 수 있습니다. 가 이미 호출된 후 콘솔 클라이언트를 빌드하고 연결하려면 다음을 수행합니다`quick-start.py`.

다음으로 이동합니다.

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

스크립트를 실행하여 클라이언트를 빌드하고 연결합니다.

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

스크립트는 다음을 수행합니다.

1. CMake를 사용하여 콘솔 클라이언트를 빌드합니다.
2. 지정된 IP 주소와 포트 번호로 빌드된 실행 파일을 시작합니다.

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

Unreal Engine 클라이언트

Unreal Engine 뷰 클라이언트 시작을(를) 참조하세요.

5단계: 시뮬레이션 중지 및 삭제

다음으로 이동합니다.

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

시뮬레이션의 이름을 찾습니다.

```
aws simspaceweaver list-simulations
```

시뮬레이션을 중지하고 삭제합니다.

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

스크립트는 다음을 수행합니다`stop-and-delete.py`.

1. AWS CLI 명령을 호출하여 시뮬레이션을 중지합니다.
   - `aws simspaceweaver stop-simulation`
   - 자세한 내용은 AWS CLI 명령 참조를 참조하세요 SimSpace Weaver.
2. AWS CLI 명령을 호출하여 시뮬레이션을 삭제합니다.
   - `aws simpaceweaver delete-simulation`
   - 자세한 내용은 AWS CLI 명령 참조를 참조하세요 SimSpace Weaver.

stop-and-delete 파라미터

- -h, --help
  - 이러한 파라미터를 나열합니다.
- --시뮬레이션 시뮬레이션
  - stop-and-delete 시뮬레이션의 이름입니다.
- --stop
  - 시뮬레이션만 중지합니다. 삭제하지 않습니다.
- - 삭제
  - 시뮬레이션만 삭제합니다. 시뮬레이션이 `STOPPED` 또는 인 경우에만 작동합니다`FAILED`.

문제 해결

빠른 시작 자습서문제 해결의 섹션을 참조하세요.

작업 SimSpace Weaver

이 장에서는 SimSpace Weaver에서 자체 애플리케이션을 구축하는 데 도움이 되는 정보와 지침을 제공합니다.

주제

시뮬레이션 구성

시뮬레이션 스키마(또는 스키마)는 시뮬레이션 구성을 지정하는 YAML 형식이 지정된 텍스트 파일입니다. 동일한 스키마를 사용하여 여러 시뮬레이션을 시작할 수 있습니다. 스키마 파일은 시뮬레이션을 위한 프로젝트 폴더에 있습니다. 텍스트 편집기를 사용하여 파일을 편집할 수 있습니다.는 시뮬레이션을 시작할 때 SimSpace Weaver 만 스키마를 읽습니다. 스키마 파일을 편집하면 편집 후에 시작하는 새 시뮬레이션에만 영향을 줍니다.

시뮬레이션을 구성하려면 시뮬레이션 스키마 파일을 편집합니다(운영 체제에 적합한 경로 구분자 사용).

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

새 시뮬레이션을 생성할 때 시뮬레이션 스키마를 업로드합니다. 프로젝트의 빠른 시작 도우미 스크립트는 시뮬레이션을 빌드하는 프로세스의 일부로 스키마를 업로드합니다.

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

빠른 시작 스크립트 실행에 대한 자세한 내용은이 가이드의 시작 장상세 자습서에 있는 단원을 참조하십시오.

시뮬레이션 구성 파라미터

시뮬레이션 스키마에는 다음과 같은 자동 구성 정보가 포함됩니다.

- 시뮬레이션 속성 - SDK 버전 및 컴퓨팅 구성(작업자 유형 및 수)
- 클럭 - 틱 속도 및 허용오차
- 공간 분할 전략 - 공간 토폴로지(예: 그리드), 경계 및 배치 그룹(작업자에 대한 공간 분할 그룹화)
- 도메인 및 해당 앱 - 앱 버킷, 경로, 실행 명령

SimSpace Weaver 는 스키마 구성을 사용하여 공간 파티션을 구성 및 정렬하고, 앱을 시작하고, 지정된 틱 속도로 시뮬레이션을 진행합니다.

> **Note**
>
> SimSpace Weaver 앱 SDK의 create-project 스크립트는 샘플 애플리케이션을 기반으로 시뮬레이션 스키마를 자동으로 생성합니다.

다음 주제에서는 시뮬레이션 스키마의 파라미터에 대해 설명합니다. 시뮬레이션 스키마에 대한 전체 설명은 SimSpace Weaver 시뮬레이션 스키마 참조 섹션을 참조하세요.

주제

- SDK 버전

- 시뮬레이션 속성
- 작업자
- 클럭
- 파티셔닝 전략
- 도메인

SDK 버전

`sdk_version` 필드는 스키마의 형식 SimSpace Weaver 이 지정된 버전을 지정합니다. 유효한 값: `1.17`, `1.16`, `1.15`, `1.14`, `1.13`, `1.12`

> **Important**
>
> `sdk_version`의 값에는 메이저 버전 번호와 첫 번째 마이너 버전 번호만 포함됩니다. 예를 들어, `1.12` 값은 `1.12.0`, `1.12.1`, `1.12.2` 등의 모든 버전 `1.12.x`를 지정합니다.

시뮬레이션 속성

스키마 `simulation_properties` 섹션은 엔터티의 인덱스 필드(일반적으로 공간 위치)에 대한 로깅 구성 및 데이터 유형을 지정합니다.

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

`log_destination_service`의 값에 따라 `log_destination_resource_name` 값의 해석이 결정됩니다. 현재 지원되는 값은 `logs`입니다. 즉, `log_destination_resource_name`의 값은 Amazon CloudWatch Logs의 로그 그룹 이름입니다.

> **Note**
>
> 로깅은 선택 사항입니다. 로그 대상 속성을 구성하지 않으면 시뮬레이션에서 로그가 생성되지 않습니다.

default_entity_index_key_type 속성은 필수입니다. 유일한 유효 값은 Vector3<f32>입니다.

작업자

workers 섹션은 시뮬레이션에 사용할 작업자의 유형과 수를 지정합니다.는 Amazon EC2 인스턴스 유형에 매핑되는 자체 작업자 유형을 SimSpace Weaver 사용합니다.

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

다중 작업자 시뮬레이션 지원

하나 이상의 작업자를 사용하는 시뮬레이션을 생성할 수 있습니다. 기본적으로 시뮬레이션에서는 하나의 작업자를 사용합니다. 시뮬레이션을 시작하기 전에 시뮬레이션 스키마를 수정해야 합니다.

> **Note**
>
> 이미 시작된 시뮬레이션은 변경할 수 없습니다. 실행 중인 시뮬레이션에 대해 다중 작업자를 활성화하려면 먼저 시뮬레이션을 중지하고 삭제해야 합니다.

둘 이상의 작업자를 사용하려면 컴퓨팅 인스턴스의 desired 수를 1보다 큰 값으로 설정합니다. 각 작업자에는 최대 앱 수가 있습니다. 자세한 내용은 단원을 참조하십시오SimSpace Weaver 엔드포인트 및 할당량. SimSpace Weaver 는 작업자의 앱 수가이 제한을 초과할 때 1명 이상의 작업자만 사용합니다.는 사용 가능한 작업자 중 하나에 앱을 배치할 SimSpace Weaver 수 있습니다. 특정 작업자에 대한 앱 배치는 보장되지 않습니다.

다음 스키마 스니펫은 작업자 둘을 요청하는 시뮬레이션의 구성을 보여줍니다. SimSpace Weaver 는 앱 수가 한 작업자의 최대 앱 수를 초과하는 경우 두 번째 작업자 할당을 시도합니다.

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

클럭

이 clock 섹션에서는 시뮬레이션 클럭의 속성을 지정합니다. 현재는 틱 속도(클럭이 앱으로 보내는 초당 틱 수)만 구성할 수 있습니다. 틱 속도는 최대 속도입니다. 틱에 대한 모든 작업(예: 엔터티 업데이트)이 완료되어야 다음 틱이 시작되기 때문에 유효 틱 속도는 더 낮을 수 있습니다. 틱 속도는 클럭 속도라고도 합니다.

tick_rate의 유효한 값은 스키마에 지정된 sdk_version에 따라 달라집니다.

틱 속도의 유효 값

- "1.14" 이전 버전:
  - 10
  - 15
  - 30
- "1.14" 이후 이상:
  - "10"
  - "15"
  - "30"
  - "unlimited"

    자세한 내용은 무제한 틱 속도 단원을 참조하십시오.

> **Important**
>
> - "1.14" 이전 sdk_version의 스키마의 경우 tick_rate 값은 30과 같은 정수입니다.
> - "1.14" 이후 sdk_version의 스키마의 경우 tick_rate 값은 "30"과 같은 문자열입니다. 값에는 큰따옴표가 포함되어야 합니다.
>
>   버전 "1.12" 또는 "1.13" 스키마를 "1.14" 버전 이상으로 변환하는 경우 tick_rate의 값을 큰따옴표로 묶어야 합니다.

무제한 틱 속도

`tick_rate`를 `"unlimited"`로 설정하여 코드를 실행할 수 있는 속도만큼 빠르게 시뮬레이션을 실행할 수 있습니다. 무제한 틱 속도를 사용하면 모든 앱이 현재 틱에 대한 커밋을 완료한 직후 다음 틱을 SimSpace Weaver 보냅니다.

> **Important**
>
> 1.14.0 이전 SimSpace Weaver 버전에서는 무제한 틱 속도가 지원되지 않습니다. 스키마 `sdk_version`의 최소값은 `"1.14"`입니다.

SimSpace Weaver Local의 무제한 틱 속도

SimSpace Weaver Local은 스키마가 10kHz(10000)의 틱 속도를 지정한 것처럼 `"unlimited"`를 구현합니다. 효과는 AWS 클라우드에서 무제한 틱 속도를 적용한 것과 같습니다. 여전히 스키마에 `tick_rate: "unlimited"`를 지정합니다. SimSpace Weaver Local에 대한 자세한 정보는 의 로컬 개발 SimSpace Weaver 섹션을 참조하세요.

클럭에 대한 FAQ

Q1. 다른 틱 속도를 사용하도록 STARTED 시뮬레이션을 변경할 수 있나요?

수명 주기의 어떤 단계에서도 AWS 클라우드 에 이미 존재하는 시뮬레이션의 틱 속도를 변경할 수 없습니다. 또한 SimSpace Weaver Local에서 실행 중인 시뮬레이션의 틱 속도도 변경할 수 없습니다. 스키마에서 `tick_rate`를 설정하고 해당 스키마에서 새 시뮬레이션을 시작할 수 있습니다.

Q2. 1.14 이전 버전에서 무제한 틱 속도로 시뮬레이션을 실행할 수 있나요?

아니요, 1.14.0 이전 버전에서는 무제한 틱 속도가 지원되지 않습니다.

클럭 오류 해결

시뮬레이션이 시작되지 않는 경우 DescribeSimulation API의 출력에서 `"StartError"` 값을 확인할 수 있습니다. 스키마의 `tick_rate` 값이 유효하지 않으면 다음과 같은 오류가 발생합니다.

> **Note**
>
> 여기에 표시된 오류 출력은 가독성을 높이기 위해 여러 줄에 표시됩니다. 실제 오류 출력은 한 줄입니다.

- sdk_version은 "1.14" 이전이고 tick_rate 값은 유효하지 않은 정수입니다. 유효한 값: 10, 15, 30

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

- sdk_version은 "1.14" 이전이고 tick_rate 값은 문자열입니다. 유효한 값: 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_version은 "1.14" 이후이고 tick_rate 값은 유효하지 않은 문자열입니다. 유효값: "10", "15", "30", "unlimited"

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

- sdk_version은 "1.14" 이후이고 tick_rate 값은 정수입니다. 유효값: "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\"}]"
```

파티셔닝 전략

partitioning_strategies 섹션에서는 공간 앱 파티션의 구성 속성을 지정합니다. 파티셔닝 전략에 고유한 이름(이 섹션의 속성 세트)을 입력하고 공간 앱 구성에 해당 이름을 사용합니다.

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

```
      y: [0, 1000]
    grid_placement_groups:
      x: 1
      y: 1
```

topology 속성은 시뮬레이션에서 사용하는 좌표계의 유형을 지정합니다. Grid 값은 2차원(2D) 그리드를 지정합니다.

Grid 토폴로지의 경우 시뮬레이션 공간은 축으로 정렬된 경계 상자 (AABB)로 모델링됩니다. aabb_bounds 속성에서 AABB의 각 축에 대한 좌표 경계를 지정합니다. 시뮬레이션에서 공간적으로 존재하는 모든 엔터티는 AABB 내에 위치해야 합니다.

그리드 배치 그룹

배치 그룹은 동일한 작업자에 배치 SimSpace Weaver 하려는 공간 앱 파티션의 모음입니다. grid_placement_groups 속성에서 배치 그룹(그리드 내)의 수와 배열을 지정합니다. SimSpace Weaver 는 배치 그룹 전체에 파티션을 균등하게 분산하려고 시도합니다. 동일한 배치 그룹에 파티션이 있는 공간 앱의 소유권 영역은 공간적으로 인접합니다.

x* y는 원하는 작업자 수와 같게 설정하는 것이 좋습니다. 같지 않은 경우 SimSpace Weaver 는 사용 가능한 작업자 간에 배치 그룹의 균형을 맞추려고 시도합니다.

배치 그룹 구성을 지정하지 않으면가 자동으로 배치 그룹 구성을 SimSpace Weaver 계산합니다.

도메인

도메인에 대한 구성 속성 집합의 이름을 제공합니다. 도메인의 앱 시작 설정에 따라 도메인 유형이 결정됩니다.

- **launch_apps_via_start_app_call** - 사용자 지정 도메인
- **launch_apps_by_partitioning_strategy** - 공간 도메인
- **launch_apps_per_worker**(샘플 애플리케이션에 포함되지 않음) - 서비스 도메인

> **Important**
>
> SimSpace Weaver 는 각 시뮬레이션에 대해 최대 5개의 도메인을 지원합니다. 여기에는 모든 공간, 사용자 지정 및 서비스 도메인이 포함됩니다.

```
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 앱 SDK 버전 1.12.x 프로젝트는 앱 .zip 파일 및 스키마에 대해 별도의 버킷을 사용합니다.

- weaver-*lowercase-project-name*-*account-number*-app-zips-*region*
- weaver-*lowercase-project-name*-*account-number*-schemas-*region*

주제

앱 구성

앱(app_config)의 구성을 해당 도메인 구성의 일부로 지정합니다. 모든 유형의 도메인은 이와 동일한 앱 구성 속성을 사용합니다.

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

> **Note**
>
> SimSpace Weaver 앱 SDK 버전 1.12.x 프로젝트는 앱 .zip 파일 및 스키마에 대해 별도의 버킷을 사용합니다.
>
> - weaver-*lowercase-project-name*-*account-number*-app-zips-*region*
> - weaver-*lowercase-project-name*-*account-number*-schemas-*region*

package 속성은 S3 버킷의 zip 파일 S3 URI을 지정합니다. zip 파일에는 앱 실행 파일(바이너리라고도 함) 및 필요한 기타 리소스(예: 라이브러리)가 들어 있습니다. 앱 실행 파일의 각 인스턴스는 작업자의 Docker 컨테이너에서 실행됩니다.

launch_command 속성은 실행 파일의 이름과 앱을 실행하는 데 필요한 모든 명령줄 옵션을 지정합니다. launch_command의 값은 배열입니다. 전체 시작 명령 문자열의 각 토큰은 배열의 한 요소입니다.

예제

- 시작 명령의 경우: `MyTestApp --option1 value1`
- `launch_command: ["MyTestApp", "-option1", "value1"]`를 지정합니다.

required_resource_units 속성은이 앱에 할당 SimSpace Weaver 해야 하는 컴퓨팅 리소스 단위 수를 지정합니다. 컴퓨팅 리소스 단위는 작업자의 고정된 처리 용량 (vCPU) 및 메모리 (RAM) 양입니다. 이 값을 늘려 앱이 작업자에서 실행될 때 사용할 수 있는 컴퓨팅 성능의 양을 늘릴 수 있습니다. 각 작업자에는 제한된 수의 컴퓨팅 리소스 단위가 있습니다. 자세한 내용은 SimSpace Weaver 엔드포인트 및 할당량 단원을 참조하십시오.

공간 도메인 구성

공간 도메인의 경우 partitioning_strategy를 지정해야 합니다. 이 속성의 값은 스키마의 다른 부분에서 정의한 분할 전략에 지정한 이름입니다.

```
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 앱 SDK 버전 1.12.x 프로젝트는 앱 .zip 파일 및 스키마에 대해 별도의 버킷을 사용합니다.
>
> - weaver-*lowercase-project-name*-*account-number*-app-zips-*region*
> - weaver-*lowercase-project-name*-*account-number*-schemas-*region*

Grid 토폴로지(이 릴리스에서 지원되는 유일한 토폴로지)가 있는 파티셔닝 전략은 그리드에서이 도메인의 공간 앱 파티션을 정렬 SimSpace Weaver 하도록 지시합니다. grid_partition 속성은 파티션 그리드의 행과 열 수를 지정합니다.

SimSpace Weaver 는 파티션 그리드의 각 셀에 대해 공간 앱의 인스턴스 1개를 시작합니다. 예를 들어 공간 도메인에 x: 2 및 grid_partition 값이 있는 경우 공간 도메인에는 2 * 2 = 4개의 파티션이 y: 2있습니다. SimSpace Weaver 는 공간 도메인에 구성된 앱 인스턴스 4개를 시작하고 각 앱 인스턴스에 파티션 1개를 할당합니다.

주제

- 공간 도메인의 리소스 요구 사항
- 공간 도메인 정보

- 공간 도메인에 대한 FAQ
- 공간 도메인 문제 해결

공간 도메인의 리소스 요구 사항

각 작업자에 최대 17개의 컴퓨팅 리소스 단위를 할당할 수 있습니다. 공간 도메인 app_config 섹션에서 각 공간 앱이 사용하는 컴퓨팅 리소스 단위 수를 지정합니다.

Example공간 앱의 컴퓨팅 리소스 단위를 보여주는 스키마 스니펫

```
  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
```

도메인에 필요한 컴퓨팅 리소스 단위의 수를 계산하려면 그리드의 셀 수(grid_partition, x *y)에 공간 앱에 할당된 컴퓨팅 리소스 단위의 수를 곱합니다.

이전 예제에서 도메인 MySpatialDomain는 다음을 지정했습니다.

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

MySpatialDomain 그리드는 2* 2 = 4셀입니다. 공간 도메인에는 4* 1 = 4 컴퓨팅 리소스 단위가 필요합니다.

스키마에 지정된 모든 도메인의 총 컴퓨팅 리소스 단위 수는 작업자의 desired 수에 각 작업자의 최대 컴퓨팅 리소스 단위 수(17)를 곱한 값보다 작거나 같아야 합니다.

공간 도메인 정보

둘 이상의 공간 도메인을 사용하도록 시뮬레이션을 구성할 수 있습니다. 예를 들어, 하나의 공간 도메인을 사용하여 시뮬레이션의 주요 작업자(예: 사람, 자동차)를 제어하고 다른 공간 도메인을 사용하여 환경을 제어할 수 있습니다.

또한 여러 공간 도메인을 사용하여 시뮬레이션의 각 부분에 서로 다른 리소스를 할당할 수 있습니다. 예를 들어 시뮬레이션에 다른 유형보다 10배 더 많은 엔터티 인스턴스가 있는 엔터티 유형이 있는 경우 각 엔터티 유형을 처리하는 다른 도메인을 만들고 엔터티가 더 많은 도메인에 더 많은 리소스를 할당할 수 있습니다.

> **Important**
>
> SimSpace Weaver 1.14.0 이전 버전은 여러 공간 도메인을 지원하지 않습니다.

> **Important**
>
> AWS SimSpace Weaver Local는 현재 여러 공간 도메인을 지원하지 않습니다. SimSpace Weaver Local에 대한 자세한 정보는 의 로컬 개발 SimSpace Weaver 섹션을 참조하세요.

> **Important**
>
> SimSpace Weaver 는 각 시뮬레이션에 대해 최대 5개의 도메인을 지원합니다. 여기에는 모든 공간, 사용자 지정 및 서비스 도메인이 포함됩니다.

여러 공간 도메인 구성

두 개 이상의 공간 도메인을 구성하려면 다른 공간 도메인 정의를 스키마에 별도의 명명된 섹션으로 추가합니다. 각 도메인은 `launch_apps_by_partitioning_strategy` 키를 지정해야 합니다. 다음의 스키마 예제를 참조하세요.

```
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
```

공간 도메인을 함께 배치

일부 시나리오에서는 작업자의 공간 도메인용 파티션을 다른 도메인의 파티션 옆에 배치할 수 있습니다. 그러면 파티션이 서로에 대한 크로스 도메인 구독을 생성하는 경우 성능 특성이 향상될 수 있습니다.

`placement_constraints` 스키마에 최상위 키를 추가하여 함께 배치 SimSpace Weaver 할 도메인을 지정합니다. 필수 `on_workers` 키는 스키마의 명명된 `workers` 구성을 참조해야 합니다.

Example함께 배치된 공간 도메인을 보여주는 스키마 스니펫

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

Important

- 배치 그룹을 사용하는 경우 다음을 수행합니다.
  - x * y가 작업자 수의 배수인지 확인합니다.
  - 배치 그룹 값이 함께 배치하는 도메인의 그리드 크기를 나타내는 공약수인지 확인합니다.
- 배치 그룹을 사용하지 않는 경우 다음을 수행합니다.
  - 공간 도메인 그리드의 한 축에 작업자 수와 동일한 공약수가 있는지 확인합니다.

배치 그룹에 대한 자세한 내용은 파티셔닝 전략 섹션을 참조하세요.

공간 도메인에 대한 FAQ

Q1. 기존 시뮬레이션에 다른 공간 도메인을 추가하려면 어떻게 해야 하나요?

- 실행 중인 시뮬레이션의 경우 - 실행 중인 시뮬레이션의 구성을 변경할 수 없습니다. 스키마에서 도메인 구성을 변경하고, 스키마와 앱 zip을 업로드하고, 새 시뮬레이션을 시작합니다.
- 새 시뮬레이션의 경우 - 스키마에서 도메인 구성을 추가하고, 스키마와 앱 zip을 업로드하고, 새 시뮬레이션을 시작합니다.

공간 도메인 문제 해결

시뮬레이션을 시작하려고 했지만 도메인 구성이 유효하지 않은 경우 다음의 오류가 발생할 수 있습니다.

```
"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].\"}]"
```

예상 원인

- 스키마는 작업자가 사용할 수 있는 것보다 더 많은 컴퓨팅 리소스 단위를 앱에 할당합니다.
- SimSpace Weaver 는 작업자에 도메인을 함께 배치하는 배열을 결정할 수 없습니다. 이는 공간 도메인을 여러 개 지정했지만 도메인 그리드 사이에 공약수나 배수가 없는 경우(예: 2x4 그리드와 3x5 그리드 사이)에 발생합니다.

네트워크 엔드포인트

사용자 지정 및 서비스 앱에는 외부 클라이언트가 연결할 수 있는 네트워크 엔드포인트가 있을 수 있습니다. 포트 번호 목록을 `endpoint_config`의 `ingress_ports` 값으로 지정합니다. 이러한 포트 번호는 모두 TCP와 UDP입니다. 사용자 지정 또는 서비스 앱은에서 지정한 포트 번호에 바인딩되어야 합니다`ingress_ports`.는 런타임 시 포트 번호를 SimSpace Weaver 동적으로 할당하고 이러한 포트를 동적 포트에 매핑합니다. 앱이 동적(실제) 포트 번호를 찾기 시작한 후 describe-app API를 호출할 수 있습니다. 자세한 내용은 빠른 시작 자습서의 사용자 지정 앱의 IP 주소 및 포트 번호 가져오기 섹션을 참조하세요.

```
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 앱 SDK 버전 1.12.x 프로젝트는 앱 .zip 파일 및 스키마에 대해 별도의 버킷을 사용합니다.
>
> - weaver-*lowercase-project-name*-*account-number*-app-zips-*region*

- weaver-*lowercase-project-name*-*account-number*-schemas-*region*

> **Note**
>
> endpoint_config는 사용자 지정 앱 및 서비스 앱의 선택적 속성입니다. endpoint_config를 지정하지 않으면 앱에 네트워크 엔드포인트가 없습니다.

서비스 도메인 구성

도메인 구성launch_apps_per_worker:에가 있으면 서비스 앱이 있는 서비스 도메인임을 나타냅니다.는 서비스 앱을 SimSpace Weaver 시작하고 중지합니다. 가 앱을 SimSpace Weaver 시작하고 중지하면 앱에 관리형 수명 주기가 있는 것으로 간주됩니다. SimSpace Weaver 현재는 각 작업자에서 1개 또는 2개의 서비스 앱 시작을 지원합니다.

Example각 작업자에서 1개의 서비스 앱을 실행하도록 구성된 도메인의 예제

```
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
```

Example각 작업자에서 2개의 서비스 앱을 실행하도록 구성된 도메인의 예제

```
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
```

시뮬레이션의 최대 지속 시간

의 각 시뮬레이션 AWS SimSpace Weaver 에는 시뮬레이션을 실행할 수 있는 최대 시간을 지정하는 최대 기간 설정이 있습니다. 시뮬레이션을 시작할 때 최대 지속 시간을 파라미터로 제공합니다. StartSimulation 애플리케이션 프로그래밍 인터페이스(API)에는 선택적 파라미터 MaximumDuration이 있습니다. 파라미터 값은 분(m 또는 M), 시간(h 또는 H) 또는 일(d 또는 D)입니다. 예를 들어, 1h 또는 1H는 1시간을 의미합니다. SimSpace Weaver 은 이 제한에 도달하면 시뮬레이션을 중지합니다.

최대값

MaximumDuration에 대한 가장 높은 유효 값은 14D 또는 이에 상응하는 시간(336H) 또는 분(20160M)입니다.

기본값

MaximumDuration 파라미터는 선택 항목입니다. 값을 제공하지 않으면는 값을 SimSpace Weaver 사용합니다14D.

최소값

MaximumDuration에 대한 가장 낮은 유효 값은 수치적으로 0에 상응하는 값입니다. 예를 들어,0M, 0H, 0D 값은 모두 수치적으로 0에 상응합니다.

최대 지속 시간에 대한 최소값을 제공하면 시뮬레이션이 STOPPING 상태에 도달하는 즉시 STARTED 상태로 전환됩니다.

콘솔을 사용하여 시뮬레이션 시작

SimSpace Weaver 콘솔에서 시뮬레이션을 시작할 때 최대 지속 시간 값을 제공할 수 있습니다. 시뮬레이션 시작을 선택할 때 시뮬레이션 설정 양식의 최대 지속 시간 필드에 값을 입력합니다.

> **Important**
>
> 최대 지속 시간에 값을 입력하지 않으면 SimSpace Weaver 에서 기본값(14D)을 사용합니다.

최대 지속 시간에 도달한 시뮬레이션의 상태

가 최대 지속 시간에 도달한 시뮬레이션을 SimSpace Weaver 자동으로 중지하면 시뮬레이션의 상태는 STOPPING (진행 중인 경우) 또는 입니다STOPPED. SimSpace Weaver 콘솔에서 시뮬레이션의 대상 상태가 계속 STARTED 상태로 표시되는 이유는 사용자가 요청한 마지막 상태이기 때문입니다.

앱 개발

SimSpace Weaver 개발에는 Amazon Linux에서 시뮬레이션이 실행되기 때문에 앱을 빌드하는 Amazon Linux 2 (AL2) 환경이 필요합니다AWS Cloud. 를 사용하는 경우 SimSpace Weaver 앱 SDK의 스크립트를 사용하여 SimSpace Weaver 앱을 빌드하는 데 필요한 종속성AL2으로 실행되는 Docker 컨테이너를 생성하고 시작할 Windows수 있습니다. Windows Subsystem for Linux (WSL)를 사용하여 AL2 환경을 시작하거나 네이티브 AL2 시스템을 사용할 수도 있습니다. 자세한 내용은 에 대한 로컬 환경 설정 SimSpace Weaver 단원을 참조하십시오.

> **Note**
>
> 로컬 개발 환경을 구성하는 방법에 관계없이 AWS 클라우드에서 실행되도록 앱을 업로드하면 앱이 Docker 컨테이너에서 실행됩니다. 앱은 호스트 운영 체제에 직접 액세스할 수 없습니다.

SimSpace Weaver 앱의 일반 흐름

1. 애플리케이션을 생성합니다.
2. 루프:
   a. `Transaction`을 생성하여 업데이트를 시작합니다.

- 시뮬레이션이 종료되면 루프를 종료합니다.

b. 구독 및 소유권 엔터티 이벤트를 처리합니다.

c. 시뮬레이션을 업데이트합니다.

d. 업데이트를 종료하려면 `Transaction`을 커밋합니다.

3. 애플리케이션을 삭제합니다.

공간 앱

각 공간 앱에는 시뮬레이션 세계의 공간 영역인 소유권 영역이 있습니다. 공간 앱의 소유권 영역에 있는 엔터티는 앱에 할당된 파티션에 저장됩니다. 단일 공간 앱은 할당된 파티션 내의 모든 엔터티에 대한 완전한 소유권(읽기 및 쓰기 권한)을 갖습니다. 다른 앱은 이러한 엔터티에 쓸 수 없습니다. 공간 앱은 해당 엔터티의 상태를 발전시킵니다. 각 공간 앱은 1개의 파티션만 소유합니다. SimSpace Weaver는 엔터티의 공간적 위치를 사용하여 엔터티를 인덱싱하고 공간 앱 파티션에 할당합니다.

SimSpace Weaver 앱 SDK는 샘플 애플리케이션을 제공합니다. 다음 폴더에서 샘플 애플리케이션의 공간 앱에 대한 소스 코드를 찾을 수 있습니다(운영 체제에 적합한 경로 구분자 사용).

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

사용자 지정 앱

사용자 지정 앱을 만들고 사용하여 시뮬레이션과 상호 작용합니다.

사용자 지정 앱은 다음을 수행할 수 있습니다.

- 엔터티 생성
- 다른 파티션 구독
- 변경 사항 커밋

사용자 지정 앱의 일반적인 흐름

1. 애플리케이션을 생성합니다.
2. 다음과 같이 시뮬레이션에서 특정 리전을 구독합니다.

   a. `Transaction`을 생성하여 첫 번째 업데이트를 시작합니다.

   b. 특정 리전에 대한 구독을 생성합니다.

c. 첫 번째 업데이트를 종료하려면 `Transaction`을 커밋합니다.

3. 루프:

   a. `Transaction`을 생성하여 업데이트를 시작합니다.

      - 시뮬레이션이 종료되면 루프를 종료합니다.

   b. 프로세스 상태가 변경됩니다.

   c. 업데이트를 종료하려면 `Transaction`을 커밋합니다.

4. 애플리케이션을 삭제합니다.

사용자 지정 앱이 개체를 생성한 후 개체가 시뮬레이션 내에 공간적으로 존재하도록 하려면 개체를 공간 도메인으로 전송해야 합니다.는 개체의 공간 위치를 SimSpace Weaver 사용하여 개체를 적절한 공간 앱 파티션에 배치합니다. 엔터티를 만든 사용자 지정 앱은 엔터티를 공간 도메인으로 전송한 후에는 엔터티를 업데이트하거나 삭제할 수 없습니다.

SimSpace Weaver 앱 SDK는 샘플 애플리케이션을 제공합니다. 샘플 애플리케이션에 포함된 사용자 지정 앱을 자체 사용자 지정 앱의 모델로 사용할 수 있습니다. 다음 폴더에서 샘플 애플리케이션의 보기 앱(사용자 지정 앱)에 대한 소스 코드를 찾을 수 있습니다(운영 체제에 올바른 경로 구분 기호 사용).

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

클라이언트 애플리케이션 개발

클라이언트를 시뮬레이션에 연결해야 하는 몇 가지 이유는 다음과 같습니다.

- 도시 규모의 시뮬레이션에 실시간 교통 정보를 주입합니다.
- 인간 운영자가 시뮬레이션의 일부 측면을 제어하는 human-in-the-loop 시뮬레이션을 만듭니다.
- 훈련 시뮬레이션과 같이 사용자가 시뮬레이션과 상호 작용할 수 있도록 합니다.

이 예제의 사용자 지정 앱은 시뮬레이션 상태와 외부 세계 사이의 인터페이스 역할을 합니다. 클라이언트는 사용자 지정 앱에 연결하여 시뮬레이션과 상호 작용합니다.

SimSpace Weaver 는 클라이언트 애플리케이션과 사용자 지정 앱과의 통신을 처리하지 않습니다. 클라이언트 애플리케이션의 설계, 생성, 운영, 보안 및 사용자 지정 앱과의 커뮤니케이션은 사용자의 책임입니다. SimSpace Weaver 는 클라이언트가 연결할 수 있도록 각 사용자 지정 앱의 IP 주소 및 포트 번호만 노출합니다.

SimSpace Weaver 앱 SDK는 샘플 애플리케이션을 위한 클라이언트를 제공합니다. 이러한 클라이언트를 자체 클라이언트 애플리케이션의 모델로 사용할 수 있습니다. 다음 폴더에서 샘플 애플리케이션 클라이언트의 소스 코드를 찾을 수 있습니다.

Docker

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

WSL

> **Important**
>
> 편의를 위해 이 지침을 제공합니다. 이 지침은 Windows Subsystem for Linux (WSL)와 함께 사용할 수 있도록 지원되지 않습니다. 자세한 내용은 에 대한 로컬 환경 설정 SimSpace Weaver 단원을 참조하십시오.

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

샘플 애플리케이션 클라이언트를 빌드하고 사용하는 방법에 대한 자세한 내용은의 자습서를 참조하세요시작하기 SimSpace Weaver.

사용자 지정 앱의 IP 주소 및 포트 번호 가져오기

시뮬레이션을 보려면 사용자 지정 앱을 생성하고 클라이언트를 사용하여 연결합니다. 자세한 내용은의 자습서를 참조하세요시작하기 SimSpace Weaver. 다음 절차에 따라 사용자 지정 앱의 IP 주소와 포트 번호를 가져올 수 있습니다. 운영 체제에 적합한 경로 구분자를 사용합니다(예: \ Windows의 경우 , Linux/의 경우).

IP 주소 및 포트 번호 가져오기

1. ListSimulation API를 사용하여 시뮬레이션 이름을 가져올 수 있습니다.

   ```
   aws simspaceweaver list-simulations
   ```

   출력 예시:

```
{
    "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. DescribeSimulation API를 사용하여 시뮬레이션의 도메인 목록을 가져옵니다.

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

출력의 `LiveSimulationState` 섹션에서 `Domains` 섹션을 찾습니다.

출력 예시:

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

3. ListApps API를 사용하여 도메인의 사용자 지정 앱 목록을 가져옵니다. 예를 들어 샘플 프로젝트에서 뷰(사용자 지정) 앱의 도메인 이름은 입니다`MyViewDomain`. 출력에서 앱 이름을 찾습니다.

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

출력 예시:

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

4. DescribeApp API를 사용하여 IP 주소와 포트 번호를 가져옵니다. 샘플 프로젝트의 경우 도메인 이름은 `MyViewDomain`이고 앱 이름은 `ViewApp`입니다.

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

IP 주소와 포트 번호는 출력의 `EndpointInfo` 블록에 있습니다. IP 주소는 `Address`의 값이고 포트 번호는 `Actual`의 값입니다.

출력 예시:

```
{
    "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**
>
> Declared의 값은 앱 코드가 바인딩되어야 하는 포트 번호입니다. 의 값은가 클라이언트에 SimSpace Weaver 노출하여 앱에 연결하는 포트 번호Actual입니다.는 Declared포트를 Actual 포트에 SimSpace Weaver 매핑합니다.

Unreal Engine 뷰 클라이언트 시작

다음으로 이동합니다.

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

1. 다음 명령 중 하나를 실행합니다.
   - 도커: `python quick-start.py`
   - WSL: `python quick-start.py --al2`
2. IP 주소와 “실제” 포트 번호를 가져옵니다. 이는 quick-start.py 실행하여 콘솔 출력에 있거나의 절차에 따라 가져옵니다사용자 지정 앱의 IP 주소 및 포트 번호 가져오기.
3. 다음으로 이동합니다.

   ```
   sdk-folder/Clients/TCP/UnrealClient/lib
   ```
4. 다음 명령을 실행하여 NNG 라이브러리를 빌드합니다.

   ```
   cmake -S . -B build
   ```

```
cmake --build build --config RelWithDebInfo
cmake --install build
```

5. 텍스트 편집기에서 `view_app_url.txt`를 엽니다.
6. URL을 뷰 앱의 IP 주소 및 포트 번호로 업데이트합니다. `tcp://ip-address:actual-port-number`(`tcp://198.51.100.135:1234`와 같아야 함).
7. Unreal 편집기에서 재생을 선택합니다.

문제 해결

- NNG CMake 설치 단계가 실패하고 "관리 권한이 필요할 수 있음":

```
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)
```

  - 해결 방법: UnrealClient/lib 디렉터리에 `nng.lib` 또는가 `nng.so` 있는 경우이 오류를 무시해도 됩니다. 그렇지 않은 경우 관리자 권한이 있는 터미널에서 cmake 빌드 명령을 실행해 보세요.

- “CMake를 사용하여 nng에서 제공하는 패키지 구성 파일 찾기”:

```
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.
```

  - 해결 방법: CMake에서 `Findnng.cmake` 파일을 찾는 데 문제가 있습니다. CMake로 빌드할 때 인수를 추가합니다`-DTHIRD_PARTY_LIB_PATH sdk-folder/ThirdParty`. CMake 빌드를 다시 실행하기 전에 `Findnng.cmake` 파일이 `ThirdParty` 디렉터리에 여전히 있는지 확인합니다.

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

의 로컬 개발 SimSpace Weaver

신속한 테스트 및 디버깅을 위해 SimSpace Weaver 애플리케이션을 로컬에 배포할 수 있습니다.

요구 사항

- 에 대한 설정 SimSpace Weaver의 단계를 수행하세요.

주제

1단계: 로컬 시뮬레이션 시작

1. 로 이동

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

2. 다음 명령을 실행하여 시뮬레이션을 로컬로 빌드하고 시작합니다.

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

   이 스크립트는 다음을 수행합니다.

   1. 프로젝트를 빌드합니다.
      - `quick-start.py`는 build.py 정의된 `build_project` 함수를 호출합니다. 이 단계는 프로젝트에 따라 달라집니다. PathfindingSample의 경우 CMake가 사용됩니다. build.py 찾을 수 있는 CMake 및 Docker 명령입니다.
   2. 로컬 시뮬레이션 시작
      - 스크립트는 스키마에 정의된 각 공간 파티션에 대해 하나의 로컬 프로세스를 시작합니다.
      - 스크립트는 스키마에 정의된 각 사용자 지정 앱에 대해 하나의 프로세스를 시작합니다.
      - 공간 앱이 먼저 시작된 다음 사용자 지정 앱이 시작됩니다. 각 앱은 스키마에 나타나는 순서대로 시작됩니다.

> **Important**
>
> 콘솔 SSH 세션과 같이 GUI를 지원하지 않는 환경에서 시작할 때는 `--noappwindow` 옵션을 사용하여 모든 출력을 현재 터미널로 리디렉션합니다.

> **Important**
>
> Linux 사용자의 경우 스크립트는 시스템에 `xterm` 명령이 있다고 가정합니다. Linux 배포에 `xterm` 명령이 없는 경우 `--noappwindow` 옵션을 사용하여 모든 출력을 현재 터미널로 리디렉션합니다.

- -h, --help
  - 이러한 파라미터를 나열합니다.
- --정리
  - 빌드하기 전에 빌드 디렉터리의 내용을 삭제합니다.
- --nobuild
  - 프로젝트 재구축을 건너뜁니다.
- --noappwindow
  - 각 앱에 대해 새 창을 열지 마세요. 대신 stdout을 현재 터미널로 리디렉션합니다.
- --logfile
  - 로그 파일에 콘솔 출력을 씁니다.
- --콘솔리언트
  - 구성에 나열된 콘솔 클라이언트를 자동으로 연결합니다.
- --스키마 SCHEMA
  - 이 호출에서 사용할 스키마입니다. config.py://https://https://https://https://https://https://https://https://https://https://https://https://https

2단계: 로컬 시뮬레이션 보기

로컬 시뮬레이션을 보기 위해 SimSpaceWeaverAppSdkDistributable에 포함된 모든 클라이언트를 사용할 수 있습니다. 샘플 클라이언트를 빌드하고 사용하는 방법에 대한 자세한 내용은의 자습서를 참조하세요시작하기 SimSpace Weaver.

로컬 시뮬레이션을 위해 보기 앱에 연결하려면 클라이언트의 IP 주소와 포트 번호를 업데이트해야 합니다. 항상 다음 값을 SimSpace Weaver Local과 함께 사용합니다.

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

선택한 클라이언트에 따라 다음과 같이 IP 주소 및 포트 번호를 업데이트할 수 있습니다.

- Unreal - `view_app_url.txt`의 첫 번째 줄에서 URL을 변경합니다.
- 콘솔 - IP 주소와 포트 번호(URL 를 파라미터로 사용하여 클라이언트를 시작합니다.

3단계: 로컬 시뮬레이션 중지(Windows에서는 선택 사항)

> **Note**
>
> 이 단계는 Linux에서는 필요하지만 Windows에서는 선택 사항입니다.

1. 다음으로 이동합니다.

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

2. 다음 명령을 실행하여 로컬 시뮬레이션을 중지하고 공유 메모리 리소스를 삭제합니다.

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

   이 스크립트는 다음을 수행합니다.

   - 로컬 프로세스를 중지합니다.
   - 공유 메모리 객체를 삭제합니다(Linux에서만 필요).

stop-and-delete.py 파라미터

- -h, --help
  - 이러한 파라미터를 나열합니다.
- --중지
  - 프로세스만 중지하려고 시도합니다.

- - 삭제
  - 공유 메모리 리소스만 삭제하려고 시도합니다.
- --프로세스
  - 중지할 프로세스의 이름입니다. 프로세스 이름이 스키마의 패키지 이름과 일치하지 않는 경우 이를 사용합니다.
- --스키마 SCHEMA
  - 이 호출에서 사용할 스키마입니다. 기본값은 config.pySCHEMA'입니다.

에서 로컬 개발 문제 해결 SimSpace Weaver

- Linux: xterm 명령을 찾을 수 없음/ 열 수 없음
  - 로컬 스크립트는 Linux에서 실행할 때 xterm 명령이 존재한다고 가정합니다. xterm 명령이 없거나 GUI를 지원하지 않는 환경에서 실행 중인 경우 빠른 시작 스크립트를 실행할 때 `--noappwindow` 옵션을 사용합니다.
- 열려 있는 앱 창이 없습니다!
  - 이는 로컬 시뮬레이션이 즉시 충돌할 때 발생합니다. 충돌 후 콘솔 출력을 보려면 빠른 시작 스크립트를 실행할 때 `--noappwindow` 또는 `--logfile` 옵션을 사용합니다.
- 뷰 앱이 시작되거나 뷰 클라이언트가 연결된 후 시뮬레이션이 틱하지 않습니다!
  - –noappwindow 옵션을 사용하여를 실행하면 일반적으로 이러한 종류의 문제가 해결됩니다. 그렇지 않으면 몇 번 다시 시작해도 성공합니다(비록 훨씬 낮은 속도임).

AWS SimSpace Weaver 앱 SDK

SimSpace Weaver 앱 SDK는 시뮬레이션에서 개체를 제어하고 SimSpace Weaver 이벤트에 응답하는 데 사용할 수 있는 APIs를 제공합니다. 여기에는 다음 네임스페이스가 포함됩니다.

- API - API의 핵심 정의 및 해당 사용

다음 라이브러리와 연결합니다.

- `libweaver_app_sdk_cxx_v1_full.so`

Important

라이브러리는 AWS 클라우드에서 앱을 실행할 때 동적 연결에 사용할 수 있습니다. 앱과 함께 업로드할 필요가 없습니다.

Note

SimSpace Weaver 앱 SDK APIs는 시뮬레이션 내에서 데이터를 제어합니다. 이러한 APIs는 SimSpace Weaver 서비스 리소스(예: 시뮬레이션, 앱 및 클럭)를 제어하는 SimSpace Weaver 서비스 APIs와는 별개입니다 AWS. 자세한 내용은 SimSpace Weaver API 참조 단원을 참조하십시오.

주제

API 메서드는 Result를 반환합니다.

대부분의 SimSpace Weaver API 함수에는 반환 유형이 있습니다`Aws::WeaverRuntime::Result<T>`. 함수가 성공적으로 실행되면 `Result`에 `T`가 포함됩니다. 그렇지 않으면 `Result`에 Rust App SDK의 오류 코드를 나타내는 `Aws::WeaverRuntime::ErrorCode`가 포함됩니다.

Example예제

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

이 메서드는 다음을 수행합니다.

- `BeginUpdate()` 실행이 성공하면 `Transaction`을 반환합니다.
- `BeginUpdate()` 실행이 실패하면 `Aws::WeaverRuntime::ErrorCode`를 반환합니다.

최상위의 앱 SDK와 상호 작용

수명 주기

- SimSpace Weaver 앱 SDK는 앱 수명 주기를 관리합니다. 앱의 수명 주기 상태를 읽거나 쓸 필요가 없습니다.

파티션

- `Result <PartitionSet> AssignedPartitions(Transaction& txn);`를 사용하여 소유한 파티션을 가져옵니다.
- `Result <PartitionSet> AllPartitions(Transaction& txn);`를 사용하여 시뮬레이션의 모든 파티션을 가져옵니다.

시뮬레이션 관리

이 섹션에서는 일반적인 시뮬레이션 관리 작업에 대한 솔루션을 설명합니다.

주제

- 시뮬레이션 시작
- 시뮬레이션 업데이트
- 시뮬레이션 종료

시뮬레이션 시작

`CreateApplication()`을 사용하여 앱을 생성합니다.

Example예제

```
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()));
```

시뮬레이션 업데이트

다음 BeginUpdate 함수를 사용하여 앱을 업데이트합니다.

- `Result<Transaction> BeginUpdate(Application& app)`
- `Result<bool> BeginUpdateWillBlock(Application& app)` - `BeginUpdate()`의 차단 여부를 알려줍니다.

`Result<void> Commit(Transaction& txn)`을 사용하여 변경 내용을 커밋합니다.

Example예제

```
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();
}
```

시뮬레이션 종료

`Result<void> DestroyApplication(Application&& app)`을 사용하여 앱과 시뮬레이션을 종료합니다.

다른 앱은 `BeginUpdateWillBlock()` 또는 `BeginUpdate()`에 대한 호출에서 `ErrorCode::ShuttingDown`이 수신되는 경우 시뮬레이션이 종료된다는 사실을 알게 됩니다. 앱이 `ErrorCode::ShuttingDown`을 수신하면 `Result<void> DestroyApplication(Application&& app)` 호출을 자체적으로 종료할 수 있습니다.

Example예제

```
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**
>
> Api::Commit() 다음에 Result<void> DestroyApplication(Application&& app)을 호출합니다. 업데이트 중에 애플리케이션을 삭제하면 정의되지 않은 동작이 발생할 수 있습니다.

> **Important**
>
> 프로그램이 종료되기 전에 DestroyApplication()을 호출하여 애플리케이션이 성공적으로 종료되었다고 보고하는지 확인해야 합니다.
> 프로그램이 종료될 때 DestroyApplication()을 호출하지 않으면 FATAL 상태로 간주됩니다.

구독

구독 영역과 도메인 ID를 사용하여 구독을 생성합니다. 도메인 ID는 해당 구독 영역을 소유한 도메인을 나타냅니다. BoundingBox2F32는 구독 영역을 설명합니다. 다음 함수를 사용하여 구독을 생성합니다.

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

Example예제

```
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();
}
```

CreateSubscriptionBoundingBox2F32()에서 반환된 Api::SubscriptionHandle을 사용하여 구독을 수정할 수 있습니다. 다음 함수에 인수로 전달합니다.

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

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

개체

CreateEntity()에서 반환된 Result<Api::Entity>의 Api:Entity을 사용하거나 엔터티가 앱의 구독 영역에 진입할 때 소유권 변경 이벤트에서 Store 및 Load API를 호출합니다(자세한 내용은 엔터티 이벤트 섹션 참조). 이러한 API와 함께 사용할 수 있도록 Api::Entity 객체를 추적하는 것이 좋습니다.

주제

- 엔터티 생성

- 공간 도메인으로 엔터티 이전
- 엔터티에 필드 데이터 쓰기 및 읽기
- 엔터티에 대한 위치 저장
- 엔터티에 대한 위치 로드

엔터티 생성

`CreateEntity()`를 사용하여 엔터티를 생성합니다. 이 함수에 전달하는 `Api::TypeId`의 의미를 정의합니다.

```
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`에 0~511의 값이 예약되어 있습니다. 엔터티 TypeID(이 예제에서는 `k_entityTypeId`)의 값은 512 이상이어야 합니다.

공간 도메인으로 엔터티 이전

사용자 지정 앱 또는 서비스 앱이 엔터티를 생성한 후 엔터티가 시뮬레이션에서 공간적으로 존재하도록 하려면 앱이 엔터티를 공간 도메인으로 전송해야 합니다. 공간 도메인의 엔터티는 다른 앱에서 읽을 수 있고 공간 앱에서 업데이트할 수 있습니다. `ModifyEntityDomain()` API를 사용하여 엔터티를 공간 도메인으로 전송할 수 있습니다.

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

DomainId가 호출 앱에 할당된 Partition과 일치하지 않는 경우 DomainId는 DomainType::Spatial Domain에 대한 ID이어야 합니다. 새 Domain으로의 소유권 전송은 Commit(Transaction&&) 중에 이루어집니다.

파라미터

txn

현재의 Transaction입니다.

entity

Domain를 변경할 대상 Entity입니다.

domainId

Entity에 대한 대상 Domain의 DomainId입니다.

이 API는 엔터티 도메인이 성공적으로 변경되면 Success를 반환합니다.

엔터티에 필드 데이터 쓰기 및 읽기

모든 엔터티 데이터 필드는 blob 유형입니다. 엔터티에 최대 1,024바이트의 데이터를 쓸 수 있습니다. Blob 크기가 클수록 성능이 저하되므로 blob을 최대한 작게 유지하는 것이 좋습니다. Blob에 쓸 때 포인터 SimSpace Weaver 를 데이터 및 길이에 전달합니다. Blob에서 읽을 때 SimSpace Weaver 는 포인터 및 읽을 길이를 제공합니다. 앱이 Commit()을 호출하기 전에 모든 읽기가 완료되어야 합니다. 읽기 호출에서 반환된 포인터는 앱이 Commit()을 호출할 때 무효화됩니다.

> **Important**
>
> - Commit() 이후에 캐시된 blob 포인터에서의 읽기는 지원되지 않으므로 시뮬레이션이 실패할 수 있습니다.
> - 읽기 호출에서 반환된 blob 포인터에서의 쓰기는 지원되지 않으므로 시뮬레이션이 실패할 수 있습니다.

주제

- 엔터티의 필드 데이터 저장
- 엔터티의 필드 데이터 불러오기

- 제거된 엔터티의 필드 데이터 로드

엔터티의 필드 데이터 저장

다음 예제는 앱이 소유한 엔터티의 필드 데이터를 저장(상태 패브릭에 쓰기)하는 방법을 보여줍니다. 해당 예제에는 다음 함수가 사용됩니다.

```
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` 파라미터는 전달된 데이터의 데이터 유형을 나타냅니다.

`Api::TypeId keyTypeId` 파라미터는 `Api::BuiltinTypeId`에서 해당 `Api::TypeId`를 수신해야 합니다. 적절한 변환이 없는 경우 `Api::BuiltinTypeId::Dynamic`을 사용할 수 있습니다.

복잡한 데이터 유형의 경우 `Api::BuiltInTypeId::Dynamic`을 사용합니다.

> **Note**
>
> `FieldIndex index` 값은 0보다 커야 합니다. 0 값은 인덱스 키용으로 예약되어 있습니다(`StoreEntityIndexKey()` 참조).

Example프리미티브 데이터 유형을 사용한 예제

```
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 struct를 사용한 데이터 보관 예제

```
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));
```

```
}
```

엔터티의 필드 데이터 불러오기

다음 예제는 엔터티의 필드 데이터를 로드(상태 패브릭에서 읽기)하는 방법을 보여줍니다. 해당 예제에는 다음 함수가 사용됩니다.

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

`Api::TypeId keyTypeId` 파라미터는 `Api::BuiltinTypeId`에서 해당 `Api::TypeId`를 수신해야 합니다. 적절한 변환이 없는 경우 `Api::BuiltinTypeId::Dynamic`을 사용할 수 있습니다.

> **Note**
>
> `FieldIndex` 인덱스 값은 0보다 커야 합니다. 0 값은 인덱스 키용으로 예약되어 있습니다(`StoreEntityIndexKey()` 참조).

Example프리미티브 데이터 유형을 사용한 예제

```
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 struct를 사용한 데이터 보관 예제

```
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);
}
```

제거된 엔터티의 필드 데이터 로드

앱의 소유권 및 구독 영역에서 제거된 엔터티의 엔터티 필드 데이터는 로드(상태 패브릭에서 읽기)할 수 없습니다. 다음 예제에서는 `Api::ChangeListAction::Remove`의 결과로 엔터티에 대한 `Api::LoadIndexKey()` 호출로 인해 오류가 발생합니다. 두 번째 예제는 앱에서 직접 항목 데이터를 저장하고 로드하는 올바른 방법을 보여줍니다.

Example잘못된 코드의 예제

```
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;
            }
        }

    }

    /* ... */
}
```

Example앱에서 항목 데이터를 저장하고 로드하는 올바른 방법의 예제

```
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;
            }
        }
    }

    /* ... */
}
```

엔터티에 대한 위치 저장

정수 데이터 구조를 사용하여 엔터티의 위치를 저장(상태 패브릭에 쓰기)할 수 있습니다. 해당 예제에는 다음 함수가 사용됩니다.

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

> **Note**
>
> 다음 예제에서와 같이 `Api::BuiltinTypeId::Vector3F32`를 `Api::StoreEntityIndexKey()`에 제공해야 합니다.

Example배열을 사용하여 위치를 나타내는 예제

```
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 struct을 사용하여 위치를 나타내는 예제

```
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));
}
```

엔터티에 대한 위치 로드

정수 데이터 구조를 사용하여 엔터티의 위치를 로드(상태 패브릭에서 읽기)할 수 있습니다. 해당 예제에는 다음 함수가 사용됩니다.

> **Note**
>
> 다음 예제에서와 같이 `Api::BuiltinTypeId::Vector3F32`를 `Api::LoadEntityIndexKey()`에 제공해야 합니다.

Example배열을 사용하여 위치를 나타내는 예제

```
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 struct을 사용하여 위치를 나타내는 예제

```
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);
}
```

엔터티 이벤트

SimSpace Weaver 앱 SDK에서 다음 함수를 사용하여 모든 소유권 및 구독 이벤트를 가져올 수 있습니다.

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

콜백 기반 엔터티 이벤트 처리가 필요한 경우 SimSpace Weaver 데모 프레임워크를 사용할 수 있습니다. 자세한 내용은 다음 헤더 파일을 참조하세요.

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

고유한 엔터티 이벤트 처리를 만들 수도 있습니다.

주제

- 소유한 엔터티에 대한 이벤트 반복 실행
- 구독한 엔터티의 이벤트 반복 실행

- 엔터티에 대한 소유권 변경 이벤트 반복 실행

소유한 엔터티에 대한 이벤트 반복 실행

`OwnershipChanges()`를 사용하여 소유한 엔터티(앱 소유권 영역에 있는 엔터티)에 대한 이벤트 목록을 가져옵니다. 함수의 서명은 다음과 같습니다.

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

그런 후 다음 예제와 같이 루프를 사용하여 엔터티를 반복합니다.

Example예제

```
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;
    }
}
```

이벤트 유형

- None - 엔터티가 해당 영역에 있고 위치 및 필드 데이터가 수정되지 않았습니다.
- Remove - 엔터티가 영역에서 제거되었습니다.
- Add - 엔터티가 영역에 추가되었습니다.
- Update - 엔터티가 해당 영역에 있으며 수정되었습니다.
- Reject - 앱이 해당 영역에서 엔터티를 제거하지 못했습니다.

> **Note**
>
> Reject 이벤트가 발생한 경우 앱은 다음 틱에서 전송을 다시 시도합니다.

구독한 엔터티의 이벤트 반복 실행

AllSubscriptionEvents()를 사용하여 구독한 엔터티(앱 구독 영역에 있는 엔터티)에 대한 이벤트 목록을 가져옵니다. 함수의 서명은 다음과 같습니다.

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

그런 후 다음 예제와 같이 루프를 사용하여 엔터티를 반복합니다.

Example예제

```
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;
    }
}
```

이벤트 유형

- `None` - 엔터티가 해당 영역에 있고 위치 및 필드 데이터가 수정되지 않았습니다.
- `Remove` - 엔터티가 영역에서 제거되었습니다.
- `Add` - 엔터티가 영역에 추가되었습니다.
- `Update` - 엔터티가 해당 영역에 있으며 수정되었습니다.
- `Reject` - 앱이 해당 영역에서 엔터티를 제거하지 못했습니다.

> **Note**
>
> `Reject` 이벤트가 발생한 경우 앱은 다음 틱에서 전송을 다시 시도합니다.

엔터티에 대한 소유권 변경 이벤트 반복 실행

엔터티가 소유권 영역과 구독 영역 사이를 이동하는 이벤트를 가져오려면 현재 및 이전 엔터티 소유권 및 구독 이벤트 간의 변경 사항을 비교합니다.

다음을 읽으면 이러한 이벤트를 처리할 수 있습니다.

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

그런 다음 변경 내용을 이전에 저장된 데이터와 비교할 수 있습니다.

다음 예제에서는 엔터티 소유권 변경 이벤트를 처리하는 방법을 보여줍니다. 이 예제에서는 구독 엔터티와 소유 엔터티 사이를 전환하는 엔터티의 경우 (어느 방향으로든) 소유권 제거/추가 이벤트가 먼저 발생하고 다음 틱에서 구독 제거/추가 이벤트가 발생한다고 가정합니다.

Example예제

```
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 및 오류 처리

Aws::WeaverRuntime::Result<T> 클래스는 타사 Outcome 라이브러리를 사용합니다. 다음 패턴을 사용하여 Result를 확인하고 API 직접 호출에서 반환된 오류를 캐치할 수 있습니다.

```
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 제어문 매크로

Aws::WeaverRuntime::Result<T> 반환 유형이 있는 함수 내에서 이전 코드 패턴 대신 WEAVERRUNTIME_TRY 매크로를 사용할 수 있습니다. 매크로는 전달된 함수를 실행합니다. 전달된 함수가 실패하면 매크로는 둘러싸는 함수가 오류를 반환하도록 합니다. 전달된 함수가 성공하면 실행은 다음 줄로 진행됩니다. 다음 예제에서는 이전 DoBeginUpdate() 함수의 재작성을 보여줍니다. 이 버전에서는 if-else 제어 구조 대신 WEAVERRUNTIME_TRY 매크로를 사용합니다. 함수의 반환 형식은 Aws::WeaverRuntime::Result<void>입니다.

```
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();
}
```

BeginUpdate()가 실패하면 매크로는 실패와 DoBeginUpdate()를 조기에 반환합니다. WEAVERRUNTIME_EXPECT_ERROR 매크로를 사용하여 BeginUpdate()에서 Aws::WeaverRuntime::ErrorCode를 가져올 수 있습니다. 다음 예제는 Update() 함수의 DoBeginUpdate() 호출 및 실패 시 오류 코드를 가져오는 방법을 보여줍니다.

```
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);

    }
}
```

Update()의 반환 유형을 Aws::WeaverRuntime::Result<void>로 변경하여 Update()를 호출하는 함수에서 BeginUpdate()의 오류 코드를 사용할 수 있도록 할 수 있습니다. 이 프로세스를 반복하여 호출 스택 아래쪽으로 오류 코드를 계속 보낼 수 있습니다.

일반 및 도메인 유형

SimSpace Weaver 앱 SDK는 단일 정밀도 데이터 형식 Api::Vector2F32 및 Api::BoundingBox2F32, 이중 정밀도 Api::Vector2F64 및를 제공합니다Api::BoundingBox2F64. 이러한 데이터 유형은 편리한 방법이 없는 수동적 데이터 구조입니다. 참고로 API는 Api::Vector2F32 및 Api::BoundingBox2F32만 사용합니다. 이러한 데이터 유형을 사용하여 구독을 만들고 수정할 수 있습니다.

SimSpace Weaver 데모 프레임워크는 Vector3 및 Aabb를 포함하는 수AzCore학 라이브러리의 최소 버전을 제공합니다. 자세한 내용은 다음의 파일 헤더를 참조하세요.

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

기타 앱 SDK 작업

주제

- AllSubscriptionEvents 및 OwnershipChanges에는 마지막 호출의 이벤트가 포함됨
- SubscriptionChangeList 처리 후 읽기 잠금 해제
- 테스트용 독립 실행형 앱 인스턴스 생성

AllSubscriptionEvents 및 OwnershipChanges에는 마지막 호출의 이벤트가 포함됨

Api::AllSubscriptionEvents() 및 Api::OwnershipChanges()에 대한 호출의 반환 값에는 마지막 틱이 아닌 마지막 호출의 이벤트가 포함됩니다. 다음 예제에서는 첫 번째 호출 후 해당 함수가 즉시 호출되기 때문에 secondSubscriptionEvents 및 secondOwnershipChangeList가 비어 있습니다.

10개의 틱을 기다린 다음 Api::AllSubscriptionEvents() 및 Api::OwnershipChanges()를 호출하면 해당 결과에 마지막 틱이 아닌 최근 10개 틱의 이벤트와 변경 내용이 포함됩니다.

Example예제

```
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**
>
> `AllSubscriptionEvents()` 함수는 구현되었지만 `SubscriptionEvents()` 함수는 구현되지 않았습니다.

SubscriptionChangeList 처리 후 읽기 잠금 해제

업데이트를 시작하면 이전 틱의 다른 파티션에 커밋된 데이터에 대한 공유 메모리 세그먼트가 생깁니다. 독자는 이러한 공유 메모리 세그먼트를 잠글 수 있습니다. 모든 독자가 잠금을 해제하기 전까지는 앱을 완전히 커밋할 수 없습니다. 최적화를 위해 앱은 `Api::SubscriptionChangelist` 항목을 처리한 후 잠금을 해제하도록 `Api::ReleaseReadLeases()`를 호출해야 합니다. 그러면 커밋 시 경합이 줄어듭니다. `Api::Commit()`은 기본적으로 읽기 임대를 해제하지만 구독 업데이트를 처리한 후 읽기 임대를 수동으로 릴리스하는 것이 가장 좋습니다.

Example예제

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

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

    WEAVERRUNTIME_EXPECT(Api::ReleaseReadLeases(transaction));

    ...
}
```

테스트용 독립 실행형 앱 인스턴스 생성

`Api::CreateStandaloneApplication()`을 통해 실제 시뮬레이션에서 코드를 실행하기 전에 독립 실행형 앱을 생성하여 앱 로직을 테스트할 수 있습니다.

Example예제

```
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 데모 프레임워크

AWS SimSpace Weaver 데모 프레임워크(데모 프레임워크)는 SimSpace Weaver 앱을 개발하는 데 사용할 수 있는 유틸리티 라이브러리입니다.

데모 프레임워크는 다음을 제공합니다.

- 사용 및 검토할 수 있는 코드 샘플 및 프로그래밍 패턴
- 간단한 앱 개발을 간소화하는 추상화 및 유틸리티 함수
- SimSpace Weaver 앱 SDK의 실험 기능을 테스트하는 더 간단한 방법

더 높은 성능을 제공하기 위해 SimSpace Weaver APIs에 대한 낮은 수준의 액세스를 갖춘 SimSpace Weaver 앱 SDK를 설계했습니다. 반대로 데모 프레임워크는 더 높은 수준의 추상화와 SimSpace Weaver 가 사용하기 쉬운 API에 대한 액세스를 제공하도록 설계했습니다. 사용 편의성 비용은 SimSpace Weaver 앱 SDK를 직접 사용할 때보다 성능이 낮습니다. 낮은 성능을 견딜 수 있는 시뮬레이션(예: 실시간 성능 요구 사항이 없는 시뮬레이션)은 데모 프레임워크를 사용하기에 좋은 후보일 수 있습니다. 데모 프레임워크는 완전한 툴킷이 아니므로 복잡한 애플리케이션에는 SimSpace Weaver 앱 SDK의 네이티브 기능을 사용하는 것이 좋습니다.

데모 프레임워크에는 다음이 포함되어 있습니다.

- 다음을 지원하고 시연하는 작업 코드 샘플:
  - 앱 흐름 관리
  - 콜백 기반 엔터티 이벤트 처리
- 다음의 타사 유틸리티 라이브러리 세트:
  - spdlog(로깅 라이브러리)
  - 다음만 포함하는 AZCore(수학 라이브러리)의 최소 버전
    - Vector3
    - Aabb
  - cxxopts(명령줄 옵션 구문 분석 라이브러리)
- 관련 유틸리티 함수 SimSpace Weaver

데모 프레임워크는 라이브러리, 소스 파일, CMakeLists 등으로 구성되어 있습니다. 파일은 SimSpace Weaver 앱 SDK 배포 가능 패키지에 포함되어 있습니다.

Service Quotas 작업

이 섹션에서는에 대한 서비스 할당량으로 작업하는 방법을 설명합니다 SimSpace Weaver. 할당량은 제한이라고도 합니다. Service Quotas 목록은 SimSpace Weaver 엔드포인트 및 할당량 섹션을 참조하세요. 이 섹션의 API는 앱 API 세트에서 가져온 것입니다. 앱 API는 서비스 API와 다릅니다. 앱 APIs입니다. SimSpace Weaver 로컬 시스템의 앱 SDK 폴더에서 앱 API에 대한 설명서를 찾을 수 있습니다.

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

주제

- 앱 제한 확인하기

앱 제한 확인하기

RuntimeLimits앱 API를 사용하여 앱의 제한을 쿼리할 수 있습니다.

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

파라미터

Application 및 앱

앱에 대한 참조입니다.

LimitType 유형

다음과 같은 제한 유형을 가진 열거형입니다.

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

다음 예제에서는 엔터티 수 제한을 쿼리합니다.

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

앱에서 사용하는 리소스의 양 가져오기

RuntimeMetrics 앱 API를 호출하여 앱에서 사용하는 리소스의 양을 가져올 수 있습니다.

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

파라미터

Application 및 앱

앱에 대한 참조입니다.

API는 지표가 포함된 struct에 대한 참조를 반환합니다. 카운터 지표에는 실행 중인 총 값이 포함되며 증가만 합니다. 게이지 지표에는 증가 또는 감소할 수 있는 값이 있습니다. 애플리케이션 런타임은 이벤트로 인해 값이 증가할 때마다 카운터를 업데이트합니다. 런타임은 API를 호출할 때만 게이지를 업데이트합니다. SimSpace Weaver 는 참조가 앱 수명 주기 동안 유효하도록 보장합니다. API를 반복해서 호출해도 참조는 변경되지 않습니다.

```
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
}
```

지표 재설정

ResetRuntimeMetrics 앱 API는 `AppRuntimeMetrics` struct의 값을 재설정합니다.

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

다음 예제에서는 앱에서 ResetRuntimeMetrics를 호출하는 방법을 보여줍니다.

```
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));
}
```

제한 초과됨

제한을 초과하는 앱 API 직접 호출은 `ErrorCode::CapacityExceeded`를 반환합니다. 단, 엔터티 전송은 예외입니다. SimSpace Weaver 는 커밋 및 BeginUpdate 앱 API 작업의 일부로 엔터티 전송을 비동기적으로 처리하므로 엔터티 전송 제한 때문에 전송이 실패할 경우 오류를 반환하는 특정 작업은 없습니다. 전송 실패를 감지하기 위해 `rejected_incoming_transfer_counter` 및 `rejected_outgoing_transfer_counter`(AppRuntimeMetrics struct)의 현재 값을 이전 값과 비교할 수 있습니다. 거부된 엔터티는 파티션에 포함되지 않지만 앱은 여전히 이를 시뮬레이션할 수 있습니다.

메모리 부족

SimSpace Weaver 는 가비지 수집기 프로세스를 사용하여 여유 메모리를 정리하고 해제합니다. 가비지 수집기가 메모리를 해제할 수 있는 속도보다 빠르게 데이터를 쓸 수 있습니다. 이 경우 쓰기 작업이 앱의 예약 메모리 제한을 초과할 수 있습니다. SimSpace Weaver 는 `OutOfMemory`(및 추가 세부 정보)가 포함된 메시지와 함께 내부 오류를 반환합니다. 자세한 내용은 여러 시간에 걸쳐 쓰기 작업 분산 단원을 참조하십시오.

모범 사례

다음의 모범 사례는 제한 초과되지 않도록 앱을 설계하기 위한 일반적인 지침입니다. 이는 특정 앱 디자인에는 적용되지 않을 수 있습니다.

자주 모니터링하고 속도 줄이기

지표를 자주 모니터링하여 제한에 가깝게 도달하는 작업의 속도를 늦춰야 합니다.

구독 제한 및 전송 제한 초과 방지

가능한 경우 시뮬레이션을 설계하여 원격 구독 및 엔터티 전송 횟수를 줄입니다. 배치 그룹을 사용하여 동일한 작업자에 여러 파티션을 배치하고 작업자 간에 엔터티를 원격으로 전송해야 하는 필요성을 줄일 수 있습니다.

여러 시간에 걸쳐 쓰기 작업 분산

틱의 업데이트 수와 크기는 트랜잭션을 커밋하는 데 필요한 시간과 메모리에 상당한 영향을 미칠 수 있습니다. 메모리 요구 사항이 크면 애플리케이션 런타임에 메모리가 부족해질 수 있습니다. 쓰기를 여러 시간에 걸쳐 분산하여 틱당 평균 총 업데이트 크기를 줄일 수 있습니다. 이를 통해 성능을 개선하고 제한 초과를 방지할 수 있습니다. 각 틱에 평균 12MB 또는 각 엔터티에 1.5KB를 초과하여 쓰지 않는 것이 좋습니다.

디버깅 시뮬레이션

다음과 같은 방법을 사용하여 시뮬레이션에 대한 정보를 확인할 수 있습니다.

주제

- SimSpace Weaver Local 사용 및 콘솔 출력 살펴보기
- Amazon CloudWatch Logs의 로그 살펴보기
- describe API 직접 호출 사용
- 클라이언트 연결

SimSpace Weaver Local 사용 및 콘솔 출력 살펴보기

먼저 로컬에서 시뮬레이션을 개발한 다음 AWS 클라우드에서 실행하는 것이 좋습니다. SimSpace Weaver Local를 실행하면 콘솔 출력을 직접 볼 수 있습니다. 자세한 내용은 의 로컬 개발 SimSpace Weaver 단원을 참조하십시오.

Amazon CloudWatch Logs의 로그 살펴보기

시뮬레이션 AWS 클라우드 을 실행하면 앱의 콘솔 출력이 Amazon CloudWatch Logs의 로그 스트림으로 전송됩니다. 시뮬레이션은 다른 로그 데이터도 씁니다. 시뮬레이션에서 로그 데이터를 쓰려면 시뮬레이션 스키마에서 로깅을 활성화해야 합니다. 자세한 내용은 SimSpace Weaver Amazon CloudWatch Logs의 로그 단원을 참조하십시오.

> **Warning**
>
> 시뮬레이션을 통해 대량의 로그 데이터가 생성될 수 있습니다. 로그 데이터는 매우 빠르게 증가할 수 있습니다. 로그를 면밀히 관찰하고 더 이상 실행할 필요가 없을 때는 시뮬레이션을 중단해야 합니다. 로깅으로 인해 비용이 많이 들 수 있습니다.

describe API 직접 호출 사용

다음과 서비스 API를 사용하여 AWS 클라우드에서 시뮬레이션에 대한 정보를 확인할 수 있습니다.

- ListSimulations -의 모든 시뮬레이션 목록을 가져옵니다 AWS 클라우드.

  Example예제

  ```
  aws simspaceweaver list-simulations
  ```

- DescribeSimulation - 시뮬레이션에 대한 세부 정보를 확인합니다.

  Example예제

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

- DescribeApp - 앱에 대한 세부 정보를 확인합니다.

  Example예제

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

SimSpace Weaver APIsSimSpace Weaver API 참조.

클라이언트 연결

시뮬레이션 스키마에서 `endpoint_config`를 사용하여 정의한 실행 중인 사용자 지정 또는 서비스 앱에 클라이언트를 연결할 수 있습니다. SimSpace Weaver 앱 SDK에는 샘플 애플리케이션을 보는 데 사용할 수 있는 샘플 클라이언트가 포함되어 있습니다. 이러한 샘플 클라이언트의 소스 코드와 샘플 애플리케이션을 살펴보고 자체 클라이언트를 만드는 방법을 확인할 수 있습니다. 샘플 클라이언트를 빌드하고 실행하는 방법에 대한 자세한 내용은의 자습서를 참조하세요시작하기 SimSpace Weaver.

다음 폴더에서 샘플 클라이언트의 소스 코드를 찾을 수 있습니다.

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

디버깅 로컬 시뮬레이션

Microsoft Visual Studio를 사용하여 SimSpace Weaver Local 앱을 디버깅할 수 있습니다. Visual Studio를 사용하여 디버깅하는 방법에 대한 자세한 내용은 Microsoft Visual Studio documentation 섹션을 참조하세요.

로컬 시뮬레이션 디버깅

1. schema.yaml이 작업 디렉터리에 있는지 확인합니다.
2. Visual Studio에서 디버깅하려는 각 앱(예: PathfindingSampleLocalSpatial 또는PathfindingSampleLocalView)의 컨텍스트 메뉴를 열고 디버깅 섹션에서 작업 디렉터리를 설정합니다.
3. 디버깅하려는 앱의 컨텍스트 메뉴를 열고 시작 프로젝트로 설정을 선택합니다.
4. F5를 선택하여 앱 디버깅을 시작합니다.

시뮬레이션을 디버깅하기 위한 요구 사항은 시뮬레이션을 정상적으로 실행하기 위한 요구 사항과 동일합니다. 스키마에 지정된 수의 공간 앱을 시작해야 합니다. 예를 들어 스키마가 2x2 그리드를 지정하고 디버그 모드에서 공간 앱을 시작하는 경우 디버그 모드인지 여부에 관계없이 공간 앱을 3개 더 시작할 때까지 시뮬레이션이 실행되지 않습니다.

사용자 지정 앱을 디버깅하려면 먼저 공간 앱을 시작한 다음 디버거에서 사용자 지정 앱을 시작해야 합니다.

참고로 시뮬레이션은 잠금 단계로 실행됩니다. 앱이 중단점에 도달하면 다른 모든 앱이 일시 중지됩니다. 해당 중단점에서 계속 진행하면 다른 앱도 계속 실행됩니다.

사용자 지정 컨테이너

AWS SimSpace Weaver 앱은 컨테이너화된 Amazon Linux 2 (AL2) 환경에서 실행됩니다. 에서 AWS 클라우드는 Amazon Elastic Container Registry(Amazon ECR)에서 제공하는 amazonlinux:2 이미지로 빌드된 Docker 컨테이너에서 시뮬레이션을 SimSpace Weaver 실행합니다. 사용자 지정 도커 이미지를 생성하여 Amazon ECR에 저장한 다음, 당사에서 제공하는 기본 도커 이미지 대신 이 이미지를 시뮬레이션에 사용할 수 있습니다.

사용자 지정 컨테이너를 사용하여 소프트웨어 종속성을 관리하고 표준 도커 이미지에 없는 추가 소프트웨어 구성 요소를 포함할 수 있습니다. 예를 들어 앱에서 사용하는 공개적으로 사용 가능한 소프트웨어 라이브러리를 컨테이너에 추가하고 앱 zip 파일에는 사용자 지정 코드만 넣을 수 있습니다.

> **Important**
>
> Amazon ECR 퍼블릭 갤러리 또는 프라이빗 Amazon ECR 레지스트리의 Amazon ECR 리포지토리에 호스팅된 AL2 도커 이미지만 지원합니다. Amazon ECR 외부에서 호스팅되는 도커 이미지는 지원되지 않습니다. Amazon ECR에 대한 자세한 내용은 Amazon Elastic Container Registry 설명서를 참조하세요.

주제

사용자 지정 컨테이너 생성

이 지침에서는 Docker 및 Amazon Elastic Container Registry(Amazon ECR)를 사용하는 방법을 알고 있다고 가정합니다. Amazon ECR에 대한 자세한 내용은 Amazon ECR 사용 설명서를 참조하세요.

사전 조건

- 이러한 작업을 수행하는 데 사용하는 IAM 자격 증명(사용 또는 역할)에는 Amazon ECR을 사용할 수 있는 올바른 권한이 있습니다.
- 도커는 로컬 시스템에 설치됨

사용자 지정 컨테이너 생성

1. `Dockerfile`을 생성합니다.

   AWS SimSpace Weaver 앱을 실행`Dockerfile`하기 위한는 Amazon ECR의 Amazon Linux 2 이미지로 시작합니다.

   ```
   # parent image required to run AWS SimSpace Weaver apps
   ```

```
FROM public.ecr.aws/amazonlinux/amazonlinux:2
```

2. Dockerfile을 구축합니다.
3. 컨테이너 이미지를 Amazon ECR로 업로드합니다.
   - AWS Management Console을 사용합니다.
   - AWS Command Line Interface를 사용합니다.

> **Note**
>
> Amazon ECR에 컨테이너 이미지를 업로드하려고 할 때 AccessDeniedException 오류가 발생하는 경우, IAM 자격 증명(사용자 또는 역할)에 Amazon ECR을 사용하는 데 필요한 권한이 없는 것일 수 있습니다. AmazonEC2ContainerRegistryPowerUser AWS 관리형 정책을 IAM 자격 증명에 연결하고 다시 시도할 수 있습니다. 정책 연결 방법에 대한 자세한 내용은 AWS Identity and Access Management 사용 설명서의 IAM 자격 증명 권한 추가 및 제거를 참조하세요.

사용자 지정 컨테이너를 사용하도록 프로젝트 수정

이 지침에서는를 사용하는 방법을 이미 알고 있으며 앱 스토리지 AWS SimSpace Weaver 및 개발 워크플로를 AWS 클라우드 보다 효율적으로 만들고 싶다고 가정합니다.

사전 조건

- Amazon Elastic Container Registry(Amazon ECR)에 사용자 지정 컨테이너가 있습니다. 사용자 지정 컨테이너 생성에 대한 자세한 내용은 사용자 지정 컨테이너 생성 섹션을 참조하세요.

사용자 지정 컨테이너를 사용하도록 프로젝트 수정

1. 프로젝트의 시뮬레이션 앱 역할에 Amazon ECR을 사용할 수 있는 권한을 추가합니다.
   a. 다음 권한이 있는 IAM 정책이 아직 없다면 정책을 생성합니다. 정책 이름 simspaceweaver-ecr을 사용하는 것이 좋습니다. IAM 정책 생성 방법에 대한 자세한 내용은 AWS Identity and Access Management 사용 설명서의 IAM 정책 생성을 참조하세요.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
```

```
            "Sid": "Statement",
            "Effect": "Allow",
            "Action": [
                "ecr:BatchGetImage",
                "ecr:GetDownloadUrlForLayer",
                "ecr:GetAuthorizationToken"
            ],
            "Resource": "*"
        }
    ]
}
```

b. 프로젝트의 시뮬레이션 앱 역할 이름을 찾으려면 다음을 수행합니다.

   i. 텍스트 편집기에서 CloudFormation 템플릿을 엽니다.

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

   ii. `WeaverAppRole`에서 `RoleName` 속성을 찾습니다. 해당 값은 프로젝트의 시뮬레이션 앱 역할 이름입니다.

   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. `simspaceweaver-ecr` 정책을 프로젝트의 시뮬레이션 앱 역할에 연결합니다. 정책 연결 방법에 대한 자세한 내용은 AWS Identity and Access Management 사용 설명서의 IAM 자격 증명 권한 추가 및 제거를 참조하세요.

d. 다음 명령으로 이동하여 *sdk-folder* 실행하여 샘플 SimSpace Weaver 스택을 업데이트합니다.

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

2. 프로젝트의 시뮬레이션 스키마에 컨테이너 이미지를 지정합니다.

   - default_image에서 선택적 simulation_properties 속성을 추가하여 모든 도메인의 기본 사용자 지정 컨테이너 이미지를 지정할 수 있습니다.
   - 사용자 지정 컨테이너 이미지로 사용하려는 도메인에 대해 image 속성을 app_config에 추가합니다. Amazon ECR 리포지토리 URI를 값으로 지정합니다. 도메인마다 다른 이미지를 지정할 수 있습니다.
     - 도메인에 image이 지정되지 않고 default_image가 지정된 경우 해당 도메인의 앱이 기본 이미지를 사용합니다.
     - image가 도메인에 지정되지 않고가 지정되지 않은 경우 해당 도메인의 앱default_image은 표준 SimSpace Weaver 컨테이너에서 실행됩니다.

   Example사용자 지정 컨테이너 설정이 포함된 스키마 스니펫

```
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. 평소와 같이 프로젝트를 빌드하고 업로드합니다.

사용자 지정 컨테이너에 대한 FAQ

Q1. 컨테이너 내용을 변경하려면 어떻게 해야 하나요?

- 실행 중인 시뮬레이션의 경우 - 실행 중인 시뮬레이션의 컨테이너는 변경할 수 없습니다. 새 컨테이너를 빌드하고 해당 컨테이너를 사용하는 새 시뮬레이션을 시작해야 합니다.
- 새 시뮬레이션의 경우 - 새 컨테이너를 구축하고 Amazon Elastic Container Registry(Amazon ECR)에 업로드한 다음 해당 컨테이너를 사용하는 새 시뮬레이션을 시작합니다.

Q2. 시뮬레이션에 사용할 컨테이너 이미지를 변경하려면 어떻게 해야 하나요?

- 실행 중인 시뮬레이션의 경우 - 실행 중인 시뮬레이션의 컨테이너는 변경할 수 없습니다. 새 컨테이너를 사용하는 새 시뮬레이션을 시작해야 합니다.
- 새 시뮬레이션의 경우 - 프로젝트의 시뮬레이션 스키마에 새 컨테이너 이미지를 지정합니다. 자세한 내용은 사용자 지정 컨테이너를 사용하도록 프로젝트 수정 단원을 참조하십시오.

사용자 지정 컨테이너 문제 해결

주제

- Amazon Elastic Container Registry(Amazon ECR)에 이미지를 업로드하는 경우의 AccessDeniedException
- 사용자 지정 컨테이너를 사용하는 시뮬레이션이 시작되지 않음

Amazon Elastic Container Registry(Amazon ECR)에 이미지를 업로드하는 경우의 AccessDeniedException

Amazon ECR에 컨테이너 이미지를 업로드하려고 할 때 `AccessDeniedException` 오류가 발생하는 경우, IAM 자격 증명(사용자 또는 역할)에 Amazon ECR을 사용하는 데 필요한 권한이 없는 것일 수 있습니다. `AmazonEC2ContainerRegistryPowerUser` AWS 관리형 정책을 IAM 자격 증명에 연결하고 다시 시도할 수 있습니다. 정책 연결 방법에 대한 자세한 내용은 AWS Identity and Access Management 사용 설명서의 IAM 자격 증명 권한 추가 및 제거를 참조하세요.

사용자 지정 컨테이너를 사용하는 시뮬레이션이 시작되지 않음

문제 해결 팁

- 시뮬레이션에 로깅이 활성화된 경우 오류 로그를 확인하세요.
- 사용자 지정 컨테이너 없이 시뮬레이션을 테스트합니다.
- 시뮬레이션을 로컬에서 테스트합니다. 자세한 내용은 의 로컬 개발 SimSpace Weaver 단원을 참조하십시오.

Python 작업

Python을 SimSpace Weaver 앱과 클라이언트에 사용할 수 있습니다. Python 소프트웨어 개발 키트(Python SDK)는 표준 SimSpace Weaver 앱 SDK 배포 가능 패키지의 일부로 포함되어 있습니다. Python을 사용한 개발은 지원되는 다른 언어에서의 개발과 비슷한 방식으로 작동합니다.

> **Important**
>
> SimSpace Weaver 는 Python 버전 3.9만 지원합니다.

> **Important**
>
> SimSpace Weaver Python에 대한 지원에는 SimSpace Weaver 버전 1.15.0 이상이 필요합니다.

주제

- Python 프로젝트 생성

- Python 시뮬레이션 시작하기
- 샘플 Python 클라이언트
- Python 사용에 대한 FAQ
- Python과 관련된 문제 해결

Python 프로젝트 생성

Python 사용자 지정 컨테이너

에서 Python 기반 SimSpace Weaver 시뮬레이션을 실행하려면 필요한 종속성이 포함된 사용자 지정 컨테이너를 생성할 AWS 클라우드수 있습니다. 자세한 내용은 사용자 지정 컨테이너 단원을 참조하십시오.

Python 사용자 지정 컨테이너에는 다음이 포함되어야 합니다.

- gcc
- openssl-devel
- bzip2-devel
- libfi-devel
- wget
- tar
- gzip
- make
- Python(버전 3.9)

`PythonBubblesSample` 템플릿을 사용하여 프로젝트를 생성하는 경우 프로젝트의 `tools` 폴더에 있는 `quick-start.py` 스크립트를 실행하여 필요한 종속성이 있는 도커 이미지를 만들 수 있습니다. 이 스크립트는 Amazon Elastic Container Registry(Amazon ECR)에 이미지를 업로드합니다.

`quick-start.py` 스크립트는 `Dockerfile`을 사용합니다.

```
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
```

`Dockerfile`에 자체 종속성을 추가할 수 있습니다.

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

`requirements.txt` 파일에는 `PythonBubblesSample` 샘플 시뮬레이션에 필요한 Python 패키지 목록이 들어 있습니다.

```
Flask==2.1.1
```

`requirements.txt`에 자체 Python 패키지 종속성을 추가할 수 있습니다.

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

`Dockerfile` 및 `requirements.txt`는 프로젝트의 `tools` 폴더에 있습니다.

> **Important**
>
> 엄밀히 따지자면 Python 시뮬레이션에서 사용자 지정 컨테이너를 사용할 필요는 없지만 사용자 지정 컨테이너를 사용하는 것이 좋습니다. 우리가 제공하는 표준 Amazon Linux 2(AL2) 컨테이너에는 Python이 없습니다. 따라서 Python이 있는 사용자 지정 컨테이너를 사용하지 않는 경우 업로드하는 각 앱 zip 파일에 Python과 필수 종속성을 포함해야 합니다 SimSpace Weaver.

Python 시뮬레이션 시작하기

의 SimSpace Weaver Local 및에서 일반 시뮬레이션과 동일한 방식으로 Python 기반 SimSpace Weaver 시뮬레이션을 시작할 수 SimSpace Weaver 있습니다 AWS 클라우드. 자세한 내용은의 자습서를 참조하세요시작하기 SimSpace Weaver.

PythonBubblesSample에는 자체 Python 샘플 클라이언트가 포함되어 있습니다. 자세한 내용은 샘플 Python 클라이언트 단원을 참조하십시오.

샘플 Python 클라이언트

PythonBubblesSample 템플릿을 사용하여 프로젝트를 만드는 경우 프로젝트에 Python 샘플 클라이언트가 포함됩니다. 샘플 클라이언트를 사용하여 PythonBubblesSample 시뮬레이션을 볼 수 있습니다. 샘플 클라이언트를 시작점으로 사용하여 자체 Python 클라이언트를 만들 수도 있습니다.

다음 절차에서는 PythonBubblesSample 프로젝트를 생성하고 시뮬레이션을 시작했다고 가정합니다.

Python 클라이언트 시작

1. 명령 프롬프트 창에서 PyBubbleClient 샘플 프로젝트 폴더로 이동합니다.

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

2. Python 클라이언트를 실행합니다.

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

   파라미터

   host

   시뮬레이션의 IP 주소입니다. 에서 시작된 시뮬레이션의 경우 SimSpace Weaver 콘솔에서 시뮬레이션의 IP 주소를 찾거나 빠른 시작 자습서사용자 지정 앱의 IP 주소 및 포트 번호 가져오기의의 절차를 사용할 AWS 클라우드수 있습니다. 로컬 시뮬레이션의 경우 127.0.0.1을 IP 주소로 사용합니다.

   port

   시뮬레이션의 포트 번호입니다. 에서 시작된 시뮬레이션의 경우 AWS 클라우드포트 Actual 번호입니다. SimSpace Weaver 콘솔에서 시뮬레이션의 포트 번호를 찾거나 빠른 시작 자습서의 사용자 지정 앱의 IP 주소 및 포트 번호 가져오기 절차를 사용할 수 있습니다. 로컬 시뮬레이션의 경우 7000을 포트 번호로 사용합니다.

   simsize

   클라이언트에 표시할 최대 엔터티 수입니다.

Python 사용에 대한 FAQ

Q1. 어떤 버전의 Python이 지원되나요?

SimSpace Weaver 는 Python 버전 3.9만 지원합니다.

Python과 관련된 문제 해결

주제

- 사용자 지정 컨테이너 생성 중 실패
- Python 시뮬레이션이 시작되지 않음
- Python 시뮬레이션 또는 보기 클라이언트에서 ModuleNotFound 오류 발생

사용자 지정 컨테이너 생성 중 실패

`quick-start.py` 실행 후 `no basic auth credentials` 오류가 발생하면 Amazon ECR의 임시 보안 인증에 문제가 있을 수 있습니다. AWS 리전 ID 및 AWS 계정 번호로 다음 명령을 실행합니다.

```
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**
>
> AWS 리전 지정한가 시뮬레이션에 사용하는 것과 동일한지 확인합니다. 에서 AWS 리전 SimSpace Weaver 지원하는 중 하나를 사용합니다. 자세한 내용은 SimSpace Weaver 엔드포인트 및 할당량 단원을 참조하십시오.

`aws ecr` 명령을 실행한 후 `quick-start.py`를 다시 실행합니다.

확인할 기타 문제 해결 리소스

- 사용자 지정 컨테이너 문제 해결

- Amazon ECR 사용 설명서의 Amazon ECR 문제 해결
- 자세한 내용은 Amazon ECR 사용 설명서의 Amazon ECR을 사용하여 설정을 참조하세요.

Python 시뮬레이션이 시작되지 않음

시뮬레이션의 관리 로그에 `Unable to start app` 오류가 표시될 수 있습니다. 이는 사용자 지정 컨테이너 생성이 실패한 경우 발생할 수 있습니다. 자세한 내용은 사용자 지정 컨테이너 생성 중 실패 단원을 참조하십시오. 로그에 대한 자세한 내용은 SimSpace Weaver Amazon CloudWatch Logs의 로그 섹션을 참조하세요.

컨테이너에 문제가 없는 것이 확실하다면 앱의 Python 소스 코드를 확인합니다. SimSpace Weaver Local을 사용하여 앱을 테스트할 수 있습니다. 자세한 내용은 의 로컬 개발 SimSpace Weaver 단원을 참조하십시오.

Python 시뮬레이션 또는 보기 클라이언트에서 ModuleNotFound 오류 발생

필요한 Python 패키지를 찾을 수 없는 경우 Python에서 `ModuleNotFound` 오류가 발생합니다.

시뮬레이션이에 있는 경우 사용자 지정 컨테이너에에 나열된 모든 필수 종속성이 있는지 AWS 클라우드확인합니다`requirements.txt`. `requirements.txt`를 편집하는 경우 `quick-start.py`를 다시 실행해야 한다는 점을 잊지 마세요.

`PythonBubblesSample` 클라이언트에서 오류가 발생하는 경우 `pip`를 사용하여 표시된 패키지를 설치합니다.

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

기타 엔진 지원

자체 사용자 지정 C++ 엔진을와 함께 사용할 수 있습니다 SimSpace Weaver. 현재 다음 엔진에 대한 지원을 개발 중입니다. 각 엔진에 대한 별도의 설명서가 있습니다.

> **Important**
>
> 여기에 나열된 엔진과의 통합은 실험용입니다. 미리 보기를 사용할 수 있습니다.

엔진

- Unity (최소 버전 2022.3.19.F1)
- Unreal Engine(최소 버전 5.0)

Unity

Unity로 SimSpace Weaver 시뮬레이션을 빌드하기 전에 Unity 개발 환경이 이미 설치되어 있어야 합니다. 자세한 내용은 별도의 지침을 참조하세요.

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

Unreal Engine

소스 코드에서 Unreal Engine 전용 서버를 빌드해야 합니다. SimSpaceWeaverAppSdkDistributable에는 Unreal Engine에 대한 PathfindingSample이 포함되어 있습니다. 자세한 내용은 별도의 지침을 참조하세요.

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

에서 라이선스 소프트웨어 사용 AWS SimSpace Weaver

AWS SimSpace Weaver 를 사용하면 선택한 시뮬레이션 엔진 및 콘텐츠로 시뮬레이션을 빌드할 수 있습니다. 사용과 관련하여 시뮬레이션에 사용하는 소프트웨어 또는 콘텐츠의 라이선스 조건을 획득, 유지 및 준수할 SimSpace Weaver책임은 사용자에게 있습니다. 라이선싱 계약에서 소프트웨어와 콘텐츠를 가상 호스팅 환경에서 배포하도록 허용하는지 확인합니다.

를 사용하여 리소스 관리 AWS CloudFormation

AWS CloudFormation 를 사용하여 AWS SimSpace Weaver 리소스를 관리할 수 있습니다. CloudFormation 는 AWS 인프라를 코드로 지정, 프로비저닝 및 관리하는 데 도움이 되는 별도의 AWS 서비스입니다. 를 CloudFormation 사용하여 템플릿이라는 JSON 또는 YAML 파일을 생성합니다. 템플릿은 인프라의 세부 정보를 지정합니다. CloudFormation 은 템플릿을 사용하여 인프라를 스택이라는 단일 단위로 프로비저닝합니다. 스택을 삭제하면 스택의 모든 항목을 동시에 CloudFormation 삭제할 수 있습니다. 표준 소스 코드 관리 프로세스(예: Git와 같은 버전 관리 시스템에서 추적)를 사용하여 템플릿을 관리할 수 있습니다. 에 대한 자세한 내용은 AWS CloudFormation 사용 설명서를 CloudFormation참조하세요.

시뮬레이션 리소스

에서 AWS리소스는 작업할 수 있는 엔터티입니다. 예를 들어 Amazon EC2 인스턴스, Amazon S3 버킷 또는 IAM 역할이 있습니다. SimSpace Weaver 시뮬레이션은 리소스입니다. 구성에서는 일반적으로 형식으로 AWS 리소스를 지정합니다`AWS::service::resource`. 의 SimSpace Weaver경우 시뮬레이션 리소스를 로 지정합니다`AWS::SimSpaceWeaver::Simulation`. 의 시뮬레이션 리소스에 대한 자세한 내용은 AWS CloudFormation 사용 설명서의 SimSpace Weaver 섹션을 CloudFormation참조하세요.

와 CloudFormation 함께를 사용하려면 어떻게 해야 하나요 SimSpace Weaver?

프로비저닝하려는 AWS 리소스를 지정하는 CloudFormation 템플릿을 생성할 수 있습니다. 템플릿은 전체 아키텍처, 아키텍처의 일부 또는 소규모 솔루션을 지정할 수 있습니다. 예를 들어 Amazon S3 버킷, IAM 권한, Amazon Relational Database Service 또는 Amazon DynamoDB의 지원 데이터베이스, `Simulation` 리소스를 포함하는 SimSpace Weaver 솔루션의 아키텍처를 지정할 수 있습니다. 그런 다음 CloudFormation 를 사용하여 이러한 모든 리소스를 하나의 단위로 프로비저닝하면서 동시에 프로비저닝할 수 있습니다.

Example IAM 리소스를 생성하고 시뮬레이션을 시작하는 템플릿

다음 예제 템플릿은 SimSpace Weaver 가 계정에서 작업을 수행하는 데 필요한 IAM 역할 및 권한을 생성합니다. SimSpace Weaver 앱 SDK 스크립트는 프로젝트를 생성할 AWS 리전 때 특정에서 역할과 권한을 생성하지만, CloudFormation 템플릿을 사용하여 스크립트를 다시 실행하지 않고도 시뮬레이션을 다른 AWS 리전 에 배포할 수 있습니다. 예를 들어 재해 복구를 위한 백업 시뮬레이션을 설정하기 위해 이 작업을 수행할 수 있습니다.

이 예제에서 원래 시뮬레이션 이름은 `MySimulation`입니다. 스키마용 버킷은 CloudFormation 가 스택을 빌드 AWS 리전 하는에 이미 있습니다. 버킷에는 해당 AWS 리전에서 시뮬레이션을 실행하도록 적절하게 구성된 스키마 버전이 포함되어 있습니다. 스키마는 시뮬레이션과 동일한 AWS 리전 의 Amazon S3 버킷인 앱 zip 파일의 위치를 지정한다는 점을 기억하세요. 앱은 스택을 CloudFormation 빌드할 AWS 리전 때 버킷과 파일이 이미에 있어야 합니다. 그렇지 않으면 시뮬레이션이 시작되지 않습니다. 이 예제의 버킷 이름에는이 포함되지만 AWS 리전,는 버킷이 실제로 어디에 있는지를 결정하지 않습니다. 버킷이 실제로 여기에 있는지 확인해야 합니다 AWS 리전 (Amazon S3 콘솔, Amazon S3 APIs 또는의 Amazon S3 Amazon S3 명령을 사용하여 버킷 속성을 확인할 수 있음 AWS CLI).

이 예제에서는의 일부 내장 함수와 파라미터를 사용하여 변수 대체 CloudFormation 를 수행합니다. 자세한 내용은 AWS CloudFormation 사용 설명서의 내장 함수 참조 및 유사 파라미터 참조를 참조하세요.

```
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'
```

에서 스냅샷 사용 AWS CloudFormation

스냅샷은 시뮬레이션의 백업입니다. 다음 예제에서는 스키마 대신 스냅샷에서 새 시뮬레이션을 시작합니다. 이 예제의 스냅샷은 SimSpace Weaver 앱 SDK 프로젝트 시뮬레이션에서 생성되었습니다.는 새 시뮬레이션 리소스를 CloudFormation 생성하고 스냅샷의 데이터로 초기화합니다. 새 시뮬레이션은 원래 시뮬레이션과 `MaximumDuration`이 다를 수 있습니다.

원래 시뮬레이션의 앱 역할 사본을 만들어 사용하는 것이 좋습니다. 시뮬레이션의 CloudFormation 스택을 삭제하면 원래 시뮬레이션의 앱 역할이 삭제될 수 있습니다.

```
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"
```

스냅샷

스냅샷을 생성하여 언제든지 시뮬레이션 엔터티 데이터를 백업할 수 있습니다. SimSpace Weaver 는 Amazon S3 버킷에서.zip 파일을 만듭니다. 스냅샷으로 새 시뮬레이션을 생성할 수 있습니다.는 스냅샷에 저장된 개체 데이터로 새 시뮬레이션의 상태 패브릭을 SimSpace Weaver 초기화하고, 스냅샷이 생성될 때 실행 중인 공간 및 서비스 앱을 시작하고, 클럭을 적절한 틱으로 설정합니다.는 스키마 파일 대신 스냅샷에서 시뮬레이션 구성을 SimSpace Weaver 가져옵니다. 앱 .zip 파일은 원래 시뮬레이션과 동일한 Amazon S3 위치에 있어야 합니다. 모든 사용자 지정 앱을 개별적으로 시작해야 합니다.

주제

- 스냅샷의 사용 사례

- SimSpace Weaver 콘솔을 사용하여 스냅샷 작업
- AWS CLI 를 사용하여 스냅샷 작업
- 에서 스냅샷 사용 AWS CloudFormation
- 스냅샷에 대한 FAQ

스냅샷의 사용 사례

이전 상태로 돌아가 분기 시나리오를 살펴봅니다.

시뮬레이션의 스냅샷을 생성하여 특정 상태로 저장할 수 있습니다. 그런 다음 해당 스냅샷에서 새 시뮬레이션을 여러 개 만들고 해당 상태에서 분기할 수 있는 다양한 시나리오를 탐색할 수 있습니다.

재해 복구 및 보안 모범 사례

시뮬레이션을 정기적으로 백업하는 것이 좋습니다. 특히 1시간 이상 실행하거나 여러 작업자를 사용하는 시뮬레이션의 경우에는 더욱 그렇습니다. 백업은 재해 및 보안 사고로부터 복구하는 데 도움이 될 수 있습니다. 스냅샷은 시뮬레이션을 백업할 수 있는 방법을 제공합니다. 스냅샷을 사용하려면 앱 .zip 파일이 Amazon S3의 이전 위치와 동일한 위치에 있어야 합니다. 앱 .zip 파일을 다른 위치로 이동할 수 있어야 하는 경우 사용자 지정 백업 솔루션을 사용해야 합니다.

모범 사례에 대한 자세한 내용은 작업 시 모범 사례 SimSpace Weaver 및 에 대한 보안 모범 사례 SimSpace Weaver 섹션을 참조하세요.

시뮬레이션 지속 시간 연장

시뮬레이션 리소스는 SimSpace Weaver에서 시뮬레이션을 표현한 것입니다. 모든 시뮬레이션 리소스에는 `MaximumDuration` 설정이 있습니다. 시뮬레이션 리소스가 해당 `MaximumDuration`에 도달하면 자동으로 중지됩니다. `MaximumDuration`의 최대값은 `14D`(14일)입니다.

시뮬레이션을 해당 시뮬레이션 리소스의 `MaximumDuration`보다 오래 지속해야 하는 경우 시뮬레이션 리소스가 해당 `MaximumDuration`에 도달하기 전에 스냅샷을 만들 수 있습니다. 스냅샷으로 새 시뮬레이션을 시작(새 시뮬레이션 리소스 생성)할 수 있습니다. SimSpace Weaver 는 스냅샷에서 엔터티 데이터를 초기화하고, 이전에 실행했던 것과 동일한 공간 및 서비스 앱을 시작하고, 클럭을 복원합니다. 사용자 지정 앱을 시작하고 추가 사용자 지정 초기화를 수행할 수 있습니다. 새 시뮬레이션 리소스를 시작하는 경우 새 시뮬레이션 리소스의 `MaximumDuration`을 다른 값으로 설정할 수 있습니다.

SimSpace Weaver 콘솔을 사용하여 스냅샷 작업

SimSpace Weaver 콘솔을 사용하여 시뮬레이션의 스냅샷을 생성할 수 있습니다.

주제

콘솔을 사용하여 스냅샷 생성

스냅샷 생성

1. 에 로그인 AWS Management Console 하고 SimSpace Weaver 콘솔에 연결합니다.
2. 탐색 창에서 시뮬레이션을 선택합니다.
3. 시뮬레이션 이름 옆의 라디오 버튼을 선택합니다. 시뮬레이션 상태가 시작됨이어야 합니다.
4. 페이지 상단에서 스냅샷 생성을 선택합니다.
5. 스냅샷 설정의 스냅샷 대상에 스냅샷을 생성하려는 버킷 또는 버킷 및 폴더의 Amazon S3 URI SimSpace Weaver 를 입력합니다. 사용 가능한 버킷을 찾아보고 위치를 선택하려는 경우 S3 찾아보기를 선택할 수 있습니다.

   > **Important**
   >
   > Amazon S3 버킷은 시뮬레이션과 동일한 AWS 리전 에 있어야 합니다.

   > **Note**
   >
   > SimSpace Weaver 는 선택한 스냅샷 대상 내에 `snapshot` 폴더를 생성합니다.는 해당 `snapshot` 폴더에 스냅샷 .zip 파일을 SimSpace Weaver 생성합니다.

6. 스냅샷 생성(Create snapshot)을 선택합니다.

콘솔을 사용하여 스냅샷에서 시뮬레이션 시작

스냅샷에서 시뮬레이션을 시작하려면 시뮬레이션에서 액세스할 수 있는 Amazon S3 버킷에 스냅샷 .zip 파일이 있어야 합니다. 시뮬레이션에서는 시뮬레이션을 시작할 때 선택한 앱 역할에 정의된 권한을 사용합니다. 원본 시뮬레이션의 모든 앱 .zip 파일은 스냅샷을 만들 때와 같은 위치에 있어야 합니다.

스냅샷에서 시뮬레이션 시작

1. 에 로그인 AWS Management Console 하고 SimSpace Weaver 콘솔에 연결합니다.
2. 탐색 창에서 시뮬레이션을 선택합니다.
3. 페이지 상단에서 시뮬레이션 시작을 선택합니다.
4. 시뮬레이션 설정에서 시뮬레이션의 이름 및 선택적 설명을 입력합니다. 시뮬레이션 이름은 AWS 계정에서 고유해야 합니다.
5. 시뮬레이션 시작 방법에서 Amazon S3의 스냅샷 사용을 선택합니다.
6. 스냅샷용 Amazon S3 URI의 경우 스냅샷 파일의 Amazon S3 URI를 입력하거나 S3 찾아보기를 선택하여 파일을 찾아 선택합니다.

   > **Important**
   >
   > Amazon S3 버킷은 시뮬레이션과 동일한 AWS 리전 에 있어야 합니다.

7. IAM 역할의 경우 시뮬레이션에서 사용할 앱 역할을 선택합니다.
8. 최대 지속 시간에는 시뮬레이션 리소스를 실행해야 하는 최대 시간을 입력합니다. 최대 값은 14D입니다. 시뮬레이션의 최대 지속 시간에 대한 자세한 내용은 https://docs.aws.amazon.com/simspaceweaver/latest/APIReference/API_StartSimulation.html 섹션을 참조하세요.
9. 태그 - 선택 사항에서 태그를 추가하려면 새 태그 추가를 선택합니다.
10. 시뮬레이션 시작을 선택합니다.

AWS CLI 를 사용하여 스냅샷 작업

를 사용하여 명령 프롬프트에서 SimSpace Weaver APIs를 호출 AWS CLI 할 수 있습니다. 를 올바르게 AWS CLI 설치하고 구성해야 합니다. 자세한 내용은 버전 2 사용 설명서의 최신 버전의 AWS CLI 설치 또는 업데이트를 참조하세요. AWS Command Line Interface

주제

- AWS CLI 를 사용하여 스냅샷 생성
- AWS CLI 를 사용하여 스냅샷에서 시뮬레이션 시작

AWS CLI 를 사용하여 스냅샷 생성

스냅샷 생성

- 명령 프롬프트에서 CreateSnapshot API를 호출합니다.

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

파라미터

시뮬레이션

시작된 시뮬레이션의 이름입니다. aws simspaceweaver list-simulations를 사용하여 시뮬레이션의 이름과 상태를 볼 수 있습니다.

destination

대상 Amazon S3 버킷과 스냅샷 파일의 선택적 객체 키 접두사를 지정하는 문자열입니다. 객체 키 접두사는 일반적으로 버킷의 폴더입니다.는이 대상의 snapshot 폴더 내에 스냅샷을 SimSpace Weaver 생성합니다.

> **Important**
>
> Amazon S3 버킷은 시뮬레이션과 동일한 AWS 리전 에 있어야 합니다.

예

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

CreateSnapshot API에 대한 자세한 내용은 AWS SimSpace Weaver API 참조의 CreateSnapshot을 참조하세요.

AWS CLI 를 사용하여 스냅샷에서 시뮬레이션 시작

스냅샷에서 시뮬레이션 시작

- 명령 프롬프트에서 `StartSimulation` API를 호출합니다.

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

파라미터

이름

새 시뮬레이션의 이름입니다. 시뮬레이션 이름은에서 고유해야 합니다 AWS 계정. `aws simspaceweaver list-simulations`를 사용하여 기존 시뮬레이션의 이름을 볼 수 있습니다.

role-arn

시뮬레이션에서 사용할 앱 역할의 Amazon 리소스 이름(ARN)입니다.

snapshot-s3-location

대상 Amazon S3 버킷과 스냅샷 파일의 객체 키를 지정하는 문자열입니다.

> **Important**
> Amazon S3 버킷은 시뮬레이션과 동일한 AWS 리전 에 있어야 합니다.

예

```
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
```

`StartSimulation` API에 대한 자세한 내용은 AWS SimSpace Weaver API 참조의 StartSimulation을 참조하세요.

스냅샷에 대한 FAQ

스냅샷 중에도 시뮬레이션이 계속 실행되나요?

시뮬레이션 리소스는 스냅샷이 생성되는 동안 계속 실행되며 해당 기간에 대한 결제 요금은 계속 청구됩니다. 시간은 시뮬레이션의 최대 지속 시간에 포함됩니다. 스냅샷이 진행되는 동안에는 앱에 틱이 수신되지 않습니다. 클럭 상태가 스냅샷 생성을 시작했던 STARTED 상태인 경우 클럭에는 계속 STARTED 상태가 표시됩니다. 스냅샷이 완료된 후 앱이 틱을 다시 수신합니다. 클럭 상태가 STOPPED 상태였던 경우 클럭 상태는 STOPPED 상태로 그대로 유지됩니다. STARTED 상태의 시뮬레이션은 클럭 상태가 STOPPED 상태이더라도 실행 중이라는 점에 유의하세요.

스냅샷이 진행 중이고 시뮬레이션이 최대 지속 시간에 도달하면 어떻게 되나요?

시뮬레이션은 스냅샷을 완료한 다음 스냅샷 프로세스가 종료되는 즉시 중지됩니다(성공 또는 실패). 스냅샷 프로세스를 미리 테스트하여 소요 시간, 예상 스냅샷 파일 크기, 성공적으로 완료되는지 여부 등을 확인하는 것이 좋습니다.

스냅샷이 진행 중인 시뮬레이션을 중지하면 어떻게 되나요?

시뮬레이션을 중지하면 진행 중인 스냅샷이 즉시 중지됩니다. 스냅샷 파일은 생성되지 않습니다.

진행 중인 스냅샷을 어떻게 중지할 수 있나요?

진행 중인 스냅샷을 중지하는 유일한 방법은 시뮬레이션을 중지하는 것입니다. 시뮬레이션을 중지한 후에는 다시 시작할 수 없습니다.

스냅샷을 완료하는 데 얼마나 걸리나요?

스냅샷을 만드는 데 필요한 시간은 시뮬레이션에 따라 달라집니다. 스냅샷 프로세스를 미리 테스트하여 시뮬레이션에 소요되는 시간을 확인하는 것이 좋습니다.

스냅샷 파일 크기는 얼마나 되나요?

스냅샷 파일의 크기는 시뮬레이션에 따라 달라집니다. 스냅샷 프로세스를 미리 테스트하여 시뮬레이션에 사용할 수 있는 파일 크기를 확인하는 것이 좋습니다.

메시징

메시징 API는 시뮬레이션 내에서 애플리케이션 간 통신을 간소화합니다. 메시지를 보내고 받는 APIs는 SimSpace Weaver 앱 SDK의 일부입니다. 메시징은 현재 메시지를 보내고 받는 데 최선의 노력을 기울입니다. SimSpace Weaver는 다음 시뮬레이션 틱에서 메시지를 전송/수신하려고 시도하지만 배달, 주문 또는 도착 시간 보장은 없습니다.

주제

메시징 사용 사례

시뮬레이션 애플리케이션 간 통신

메시징 API를 사용하여 시뮬레이션의 애플리케이션 간에 통신합니다. 이를 사용하여 멀리서 개체의 상태를 변경하거나, 개체 동작을 변경하거나, 정보를 전체 시뮬레이션으로 브로드캐스트할 수 있습니다.

메시지 수신 확인

전송된 메시지에는 메시지 헤더의 발신자에 대한 정보가 포함됩니다. 이 정보를 사용하여 메시지 수신 시 확인 응답을 다시 보냅니다.

사용자 지정 앱에서 수신한 데이터를 시뮬레이션 내의 다른 앱으로 전달

메시징은 클라이언트가에서 실행되는 사용자 지정 앱에 연결하는 방법을 대체하지 않습니다 SimSpace Weaver. 그러나 메시징을 통해 사용자는 클라이언트 데이터를 수신하는 사용자 지정 앱의 데이터를 외부 연결이 없는 다른 앱으로 전달할 수 있습니다. 메시지 흐름은 반대로 작동하여 외부 연결이 없는 앱이 데이터를 사용자 지정 앱에 전달한 다음 클라이언트에 전달할 수 있습니다.

메시징 APIs 사용

메시징 APIs는 SimSpace Weaver 앱 SDK(최소 버전 1.16.0)에 포함되어 있습니다. 메시징은 C++, Python 및 Unreal Engine 5 및 Unity와의 통합에서 지원됩니다.

메시지 트랜잭션을 처리하는 함수는 `SendMessage` 및 입니다`ReceiveMessages`. 전송된 모든 메시지에는 대상과 페이로드가 포함됩니다. `ReceiveMessages` API는 현재 앱의 인바운드 메시지 대기열에 있는 메시지 목록을 반환합니다.

C++

메시지 전송

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

메시지 수신

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

Python

메시지 전송

```
api.send_message(
 txn, # Transaction
 payload, # api.MessagePayload
 destination, # api.MessageDestination
 api.MessageDeliveryType.BestEffort # api.MessageDeliveryType
)
```

메시지 수신

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

주제

- 메시지 보내기
- 메시지 수신
- 발신자에게 회신

메시지 보내기

메시지는 트랜잭션(다른 Weaver API 호출과 유사), 페이로드 및 대상으로 구성됩니다.

메시지 페이로드

메시지 페이로드는 최대 256바이트의 유연한 데이터 구조입니다. 메시지 페이로드를 생성하는 모범 사례로 다음을 사용하는 것이 좋습니다.

메시지 페이로드를 생성하려면

1. 메시지 내용을 정의하는 데이터 구조(예: `struct` C++의)를 생성합니다.
2. 메시지에 보낼 값이 포함된 메시지 페이로드를 생성합니다.
3. `MessagePayload` 객체를 생성합니다.

메시지 대상

메시지의 대상은 `MessageEndpoint` 객체에 의해 정의됩니다. 여기에는 엔드포인트 유형과 엔드포인트 ID가 모두 포함됩니다. 현재 지원되는 유일한 엔드포인트 유형은 시뮬레이션의 다른 파티션으로 메시지를 보낼 수 `Partition`있는 입니다. 엔드포인트 ID는 대상 대상의 파티션 ID입니다.

메시지에는 대상 주소를 하나만 제공할 수 있습니다. 둘 이상의 파티션에 동시에 메시지를 보내려면 여러 메시지를 생성하고 전송합니다.

위치에서 메시지 엔드포인트를 확인하는 방법에 대한 지침은 섹션을 참조하세요메시징 작업 시 팁.

메시지 전송

대상 및 페이로드 객체를 생성한 후 `SendMessage` API를 사용할 수 있습니다.

C++

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

Python

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

메시지 전송의 전체 예

다음 예제에서는 일반 메시지를 구성하고 전송하는 방법을 보여줍니다. 이 예제에서는 16개의 개별 메시지를 보냅니다. 각 메시지에는 값이 0과 15 사이인 페이로드와 현재 시뮬레이션 틱이 포함되어 있습니다.

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
```

메시지 수신

SimSpace Weaver 는 파티션의 인바운드 메시지 대기열로 메시지를 전송합니다. `ReceiveMessages` API를 사용하여 대기열에서 메시지가 포함된 `MessageList` 객체를 가져옵니다. `ExtractMessage` API로 각 메시지를 처리하여 메시지 데이터를 가져옵니다.

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
```

발신자에게 회신

수신된 모든 메시지에는 메시지의 원래 발신자에 대한 정보가 포함된 메시지 헤더가 포함되어 있습니다. message.header.source_endpoint를 사용하여 회신을 보낼 수 있습니다.

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
```

메시징을 사용해야 하는 경우

의 메시징 SimSpace Weaver 은 시뮬레이션 애플리케이션 간에 정보를 교환하기 위한 또 다른 패턴을 제공합니다. 구독은 시뮬레이션의 특정 애플리케이션 또는 영역에서 데이터를 읽을 수 있는 풀 메커니

즘을 제공하며, 메시지는 시뮬레이션의 특정 애플리케이션 또는 영역으로 데이터를 전송하는 푸시 메커니즘을 제공합니다.

다음은 구독을 통해 데이터를 가져오거나 읽는 대신 메시징을 사용하여 데이터를 푸시하는 것이 더 유용한 두 가지 사용 사례입니다.

Example 1: 다른 앱에 명령을 전송하여 개체 위치 변경

```
// 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);
```

수신 측에서 앱은 개체의 위치를 업데이트하고 상태 패브릭에 기록합니다.

```
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: 공간 앱에 개체 생성 메시지 전송

```
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);
```

수신 측에서 앱은 상태 패브릭에 새 개체를 생성하고 위치를 업데이트합니다.

```
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)));
}
```

메시징 작업 시 팁

위치 또는 앱 이름에서 엔드포인트 확인

`AllPartitions` 함수를 사용하여 메시지 파티션 ID 및 메시지 대상을 결정하는 데 필요한 공간 경계와 도메인 IDs 가져올 수 있습니다. 그러나 메시지를 보낼 위치를 알고 있지만 파티션 ID는 모르는 경우 MessageEndpointResolver 함수를 사용할 수 있습니다.

```
/**
* 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);
};
```

메시지 페이로드 직렬화 및 역직렬화

다음 함수를 사용하여 메시지 페이로드를 생성하고 읽을 수 있습니다. 자세한 내용은 로컬 시스템의 앱 SDK 라이브러리에서 MessagingUtils.h를 참조하세요.

```
/**
* 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());
}
```

메시징 오류 및 문제 해결

메시징 APIs.

엔드포인트 해결 오류

이러한 오류는 앱이 메시지를 보내기 전에 발생할 수 있습니다.

도메인 이름 확인

잘못된 엔드포인트에 메시지를 보내면 다음 오류가 발생합니다.

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

이는 사용자 지정 앱에 메시지를 보내려고 하는데 해당 사용자 지정 앱이 아직 시뮬레이션에 조인하지 않은 경우에 발생할 수 있습니다. `DescribeSimulation` API를 사용하여 메시지를 보내기 전에 사용자 지정 앱이 시작되었는지 확인합니다. 이 동작은 SimSpace Weaver Local 및에서 동일합니다 AWS 클라우드.

위치 확인

도메인 이름이 유효하지만 위치가 잘못된 엔드포인트를 확인하려고 하면 다음 오류가 발생합니다.

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

SimSpace Weaver 앱 SDK`MessageEndpointResolver`에 포함된 `MessageUtils` 라이브러리에서를 사용하는 것이 좋습니다.

메시지 전송 오류

앱이 메시지를 전송할 때 다음과 같은 오류가 발생할 수 있습니다.

앱당, 틱당, 초과된 메시지 전송 한도

시뮬레이션 틱당 앱당 전송할 수 있는 메시지 수의 현재 제한은 128개입니다. 동일한 틱에 대한 후속 호출은 다음 오류와 함께 실패합니다.

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

SimSpace Weaver 는 다음 틱에서 전송되지 않은 메시지를 보내려고 시도합니다. 이 문제를 해결하려면 전송 빈도를 낮춥니다. 256바이트 제한보다 작은 메시지 페이로드를 결합하여 아웃바운드 메시지 수를 줄입니다.

이 동작은 SimSpace Weaver Local 및에서 동일합니다 AWS 클라우드.

메시지 페이로드 크기 제한을 초과했습니다.

메시지 페이로드 크기의 현재 제한은 및 모두에서 256바이트SimSpace Weaver Local입니다 AWS 클라우드. 페이로드가 256바이트보다 큰 메시지를 전송하면 다음 오류가 발생합니다.

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

SimSpace Weaver 는 각 메시지를 확인하고 한도를 초과하는 메시지만 거부합니다. 예를 들어 앱이 10개의 메시지를 보내려고 하는데 1개가 검사에 실패하면 해당 메시지 1개만 거부됩니다.는 다른 9개의 메시지를 SimSpace Weaver 보냅니다.

이 동작은 SimSpace Weaver Local 및에서 동일합니다 AWS 클라우드.

대상이 소스와 동일함

앱은 자신이 소유한 파티션에 메시지를 보낼 수 없습니다. 앱이 소유한 파티션에 메시지를 보내는 경우 다음 오류가 발생합니다.

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

이 동작은 SimSpace Weaver Local 및에서 동일합니다 AWS 클라우드.

최선의 메시지 전송

SimSpace Weaver 는 메시지 전송을 보장하지 않습니다. 서비스는 후속 시뮬레이션 틱에서 메시지 전송을 완료하려고 시도하지만 메시지가 손실되거나 지연될 수 있습니다.

작업 시 모범 사례 SimSpace Weaver

작업 시 다음 모범 사례를 따르는 것이 좋습니다 SimSpace Weaver.

주제

결제 경보 설정

에서 리소스를 쉽게 프로비저닝 AWS 하고 더 이상 필요하지 않은 경우에도 항상 실행 상태로 둘 수 있습니다. 이로 인해 엄청난 비용이 발생하여 청구서를 받고 놀랄 수 있습니다. Amazon CloudWatch에서 경보를 구성하여 비용이 설정한 임계값을 초과할 때 트리거하고 알림을 받을 수 있습니다. 비용 관리 도구를 사용하여 비용을 검토할 수 있습니다. 자세한 내용은 다음을 참조하세요.

- 예상 AWS 요금을 모니터링하기 위한 결제 경보 생성
- 정의 AWS Cost Management

SimSpace Weaver Local 사용

에서 시뮬레이션SimSpace Weaver Local을 SimSpace Weaver 서비스에 업로드하기 전에를 사용하여 시뮬레이션을 개발하고 테스트하는 것이 좋습니다 AWS 클라우드. SimSpace Weaver Local을 사용하여 개발하면 다음과 같은 이점이 있습니다.

- 대규모 업로드를 기다릴 필요가 없음
- 생성할 수 있는 로컬 시뮬레이션 수에 제한이 없음
- 로컬 컴퓨터의 계산 시간에 대한 요금이 부과되지 않음
- 앱의 콘솔 출력에 직접 액세스
- 에서 다시 생성할 필요 없이 로컬 시뮬레이션을 수정, 재구축 및 다시 시작 AWS 클라우드

필요 없는 시뮬레이션 중지

시뮬레이션을 실행하는 동안에는 시뮬레이션에 대한 요금이 청구됩니다. 요금 부과를 중지하려면 시뮬레이션을 중지해야 합니다. 실행 중인 시뮬레이션은 최대 시뮬레이션 횟수 할당량에도 포함됩니다. 또한 로깅이 구성된 시뮬레이션을 실행하면 대량의 로그가 생성될 수 있으며 이에 대한 청구 비용도 부과됩니다. 추가 요금이 부과되지 않도록 하려면 필요하지 않은 시뮬레이션은 모두 중지해야 합니다.

> **Important**
>
> 시뮬레이션 클럭을 중지해도 시뮬레이션이 중지되는 것은 아니며, 클럭은 앱에 틱을 게시하는 작업만 중지합니다. 시뮬레이션을 중지한 후에는 다시 시작할 수 없습니다.

필요 없는 리소스 삭제

에서 생성하는 각 시뮬레이션은 다른 AWS 서비스에 리소스 SimSpace Weaver 도 생성합니다. 이러한 다른 서비스의 리소스 및 데이터에 대한 요금이 부과될 수 있습니다. 실행 중 및 실패한 시뮬레이션은 최대 시뮬레이션 횟수 할당량에도 포함됩니다. 새 시뮬레이션을 시작하려면 불필요한 실패 시뮬레이션을 삭제해야 합니다. 시뮬레이션을 삭제하면 다른 AWS 서비스에 있는 시뮬레이션의 리소스가 삭제되지 않을 수 있습니다. 예를 들어 Amazon CloudWatch Logs의 모든 시뮬레이션 로그 데이터는 사용자가 삭제할 때까지 그대로 유지됩니다. 해당 로그 데이터에 대한 요금이 부과됩니다. 시뮬레이션과 관련된 리소스가 더 이상 필요 없는 경우 시뮬레이션을 위한 모든 관련 리소스를 정리해야 합니다.

백업하기

모든 것을 백업하고 백업 계획을 세우는 것이 좋습니다. 데이터가에 AWS 있다고 해서 백업할 필요가 없다고 가정해서는 안 됩니다. 시뮬레이션 상태를 백업해야 하는 경우 자체 시스템을 만들어야 합니다. 필요한 AWS 리전 경우 여러을 사용하고 프로덕션 워크로드를 다른 워크로드로 빠르게 전환할 수 있는 계획을 세우 AWS 리전 는 것이 좋습니다. AWS 리전 해당 지원에 대한 자세한 내용은 섹션을 SimSpace Weaver참조하세요SimSpace Weaver 엔드포인트 및 할당량.

의 보안 AWS SimSpace Weaver

의 클라우드 보안 AWS 이 최우선 순위입니다. AWS 고객은 보안에 가장 민감한 조직의 요구 사항을 충족하도록 구축된 데이터 센터 및 네트워크 아키텍처의 이점을 누릴 수 있습니다.

보안은 AWS 와 사용자 간의 공동 책임입니다. 공동 책임 모델은 이 사항을 클라우드의 보안 및 클라우드 내 보안으로 설명합니다.

- 클라우드 보안 - AWS 는에서 AWS 서비스를 실행하는 인프라를 보호할 책임이 있습니다 AWS 클라우드. AWS 또한는 안전하게 사용할 수 있는 서비스를 제공합니다. 타사 감사자는 AWS 규정 준수 프로그램 일환으로 보안의 효과를 정기적으로 테스트하고 확인합니다. 에 적용되는 규정 준수 프로그램에 대한 자세한 내용은 규정 준수 프로그램 AWS 제공 범위 내 서비스규정 준수 프로그램 제공 범위 내 서비스를 AWS SimSpace Weaver참조하세요.
- 클라우드의 보안 - 사용자의 책임은 사용하는 AWS 서비스에 따라 결정됩니다. 또한 귀하는 귀사의 데이터 민감도, 귀사의 요구 사항, 관련 법률 및 규정을 비롯한 기타 요소에 대해서도 책임이 있습니다.

이 설명서는를 사용할 때 공동 책임 모델을 적용하는 방법을 이해하는 데 도움이 됩니다 SimSpace Weaver. 다음 주제에서는 보안 및 규정 준수 목표를 충족하도록 SimSpace Weaver 를 구성하는 방법을 보여줍니다. 또한 SimSpace Weaver 리소스를 모니터링하고 보호하는 데 도움이 되는 다른 AWS 서비스를 사용하는 방법을 알아봅니다.

주제

의 데이터 보호 AWS SimSpace Weaver

AWS 공동 책임 모델의 데이터 보호에 적용됩니다 AWS SimSpace Weaver. 이 모델에 설명된 대로 AWS 는 모든를 실행하는 글로벌 인프라를 보호할 책임이 있습니다 AWS 클라우드. 사용자는 이 인프라에 호스팅되는 콘텐츠에 대한 통제 권한을 유지할 책임이 있습니다. 사용하는 AWS 서비스 의 보안 구성과 관리 태스크에 대한 책임도 사용자에게 있습니다. 데이터 프라이버시에 관한 자세한 내용은 데이터 프라이버시 FAQ를 참조하세요. 유럽의 데이터 보호에 대한 자세한 내용은 AWS 보안 블로그의 AWS 공동 책임 모델 및 GDPR 블로그 게시물을 참조하세요.

데이터 보호를 위해 자격 증명을 보호하고 AWS 계정 AWS IAM Identity Center 또는 AWS Identity and Access Management (IAM)를 사용하여 개별 사용자를 설정하는 것이 좋습니다. 이렇게 하면 개별 사용자에게 자신의 직무를 충실히 이행하는 데 필요한 권한만 부여됩니다. 또한 다음과 같은 방법으로 데이터를 보호하는 것이 좋습니다.

- 각 계정에 다중 인증(MFA)을 사용합니다.
- SSL/TLS를 사용하여 AWS 리소스와 통신합니다. TLS 1.2는 필수이며 TLS 1.3을 권장합니다.
- 를 사용하여 API 및 사용자 활동 로깅을 설정합니다 AWS CloudTrail. CloudTrail 추적을 사용하여 AWS 활동을 캡처하는 방법에 대한 자세한 내용은 AWS CloudTrail 사용 설명서의 CloudTrail 추적 작업을 참조하세요.
- 내부의 모든 기본 보안 제어와 함께 AWS 암호화 솔루션을 사용합니다 AWS 서비스.
- Amazon S3에 저장된 민감한 데이터를 검색하고 보호하는 데 도움이 되는 Amazon Macie와 같은 고급 관리형 보안 서비스를 사용합니다.
- 명령줄 인터페이스 또는 API를 AWS 통해에 액세스할 때 FIPS 140-3 검증 암호화 모듈이 필요한 경우 FIPS 엔드포인트를 사용합니다. 사용 가능한 FIPS 엔드포인트에 대한 자세한 내용은 연방 정보 처리 표준(FIPS) 140-3을 참조하세요.

고객의 이메일 주소와 같은 기밀 정보나 중요한 정보는 태그나 이름 필드와 같은 자유 형식 텍스트 필드에 입력하지 않는 것이 좋습니다. 여기에는 SimSpace Weaver 또는 기타 AWS 서비스 에서 콘솔, API AWS CLI또는 AWS SDKs를 사용하여 작업하는 경우가 포함됩니다. 이름에 사용되는 태그 또는 자유 형식 텍스트 필드에 입력하는 모든 데이터는 청구 또는 진단 로그에 사용될 수 있습니다. 외부 서버에 URL을 제공할 때 해당 서버에 대한 요청을 검증하기 위해 보안 인증 정보를 URL에 포함시켜서는 안 됩니다.

저장된 데이터 암호화

데이터가 디스크와 같은 비휘발성(영구) 데이터 스토리지에 있는 경우 데이터가 저장된 것으로 간주됩니다. 메모리, 레지스터 등 휘발성 데이터 스토리지에 있는 데이터는 저장 상태로 간주되지 않습니다.

SimSpace Weaver를 사용할 때 유일하게 저장된 데이터는 다음과 같습니다.

- Amazon Simple Storage Service(Amazon S3)에 업로드하는 앱 및 스키마
- Amazon CloudWatch에 저장된 시뮬레이션 로그 데이터

에서 내부적으로 SimSpace Weaver 사용하는 다른 데이터는 시뮬레이션을 중지한 후에도 지속되지 않습니다.

저장 데이터를 암호화하는 방법을 알아보려면 다음을 참조하세요.

- Amazon S3에서 데이터 암호화
- 로그 데이터 암호화

전송 중 암호화

AWS Command Line Interface (AWS CLI), AWS SDK 및 SimSpace Weaver 앱 SDK를 통한 SimSpace Weaver API 연결은 서명 버전 4 서명 프로세스와 함께 TLS 암호화를 사용합니다.는 연결에 사용하는 보안 자격 증명에 대한 IAM 정의 액세스 정책을 사용하여 인증을 AWS 관리합니다.

내부적으로 TLS를 SimSpace Weaver 사용하여 사용하는 다른 AWS 서비스에 연결합니다.

> **Important**
>
> 앱과 클라이언트 간의 통신에는가 포함되지 않습니다 SimSpace Weaver. 필요한 경우 시뮬레이션 클라이언트와의 통신을 암호화하는 것은 사용자의 책임입니다. 클라이언트 연결 간에 전송되는 모든 데이터를 암호화하는 솔루션을 만드는 것이 좋습니다.

암호화 솔루션을 지원할 수 있는 AWS 서비스에 대한 자세한 내용은 AWS 보안 블로그를 참조하세요.

인터네트워크 트래픽 개인 정보 보호

SimSpace Weaver 컴퓨팅 리소스는 모든 SimSpace Weaver 고객이 공유하는 1개의 Amazon VPC 내에 있습니다. 모든 내부 SimSpace Weaver 서비스 트래픽은 AWS 네트워크 내에 유지되며 인터넷을 통과하지 않습니다. 시뮬레이션 클라이언트와 앱 간의 통신은 인터넷을 통해 이루어집니다.

에 대한 자격 증명 및 액세스 관리 AWS SimSpace Weaver

AWS Identity and Access Management (IAM)는 관리자가 AWS 리소스에 대한 액세스를 안전하게 제어하는 데 도움이 AWS 서비스 되는 입니다. IAM 관리자는 누가 SimSpace Weaver 리소스를 사용할 수 있는 인증(로그인) 및 권한(권한 있음)을 받을 수 있는지 제어합니다. IAM은 추가 비용 없이 사용할 수 AWS 서비스 있는 입니다.

주제

대상

AWS Identity and Access Management (IAM)를 사용하는 방법은 역할에 따라 다릅니다.

- 서비스 사용자 - 기능에 액세스할 수 없는 경우 관리자에게 권한 요청(참조AWS SimSpace Weaver 자격 증명 및 액세스 문제 해결)
- 서비스 관리자 - 사용자 액세스 결정 및 권한 요청 제출(AWS SimSpace Weaver 에서 IAM을 사용하는 방법 참조)
- IAM 관리자 - 액세스를 관리하기 위한 정책 작성(에 대한 자격 증명 기반 정책 예제 AWS SimSpace Weaver 참조)

ID를 통한 인증

인증은 AWS 자격 증명으로에 로그인하는 방법입니다. AWS 계정 루트 사용자, IAM 사용자 또는 IAM 역할을 수임하여 인증되어야 합니다.

AWS IAM Identity Center (IAM Identity Center), Single Sign-On 인증 또는 Google/Facebook 자격 증명과 같은 자격 증명 소스의 자격 증명을 사용하여 페더레이션 자격 증명으로 로그인할 수 있습니다. 로그인하는 방법에 대한 자세한 내용은 AWS 로그인 사용 설명서의 AWS 계정에 로그인하는 방법 섹션을 참조하세요.

프로그래밍 방식 액세스를 위해는 요청에 암호화 방식으로 서명할 수 있는 SDK 및 CLI를 AWS 제공합니다. 자세한 내용은 IAM 사용 설명서의 API 요청용AWS Signature Version 4 섹션을 참조하세요.

AWS 계정 루트 사용자

를 생성할 때 모든 AWS 서비스 및 리소스에 대한 완전한 액세스 권한이 있는 AWS 계정 theroot 사용자라는 하나의 로그인 자격 증명으로 AWS 계정시작합니다. 일상적인 태스크에 루트 사용자를 사용하지 않을 것을 강력히 권장합니다. 루트 사용자 자격 증명이 필요한 작업은 IAM 사용 설명서의 루트 사용자 자격 증명이 필요한 작업 섹션을 참조하세요.

페더레이션 ID

가장 좋은 방법은 인간 사용자에게 자격 증명 공급자와의 페더레이션을 사용하여 임시 자격 증명을 AWS 서비스 사용하여에 액세스하도록 요구하는 것입니다.

페더레이션 자격 증명은 엔터프라이즈 디렉터리, 웹 자격 증명 공급자 또는 자격 증명 소스의 자격 증명을 AWS 서비스 사용하여 Directory Service 에 액세스하는 사용자입니다. 페더레이션 ID는 임시 자격 증명을 제공하는 역할을 수임합니다.

중앙 집중식 액세스 관리를 위해 AWS IAM Identity Center를 추천합니다. 자세한 정보는 AWS IAM Identity Center 사용 설명서의 What is IAM Identity Center?를 참조하세요.

IAM 사용자 및 그룹

IAM 사용자는 단일 개인 또는 애플리케이션에 대한 특정 권한을 가진 ID입니다. 장기 자격 증명이 있는 IAM 사용자 대신 임시 자격 증명을 사용하는 것이 좋습니다. 자세한 내용은 IAM 사용 설명서의 자격 증명 공급자와의 페더레이션을 사용하여 임시 자격 증명을 AWS 사용하여에 액세스하도록 인간 사용자에게 요구하기를 참조하세요.

IAM 그룹은 IAM 사용자 모음을 지정하고 대규모 사용자 집합에 대한 관리 권한을 더 쉽게 만듭니다. 자세한 내용은 IAM 사용 설명서의 IAM 사용자 사용 사례 섹션을 참조하세요.

IAM 역할

IAM 역할은 임시 자격 증명을 제공하는 특정 권한이 있는 자격 증명입니다. 사용자에서 IAM 역할(콘솔)로 전환하거나 또는 API 작업을 호출하여 역할을 수임할 수 있습니다. AWS CLI AWS 자세한 내용은 IAM 사용 설명서의 역할 수임 방법을 참조하세요.

IAM 역할은 페더레이션 사용자 액세스, 임시 IAM 사용자 권한, 교차 계정 액세스, 교차 서비스 액세스 및 Amazon EC2에서 실행되는 애플리케이션에 유용합니다. 자세한 내용은 IAM 사용 설명서의 교차 계정 리소스 액세스를 참조하세요.

정책을 사용하여 액세스 관리

정책을 AWS 생성하고 자격 증명 또는 리소스에 연결하여 AWS 에서 액세스를 제어합니다. 정책은 자격 증명 또는 리소스와 연결될 때 권한을 정의합니다.는 보안 주체가 요청할 때 이러한 정책을 AWS 평가합니다. 대부분의 정책은에 JSON 문서 AWS 로 저장됩니다. JSON 정책 문서에 대한 자세한 내용은 IAM 사용 설명서의 JSON 정책 개요 섹션을 참조하세요.

정책을 사용하여 관리자는 어떤 보안 주체가 어떤 리소스에 대해 어떤 조건에서 작업을 수행할 수 있는지 정의하여 누가 무엇을 액세스할 수 있는지 지정합니다.

기본적으로 사용자 및 역할에는 어떠한 권한도 없습니다. IAM 관리자는 IAM 정책을 생성하고 사용자가 수임할 수 있는 역할에 추가합니다. IAM 정책은 작업을 수행하기 위해 사용하는 방법과 관계없이 작업에 대한 권한을 정의합니다.

ID 기반 정책

ID 기반 정책은 ID(사용자, 사용자 그룹 또는 역할)에 연결하는 JSON 권한 정책 문서입니다. 이러한 정책은 자격 증명이 수행할 수 있는 작업, 대상 리소스 및 이에 관한 조건을 제어합니다. ID 기반 정책을 생성하는 방법을 알아보려면 IAM 사용 설명서에서 고객 관리형 정책으로 사용자 지정 IAM 권한 정의를 참조하세요.

ID 기반 정책은 인라인 정책(단일 ID에 직접 포함) 또는 관리형 정책(여러 ID에 연결된 독립 실행형 정책)일 수 있습니다. 관리형 정책 또는 인라인 정책을 선택하는 방법을 알아보려면 IAM 사용 설명서의 관리형 정책 및 인라인 정책 중에서 선택 섹션을 참조하세요.

리소스 기반 정책

리소스 기반 정책은 리소스에 연결하는 JSON 정책 설명서입니다. 예를 들어 IAM 역할 신뢰 정책 및 Amazon S3 버킷 정책이 있습니다. 리소스 기반 정책을 지원하는 서비스에서 서비스 관리자는 이러한

정책을 사용하여 특정 리소스에 대한 액세스를 통제할 수 있습니다. 리소스 기반 정책에서 보안 주체를 지정해야 합니다.

리소스 기반 정책은 해당 서비스에 있는 인라인 정책입니다. 리소스 기반 정책에서는 IAM의 AWS 관리형 정책을 사용할 수 없습니다.

기타 정책 유형

AWS 는 보다 일반적인 정책 유형에서 부여한 최대 권한을 설정할 수 있는 추가 정책 유형을 지원합니다.

- 권한 경계 - ID 기반 정책에서 IAM 엔터티에 부여할 수 있는 최대 권한을 설정합니다. 자세한 정보는 IAM 사용 설명서의 IAM 엔터티의 권한 범위를 참조하세요.
- 서비스 제어 정책(SCP) - AWS Organizations내 조직 또는 조직 단위에 대한 최대 권한을 지정합니다. 자세한 내용은AWS Organizations 사용 설명서의 서비스 제어 정책을 참조하세요.
- 리소스 제어 정책(RCP) – 계정의 리소스에 사용할 수 있는 최대 권한을 설정합니다. 자세한 내용은 AWS Organizations 사용 설명서의 리소스 제어 정책(RCP)을 참조하세요.
- 세션 정책 – 역할 또는 페더레이션 사용자에 대해 임시 세션을 프로그래밍 방식으로 생성할 때 파라미터로 전달하는 고급 정책입니다. 자세한 내용은 IAM 사용 설명서의 세션 정책을 참조하세요.

여러 정책 유형

여러 정책 유형이 요청에 적용되는 경우, 결과 권한은 이해하기가 더 복잡합니다. 에서 여러 정책 유형이 관련될 때 요청을 허용할지 여부를 AWS 결정하는 방법을 알아보려면 IAM 사용 설명서의 정책 평가 로직을 참조하세요.

AWS SimSpace Weaver 에서 IAM을 사용하는 방법

IAM을 사용하여에 대한 액세스를 관리하기 전에 사용할 수 있는 IAM 기능을 SimSpace Weaver알아봅니다 SimSpace Weaver.

에서 사용할 수 있는 IAM 기능 AWS SimSpace Weaver

IAM 특성	SimSpace Weaver 지원
자격 증명 기반 정책	예
리소스 기반 정책	아니요

IAM 특성	SimSpace Weaver 지원
정책 작업	예
정책 리소스	예
정책 조건 키(서비스별)	예
ACL	아니요
ABAC(정책의 태그)	예
임시 보안 인증	예
엔터티 권한	예
서비스 역할	예
서비스 연결 역할	아니요

SimSpace Weaver 및 기타 AWS 서비스가 대부분의 IAM 기능과 작동하는 방법을 개괄적으로 알아보려면 IAM 사용 설명서의 AWS IAM으로 작업하는 서비스를 참조하세요.

에 대한 자격 증명 기반 정책 SimSpace Weaver

ID 기반 정책 지원: 예

ID 기반 정책은 IAM 사용자, 사용자 그룹 또는 역할과 같은 ID에 연결할 수 있는 JSON 권한 정책 문서입니다. 이러한 정책은 사용자 및 역할이 어떤 리소스와 어떤 조건에서 어떤 작업을 수행할 수 있는지를 제어합니다. 자격 증명 기반 정책을 생성하는 방법을 알아보려면 IAM 사용 설명서에서 고객 관리형 정책으로 사용자 지정 IAM 권한 정의를 참조하세요.

IAM ID 기반 정책을 사용하면 허용되거나 거부되는 작업과 리소스뿐 아니라 작업이 허용되거나 거부되는 조건을 지정할 수 있습니다. JSON 정책에서 사용할 수 있는 모든 요소에 대해 알아보려면 IAM 사용 설명서의 IAM JSON 정책 요소 참조를 참조하세요.

에 대한 자격 증명 기반 정책 예제 SimSpace Weaver

자격 SimSpace Weaver 증명 기반 정책의 예를 보려면 섹션을 참조하세요에 대한 자격 증명 기반 정책 예제 AWS SimSpace Weaver.

내의 리소스 기반 정책 SimSpace Weaver

리소스 기반 정책 지원: 아니요

리소스 기반 정책은 리소스에 연결하는 JSON 정책 설명서입니다. 리소스 기반 정책의 예제는 IAM 역할 신뢰 정책과 Amazon S3 버킷 정책입니다. 리소스 기반 정책을 지원하는 서비스에서 서비스 관리자는 이러한 정책을 사용하여 특정 리소스에 대한 액세스를 통제할 수 있습니다. 정책이 연결된 리소스의 경우 정책은 지정된 보안 주체가 해당 리소스와 어떤 조건에서 어떤 작업을 수행할 수 있는지를 정의합니다. 리소스 기반 정책에서 보안 주체를 지정해야 합니다. 보안 주체에는 계정, 사용자, 역할, 페더레이션 사용자 또는이 포함될 수 있습니다 AWS 서비스.

교차 계정 액세스를 활성화하려는 경우, 전체 계정이나 다른 계정의 IAM 개체를 리소스 기반 정책의 보안 주체로 지정할 수 있습니다. 자세한 내용은 IAM 사용 설명서의 IAM에서 교차 계정 리소스 액세스를 참조하세요.

에 대한 정책 작업 SimSpace Weaver

정책 작업 지원: 예

관리자는 AWS JSON 정책을 사용하여 누가 무엇에 액세스할 수 있는지 지정할 수 있습니다. 즉, 어떤 보안 주체가 어떤 리소스와 어떤 조건에서 작업을 수행할 수 있는지를 지정할 수 있습니다.

JSON 정책의 `Action`요소는 정책에서 액세스를 허용하거나 거부하는 데 사용할 수 있는 작업을 설명합니다. 연결된 작업을 수행할 수 있는 권한을 부여하기 위한 정책에 작업을 포함하세요.

SimSpace Weaver 작업 목록을 보려면 서비스 승인 참조의에서 정의한 작업을 AWS SimSpace Weaver 참조하세요.

의 정책 작업은 작업 앞에 다음 접두사를 SimSpace Weaver 사용합니다.

```
simspaceweaver
```

단일 문에서 여러 작업을 지정하려면 쉼표로 구분합니다.

```
"Action": [
      "simspaceweaver:action1",
      "simspaceweaver:action2"
         ]
```

자격 SimSpace Weaver 증명 기반 정책의 예를 보려면 섹션을 참조하세요에 대한 자격 증명 기반 정책 예제 AWS SimSpace Weaver.

에 대한 정책 리소스 SimSpace Weaver

정책 리소스 지원: 예

관리자는 AWS JSON 정책을 사용하여 누가 무엇에 액세스할 수 있는지 지정할 수 있습니다. 즉, 어떤 보안 주체가 어떤 리소스와 어떤 조건에서 작업을 수행할 수 있는지를 지정할 수 있습니다.

`Resource` JSON 정책 요소는 작업이 적용되는 하나 이상의 객체를 지정합니다. 모범 사례에 따라 Amazon 리소스 이름(ARN)을 사용하여 리소스를 지정합니다. 리소스 수준 권한을 지원하지 않는 작업의 경우, 와일드카드(*)를 사용하여 해당 문이 모든 리소스에 적용됨을 나타냅니다.

```
"Resource": "*"
```

SimSpace Weaver 리소스 유형 및 해당 ARNs 목록을 보려면 서비스 승인 참조의에서 정의한 리소스를 AWS SimSpace Weaver 참조하세요. 각 리소스의 ARN을 지정할 수 있는 작업을 알아보려면 AWS SimSpace Weaver에서 정의한 작업을 참조하세요.

자격 SimSpace Weaver 증명 기반 정책의 예를 보려면 섹션을 참조하세요에 대한 자격 증명 기반 정책 예제 AWS SimSpace Weaver.

에 대한 정책 조건 키 SimSpace Weaver

서비스별 정책 조건 키 지원: 예

관리자는 AWS JSON 정책을 사용하여 누가 무엇에 액세스할 수 있는지 지정할 수 있습니다. 즉, 어떤 보안 주체가 어떤 리소스와 어떤 조건에서 작업을 수행할 수 있는지를 지정할 수 있습니다.

`Condition` 요소는 정의된 기준에 따라 문이 실행되는 시기를 지정합니다. 같음(equals) 또는 미만(less than)과 같은 조건 연산자를 사용하여 정책의 조건을 요청의 값과 일치시키는 조건식을 생성할 수 있습니다. 모든 AWS 전역 조건 키를 보려면 IAM 사용 설명서의 AWS 전역 조건 컨텍스트 키를 참조하세요.

SimSpace Weaver 조건 키 목록을 보려면 서비스 승인 참조의에 대한 조건 키를 AWS SimSpace Weaver 참조하세요. 조건 키를 사용할 수 있는 작업과 리소스를 알아보려면 에서 정의한 작업을 AWS SimSpace Weaver 참조하세요.

자격 SimSpace Weaver 증명 기반 정책의 예를 보려면 섹션을 참조하세요에 대한 자격 증명 기반 정책 예제 AWS SimSpace Weaver.

의 액세스 제어 목록(ACLs) SimSpace Weaver

ACL 지원: 아니요

액세스 제어 목록(ACL)은 어떤 보안 주체(계정 멤버, 사용자 또는 역할)가 리소스에 액세스할 수 있는 권한을 가지고 있는지를 제어합니다. ACL은 JSON 정책 문서 형식을 사용하지 않지만 리소스 기반 정책과 유사합니다.

를 사용한 ABAC(속성 기반 액세스 제어) SimSpace Weaver

ABAC 지원(정책의 태그): 예

속성 기반 액세스 제어(ABAC)는 태그라고 불리는 속성을 기반으로 권한을 정의하는 권한 부여 전략입니다. IAM 엔터티 및 AWS 리소스에 태그를 연결한 다음 보안 주체의 태그가 리소스의 태그와 일치할 때 작업을 허용하는 ABAC 정책을 설계할 수 있습니다.

태그에 근거하여 액세스를 제어하려면 `aws:ResourceTag/key-name`, `aws:RequestTag/key-name`또는 `aws:TagKeys`조건 키를 사용하여 정책의 조건 요소에 태그 정보를 제공합니다.

서비스가 모든 리소스 유형에 대해 세 가지 조건 키를 모두 지원하는 경우, 값은 서비스에 대해 예입니다. 서비스가 일부 리소스 유형에 대해서만 세 가지 조건 키를 모두 지원하는 경우, 값은 부분적입니다.

ABAC에 대한 자세한 내용은 IAM 사용 설명서의 ABAC 권한 부여를 통한 권한 정의를 참조하세요. ABAC 설정 단계가 포함된 자습서를 보려면 IAM 사용 설명서의 속성 기반 액세스 제어(ABAC) 사용을 참조하세요.

에서 임시 자격 증명 사용 SimSpace Weaver

임시 자격 증명 지원: 예

임시 자격 증명은 AWS 리소스에 대한 단기 액세스를 제공하며 페더레이션 또는 전환 역할을 사용할 때 자동으로 생성됩니다. 장기 액세스 키를 사용하는 대신 임시 자격 증명을 동적으로 생성하는 것이 AWS 좋습니다. 자세한 내용은 IAM 사용 설명서의 IAM의 임시 보안 자격 증명 및 IAM으로 작업하는 AWS 서비스 섹션을 참조하세요.

에 대한 교차 서비스 보안 주체 권한 SimSpace Weaver

전달 액세스 세션(FAS) 지원: 예

전달 액세스 세션(FAS)은를 호출하는 보안 주체의 권한을 다운스트림 서비스에 AWS 서비스 대한 요청과 AWS 서비스함께 사용합니다. FAS 요청 시 정책 세부 정보는 전달 액세스 세션을 참조하세요.

에 대한 서비스 역할 SimSpace Weaver

서비스 역할 지원: 예

서비스 역할은 서비스가 사용자를 대신하여 작업을 수행하는 것으로 가정하는 IAM 역할입니다. IAM 관리자는 IAM 내에서 서비스 역할을 생성, 수정 및 삭제할 수 있습니다. 자세한 내용은 IAM 사용 설명서의 AWS 서비스 AWS에 권한을 위임할 역할 생성을 참조하세요.

> **Warning**
>
> 서비스 역할에 대한 권한을 변경하면 SimSpace Weaver 기능이 중단될 수 있습니다. 에서 관련 지침을 SimSpace Weaver 제공하는 경우에만 서비스 역할을 편집합니다.

SimSpace Weaver 앱 SDK 스크립트는 CloudFormation 템플릿을 사용하여 시뮬레이션을 지원하는 다른 AWS 서비스에 리소스를 생성합니다. 이러한 리소스 중 하나는 시뮬레이션을 위한 앱 역할입니다.는 CloudWatch Logs에 로그 데이터를 쓰는 등 사용자를 AWS 계정 대신하여에서 작업을 수행하는 앱 역할을 수 SimSpace Weaver 임합니다. 앱 역할에 대한 자세한 내용은 에서 자동으로 SimSpace Weaver 생성하는 권한 섹션을 참조하세요.

에 대한 서비스 연결 역할 SimSpace Weaver

서비스 연결 역할 지원: 아니요

서비스 연결 역할은에 연결된 서비스 역할의 한 유형입니다 AWS 서비스. 서비스는 사용자를 대신하여 작업을 수행하기 위해 역할을 수임할 수 있습니다. 서비스 연결 역할은에 표시 AWS 계정 되며 서비스가 소유합니다. IAM 관리자는 서비스 연결 역할의 권한을 볼 수 있지만 편집은 할 수 없습니다.

서비스 연결 역할 생성 또는 관리에 대한 자세한 내용은 IAM으로 작업하는AWS 서비스를 참조하세요. 서비스 연결 역할 열에서 Yes가 포함된 서비스를 테이블에서 찾습니다. 해당 서비스에 대한 서비스 연결 역할 설명서를 보려면 예(Yes) 링크를 선택합니다.

에 대한 자격 증명 기반 정책 예제 AWS SimSpace Weaver

기본적으로 사용자 및 역할에는 SimSpace Weaver 리소스를 생성하거나 수정할 수 있는 권한이 없습니다. 사용자에게 사용자가 필요한 리소스에서 작업을 수행할 권한을 부여하려면 IAM 관리자가 IAM 정책을 생성하면 됩니다.

이러한 예제 JSON 정책 문서를 사용하여 IAM ID 기반 정책을 생성하는 방법을 알아보려면 IAM 사용 설명서의 IAM 정책 생성(콘솔)을 참조하세요.

각 리소스 유형에 대한 ARNs 형식을 포함하여 SimSpace Weaver에서 정의한 작업 및 리소스 유형에 대한 자세한 내용은 서비스 승인 참조의 에 대한 작업, 리소스 및 조건 키를 AWS SimSpace Weaver 참조하세요.

주제

- 정책 모범 사례
- SimSpace Weaver 콘솔 사용
- 사용자가 자신의 고유한 권한을 볼 수 있도록 허용
- 사용자가 시뮬레이션을 생성하고 실행하도록 허용

정책 모범 사례

자격 증명 기반 정책에 따라 계정에서 사용자가 SimSpace Weaver 리소스를 생성, 액세스 또는 삭제할 수 있는지 여부가 결정됩니다. 이 작업으로 인해 AWS 계정에 비용이 발생할 수 있습니다. ID 기반 정책을 생성하거나 편집할 때는 다음 지침과 권장 사항을 따르세요.

- AWS 관리형 정책을 시작하고 최소 권한으로 전환 - 사용자 및 워크로드에 권한 부여를 시작하려면 많은 일반적인 사용 사례에 대한 권한을 부여하는 AWS 관리형 정책을 사용합니다. 에서 사용할 수 있습니다 AWS 계정. 사용 사례에 맞는 AWS 고객 관리형 정책을 정의하여 권한을 추가로 줄이는 것이 좋습니다. 자세한 내용은 IAM 사용 설명서의 AWS 관리형 정책 또는 AWS 직무에 대한 관리형 정책을 참조하세요.
- 최소 권한 적용 – IAM 정책을 사용하여 권한을 설정하는 경우, 작업을 수행하는 데 필요한 권한만 부여합니다. 이렇게 하려면 최소 권한으로 알려진 특정 조건에서 특정 리소스에 대해 수행할 수 있는 작업을 정의합니다. IAM을 사용하여 권한을 적용하는 방법에 대한 자세한 정보는 IAM 사용 설명서에 있는 IAM의 정책 및 권한을 참조하세요.
- IAM 정책의 조건을 사용하여 액세스 추가 제한 – 정책에 조건을 추가하여 작업 및 리소스에 대한 액세스를 제한할 수 있습니다. 예를 들어, SSL을 사용하여 모든 요청을 전송해야 한다고 지정하는 정책 조건을 작성할 수 있습니다. AWS 서비스와 같은 특정를 통해 사용되는 경우 조건을 사용하여 서비스 작업에 대한 액세스 권한을 부여할 수도 있습니다 CloudFormation. 자세한 내용은 IAM 사용 설명서의 IAM JSON 정책 요소: 조건을 참조하세요.
- IAM Access Analyzer를 통해 IAM 정책을 확인하여 안전하고 기능적인 권한 보장 - IAM Access Analyzer에서는 IAM 정책 언어(JSON)와 모범 사례가 정책에서 준수되도록 새로운 및 기존 정책을 확인합니다. IAM Access Analyzer는 100개 이상의 정책 확인 항목과 실행 가능한 추천을 제공하여 안전하고 기능적인 정책을 작성하도록 돕습니다. 자세한 내용은 IAM 사용 설명서의 IAM Access Analyzer에서 정책 검증을 참조하세요.

- 다중 인증(MFA) 필요 -에서 IAM 사용자 또는 루트 사용자가 필요한 시나리오가 있는 경우 추가 보안을 위해 MFA를 AWS 계정켭니다. API 작업을 직접적으로 호출할 때 MFA가 필요하면 정책에 MFA 조건을 추가합니다. 자세한 내용은 IAM 사용 설명서의 MFA를 통한 보안 API 액세스를 참조하세요.

IAM의 모범 사례에 대한 자세한 내용은 IAM 사용 설명서의 IAM의 보안 모범 사례를 참조하세요.

SimSpace Weaver 콘솔 사용

AWS SimSpace Weaver 콘솔에 액세스하려면 최소 권한 집합이 있어야 합니다. 이러한 권한은의 SimSpace Weaver 리소스에 대한 세부 정보를 나열하고 볼 수 있도록 허용해야 합니다 AWS 계정. 최소 필수 권한보다 더 제한적인 ID 기반 정책을 생성하는 경우, 콘솔이 해당 정책에 연결된 엔티티(사용자 또는 역할)에 대해 의도대로 작동하지 않습니다.

AWS CLI 또는 AWS API만 호출하는 사용자에게는 최소 콘솔 권한을 허용할 필요가 없습니다. 대신, 수행하려는 API 작업과 일치하는 작업에만 액세스할 수 있도록 합니다.

사용자와 역할이 SimSpace Weaver 콘솔을 계속 사용할 수 있도록 하려면 SimSpace Weaver *ConsoleAccess* 또는 *ReadOnly* AWS 관리형 정책도 엔터티에 연결합니다. 자세한 내용은 IAM 사용 설명서의 사용자에게 권한 추가를 참조하세요.

사용자가 자신의 고유한 권한을 볼 수 있도록 허용

이 예제는 IAM 사용자가 자신의 사용자 ID에 연결된 인라인 및 관리형 정책을 볼 수 있도록 허용하는 정책을 생성하는 방법을 보여줍니다. 이 정책에는 콘솔에서 또는 AWS CLI 또는 AWS API를 사용하여 프로그래밍 방식으로이 작업을 완료할 수 있는 권한이 포함됩니다.

```
{
    "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": "*"
        }
    ]
}
```

사용자가 시뮬레이션을 생성하고 실행하도록 허용

이 예제 IAM 정책은 SimSpace Weaver에서 시뮬레이션을 생성하고 실행하는 데 필요한 기본 권한을 제공합니다.

```
{
    "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"
                }
            }
        }
    ]
}
```

에서 자동으로 SimSpace Weaver 생성하는 권한

SimSpace Weaver 프로젝트를 생성하면 서비스가 이름 `weaver-`*`project-name`*`-app-role` 및 IAM 신뢰 정책을 사용하여 AWS Identity and Access Management (IAM) 역할을 생성합니다. 신뢰 정책은 가 사용자를 대신하여 작업을 수행할 수 있도록 역할을 SimSpace Weaver 수임하도록 허용합니다.

앱 역할 권한 정책

시뮬레이션 앱 역할에는 다음과 같은 권한 정책이 있습니다.

```
{
    "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": "*"
        }
    ]
}
```

앱 역할 신뢰 정책

SimSpace Weaver 는 시뮬레이션 앱 역할에 신뢰 관계를 신뢰 정책으로 추가합니다.는 다음 예제와 마찬가지로 각 시뮬레이션에 대한 신뢰 정책을 SimSpace Weaver 생성합니다.

```
{
  "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**
>
> 이 예제에서 계정 번호는 111122223333이고 시뮬레이션 이름은 MySimName입니다. 이러한 값은 신뢰 정책에서 다릅니다.

교차 서비스 혼동된 대리자 방지

혼동된 대리자 문제는 작업을 수행할 권한이 없는 엔터티가 권한이 더 많은 엔터티에 작업을 수행하도록 속일 수 있는 보안 문제입니다. 에서 AWS교차 서비스 가장은 혼동된 대리자 문제를 초래할 수 있습니다. 교차 서비스 가장은 한 서비스(직접적으로 호출하는 서비스)가 다른 서비스(직접적으로 호출되는 서비스)를 직접적으로 호출할 때 발생할 수 있습니다. 호출하는 서비스는 다른 고객의 리소스에 대해 액세스 권한이 없는 방식으로 작동하게 권한을 사용하도록 조작될 수 있습니다. 이를 방지하기 위해 AWS 에서는 계정의 리소스에 대한 액세스 권한이 부여된 서비스 위탁자를 사용하여 모든 서비스에 대한 데이터를 보호하는 데 도움이 되는 도구를 제공합니다.

리소스 정책에서 aws:SourceArn 및 aws:SourceAccount 전역 조건 컨텍스트 키를 사용하여 리소스에 다른 서비스를 AWS SimSpace Weaver 제공하는 권한을 제한하는 것이 좋습니다. 만약 aws:SourceArn 값에 Amazon S3 버킷 Amazon 리소스 이름(ARN)과 같은 계정 ID가 포함되어 있지 않은 경우 권한을 제한하려면 두 전역 조건 컨텍스트 키를 모두 사용해야 합니다. 두 전역 조건 컨텍스트 키와 계정을 포함한 aws:SourceArn 값을 모두 사용하는 경우, aws:SourceAccount 값 및 aws:SourceArn 값의 계정은 동일한 정책 명령문에서 사용할 경우 반드시 동일한 계정 ID를 사용해야 합니다. 하나의 리소스만 교차 서비스 액세스와 연결되도록 허용하려는 경우 aws:SourceArn을 사용하세요. 해당 계정의 모든 리소스가 교차 서비스 사용과 연결되도록 허용하려는 경우 aws:SourceAccount를 사용하세요.

aws:SourceArn 값은 익스텐션의 ARN을 사용해야 합니다.

혼동된 대리인 문제로부터 보호하는 가장 효과적인 방법은 리소스의 전체 ARN이 포함된 aws:SourceArn글로벌 조건 컨텍스트 키를 사용하는 것입니다. 익스텐션의 전체 ARN

을 모를 경우 또는 여러 익스텐션을 지정하는 경우 ARN의 알 수 없는 부분에 대해 와일드카드(*)를 포함한 aws:SourceArn 글로벌 조건 컨텍스트 키를 사용합니다. 예를 들어 arn:aws:*simspaceweaver*:*:*111122223333*:*입니다.

다음 예제에서는의 aws:SourceArn 및 aws:SourceAccount 전역 조건 컨텍스트 키를 사용하여 혼동된 대리자 문제를 SimSpace Weaver 방지하는 방법을 보여줍니다. 이 정책은 요청이 지정된 소스 계정에서 왔고 지정된 ARN과 함께 제공된 경우에만가 역할을 수임 SimSpace Weaver 하도록 허용합니다. 이 경우는 요청자의 자체 계정(*111122223333*) 및 지정된 리전(*us-west-2*)에서만 시뮬레이션 요청의 역할을 SimSpace Weaver 수임할 수 있습니다.

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/*"
        }
      }
    }
  ]
}
```

이 정책을 작성하는 보다 안전한 방법은 다음 예제와 같이 aws:SourceArn에 시뮬레이션 이름을 포함하는 것입니다. 그러면 정책이 MyProjectSimulation_22-10-04_22_10_15라는 시뮬레이션으로 제한됩니다.

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"
        }
      }
    }
  ]
}
```

`aws:SourceArn`에 계정 번호를 명시적으로 포함하는 경우 다음의 간소화된 정책과 같은 `aws:SourceAccount`(자세한 내용은 IAM 사용 설명서 참조)의 `Condition` 요소 테스트를 생략할 수 있습니다.

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"
        }
      }
    }
  ]
}
```

AWS SimSpace Weaver 자격 증명 및 액세스 문제 해결

다음 정보를 사용하여 및 IAM으로 작업할 때 발생할 수 있는 일반적인 문제를 진단 SimSpace Weaver 하고 수정할 수 있습니다.

주제

- 에서 작업을 수행할 권한이 없음 SimSpace Weaver
- iam:PassRole을 수행하도록 인증되지 않음
- 액세스 키를 보아야 합니다.
- 관리자인데 다른 사람이에 액세스하도록 허용하려고 함 SimSpace Weaver
- 내 외부의 사람이 내 SimSpace Weaver 리소스에 액세스 AWS 계정 하도록 허용하고 싶습니다.

에서 작업을 수행할 권한이 없음 SimSpace Weaver

에서 작업을 수행할 권한이 없다는 AWS Management Console 메시지가 표시되면 관리자에게 문의하여 도움을 받아야 합니다. 관리자는 사용자 이름과 암호를 제공한 사람입니다.

다음 예제 오류는 mateojackson IAM 사용자가 콘솔을 사용하여 가상 *my-example-widget* 리소스에 대한 세부 정보를 보려고 하지만 가상 simspaceweaver:*GetWidget* 권한이 없을 때 발생합니다.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 simspaceweaver:GetWidget on resource: my-example-widget
```

이 경우, Mateo는 *my-example-widget* 작업을 사용하여 simspaceweaver:*GetWidget* 리소스에 액세스하도록 허용하는 정책을 업데이트하라고 관리자에게 요청합니다.

iam:PassRole을 수행하도록 인증되지 않음

`iam:PassRole` 작업을 수행할 수 있는 권한이 없다는 오류가 수신되면 SimSpace Weaver에 역할을 전달할 수 있도록 정책을 업데이트해야 합니다.

일부 AWS 서비스 에서는 새 서비스 역할 또는 서비스 연결 역할을 생성하는 대신 기존 역할을 해당 서비스에 전달할 수 있습니다. 이렇게 하려면 역할을 서비스에 전달할 권한이 있어야 합니다.

다음 예 오류는 `marymajor`라는 IAM 사용자가 콘솔을 사용하여 SimSpace Weaver에서 작업을 수행하려고 하는 경우에 발생합니다. 하지만 작업을 수행하려면 서비스 역할이 부여한 권한이 서비스에 있어야 합니다. Mary는 서비스에 역할을 전달할 권한이 없습니다.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

이 경우, Mary가 `iam:PassRole`작업을 수행할 수 있도록 Mary의 정책을 업데이트해야 합니다.

도움이 필요한 경우 AWS 관리자에게 문의하세요. 관리자는 로그인 자격 증명을 제공한 사람입니다.

액세스 키를 보아야 합니다.

IAM 사용자 액세스 키를 생성한 후에는 언제든지 액세스 키 ID를 볼 수 있습니다. 하지만 보안 액세스 키는 다시 볼 수 없습니다. 보안 액세스 키를 잃어버린 경우 새로운 액세스 키 페어를 생성해야 합니다.

액세스 키는 액세스 키 ID(예: `AKIAIOSFODNN7EXAMPLE`)와 보안 액세스 키(예: `wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY`)의 두 가지 부분으로 구성됩니다. 사용자 이름 및 암호와 같이 액세스 키 ID와 보안 액세스 키를 함께 사용하여 요청을 인증해야 합니다. 사용자 이름과 암호를 관리하는 것처럼 안전하게 액세스 키를 관리합니다.

> **Important**
>
> 정식 사용자 ID를 찾는 데 도움이 되더라도 액세스 키를 타사에 제공하지 마시기 바랍니다. 이렇게 하면 누군가에게에 대한 영구 액세스 권한을 부여할 수 있습니다 AWS 계정.

액세스 키 페어를 생성할 때는 액세스 키 ID와 보안 액세스 키를 안전한 위치에 저장하라는 메시지가 나타납니다. 보안 액세스 키는 생성할 때만 사용할 수 있습니다. 하지만 보안 액세스 키를 잃어버린 경우 새로운 액세스 키를 IAM 사용자에게 추가해야 합니다. 최대 두 개의 액세스 키를 가질 수 있습니다. 이미 두 개가 있는 경우 새로 생성하려면 먼저 키 페어 하나를 삭제해야 합니다. 지침을 보려면 IAM 사용 설명서의 액세스 키 관리를 참조하십시오.

관리자인데 다른 사람이에 액세스하도록 허용하려고 함 SimSpace Weaver

다른 사용자에게 액세스를 허용하려면 액세스 권한이 필요한 사용자 또는 애플리케이션에 권한을 부여해야 SimSpace Weaver합니다. AWS IAM Identity Center 를 사용하여 사용자 및 애플리케이션을 관리하는 경우 사용자 또는 그룹에 권한 세트를 할당하여 액세스 수준을 정의합니다. 권한 세트는 IAM 정책을 자동으로 생성하고 사용자 또는 애플리케이션과 연결된 IAM 역할에 할당합니다. 자세한 내용은 AWS IAM Identity Center 사용 설명서에서 권한 세트를 참조하세요.

IAM Identity Center를 사용하지 않는 경우 액세스가 필요한 사용자 또는 애플리케이션에 대한 IAM 엔터티(사용자 또는 역할)를 생성해야 합니다. 그런 다음 SimSpace Weaver에 대한 올바른 권한을 부여하는 정책을 엔터티에 연결해야 합니다. 권한이 부여되면 사용자 또는 애플리케이션 개발자에게 자격 증명을 제공합니다. 이들은 이 자격 증명을 사용하여 AWS에 액세스합니다. IAM 사용자, 그룹, 정책 및 권한 생성에 대해 자세히 알아보려면 IAM 사용자 설명서의 IAM 자격 증명과 IAM의 권한 및 정책을 참조하세요.

내 외부의 사람이 내 SimSpace Weaver 리소스에 액세스 AWS 계정 하도록 허용하고 싶습니다.

다른 계정의 사용자 또는 조직 외부의 사람이 리소스에 액세스할 때 사용할 수 있는 역할을 생성할 수 있습니다. 역할을 수임할 신뢰할 수 있는 사람을 지정할 수 있습니다. 리소스 기반 정책 또는 액세스 제어 목록(ACL)을 지원하는 서비스의 경우, 이러한 정책을 사용하여 다른 사람에게 리소스에 대한 액세스 권한을 부여할 수 있습니다.

자세한 내용은 다음을 참조하세요.

- 에서 이러한 기능을 SimSpace Weaver 지원하는지 여부를 알아보려면 섹션을 참조하세요AWS SimSpace Weaver 에서 IAM을 사용하는 방법.
- 소유 AWS 계정 한의 리소스에 대한 액세스 권한을 제공하는 방법을 알아보려면 IAM 사용 설명서의 소유한 다른의 IAM 사용자에게 액세스 권한 제공을 참조 AWS 계정 하세요.
- 타사에 리소스에 대한 액세스 권한을 제공하는 방법을 알아보려면 IAM 사용 설명서의 타사가 AWS 계정 소유한에 대한 액세스 권한 제공을 AWS 계정참조하세요.
- ID 페더레이션을 통해 액세스 권한을 제공하는 방법을 알아보려면 IAM 사용 설명서의 외부에서 인증된 사용자에게 액세스 권한 제공(ID 페더레이션)을 참조하세요.
- 크로스 계정 액세스에 대한 역할과 리소스 기반 정책 사용의 차이점을 알아보려면 IAM 사용 설명서의 IAM의 크로스 계정 리소스 액세스를 참조하세요.

의 보안 이벤트 로깅 및 모니터링 AWS SimSpace Weaver

모니터링은 SimSpace Weaver 및 AWS 솔루션의 안정성, 가용성 및 성능을 유지하는 데 중요한 부분입니다. 다중 지점 실패가 발생할 경우 보다 쉽게 디버깅할 수 있도록 AWS 솔루션의 모든 부분으로부터 모니터링 데이터를 수집해야 합니다.

AWS 및 SimSpace Weaver 는 시뮬레이션 리소스를 모니터링하고 잠재적 인시던트에 대응하기 위한 여러 도구를 제공합니다.

Amazon CloudWatch의 로그

SimSpace Weaver 는 로그를 CloudWatch에 저장합니다. 이러한 로그를 사용하여 시뮬레이션의 이벤트(예: 앱 시작 및 중지)를 모니터링하고 디버깅할 수 있습니다. 자세한 내용은 SimSpace Weaver Amazon CloudWatch Logs의 로그 단원을 참조하십시오.

Amazon CloudWatch 경보

Amazon CloudWatch 경보를 사용하면 지정한 기간 동안 단일 지표를 감시합니다. 지표가 지정된 임계값을 초과하면 Amazon SNS 주제 또는 AWS Auto Scaling 정책으로 알림이 전송됩니다. CloudWatch 경보는 특정 상태가 아니라 상태 변경에 의해 트리거되고 지정된 수의 기간 동안 유지됩니다. 자세한 내용은 Amazon CloudWatch SimSpace Weaver 를 사용한 모니터링 단원을 참조하십시오.

AWS CloudTrail 로그

CloudTrail은 사용자, 역할 또는 AWS 서비스가 수행한 작업에 대한 레코드를 제공합니다 SimSpace Weaver. CloudTrail에서 수집한 정보를 사용하여 수행된 요청, 요청이 수행된 SimSpace Weaver IP 주소, 요청을 수행한 사람, 요청이 수행된 시간 및 추가 세부 정보를 확인할 수 있습니다. 자세한 내용은 를 사용하여 AWS SimSpace Weaver API 호출 로깅 AWS CloudTrail 단원을 참조하십시오.

에 대한 규정 준수 검증 AWS SimSpace Weaver

SimSpace Weaver 는 규정 AWS 준수 프로그램의 범위에 속하지 않습니다.

타사 감사자는 여러 규정 준수 프로그램의 일환으로 다른 AWS 서비스의 보안 및 AWS 규정 준수를 평가합니다. 여기에는 SOC, PCI, FedRAMP, HIPAA 등이 포함됩니다.

AWS 서비스 가 특정 규정 준수 프로그램의 범위 내에 있는지 알아보려면 AWS 서비스 규정 준수 프로그램 범위 내 참조하고 관심 있는 규정 준수 프로그램을 선택합니다. 일반 정보는 AWS 규정 준수 프로그램.

를 사용하여 타사 감사 보고서를 다운로드할 수 있습니다 AWS Artifact. 자세한 내용은 Downloading Reports inDownloading AWS Artifact을 참조하세요.

사용 시 규정 준수 책임은 데이터의 민감도, 회사의 규정 준수 목표, 관련 법률 및 규정에 따라 AWS 서비스 결정됩니다. 사용 시 규정 준수 책임에 대한 자세한 내용은 AWS 보안 설명서를 AWS 서비스참조하세요.

의 복원력 AWS SimSpace Weaver

AWS 글로벌 인프라는 AWS 리전 및 가용 영역을 중심으로 구축됩니다.는 물리적으로 분리되고 격리된 여러 가용 영역을 AWS 리전 제공하며,이 가용 영역은 지연 시간이 짧고 처리량이 높으며 중복성이 높은 네트워킹과 연결됩니다. 가용 영역을 사용하면 중단 없이 영역 간에 자동으로 장애 극복 조치가 이루어지는 애플리케이션 및 데이터베이스를 설계하고 운영할 수 있습니다. 가용 영역은 기존의 단일 또는 다중 데이터 센터 인프라보다 가용성, 내결함성, 확장성이 뛰어납니다.

AWS 리전 및 가용 영역에 대한 자세한 내용은 AWS 글로벌 인프라를 참조하세요.

의 인프라 보안 AWS SimSpace Weaver

관리형 서비스인는 AWS 글로벌 네트워크 보안으로 보호 AWS SimSpace Weaver 됩니다. AWS 보안 서비스 및가 인프라를 AWS 보호하는 방법에 대한 자세한 내용은 AWS 클라우드 보안을 참조하세요. 인프라 보안 모범 사례를 사용하여 환경을 설계하려면 보안 원칙 AWS Well-Architected Framework의 인프라 보호를 참조하세요 AWS .

AWS 에서 게시한 API 호출을 사용하여 네트워크를 SimSpace Weaver 통해에 액세스합니다. 클라이언트는 다음을 지원해야 합니다.

- Transport Layer Security(TLS). TLS 1.2는 필수이며 TLS 1.3을 권장합니다.
- DHE(Ephemeral Diffie-Hellman) 또는 ECDHE(Elliptic Curve Ephemeral Diffie-Hellman)와 같은 완전 전송 보안(PFS)이 포함된 암호 제품군. Java 7 이상의 최신 시스템은 대부분 이러한 모드를 지원합니다.

네트워크 연결 보안 모델

시뮬레이션은 선택한 AWS 리전 내에 위치한 Amazon VPC 내의 컴퓨팅 인스턴스에서 실행됩니다. Amazon VPC는 AWS 클라우드의 가상 네트워크로, 워크로드 또는 조직 엔터티별로 인프라를 격리합니다. Amazon VPC 내의 컴퓨팅 인스턴스 간 통신은 AWS 네트워크 내에 유지되며 인터넷을 통해 이동하지 않습니다. 일부 내부 서비스 통신은 인터넷을 통해 전달되며 암호화됩니다. 동일한 AWS 리전

에서 실행되는 모든 고객에 대한 시뮬레이션은 동일한 Amazon VPC를 공유합니다. 여러 고객을 위한 시뮬레이션은 동일한 Amazon VPC 내에서 별도의 컴퓨팅 인스턴스를 사용합니다.

시뮬레이션 클라이언트와에서 실행되는 시뮬레이션 간의 통신은 인터넷을 통해 SimSpace Weaver 이루어집니다.는 이러한 연결을 처리하지 SimSpace Weaver 않습니다. 클라이언트 연결을 보호하는 것은 사용자의 책임입니다.

SimSpace Weaver 서비스에 대한 연결은 인터넷을 통해 암호화됩니다. 여기에는 AWS Management Console AWS Command Line Interface (AWS CLI), AWS 소프트웨어 개발 키트(SDK) 및 SimSpace Weaver 앱 SDK를 사용하는 연결이 포함됩니다.

의 구성 및 취약성 분석 AWS SimSpace Weaver

구성 및 IT 제어는 AWS 와 사용자 간의 공동 책임입니다. 자세한 내용은 AWS 공동 책임 모델 참조하세요. 컴퓨팅 인스턴스에서 운영 체제 패치 적용, 방화벽 구성 및 인프라 재해 복구와 같은 기본 AWS 인프라에 대한 기본 보안 작업을 AWS 처리합니다. 적합한 제3자가 이 절차를 검토하고 인증하였습니다. 자세한 내용은 보안, 자격 증명 및 규정 준수를 위한 모범 사례를 참조하세요.

시뮬레이션 소프트웨어의 보안에 대한 책임은 사용자에게 있습니다.

- 업데이트 및 보안 패치를 포함하여 앱 코드를 유지 관리합니다.
- 시뮬레이션 클라이언트와 해당 클라이언트가 연결하는 앱 간의 통신을 인증하고 암호화합니다.
- SDK 및 SimSpace Weaver 앱 SDK를 포함한 최신 AWS SDK 버전을 사용하도록 시뮬레이션을 업데이트합니다.

> **Note**
>
> SimSpace Weaver 는 실행 중인 시뮬레이션에서 앱에 대한 업데이트를 지원하지 않습니다. 앱을 업데이트해야 하는 경우 시뮬레이션을 중지하고 삭제한 다음 업데이트된 앱 코드로 새 시뮬레이션을 생성해야 합니다. 시뮬레이션을 다시 생성해야 하는 경우 복원할 수 있도록 외부 데이터 스토어에 시뮬레이션 상태를 저장하는 것이 좋습니다.

에 대한 보안 모범 사례 SimSpace Weaver

이 섹션에서는 관련 보안 모범 사례를 설명합니다 SimSpace Weaver. 의 보안 모범 사례에 대한 자세한 내용은 보안, 자격 증명 및 규정 준수 모범 사례를 AWS참조하세요. https://aws.amazon.com/architecture/security-identity-compliance

주제

앱과 해당 클라이언트 간 통신 암호화

SimSpace Weaver 는 앱과 클라이언트 간의 통신을 관리하지 않습니다. 클라이언트 세션에 대해 특정 형태의 인증 및 암호화를 구현해야 합니다.

시뮬레이션 상태를 주기적 백업

SimSpace Weaver 는 시뮬레이션 상태를 저장하지 않습니다. API 직접 호출, 콘솔 옵션 또는 시스템 충돌로 인해 중지된 시뮬레이션은 해당 상태를 저장하지 않으며 복구할 고유한 방법도 없습니다. 중지된 시뮬레이션은 다시 시작할 수 없습니다. 재시작과 동일한 작업을 수행하는 유일한 방법은 동일한 구성과 데이터를 사용하여 시뮬레이션을 다시 생성하는 것입니다. 시뮬레이션 상태의 백업을 사용하여 새 시뮬레이션을 초기화할 수 있습니다. AWS 는 시뮬레이션 상태를 저장하는 데 사용할 수 있는 매우 안정적이고 가용성이 높은 클라우드 스토리지 및 데이터베이스 서비스를 제공합니다.

앱 및 SDK 유지 관리

앱, AWS 소프트웨어 개발 키트(SDKs)의 로컬 설치 및 SimSpace Weaver 앱 SDK를 유지 관리합니다. 새 버전의 AWS SDKs. 비프로덕션 SimSpace Weaver 앱 빌드로 앱 SDK의 새 버전을 테스트하여 앱이 예상대로 계속 실행되는지 확인합니다. 실행 중인 시뮬레이션에서는 앱을 업데이트할 수 없습니다. 앱을 업데이트하려면 다음을 수행합니다.

1. 로컬(또는 테스트 환경)에서 앱 코드를 업데이트하고 테스트합니다.
2. 시뮬레이션 상태 변경을 중지하고 저장합니다(필요한 경우).
3. 시뮬레이션을 중지합니다(중지된 후에는 다시 시작할 수 없음).
4. 시뮬레이션을 삭제합니다(삭제되지 않은 중지된 시뮬레이션은 서비스 제한에 포함됨).
5. 동일한 구성과 업데이트된 앱 코드로 시뮬레이션을 다시 생성합니다.
6. 저장된 상태 데이터(사용 가능한 경우)를 사용하여 시뮬레이션을 초기화합니다.
7. 새 시뮬레이션을 시작합니다.

> **Note**
>
> 동일한 구성으로 만든 새 시뮬레이션은 이전 시뮬레이션과 별개입니다. 그러면 새 시뮬레이션 ID가 생기고 Amazon CloudWatch의 새 로그 스트림으로 로그를 전송합니다.

에서 로깅 및 모니터링 SimSpace Weaver

모니터링은 SimSpace Weaver 및 다른 AWS 솔루션의 안정성, 가용성 및 성능을 유지하는 데 중요한 부분입니다.는 다음과 같은 모니터링 도구를 AWS 제공하여 모니터링 SimSpace Weaver, 보고 및 문제 발생 시 적절한 경우 자동 조치를 취합니다.

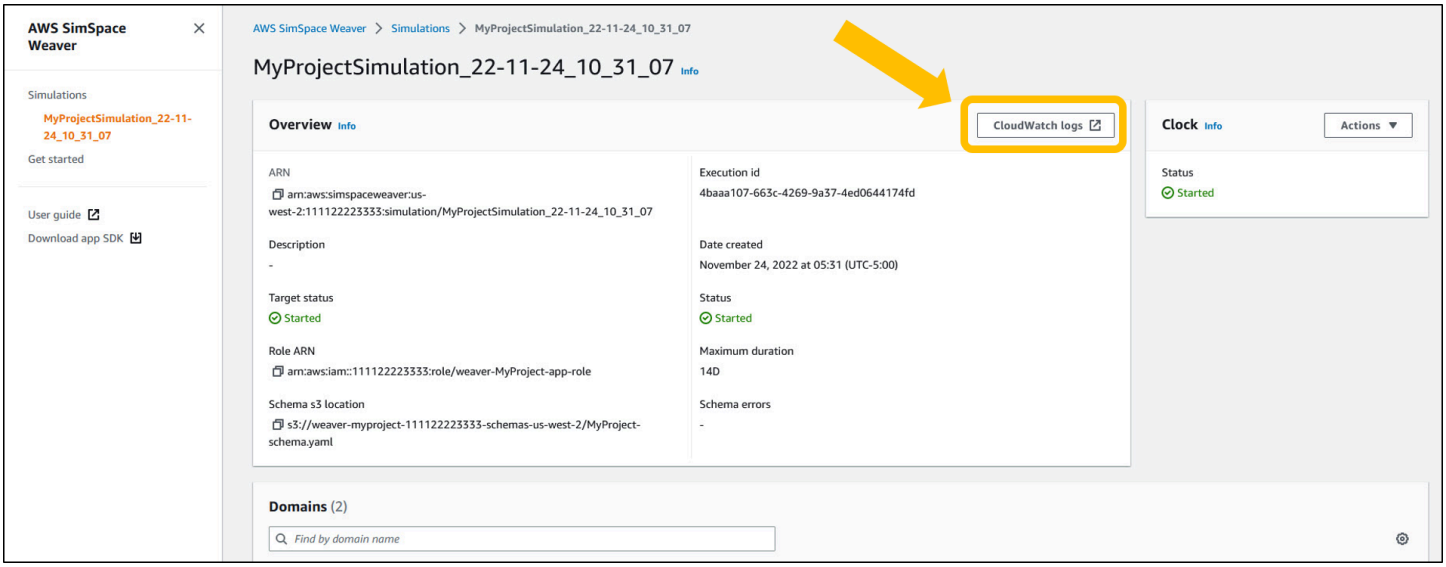
- Amazon CloudWatch는 AWS 리소스와 AWS 실행 중인 애플리케이션을 실시간으로 모니터링합니다. 지표를 수집 및 추적하고, 사용자 지정 대시보드를 생성할 수 있으며, 지정된 지표가 지정한 임곗값에 도달하면 사용자에게 알리거나 조치를 취하도록 경보를 설정할 수 있습니다. 자세한 내용은 Amazon CloudWatch 사용 설명서를 참조하십시오.
- Amazon CloudWatch Logs를 사용하면 SimSpace Weaver 작업자, CloudTrail 및 기타 소스의 로그 데이터를 모니터링, 저장 및 액세스할 수 있습니다. CloudWatch Logs는 로그 데이터의 정보를 모니터링하고 특정 임계값에 도달하면 사용자에게 알릴 수 있습니다. 또한 매우 내구성이 뛰어난 스토리지에 로그 데이터를 저장할 수 있습니다. 자세한 내용은 Amazon CloudWatch Logs 사용 설명서를 참조하십시오.
- AWS CloudTrail은 직접 수행하거나 AWS 계정 를 대신하여 수행한 API 호출 및 관련 이벤트를 캡처하고 지정한 Amazon S3 버킷에 로그 파일을 전송합니다. 어떤 사용자 및 계정이 AWS를 호출했는지 어떤 소스 IP 주소에 호출이 이루어졌는지 언제 호출이 발생했는지 확인할 수 있습니다. 자세한 내용은 AWS CloudTrail 사용 설명서를 참조하십시오.

주제

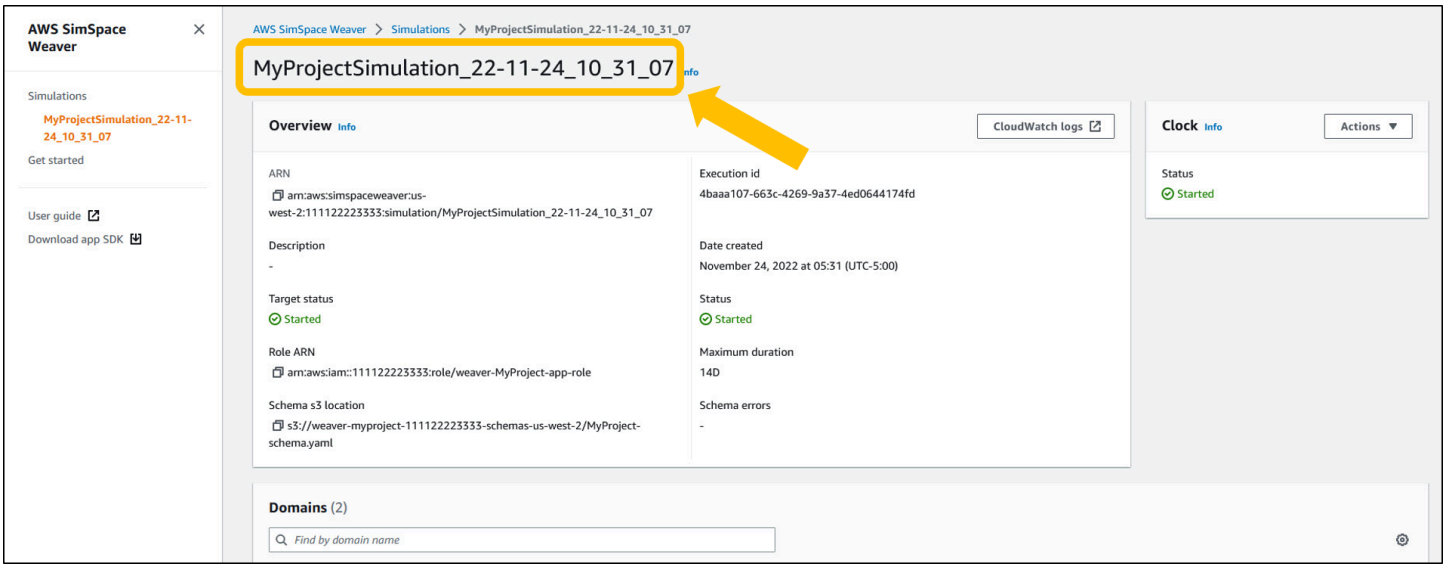
SimSpace Weaver Amazon CloudWatch Logs의 로그

SimSpace Weaver 로그 액세스

SimSpace Weaver 시뮬레이션에서 생성된 모든 로그는 Amazon CloudWatch Logs에 저장됩니다. 로그에 액세스하려면 SimSpace Weaver 콘솔에서 시뮬레이션의 개요 창에 있는 CloudWatch logs 버튼을 사용하면 됩니다. 그러면 해당 시뮬레이션의 로그로 바로 이동합니다.



CloudWatch 콘솔을 통해 로그에 액세스할 수도 있습니다. 로그를 검색하려면 시뮬레이션 이름이 필요합니다.



SimSpace Weaver 로그

SimSpace Weaver 는 앱의 시뮬레이션 관리 메시지와 콘솔 출력을 Amazon CloudWatch Logs에 기록합니다. 로그 작업에 대한 자세한 내용은 Amazon CloudWatch Logs 사용 설명서의 로그 그룹 및 로그 스트림 작업을 참조하세요.

CloudWatch Logs에서 생성하는 각 시뮬레이션에는 자체 로그 그룹이 있습니다. 로그 그룹의 이름은 시뮬레이션 스키마에 지정됩니다. 다음 스키마 스니펫에서 `log_destination_service`의 값은 `logs`입니다. 즉, `log_destination_resource_name`의 값은 로그 그룹의 이름입니다. 이 경우 로그 그룹은 `MySimulationLogs`입니다.

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

시뮬레이션을 시작한 후 DescribeSimulation API를 사용하여 시뮬레이션에 사용할 로그 그룹의 이름을 찾을 수도 있습니다.

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

다음 예제는 로깅 구성을 설명하는 DescribeSimulation의 출력 부분을 보여줍니다. 로그 그룹의 이름은 LogGroupArn 끝에 표시됩니다.

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

각 시뮬레이션 로그 그룹에는 여러 로그 스트림이 포함되어 있습니다.

- 관리 로그 스트림 - SimSpace Weaver 서비스에서 생성된 시뮬레이션 관리 메시지입니다.

  ```
  /sim/management
  ```

- 오류 로그 스트림 - SimSpace Weaver 서비스에서 생성된 오류 메시지입니다. 이 로그 스트림은 오류가 있는 경우에만 존재합니다.는 앱에서 작성한 오류를 자체 앱 로그 스트림에 SimSpace Weaver 저장합니다(다음 로그 스트림 참조).

```
/sim/errors
```

- 공간 앱 로그 스트림(각 작업자의 공간 앱당 1개) - 공간 앱에서 생성된 콘솔 출력입니다. 각 공간 앱은 자체 로그 스트림에 씁니다. *spatial-app-id*는 *worker-id* 끝의 후행 슬래시 뒤에 오는 모든 문자입니다.

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

- 사용자 지정 앱 로그 스트림(각 사용자 지정 앱 인스턴스당 1개) - 사용자 지정 앱에서 생성된 콘솔 출력입니다. 각 사용자 지정 앱 인스턴스는 자체 로그 스트림에 씁니다.

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

- 서비스 앱 로그 스트림(각 서비스 앱 인스턴스당 1개) - 서비스 앱에서 생성된 콘솔 출력입니다. 각 공간 앱은 자체 로그 스트림에 씁니다. *service-app-id*는 *service-app-name* 끝의 후행 슬래시 뒤에 오는 모든 문자입니다.

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

Amazon CloudWatch SimSpace Weaver 를 사용한 모니터링

원시 데이터를 수집하여 읽기 가능하며 실시간에 가까운 지표로 처리하는 Amazon CloudWatch를 SimSpace Weaver 사용하여 모니터링할 수 있습니다. 이러한 통계는 15개월간 보관되므로 기록 정보에 액세스하고 웹 애플리케이션 또는 서비스가 어떻게 실행되고 있는지 전체적으로 더 잘 파악할 수 있습니다. 특정 임곗값을 주시하다가 해당 임곗값이 충족될 때 알림을 전송하거나 조치를 취하도록 경보를 설정할 수도 있습니다. 자세한 내용은 Amazon CloudWatch 사용 설명서를 참조하세요.

SimSpace Weaver 서비스는 `AWS/simspaceweaver` 네임스페이스에 다음 지표를 보고합니다.

SimSpace Weaver 계정 수준의 지표

SimSpace Weaver 네임스페이스에는 AWS 계정 수준의 활동과 관련된 다음 지표가 포함됩니다.

지표	설명
`SimulationCount`	현재 계정의 시뮬레이션 횟수. 단위: 개

지표	설명
	측정 기준: 없음 통계: 평균, 최소값, 최대값, 합계

를 사용하여 AWS SimSpace Weaver API 호출 로깅 AWS CloudTrail

AWS SimSpace Weaver 는 사용자 AWS CloudTrail, 역할 또는 서비스가 수행한 작업에 대한 레코드를 제공하는 AWS 서비스와 통합됩니다 SimSpace Weaver. CloudTrail은 SimSpace Weaver 에 대한 모든 API 직접 호출을 이벤트로 캡처합니다. 캡처되는 호출에는 SimSpace Weaver 콘솔의 호출과 SimSpace Weaver API 작업에 대한 코드 호출이 포함됩니다. 추적을 생성하면 이벤트를 포함하여 CloudTrail 이벤트를 지속적으로 Amazon S3 버킷에 배포할 수 있습니다 SimSpace Weaver. 추적을 구성하지 않은 경우에도 Event history의 CloudTrail 콘솔 이벤트 기록에서 최신 이벤트를 볼 수 있습니다. CloudTrail에서 수집한 정보를 사용하여 수행된 요청, 요청이 수행된 SimSpace Weaver IP 주소, 요청을 수행한 사람, 요청이 수행된 시간 및 추가 세부 정보를 확인할 수 있습니다.

CloudTrail에 대한 자세한 설명은 AWS CloudTrail 사용자 가이드를 참조하십시오.

SimSpace Weaver CloudTrail의 정보

CloudTrail은 계정을 생성할 AWS 계정 때에서 활성화됩니다. 에서 활동이 발생하면 SimSpace Weaver해당 활동이의 다른 AWS 서비스 이벤트와 함께 CloudTrail 이벤트에 기록됩니다Event history. 에서 최근 이벤트를 보고 검색하고 다운로드할 수 있습니다 AWS 계정. 자세한 내용은 CloudTrail 이벤트 기록을 사용하여 이벤트 보기를 참조하세요.

에 대한 이벤트를 AWS 계정포함하여에 이벤트를 지속적으로 기록하려면 추적을 SimSpace Weaver 생성합니다. CloudTrail은 추적을 사용하여 Amazon S3 버킷으로 로그 파일을 전송할 수 있습니다. 콘솔에서 트레일을 생성하면 기본적으로 모든 AWS 리전에 트레일이 적용됩니다. 추적은 AWS 파티션의 모든 리전에서 이벤트를 로깅하고 지정한 Amazon S3 버킷으로 로그 파일을 전송합니다. 또한 CloudTrail 로그에서 수집된 이벤트 데이터를 추가로 분석하고 조치를 취하도록 다른 AWS 서비스를 구성할 수 있습니다. 자세한 내용은 다음 자료를 참조하세요.

- 추적 생성 개요
- CloudTrail 지원 서비스 및 통합
- CloudTrail에 대한 Amazon SNS 알림 구성

- 여러 리전에서 CloudTrail 로그 파일 받기 및 여러 계정에서 CloudTrail 로그 파일 받기

모든 SimSpace Weaver 작업은 CloudTrail에서 로깅되며 AWS SimSpace Weaver API 참조에 문서화됩니다. 예를 들어 `ListSimulations`, `DescribeSimulation`, `DeleteSimulation` 작업을 직접 호출하면 CloudTrail 로그 파일에 항목이 생성됩니다.

모든 이벤트 또는 로그 항목에는 요청을 생성했던 사용자에 관한 정보가 포함됩니다. ID 정보를 이용하면 다음을 쉽게 판단할 수 있습니다.

- 요청이 루트 또는 AWS Identity and Access Management (IAM) 사용자 자격 증명으로 이루어졌는지 여부입니다.
- 역할 또는 페더레이션 사용자의 임시 자격 증명을 사용하여 요청이 생성되었는지 여부.
- 요청이 다른 AWS 서비스에서 이루어졌는지 여부입니다.

자세한 내용은 CloudTrail userIdentity 요소를 참조하세요.

SimSpace Weaver 로그 파일 항목 이해

트레일이란 지정한 S3 버킷에 이벤트를 로그 파일로 입력할 수 있게 하는 구성입니다. CloudTrail 로그 파일에는 하나 이상의 로그 항목이 포함될 수 있습니다. 이벤트는 모든 소스로부터의 단일 요청을 나타내며 작업 날짜와 시간, 요청 파라미터 및 기타 세부 사항과 같은 요청 작업에 관한 정보가 들어 있습니다. CloudTrail 로그 파일은 퍼블릭 API 직접 호출의 주문 스택 트레이스가 아니므로 특정 순서로 표시되지 않습니다.

다음은 `ListSimulations` 작업을 보여주는 CloudTrail 로그 항목이 나타낸 예시입니다.

```
{
    "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 엔드포인트 및 할당량

다음 테이블에서는 SimSpace Weaver에 대한 서비스 엔드포인트 및 Service Quotas에 대해 설명합니다. Service Quotas(제한이라고도 함)은 AWS 계정의 최대 서비스 리소스 또는 작업 수입니다. 자세한 내용을 알아보려면 AWS 일반 참조의 AWS 서비스 할당량을 참조하세요.

Service endpoints

지역명	리전	엔드포인트	프로토콜
미국 동부(버지니아 북부)	us-east-1	simspaceweaver.us-east-1.amazonaws.com	HTTPS
미국 동부(오하이오)	us-east-2	simspaceweaver.us-east-2.amazonaws.com	HTTPS
미국 서부(오레곤)	us-west-2	simspaceweaver.us-west-2.amazonaws.com	HTTPS
아시아 태평양(싱가포르)	ap-southeast-1	simspaceweaver.ap-southeast-1.amazonaws.com	HTTPS
아시아 태평양(시드니)	ap-southeast-2	simspaceweaver.ap-southeast-2.amazonaws.com	HTTPS
유럽(스톡홀름)	eu-north-1	simspaceweaver.eu-north-1.amazonaws.com	HTTPS
유럽(프랑크푸르트)	eu-central-1	simspaceweaver.eu-central-1.amazonaws.com	HTTPS

지역명	리전	엔드포인트	프로토콜
유럽(아일랜드)	eu-west-1	simspaceweaver.eu-west-1.amazonaws.com	HTTPS
AWS GovCloud(미국 동부)	us-gov-east-1	simspaceweaver.us-gov-east-1.amazonaws.com	HTTPS
AWS GovCloud(미국 서부)	us-gov-west-1	simspaceweaver.us-gov-west-1.amazonaws.com	HTTPS

서비스 할당량

명칭	기본값	조정 가능	설명
각 앱의 리소스 단위를 계산합니다.	지원되는 각 리전: 4	아니요	각 앱에 할당할 수 있는 최대 컴퓨팅 리소스 단위 수입니다.
각 작업자의 리소스 단위를 계산합니다.	지원되는 각 리전: 17	아니요	각 작업자에 사용할 수 있는 컴퓨팅 리소스 단위 수입니다.
각 엔터티의 데이터 필드	지원되는 각 리전: 7	아니요	엔터티가 가질 수 있는 최대 데이터(비인덱스) 필드 수입니다.
파티션의 엔터티	지원되는 각 리전: 8,192	아니요	한 파티션의 최대 엔터티 수입니다.

명칭	기본값	조정 가능	설명
항목 데이터 필드 크기	지원되는 각 리전: 1,024바이트	아니요	엔터티의 데이터(비인덱스) 필드의 최대 크기입니다.
작업자 간 엔터티 전송	지원되는 각 리전: 25개	아니요	각 파티션 및 각 틱에 대한 작업자 간 최대 엔터티 전송 수입니다.
동일한 작업자에서의 엔터티 전송	지원되는 각 리전: 500개	아니요	각 파티션과 각 틱에 대해 동일한 작업자에서 엔터티를 전송할 수 있는 최대 수입니다.
각 엔터티의 인덱스 필드	지원되는 각 리전: 1	아니요	엔터티가 가질 수 있는 인덱스 필드의 최대 수입니다.
시뮬레이션의 가장 큰 최대 지속 시간(일 수)	지원되는 각 리전: 14	아니요	시뮬레이션의 최대 지속 시간으로 지정할 수 있는 최대 일수입니다. 값을 지정하지 않더라도 모든 시뮬레이션에는 최대 지속 시간이 있습니다. 시뮬레이션이 해당 최대 지속 시간에 도달하면 자동으로 중지됩니다.
앱 컴퓨팅 리소스 단위 메모리	지원되는 각 리전: 1기가바이트	아니요	앱이 각 컴퓨팅 리소스 단위에 대해 확보하는 랜덤 액세스 메모리(RAM)의 양입니다.

명칭	기본값	조정 가능	설명
각 작업자에 대한 원격 구독	지원되는 각 리전: 24	아니요	각 작업자의 최대 원격 구독 수입니다.
시뮬레이션 횟수	지원되는 각 리전: 2	예	계정에서 목표 상태가 STARTED인 시뮬레이션의 최대 수입니다. 10개까지 할당량 증가를 요청할 수 있습니다.
시뮬레이션을 위한 작업자	지원되는 각 리전: 2	예	시뮬레이션 1개에 할당할 수 있는 최대 작업자 수입니다. 10개까지 할당량 증가를 요청할 수 있습니다.
각 컴퓨팅 리소스 단위의 vCPU	지원되는 각 리전: 2	아니요	앱이 각 컴퓨팅 리소스 단위에 할당하는 vCPU(가상 중앙 처리 단위)의 수입니다.

메시징 할당량

다음 할당량은 및의 앱 간 메시징SimSpace Weaver Local에 적용됩니다 AWS 클라우드.

명칭	기본값	조정 가능	설명
최대 메시지 크기	지원되는 각 리전: 256 바이트	아니요	메시지 페이로드의 최대 크기입니다.
최대 메시지 전송 속도	지원되는 각 리전: 128 개	아니요	각 앱이 틱당 보낼 수 있는 최대 메시지 수입니다.

클럭 속도

시뮬레이션 스키마는 시뮬레이션의 클럭 속도(틱 레이트라고도 함)를 지정합니다. 사용할 수 있는 유효한 클럭 속도가 다음 테이블에 나와 있습니다.

명칭	유효값	설명
클럭 속도	지원되는 각 리전: "10", "15", "30", "무제한"	시뮬레이션에 유효한 클럭 속도입니다.
클럭 속도(버전 1.13 및 1.12)	지원되는 각 리전: 10, 15, 30	시뮬레이션에 유효한 클럭 속도입니다.

SimSpace Weaver Local의 Service Quotas

다음 Service Quotas는 SimSpace Weaver Local에만 적용됩니다. 다른 모든 할당량도 SimSpace Weaver Local에 적용됩니다.

명칭	기본값	조정 가능	설명
최대 파티션	SimSpace Weaver Local: 24회	아니요	시뮬레이션의 최대 파티션 수입니다.
최대 앱	SimSpace Weaver Local: 24회	아니요	시뮬레이션을 위한 최대 총 앱 수(모든 유형)입니다.
최대 도메인	SimSpace Weaver Local: 24회	아니요	시뮬레이션을 위한 최대 총 도메인 수(모든 유형)입니다.
파티션의 엔터티	SimSpace Weaver Local: 4,096	아니요	각 파티션의 최대 엔터티 수입니다.
엔터티당 필드	SimSpace Weaver Local: 8	아니요	각 엔터티에 대한 최대 필드 수입니다.

명칭	기본값	조정 가능	설명
필드 크기	SimSpace Weaver Local: 1,024바이트	아니요	엔터티 필드의 최대 크기입니다.

의 문제 해결 SimSpace Weaver

주제

AssumeRoleAccessDenied

시뮬레이션 시작에 실패할 경우 다음 오류가 발생할 수 있습니다.

```
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
```

시뮬레이션의 AWS Identity and Access Management (IAM) 역할에 대해 다음 중 하나에 해당하는 경우이 오류가 발생할 수 있습니다.

- Amazon 리소스 이름(ARN)은 존재하지 않는 IAM 역할을 가리킵니다.
- 새 시뮬레이션의 이름이 역할을 맡는 것을 허용하지 않는 IAM 역할에 대한 신뢰 정책입니다.

역할이 존재하는지 확인합니다. 역할이 존재하는 경우 역할에 대한 신뢰 정책을 확인합니다. 다음 예제의 신뢰 정책의 `aws:SourceArn`은 이름이 `MySimulation`으로 시작하는 시뮬레이션(계정 111122223333)만 역할을 맡도록 허용합니다.

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*"
                }
            }
        }
    ]
}
```

이름이 MyOtherSimulation으로 시작하는 다른 시뮬레이션이 역할을 맡도록 허용하려면 다음 편집된 예제와 같이 신뢰 정책을 수정해야 합니다.

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

프로젝트를 만드는 동안 다음과 같은 오류가 발생할 수 있습니다.

```
An error occurred (InvalidBucketName) when calling the CreateBucket operation: The
 specified bucket is not valid.
```

Amazon Simple Storage Service(Amazon S3)에 SimSpace Weaver 전달된 이름이 버킷 이름 지정 규칙을 위반했기 때문에이 오류가 발생했습니다(자세한 내용은 Amazon Simple Storage Service 사용 설명서의 버킷 이름 지정 규칙 참조).

SimSpace Weaver 앱 SDK의 `create-project` 스크립트는 스크립트에 제공한 프로젝트 이름을 사용하여 버킷 이름을 생성합니다. 버킷 이름은 다음 형식을 사용합니다.

- 버전 1.13.x 이상
  - `weaver-`*`lowercase-project-name`*`-`*`account-number`*`-`*`region`*
- 버전 1.12.x
  - `weaver-`*`lowercase-project-name`*`-`*`account-number`*`-app-zips-`*`region`*
  - `weaver-`*`lowercase-project-name`*`-`*`account-number`*`-schemas-`*`region`*

예를 들어, 다음과 같은 프로젝트 속성이 주어집니다.

- 프로젝트 이름: `MyProject`
- AWS 계정 번호: `111122223333`
- AWS 리전: `us-west-2`

프로젝트에는 다음과 같은 버킷이 있을 것입니다.

- 버전 1.13.x 이상
  - `weaver-myproject-111122223333-us-west-2`
- 버전 1.12.x

- `weaver-myproject-111122223333-app-zips-us-west-2`
- `weaver-myproject-111122223333-schemas-us-west-2`

프로젝트 이름이 Amazon S3 이름 지정 규칙을 위반해서는 안 됩니다. 또한 `create-project` 스크립트로 생성한 버킷 이름이 Amazon S3 버킷의 이름 길이 제한을 초과하지 않도록 충분히 짧은 프로젝트 이름을 사용해야 합니다.

ServiceQuotaExceededException

시뮬레이션을 시작하면 다음 오류가 발생할 수 있습니다.

```
An error occurred (ServiceQuotaExceededException) when calling the StartSimulation
 operation: Failed to start simulation due to: simulation quota has already been
 reached.
```

새 시뮬레이션을 시작하려고 하는데 현재 계정에 목표 상태가 STARTED인 최대 시뮬레이션 수가 있는 경우 이 오류가 발생합니다. 여기에는 실행 중인 시뮬레이션, 실패한 시뮬레이션, 최대 지속 시간에 도달하여 중지된 시뮬레이션이 포함됩니다. 중지되거나 실패한 시뮬레이션을 삭제하여 새 시뮬레이션을 시작할 수 있습니다. 모든 시뮬레이션이 실행 중인 경우 실행 중인 시뮬레이션을 중지하고 삭제할 수 있습니다. 아직 요청 제한에 도달하지 않은 경우 Service Quotas 증가를 요청할 수도 있습니다.

TooManyBuckets

프로젝트를 만드는 동안 다음과 같은 오류가 발생할 수 있습니다.

```
An error occurred (TooManyBuckets) when calling the CreateBucket operation: You have
 attempted to create more buckets than allowed.
```

Amazon Simple Storage Service(Amazon S3)에는 AWS 계정에 포함할 수 있는 버킷 수에 제한이 있습니다(자세한 내용은 Amazon Simple Storage Service 사용 설명서의 버킷 제한 및 제한 참조).

계속하려면 다음 중 하나를 수행해야 합니다.

- 필요 없는 기존 Amazon S3 버킷을 2개 이상 삭제합니다.
- Amazon S3 한도 증가를 요청합니다(자세한 내용은 Amazon Simple Storage Service 사용 설명서의 버킷 규제 및 제한 참조).

- 다른 AWS 계정을 사용합니다.

> **Note**
>
> 의 `DeleteSimulation` API는 시뮬레이션과 연결된 Amazon S3 리소스를 삭제하지 SimSpace Weaver 않습니다. 시뮬레이션과 관련된 리소스가 더 이상 필요하지 않은 경우 모두 제거하는 것이 좋습니다.

시뮬레이션 시작 중 권한 거부됨

시뮬레이션을 시작하면 권한이 거부되었거나 앱 아티팩트에 액세스하는 동안 오류가 발생했다는 오류 메시지가 표시될 수 있습니다. 이 문제는에서 자동으로 생성하지 SimSpace Weaver 않은 시뮬레이션용 Amazon S3 버킷을 지정할 때(콘솔 또는 SimSpace Weaver 앱 SDK 스크립트를 통해) 발생할 수 있습니다.

근본 원인일 가능성이 가장 높은 상황은 다음과 같습니다.

- 서비스에는 시뮬레이션 스키마에서 지정한 하나 이상의 Amazon S3 버킷에 액세스할 권한이 없습니다. 앱 역할 권한 정책, Amazon S3 버킷 정책, Amazon S3 버킷 권한을 확인하여 `simspaceweaver.amazonaws.com`에 버킷에 액세스할 수 있는 올바른 권한이 있는지 확인합니다. 앱 역할 권한 정책 편집에 대한 자세한 정보는 에서 자동으로 SimSpace Weaver 생성하는 권한 섹션을 참조하세요.
- Amazon S3 버킷은 시뮬레이션과 다른 AWS 리전 에 있을 수 있습니다. 시뮬레이션 아티팩트의 Amazon S3 버킷은 시뮬레이션 AWS 리전 과 동일한에 있어야 합니다. Amazon S3 콘솔에서 AWS 리전 버킷이 무엇인지 확인합니다. Amazon S3 버킷이 다른에 있는 경우 시뮬레이션 AWS 리전 과 동일한에 있는 버킷을 AWS 리전선택합니다.

Docker 사용 시 시간과 관련된 문제

를 사용하고 Docker 앱 SimSpace Weaver SDK에서 스크립트를 실행하는 동안 시간 관련 오류가 발생하는 경우 Docker 가상 머신 클럭이 잘못되었기 때문일 수 있습니다. 컴퓨터가 Docker를 실행 중이었다가 절전 모드나 최대 절전 모드에서 다시 시작하는 경우 이 문제가 발생할 수 있습니다.

시도해 볼 수 있는 해결 방법

- Docker를 다시 시작합니다.

- Windows PowerShell에서 시간 동기화를 비활성화한 다음 다시 활성화합니다.

```
Get-VMIntegrationService -VMName DockerDesktopVM -Name "Time Synchronization" |
 Disable-VMIntegrationService
Get-VMIntegrationService -VMName DockerDesktopVM -Name "Time Synchronization" |
 Enable-VMIntegrationService
```

PathfindingSample 콘솔 클라이언트가 연결되지 않음

의 자습서에 설명된 PathfindingSample 시뮬레이션에 연결할 때 콘솔 클라이언트에서 다음 오류가 발생할 수 있습니다시작하기 SimSpace Weaver. 이 오류는 클라이언트가 제공된 통합 IP 주소 및 포트 번호로 ViewApp에 대한 네트워크 연결을 열 수 없기 때문에 발생합니다.

```
Fatal error in function nng_dial. Error code: 268435577. Error message: no link
```

의 시뮬레이션의 경우 AWS 클라우드

- 네트워크 연결이 제대로 작동하고 있나요? 작동해야 하는 다른 IP 주소나 웹 사이트에 연결할 수 있는지 확인합니다. 웹 브라우저가 캐시에서 웹 사이트를 로드하고 있지 않은지 확인합니다.
- 시뮬레이션이 실행 중인가요? ListSimulations API를 사용하여 시뮬레이션 상태를 가져올 수 있습니다. 자세한 내용은 사용자 지정 앱의 IP 주소 및 포트 번호 가져오기 단원을 참조하십시오. SimSpace Weaver 콘솔을 사용하여 시뮬레이션 상태를 확인할 수도 있습니다.
- 앱이 실행 중인가요? DescribeApp API를 사용하여 앱의 상태를 가져올 수 있습니다. 자세한 내용은 사용자 지정 앱의 IP 주소 및 포트 번호 가져오기 단원을 참조하십시오. SimSpace Weaver 콘솔을 사용하여 시뮬레이션 상태를 확인할 수도 있습니다.
- 앱이 실행 중인가요? DescribeApp API를 사용하여 앱의 상태를 가져올 수 있습니다. 자세한 내용은 사용자 지정 앱의 IP 주소 및 포트 번호 가져오기 단원을 참조하십시오. SimSpace Weaver 콘솔을 사용하여 시뮬레이션 상태를 확인할 수도 있습니다.
- 올바른 IP 주소와 포트 번호를 사용했나요? 인터넷을 통해 연결할 때는 ViewApp의 IP 주소와 Actual 포트 번호를 사용해야 합니다. DescribeApp API 출력의 EndpointInfo 블록에서 IP Address 및 Actual 포트 번호를 찾을 수 있습니다. 또한 SimSpace Weaver 콘솔을 사용하여 MyViewDomain 세부 정보 페이지에서 해당 IP 주소(URI)와 포트 번호(수집 포트)를 찾을 수 있습니다ViewApp.
- 네트워크 연결이 방화벽을 통과하고 있나요? 방화벽이 IP 주소 또는 포트 번호(또는 둘 다)에 대한 연결을 차단할 수 있습니다. 방화벽 설정을 확인하거나 방화벽 관리자에게 확인합니다.

로컬 시뮬레이션의 경우

- 루프백 주소(127.0.0.1)에 연결할 수 있나요? Windows에서 `ping` 명령줄 도구를 사용하는 경우 명령 프롬프트 창을 열고 127.0.0.1을 ping해 볼 수 있습니다. Ping을 종료하려면 Ctrl-C를 누릅니다.

  ```
  ping 127.0.0.1
  ```

  Example Ping 출력

  ```
  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:\>
  ```

  Ping에서 패킷이 손실되었다고 표시된 경우 다른 소프트웨어(예: 로컬 방화벽, 보안 설정 또는 맬웨어 방지 프로그램)가 연결을 차단하고 있을 수 있습니다.
- 앱이 실행 중인가요? 로컬 시뮬레이션은 각 앱에 대해 별도의 창으로 실행됩니다. 공간 앱 및 `ViewApp`의 창이 열려 있는지 확인합니다. 자세한 내용은 의 로컬 개발 SimSpace Weaver 단원을 참조하십시오.
- 올바른 IP 주소와 포트 번호를 사용했나요? `tcp://127.0.0.1:7000`은 로컬 시뮬레이션에 연결할 때 사용해야 합니다. 자세한 내용은 의 로컬 개발 SimSpace Weaver 단원을 참조하십시오.
- 연결을 차단할 수 있는 로컬 보안 소프트웨어가 있나요? 보안 설정, 로컬 방화벽 또는 맬웨어 방지 프로그램이 TCP 포트 `7000`에서 `127.0.0.1`에 대한 연결을 차단하고 있는지 확인합니다.

AWS CLI 가 인식하지 못함 **`simspaceweaver`**

AWS CLI 에서 알 수 없음을 암시하는 오류가 발생하면 다음 명령을 SimSpace Weaver실행합니다.

```
aws simspaceweaver help
```

다음 줄로 시작하고 사용 가능한 모든 선택 항목을 나열하는 오류가 발생하면가 이전 버전일 AWS CLI 수 있습니다.

```
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:
```

다음 명령을 실행하여의 버전을 확인합니다 AWS CLI.

```
aws --version
```

버전 번호가 2.9.19 이전인 경우 AWS CLI를 업데이트해야 합니다. 의 현재 버전 AWS CLI 은 2.9.19보다 이후입니다.

를 업데이트하려면 버전 2 사용 설명서의 최신 버전 설치 또는 업데이트를 AWS CLI AWS CLI참조하세요. AWS Command Line Interface

SimSpace Weaver 시뮬레이션 스키마 참조

SimSpace Weaver 는 YAML 파일을 사용하여 시뮬레이션의 속성을 구성합니다. 이 파일을 시뮬레이션 스키마(또는 그냥 스키마)라고 합니다. SimSpace Weaver 앱 SDK에 포함된 샘플 시뮬레이션에는 자체 시뮬레이션을 위해 복사하고 편집할 수 있는 스키마가 포함되어 있습니다.

주제

전체 스키마의 예

다음 예제에서는 시뮬레이션을 설명하는 YAML SimSpace Weaver 형식의 텍스트 파일을 보여줍니다. 이 예제에는 속성에 대한 더미 값이 포함되어 있습니다. 파일 형식은 파일에 지정된 `sdk_version` 값에 따라 달라집니다. 속성 및 유효한 값에 대한 전체 설명은 스키마 형식 섹션을 참조하세요.

```
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"]
```

스키마 형식

다음 예제에서는 스키마의 전체 구조를 보여줍니다. 상위-하위 관계가 동일하기만 하면 각 스키마 수준의 속성 순서는 중요하지 않습니다. 배열의 요소 순서는 중요합니다.

```
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
```

Sections

SDK 버전

`sdk_version` 섹션(필수)은이 스키마를 지원하는 SimSpace Weaver 앱 SDK의 버전을 식별합니다. 유효한 값: `1.17`, `1.16`, `1.15`, `1.14`, `1.13`, `1.12`

> **Important**
>
> `sdk_version`의 값에는 메이저 버전 번호와 첫 번째 마이너 버전 번호만 포함됩니다. 예를 들어, `1.12` 값은 `1.12.0`, `1.12.1`, `1.12.2` 등의 모든 버전 `1.12.x`를 지정합니다.

```
sdk_version: "1.17"
```

시뮬레이션 속성

`simulation_properties` 섹션(필수)은 시뮬레이션의 다양한 속성을 지정합니다. 이 섹션을 사용하여 로깅을 구성하고 기본 컨테이너 이미지를 지정할 수 있습니다. 이 섹션은 로깅을 구성하지 않거나 기본 컨테이너 이미지를 지정하도록 선택한 경우에도 필요합니다.

```
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"
```

속성

`log_destination_resource_name`

로그를 SimSpace Weaver 쓸 리소스를 지정합니다.

필수: 아니요. 이 속성이 포함되지 않은 경우 SimSpace Weaver 는 시뮬레이션에 대한 로그를 작성하지 않습니다.

유형: 문자열

유효한 값:

- CloudWatch Logs 로그 그룹의 이름(예: `MySimulationLogs`)
- CloudWatch Logs 로그 그룹의 Amazon 리소스 이름(ARN)(예: `arn:aws:logs:us-west-2:111122223333:log-group/MySimulationLogs`)

> **Note**
>
> SimSpace Weaver 는 시뮬레이션과 동일한 계정 및 AWS 리전 의 로그 대상만 지원합니다.

`log_destination_service`

ARN이 아닌 `logging_destination resource_name`을 지정하는 경우 로깅 대상 리소스의 유형을 나타냅니다.

필수: `log_destination_resource_name`이 지정되어 있고 ARN이 아닌 경우 이 속성을 지정해야 합니다. `log_destination_resource_name`이 지정되어 있지 않고 ARN인 경우 이 속성을 지정할 수 없습니다.

유형: 문자열

유효한 값:

- `logs`: 로그 대상 리소스는 로그 그룹입니다.

`default_entity_index_key_type`

시뮬레이션 엔터티의 인덱스 키 필드에 대한 데이터 유형을 지정합니다.

필수 항목 여부: 예

유형: 문자열

유효한 값: `Vector3<f32>`

`default_image`

시뮬레이션의 기본 컨테이너 이미지를 지정합니다(`1.13` 및 `1.12` 버전에서는 지원되지 않음). 이 속성을 지정하면 `image`를 지정하지 않은 도메인에서 `default_image`를 사용합니다.

필수 항목 여부: 아니요

유형: 문자열

유효한 값:

- Amazon Elastic Container Registry(Amazon ECR)에 있는 리포지토리의 URI(예: `111122223333.dkr.ecr.us-west-2.amazonaws.com/my-ecr-repository:latest`)

작업자

`workers` 섹션(필수)은 작업자 그룹(작업자의 그룹)의 구성을 지정합니다. SimSpace Weaver 는 이 정보와 함께 `placement_constraints`를 사용하여 시뮬레이션의 기본 인프라를 구성합니다. 현재 하나의 작업자 그룹만 지원됩니다.

작업자 그룹의 속성을 지정하려면 *worker-group-name*을 원하는 이름으로 바꿉니다. 이름은 3~64자 길이여야 하며 A~Z, a~z, 0~9, _~(하이픈) 문자를 포함할 수 있습니다. 이름 뒤에 작업자 그룹의 속성을 지정합니다.

```
workers:
  worker-group-name:
    type: "sim.c5.24xlarge"
    desired: number-of-workers
```

속성

`type`

작업자 유형을 지정합니다.

필수 항목 여부: 예

유형: 문자열

유효한 값: `sim.c5.24xlarge`

`desired`

이 작업자 그룹에 원하는 작업자 수를 지정합니다.

필수 항목 여부: 예

유형: 정수

유효한 값: 1~3. 시뮬레이션의 작업자 수에 대한 Service Quotas(한도)에 따라 이 속성의 최대값이 결정됩니다. 예를 들어, Service Quotas가 2인 경우 이 속성의 최대값은 2입니다. Service Quotas 증가를 요청할 수도 있습니다. 자세한 내용은 SimSpace Weaver 엔드포인트 및 할당량 단원을 참조하십시오.

클럭

`clock` 섹션(필수)에서는 시뮬레이션 클럭의 속성을 지정합니다.

```
clock:
  tick_rate: tick-rate
```

속성

`tick_rate`

클럭이 앱에 게시하는 초당 틱 수를 지정합니다.

필수 항목 여부: 예

Type:

- 버전 *1.14* 및 *1.15*: 문자열
- 버전 *1.13* 및 *1.12*: 정수

유효한 값:

- 버전 *1.14* 및 *1.15*: `"10" | "15" | "30" | "unlimited"`
  - `"unlimited"`: 클럭은 모든 앱이 현재 틱에 대한 커밋 작업을 완료하는 즉시 다음 틱을 전송합니다.
- 버전 *1.13* 및 *1.12*: `10 | 15 | 30`

파티셔닝 전략

`partitioning_strategies` 섹션(필수)은 공간 도메인의 파티션 구성을 지정합니다.

> **Note**
>
> SimSpace Weaver 는 1개의 파티셔닝 전략만 지원합니다.

파티셔닝 전략의 속성을 지정하려면 *`partitioning-strategy-name`*을 원하는 이름으로 바꿉니다. 이름은 3~64자 길이여야 하며 A~Z, a~z, 0~9, _~(하이픈) 문자를 포함할 수 있습니다. 이름 뒤에 파티셔닝 전략의 속성을 지정합니다.

```
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
```

속성

topology

이 분할 전략의 토폴로지(파티션 정렬 스키마)를 지정합니다.

필수 항목 여부: 예

유형: 문자열

유효한 값: "Grid"

aabb_bounds

시뮬레이션을 위한 주축 정렬 경계 상자 (AABB)의 경계를 지정합니다. 경계를 각 축(x 및 y)의 최소값과 최대값(순서대로)을 설명하는 2 요소 순서 배열로 지정합니다.

필수 항목 여부: 조건부. 토폴로지가 "Grid"로 설정된 경우 이 속성은 필수이며 지정만 가능합니다.

유형: Float 배열(각 축용)

유효한 값: -3.4028235e38~3.4028235e38

grid_placement_groups

그리드 토폴로지에서 각 축(x 및 y)의 배치 그룹 수를 지정합니다. 배치 그룹은 공간적으로 인접한 파티션(동일한 도메인 내)의 모음입니다.

필수 항목 여부: 조건부. 토폴로지가 "Grid"로 설정된 경우 이 속성은 필수이며 지정만 가능합니다. 배치 그룹 구성을 지정하지 않으면 SimSpace Weaver 가 계산합니다. 배치 그룹 구성 없이 분할 전략을 사용하는 모든 도메인은 grid_partition을 지정해야 합니다(공간 도메인 파티셔닝 전략 참조).

유형: 정수(각 축용)

유효한 값: 1~20. x* y는 원하는 작업자 수와 같게 설정하는 것이 좋습니다. 그렇지 않으면 SimSpace Weaver 는 사용 가능한 작업자 간에 배치 그룹의 균형을 맞추려고 시도합니다.

도메인

domains 섹션(필수)은 각 도메인의 속성을 지정합니다. 모든 시뮬레이션에는 하나 이상의 공간 영역 섹션이 있어야 합니다. 추가 도메인에 대해 여러 섹션을 생성할 수 있습니다. 각 도메인 유형에는 고유한 구성 형식이 있습니다.

> **Important**
>
> 1.13 및 1.12 버전에서는 다중 공간 도메인을 지원하지 않습니다.

> **Important**
>
> SimSpace Weaver 는 각 시뮬레이션에 대해 최대 5개의 도메인을 지원합니다. 여기에는 모든 공간, 사용자 지정 및 서비스 도메인이 포함됩니다.

```
domains:
  domain-name:
    domain-configuration
  domain-name:
    domain-configuration
  ...
```

도메인 구성

- 공간 도메인 구성
- 사용자 지정 도메인 구성
- 서비스 도메인 구성

공간 도메인 구성

공간 도메인의 속성을 지정하려면 *spatial-domain-name*을 원하는 이름으로 바꿉니다. 이름은 3~64자 길이여야 하며 A~Z, a~z, 0~9, _~(하이픈) 문자를 포함할 수 있습니다. 이름 뒤에 공간 도메인의 속성을 지정합니다.

```
  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"
```

공간 도메인 파티셔닝 전략

`launch_apps_by_partitioning_strategy` 섹션(필수)은 시뮬레이션 공간의 분할 전략 및 크기(파티션 수)를 지정합니다.

```
    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
```

속성

`partitioning_strategy`

이 공간 도메인의 분할 전략을 지정합니다.

필수 항목 여부: 예

유형: 문자열

유효한 값: 이 속성의 값은 `partitioning_strategies` 섹션에 정의된 파티션 전략의 이름과 일치해야 합니다. 자세한 내용은 파티셔닝 전략 단원을 참조하십시오.

`grid_partition`

그리드 토폴로지에서 각 축(x 및 y)과 함께 파티션 수를 지정합니다. 이 크기는 이 도메인의 전체 시뮬레이션 공간을 나타냅니다.

필수 항목 여부: 조건부. 토폴로지가 `"Grid"`로 설정된 경우 이 속성은 지정만 가능합니다. 이 속성은 이 도메인에 지정된 파티셔닝 전략의 `grid_placement_groups` 속성에 따라 달라집니다.

- 이 속성은 이 도메인의 파티셔닝 전략이 `grid_placement_groups` 구성을 지정하지 않는 경우 필요합니다.

- grid_placement_groups 구성이 있지만 grid_partition을 지정하지 않은 경우 SimSpace Weaver 는 grid_placment_groups 구성과 동일한 크기를 사용합니다.
- grid_placement_groups 및 grid_partition 모두 지정하는 경우 grid_partition의 크기는 grid_placement_groups 크기의 배수여야 합니다. 예를 들어 grid_placement_groups 크기가 2x2인 경우 유효한 grid_partition 크기는 2x2, 4x4, 6x6, 8x8, 10x10입니다.

유형: 정수(각 축용)

유효한 값: 1~20

공간 앱 구성

app_config 섹션(필수)은 이 도메인의 앱에 대한 패키지, 시작 구성 및 리소스 요구 사항을 지정합니다.

```
app_config:
  package: "app-package-s3-uri"
  launch_command: ["app-launch-command", "parameter1", ...]
  required_resource_units:
    compute: app-resource-units
```

속성

package

앱 실행 파일/바이너리가 포함된 패키지(zip 파일)를 지정합니다. 패키지는 Amazon S3 버킷에 저장해야 합니다. zip 파일 형식만 지원됩니다.

필수 항목 여부: 예

유형: 문자열

유효한 값: Amazon S3 버킷에서 패키지의 Amazon S3 URI입니다. 예: s3://weaver-myproject-111122223333-app-zips-us-west-2/MySpatialApp.zip.

launch_command

앱을 시작하기 위한 실행/바이너리 파일 이름 및 명령줄 파라미터를 지정합니다. 각 명령줄 문자열 토큰은 배열의 한 요소입니다.

필수 항목 여부: 예

유형: 문자열 배열

`required_resource_units`

SimSpace Weaver 가 이 앱의 각 인스턴스에 할당해야 하는 리소스 단위의 수를 지정합니다. 리소스 단위는 작업자에 있는 고정된 양의 가상 중앙 처리 장치 (vCPUs) 및 임의 액세스 메모리 (RAM) 입니다. 리소스 단위에 대한 자세한 내용은 엔드포인트 및 Service Quotas 섹션을 참조하세요. `compute` 속성은 작업자의 `compute` 패밀리를 위한 자원 단위 할당을 지정하며, 현재 유일하게 유효한 할당 유형입니다.

필수 항목 여부: 예

유형: 정수

유효한 값: 1~4

사용자 지정 컨테이너 이미지

`image` 속성(선택 사항)은가이 도메인에서 앱을 실행하는 데 SimSpace Weaver 사용하는 컨테이너 이미지의 위치를 지정합니다(버전 `1.13` 및 에서는 지원되지 않음`1.12`). 이미지를 포함하는 Amazon Elastic Container Registry(Amazon ECR) 리포지토리에 URI를 제공합니다. 이 속성이 지정되지 않았지만 `default_image`가 최상위 `simulation_properties` 섹션에 지정된 경우 이 도메인의 앱은 `default_image`를 사용합니다. 자세한 내용은 사용자 지정 컨테이너 단원을 참조하십시오.

```
    image: "ecr-repository-uri"
```

속성

`image`

이 도메인에서 앱을 실행할 컨테이너 이미지의 위치를 지정합니다.

필수 항목 여부: 아니요

유형: 문자열

유효한 값:

- Amazon Elastic Container Registry(Amazon ECR)에 있는 리포지토리의 URI(예: `111122223333.dkr.ecr.us-west-2.amazonaws.com/my-ecr-repository:latest`)

사용자 지정 도메인 구성

사용자 지정 도메인의 속성을 지정하려면 *custom-domain-name*을 원하는 이름으로 바꿉니다. 이름은 3~64자 길이여야 하며 A~Z, a~z, 0~9, _~(하이픈) 문자를 포함할 수 있습니다. 이름 뒤에 사용자 지정 도메인의 속성을 지정합니다. 각 사용자 지정 도메인에 대해 이 절차를 반복합니다.

```
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"
```

속성

`launch_apps_via_start_app_call`

이 속성은 StartApp API를 사용하여 사용자 지정 앱을 시작하는 데 필요합니다.

필수 항목 여부: 예

유형: N/A

유효한 값: {}

사용자 지정 앱 구성

`app_config section` 섹션(필수)은 이 사용자 지정 도메인의 앱에 대한 패키지, 시작 구성, 리소스 요구 사항 및 네트워크 포트를 지정합니다.

```
  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, ...]
```

속성

`package`

앱 실행 파일/바이너리가 포함된 패키지(zip 파일)를 지정합니다. 패키지는 Amazon S3 버킷에 저장해야 합니다. zip 파일 형식만 지원됩니다.

필수 항목 여부: 예

유형: 문자열

유효한 값: Amazon S3 버킷에서 패키지의 Amazon S3 URI입니다. 예: `s3://weaver-myproject-111122223333-app-zips-us-west-2/MyCustomApp.zip`.

`launch_command`

앱을 시작하기 위한 실행/바이너리 파일 이름 및 명령줄 파라미터를 지정합니다. 각 명령줄 문자열 토큰은 배열의 한 요소입니다.

필수 항목 여부: 예

유형: 문자열 배열

`required_resource_units`

SimSpace Weaver 가 이 앱의 각 인스턴스에 할당해야 하는 리소스 단위의 수를 지정합니다. 리소스 단위는 작업자에 있는 고정된 양의 가상 중앙 처리 장치 (vCPUs) 및 임의 액세스 메모리 (RAM) 입니다. 리소스 단위에 대한 자세한 내용은 엔드포인트 및 Service Quotas 섹션을 참조하세요. `compute` 속성은 작업자의 `compute` 패밀리를 위한 자원 단위 할당을 지정하며, 현재 유일하게 유효한 할당 유형입니다.

필수 항목 여부: 예

유형: 정수

유효한 값: `1~4`

`endpoint_config`

이 도메인에 있는 앱의 네트워크 엔드포인트를 지정합니다. 의 값은 수신 클라이언트 연결을 위해 사용자 지정 앱이 바인딩하는 포트를 `ingress_ports` 지정합니다.는 동적으로 할당된 포트를 지정된 수신 포트에 SimSpace Weaver 매핑합니다. 수집 포트는 모두 TCP와 UDP입니다. DescribeApp API를 사용하여 클라이언트를 연결하는 데 필요한 실제 포트 번호를 찾을 수 있습니다.

필수: 아니요. 엔드포인트 구성을 지정하지 않으면 이 도메인의 사용자 지정 앱에 네트워크 엔드포인트가 없습니다.

유형: 정수 배열

유효한 값: 1024~49152. 값은 고유해야 합니다.

사용자 지정 컨테이너 이미지

image 속성(선택 사항)은가이 도메인에서 앱을 실행하는 데 SimSpace Weaver 사용하는 컨테이너 이미지의 위치를 지정합니다(버전 1.13 및 에서는 지원되지 않음1.12). 이미지를 포함하는 Amazon Elastic Container Registry(Amazon ECR) 리포지토리에 URI를 제공합니다. 이 속성이 지정되지 않았지만 default_image가 최상위 simulation_properties 섹션에 지정된 경우 이 도메인의 앱은 default_image를 사용합니다. 자세한 내용은 사용자 지정 컨테이너 단원을 참조하십시오.

```
    image: "ecr-repository-uri"
```

속성

image

이 도메인에서 앱을 실행할 컨테이너 이미지의 위치를 지정합니다.

필수 항목 여부: 아니요

유형: 문자열

유효한 값:

- Amazon Elastic Container Registry(Amazon ECR)에 있는 리포지토리의 URI(예: 111122223333.dkr.ecr.us-west-2.amazonaws.com/my-ecr-repository:latest)

서비스 도메인 구성

서비스 도메인의 속성을 지정하려면 *service-domain-name*을 원하는 이름으로 바꿉니다. 이름은 3~64자 길이여야 하며 A~Z, a~z, 0~9, _~(하이픈) 문자를 포함할 수 있습니다. 이름 뒤에 서비스 도메인의 속성을 지정합니다. 각 사용자 지정 도메인에 대해 이 절차를 반복합니다.

```
  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"
```

작업자당 앱 시작

launch_apps_per_worker 섹션(필수)은 이 구성이 서비스 도메인 구성임을 나타내며 작업자당 시작할 서비스 앱 수를 지정합니다.

```
    launch_apps_per_worker:
      count: number-of-apps-to-launch
```

속성

count

이 속성은 작업자당 시작할 서비스 앱 수를 지정합니다.

필수 항목 여부: 예

유형: 정수

유효한 값: {} | 1 | 2. {} 값은 1 기본값을 지정합니다.

서비스 앱 구성

app_config section 섹션(필수)은 이 서비스 도메인의 앱에 대한 패키지, 시작 구성, 리소스 요구 사항 및 네트워크 포트를 지정합니다.

```
    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, ...]
```

속성

`package`

앱 실행 파일/바이너리가 포함된 패키지(zip 파일)를 지정합니다. 패키지는 Amazon S3 버킷에 저장해야 합니다. zip 파일 형식만 지원됩니다.

필수 항목 여부: 예

유형: 문자열

유효한 값: Amazon S3 버킷에서 패키지의 Amazon S3 URI입니다. 예: `s3://weaver-myproject-111122223333-app-zips-us-west-2/MyServiceApp.zip`.

`launch_command`

앱을 시작하기 위한 실행/바이너리 파일 이름 및 명령줄 파라미터를 지정합니다. 각 명령줄 문자열 토큰은 배열의 한 요소입니다.

필수 항목 여부: 예

유형: 문자열 배열

`required_resource_units`

SimSpace Weaver 가 이 앱의 각 인스턴스에 할당해야 하는 리소스 단위의 수를 지정합니다. 리소스 단위는 작업자에 있는 고정된 양의 가상 중앙 처리 장치 (vCPUs) 및 임의 액세스 메모리 (RAM) 입니다. 리소스 단위에 대한 자세한 내용은 엔드포인트 및 Service Quotas 섹션을 참조하세요. `compute` 속성은 작업자의 `compute` 패밀리를 위한 자원 단위 할당을 지정하며, 현재 유일하게 유효한 할당 유형입니다.

필수 항목 여부: 예

유형: 정수

유효한 값: `1~4`

`endpoint_config`

이 도메인에 있는 앱의 네트워크 엔드포인트를 지정합니다. `ingress_ports` 값은 들어오는 클라이언트 연결을 위해 서비스 앱이 바인딩하는 포트를 지정합니다. SimSpace Weaver 는 동적으로 할당된 포트를 지정된 수집 포트에 매핑합니다. 수집 포트는 모두 TCP와 UDP입니다. DescribeApp API를 사용하여 클라이언트를 연결하는 데 필요한 실제 포트 번호를 찾을 수 있습니다.

필수: 아니요. 엔드포인트 구성을 지정하지 않으면 이 도메인의 서비스 앱에 네트워크 엔드포인트가 없습니다.

유형: 정수 배열

유효한 값: 1024~49152. 값은 고유해야 합니다.

사용자 지정 컨테이너 이미지

image 속성(선택 사항)은가이 도메인에서 앱을 실행하는 데 SimSpace Weaver 사용하는 컨테이너 이미지의 위치를 지정합니다(버전 1.13 및 에서는 지원되지 않음1.12). 이미지를 포함하는 Amazon Elastic Container Registry(Amazon ECR) 리포지토리에 URI를 제공합니다. 이 속성이 지정되지 않았지만 default_image가 최상위 simulation_properties 섹션에 지정된 경우 이 도메인의 앱은 default_image를 사용합니다. 자세한 내용은 사용자 지정 컨테이너 단원을 참조하십시오.

```
    image: "ecr-repository-uri"
```

속성

image

이 도메인에서 앱을 실행할 컨테이너 이미지의 위치를 지정합니다.

필수 항목 여부: 아니요

유형: 문자열

유효한 값:

- Amazon Elastic Container Registry(Amazon ECR)에 있는 리포지토리의 URI(예: 111122223333.dkr.ecr.us-west-2.amazonaws.com/my-ecr-repository:latest)

배치 제약 조건

placement_constraints 섹션(선택 사항)은 SimSpace Weaver 가 동일한 작업자에 함께 배치해야 하는 공간 도메인을 지정합니다. 자세한 내용은 공간 도메인 구성 단원을 참조하십시오.

> **Important**
>
> 1.13 및 1.12 버전에서는 placement_constraints를 지원하지 않습니다.

```
placement_constraints:
  - placed_together: ["spatial-domain-name", "spatial-domain-name", ...]
    on_workers: ["worker-group-name"]
```

속성

`placed_together`

함께 배치 SimSpace Weaver 해야 하는 공간 도메인을 지정합니다.

필수 항목 여부: 예

유형: 문자열 배열

유효한 값: 스키마에 지정된 공간 도메인의 이름

`on_workers`

도메인을 배치 SimSpace Weaver 해야 하는 작업자 그룹을 지정합니다.

필수 항목 여부: 예

유형: 1요소 문자열 배열

유효한 값: 스키마에 지정된 작업자 그룹의 이름

SimSpace Weaver API 참조

SimSpace Weaver 에는 2개의 서로 다른 애플리케이션 프로그래밍 인터페이스(APIs.

- 서비스 API - 이러한 API는 서비스 및 서비스 리소스(예: 시뮬레이션, 클럭, 앱)를 제어합니다. 기본 소프트웨어 개발 키트(SDK)의 AWS 일부이며 AWS 명령줄 인터페이스(CLI)를 사용하여 호출할 수 있습니다. 서비스 API에 대한 자세한 내용은 SimSpace Weaver API 참조를 참조하세요.
- 앱 SDK API - 이러한 API는 시뮬레이션 내 데이터를 제어합니다. 앱 코드에서 이를 사용하여 항목 필드 데이터 읽기 및 쓰기, 구독 관련 작업, 시뮬레이션의 이벤트 모니터링 등의 작업을 수행할 수 있습니다. 자세한 내용은 SimSpace Weaver 앱 SDK 폴더의 앱 SDK 설명서를 참조하세요. *sdk-folder*\SimSpaceWeaverAppSdk\documentation

> **Note**
>
> *sdk-folder*는 SimSpaceWeaverAppSdkDistributable 패키지의 압축을 푼 폴더입니다.

AWS SimSpace Weaver 버전

우리는 AWS SimSpace Weaver를 지속적으로 개선하고 있습니다. 새로운 기능 및 기능 업데이트를 활용하려면 새 버전을 릴리스할 때 최신 SimSpace Weaver 앱 SDK를 다운로드해야 합니다. 새 버전으로 기존 시뮬레이션을 실행하려면 스키마와 코드를 업데이트한 다음 시뮬레이션의 새 인스턴스를 시작해야 할 수 있습니다. 업그레이드할 필요가 없으며 이전 버전으로 기존 시뮬레이션을 계속 실행할 수 있습니다. 이 페이지에서 버전 간 차이점을 확인할 수 있습니다. 현재 모든 버전이 지원됩니다.

Important

AWS SimSpace Weaver 사용 설명서의 최신 버전은 최신 버전의 서비스만 다룹니다. 기본 설명서 랜딩 페이지에서 제공되는 AWS SimSpace Weaver 가이드 카탈로그에서 이전 버전에 대한 설명서를 찾을 수 있습니다. https://docs.aws.amazon.com/simspaceweaver

최신 버전

최신 버전: 1.17.0

현재 버전 검색 방법

SimSpace Weaver 앱 SDK로 시뮬레이션을 생성한 경우 `create-project` 스크립트는 SDK 라이브러리 버전을의 하위 디렉터리로 다운로드했습니다*`sdk-folder`*. SDK 라이브러리가 포함된 하위 디렉터리의 이름에는 SDK 버전 번호 `SimSpaceWeaverAppSdk-`*`sdk-version`*이 포함되어 있습니다. 예를 들어 버전 1.16.0의 라이브러리는에 있습니다`SimSpaceWeaverAppSdk-1.16.0`.

의 텍스트 파일에서 SimSpace Weaver 앱 SDK 배포 가능 패키지의 버전을 찾을 수도 `app_sdk_distributable_version.txt` 있습니다*`sdk-folder`*.

최신 버전 다운로드

다음 링크 중 하나를 사용하여 최신 버전을 다운로드합니다.

- 전체 앱 SDK 배포 가능 패키지
- 앱 SDK 라이브러리만

의 SimSpace Weaver 콘솔에서 전체 SimSpace Weaver 앱 SDK 배포 가능 패키지를 다운로드할 수도 있습니다 AWS Management Console. 탐색 창에서 앱 SDK 다운로드를 선택합니다.

> **Warning**
>
> AWS CLI 를 사용하여 SimSpace Weaver 앱 SDK 배포 가능 패키지로 보이는 것을 다운로드하지 마십시오. 이 페이지의 다운로드 링크 또는 콘솔의 다운로드 링크만 사용합니다. 다른 다운로드 방법 또는 위치는 지원되지 않으며 더 이상 사용되지 않거나 올바르지 않거나 악성 코드가 포함되어 있을 수 있습니다.

앱 SDK 다운로드 문제 해결

우리는 앱 SDK .zip 파일을 배포하기 위해 Amazon CloudFront(CloudFront)를 사용합니다. 일부의 경우 다음과 같은 상황이 발생할 수 있습니다.

- 다운로드한 패키지가 최신 버전이 아님
  - 다운로드한 .zip 파일에 최신 버전이 없는 경우 CloudFront 엣지 로케이션의 캐시가 아직 업데이트되지 않았을 수 있습니다. 24시간 이후에 다시 다운로드합니다.
- 다운로드 링크를 사용할 때 HTTP 4xx 또는 5xx 오류 발생
  - 24시간 이후에 다시 시도합니다. 같은 오류가 발생하는 경우 SimSpace Weaver 콘솔 하단에 있는 피드백 링크를 사용하여 문제를 신고하세요. 피드백 유형으로 문제 신고를 선택합니다.
- 브라우저가 페이지를 로드할 수 없다고 보고함
  - 로컬 네트워크 또는 브라우저 구성 문제가 있을 수 있습니다. 다른 페이지를 로드할 수 있는지 확인합니다. 브라우저 캐시를 지우고 다시 시도합니다. 다운로드 URL을 차단할 수 있는 방화벽 규칙이 적용되어 있지 않은지 확인합니다.
- 파일을 저장하려고 하면 오류가 발생함
  - 파일을 저장할 수 있는 올바른 권한이 적용되도록 로컬 파일 시스템 권한을 확인합니다.
- 브라우저에 AccessDenied가 표시됨
  - 브라우저에 URL을 수동으로 입력한 경우 정확한지 확인합니다. 다운로드 링크를 사용한 경우 브라우저의 URL에 방해가 되지 않았는지 확인하고 링크를 다시 사용합니다.

최신 버전 설치

최신 버전 설치

1. 최신 버전 다운로드
2. 폴더에 SimSpaceWeaverAppSdkDistributable.zip의 압축을 풉니다.
3. 압축을 푼 최신 버전 SimSpace Weaver 앱 SDK 폴더`python setup.py`에서를 실행합니다.
4. 이전 버전 대신 압축을 푼 최신 버전 SimSpace Weaver 앱 SDK 폴더를 사용합니다.

서비스 버전

버전	Notes	릴리스 날짜	설명서	앱 SDK 다운로드
1.17.0	SimSpace Weaver 앱 SDK 배포 가능 패키지의 주요 변경 사항 • Windows 배치 및 Linux Bash 스크립트를 Python 기반 스크립트로 대체했습니다. 따라서 Python SDK를 사용하지 않더라도(또는 사용하려는 경우에도) 스크립트와 샘플을 사용하려면 Python 3.9가 필요합니다. • 이 릴리스는 Amazon Linux 2에 대한 지원을 늘립니다.	2024년 4월 17일	이 설명서는	• 전체 패키지 • 라이브러리 전용 문제 해결을 참조하세요.

버전	Notes	릴리스 날짜	설명서	앱 SDK 다운로드
	• 에서 여러 버그를 수정했습니다SimSpace Weaver Local. 자세한 내용은 릴리스 정보를 참조하세요. 버그 수정 • 원격 작업자 간의 이전을 완료하지 않은 경우 엔터티가 소유되지 않도록 하는 버그를 수정했습니다.			
1.16.0	새로운 기능: • 이제 SimSpace Weaver 앱 SDK에서 메시징 APIs를 사용하여 앱 간에 메시지를 보내고 받을 수 있습니다. 이 기능은 C++, Python, Unity 및 Unreal Engine 5 통합에 사용할 수 있습니다.	2024년 2월 12일	AWS SimSpace Weaver 가이드 카탈로그를 참조하세요.	• 전체 패키지 • 라이브러리 전용 문제 해결을 참조하세요.

버전	Notes	릴리스 날짜	설명서	앱 SDK 다운로드
1.15.3	SimSpace Weaver Local 업데이트: • AWS 클라우드를 위한 개발에 더 적합하도록 SimSpace Weaver Local을 변경했습니다. 이러한 변경 사항은 SimSpace Weaver Local을 위한 C++, Python, Unity, Unreal Engine 프로젝트 및 워크플로에 영향을 미칩니다.	2023년 12월 4일	AWS SimSpace Weaver 가이드 카탈로그를 참조하세요.	다운로드할 수 없음
1.15.2	앱 SDK 배포 가능 패키지 업데이트: • cmake의 특정 필수 버전을 사용하도록 `Dockerfile` 을 업데이트했습니다. 이렇게 변경하지 않으면 Docker 컨테이너 빌드가 실패할 수 있습니다.	2023년 11월 2일	AWS SimSpace Weaver 가이드 카탈로그를 참조하세요.	다운로드할 수 없음

버전	Notes	릴리스 날짜	설명서	앱 SDK 다운로드
1.15.1	기능 업데이트: • Python SDK: 이 릴리스에서는 AWS 클라우드에서 Python 기반 시뮬레이션이 실패하던 문제가 수정되었습니다.	2023년 9월 22일	AWS SimSpace Weaver 가이드 카탈로그를 참조하세요.	다운로드할 수 없음
1.15.0	새로운 기능: • Python SDK: 이제 Python에서 시뮬레이션을 배포할 수 있습니다. SimSpace Weaver 앱 SDK 배포 가능 패키지에는 샘플 Python 프로젝트 및 Python 보기 클라이언트용 템플릿이 포함되어 있습니다.	2023년 8월 31일	AWS SimSpace Weaver 가이드 카탈로그를 참조하세요.	다운로드할 수 없음

버전	Notes	릴리스 날짜	설명서	앱 SDK 다운로드
1.14.0	새로운 기능: • 사용자 지정 컨테이너: Amazon Linux 2(AL2) 기반 컨테이너 이미지를 직접 생성하여 Amazon Elastic Container Registry(Amazon ECR)에 저장한 다음, 이를 사용하여 AWS 클라우드에서 SimSpace Weaver 앱을 실행할 수 있습니다. • 다중 공간 도메인: 시뮬레이션에서 1개 이상의 공간 도메인을 생성합니다. 시뮬레이션 로직을 모두 하나의 공간 앱으로 결합하는 대신 분리합니다. 해당 요구 사항에 따라 공간 도메인에 다양한 리소스를 할당합니다. • 무제한 틱 속도: 코드를 실행할 수 있는 속도만큼 빠	2023년 7월 26일	AWS SimSpace Weaver 가이드 카탈로그를 참조하세요.	다운로드할 수 없음

버전	Notes	릴리스 날짜	설명서	앱 SDK 다운로드
	르게 시뮬레이션을 실행할 수 있습니다. 모든 앱이 현재 틱에 대한 커밋 작업을 완료하는 즉시 다음 틱을 전송하도록 시뮬레이션의 클럭을 설정합니다. SimSpace Weaver 앱 SDK: • `tick_rate` 의 값은 이제 문자열입니다. 값에는 큰따옴표(")가 포함되어야 합니다. 이전 버전의 틱 속도는 여전히 정수입니다.			
1.13.1	SimSpace Weaver 앱 SDK: • 기능 업데이트: 이제 `PathfindingSampleUnreal` 템플릿에서 프로젝트 생성이 제대로 작동합니다.	2023년 6월 7일	AWS SimSpace Weaver 가이드 카탈로그를 참조하세요.	다운로드할 수 없음

버전	Notes	릴리스 날짜	설명서	앱 SDK 다운로드
1.13.0	SimSpace Weaver 서비스 APIs: • • 새로운 CreateSnapshot 작업 • StartSimulation 작업 변경 사항: • 스냅샷에서 시작하는 SnapshotS3Location 파라미터를 추가했습니다. • 이제 SchemaS3Location 파라미터는 선택 항목입니다. • DescribeSimulation 출력 변경 사항: • SchemaError 는 더 이상 사용되지 않습니다. • StartError 필드를	2023년 4월 29일	AWS SimSpace Weaver 가이드 카탈로그를 참조하세요.	다운로드할 수 없음

버전	Notes	릴리스 날짜	설명서	앱 SDK 다운로드
	추가했습니다. • SnapshotS3Location 필드를 추가했습니다. • SNAPSHOT_IN_PROGRESS 시뮬레이션 상태를 추가했습니다. • 새 S3Destination 데이터 유형 SimSpace Weaver 콘솔: • 스냅샷을 생성하는 새로운 기능입니다. • 스냅샷에서 시뮬레이션을 시작하는 기능입니다. SimSpace Weaver 앱 SDK: • 스냅샷을 지원하는 새 스크립트 • create-snapshot- *project-name* .bat			

버전	Notes	릴리스 날짜	설명서	앱 SDK 다운로드
	• start-from-snapshot- *project-name* .bat • quick-start-from-snapshot- *pro name* - cli.bat • list-snapshots- *proje name* .bat • 이제 플젝트는 프로젝트당 단일 Amazon S3 버킷(weaver -*lowercase -project-name* -*account-number* -*region*)을 사용합니다.			

버전	Notes	릴리스 날짜	설명서	앱 SDK 다운로드
1.12.3	SimSpace Weaver 앱 SDK: • 이제 다음 스크립트가 `--maximum-duration` 파라미터를 지원합니다. • `quick-start-`*`project-name`*`-cli.bat` • `quick-start-`*`project-name`*`-cli.sh` • `start-simulation-`*`proj`*`name`*`.bat` • `start-simulation-`*`proj`*`name`*`.sh` • `run-`*`project-name`*`.bat` • `run-`*`project-name`*`.sh`	2023년 3월 27일	AWS SimSpace Weaver 가이드 카탈로그를 참조하세요.	다운로드할 수 없음

버전	Notes	릴리스 날짜	설명서	앱 SDK 다운로드
1.12.2	SimSpace Weaver 앱 SDK: • 버그 수정: 이제 `docker-create-image.bat` 가 제대로 실행됩니다.	2023년 3월 1일	AWS SimSpace Weaver 가이드 카탈로그를 참조하세요.	다운로드할 수 없음
1.12.1	SimSpace Weaver 앱 SDK: • 이제 스크립트는 AWS 인증에 사용할 AWS CLI 프로필을 수락합니다. • 이제 스크립트가 AWS IAM Identity Center AWS 인증을 지원합니다. SimSpace Weaver Local: • 버그 수정: 이제 `Api::BeginUpdateWillBlock` 은 모든 공간 앱이 시뮬레이션에 참여하지 않은 경우 `true`를 올바르게 반환됩니다.	2023년 2월 28일	AWS SimSpace Weaver 가이드 카탈로그를 참조하세요.	다운로드할 수 없음

버전	Notes	릴리스 날짜	설명서	앱 SDK 다운로드
1.12.0	정식 출시(GA)를 위한 릴리스	2022년 11월 29일	AWS SimSpace Weaver 가이드 카탈로그를 참조하세요.	다운로드할 수 없음

AWS SimSpace Weaver 버전 1.17.0

이 릴리스는 SimSpace Weaver 앱 SDK 배포 가능 패키지의 점검입니다. 오래된 Windows 배치 및 Linux Bash 스크립트를 Python 기반 스크립트로 대체했습니다.

> **Important**
>
> 이제 Python SDK뿐만 아니라 Python 3.9도 스크립트와 샘플을 사용해야 합니다.

목차

1.17.0의 주요 변경 사항

- 간소화된 프로젝트 생성
  - 를 실행한 후 샘플을 복사하여 붙여넣기만 `setup.py`하면 자체 프로젝트를 생성할 수 있습니다.
- 원클릭 샘플
  - 이제 배포 zip 파일에는 배포를 설정한 후 작동하는 ready-to-use 수 있는 샘플이 포함되어 있습니다.
- 이제 각 SDK는 자체 디렉터리인 `cpp`, `python`, `unreal`및에 존재합니다`unity`. 사용하는 SDK에 따라 경로를 업데이트해야 할 수 있습니다.
- 헬퍼 스크립트 개선.
  - 이제 스크립트에는 유연성을 극대화하는 여러 AWS CLI 옵션이 포함되어 있습니다.
  - 빠른 시작의 일부로 콘솔 클라이언트 시작 및 연결 통합.

  - 콘솔 출력이 개선되었습니다.
  - Unreal 및 Unity 샘플 빌드는 이제 더 이상 수동 단계가 필요 `quick-start`없이에서 작동합니다.
  - SimSpace Weaver Local 이제를 호출하기만 하면가 작동하므로 수동 빌드 및 시작이 더 이상 필요하지 `quick-start`않습니다.
  - SimSpace Weaver Local `quick-start`는 앱 출력 로깅을 통합 지원합니다.
  - SimSpace Weaver Local는 이제 ssh 세션과 같은 비 GUI 환경에서 시작할 수 있습니다.
  - 이제 "사용자 지정 컨테이너" 기능이 `quick-start` 스크립트에 통합되었습니다.
  - Amazon Linux 2(AL2) 지원 향상: 이제 Windows 및 AL2용 스크립트 워크플로가 비슷합니다. 이전에는 AL2 프로젝트에 더 많은 수동 단계가 필요했으며 AL2에 대해 지원되지 SimSpace Weaver Local 않았습니다.
- Unreal Engine 및 Unity 플러그인이 이제 SimSpace Weaver 앱 SDK 배포 가능 패키지의 일부로 포함됩니다.
- 에 대한 버그 수정 SimSpace Weaver Local
  - 엔터티에 동일한 엔터티 ID를 할당할 수 있는 버그를 수정했습니다.
  - 두 파티션에 동일한 파티션 ID를 할당할 수 있는 버그를 수정했습니다.
  - 앱이 소유하지 않은 엔터티에 쓰려고 시도하는 것과 관련된 버그를 수정했습니다.
  - 메모리 누수 문제를 해결했습니다.

프로젝트를 1.17.0으로 업데이트

1. 1.17.0 배포 설정: 1.17.0에서 변경했기 때문에 설정 절차를 다시 진행합니다. 자세한 내용은 에 대한 설정 SimSpace Weaver 단원을 참조하십시오.
2. 이제 각 Weaver 앱 SDK가 자체 디렉터리에 존재합니다. 이를 반영하도록 빌드 경로를 업데이트합니다.

   a. C++ 디렉터리: `SimSpaceWeaverAppSdk/cpp`

      - 이제 SimSpace Weaver C++ 앱 SDK가 `FindSimSpaceWeaverAppSdk.cmake` 파일을 사용합니다. 이 파일은에 연결되는 `weaver` 대상을 설정하고에서 Weaver를 위해 빌드할 때 중요한 버그 수정을 포함합니다 AWS 클라우드. 바이너리에 직접 연결하는 대신이 옵션을 사용해야 합니다.

   b. Python 디렉터리: `SimSpaceWeaverAppSdk/python`

   c. Unity 플러그인: `SimSpaceWeaverAppSdk/unity`

d. Unreal Engine 플러그인: `SimSpaceWeaverAppSdk/unreal`

3. 이전 `tools` 스크립트는 새 SimSpace Weaver 배포에서는 작동하지 않습니다. 프로젝트에서 새 `tools` 스크립트를 사용하려면:

   a. 이전 `tools/windows`, `tools/linux`및 `tools/local` 디렉터리를 삭제합니다.

   b. 프로젝트와 동일한 SimSpace Weaver 앱 SDK를 사용하는 샘플 프로젝트의 `tools` 디렉터리를 복사합니다. 이 디렉터리를 복사하기 `setup.py` 전에를 실행했는지 확인합니다.

> **Important**
>
> 도구 스크립트는 샘플 프로젝트에서만 작동하도록 보장됩니다. 프로젝트 작업을 위해 이러한 스크립트, 특히 스크립트를 편집해야 `build.py` 할 수 있습니다. 편집은 프로젝트에 고유하므로 지침을 제공할 수 없습니다.

버전 1.17.0에 대해 자주 묻는 질문

버전 1.17.0으로 업데이트해야 합니까?

SimSpace Weaver API 또는 SimSpace Weaver 앱 SDK에는 변경 사항이 없으므로 필수 업데이트는 아닙니다. 여러 버그 수정이 SimSpace Weaver Local포함된 1.17.0를 사용하려면 1.17.0으로 업데이트해야 합니다.

필요한 최소 Python 버전은 무엇입니까?

Python 3.9는 최소 버전입니다.

필요한 최소 CMake 버전은 무엇입니까?

CMake 버전 3.13이 최솟값입니다.

필요한 Unreal Engine의 최소 버전은 무엇입니까?

Unreal Engine 5.0이 최솟값입니다.

필요한 Unity의 최소 버전은 무엇입니까?

Unity 버전 2022.3.19.F1이 최솟값입니다.

AWS SimSpace Weaver 버전 1.15.1

이 릴리스는 원래 SimSpace Weaver 버전 1.15.0에서 릴리스된 Python SDK의 필수 업데이트입니다. Python 기반 시뮬레이션이 AWS 클라우드에서 실패했던 버전 불일치 문제를 수정합니다. 1.15.0 대신 이 버전을 사용하비낟.

기존 Python 프로젝트를 1.15.1로 업데이트

버전 1.15.0 Python SDK로 만든 기존 Python 프로젝트가 있는 경우 다음 단계를 수행하여 AWS 클라우드에서 실행할 수 있도록 1.15.1로 업데이트해야 합니다.

이 절차를 따르는 대신 1.15.1 Python SDK를 사용하여 새 Python 프로젝트를 만들고 사용자 지정 코드를 새 프로젝트로 이동할 수도 있습니다.

1.15.0 Python 프로젝트를 1.15.1로 업데이트

1. Python 프로젝트의 폴더로 이동합니다.
2. `src/PythonBubblesSample/bin/run-python`에서 다음 줄을 변경합니다.

   ```
   export PYTHONPATH=$PYTHONPATH:/roapp/lib
   ```

   다음의 것들로 변경됩니다.

   ```
   export PYTHONPATH=$PYTHONPATH:$LD_LIBRARY_PATH:/roapp/lib
   ```

3. `CMakeLists.txt`에서 다음 줄을 삭제합니다.

   - ```
     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)
     ```

버전 1.15.1에 대한 문제 해결

1.15.0 Python 시뮬레이션을 업데이트한 후에서 시작하지 못함 AWS 클라우드

증상: 시뮬레이션을 시작한 후 약 5~10분 후에 시뮬레이션 관리 로그에 `internal error`가 보고되고 시뮬레이션 상태는 `FAILED`입니다.

이는 1.15.0 Python SDK의 라이브러리 파일이 앱 zip 파일에 포함된 경우 발생할 수 있습니다. 프로젝트 업데이트 단계를 완료했는지 확인하고, `libweaver_app_sdk_python_v1.so`가 zip 파일에 없거나 어떤 식으로든 참조되지 않았는지 확인합니다.

버전 1.15.1에 대한 FAQ

이 릴리스는 Python SDK 이외의 다른 것에도 영향을 미치나요?

아니요.

버전 1.15.1로 업데이트해야 하나요?

공간 앱에 Python을 사용하지 않으려는 경우 1.15.1로 업데이트할 필요가 없습니다. 1.15.0으로 업데이트한 경우 Python 기반 시뮬레이션은에서 실행되지 않습니다 AWS 클라우드. 1.15.0을 사용하는 경우 1.15.1로 업데이트하는 것이 좋습니다.

`$LD_LIBRARY_PATH`이란 무엇인가요?

이는 AWS 클라우드에서 시뮬레이션을 실행하는 경우의 Python SDK 위치입니다. 1.15.1이 새 버전입니다. 향후 Python 버전 문제가 발생하지 않도록 하기 위해 이렇게 변경했습니다. 해당 디렉터리에 연결하는 것은 1.15.0에서의 `libweaver_app_sdk_python_v1.so` 링크와 기능적으로 동일합니다.

에 대한 문서 기록 AWS SimSpace Weaver

다음 표에서는 SimSpace Weaver 설명서의 중요한 변경 사항을 설명합니다.

날짜	변경 사항	설명서 업데이트	API 버전 업데이트됨
2025년 5월 20일	지원 종료 알림	지원 종료 공지: 2026 AWS 년 5월 20일에에 대한 지원이 종료됩니다 AWS SimSpace Weaver. 2026년 5월 20일 이후에는 SimSpace Weaver 콘솔 또는 SimSpace Weaver 리소스에 더 이상 액세스할 수 없습니다. 자세한 내용은 AWS SimSpace Weaver 지원 종료를 참조하세요.	N/A
2024년 4월 17일	업데이트된 콘텐츠	버전 1.17.0 릴리스의 사용 설명서 전체에서 업데이트되었습니다. 설정 장 및 시작 자습서의 주요 변경 사항입니다. 자세한 내용은 릴리스 정보를 참조하세요.	N/A
2024년 2월 12일	업데이트된 콘텐츠	버전 1.16.0 릴리스에 대한 AWS SimSpace Weaver 버전장을 업데이트했습니다.	N/A
2024년 2월 12일	새로운 내용	버전 1.16.0 릴리스의 일부로 메시징 섹션을 추가했습니다. 이 섹션에서는 SimSpace Weaver 앱 SDK에 추가된 메시징	N/A

날짜	변경 사항	설명서 업데이트	API 버전 업데이트됨
		APIs에 대해 설명합니다. 이러한 APIs 사용하여 앱 간에 메시지를 보내고 받을 수 있습니다.	
2024년 2월 12일	업데이트된 콘텐츠	버전 1.16.0에 대한 SimSpace Weaver 시뮬레이션 스키마 참조장을 업데이트했습니다.	N/A
2024년 2월 12일	업데이트된 콘텐츠	SimSpace Weaver 엔드포인트 및 할당량 장에 메시징을 위한 서비스 할당량을 추가했습니다.	N/A
2024년 2월 12일	새 가이드	1.16.0 이전 버전의 콘텐츠를 별도의 가이드로 분할합니다. 이전 버전의 AWS SimSpace Weaver 가이드에 액세스할 수 있도록 가이드 카탈로그(기본 설명서 랜딩 페이지에서 사용 가능)가 추가되었습니다.	N/A
2023년 12월 4일	업데이트된 콘텐츠	버전 1.15.3 릴리스의 AWS SimSpace Weaver 버전 장을 업데이트했습니다.	N/A
2023년 12월 4일	업데이트된 콘텐츠	최신 버전의 설치 지침을 포함하도록 AWS SimSpace Weaver 버전 장을 업데이트했습니다.	N/A

날짜	변경 사항	설명서 업데이트	API 버전 업데이트됨
2023년 12월 4일	업데이트된 콘텐츠	SimSpace Weaver Local의 Service Quotas를 업데이트했습니다.	N/A
2023년 12월 4일	신규 및 업데이트된 콘텐츠	의 로컬 개발 SimSpace Weaver 섹션을 재구성하고 버전 1.15.3에 도입된 SimSpace Weaver Local의 차이점을 설명하는 새 페이지를 추가했습니다.	N/A
2023년 11월 7일	업데이트된 콘텐츠	Docker 및 WSL이 앱 SDK용 직접 다운로드 링크/URL을 사용하도록 설정하는 지침이 업데이트되었습니다. 자세한 내용은 에 대한 로컬 환경 설정 SimSpace Weaver 단원을 참조하십시오.	N/A
2023년 11월 2일	업데이트된 콘텐츠	1.15.2 릴리스의 서비스 버전 페이지가 업데이트되었습니다. 자세한 내용은 서비스 버전 단원을 참조하십시오.	N/A

날짜	변경 사항	설명서 업데이트	API 버전 업데이트됨
2023년 10월 23일	업데이트된 콘텐츠	앱 SDK 배포 패키지를 다운로드하기 위한 새로운 지침으로 서비스 버전 페이지가 업데이트되었습니다. 이제 고객은 승인된 직접 다운로드 링크 중 하나만 사용해야 하며 AWS CLI 를 사용하여 앱 SDK 배포 가능 패키지를 다운로드하면 안 됩니다. 자세한 내용은 최신 버전 다운로드 단원을 참조하십시오.	N/A
2023년 9월 22일	새로운 내용	1.15.1 릴리스에 대한 업데이트 지침이 포함된 버전 노트 페이지가 추가되었습니다. 자세한 내용은 AWS SimSpace Weaver 버전 1.15.1 단원을 참조하십시오.	N/A
2023년 9월 10일	새로운 내용	가를 인식하지 AWS CLI 못하는 상황에 대한 문제 해결 섹션을 추가했습니다 SimSpace Weaver. 자세한 내용은 AWS CLI 가 인식하지 못함 `simspaceweaver` 단원을 참조하십시오.	N/A

날짜	변경 사항	설명서 업데이트	API 버전 업데이트됨
2023년 9월 10일	업데이트된 콘텐츠	WSL AWS CLI 의에 대한 설치 지침이 업데이트되었습니다. 자세한 내용은 에서 Amazon Linux 2 (AL2)에 대한 SimSpace Weaver 배포 패키지 설정 Windows Subsystem for Linux (WSL) 단원을 참조하십시오.	N/A
2023년 9월 7일	API 업데이트	이제 S3Location 데이터 유형에는 BucketName 및 ObjectKey가 필요합니다. 이제 S3Destination 데이터 유형에는 BucketName이 필요합니다.	AWS SDK: 2023-09-07
2023년 8월 31일	새로운 내용	릴리스 1.15.0에 대한 새 섹션 추가: Python 작업.	N/A
2023년 8월 15일	업데이트된 콘텐츠	공식 SimSpace Weaver Amazon S3 버킷만 나열하도록 AWS SimSpace Weaver 버전의 다운로드 지침이 업데이트되었습니다. 다른 다운로드 위치는 에서 제어하지 AWS 않으며 악성 코드를 포함할 수 있습니다.	N/A
2023년 7월 26일	업데이트된 콘텐츠	클럭 업데이트됨	N/A
2023년 7월 26일	업데이트된 콘텐츠	공간 도메인 구성 업데이트됨	N/A

날짜	변경 사항	설명서 업데이트	API 버전 업데이트됨
2023년 7월 26일	새로운 내용	새로운 단원(사용자 지정 컨테이너)이 추가되었습니다.	N/A
2023년 7월 26일	업데이트된 콘텐츠	릴리스 1.14.0 AWS SimSpace Weaver 버전을 업데이트했습니다.	N/A
2023년 7월 6일	새로운 내용	새로운 단원(PathfindingSample 콘솔 클라이언트가 연결되지 않음)이 추가되었습니다.	N/A
2023년 6월 7일	업데이트된 콘텐츠	릴리스 1.13.1 AWS SimSpace Weaver 버전을 업데이트했습니다.	N/A
2023년 5월 15일	새로운 내용	새로운 단원(에서 스냅샷 사용 AWS CloudFormation)이 추가되었습니다.	N/A
2023년 4월 29일	새로운 내용	릴리스 1.13.0의 콘텐츠가 추가되었습니다. 자세한 내용은 AWS SimSpace Weaver 버전 단원을 참조하십시오.	AWS SDK: 2023-04-28

날짜	변경 사항	설명서 업데이트	API 버전 업데이트됨
2023년 3월 27일	새로운 내용	시뮬레이션의 최대 지속 시간을 설명하는 섹션이 추가되었습니다. 릴리스 1.12.3의 자습서에 SimSpace Weaver 앱 SDK 스크립트에 `--maximum-duration` 파라미터에 대한 지원을 추가한 참고 사항이 추가되었습니다.	N/A
2023년 3월 9일	콘텐츠 변경	당사는 Windows 및 Windows Subsystem for Linux (WSL)용 Docker에 대한 지침만 제공하며 WSL(및 기타 Linux 환경)은 지원되지 않는다는 점을 명확히 했습니다.	N/A
2023년 2월 28일	새로운 내용	SimSpace Weaver 버전을 설명하는 장이 추가되었습니다.	N/A
2023년 2월 28일	콘텐츠 변경	AWS Command Line Interface ()에 대한 AWS IAM Identity Center 및 명명된 프로필 사용을 포함하도록 인증에 대한 내용을 변경했습니다AWS CLI.	N/A
2023년 2월 17일	새로운 내용	를 사용하여 리소스를 관리하는 방법에 대한 섹션을 추가했습니다 AWS CloudFormation.	N/A

날짜	변경 사항	설명서 업데이트	API 버전 업데이트됨
2023년 1월 23일	새로운 내용	로컬 시뮬레이션을 디버깅하기 위한 지침이 추가되었습니다.	N/A
2022년 11월 29일	서비스 시작	SimSpace Weaver에 대한 사용 설명서 및 API 참조가 릴리스되었습니다.	AWS SDK: 2022-11-29

용어집

이 용어집은 특정 용어를 정의합니다 AWS SimSpace Weaver.

최신 AWS 용어는 AWS 일반 참조의 AWS 용어집을 참조하세요.

A

앱
사용자가 만든 실행 코드(바이너리락도 함). 앱이라는 용어는 코드 또는 해당 코드의 실행 중인 인스턴스를 가리킬 수 있습니다. 앱은 시뮬레이션 동작을 캡슐화합니다. 앱은 엔터티를 만들고, 삭제하고, 읽고, 업데이트합니다.

앱 SDK
SimSpace Weaver에서 앱을 통합하는 데 사용하는 소프트웨어 개발 키트(SDK)입니다. SDK는 엔터티 데이터를 읽고 쓰며 시뮬레이션 시간을 추적하기 위한 API를 제공합니다. 자세한 내용은 SimSpace Weaver 앱 SDK 단원을 참조하십시오.

C

클라이언트
외부에 존재 SimSpace Weaver 하고 사용자 지정 앱 또는 서비스 앱을 통해 시뮬레이션과 상호 작용하는 프로세스(또는 정의)입니다. 클라이언트를 사용하여 시뮬레이션 상태를 보거나 변경할 수 있습니다.

클럭
내부 예약 프로세스의 추상화 SimSpace Weaver입니다. 클럭은 틱을 앱에 게시하여 시간 동기화를 유지합니다. 각 시뮬레이션에는 자체 클럭이 있습니다.

클럭 속도
클럭이 앱에 게시하는 초당 틱 수입니다. 지원되는 클럭 속도에 대한 자세한 내용은 SimSpace Weaver 엔드포인트 및 할당량 섹션을 참조하세요.

클럭 틱 속도
클럭 속도를 참조하세요.

컴퓨팅 리소스 단위
작업자의 컴퓨팅 리소스 단위(프로세서 및 메모리)입니다. 일반적으로 앱의 단일 인스턴스에는 컴퓨팅 리소스 단위 1개가 할당됩니다. 각 앱에 둘 이상의 컴퓨팅 리소스 단위를 할당할 수 있습니다.

사용자 지정 앱
시뮬레이션 상태를 읽고 상호 작용하는 데 사용하는 앱 유형입니다. 사용자 지정 앱은 시뮬레이션에서 엔터티를 만들 수 있지만 엔터티를 소유하지는

	않습니다. 사용자 지정 앱이 엔터티를 생성할 때 엔티티를 공간 도메인으로 이전해야 합니다. 앱 API를 사용하여 사용자 지정 앱의 수명 주기를 제어할 수 있습니다. SimSpace Weaver APIsSimSpace Weaver API 참조.
사용자 지정 도메인	사용자 지정 앱이 포함된 도메인입니다.
사용자 지정 파티션	사용자 지정 앱의 파티션입니다.

D

기한	작업(예: 틱 처리)이 완료되어야 하는 실제 시간입니다.
도메인	동일한 실행 코드(앱 바이너리)를 실행하고 시작 옵션이 동일한 앱 인스턴스 그룹입니다.

E

엔드포인트(서비스)	프로그램(예:)이 SimSpace Weaver 서비스에 연결하는 데 사용하는 정규화된 도메인 이름(FQDN AWS Command Line Interface)입니다.
엔드포인트(시뮬레이션)	클라이언트가 시뮬레이션에 연결하기 위해 사용하는 IP 주소 및 포트 번호입니다. 사용자 지정 앱 및 서비스 앱에서 엔드포인트를 구성할 수 있습니다.
엔터티	고객 데이터 객체(또는 해당 정의)입니다. 엔터티는 정적(한 위치에 유지) 또는 동적(시뮬레이션 공간 내 이동)일 수 있습니다. 시뮬레이션 내 사람과 건물을 예로 들 수 있습니다.

정보

인덱스(시뮬레이션)	공간 경계 및 좌표계를 포함한 시뮬레이션의 공간 속성에 대한 설명입니다.

L

수명 주기(앱)	시뮬레이션 중에 앱 거칠 것으로 예상되는 논리적 단계에 대한 설명입니다. 수명 주기는 관리형(앱SimSpace Weaver 시작 및 중지) 또는 비관리형(앱 시작 및 중지)입니다.

로드(엔터티 필드 데이터)	State Fabric에서 엔터티 필드 데이터를 읽습니다.

P

파티션	작업자의 공유 메모리 세그먼트입니다. 각 파티션에는 도메인 내 개별 엔터티 하위 집합이 포함됩니다. 각 앱에는 할당된 파티션이 있습니다. 앱은 파티션의 모든 엔터티를 소유합니다. 앱은 엔터티를 만들 때 해당 파티션에 엔터티를 만듭니다. 엔터티가 한 파티션에서 다른 파티션으로 이동하면 원본 파티션의 앱에서 대상 파티션의 앱으로 소유권이 이전됩니다.

R

리소스 단위	???을(를) 참조하세요.

S

스키마	시뮬레이션 구성을 설명하는 YAML 또는 JSON 문서입니다. SimSpace Weaver 는 스키마를 사용하여 시뮬레이션 리소스를 생성합니다.
서비스 앱	시뮬레이션 상태를 읽고 상호 작용하는 데 사용하는 앱 유형입니다. 서비스 앱은 시뮬레이션에서 엔터티를 만들 수 있지만 이를 공간 도메인으로 전송해야 합니다. SimSpace Weaver 는 서비스 앱의 수명 주기를 관리하고 시뮬레이션의 각 작업자에서 하나(또는 시뮬레이션 스키마에 지정된 대로)를 시작합니다.
서비스 도메인	서비스 앱이 포함된 도메인입니다.
서비스 파티션	서비스 앱의 파티션입니다.
시뮬레이션(리소스)	시뮬레이션된 가상 공간을 실행하는 컴퓨팅 클러스터를 추상화한 것입니다. 여러 시뮬레이션이 있을 수 있습니다. 스키마를 사용하여 시뮬레이션을 구성합니다.
공간 앱	핵심 시뮬레이션 로직을 캡슐화하는 앱 유형입니다. 각 공간 앱은 1개(단 1개)의 파티션을 소유합니다.
공간 도메인	공간 앱이 포함된 도메인입니다.
공간 파티션	공간 앱의 파티션입니다.

State Fabric	SimSpace Weaver의 인 메모리 데이터베이스입니다. 는 개체 및 내부 SimSpace Weaver 데이터를 포함한 시뮬레이션의 상태를 State Fabric 저장합니다.
저장(엔터티 필드 데이터)	엔터티 필드 데이터를 State Fabric에 씁니다.
구독	구독 영역에서 데이터를 수신하기 위한 특정 앱 인스턴스의 장기 요청입니다. 구독 앱은 구독을 사용하여 구독 영역 내의 엔터티에 대한 변경 사항을 검색합니다.
구독 영역	시뮬레이션 공간의 2차원 리전입니다. 구독은 구독 영역을 의미합니다. 구독 영역은 2개 이상의 파티션에 걸쳐 있을 수 있으며 파티션의 일부도 포함할 수 있습니다. 구독 영역은 정의된 범위 내에서 연속적입니다.

T

틱	시간에 대한 불연속형 값(벽시계 시간 또는 시뮬레이션 시간)입니다. 앱은 틱 지속 시간보다 빠르게 반복할 수 있지만 지정된 틱을 특정 기한 내에 기록해야 합니다. 주어진 틱에 대한 모든 앱 작업은 다음 틱이 시작되기 전에 완료되어야 합니다.
틱 속도	클럭 속도를 참조하세요.
시간(실제)	Reality의 관점에서 현재 시간입니다.는 Unix 에포크 이후 나노초 수인 64비트 POSIX 타임스탬프를 SimSpace Weaver 사용합니다. (January 1, 1970, 00:00:00 UTC).
시간(시뮬레이션)	simulation.use의 관점에서 현재 시간은 실제 시간에 직접 해당하지 않을 수 있는 64비트 정수 논리적 틱 카운터를 SimSpace Weaver 사용합니다.

W

작업자	시뮬레이션 코드를 실행하는 Amazon Elastic Compute Cloud(Amazon EC2) 인스턴스입니다.