バージョン 1.17.0 のユーザーガイド

AWS SimSpace Weaver

Copyright © 2025 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

AWS SimSpace Weaver: バージョン 1.17.0 のユーザーガイド

Copyright © 2025 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon の商標およびトレードドレスはAmazon 以外の製品およびサービスに使用することはできません。また、お客様に誤解を与える可能性がある形式で、または Amazon の信用を損なう形式で使用することもできません。Amazon が所有していないその他のすべての商標は Amazon との提携、関連、支援関係の有無にかかわらず、それら該当する所有者の資産です。

Table of Contents

..... ix
とは SimSpace Weaver 1
主要なコンセプト 1
の SimSpace Weaver 仕組み 2
の使用方法 SimSpace Weaver 5
シミュレーションスキーマ 6
ワーカーおよびリソースユニット 6
シミュレーションクロック 7
パーティション 7
ステートファブリック 7
エンティティ 8
アプリケーション 8
ユースケースの例 11
AWS SimSpace Weaver サポート終了 13
設定 14
アカウントを設定する 14
にサインアップする AWS アカウント 14
管理アクセスを持つユーザーを作成する 15
SimSpace Weaverを使用するアクセス許可を追加する 16
ローカル環境をセットアップする 18
Docker での AL2 18
WSL での AL2 20
ライセンス対象ソフトウェアを使用する 24
入門 25
クイックスタートチュートリアル 25
ステップ 1: ログ記録を有効にする (オプション) 26
ステップ 2: コンソールクライアントでクイックスタートする (オプション 1) 27
ステップ 2: Unreal Engine クライアントでクイックスタートする (オプション 2) 27
シミュレーションを停止して削除する 27
トラブルシューティング 28
詳細なチュートリアル 28
ステップ 1: ログ記録を有効にする (オプション) 29
ステップ 2: シミュレーションを開始する 29
ステップ 3: ログを確認する (オプション) 36

ステップ 4: シミュレーションを表示する 38
ステップ 5: シミュレーションを停止して削除する 39
トラブルシューティング 40
の使用 SimSpace Weaver 41
シミュレーションの設定 41
シミュレーションの設定パラメータ 42
SDK のバージョン 43
シミュレーションプロパティ 43
ワーカー 44
クロック 45
パーティショニング戦略 48
ドメイン 49
最大期間 59
最大値 59
デフォルト値 59
最小値 59
コンソールを使用したシミュレーションの開始 60
最大期間に達したシミュレーションのステータス 60
アプリケーション開発 60
空間アプリケーション 61
カスタムアプリケーション 61
クライアントアプリケーションの開発 62
IP アドレスとポート番号を取得する 64
Unreal Engine ビュークライアントの起動 67
トラブルシューティング 67
ローカル開発 68
ステップ 1: ローカルシミュレーションを起動する 69
ステップ 2: ローカルシミュレーションを表示する 70
ステップ 3: ローカルシミュレーションを停止する (Windows ではオプション) 71
ローカル開発のトラブルシューティング 72
SimSpace Weaver アプリケーション SDK 72
API メソッドは Result を返します。 73
最上位レベルでのアプリケーション SDK とのやり取り 74
シミュレーション管理 74
サブスクリプション 77
エンティティ 78

エンティティイベント 90
Result とエラー処理 97
ジェネリックとドメインタイプ 99
その他のアプリケーション SDK 操作 99
SimSpace Weaver デモフレームワーク 102
Quotas の使用 103
アプリケーションの制限を取得する 103
アプリケーションが使用しているリソース量を取得する 104
メトリクスをリセットする 105
制限の超過 105
メモリの不足 106
ベストプラクティス 106
シミュレーションのデバッグ 106
SimSpace Weaver Local を使用してコンソール出力を確認する 107
Amazon CloudWatch Logs のログを確認する 107
describe API コールを使用する 107
クライアントを接続する 108
ローカルシミュレーションのデバッグ 108
カスタムコンテナ 109
カスタムコンテナを作成する 110
カスタムコンテナを使用するようにプロジェクトを変更する 111
よくある質問 114
トラブルシューティング 114
Python の使用 115
Python プロジェクトの作成 116
Python シミュレーションを開始する 118
Python のサンプルクライアント 118
よくある質問 119
トラブルシューティング 119
他のエンジンのサポート 121
Unity 121
Unreal Engine 121
ライセンス対象ソフトウェアを使用する 122
を使用したリソースの管理 AWS CloudFormation 122
スナップショット 124
スナップショット 125

スナップショットのユースケース 126
SimSpace Weaver コンソール 127
AWS CLI 128
よくある質問 131
メッセージング 132
メッセージングのユースケース 132
メッセージング APIs 133
メッセージングを使用するタイミング 140
メッセージングを使用する際のヒント 144
メッセージングエラーとトラブルシューティング 145
ベストプラクティス 148
請求アラームの設定 148
SimSpace Weaver Local を使用する 148
不要なシミュレーションを停止します 149
不要なリソースの削除 149
バックアップの取得 149
セキュリティ 150
データ保護 151
保管中の暗号化 152
転送中の暗号化 152
ネットワーク間トラフィックのプライバシー 153
Identity and Access Management 153
対象者 153
アイデンティティを使用した認証 154
ポリシーを使用したアクセスの管理 158
が IAM と AWS SimSpace Weaver 連携する方法 160
アイデンティティベースのポリシーの例 167
がユーザーに代わって SimSpace Weaver 作成するアクセス許可 171
サービス間の混乱した代理の防止 173
トラブルシューティング 176
セキュリティイベントのロギングとモニタリング 179
コンプライアンス検証 180
耐障害性 181
インフラストラクチャセキュリティ 182
ネットワーク接続セキュリティモデル 182
設定と脆弱性の分析 183

セキュリティに関するベストプラクティス 184
アプリケーションとクライアント間の通信を暗号化します。 184
シミュレーション状態の定期的なバックアップ 184
アプリケーションと SDK の維持 184
ログ記録とモニタリング 186
CloudWatch のログ 186
SimSpace Weaver ログのアクセス 186
SimSpace Weaver ログ 187
CloudWatch によるモニタリング 189
SimSpace Weaver アカウントレベルの メトリクス 190
CloudTrail ログ 190
SimSpace Weaver CloudTrail の情報 190
SimSpace Weaver ログファイルエントリについて 191
エンドポイントと Service Quotas 194
サービスエンドポイント 194
Service Quotas 195
メッセージングクォータ 198
クロックレート 199
SimSpace Weaver Local の Service Quotas 199
トラブルシューティング 201
AssumeRoleAccessDenied 201
InvalidBucketName 203
ServiceQuotaExceededException 204
TooManyBuckets 204
シミュレーション開始中にアクセスが拒否される 205
Docker を使用時の時間に関する問題 205
コンソールクライアントが接続に失敗する 206
`simspaceweaver` にいいえ AWS CLI 208
スキーマリファレンス 210
完全なスキーマの例 210
スキーマフォーマット 212
SDK のバージョン 213
シミュレーションのプロパティ 213
ワーカー 215
クロック 216
パーティショニング戦略 216

ドメイン 218
配置の制約事項 228
API リファレンス 230
SimSpace Weaver バージョン 231
最新バージョン 231
現在のバージョンを検索する方法 231
最新バージョンをダウンロードする 231
アプリケーション SDK のダウンロードに関するトラブルシューティング 232
最新バージョンをインストールする 233
サービスバージョン 233
1.17.0 247
1.17.0 の主な変更点 247
プロジェクトを 1.17.0 に更新する 248
バージョン 1.17.0 に関するよくある質問 249
1.15.1 250
既存の Python プロジェクトを 1.15.1 に更新します 250
バージョン 1.15.1 のトラブルシューティング 251
バージョン 1.15.1 に関するよくある質問 251
ドキュメント履歴 253
用語集 261

サポート終了通知: 2026 年 5 月 20 日に、 AWS は のサポートを終了します AWS SimSpace Weaver。2026 年 5 月 20 日以降、 SimSpace Weaver コンソールまたは SimSpace Weaver リソースにアクセスできなくなります。詳細については、AWS SimSpace Weaver 「サポート終了」を参照してください。

翻訳は機械翻訳により提供されています。提供された翻訳内容と英語版の間で齟齬、不一致または矛盾がある場合、英語版が優先します。

とは AWS SimSpace Weaver

AWS SimSpace Weaver は、 で大規模な空間シミュレーションを構築および実行するために使用できるサービスです AWS クラウド。例えば、群集シミュレーション、大規模な現実世界環境、没入感のあるインタラクティブな体験を作成できます。

を使用すると SimSpace Weaver、シミュレーションワークロードを複数の Amazon Elastic Compute Cloud (Amazon EC2) インスタンスに分散できます。 は基盤となる AWS インフラストラクチャ SimSpace Weaver をデプロイし、シミュレーションを実行している Amazon EC2 インスタンス間のシミュレーションデータ管理とネットワーク通信を処理します。

の主な概念 SimSpace Weaver

シミュレーションやゲームは、それを実行するコンピューターによって制限されます。仮想世界の規模と複雑さが増すにつれて、処理性能は低下し始めます。計算に時間がかかり、システムのメモリが不足し、クライアントのフレームレートが低下します。リアルタイムのパフォーマンスを必要としないシミュレーションでは、これは煩わしいだけである可能性があります。あるいは、処理の遅延が増えるとコストが増加する、ビジネスクリティカルな状況となる可能性があります。シミュレーションやゲームにリアルタイムのパフォーマンスが必要な場合、パフォーマンスの低下は間違いなく問題です。

パフォーマンスの限界に達したシミュレーションの一般的なソリューションは、シミュレーションを単純化することです。多くのユーザーがいるオンラインゲームでは、仮想世界のコピーを異なるサーバー上に作成し、ユーザーをサーバー全体に分散させることで、スケールの問題に対処することがよくあります。

SimSpace Weaver は、仮想世界を空間的に分割し、 で実行されるコンピューティングインスタンスのクラスター全体に分散することで、スケーリングの問題を解決します AWS クラウド。コンピュートインスタンスは連携して、シミュレーション世界全体をパラレル処理します。シミュレーション世界は、その中のすべてのものと、それに接続するすべてのクライアントにとって、単一の統合空間のように見えます。ハードウェアのパフォーマンスの限界のためにシミュレーションを単純化する必要はもうありません。代わりに、クラウドにコンピューティング容量を追加することもできます。

トピック

- の SimSpace Weaver 仕組み
- の使用方法 SimSpace Weaver

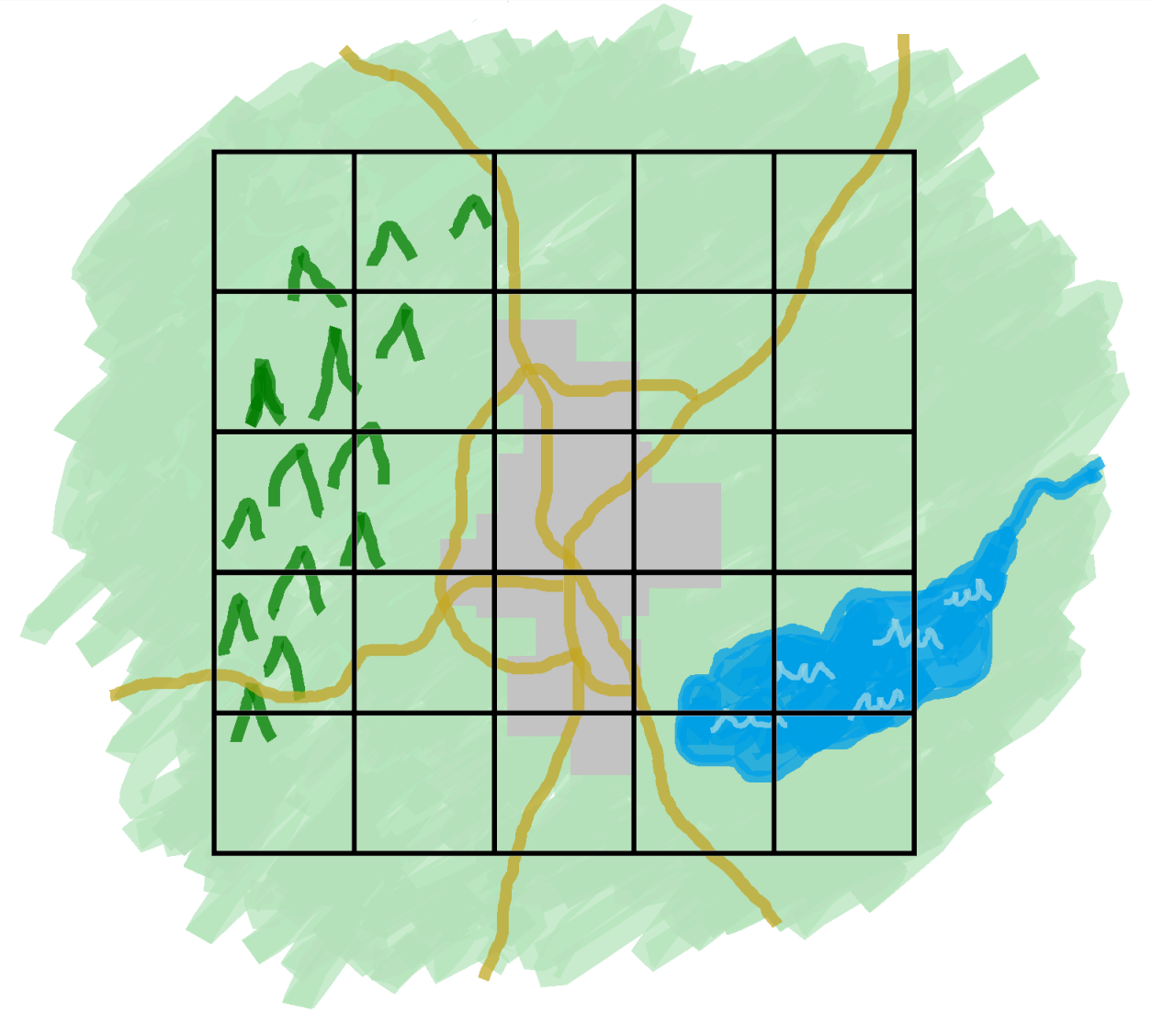
- シミュレーションスキーマ
- ワーカーおよびリソースユニット
- シミュレーションクロック
- パーティション
- ステートファブリック
- エンティティ
- アプリケーション

の SimSpace Weaver 仕組み

シミュレーションは、その中にオブジェクトが存在する世界で構成されます。一部のオブジェクト (人や乗り物など) は動いて何かをします。他のオブジェクト (木や建物など) は静的です。では SimSpace Weaver、エンティティはシミュレーション世界のオブジェクトです。

シミュレーション世界の境界を定義し、それをグリッドに分割します。グリッド全体で動作するシミュレーションロジックを作成する代わりに、グリッドの 1 つのセルで動作するシミュレーションロジックを作成します。空間アプリケーションは SimSpace Weaver、グリッドのセルのシミュレーションロジックを実装するプログラムです。これには、そのセル内のすべてのエンティティのロジックが含まれます。空間アプリケーションの所有領域は、空間アプリケーションが制御するグリッドセルです。

> **Note**
>
> では SimSpace Weaver、「app」という用語は、アプリのコードまたはそのコードの実行中のインスタンスを指します。



グリッドに分割されるシミュレーション世界

シミュレーション世界をグリッドに分割します。各空間アプリケーションは、そのグリッド内の 1 つのセルに対してシミュレーションロジックを実装します。

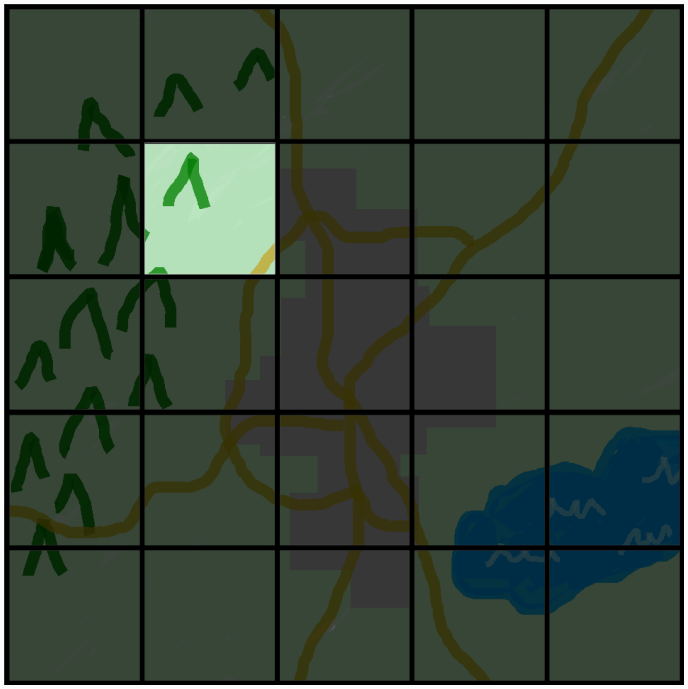
SimSpace Weaver は、グリッドの各セルに対して空間アプリケーションコードのインスタンスを実行します。すべての空間アプリケーションインスタンスはパラレル実行されます。基本的に、 はシミュレーション全体を複数の小さなシミュレーションに SimSpace Weaver 分割します。より小さなシミュレーションはそれぞれ、シミュレーションの世界全体の一部を処理します。 は、これらのより小さなシミュレーションを の複数の Amazon Elastic Compute Cloud (Amazon EC2) インスタンス

(ワーカーと呼ばれる) に分散して実行 SimSpace Weaver できます AWS クラウド。単一のワーカーで複数の空間アプリケーションを実行できます。

エンティティはシミュレーション世界内を移動できます。エンティティが別の空間アプリケーションの所有領域 (グリッド内の別のセル) に入ると、新しい領域の空間アプリケーションの所有者がエンティティの制御を引き継ぎます。シミュレーションが複数のワーカーで実行される場合、エンティティはあるワーカーの空間アプリケーションの制御から別のワーカーの空間アプリケーションの制御に移る可能性があります。エンティティが別のワーカーに移動すると、 は基盤となるネットワーク通信 SimSpace Weaver を処理します。

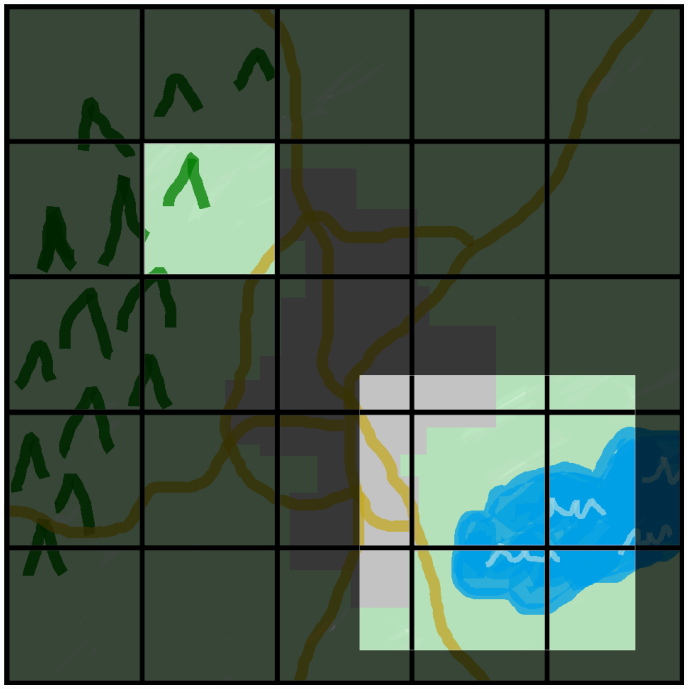
サブスクリプション

世界の空間アプリケーションビューは、それ自体が所有する領域です。シミュレーション世界の別の場所で何が起きているかを調べるために、空間アプリケーションはサブスクリプションを作成します。サブスクリプション領域は、シミュレーション世界領域全体のサブセットです。サブスクリプションエリアには、空間アプリの所有エリアなど、複数の所有エリアの一部を含めることができます。 は、サブスクリプションエリア内で発生するすべてのエンティティイベント (入力、終了、作成、更新、削除など) を空間アプリ SimSpace Weaver に通知します。



世界の空間アプリケーションビュー

世界の空間アプリケーションビューは所有領域であり、世界グリッド内の 1 つのセルです。



サブスクリプション領域が追加された空間アプリケーションビュー

空間アプリケーションは、サブスクリプションを使用してシミュレーション世界の別の場所で何が起こっているかを調べます。サブスクリプション領域には、複数のグリッドセルとセルの一部を含めることができます。

例えば、物理的に相互作用するエンティティをシミュレーションするアプリケーションでは、所有領域の空間的境界のすぐ向こう側にあるエンティティについて知る必要がある場合があります。これを実現するために、アプリケーションは所有領域の境界となる領域をサブスクライブできます。サブスクリプションを作成すると、アプリケーションはその領域のエンティティイベントに関する通知を受け取り、エンティティを読み取ることができます。もう 1 つの例として、領域を所有するアプリケーションに関係なく、200 メートル先にあるすべてのエンティティを確認する必要がある自動走行車があります。車両用アプリケーションでは、表示可能な領域をカバーする軸に沿ったバウンディングボックス (AABB) としてフィルターを使用してサブスクリプションを作成できます。

シミュレーションの空間的側面を管理する必要のないシミュレーションロジックを作成できます。カスタムアプリケーションは 単一のワーカーで実行される実行可能なプログラムです。カスタムアプリケーションのライフサイクル (開始と停止) を制御します。シミュレーションクライアントはカスタムアプリケーションに接続して、シミュレーションを表示したり操作したりできます。すべてのワーカーで実行されるサービスアプリケーションを作成することもできます。 は、シミュレーションを実行するすべてのワーカーでサービスアプリケーションのインスタンス SimSpace Weaver を起動します。

カスタムアプリケーションとサービスアプリケーションは、エンティティイベントについて学習したりエンティティを読み取ったりするためのサブスクリプションを作成します。これらのアプリケーションは空間的ではないため、所有領域はありません。サブスクリプションを使用することが、シミュレーションの世界で何が起こっているのかを知る唯一の方法です。

の使用方法 SimSpace Weaver

を使用する場合 SimSpace Weaver、以下の主要なステップに従います。

1. C++ アプリケーション SDK を統合する SimSpace Weaver アプリケーションを記述して構築します。
   a. アプリケーションは API コールを行ってシミュレーション状態とやりとりします。
2. 一部のアプリケーションを通じてシミュレーションを表示したり操作したりするクライアントを作成します。

3. シミュレーションをテキストファイルで構成します。
4. アプリケーションパッケージとシミュレーション構成をサービスにアップロードします。
5. シミュレーションを開始します。
6. カスタムアプリケーションを、必要に応じて起動または停止します。
7. クライアントをカスタムアプリケーションまたはサービスアプリケーションに接続して、シミュレーションを表示または操作します。
8. Amazon CloudWatch Logs でシミュレーションログを確認します。
9. シミュレーションを停止します。
10. シミュレーションをクリーンアップします。

シミュレーションスキーマ

シミュレーションスキーマ (またはスキーマ) は、シミュレーションの設定情報を含む YAML形式のテキストファイルです。 は、シミュレーションを開始するときにスキーマ SimSpace Weaver を使用します。 SimSpace Weaver アプリケーション SDK 配布パッケージには、サンプルプロジェクトのスキーマが含まれています。自身のスキーマの開始ポイントとしてこれを使用できます。シミュレーションスキーマの詳細については、「SimSpace Weaver シミュレーションスキーマリファレンス」を参照してください。

ワーカーおよびリソースユニット

ワーカーは、シミュレーションを実行する Amazon EC2 インスタンスです。シミュレーションスキーマでワーカータイプを指定します。 は、ワーカータイプをサービスが使用する特定の Amazon EC2 インスタンスタイプに SimSpace Weaver マッピングします。 はワーカーを自動的に SimSpace Weaver 起動および停止し、ワーカー間のネットワーク通信を管理します。 はシミュレーションごとにワーカーのセット SimSpace Weaver を開始します。シミュレーションが異なれば、使用するワーカーも異なります。

ワーカーで使用できるコンピューティング (プロセッサおよびメモリ) 容量は、コンピュートリソースユニット (またはリソースユニット) と呼ばれる論理ユニットに分割されます。リソースユニットは、一定量のプロセッサおよびメモリの容量を表します。

> **Note**
>
> 以前はコンピュートリソースユニットをスロットと呼んでいました。ドキュメントには、現在でもこの過去の用語が使用されている可能性があります。

シミュレーションクロック

各シミュレーションには独自のクロックがあります。API コールまたは [SimSpace Weaver] コンソールを使用してクロックを起動および停止します。シミュレーションはクロックが動作しているときにのみ更新されます。シミュレーションのすべての操作は、ティックと呼ばれる時間セグメント内で行われます。クロックはすべてのワーカーに各ティックの開始時間を通知します。

クロックレート (またはティックレート) は、クロックが通知する 1 秒あたりのティック数 (ヘルツ、または Hz) です。シミュレーションに必要なクロックレートは、シミュレーションスキーマの一部です。ティックのすべての操作は、次のティックが開始される前に完了する必要があります。このため、実効クロックレートは目的のクロックレートよりも低くなる可能性があります。実効クロックレートが目的のクロックレートより高くなることはありません。

パーティション

パーティションは、ワーカーの共有メモリの一部です。各パーティションにはシミュレーション状態データの一部が格納されます。

空間アプリケーションのパーティション (空間アプリケーションパーティションまたは空間パーティションとも呼ばれます) には、空間アプリケーションの所有権領域内のすべてのエンティティが含まれます。 は、各エンティティの空間位置に基づいて、空間アプリケーションパーティションにエンティティ SimSpace Weaver を配置します。つまり、 SimSpace Weaver は互いに空間的に近いエンティティを同じワーカーに配置しようとします。これにより、所有していないエンティティについてアプリケーションが必要とする知識の量を最小限に抑え、所有しているエンティティをシミュレートできます。

ステートファブリック

ステートファブリックは、すべてのワーカー上の共有メモリ (すべてのパーティションの集まり) のシステムです。シミュレーションのすべての状態データが格納されます。

ステートファブリックは、エンティティをそのエンティティの各データフィールドの初期データと更新ログのセットとして記述するカスタムバイナリフォーマットを使用します。このフォーマットを使用すると、シミュレーション時間内の前の時点におけるエンティティの状態にアクセスし、それを現実世界の特定の時点にマップし直すことができます。バッファのサイズは限られており、バッファ内のサイズを超えて時間をさかのぼることはできません。 は、各フィールドの更新ログの現在のオフセットへのポインタ SimSpace Weaver を使用し、フィールドの更新の一部としてポインタを更新します。 は、これらの更新ログを共有メモリを使用してアプリケーションのプロセススペースに SimSpace Weaver マッピングします。

このオブジェクト形式は、オーバーヘッドが低く、シリアル化コストも発生しません。 SimSpace Weaver また、 は、このオブジェクト形式を使用してインデックスフィールド (エンティティの位置など) を解析および識別します。

エンティティ

エンティティは、シミュレーションにおけるデータの最小の構成要素です。エンティティの例としては、アクター (人や乗り物など) や静的オブジェクト (建物や障害物など) があります。エンティティには、 SimSpace Weaverに永続データとして保存できるプロパティ (位置や向きなど) があります。エンティティはパーティション内に存在します。

アプリケーション

A SimSpace Weaver app は、各シミュレーションティックを実行するカスタムロジックを含む、ユーザーが記述するソフトウェアです。ほとんどのアプリケーションの目的は、シミュレーションの実行時にエンティティを更新することです。アプリケーションは、 SimSpace Weaver アプリケーション SDK で APIs を呼び出して、シミュレーション内のエンティティに対してアクションを実行 (読み取りや更新など) します。

アプリとその必要なリソース (ライブラリなど) を .zip ファイルとしてパッケージ化し、 にアップロードします SimSpace Weaver。アプリケーションはワーカーの Docker コンテナで実行されます。 SimSpace Weaver は各アプリケーションに一定数のワーカーのリソースユニットを割り当てます。

SimSpace Weaver は、各アプリケーションに 1 つの (1 つのみ) パーティションの所有権を割り当てます。アプリケーションとそのパーティションは同じワーカー上にあります。各パーティションにはアプリケーション所有者が 1 人しかいません。アプリケーションは、パーティション内のエンティティを作成、読み取り、更新、および削除できます。アプリケーションはパーティション内のすべてのエンティティを所有します。

アプリケーションには、空間アプリケーション、カスタムアプリケーション、サービスアプリケーションの 3 種類があります。ユースケースやライフサイクルによって異なります。

> **Note**
>
> では SimSpace Weaver、「app」という用語は、アプリケーションまたはそのコードの実行中のインスタンスのコードを参照できます。

空間アプリケーション

空間アプリケーションは、シミュレーションに空間的に存在するエンティティの状態を更新します。例えば、ティックごとに速度、形状、サイズに基づいてエンティティを移動したり衝突させたりする `Physics` アプリケーションを定義できます。この場合、 SimSpace Weaver は `Physics` アプリケーションの複数のインスタンスをパラレル実行して、ワークロードのサイズを処理します。

SimSpace Weaver は空間アプリケーションのライフサイクルを管理します。シミュレーションスキーマで空間アプリケーションパーティションの配置を指定します。シミュレーションを起動すると、 SimSpace Weaver は空間アプリケーションのパーティションごとに空間アプリケーションを起動します。シミュレーションを停止すると、 は空間アプリケーションを SimSpace Weaver シャットダウンします。

他の種類のアプリケーションはエンティティを作成できますが、エンティティを更新できるのは空間アプリケーションだけです。他のタイプのアプリケーションでは、作成したエンティティを空間ドメインに転送する必要があります。 はエンティティの空間位置 SimSpace Weaver を使用して、エンティティを空間アプリケーションのパーティションに移動します。これにより、エンティティの所有権が空間アプリケーションに転送されます。

カスタムアプリケーション

カスタムアプリケーションを使用してシミュレーションを操作します。カスタムアプリケーションはサブスクリプションを使用してエンティティデータを読み取ります。カスタムアプリケーションはエンティティを作成できます。ただし、エンティティをシミュレーションに含めて更新するには、アプリケーションがエンティティを空間アプリケーションに転送する必要があります。カスタムアプリに SimSpace Weaver ネットワークエンドポイントを割り当てることができます。シミュレーションクライアントはネットワークエンドポイントに接続してシミュレーションを操作できます。シミュレーションスキーマでカスタムアプリケーションを定義しますが、(SimSpace Weaver API コールを使用して) 起動および停止するのはユーザーの責任です。ワーカーでカスタムアプリケーションインスタンスを起動した後、 SimSpace Weaver はインスタンスを別のワーカーに転送しません。

サービスアプリケーション

サービスアプリケーションは、すべてのワーカーで読み取り専用プロセスを実行する必要がある場合に使用できます。例えば、大規模なシミュレーションを行っていて、シミュレーション内を移動して表示されているエンティティだけをユーザーに表示する表示クライアントが必要な場合は、サービスアプリケーションを使用できます。この場合、1 つのカスタムアプリケーションインスタンスではシミュレーション内のすべてのエンティティを処理することはできません。すべてのワーカーで起動するようにサービスアプリケーションを構成できます。これらのサービスアプリケーションはそれ

ぞれ、割り当てられたワーカーのエンティティをフィルタリングして、関連するエンティティだけを接続しているクライアントに送信できます。これにより、閲覧中のクライアントは、シミュレーション空間内を移動する際にさまざまなサービスアプリケーションに接続できます。シミュレーションスキーマでサービスアプリを設定します。 は、サービスアプリを自動的に SimSpace Weaver 起動および停止します。

アプリケーションの要約

以下の表は、 SimSpace Weaver アプリケーションのタイプ別の特性をまとめたものです。

	空間アプリケーション	カスタムアプリケーション	サービスアプリケーション
エンティティの読み込み	あり	あり	はい
エンティティの更新	はい	いいえ	いいえ
エンティティを作成する	はい	Yes*	Yes*
ライフサイクル	管理 (SimSpace Weaver 制御)	管理対象外 (ユーザーが制御)	管理 (SimSpace Weaver 制御)
起動方法	SimSpace Weaver は、スキーマで指定されているように、空間パーティションごとに 1 つのアプリケーションインスタンスを開始します。	ユーザーが各アプリケーションインスタンスを起動します。	SimSpace Weaver は、スキーマで指定されているように、各ワーカーで 1 つ以上のアプリケーションインスタンスを起動します。
クライアントが接続できる	いいえ	はい	はい

* カスタムアプリケーションまたはサービスアプリケーションでエンティティを作成する場合、空間アプリケーションがエンティティの状態を更新できるように、アプリケーションはそのエンティティの所有権を空間アプリケーションに転送する必要があります。

ドメイン

A SimSpace Weaver domain は、同じ実行可能アプリケーションコードを実行し、同じ起動オプションとコマンドを持つアプリケーションインスタンスのコレクションです。ドメインは、その中に含まれるアプリケーションの種類 (空間ドメイン、カスタムドメイン、サービスドメイン) で呼ばれます。アプリケーションはドメイン内で設定します。

サブスクリプションとレプリケーション

アプリケーションは空間領域へのサブスクリプションを作成して、その領域内のエンティティイベント (入力、終了、作成、更新、削除など) を学習します。アプリケーションは、所有していないパーティション内のエンティティのデータを読み取る前に、サブスクリプションからのエンティティイベントを処理します。

パーティションはアプリケーションと同じワーカーに存在できますが (ローカルパーティションと呼ばれます)、別のアプリケーションがそのパーティションを所有できます。パーティションは別のワーカー (リモートパーティションと呼ばれます) に存在することもできます。サブスクリプションがリモートパーティションに対するものである場合、ワーカーはレプリケーションと呼ばれるプロセスを通じてリモートパーティションのローカルコピーを作成します。次に、ワーカーはローカルコピー (複製されたリモートパーティション) を読み取ります。ワーカー上の別のアプリケーションが同じティックでそのパーティションから読み取る必要がある場合、ワーカーは同じローカルコピーを読み取ります。

のユースケースの例 SimSpace Weaver

は、空間コンポーネントを使用したエージェントベースのモデルや離散時間ステップシミュレーション SimSpace Weaver に使用できます。

大勢の群集シミュレーションを作成する

を使用して SimSpace Weaver 、実際の環境で群衆をシミュレートできます。 SimSpace Weaver を使用すると、シミュレーションを独自の動作で数百万の動的オブジェクトにスケールできます。

都市規模の環境を作成する

SimSpace Weaver を使用して、都市全体のデジタルツインを作成します。都市計画、交通経路の設計、および環境ハザード対応の計画のためのシミュレーションを作成します。独自の地理空間データソースを環境の構成要素として使用できます。

没入感のあるインタラクティブな体験の作成

複数のユーザーが参加して交流できるシミュレーション体験を作成します。Unreal Engine や Unity などの一般的な開発ツールを使用して、3 次元 (3D) の仮想世界を構築します。独自のコンテンツと動作で 3D 体験をカスタマイズします。

AWS SimSpace Weaver サポート終了

慎重に検討した結果、2026 年 5 月 20 日以降、 のサポートを終了することにしました。 AWS SimSpace Weaver は AWS SimSpace Weaver 2025 年 5 月 20 日以降、新規顧客を受け入れなくなります。2025 年 5 月 20 日より前にサービスにサインアップしたアカウントを持つ既存のお客様は、引き続き AWS SimSpace Weaver 機能を使用できます。2026 年 5 月 20 日以降、 を使用することはできません AWS SimSpace Weaver。

コンテナ化されたシミュレーションの実行に役立つ AWS Batch への移行の詳細については、このブログ記事を参照してください。

のセットアップ SimSpace Weaver

SimSpace Weaver を初めて使用するための設定を行うには、 AWS アカウント とローカル環境を設定する必要があります。これらのタスクを完了したら、開始方法のチュートリアルに進むことができます。

タスクの設定

1. を使用する AWS アカウント ように を設定する SimSpace Weaver.
2. のローカル環境をセットアップする SimSpace Weaver.

を使用する AWS アカウント ように を設定する SimSpace Weaver

を使用する AWS アカウント ように を設定するには、次のタスクを実行します SimSpace Weaver。

にサインアップする AWS アカウント

がない場合は AWS アカウント、次の手順を実行して作成します。

にサインアップするには AWS アカウント

1. https://portal.aws.amazon.com/billing/signup を開きます。
2. オンラインの手順に従います。

   サインアップ手順の一環として、電話またはテキストメッセージを受け取り、電話キーパッドで検証コードを入力します。

   にサインアップすると AWS アカウント、 AWS アカウントのルートユーザー が作成されます。ルートユーザーには、アカウントのすべての AWS のサービス とリソースへのアクセス権があります。セキュリティベストプラクティスとして、ユーザーに管理アクセス権を割り当て、ルートユーザーアクセスが必要なタスクの実行にはルートユーザーのみを使用するようにしてください。

AWS サインアッププロセスが完了すると、 から確認メールが送信されます。https://aws.amazon.com/ の [マイアカウント] をクリックして、いつでもアカウントの現在のアクティビティを表示し、アカウントを管理することができます。

管理アクセスを持つユーザーを作成する

にサインアップしたら AWS アカウント、日常的なタスクにルートユーザーを使用しないように AWS アカウントのルートユーザー、 を保護し AWS IAM Identity Center、 を有効にして管理ユーザーを作成します。

を保護する AWS アカウントのルートユーザー

1. ルートユーザーを選択し、 AWS アカウント E メールアドレスを入力して、アカウント所有者AWS Management Consoleとして にサインインします。次のページでパスワードを入力します。

   ルートユーザーを使用してサインインする方法については、AWS サインイン ユーザーガイドのルートユーザーとしてサインインするを参照してください。

2. ルートユーザーの多要素認証 (MFA) を有効にします。

   手順については、IAM ユーザーガイドの AWS アカウント 「ルートユーザー (コンソール) の仮想 MFA デバイスを有効にする」を参照してください。

管理アクセスを持つユーザーを作成する

1. IAM アイデンティティセンターを有効にします。

   手順については、「AWS IAM Identity Center ユーザーガイド」の「AWS IAM Identity Centerの有効化」を参照してください。

2. IAM アイデンティティセンターで、ユーザーに管理アクセスを付与します。

   を ID ソース IAM アイデンティティセンターディレクトリ として使用する方法のチュートリアルについては、「 AWS IAM Identity Center ユーザーガイド」の「デフォルトを使用してユーザーアクセスを設定する IAM アイデンティティセンターディレクトリ」を参照してください。

管理アクセス権を持つユーザーとしてサインインする

- IAM アイデンティティセンターのユーザーとしてサインインするには、IAM アイデンティティセンターのユーザーの作成時に E メールアドレスに送信されたサインイン URL を使用します。

  IAM Identity Center ユーザーを使用してサインインする方法については、AWS サインイン 「 ユーザーガイド」の AWS 「 アクセスポータルにサインインする」を参照してください。

追加のユーザーにアクセス権を割り当てる

1. IAM アイデンティティセンターで、最小特権のアクセス許可を適用するというベストプラクティスに従ったアクセス許可セットを作成します。

   手順については、「AWS IAM Identity Center ユーザーガイド」の「権限設定を作成する」を参照してください。
2. グループにユーザーを割り当て、そのグループにシングルサインオンアクセス権を割り当てます。

   手順については、「AWS IAM Identity Center ユーザーガイド」の「グループの結合」を参照してください。

SimSpace Weaverを使用するアクセス許可を追加する

アクセス権限を付与するには、ユーザー、グループ、またはロールにアクセス許可を追加します。

- 以下のユーザーとグループ AWS IAM Identity Center：

  アクセス許可セットを作成します。「AWS IAM Identity Center ユーザーガイド」の「権限設定を作成する」の手順に従ってください。
- IAM 内で、ID プロバイダーによって管理されているユーザー:

  ID フェデレーションのロールを作成します。詳細については「IAM ユーザーガイド」の「サードパーティー ID プロバイダー (フェデレーション) 用のロールを作成する」を参照してください。
- IAM ユーザー:
  - ユーザーが担当できるロールを作成します。手順については「IAM ユーザーガイド」の「IAM ユーザーのロールの作成」を参照してください。
  - (お奨めできない方法) ポリシーをユーザーに直接アタッチするか、ユーザーをユーザーグループに追加します。詳細については「IAM ユーザーガイド」の「ユーザー (コンソール) へのアクセス権限の追加」を参照してください。

Example を使用するアクセス許可を付与する IAM ポリシー 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"
                }
            }
        }
    ]
}
```

のローカル環境をセットアップする SimSpace Weaver

SimSpace Weaver シミュレーションは、コンテナ化された Amazon Linux 2 (AL2) 環境で実行されます。アプリケーションをコンパイルして SimSpace Weaver アプリ SDK にリンクするには、AL2 環境が必要です。標準のローカル開発環境は Docker の AL2 コンテナです。Docker を使用しない場合は、Windows Subsystem for Linux (WSL) で AL2 環境を実行するための代替手順が用意されています。また、独自の方法でローカル AL2 環境を作成できます。AL2 をローカルで実行するその他の方法については、「Amazon EC2 のドキュメント」を参照してください。

> **Important**
>
> Microsoft Windows の Docker は標準的な開発環境です。便宜上、ローカル開発環境を設定する他の方法も提案しますが、これらは標準ではなく、サポートされていません。

トピック

- で Amazon Linux 2 (AL2) の SimSpace Weaver ディストリビューションパッケージを設定する Docker
- で Amazon Linux 2 (AL2) の SimSpace Weaver ディストリビューションパッケージを設定する Windows Subsystem for Linux (WSL)

で Amazon Linux 2 (AL2) の SimSpace Weaver ディストリビューションパッケージを設定する Docker

このセクションでは、 で AL2 環境でローカル SimSpace Weaver ディストリビューション zip を設定する手順について説明しますDocker。で AL2 を使用して を設定する手順についてはWindows Subsystem for Linux (WSL)、「」を参照してくださいで Amazon Linux 2 (AL2) の SimSpace Weaver ディストリビューションパッケージを設定する Windows Subsystem for Linux (WSL)。

要件

- Microsoft Windows 10 以降、または互換性のある Linux システム
- Microsoft Visual Studio 2019 またはそれ以降、Desktop development with C++ ワークロードをインストールした状態
- CMake3
- Git

- Docker Desktop
- AWS CLI
- Python 3.9

で AL2 を使用して SimSpace Weaver ディストリビューション zip を設定するには Docker

1. の AWS 認証情報をまだ設定していない場合は AWS CLI、「AWS CLI の設定」の手順に従います。
2. SimSpace Weaver アプリケーション SDK 配布可能パッケージをダウンロードします。以下の要素が含まれます。
   - SimSpace Weaver アプリケーション開発用のバイナリとライブラリ
   - 開発ワークフローの一部を自動化するヘルパースクリプト
   - SimSpace Weaver 概念を示すサンプルアプリケーション
3. ファイルを任意の *sdk-folder* に解凍します。
4. *sdk-folder* に移動します。
5. 次のコマンドを入力して、必要な Python パッケージをインストールします。

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

6. 次のコマンドを入力して、Docker イメージで SimSpace Weaver ディストリビューションを設定します。

```
python setup.py
```

   このコマンドは次のことを行います。

   - SimSpace Weaver プロジェクトの構築に関するすべての要件がインストールされた AL2 Docker イメージを作成します。
   - シミュレーションの起動に必要な CloudFormation リソースを作成します。
     - サンプル CloudFormation スタックテンプレートは、 にあります。 *sdk-folder*/`PackagingTools/sample-stack-template.yaml`
   - ローカルシステムの正しいパスを使用して、提供されたサンプルプロジェクトを設定します。

トラブルシューティング

- Docker がスタックしているように見える
  - Docker コマンドが呼び出された後にコンソール出力が停止しているように見える場合は、Docker エンジンを再起動してみてください。それが機能しない場合は、コンピュータを再起動します。

で Amazon Linux 2 (AL2) の SimSpace Weaver ディストリビューションパッケージを設定する Windows Subsystem for Linux (WSL)

このセクションでは、 で AL2 環境で SimSpace Weaver ディストリビューション zip を設定する手順について説明しますWindows Subsystem for Linux (WSL)。Docker での AL2 を設定する手順については、「で Amazon Linux 2 (AL2) の SimSpace Weaver ディストリビューションパッケージを設定する Docker」を参照してください。

> **Important**
>
> このセクションでは、Amazon が所有、開発、サポートしていないバージョンの AL2 を使用するソリューションについて説明します。このソリューションは、Docker を使用しないことを選択した場合の利便性のみを目的として提供されています。このソリューションを使用することを選択した場合、Amazon および AWS は責任を負いません。

要件

- Windows 10 での Hyper-V
- Windows Subsystem for Linux (WSL)
- WSL 用のサードパーティ製オープンソース AL2 配布 (ダウンロードバージョン 2.0.20200722.0-update.2) (手順を参照してください)

> **Important**
>
> このWSL 手順では、WSL 用の AL2 配布の 2.0.20200722.0-update.2 バージョンを使用しています。他のバージョンを使用すると、エラーが発生する可能性があります。

で AL2 を使用して SimSpace Weaver 配布 zip を設定するには WSL

1. Windows のコマンドプロンプトで、WSL の AL2 環境を起動します。

```
wsl -d Amazon2
```

> **Important**
>
> で実行中はWSL、 にある Python `quick-start.py` ヘルパースクリプトのいずれかを実行するときに `--al2`オプションを含めます`sdky-folder/Samples/sample-name/tools/cloud/quick-start.py`。

2. Linux シェルプロンプトで、yum パッケージマネージャーを更新します。

```
yum update -y
```

> **Important**
>
> このステップがタイムアウトになった場合は、WSL1に切り替えてこれらの手順を再試行する必要がある場合があります。WSL AL2 セッションを終了し、Windows コマンドプロンプトで以下を入力します。
>
> ```
> wsl --set-version Amazon2 1
> ```

3. 解凍ツールをインストールします。

```
yum install -y unzip
```

4. yum インストール AWS CLI されている をすべて削除します。yum がインストールされているかどうかわからない場合は、次のコマンドの両方を試してください AWS CLI。

```
yum remove awscli
```

```
yum remove aws-cli
```

5. 一時ディレクトリを作成して、そこに移動します。

```
mkdir ~/temp
```

```
cd ~/temp
```

6. 以下をダウンロードしてインストールします AWS CLI。

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

7. 一時ディレクトリは削除できます。

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

8. シェルセッションを再開して、環境内のパスを更新します。

```
exec
```

9. AL2 環境で AWS CLI の AWS 認証情報を設定します。詳細については、「AWS CLI の設定」を参照してください。を使用する場合は AWS IAM Identity Center、「 AWS Command Line Interface ユーザーガイド」の「 を使用する AWS CLI ように AWS IAM Identity Centerを設定する」を参照してください。

```
aws configure
```

10. Git をインストールする。

```
yum install -y git
```

11. `wget` をインストールします。

```
yum install -y wget
```

12. SimSpace Weaver アプリケーション SDK のフォルダを作成します。

```
mkdir sdk-folder
```

13. SDK フォルダに移動します。

```
cd sdk-folder
```

14. SimSpace Weaver アプリケーション SDK 配布可能パッケージをダウンロードします。以下の要素が含まれます。

    - SimSpace Weaver アプリケーション開発用のバイナリとライブラリ
    - 開発ワークフローの一部を自動化するヘルパースクリプト
    - SimSpace Weaver 概念を示すサンプルアプリケーション

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

15. ファイル を解凍します。

    ```
    unzip *.zip
    ```

16. WSL セットアップスクリプトを実行します。

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

17. 次のコマンドを入力して、必要な Python パッケージをインストールします。

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

18. SimSpace Weaver ディストリビューション zip セットアップスクリプトを実行します。

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

    このコマンドは次のことを行います。

    - シミュレーションの起動に必要な CloudFormation リソースを作成します。
      - サンプル CloudFormation スタックテンプレートは、 にあります。 *sdk-folder*/PackagingTools/sample-stack-template.yaml
    - ローカルシステムの正しいパスを使用して、提供されたサンプルプロジェクトを設定します。

> **Note**
>
> これは、WSL の AL2 環境に対して 1 回だけ実行する必要があります。

でのライセンスソフトウェアの使用 AWS SimSpace Weaver

AWS SimSpace Weaver では、選択したシミュレーションエンジンとコンテンツを使用してシミュレーションを構築できます。の使用に関連して SimSpace Weaver、シミュレーションで使用するソフトウェアまたはコンテンツのライセンス条項を取得、維持、遵守する責任はお客様にあります。仮想化ホスト環境でのソフトウェアとコンテンツの展開がライセンス契約書上で許可されていることを確認してください。

の開始方法 SimSpace Weaver

このセクションでは、 の使用開始に役立つチュートリアルを提供します SimSpace Weaver。これらのチュートリアルでは、 でシミュレーションを構築するための一般的なワークフローを紹介します SimSpace Weaver。これらのチュートリアルでは、 でシミュレーションを作成、デプロイ、実行する方法を示します SimSpace Weaver。シミュレーションを数分で実行するには、クイックスタートチュートリアルから始めることをお勧めします。詳細については、それ以降の他のチュートリアルを参照してください。

これらのチュートリアルでは、セットアップ手順中にダウンロードした SimSpace Weaver アプリケーション SDK .zip ファイルに含まれているサンプルアプリケーション (`PathfindingSample`) を使用します。サンプルアプリケーションは、空間パーティショニング、クロスパーティションエンティティのハンドオフ、アプリケーション、サブスクリプションなど、すべての SimSpace Weaver シミュレーションが共有する概念を示しています。

チュートリアルでは、4 つの空間パーティションを含むシミュレーションを作成します。`PathfindingSample` 空間アプリケーションの個別のインスタンスが個々のパーティションを管理します。空間アプリケーションは独自のパーティションにエンティティを作成します。エンティティはシミュレーション世界の特定の位置に障害物を避けながら移動します。別のクライアントアプリケーション (SimSpace Weaver アプリケーション SDK に含まれる) を使用してシミュレーションを表示できます。

トピック

- のクイックスタートチュートリアル SimSpace Weaver
- 詳細なチュートリアル: サンプルアプリケーションを構築しながら詳細を説明します

のクイックスタートチュートリアル SimSpace Weaver

このチュートリアルでは、 SimSpace Weaver でシミュレーションを数分で構築して実行するプロセスについて解説します。このチュートリアルから始めて、後で詳細なチュートリアルを実行することをお勧めします。

要件

開始する前に、必ず「のセットアップ SimSpace Weaver」の手順を完了してください。

> **Note**
>
> ここで使用されているスクリプトは、便宜上提供されるものであり、必須ではありません。これらのステップを手動で行う方法については、詳細なチュートリアルを参照してください。

ステップ 1: ログ記録を有効にする (オプション)

ロギングを有効にする

1. 以下に移動します。

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

2. テキストエディタでスキーマファイルを開きます。

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

3. ファイルの冒頭で、`simulation_properties:` セクションを検索します。

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

4. 行 `simulation_properties:` の後に以下の 2 行を挿入します。

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

5. `simulation_properties:` セクションが以下と同じであることを確認します。

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

6. ファイルを保存し、テキストエディタを終了します。

ステップ 2: コンソールクライアントでクイックスタートする (オプション 1)

以下に移動します。

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

以下のいずれかのコマンドを実行します:

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

デフォルトでは、これにより 1 つのワーカーで 1 つのパーティションでシミュレーションが起動されます。`/Samples/PathfindingSample/tools/` フォルダ`--schema {file name}.yaml`から を渡すことで、他の設定を起動できます。

> **Note**
>
> このスクリプトの動作の詳細については、詳細なチュートリアル: サンプルアプリケーションを構築しながら詳細を説明します「」を参照してください。

ステップ 2: Unreal Engine クライアントでクイックスタートする (オプション 2)

「Unreal Engine ビュークライアントの起動」を参照してください。

シミュレーションを停止して削除する

以下に移動します。

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

シミュレーションの名前を見つけます。

```
aws simspaceweaver list-simulations
```

シミュレーションを停止して削除する

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

トラブルシューティング

- FileNotFoundError: cmake

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

  - 解決策： スクリプトはコマンド を見つけることができませんcmake。推奨される最小限の CMake バージョンがインストールされていること、および PATH の cmake コマンドを使用して呼び出すことができることを確認してください。コマンドを使用してcmake -version確認します。
- ImportError: libweaver_app_sdk_python_v1 のインポート中に DLL のロードに失敗しました: 指定されたモジュールが見つかりませんでした。
  - 解決策： このエラーは、Python 3.9 を使用して Weaver Python SDK を起動していない場合に発生します。「python」コマンドに関連付けられた Python バージョンが Python 3.9 であることを確認します。コマンドを実行して確認できますpython --version。
- クイックスタートスクリプトは、Docker Build の開始後、 にスタックして表示されます。
  - 解決策： Docker のウォームアップに数分かかる場合があります。この問題が最大 5 分以上続く場合は、Docker またはシステムを再起動してください。
- target_compile_features CXX コンパイラの既知の機能「GNU」はありません。
  - 解決策： Docker キャッシュをクリアし、Weaverappbuilder Docker イメージを削除し、プロジェクトビルドアーティファクトを削除して、 を再実行しますsetup.py。これにより、Docker 環境がリセットされ、エラーが解決されます。

詳細なチュートリアル: サンプルアプリケーションを構築しながら詳細を説明します

クイックスタートチュートリアルでは、 と を使用してサンプルシミュレーションを構築、開始、停止、削除する方法について説明しますquick-start.pystop-and-delete.py。このチュートリアルでは、これらのスクリプトの仕組みと、カスタム Weaver シミュレーションの柔軟性を最大化するためにこれらのスクリプトが実行できる追加のパラメータについて詳しく説明します。

要件

開始する前に、必ず「のセットアップ SimSpace Weaver」の手順を完了してください。

ステップ 1: ログ記録を有効にする (オプション)

ロギングを有効にする

1. 以下に移動します。

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

2. テキストエディタでスキーマファイルを開きます。

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

3. ファイルの冒頭で、`simulation_properties:` セクションを検索します。

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

4. 行 `simulation_properties:` の後に以下の 2 行を挿入します。

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

5. `simulation_properties:` セクションが以下と同じであることを確認します。

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

6. ファイルを保存し、テキストエディタを終了します。

ステップ 2: シミュレーションを開始する

クイックスタートチュートリアルで示すように、サンプルシミュレーションを起動するための最も基本的なステップは次のとおりです。

1. 以下に移動します。

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

2. 次のいずれかのコマンドを実行します。
   - Docker: `python quick-start.py`
   - WSL: `python quick-start.py --al2`

このスクリプトは、一般的なターミナルコマンドを自動化します。これらはすべて、 を使用して手動で実行できます AWS CLI。これらのステップは次のとおりです。

1. Weaver スキーマを S3 にアップロードします。
   - SimSpace Weaver はスキーマを使用してシミュレーションを設定します。スキーマは YAML 形式のプレーンテキストファイルです。詳細については、「シミュレーションの設定」を参照してください。
2. カスタムコンテナを構築してアップロードします (オプション) 。
   - スキーマがカスタムコンテナを定義する場合、クイックスタートスクリプトは Docker イメージを構築し、Amazon ECR にアップロードします。詳細については、「カスタムコンテナ」を参照してください。この機能の例については、`PythonBubblesSample`スキーマを参照してください。
3. プロジェクトをビルドします。
   - `quick-start.py` は、 で定義された `build_project`関数を呼び出します`build.py`。このステップはプロジェクトによって異なります。PathfindingSample には、CMake が使用されます。CMake コマンドと Docker コマンド。このコマンドは にあります`build.py`。
4. ビルドアーティファクトを S3 にアップロードします。
   - S3 バケットをチェックして、すべてのアップロードが成功したことを確認できます。Amazon S3 でファイルを管理する方法については、「Amazon Simple Storage Service ユーザーガイド」の「Amazon S3 バケットの作成、設定、および使用」を参照してください。
   - サンプルアプリケーションの zip と S3 バケットは、次の名前形式を使用します。
     - `weaver-sample-bucket-account-number-region`
     - 空間アプリケーション: `ProjectNameSpatial.zip`
     - 表示 (カスタム) アプリ: `ProjectNameView.zip`

5. シミュレーションを開始します。

   - これは`aws simspaceweaver start-simulation` AWS CLI 呼び出しのラッパーです。詳細については、AWS CLI 「コマンドリファレンス」を参照してください SimSpace Weaver。
   - スクリプトは、シミュレーションのステータスが STARTED または FAILED になるまでループします。シミュレーションが開始されるまで数分かかる場合があります。

6. シミュレーションの詳細を取得します。

   - DescribeSimulation API は、シミュレーションの状態など、シミュレーションに関する詳細を提供します。シミュレーションは、以下の状態のいずれかになります。

     シミュレーションライフサイクルの状態

     1. **STARTING** — StartSimulation 呼び出し後の初期状態
     2. **STARTED** — すべての空間アプリケーションが起動し、正常に動作している
     3. **STOPPING** — StopSimulation 呼び出し後の初期状態
     4. **STOPPED** — すべてのコンピュートリソースが停止している
     5. **DELETING** — DeleteSimulation 呼び出し後の初期状態
     6. **DELETED** — シミュレーションに割り当てられたすべてのリソースが削除されている
     7. **FAILED** — シミュレーションに重大なエラー/障害が発生して停止している
     8. **SNAPSHOT_IN_PROGRESS** — スナップショットが進行中

     シミュレーションの詳細を取得する

     1. ListSimulations API を呼び出します。

        ```
        aws simspaceweaver list-simulations
        ```

        このスクリプトには、以下のような各シミュレーションの詳細が表示されます。

        ```
        {
            "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 を呼び出して、シミュレーションの詳細を取得します。*simulation-name* を前のステップの出力のシミュレーションの Name に置き換えます。

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

スクリプトには、以下のように、指定したシミュレーションに関する詳細が表示されます。

```
{
    "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. カスタムアプリケーションを起動します。

- SimSpace Weaver はカスタムアプリケーションのライフサイクルを管理しません。カスタムアプリケーションを起動する必要があります。シミュレーションクロックを開始する前にカスタムアプリケーションを起動するのがベストプラクティスですが、カスタムアプリケーションはシミュレーションクロックを開始した後でも起動できます。

  StartApp API を呼び出してカスタムアプリケーションを起動できます。

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

  StartApp API コールは、指定した名前を使用してカスタムアプリケーションの新しいインスタンスを作成して起動します。既に存在するアプリケーション名を指定すると、エラーが返されます。特定のアプリケーション (インスタンス) を再起動する場合は、まずそのアプリケーションを停止して削除する必要があります。

> **Note**
>
> カスタムアプリケーションの起動前は、シミュレーションのステータスは STARTED である必要があります。

サンプルアプリケーションには、シミュレーションを表示する ViewApp カスタムアプリケーションが用意されています。このアプリケーションは、シミュレーションクライアントを接続するための静的 IP アドレスとポート番号を提供します (これについてはこのチュートリアルの後のステップで行います)。domain は、同じ実行コードと起動オプションを持つアプリケーションのクラスと考えることができます。app name はアプリケーションのインスタンスを識別します。 SimSpace Weaver 概念の詳細については、「」を参照してください[の主な概念 SimSpace Weaver](#)。

DescribeApp API を使用して、起動後にカスタムアプリケーションのステータスを確認できます。

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

このチュートリアルでビューアプリケーションを起動する

1. StartApp に を呼び出しますViewApp。

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

2. DescribeApp を呼び出して、カスタムアプリケーションのステータスを確認します。

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

カスタムアプリケーション (インスタンス) のステータスが STARTED になると、DescribeApp の出力にはそのカスタムアプリケーション (インスタンス) の IP アドレスとポート番号が含まれます。以下の出力例では、IP アドレスは Address の値で、ポート番号は EndpointInfo ブロック内の Actual の値です。

```
{
    "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` の値はアプリケーションコードのバインド先となるポート番号です。 の値は、 がクライアントに SimSpace Weaver 公開して app に接続するポート番号`Actual`です。 は`Declared`ポートを`Actual`ポートに SimSpace Weaver マッピングします。

> **Note**
>
> 「」で説明されている手順を使用してカスタムアプリケーションの IP アドレスとポート番号を取得する、起動したカスタムアプリケーションの IP アドレスとポート番号を取得できます。

8. クロックを起動します。

- シミュレーションを初めて作成したとき、クロックはありますが、作動していません。クロックが作動していないときは、シミュレーションの状態は更新されません。クロックを起動すると、アプリにティックが送信され始めます。ティックごとに、空間アプリケーションは所有するエンティティをステップスルーし、結果をコミットします。 SimSpace Weaver

> **Note**
>
> クロックの起動には 30 ~ 60 秒かかることがあります。

StartClock API を呼び出します。

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

> **Note**
>
> StartClock API は *simulation-name* を使用します。これは ListSimulations API を使用して検索できます。
>
> ```
> aws simspaceweaver list-simulations
> ```

クイックスタートパラメータ

- -h, --help
  - これらのパラメータを一覧表示します。
- --clean
  - ビルドする前に、ビルドディレクトリの内容を削除します。
- --al2
  - Docker ではなくネイティブマシンに直接ビルドします。これは、WSL などの Amazon Linux 2 環境で実行されている場合にのみ使用します。
- --uploadonly
  - スキーマとアプリの zip のみを Amazon S3 にアップロードします。シミュレーションを開始しないでください。
- --nobuild

  - プロジェクトの再構築をスキップします。
- --コンテナなし
  - スキーマにリストされているシミュレーションコンテナの再構築をスキップします。
- --consoleclient
  - config.py にリストされているコンソールクライアントを自動的に構築して接続します。
- --スキーマスキーマ
  - この呼び出しで使用するスキーマ。config.py のデフォルト値は「SCHEMA」です。
- --name NAME
  - シミュレーションが持つ名前。デフォルトは、config.py の「PROJECT_NAME」-date-time の値です。

ステップ 3: ログを確認する (オプション)

SimSpace Weaver は、シミュレーション管理メッセージとコンソール出力をアプリから Amazon CloudWatch Logs に書き込みます。ログの使用の詳細については、「Amazon CloudWatch Logs ユーザーガイド」の「ロググループとログストリームの使用」を参照してください。

CloudWatch Logs には、作成した各シミュレーションに独自のロググループがあります。ロググループの名前は、シミュレーションスキーマで指定されます。以下のスキーマスニペットでは、`log_destination_service` の値は `logs` です。つまり、`log_destination_resource_name` の値はロググループの名前です。この場合、ロググループは `MySimulationLogs` です。

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

DescribeSimulation API を使用して、シミュレーションを開始した後でシミュレーション用のロググループの名前を検索することもできます。

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

以下の例は、ロギング設定を説明する、DescribeSimulation から出力の一部を示しています。ロググループの名前は LogGroupArn の末尾に表示されます。

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

各シミュレーションロググループには、いくつかのログストリームが含まれます。

- 管理ログストリーム – SimSpace Weaver サービスによって生成されるシミュレーション管理メッセージ。

  ```
  /sim/management
  ```

- エラーログストリーム – SimSpace Weaver サービスによって生成されたエラーメッセージ。このログストリームは、エラーがある場合にのみ存在します。 は、アプリによって書き込まれたエラーを独自のアプリログストリームに SimSpace Weaver 保存します (次のログストリームを参照) 。

  ```
  /sim/errors
  ```

- 空間アプリケーションログストリーム (各ワーカーの空間アプリケーションごとに 1 つ) — 空間アプリケーションによって生成されるコンソール出力。各空間アプリケーションは、独自のログストリームに書き込みます。*spatial-app-id* は、*worker-id* の末尾にあるスラッシュの後のすべての文字です。

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

- カスタムアプリケーションログストリーム (カスタムアプリケーションインスタンスごとに 1 つ) — カスタムアプリケーションによって生成されるコンソール出力。各カスタムアプリケーションインスタンスは、独自のログストリームに書き込みます。

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

- サービスアプリケーションログストリーム (サービスアプリケーションインスタンスごとに 1 つ) — サービスアプリケーションによって生成されるコンソール出力。各サービスアプリケーションは、独自のログストリームに書き込みます。*service-app-id* は、*service-app-name* の末尾にあるスラッシュの後のすべての文字です。

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

> **Note**
>
> サンプルアプリケーションにはサービスアプリケーションはありません。

ステップ 4: シミュレーションを表示する

SimSpace Weaver アプリケーション SDK には、サンプルアプリケーションを表示するためのさまざまなオプションが用意されています。Unreal Engine 開発のローカルサポートがない場合は、サンプルコンソールクライアントを使用できます。Unreal Engine クライアントの手順は、Windows を使用していることを前提としています。

コンソールクライアントは、エンティティイベントが発生するとそのリストを表示します。クライアントは `ViewApp` からエンティティイベント情報を取得します。コンソールクライアントがイベントのリストを表示すると、シミュレーション内の `ViewApp` およびアクティビティとのネットワーク接続を確認します。

`PathfindingSample` シミュレーションでは、二次元平面上に静止しているエンティティと動いているエンティティが作成されます。移動するエンティティは静止しているエンティティの周りを移動します。Unreal Engine クライアントはエンティティイベントを視覚化します。

コンソールクライアント

`--consoleclient` オプションを含める`quick-start.py`と、 でサンプルを起動するときにコンソールクライアントを自動的に構築して接続できます。`quick-start.py` が既に呼び出された後にコンソールクライアントを構築して接続するには、次の手順を実行します。

以下に移動します。

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

スクリプトを実行して、クライアントを構築して接続します。

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

スクリプトは以下を実行します。

1. CMake を使用してコンソールクライアントを構築します。
2. 指定された IP アドレスとポート番号を使用して、ビルドされた実行可能ファイルを起動します。

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

Unreal Engine クライアント

「Unreal Engine ビュークライアントの起動」を参照してください。

ステップ 5: シミュレーションを停止して削除する

以下に移動します。

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

シミュレーションの名前を見つけます。

```
aws simspaceweaver list-simulations
```

シミュレーションを停止して削除します。

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

スクリプト`stop-and-delete.py`は以下を実行します。

1. AWS CLI コマンドを呼び出してシミュレーションを停止します。
   - `aws simspaceweaver stop-simulation`
   - 詳細については、「 のAWS CLI コマンドリファレンス」を参照してください SimSpace Weaver。

2. AWS CLI コマンドを呼び出してシミュレーションを削除します。
   - `aws simpaceweaver delete-simulation`
   - 詳細については、「 のAWS CLI コマンドリファレンス」を参照してください SimSpace Weaver。

stop-and-delete パラメータ

- -h, --help
  - これらのパラメータを一覧表示します。
- --シミュレーションシミュレーション
  - stop-and-deleteシミュレーションの名前
- --stop
  - シミュレーションのみを停止します。削除しません。
- --delete
  - シミュレーションのみを削除します。シミュレーションが `STOPPED`または の場合にのみ機能します`FAILED`。

トラブルシューティング

クイックスタートチュートリアルのトラブルシューティング「」を参照してください。

の使用 SimSpace Weaver

この章では、 SimSpace Weaverで独自のアプリケーションを構築するのに役立つ情報とガイダンスを提供します。

トピック

- シミュレーションの設定
- シミュレーションの最大期間
- アプリケーション開発
- クライアントアプリケーションの開発
- カスタムアプリケーションの IP アドレスとポート番号を取得する
- Unreal Engine ビュークライアントの起動
- でのローカル開発 SimSpace Weaver
- AWS SimSpace Weaver アプリケーション SDK
- AWS SimSpace Weaver デモフレームワーク
- Service Quotas との連携
- シミュレーションのデバッグ
- カスタムコンテナ
- Python の使用
- 他のエンジンのサポート
- でのライセンスソフトウェアの使用 AWS SimSpace Weaver
- を使用した リソースの管理 AWS CloudFormation
- スナップショット
- メッセージング

シミュレーションの設定

シミュレーションスキーマ (またはスキーマ) は、シミュレーションの設定を指定する YAML フォーマットのテキストファイルです。複数のシミュレーションの開始に同じスキーマを使用できます。スキーマファイルは、シミュレーションのプロジェクトフォルダにあります。任意のテキストエディ

タを使用してファイルを編集できます。 は、シミュレーションを開始するときに SimSpace Weaver のみスキーマを読み取ります。スキーマファイルに加えた編集は、編集後に開始する新しいシミュレーションにのみ影響します。

シミュレーションを設定するには、シミュレーションスキーマファイルを編集します (オペレーティングシステムに適したパス区切り文字を使用します) 。

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

シミュレーションスキーマは、新しいシミュレーションの作成時にアップロードします。プロジェクトのクイックスタートヘルパースクリプトは、シミュレーションを構築するプロセスの一環としてスキーマをアップロードします。

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

クイックスタートスクリプトの実行の詳細については、このガイドの入門章詳細なチュートリアルの「」を参照してください。

シミュレーションの設定パラメータ

シミュレーションスキーマには、以下のようなブートストラップ情報が含まれています。

- シミュレーションプロパティ — SDK バージョンおよびコンピューティングの設定 (ワーカーのタイプと数)
- クロック — ティックレートと許容誤差
- 空間パーティショニング戦略 — 空間トポロジ (グリッドなど)、境界、配置グループ (ワーカーの空間パーティショングループ)
- ドメインとそのアプリケーション — アプリケーションバケット、パス、起動コマンド

SimSpace Weaver は、スキーマ設定を使用して、空間パーティションの設定と配置、アプリの起動、指定したティックレートでのシミュレーションの進め方を行います。

> **Note**
>
> SimSpace Weaver アプリケーション SDK の create-project スクリプトは、サンプルアプリケーションに基づいてシミュレーションスキーマを自動的に生成します。

以下のトピックでは、シミュレーションスキーマのパラメータについて説明します。シミュレーションスキーマの詳細な説明については、SimSpace Weaver シミュレーションスキーマリファレンス を参照してください。

トピック

- SDK のバージョン
- シミュレーションプロパティ
- ワーカー
- クロック
- パーティショニング戦略
- ドメイン

SDK のバージョン

`sdk_version` フィールドは、スキーマがフォーマット SimSpace Weaver されている のバージョンを指定します。有効な値: `1.17`、`1.16`、`1.15`、`1.14`、`1.13`、`1.12`

> **Important**
>
> `sdk_version` の値には、メジャーバージョン番号と最初のマイナーバージョン番号のみが含まれます。例えば、値 `1.12` は `1.12.0`、`1.12.1`、`1.12.2` などのすべてのバージョン `1.12.x` を指定します。

シミュレーションプロパティ

スキーマの `simulation_properties` セクションでは、エンティティのインデックスフィールド (通常は空間位置) のログ記録設定とデータタイプを指定します。

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

`log_destination_service` の値によって、`log_destination_resource_name` の値の解釈が決まります。現在、サポートされている値は `logs` のみです。これ

は、`log_destination_resource_name` の値が Amazon CloudWatch Logs のロググループの名前であることを意味します。

> **Note**
>
> ログ記録はオプションです。ログ送信先のプロパティを設定しない場合、シミュレーションではログは生成されません。

`default_entity_index_key_type` プロパティは必須です。唯一の有効な値は `Vector3<f32>` です。

ワーカー

`workers` セクションでは、シミュレーションに使用するワーカーのタイプと数を指定します。は、Amazon EC2 インスタンスタイプにマッピングされる独自のワーカータイプ SimSpace Weaver を使用します。

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

マルチワーカーシミュレーションを有効にする

複数のワーカーを使用するシミュレーションを作成できます。デフォルトでは、シミュレーションは 1 人のワーカーを使用します。シミュレーションを開始する前に、シミュレーションスキーマを変更する必要があります。

> **Note**
>
> すでに開始されているシミュレーションは変更できません。実行中のシミュレーションで複数のワーカーを有効にする場合は、まずシミュレーションを停止して削除する必要があります。

複数のワーカーを使用するには、コンピュートインスタンスの `desired` の数を 1 より大きい値に設定します。各ワーカーにはアプリケーションの最大数があります。詳細については、「」を参照して

くださいSimSpace Weaver エンドポイントとクォータ。 SimSpace Weaver は、ワーカーのアプリケーション数がこの制限を超えた場合にのみ、複数のワーカーを使用します。 は、使用可能なワーカーのいずれかにアプリケーションを配置 SimSpace Weaver できます。特定のワーカーへのアプリケーションの配置は保証されません。

以下のスキーマスニペットは、2 人のワーカーをリクエストするシミュレーションの設定を示しています。 SimSpace Weaver はアプリケーションの数がワーカーあたりの最大アプリケーション数を超えると、2 番目のワーカーの割り当てを試みます。

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

クロック

この `clock` セクションではシミュレーションクロックのプロパティを指定します。現在、設定できるのはティックレート (クロックがアプリケーションに送信する 1 秒あたりのティック数) のみです。ティックレートは最大レートです。ティックに対するすべての操作 (エンティティの更新など) は次のティックの開始前に終了する必要があるため、実効ティック率が低くなる可能性があります。ティックレートはクロックレートとも呼ばれます。

`tick_rate`の有効値は、スキーマで特定される `sdk_version` によって異なります。

ティックレートの有効値

- `"1.14"` 以前のバージョン:
  - `10`
  - `15`
  - `30`
- `"1.14"` 以降のバージョン:
  - `"10"`
  - `"15"`
  - `"30"`
  - `"unlimited"`

    詳細については、「無制限のティックレート」を参照してください。

> **Important**
>
> - "1.14" 以前の sdk_version のスキーマでは、tick_rate の値は 30 のような整数です。
> - "1.14" 以降の sdk_version のスキーマでは、tick_rate の値は "30" のような文字列です。値には二重引用符を含める必要があります。
>
> バージョン "1.12" または "1.13" のスキーマをバージョン "1.14" 以降に変換する場合は、tick_rate の値を二重引用符で囲む必要があります。

無制限のティックレート

tick_rate を "unlimited" に設定すると、コードの実行と同じ速さでシミュレーションを実行できます。無制限のティックレートで、 はすべてのアプリが現在のティックのコミットを完了した直後に次のティック SimSpace Weaver を送信します。

> **Important**
>
> 1.14.0 より前の SimSpace Weaver バージョンでは、無制限のティックレートはサポートされていません。スキーマの sdk_version の最小値は "1.14" です。

SimSpace Weaver Local で無制限のティックレート

SimSpace Weaver Local はスキーマでティックレートが 10 kHz (10000) と指定されているかのように "unlimited" を実装します。その効果は、 AWS クラウドでの無制限ティックレートと同じです。スキーマでは tick_rate: "unlimited" を引き続き指定できます。SimSpace Weaver Local の詳細については、「での ローカル開発 SimSpace Weaver」を参照してください。

クロックに関するよくある質問

Q1. 開始したシミュレーションを別のティックレートを使用するように変更できますか?

ライフサイクルのどの段階においても、 AWS クラウド にすでに存在しているシミュレーションのティックレートを変更することはできません。また、SimSpace Weaver Local で実行中のシミュレーションのティックレートは、変更できません。tick_rate をスキーマに設定して、そのスキーマから新しいシミュレーションを開始できます。

Q2. 1.14 以前のバージョンで、無制限のティックレートでシミュレーションを実行できますか？

いいえ、1.14.0 以前のバージョンでは無制限のティックレートはサポートされていません。

クロックに関するエラーのトラブルシューティング

シミュレーションが開始されない場合は、DescribeSimulation API の出力で `"StartError"` の値を確認できます。スキーマに無効な `tick_rate` の値があると、以下のエラーが発生します。

> **Note**
>
> ここに示すエラー出力は、読みやすくするために複数行で表示されています。実際のエラー出力は 1 行です。

- `sdk_version` は `"1.14"` よりも前で、`tick_rate` の値は無効な整数です。有効な値: `10`、`15`、`30`

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

- `sdk_version` は `"1.14"` よりも前で、`tick_rate` の値は文字列です。有効値: `10`、`15`、`30`

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

- `sdk_version` は `"1.14"` よりも後で、`tick_rate` の値は無効な文字列です。有効な値: `"10"`、`"15"`、`"30"`、`"unlimited"`

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

- `sdk_version` は `"1.14"` よりも後で、`tick_rate` の値は整数です。有効な値: `"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\"}]"
```

パーティショニング戦略

partitioning_strategies セクションでは、空間アプリケーションのパーティションの設定プロパティを指定します。パーティショニング戦略 (このセクションではプロパティのセット) に独自の名前を指定し、それを空間アプリケーションの設定に使用します。

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

topology プロパティは、シミュレーションで使用する座標システムのタイプを指定します。Grid 値は 2 次元 (2D) グリッドを指定します。

Grid トポロジーの場合、シミュレーション空間は軸に沿ったバウンディングボックス (AABB) としてモデル化されます。AABB の各軸の座標境界を aabb_bounds プロパティで指定します。シミュレーションに空間的に存在するすべてのエンティティは、AABB 内にある必要があります。

グリッド配置グループ

プレイスメントグループは、同じワーカー SimSpace Weaver に配置する空間アプリケーションパーティションのコレクションです。grid_placement_groups プロパティでプレイスメントグループ (グリッド内) の数と配置を指定します。 SimSpace Weaver は、プレイスメントグループ間でパーティションを均等に分散しようとします。同じ配置グループ内のパーティションを持つ空間アプリケーションの所有権エリアは、空間的に隣接します。

x * y は希望するワーカー数と同じにすることをお勧めします。等しくない場合、 は使用可能なワーカー間でプレイスメントグループのバランスを取り SimSpace Weaver ます。

プレイスメントグループ設定を指定しない場合、 SimSpace Weaver はそれを自動的に計算します。

ドメイン

ドメインの設定プロパティのセットの名前を指定します。ドメイン内のアプリケーションの起動設定によって、ドメインのタイプが決まります。

- **`launch_apps_via_start_app_call`** - カスタムドメイン
- **`launch_apps_by_partitioning_strategy`** — 空間ドメイン
- **`launch_apps_per_worker`** (サンプルアプリケーションには含まれていません) — サービスドメイン

> **Important**
>
> SimSpace Weaver は、シミュレーションごとに最大 5 つのドメインをサポートします。これには、すべての空間ドメイン、カスタムドメイン、およびサービスドメインが含まれます。

```
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 app SDK バージョン 1.12.x プロジェクトでは、アプリケーション .zip ファイルとスキーマに別々のバケットを使用します。
>
> - weaver-*lowercase-project-name*-*account-number*-app-zips-*region*
> - weaver-*lowercase-project-name*-*account-number*-schemas-*region*

トピック

- アプリケーションの設定
- 空間ドメインの設定
- ネットワークエンドポイント
- サービスドメインの設定

アプリケーションの設定

アプリケーション (app_config) の設定は、そのドメインの設定の一部として指定します。すべてのタイプのドメインが同じアプリケーション設定プロパティを使用します。

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

> **Note**
>
> SimSpace Weaver app SDK バージョン 1.12.x プロジェクトでは、アプリケーション .zip ファイルとスキーマに別々のバケットを使用します。
>
> - weaver-*lowercase-project-name*-*account-number*-app-zips-*region*
> - weaver-*lowercase-project-name*-*account-number*-schemas-*region*

package プロパティは、S3 バケット内の zipP ファイルの S3 URI を指定します。zip ファイルには、アプリケーションの実行ファイル (バイナリとも呼ばれます) と、必要なその他のリソース (ライブラリなど) が含まれます。アプリケーション実行ファイルの各インスタンスは、ワーカーのDockerコンテナで実行されます。

launch_command プロパティは、実行ファイルの名前と、アプリケーションを実行するためのコマンドラインオプションを指定します。launch_command の値は配列です。起動コマンド文字列全体の各トークンは、配列内の要素です。

例

- 起動コマンドの場合: MyTestApp --option1 value1
- 指定: launch_command: ["MyTestApp", "-option1", "value1"]

required_resource_units プロパティは、 SimSpace Weaver がこのアプリに割り当てるコンピューティングリソースユニットの数を指定します。コンピュートリソースユニットは、ワーカーの容量 (vCPU) とメモリ (RAM) の固定量です。この値を増やすと、ワーカー上で実行されるアプリケーションの処理能力を増やすことができます。各ワーカーが利用できるコンピュートリソースユニットの数には制限があります。詳細については、「SimSpace Weaver エンドポイントとクォータ」を参照してください。

空間ドメインの設定

空間ドメインの場合は、partitioning_strategy を指定する必要があります。このプロパティの値は、スキーマの別の部分で定義したパーティショニング戦略に付けた名前です。

```
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 app SDK バージョン 1.12.x プロジェクトでは、アプリケーション .zip ファイルとスキーマに別々のバケットを使用します。
>
> - weaver-*lowercase-project-name*-*account-number*-app-zips-*region*
> - weaver-*lowercase-project-name*-*account-number*-schemas-*region*

`Grid` トポロジ (このリリースでサポートされている唯一のトポロジ) を使用したパーティショニング戦略は、 にこのドメインの空間アプリケーションパーティション SimSpace Weaver をグリッドに配置するように指示します。`grid_partition` プロパティは、パーティショングリッドの行数と列数を指定します。

SimSpace Weaver は、パーティショングリッド内のセルごとに空間アプリケーションの 1 つのインスタンスを開始します。例えば、空間ドメインに`grid_partition`値 `x: 2`と がある場合`y: 2`、空間ドメインには 2 * 2 = 4 パーティションがあります。 SimSpace Weaver は、空間ドメインで設定されたアプリケーションの 4 つのインスタンスを起動し、各アプリケーションインスタンスに 1 つのパーティションを割り当てます。

トピック

- 空間ドメインのリソース要件
- 複数の空間ドメイン
- 空間ドメインに関するよくある質問
- 空間ドメインのトラブルシューティング

空間ドメインのリソース要件

各ワーカーには最大 17 コンピュートリソースユニットを割り当てることができます。空間ドメインの `app_config` セクションで、各空間アプリケーションが使用するコンピュートリソースユニットの数を指定します。

Example 空間アプリケーションのコンピュートリソースユニットを示すスキーマスニペット

```
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
```

ドメインが必要とするコンピュートリソースユニットの数を計算するには、グリッド内のセル数 (grid_partition 内で x * y) に、空間アプリケーションに割り当てられたコンピュートリソースユニットの数を掛けます。

前の例では、ドメイン MySpatialDomain は以下を指定します。

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

MySpatialDomain のグリッドには 2 * 2 = 4 個のセルがあります。空間ドメインには 4 * 1 = 4 のコンピュートリソースユニットが必要です。

スキーマで指定する全ドメインのコンピュートリソースユニットの総数は、ワーカー数に各ワーカーのコンピュートリソースユニットの最大数 (17) を掛けた desired の数以下でなければなりません。

複数の空間ドメイン

複数の空間ドメインを使用するようにシミュレーションを設定できます。例えば、1 つの空間ドメインを使用してシミュレーションの主なアクター (人や車など) を制御し、別の空間ドメインを使用して環境を制御できます。

また、複数の空間ドメインを使用して、シミュレーションのさまざまな部分に異なるリソースを割り当てることもできます。例えば、あるタイプのエンティティが他のタイプの 10 倍のエンティティインスタンスを持つタイプのエンティティの場合、エンティティタイプごとに異なるドメインを作成して各エンティティタイプを処理し、そのエンティティの数が多いドメインにより多くのリソースを割り当てることができます。

> **Important**
>
> SimSpace Weaver 1.14.0 より前のバージョンでは、複数の空間ドメインはサポートされていません。

> **Important**
>
> AWS SimSpace Weaver Local は現在、複数の空間ドメインをサポートしていません。SimSpace Weaver Local の詳細については、「でのローカル開発 SimSpace Weaver」を参照してください。

> **Important**
>
> SimSpace Weaver は、シミュレーションごとに最大 5 つのドメインをサポートします。これには、すべての空間ドメイン、カスタムドメイン、およびサービスドメインが含まれます。

複数の空間ドメインを設定する

複数の空間ドメインを設定するには、他の空間ドメイン定義を個別の名前付きセクションとしてスキーマに追加します。各ドメインは `launch_apps_by_partitioning_strategy` キーを指定する必要があります。次のスキーマの例を参照してください。

```
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
```

空間ドメインをまとめて配置する

シナリオによっては、空間ドメインのパーティションを別のドメインのパーティションの隣に配置する場合があります。これにより、これらのパーティションが相互にクロスドメインサブスクリプションを作成する場合に、パフォーマンス特性が向上する可能性があります。

最上位キーplacement_constraintsをスキーマに追加して、どのドメインを一緒に配置 SimSpace Weaver するかを指定します。必要な on_workers キーは、スキーマ内の名前付き workers 設定を参照している必要があります。

Example 空間ドメインをまとめて配置したスキーマスニペット

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

```
    on_workers: ["MyComputeWorkers"]
```

> **Important**
>
> - 配置グループを使用する場合:
> - x * y がワーカー数の倍数であることを確認します。
> - 配置グループの値が、まとめて配置するドメインのグリッド寸法の共通除数であることを確認します。
> - プレイスメントグループを使用しない場合:
> - 空間ドメイングリッドの 1 つの軸に、ワーカー数と等しい公約数があることを確認します。
>
> グループ配置の詳細については、「パーティショニング戦略」を参照してください。

空間ドメインに関するよくある質問

Q1. 既存のシミュレーションに別の空間ドメインを追加する方法を教えてください。

- 実行中のシミュレーションの場合 — 実行中のシミュレーションの設定は変更できません。スキーマのドメイン設定を変更し、スキーマとアプリケーションの zip をアップロードして、新しいシミュレーションを開始します。
- 新しいシミュレーションの場合 — ドメイン設定をスキーマに追加し、スキーマとアプリケーション zip をアップロードして、新しいシミュレーションを開始します。

空間ドメインのトラブルシューティング

ドメインの設定が無効な状態でシミュレーションを開始しようとすると、以下のエラーが表示される可能性があります。

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

可能性のある原因

- スキーマが、ワーカーで使用可能な量よりも多くのコンピュートリソースユニットをアプリケーションに割り当てています。
- SimSpace Weaver は、ワーカーにドメインをまとめて配置する配置を決定できません。これは、複数の空間ドメインを指定しても、ドメイングリッド間に共通の除数や倍数がない場合 (2x4 グリッドと 3x5 グリッドの間など) に発生します。

ネットワークエンドポイント

カスタムアプリケーションとサービスアプリケーションには、外部クライアントが接続できるネットワークエンドポイントを設定できます。`endpoint_config` 内の `ingress_ports` の値としてポート番号のリストを指定します。これらのポート番号はどちらも TCP と UDP です。カスタムまたはサービスアプリケーションは、 で指定したポート番号にバインドする必要があります`ingress_ports`。 は、実行時にポート番号を SimSpace Weaver 動的に割り当て、これらのポートを動的ポートにマッピングします。describe-app API は、アプリケーションが動的 (実際の) ポート番号を見つけ始めた後に呼び出すことができます。詳細については、「クイックスタートチュートリアル」の「カスタムアプリケーションの IP アドレスとポート番号を取得する」を参照してください。

```
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 app SDK バージョン 1.12.x プロジェクトでは、アプリケーション .zip ファイルとスキーマに別々のバケットを使用します。
>
> - weaver-*lowercase-project-name*-*account-number*-app-zips-*region*

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

> **Note**
>
> `endpoint_config` はカスタムアプリケーションとサービスアプリケーションのオプションプロパティです。`endpoint_config` を指定しない場合、アプリケーションにはネットワークエンドポイントがありません。

サービスドメインの設定

ドメイン設定`launch_apps_per_worker:`に が存在する場合は、それがサービスアプリを持つサービスドメインであることを示します。 は、サービスアプリ SimSpace Weaver を起動および停止します。がアプリ SimSpace Weaver を起動および停止すると、アプリはマネージドライフサイクル を持つと見なされます。 SimSpace Weaver は現在、ワーカーごとに 1 つまたは 2 つのサービスアプリの開始をサポートしています。

Example 各ワーカーで 1 つのサービスアプリケーションを起動するように設定されたドメインの例

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

Example 各ワーカーで 2 つのサービスアプリケーションを起動するように設定されたドメインの例

```
domains:
```

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

シミュレーションの最大期間

の各シミュレーション AWS SimSpace Weaver には、シミュレーションを実行できる最大時間を指定する最大時間設定があります。シミュレーションの開始時に、最大期間をパラメータとして指定します。StartSimulation アプリケーションプログラミングインターフェイス (API) にはオプションのパラメータ MaximumDuration があります。パラメータの値は、分 (m または M)、時間 (h または H)、日 (d または D) です。たとえば、1h または 1H は 1 時間を意味します。 SimSpace Weaver はこの制限に達すると、シミュレーションを停止します。

最大値

MaximumDuration の有効な最大値は 14D、またはそれと同等の時間 (336H) または分 (20160M) で表されます。

デフォルト値

MaximumDuration パラメータはオプションです。値を指定しない場合、 は の値 SimSpace Weaver を使用します14D。

最小値

MaximumDuration の有効値の最小値は、数値的には 0 と等価な値です。例えば、0M、0H、0D の値はすべて数値的には 0 と等価です。

最大期間に最小値を指定すると、シミュレーションは STOPPING状態に達するとすぐに STARTED状態に移行します。

コンソールを使用したシミュレーションの開始

SimSpace Weaver コンソールでシミュレーションを開始するときに最大期間の値を指定できます。[シミュレーションの開始] を選択するときに、[シミュレーション設定] フォームの [最大期間] フィールドに値を入力します。

> **Important**
>
> [最大期間] に値を指定しない場合は、 SimSpace Weaver はデフォルトの値 (14D) を使用します。

最大期間に達したシミュレーションのステータス

が最大期間に達するシミュレーション SimSpace Weaver を自動的に停止すると、シミュレーションのステータスは `STOPPING` (進行中の場合) または になりますSTOPPED。SimSpace Weaver コンソールでは、シミュレーションのターゲットステータスはまだ `STARTED` です。これは、ユーザーが最後に要求した状態であったためです。

アプリケーション開発

SimSpace Weaver 開発では、シミュレーションが の Amazon Linuxで実行されるため、アプリケーションを構築するためのAmazon Linux 2 (AL2)環境が必要ですAWS Cloud。を使用している場合はWindows、 SimSpace Weaver アプリケーション SDK のスクリプトを使用して、 SimSpace Weaver アプリケーションの構築に必要な依存関係AL2で実行されるDockerコンテナを作成して起動できます。Windows Subsystem for Linux (WSL) またはネイティブ AL2 システムを使用して AL2 環境を起動することもできます。詳細については、「のローカル環境をセットアップする SimSpace Weaver」を参照してください。

> **Note**
>
> ローカル開発環境の設定にかかわらず、 AWS クラウドで実行するためにアプリケーションをアップロードして実行すると、アプリケーションは Docker コンテナ内で実行されます。アプリケーションはホストオペレーティングシステムに直接アクセスできません。

SimSpace Weaver アプリケーションの一般的なフロー

1. アプリケーションを作成します。
2. ループ:
   a. `Transaction` を作成して更新を開始します。
      - シミュレーションが停止する場合はループを終了します。
   b. サブスクリプションと所有エンティティイベントを処理します。
   c. シミュレーションを更新します。
   d. `Transaction` をコミットして更新を終了します。
3. アプリケーションを破棄します。

空間アプリケーション

各空間アプリケーションには、シミュレーション世界の空間領域である所有権エリアがあります。空間アプリケーションの所有権エリアにあるエンティティは、アプリケーションに割り当てられたパーティションに保存されます。1 つの空間アプリケーションが、割り当てられたパーティション内のすべてのエンティティに対して完全な所有権 (読み取り権と書き込み権限) を持ちます。他のアプリケーションはそれらのエンティティに書き込むことはできません。空間アプリケーションはエンティティの状態を進めます。各空間アプリケーションはパーティションを 1 つだけ所有します。SimSpace Weaver はエンティティの空間位置を使用してインデックスを作成し、空間アプリケーションパーティションに割り当てます。

SimSpace Weaver アプリケーション SDK はサンプルアプリケーションを提供します。サンプルアプリケーションの空間アプリケーションのソースコードは、次のフォルダにあります (オペレーティングシステムには正しいパス区切り文字を使用してください) 。

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

カスタムアプリケーション

シミュレーションを操作するカスタムアプリケーションを作成して使用します。

カスタムアプリケーションは以下を実行できます

- エンティティを作成する
- 他のパーティションをサブスクライブする

- 変更をコミットする

カスタムアプリケーションの一般的なフロー

1. アプリケーションを作成します。
2. シミュレーション内の特定のリージョンをサブスクライブします。
   a. `Transaction` を作成して最初の更新を開始します。
   b. 特定のリージョンのサブスクリプションを作成します。
   c. `Transaction` をコミットして最初の更新を終了します。
3. ループ:
   a. `Transaction` を作成して更新を開始します。
      - シミュレーションが停止する場合はループを終了します。
   b. 状態の変更を処理します。
   c. `Transaction` をコミットして更新を終了します。
4. アプリケーションを破棄します。

カスタムアプリケーションがエンティティを作成したら、エンティティを空間ドメインに転送して、エンティティがシミュレーション内に空間的に存在するようにする必要があります。 はエンティティの空間位置 SimSpace Weaver を使用して、エンティティを適切な空間アプリケーションパーティションに配置します。エンティティを作成したカスタムアプリケーションは、空間ドメインに転送した後でそのエンティティを更新または削除することはできません。

SimSpace Weaver アプリケーション SDK はサンプルアプリケーションを提供します。サンプルアプリケーションに含まれるカスタムアプリケーションを、独自のカスタムアプリケーションのモデルとして使用できます。サンプルアプリケーションのビューアプリ (カスタムアプリ) のソースコードは、次のフォルダにあります (オペレーティングシステムには正しいパス区切り文字を使用します) 。

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

クライアントアプリケーションの開発

クライアントをシミュレーションに接続すべき理由は、以下のとおりです。

- 都市規模のシミュレーションにリアルタイムの交通情報を注入します。
- human-in-the-loop シミュレーションを作成して、人間のオペレーターがシミュレーションの一部を制御します。
- トレーニングシミュレーションなどで、ユーザーがシミュレーションを操作できるようにします。

これらの例のカスタムアプリケーションが、シミュレーションの状態と外部とのインターフェースとして機能します。クライアントはカスタムアプリケーションに接続してシミュレーションを操作します。

SimSpace Weaver は、クライアントアプリケーションとそのカスタムアプリケーションとの通信を処理しません。クライアントアプリケーションの設計、作成、運用、セキュリティ、およびクライアントアプリケーションとカスタムアプリケーションとの通信については、お客様の責任となります。SimSpace Weaver はクライアントが接続できるように、各カスタムアプリケーションの IP アドレスとポート番号のみを公開します。

SimSpace Weaver アプリケーション SDK は、サンプルアプリケーション用のクライアントを提供します。これらのクライアントは、独自のクライアントアプリケーションのモデルとして使用できます。サンプルアプリケーションクライアントのソースコードは、以下のフォルダで確認できます。

Docker

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

WSL

> **Important**
>
> 便宜上、これらの指示を使用します。これらは Windows Subsystem for Linux (WSL) で使用するためのもので、サポートされていません。詳細については、「の ローカル環境をセットアップする SimSpace Weaver」を参照してください。

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

サンプルアプリケーションクライアントの構築と使用の詳細については、「」のチュートリアルを参照してください の開始方法 SimSpace Weaver。

カスタムアプリケーションの IP アドレスとポート番号を取得する

シミュレーションを表示するには、カスタムアプリケーションを作成し、クライアントで接続します。詳細については、「」のチュートリアルを参照してください[の開始方法 SimSpace Weaver](#)。次の手順を使用して、カスタムアプリケーションの IP アドレスとポート番号を取得できます。オペレーティングシステムに適したパス区切り文字 (\Windows や Linux など) / を使用します。

IP アドレスとポート番号を取得する

1. ListSimulations API を使用してシミュレーション名を取得します。

```
aws simspaceweaver list-simulations
```

出力例:

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

}
```

2. DescribeSimulation API を使用して、シミュレーション内のドメインのリストを取得します。

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

出力の `LiveSimulationState` セクションで `Domains` セクションを探します。

出力例:

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

3. ListApps API を使用して、ドメイン内のカスタムアプリケーションのリストを取得します。たとえば、サンプルプロジェクトのビュー (カスタム) アプリケーションのドメイン名は ですMyViewDomain。出力でアプリケーション名を探します。

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

出力例:

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

4. DescribeApp API を使用して IP アドレスとポート番号を取得します。サンプルプロジェクトでは、ドメイン名は MyViewDomain で、アプリケーション名は ViewApp です。

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

IP アドレスとポート番号は出力の `EndpointInfo` ブロックに含まれます。IP アドレスは `Address` の値で、ポート番号は `Actual` の値です。

出力例:

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

> **Note**
>
> `Declared` の値はアプリケーションコードのバインド先となるポート番号です。 の値は、 がクライアントに SimSpace Weaver 公開して app に接続するポート番号`Actual`です。 は`Declared`ポートを`Actual`ポートに SimSpace Weaver マッピングします。

Unreal Engine ビュークライアントの起動

次に移動します。

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

1. 以下のいずれかのコマンドを実行します:
   - Docker: `python quick-start.py`
   - WSL: `python quick-start.py --al2`
2. IP アドレスと「実際の」ポート番号を取得します。これらは、quick-start.py」を実行するコンソール出力に表示されるか、「」の手順に従って取得します[カスタムアプリケーションの IP アドレスとポート番号を取得する](#)。
3. 次に移動します。

   ```
   sdk-folder/Clients/TCP/UnrealClient/lib
   ```

4. 次のコマンドを実行して NNG ライブラリを構築します。

   ```
   cmake -S . -B build
   cmake --build build --config RelWithDebInfo
   cmake --install build
   ```

5. テキストエディタで `view_app_url.txt` を開きます。
6. ビューアプリケーションの IP アドレスとポート番号で URL を更新します: `tcp://ip-address:actual-port-number` (`tcp://198.51.100.135:1234` のようになります)。
7. Unreal エディタで、再生を選択します。

トラブルシューティング

- NNG CMake のインストールステップが「管理者権限が必要な場合があります」で失敗します。

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

```
  build/cmake_install.cmake:73 (include)
```

- 解決策： UnrealClient/lib ディレクトリに `nng.lib`または `nng.so`が存在する場合、このエラーは無視しても問題ありません。そうでない場合は、管理者権限を持つターミナルで cmake ビルドコマンドを実行してみてください。

- CMake to find a package configuration file provided by nng」：

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

- 解決策： CMake で`Findnng.cmake`ファイルを見つけることができません。CMake を使用して構築する場合は、引数 を追加します`-DTHIRD_PARTY_LIB_PATH sdk-folder/ThirdParty`。CMake ビルドを再実行する前に、 `Findnng.cmake` ファイルがまだ `ThirdParty` ディレクトリにあることを確認します。

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

でのローカル開発 SimSpace Weaver

SimSpace Weaver アプリケーションをローカルにデプロイして、迅速なテストとデバッグを行うことができます。

要件

- 「のセットアップ SimSpace Weaver」のステップを完了します。

トピック

- ステップ 1: ローカルシミュレーションを起動する
- ステップ 2: ローカルシミュレーションを表示する
- ステップ 3: ローカルシミュレーションを停止する (Windows ではオプション)
- でのローカル開発のトラブルシューティング SimSpace Weaver

ステップ 1: ローカルシミュレーションを起動する

1. に移動

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

2. 次のコマンドを実行して、シミュレーションをローカルでビルドして起動します。

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

このスクリプトは以下の処理を実行します。

1. プロジェクトをビルドします。
   - `quick-start.py` は、 build.py定義された `build_project`関数を呼び出します。このステップはプロジェクトによって異なります。PathfindingSample には、CMake が使用されます。CMake および Docker コマンド。 は build.py://www.com にあります。
2. ローカルシミュレーションを起動する
   - スクリプトは、スキーマで定義された空間パーティションごとに 1 つのローカルプロセスを起動します。
   - スクリプトは、スキーマで定義されたカスタムアプリケーションごとに 1 つのプロセスを起動します。
   - 空間アプリケーションが最初に起動され、次にカスタムアプリケーションが続きます。それぞれがスキーマに表示される順序で起動されます。

> **Important**
>
> コンソールの SSH セッションなど、GUI をサポートしていない環境で を起動する場合は、`--noappwindow`オプションを使用してすべての出力を現在のターミナルにリダイレクトします。

> **Important**
>
> Linux ユーザーの場合、スクリプトはシステムに `xterm` コマンドがあることを前提としています。Linux ディストリビューションに `xterm` コマンドがない場合は、 `--noappwindow`オプションを使用してすべての出力を現在のターミナルにリダイレクトします。

- -h, --help
  - これらのパラメータを一覧表示します。
- --clean
  - ビルドする前に、ビルドディレクトリの内容を削除します。
- --nobuild
  - プロジェクトの再構築をスキップします。
- --noappwindow
  - アプリごとに新しいウィンドウを開かないでください。代わりに、stdout を現在のターミナルにリダイレクトします。
- --logfile
  - コンソール出力をログファイルに書き込みます。
- --consoleclient
  - 設定にリストされているコンソールクライアントを自動的に接続します。
- --スキーマスキーマ
  - この呼び出しで使用するスキーマ。デフォルトは config.py://www.jp の「SCHEMA」です。

ステップ 2: ローカルシミュレーションを表示する

ローカルシミュレーションを表示するには、SimSpaceWeaverAppSDKDistributable に含まれている任意のクライアントを使用できます。サンプルクライアントの構築と使用の詳細については、「」のチュートリアルを参照してください[の開始方法 SimSpace Weaver](#)。

ローカルシミュレーションのビューアプリケーションに接続するには、クライアントの IP アドレスとポート番号を更新する必要があります。SimSpace Weaver Local では常に以下の値を使用します。

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

選択したクライアントによっては、IP アドレスとポート番号を以下のように更新できます。

- Unreal — `view_app_url.txt` の 1 行目の URL を変更します
- コンソール — IP アドレスとポート番号 URL をパラメータとしてクライアントを起動します

ステップ 3: ローカルシミュレーションを停止する (Windows ではオプション)

> **Note**
> このステップは Linux では必須ですが、Windows ではオプションです。

1. 以下に移動します。

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

2. 次のコマンドを実行してローカルシミュレーションを停止し、共有メモリリソースを削除します。

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

このスクリプトは以下の処理を実行します。

- ローカルプロセスを停止します。
- 共有メモリオブジェクトを削除します (Linux でのみ必要) 。

stop-and-delete.py 「https://https://https

- -h, --help
  - これらのパラメータを一覧表示します。
- --stop
  - プロセスの停止のみを試みます。
- --delete
  - 共有メモリリソースの削除のみを試みます。
- --process

  - 停止するプロセスの名前。プロセス名がスキーマのパッケージ名と一致しない場合に使用します。
- --スキーマスキーマ
  - この呼び出しで使用するスキーマ。デフォルトは、config.py://www.jp」の「SCHEMA」の値です。

でのローカル開発のトラブルシューティング SimSpace Weaver

- Linux: xterm コマンドが見つかりません/開くことができません
  - ローカルスクリプトは、Linux で を実行するときに xterm コマンドが存在することを前提としています。xterm コマンドがない場合、または GUI をサポートしていない環境で実行している場合は、クイックスタートスクリプトを実行するときに `--noappwindow`オプションを使用します。
- アプリウィンドウが開いていません。
  - これは、ローカルシミュレーションがすぐにクラッシュしたときに発生します。クラッシュ後にコンソール出力を表示するには、クイックスタートスクリプトを実行するときに `--noappwindow`または `--logfile`オプションを使用します。
- ビューアプリの起動後またはビュークライアントの接続後にシミュレーションがティックしません。
  - –`noappwindow` オプションを使用して を実行すると、通常、このような問題が解決されます。それ以外の場合は、数回再起動しても成功します (ただし、はるかに低いレートで) 。

AWS SimSpace Weaver アプリケーション SDK

SimSpace Weaver アプリケーション SDK にはAPIs が用意されています。 SimSpace Weaver これには以下の名前空間が含まれます。

- API — API のコア定義とその使用方法

以下のライブラリとリンクします。

- `libweaver_app_sdk_cxx_v1_full.so`

> **Important**
>
> ライブラリは、 AWS クラウドでアプリケーションを実行するとダイナミックリンクに使用できます。アプリケーションと一緒にアップロードする必要はありません。

> **Note**
>
> SimSpace Weaver アプリケーション SDK APIsシミュレーション内のデータを制御します。これらの APIsは、 SimSpace Weaver のサービスリソース (シミュレーション、アプリケーション、クロックなど) を制御する SimSpace Weaver サービス APIs とは別です AWS。詳細については、「SimSpace Weaver API リファレンス」を参照してください。

トピック

- API メソッドは Result を返します。
- 最上位レベルでのアプリケーション SDK とのやり取り
- シミュレーション管理
- サブスクリプション
- エンティティ
- エンティティイベント
- Result とエラー処理
- ジェネリックとドメインタイプ
- その他のアプリケーション SDK 操作

API メソッドは Result を返します。

ほとんどの SimSpace Weaver API 関数には戻り値型 がありますAws::WeaverRuntime::Result<T>。関数が正常に実行されると、Result には T が含まれます。それ以外の場合、Result には Rust App SDK からのエラーコードを表す Aws::WeaverRuntime::ErrorCode が含まれます。

Example 例

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

この方法:

- `BeginUpdate()` が正常に実行された場合は `Transaction` を返します。
- `BeginUpdate()` に失敗した場合は `Aws::WeaverRuntime::ErrorCode` を返します。

最上位レベルでのアプリケーション SDK とのやり取り

ライフサイクル

- SimSpace Weaver アプリケーション SDK はアプリケーションのライフサイクルを管理します。アプリケーションのライフサイクル状態を読み取ったり書き込んだりする必要はありません。

パーティション

- `Result <PartitionSet> AssignedPartitions(Transaction& txn);` を使用して、所有しているパーティションを取得します。
- `Result <PartitionSet> AllPartitions(Transaction& txn);` を使用して、シミュレーション内のすべてのパーティションを取得します。

シミュレーション管理

このセクションでは、一般的なシミュレーション管理タスクのソリューションについて説明します。

トピック

- シミュレーションを開始する
- シミュレーションを更新する
- シミュレーションを終了する

シミュレーションを開始する

`CreateApplication()` を使用して、アプリケーションを作成します。

Example 例

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

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

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

    return 1;
}

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

シミュレーションを更新する

以下の `BeginUpdate` 関数を使用してアプリケーションを更新します。

- `Result<Transaction> BeginUpdate(Application& app)`
- `Result<bool> BeginUpdateWillBlock(Application& app)` — `BeginUpdate()` がブロックするかしないかを伝えます。

`Result<void> Commit(Transaction& txn)` を使用して、以下の変更をコミットします。

Example 例

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

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

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

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

    return Success();
}
```

シミュレーションを終了する

`Result<void> DestroyApplication(Application&& app)` を使用して、アプリケーションとシミュレーションを終了します。

他のアプリケーションは、`BeginUpdateWillBlock()` または `BeginUpdate()` への呼び出しから `ErrorCode::ShuttingDown` を受信すると、シミュレーションがシャットダウン中であることを認識します。アプリケーションが `ErrorCode::ShuttingDown` を受信すると、`Result<void> DestroyApplication(Application&& app)` 呼び出しを行って自動的に終了できます。

Example 例

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

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

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

```
}
```

> **Important**
>
> Api::Commit() の後にのみ Result<void> DestroyApplication(Application&& app) を呼び出しできます。更新中にアプリケーションを破棄すると、未定義の動作が発生する可能性があります。

> **Important**
>
> プログラムが終了する前に DestroyApplication() を呼び出して、アプリケーションの正常終了レポートを確認する必要があります。
> プログラムの終了時に DestroyApplication() 呼び出しに失敗すると、ステータスは FATAL とみなされます。

サブスクリプション

サブスクリプション領域とドメイン ID を使用してサブスクリプションを作成します。ドメイン ID は、そのサブスクリプション領域を所有するドメインを表します。BoundingBox2F32 はサブスクリプション領域を表します。以下の関数を使用してサブスクリプションを作成します。

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

Example 例

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


    Api::DomainId spatialDomainId;

    for (const Api::Partition& partition : partitionSet.partitions)
    {
        if (partition.domain_type == Api::DomainType::Spatial)
        {
            /**
```

```
            * Get the spatial domain ID.
            */
            spatialDomainId = partition.domain_id;
            break;
        }
    }

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

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

    return Success();
}
```

`CreateSubscriptionBoundingBox2F32()` から返送された `Api::SubscriptionHandle` を使用してサブスクリプションを変更できます。以下の関数の引数として渡します。

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

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

エンティティ

`Store`と `Load` API を呼び出すには、`CreateEntity()`、またはエンティティがアプリケーションのサブスクリプション領域に入った際の所有権変更イベントから返された、`Result<Api::Entity>` の `Api:Entity` を使用します (詳細については、「エンティティイベント」を参照してください)。これらの API で使用できるように、`Api::Entity` オブジェクトを追跡することをお勧めします。

トピック

- エンティティを作成する
- エンティティを空間ドメインに転送します。

- エンティティフィールドデータの書き込みおよび読み取り
- エンティティの位置を保存する
- エンティティの位置をロードする

エンティティを作成する

CreateEntity() を使用して、エンティティを作成します。この関数に渡す、Api::TypeId の意味を定義します。

```
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**
>
> 0 ~ 511 の Api::BuiltinTypeId の値は予約されています。エンティティ TypeID (この例では k_entityTypeId) の値は 512 以上である必要があります。

エンティティを空間ドメインに転送します。

カスタムアプリケーションまたはサービスアプリケーションでエンティティを作成したら、そのエンティティを空間ドメインに転送して、そのエンティティをシミュレーションに空間的に存在させる必要があります。空間ドメイン内のエンティティは、他のアプリケーションで読み取ったり、空間アプリケーションで更新したりできます。ModifyEntityDomain() API を使用してエンティティを空間ドメインに転送します。

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

DomainId が呼び出し元のアプリケーションに割り当てられた Partition と一致しない場合、DomainId は DomainType::Spatial Domain のものである必要があります。新しい Domain への所有権の転送は、Commit(Transaction&&) の間に行われます。

パラメータ

txn

現在の Transaction。

entity

Domain の変更ターゲット Entity。

domainId

Entity の送信先 Domain の DomainId。

この API は、エンティティドメインが正常に変更された場合に Success を返します。

エンティティフィールドデータの書き込みおよび読み取り

エンティティデータフィールドはすべて BLOB タイプです。1 つのエンティティには最大 1,024 バイトのデータを書き込むことができます。BLOB のサイズを大きくするとパフォーマンスが低下するため、BLOB はできるだけ小さくすることをお勧めします。BLOB に書き込むときは、データとその長さに SimSpace Weaver ポインタを渡します。BLOB から読み取ると、 SimSpace Weaver は読み取るポインタと長さを提供します。すべての読み取りは、アプリケーションが Commit() を呼び出す前に完了する必要があります。読み取りの呼び出しから返されたポインタは、アプリケーションが Commit() を呼び出すと無効になります。

> **Important**
>
> - Commit() の後のキャッシュされた BLOB ポインタからの読み取りはサポートされないため、シミュレーションが失敗する可能性があります。
> - 読み取りの呼び出しから返された BLOB ポインタへの書き込みはサポートされないため、シミュレーションが失敗する可能性があります。

トピック

- エンティティのフィールドデータを保存する

- エンティティのフィールドデータをロードする
- 削除されたエンティティのフィールドデータの読み込み

エンティティのフィールドデータを保存する

以下の例は、アプリケーションが所有するエンティティのフィールドデータを保存する (ステートファブリックに書き込む) 方法を示しています。以下の例では、以下の関数を使用します。

```
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` パラメータは渡されたデータのデータタイプを表します。

`Api::TypeId keyTypeId` パラメータは `Api::BuiltinTypeId` から対応する `Api::TypeId` を受け取る必要があります。適切な変換がない場合は、`Api::BuiltinTypeId::Dynamic` を使用できます。

複雑なデータタイプの場合は、`Api::BuiltInTypeId::Dynamic` を使用します。

> **Note**
>
> `FieldIndex index` の値はゼロより大きい必要があります。値 0 はインデックスキー専用です (`StoreEntityIndexKey()` を参照してください)。

Example プリミティブデータタイプを使用する例

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

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

```
{
    bool value = true;

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

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

Example struct を使用してデータを保持する例

```
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));
}
```

エンティティのフィールドデータをロードする

以下の例は、エンティティのフィールドデータをロードする (ステートファブリックから読み取る) 方法を示しています。以下の例では、以下の関数を使用します。

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

`Api::TypeId keyTypeId` パラメータは `Api::BuiltinTypeId` から対応する `Api::TypeId` を受け取る必要があります。適切な変換がない場合は、`Api::BuiltinTypeId::Dynamic` を使用できます。

> **Note**
>
> `FieldIndex` インデックスの値は 0 より大きい必要があります。値 0 はインデックスキー専用です (`StoreEntityIndexKey()` を参照してください)。

Example プリミティブデータタイプを使用する例

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

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

    WEAVERRUNTIME_TRY(Api::LoadEntityField(
        transaction,
```

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

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

Example struct を使用してデータを保持する例

```
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);
}
```

削除されたエンティティのフィールドデータの読み込み

アプリケーションの所有権とサブスクリプション領域から削除されたエンティティのエンティティフィールドデータをロードする (ステートファブリックから読み取る) ことはできま

せん。以下の例では、`Api::ChangeListAction::Remove` の結果としてエンティティの `Api::LoadIndexKey()` を呼び出しているため、エラーが発生します。2 番目の例は、エンティティデータをアプリケーションに直接保存して読み込む正しい方法を示しています。

Example 誤ったコードの例

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

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

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

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

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

    }

    /* ... */
```

```
}
```

Example アプリケーションにエンティティデータを保存して読み込む正しい方法の例

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

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

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

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

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

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

                /**
```

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

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

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

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

    /* ... */
}
```

エンティティの位置を保存する

整数データ構造を使用してエンティティの位置を保存 (ステートファブリックへの書き込み) できます。以下の例では、以下の関数を使用します。

```
Result<void> StoreEntityIndexKey(
    Transaction& txn,
    const Entity& entity,
```

```
    TypeId keyTypeId,
    std::int8_t* src,
    std::size_t length)
```

> **Note**
>
> 以下の例に示すように、`Api::BuiltinTypeId::Vector3F32` から `Api::StoreEntityIndexKey()` を指定する必要があります。

Example 配列を使用して位置を表す例

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

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

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

Example struct を使用して位置を表す例

```
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));
}
```

エンティティの位置をロードする

整数データ構造を使用してエンティティの位置をロードする (ステートファブリックから読み取る) ことができます。以下の例では、以下の関数を使用します。

> **Note**
>
> 以下の例に示すように、`Api::BuiltinTypeId::Vector3F32` から `Api::LoadEntityIndexKey()` を指定する必要があります。

Example 配列を使用して位置を表す例

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

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

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

```
}
```

Example struct を使用して位置を表す例

```
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);
}
```

エンティティイベント

SimSpace Weaver アプリケーション SDK で次の関数を使用して、すべての所有権イベントとサブスクリプションイベントを取得できます。

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

コールバック駆動型のエンティティイベント処理が必要な場合は、 SimSpace Weaver デモフレームワークを使用できます。詳細については、以下のヘッダーファイルを参照してください。

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

独自のエンティティイベント処理を作成することもできます。

トピック

- 所有エンティティのイベントを繰り返し処理します。
- 登録したエンティティのイベントを繰り返し処理します。
- エンティティの所有権変更イベントを繰り返し処理します。

所有エンティティのイベントを繰り返し処理します。

`OwnershipChanges()` を使用して、所有エンティティ (アプリケーションの所有領域にあるエンティティ) のイベントのリストを取得します。関数には以下のような署名があります。

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

次に、以下の例に示すように、ループを使用してエンティティを繰り返し処理します。

Example 例

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

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

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

```
    }
}
```

イベントタイプ

- None — エンティティが領域内にあり、その位置とフィールドのデータは変更されていません。
- Remove — エンティティが領域から削除されました。
- Add — エンティティが領域に追加されました。
- Update — エンティティが領域内にあり、変更されました。
- Reject — アプリケーションは領域からエンティティを削除できませんでした。

> **Note**
>
> Reject イベントが発生した場合、アプリケーションは次のティックで転送を再試行します。

登録したエンティティのイベントを繰り返し処理します。

AllSubscriptionEvents() を使用して、登録済みエンティティ (アプリケーションのサブスクリプション領域にあるエンティティ) のイベントリストを取得します。関数には以下のような署名があります。

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

次に、以下の例に示すように、ループを使用してエンティティを繰り返し処理します。

Example 例

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

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

    switch (action)
```

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

イベントタイプ

- None — エンティティが領域内にあり、その位置とフィールドのデータは変更されていません。
- Remove — エンティティが領域から削除されました。
- Add — エンティティが領域に追加されました。
- Update — エンティティが領域内にあり、変更されました。
- Reject — アプリケーションは領域からエンティティを削除できませんでした。

> **Note**
>
> Reject イベントが発生した場合、アプリケーションは次のティックで転送を再試行します。

エンティティの所有権変更イベントを繰り返し処理します。

エンティティが所有領域とサブスクリプション領域の間を移動するイベントを取得するには、現在と以前のエンティティの所有領域とサブスクリプションイベントの変化を比較します。

これらのイベントは、以下を読むことで処理できます。

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

その後、変更内容を以前に保存したデータと比較できます。

以下の例は、エンティティの所有権の変更イベントを処理する方法を示しています。この例では、サブスクライブされたエンティティと所有されたエンティティの間を (どちらの方向でも) 移行するエンティティについて、所有権の削除/追加イベントが最初に発生し、次のティックでサブスクリプションの削除/追加イベントが発生することを前提としています。

Example 例

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

    static EntityIdsByAction m_entityIdsByPreviousOwnershipAction;

    EntityIdSetByAction entityIdSetByAction;

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

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

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

            {
```

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

    EntityIdsByAction entityIdsByAction;

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

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

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

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

```
    }

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

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


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

                entityIdSetBySubscriptionAction[
                Api::ChangeListAction::Remove].end())
        {
            fromSubscribedToOwnedEntities.push_back(id);
```

```
        }
    }

    m_entityIdsByPreviousOwnershipAction = entityIdsByOwnershipAction;

    return Success();
}
```

Result とエラー処理

この `Aws::WeaverRuntime::Result<T>` クラスはサードパーティの `Outcome` ライブラリを使用しています。以下のパターンを使用して、`Result` を確認し、API コールによって返されるエラーを発見できます。

```
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 制御文マクロ

戻り値タイプ `Aws::WeaverRuntime::Result<T>` を持つ関数内では、前のコードパターンの代わりに `WEAVERRUNTIME_TRY` マクロを使用できます。マクロは渡された関数を実行します。渡された関数が失敗すると、マクロは囲んでいる関数にエラーを返送させます。渡された関数が成功すると、

実行は次の行に進みます。以下の例は、以前の DoBeginUpdate() 関数の再記述を示しています。このバージョンでは、if-else 制御構造の代わりに WEAVERRUNTIME_TRY マクロを使用しています。関数の戻り値タイプは Aws::WeaverRuntime::Result<void> です。

```
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() に失敗した場合、マクロは障害発生時に早くDoBeginUpdate() を返送します。WEAVERRUNTIME_EXPECT_ERROR マクロを使用して BeginUpdate() から Aws::WeaverRuntime::ErrorCode を取得できます。以下の例は、障害発生時に Update() 関数が DoBeginUpdate() を呼び出してエラーコードを取得する方法を示しています。

```
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);

    }
}
```

戻り値タイプを Update() から Aws::WeaverRuntime::Result<void> に変更することで、BeginUpdate() からのエラーコードを Update() を呼び出す関数に利用できるようにすることができます。この手順を繰り返して、エラーコードを呼び出しスタックのさらに下位に送り続けることができます。

ジェネリックとドメインタイプ

SimSpace Weaver アプリケーション SDK は、単精度データ型 Api::Vector2F32と Api::BoundingBox2F32、および倍精度 Api::Vector2F64 と を提供しますApi::BoundingBox2F64。これらのデータタイプは受動的なデータ構造で、便利な方法はありません。API は Api::Vector2F32 および Api::BoundingBox2F32 のみを使用することに注意してください。これらのデータタイプを使用してサブスクリプションを作成および変更できます。

SimSpace Weaver デモフレームワークは、 AzCore Vector3と を含む数学ライブラリの最小バージョンを提供しますAabb。詳細については、以下のヘッダーファイルを参照してください。

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

その他のアプリケーション SDK 操作

トピック

- AllSubscriptionEvents および OwnershipChanges は前回の呼び出しからのイベントを含みます
- SubscriptionChangeList の処理後に読み取りロックを解除する
- テスト用のスタンドアロンアプリケーションインスタンスを作成する

AllSubscriptionEvents および OwnershipChanges は前回の呼び出しからのイベントを含みます

Api::AllSubscriptionEvents() および Api::OwnershipChanges() への呼び出しの戻り値が含むのは、最後の呼び出しからのイベントであり、最後のティックからのイベントではありません。以下の例では、secondSubscriptionEvents および secondOwnershipChangeList は空です。なぜなら、関数は最初の呼び出しの直後に呼び出されるからです。

10 ティック待ってから `Api::AllSubscriptionEvents()` および `Api::OwnershipChanges()` を呼び出した場合、その結果として (最後のティックではなく) 最後の 10 ティックからのイベントと変更の両方が含まれます。

Example 例

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

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

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

> **Note**
>
> 関数 `AllSubscriptionEvents()` は実装されていますが、関数 `SubscriptionEvents()` は実装されていません。

SubscriptionChangeList の処理後に読み取りロックを解除する

更新を開始すると、コミットされたデータ用の共有メモリセグメントが他のパーティションに前回のティックで残ります。これらの共有メモリセグメントはリーダーによってロックされている可能性があります。すべてのリーダーがロックを解除するまで、アプリケーションは完全にコミットできません。最適化のため、アプリケーションは `Api::SubscriptionChangelist` アイテムを処理した後にロックを解除するための `Api::ReleaseReadLeases()` 呼び出しを行う必要があります。これに

より、コミット時の競合が減少します。デフォルトでは `Api::Commit()` は読み取りリースを解放しますが、サブスクリプションの更新を処理した後に手動でリリースするのがベストプラクティスです。

Example 例

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

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

    WEAVERRUNTIME_EXPECT(Api::ReleaseReadLeases(transaction));

    ...
}
```

テスト用のスタンドアロンアプリケーションインスタンスを作成する

`Api::CreateStandaloneApplication()`を使用して、スタンドアロンアプリケーションを作成し、実際のシミュレーションでコードを実行する前にアプリケーションのロジックをテストできます。

Example 例

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

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

    ...
}
```

AWS SimSpace Weaver デモフレームワーク

AWS SimSpace Weaver デモフレームワーク (デモフレームワーク) は、 SimSpace Weaver アプリケーションの開発に使用できるユーティリティのライブラリです。

デモフレームワークには以下のものが含まれます

- 使用したり調べたりできるコードサンプルとプログラミングパターン
- シンプルなアプリケーションの開発を効率化する抽象化とユーティリティ関数
- SimSpace Weaver アプリケーション SDK の実験的な機能をテストする簡単な方法

パフォーマンスを向上させるために、API への低レベルアクセスを備えた SimSpace Weaver アプリケーション SDK を設計しました。 SimSpace Weaver APIs これとは対照的に、より高いレベルの抽象化と、 SimSpace Weaver を使いやすくする API へのアクセスが可能なデモフレームワークを設計しました。使いやすさのコストは、 SimSpace Weaver アプリケーション SDK を直接使用するよりもパフォーマンスが低くなります。パフォーマンスの低下を許容できるシミュレーション (リアルタイムのパフォーマンス要件がないシミュレーションなど) は、デモフレームワークの使用に適している場合があります。デモフレームワークは完全なツールキットではないため、複雑なアプリケーションに SimSpace Weaver アプリケーション SDK のネイティブ機能を使用することをお勧めします。

デモフレームワークには以下が含まれます

- 以下をサポートおよび実証する作業用コードサンプル:
  - アプリケーションフローの管理
  - コールバック主導型のエンティティイベント処理
- 以下のサードパーティー製ユーティリティライブラリのセット:
  - spdlog (ログ記録ライブラリ)
  - AZCore (数学ライブラリ) の最小バージョンで、以下のみを含むもの:
    - Vector3
    - Aabb
  - cxxopts (コマンドラインオプションパーサーライブラリ)
- に固有のユーティリティ関数 SimSpace Weaver

デモフレームワークは、ライブラリ、ソースファイル、CMakeLists で構成されています。ファイルは、 SimSpace Weaver アプリケーション SDK 配布パッケージに含まれています。

Service Quotas との連携

このセクションでは、 のサービスクォータを使用する方法について説明します SimSpace Weaver。Quotas はリミットとも呼ばれます。Service Quotas のリストについては、「SimSpace Weaver エンドポイントとクォータ」を参照してください。このセクションの API はアプリケーション API セットの一部です。アプリケーション API はサービス API とは異なります。アプリ APIsは SimSpace Weaver アプリ SDK の一部です。アプリケーション API のドキュメントは、ローカルシステムのアプリケーション SDK フォルダで確認できます。

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

トピック

- アプリケーションの制限を取得する
- アプリケーションが使用しているリソース量を取得する
- メトリクスをリセットする
- 制限の超過
- メモリの不足
- ベストプラクティス

アプリケーションの制限を取得する

RuntimeLimits アプリケーション SDK を使用してアプリケーションの制限をクエリできます。

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

パラメータ

Application& アプリケーション

アプリケーションへの参照。

LimitType タイプ

以下の制限タイプを含む列挙値。

```
enum LimitType {
    Unset = 0,
```

```
    EntitiesPerPartition = 1,
    RemoteEntityTransfers = 2,
    LocalEntityTransfers = 3
};
```

以下の例では、エンティティ数の制限をクエリします。

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

アプリケーションが使用しているリソース量を取得する

RuntimeMetrics アプリケーション SDK を呼び出して、アプリケーションが使用しているリソースの量を取得できます。

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

パラメータ

Application& アプリケーション

アプリケーションへの参照。

API は、メトリクスを含む struct への参照を返します。カウンターメトリクスには現在の合計値が含まれ、増加のみします。ゲージメトリクスには増減する値が含まれます。アプリケーションランタイムは、イベントによって値が増加するたびにカウンターを更新します。ランタイムは API を呼び出したときのみゲージを更新します。 SimSpace Weaver はリファレンスがアプリケーションの存続期間中有効であることを保証します。API を繰り返し呼び出しても、参照は変更されません。

```
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
}
```

メトリクスをリセットする

ResetRuntimeMetrics アプリケーション SDK は `AppRuntimeMetrics` struct 内の値をリセットします。

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

次の例では、アプリケーション内での ResetRuntimeMetrics の呼び出し方法を示します。

```
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));
}
```

制限の超過

アプリケーションの API コールが制限を超えると、エンティティ転送を除き、`ErrorCode::CapacityExceeded` が返されます。 SimSpace Weaver はCommit 操作および BeginUpdate アプリケーション API 操作の一部としてエンティティ転送を非同期的に処理するため、エンティティ転送の上限が原因で転送が失敗した場合にエラーを返す特定の操作はありません。転送の失敗を検出するには、 `rejected_incoming_transfer_counter` および `rejected_outgoing_transfer_counter` (`AppRuntimeMetrics` struct 内) の現在の値を以前の値と比較できます。拒否されたエンティティはパーティションには含まれませんが、アプリケーションではそのエンティティをシミュレートできます。

メモリの不足

SimSpace Weaver はガベージコレクタープロセスを使用して、解放されたメモリをクリーンアップして解放します。ガベージコレクターがメモリを解放するよりも速くデータを書き込むことができます。この場合、書き込み操作がアプリケーションの予約メモリの制限を超える可能性があります。SimSpace Weaver は `OutOfMemory` (および追加情報) を含むメッセージを含む内部エラーを返します。詳細については、「書き込みを時系列に分散させる」を参照してください。

ベストプラクティス

以下のベストプラクティスは、制限を超えないようにアプリケーションを設計するための一般的なガイドラインです。特定のアプリケーションデザインには当てはまらない場合があります。

頻繁に監視し、速度を低下させる

メトリクスを頻繁に監視し、制限に近づいている処理は遅くする必要があります。

サブスクリプション制限や転送制限の超過を避ける

可能であれば、リモートサブスクリプションとエンティティ転送の数を減らすようにシミュレーションを設計します。配置グループを使用して同じワーカーに複数のパーティションを配置し、ワーカー間のリモートエンティティ転送の必要性を減らすことができます。

書き込みを時系列に分散させる

ティック内の更新の数とサイズは、トランザクションのコミットに必要な時間とメモリに大きな影響を与える可能性があります。メモリ要件が大きいと、アプリケーションランタイムのメモリが不足する可能性があります。書き込みを時系列に分散させて、1 ティックあたりの更新の平均合計サイズを減らすことができます。これにより、パフォーマンスを向上させ、制限を超えないようにすることができます。各ティックに平均 12 MB、または各エンティティに平均 1.5 KB を超えて書き込まないことをお勧めします。

シミュレーションのデバッグ

以下の方法を使用して、シミュレーションに関する情報を取得できます。

トピック

- SimSpace Weaver Local を使用してコンソール出力を確認する

- Amazon CloudWatch Logs のログを確認する
- describe API コールを使用する
- クライアントを接続する

SimSpace Weaver Local を使用してコンソール出力を確認する

まずシミュレーションをローカルで開発してから、 AWS クラウドで実行することをお勧めします。SimSpace Weaver Local を実行するとコンソールの出力を直接表示できます。詳細については、「でのローカル開発 SimSpace Weaver」を参照してください。

Amazon CloudWatch Logs のログを確認する

コンソールでシミュレーションを実行すると AWS クラウド 、アプリケーションの出力が Amazon CloudWatch Logs のログストリームに送信されます。シミュレーションでは他のログデータも書き込まれます。シミュレーションでログデータを書き込むには、シミュレーションスキーマのログ記録を有効にする必要があります。詳細については、「SimSpace Weaver Amazon CloudWatch Logs のログ」を参照してください。

> **Warning**
>
> シミュレーションでは大量のログデータが生成される可能性があります。ログデータは非常に急速に増大する可能性があります。ログを注意深く観察し、実行する必要がなくなったらシミュレーションを停止する必要があります。ログ記録には多額のコストがかかる可能性があります。

describe API コールを使用する

以下のサービス API を使用して、 AWS クラウド内のシミュレーションに関する情報を取得できます。

- ListSimulations – ですべてのシミュレーションのリストを取得します AWS クラウド。

  Example 例

  ```
  aws simspaceweaver list-simulations
  ```

- DescribeSimulation — シミュレーションの詳細を取得します。

Example 例

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

- DescribeApp — アプリケーションの詳細を取得します。

  Example 例

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

SimSpace Weaver APIs「」を参照してくださいSimSpace Weaver API リファレンス。

クライアントを接続する

シミュレーションスキーマの `endpoint_config` で定義した実行中のカスタムアプリケーションまたはサービスアプリケーションにクライアントを接続できます。 SimSpace Weaver アプリケーション SDK には、サンプルアプリケーションを表示するために使用できるサンプルクライアントが含まれています。これらのサンプルクライアントのソースコードとサンプルアプリケーションケーションを見て、独自のクライアントを作成する方法を確認できます。サンプルクライアントを構築して実行する方法の詳細については、「」のチュートリアルを参照してくださいの開始方法 SimSpace Weaver。

サンプルクライアントのソースコードは、以下のフォルダで確認できます。

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

ローカルシミュレーションのデバッグ

Microsoft Visual Studio で SimSpace Weaver Local アプリケーションをデバッグできます。Visual Studio でデバッグする方法の詳細については、「Microsoft Visual Studio documentation」を参照してください。

ローカルシミュレーションをデバッグする方法

1. `schema.yaml` が作業ディレクトリにあることを確認します。

2. Visual Studio で、デバッグする各アプリケーションのコンテキストメニュー (`PathfindingSampleLocalSpatial` または `PathfindingSampleLocalView` など) を開き、デバッグセクションで作業ディレクトリを設定します。
3. デバッグするアプリケーションのコンテキストメニューを開き、[スタートアッププロジェクトとして設定] を選択します。
4. F5 を選択して、アプリケーションのデバッグを開始します。

シミュレーションをデバッグするための要件は、シミュレーションを正常に実行するための要件と同じです。スキーマで指定された数の空間アプリケーションを起動する必要があります。例えば、スキーマで 2x2 グリッドが指定されている場合に、空間アプリケーションをデバッグモードで起動した場合、さらに 3 つの空間アプリケーションを (デバッグモードまたはデバッグモードでない状態で) 起動するまでシミュレーションは実行されません。

カスタムアプリケーションをデバッグするには、まず空間アプリケーションを起動し、次にデバッガーでカスタムアプリケーションを起動する必要があります。

シミュレーションはロックステップで実行されることに注意してください。アプリケーションがブレークポイントに達するとすぐに、他のすべてのアプリケーションは一時停止します。そのブレークポイントから続行すると、他のアプリケーションは続行されます。

カスタムコンテナ

AWS SimSpace Weaver アプリケーションはコンテナ化された Amazon Linux 2 (AL2) 環境で実行されます。では AWS クラウド、Amazon Elastic Container Registry (Amazon ECR) から提供される`amazonlinux:2`イメージから構築された Docker コンテナでシミュレーション SimSpace Weaver を実行します。カスタム Docker イメージを作成して Amazon ECR に保存し、提供されているデフォルトの Docker イメージの代わりにそのイメージをシミュレーションに使用できます。

カスタムコンテナを使用してソフトウェアの依存関係を管理し、標準の Docker イメージにはないソフトウェアコンポーネントを追加できます。例えば、アプリケーションが使用する一般公開ソフトウェアライブラリをコンテナに追加し、カスタムコードをアプリケーションの zip ファイルのみに入れることができます。

> **Important**
>
> Amazon ECR Public Gallery またはお客様のプライベート Amazon ECR レジストリにある Amazon ECR リポジトリでホストされている AL2 Docker イメージのみをサポートします。Amazon ECR の外部でホストされている Docker イメージはサポートされていませ

ん。Amazon ECR の詳細については、「Amazon Elastic Container Registry のドキュメント」を参照してください。

トピック

- カスタムコンテナを作成する
- カスタムコンテナを使用するようにプロジェクトを変更する
- カスタムコンテナに関するよくある質問
- カスタムコンテナのトラブルシューティング

カスタムコンテナを作成する

これらの手順は、Docker と Amazon Elastic Container Registry (Amazon ECR) の使用方法についての知識があることを前提としています。Amazon ECR の詳細については、「Amazon ECR ユーザーガイド」を参照してください。

前提条件

- これらのアクションを実行するために使用する IAM アイデンティティ (ユーザーまたはロール) には、Amazon ECR を使用するための適切なアクセス許可があります。
- Docker は、ローカルシステムにインストールされます

カスタムコンテナを作成する方法

1. `Dockerfile` を作成します。

   AWS SimSpace Weaver アプリケーションを実行する `Dockerfile` は、Amazon ECR の Amazon Linux 2イメージで始まります。

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

2. `Dockerfile` を構築します。
3. コンテナイメージを Amazon ECR にアップロードします。
   - AWS Management Consoleを使用します。
   - AWS Command Line Interfaceを使用します。

> **Note**
>
> コンテナイメージを Amazon ECR にアップロードしようとして `AccessDeniedException` エラーが発生した場合、お使いの IAM アイデンティティ (ユーザーまたはロール) には Amazon ECR を使用するために必要なアクセス許可がない可能性があります。`AmazonEC2ContainerRegistryPowerUser` AWS 管理ポリシーを IAM ID にアタッチして、もう一度試すことができます。ポリシーをアタッチする方法の詳細については、「AWS Identity and Access Management ユーザーガイド」の「IAM アイデンティティの許可の追加および削除」を参照してください。

カスタムコンテナを使用するようにプロジェクトを変更する

この手順では、 の使用方法が既にわかって AWS SimSpace Weaver おり、 でアプリケーションストレージと開発ワークフローをより AWS クラウド 効率的にしたいことを前提としています。

前提条件

- Amazon Elastic Container Registry (Amazon ECR) にはカスタムコンテナが用意されています。カスタムコンテナの作成方法の詳細については、「カスタムコンテナを作成する」を参照してください。

カスタムコンテナを使用するようにプロジェクトを変更する方法

1. Amazon ECR を使用するアクセス許可をプロジェクトのシミュレーションアプリケーションロールに追加します。

   a. 以下のアクセス許可を持つ IAM ポリシーがない場合は、ポリシーを作成します。ポリシー名 `simspaceweaver-ecr` を付けることをお勧めします。IAM ポリシー作成方法の詳細については、「AWS Identity and Access Management ユーザーガイド」の「IAM ポリシーの作成」を参照してください。

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

```
            "Action": [
                "ecr:BatchGetImage",
                "ecr:GetDownloadUrlForLayer",
                "ecr:GetAuthorizationToken"
            ],
            "Resource": "*"
        }
    ]
}
```

b. プロジェクトのシミュレーションアプリケーションロールの名前を検索します。

   i. テキストエディタで、 AWS CloudFormation テンプレートを開きます。

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

   ii. `WeaverAppRole` の下の [`RoleName`] プロパティを検索します。値はプロジェクトのシミュレーションアプリケーションロール名です。

   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` ポリシーをプロジェクトのシミュレーションアプリケーションロールにアタッチします。ポリシーをアタッチする方法の詳細については、「AWS Identity and Access Management ユーザーガイド」の「IAM アイデンティティの許可の追加および削除」を参照してください。

d. に移動*sdk-folder*して次のコマンドを実行して、サンプル SimSpace Weaver スタックを更新します。

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

2. プロジェクトのシミュレーションスキーマでコンテナイメージを指定します。

   - simulation_properties の下にオプションの default_image プロパティを追加して、すべてのドメインのデフォルトのカスタムコンテナイメージを指定できます。
   - カスタムコンテナイメージを使用するドメインの image プロパティを app_config に追加します。値として Amazon ECR リポジトリ URI を指定します。ドメインごとに異なるイメージを指定できます。
     - image がドメインに指定されていない状態で default_image が指定されている場合、そのドメイン内のアプリケーションはデフォルトイメージを使用します。
     - image がドメインに指定されておらず、 が指定されていない場合、そのドメイン内のアプリケーションdefault_imageは標準 SimSpace Weaver コンテナで実行されます。

Example カスタムコンテナ設定を含むスキーマスニペット

```
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. 通常どおりプロジェクトを構築してアップロードします。

カスタムコンテナに関するよくある質問

Q1. コンテナの中身を変更したい場合はどうすればいいですか?

- 実行中のシミュレーションの場合 — 実行中のシミュレーションのコンテナは変更できません。新しいコンテナを構築し、そのコンテナを使用する新しいシミュレーションを開始する必要があります。
- 新しいシミュレーションの場合 — 新しいコンテナを構築して Amazon Elastic Container Registry (Amazon ECR) にアップロードし、そのコンテナを使用する新しいシミュレーションを開始します。

Q2. シミュレーション用のコンテナイメージを変更する方法を教えてください。

- 実行中のシミュレーションの場合 — 実行中のシミュレーションのコンテナは変更できません。新しいコンテナを使用する新しいシミュレーションを開始する必要があります。
- 新しいシミュレーションの場合 — プロジェクトのシミュレーションスキーマに新しいコンテナイメージを指定します。詳細については、「カスタムコンテナを使用するようにプロジェクトを変更する」を参照してください。

カスタムコンテナのトラブルシューティング

トピック

- Amazon Elastic Container Registry (Amazon ECR) にイメージをアップロードすると AccessDeniedException が表示される

- カスタムコンテナを使用するシミュレーションが開始できない

Amazon Elastic Container Registry (Amazon ECR) にイメージをアップロードすると AccessDeniedException が表示される

コンテナイメージを Amazon ECR にアップロードしようとして `AccessDeniedException` エラーが発生した場合、お使いの IAM アイデンティティ (ユーザーまたはロール) には Amazon ECR を使用するために必要なアクセス許可がない可能性があります。`AmazonEC2ContainerRegistryPowerUser` AWS 管理ポリシーを IAM ID にアタッチして、もう一度試すことができます。ポリシーをアタッチする方法の詳細については、「AWS Identity and Access Management ユーザーガイド」の「IAM アイデンティティの許可の追加および削除」を参照してください。

カスタムコンテナを使用するシミュレーションが開始できない

トラブルシューティングのヒント

- シミュレーションでログ記録が有効になっている場合は、エラーログを確認します。
- カスタムコンテナなしでシミュレーションをテストします。
- シミュレーションをローカルでテストします。詳細については、「でのローカル開発 SimSpace Weaver」を参照してください。

Python の使用

SimSpace Weaver アプリケーションとクライアントには Python を使用できます。Python ソフトウェア開発キット (Python SDK) は、標準の SimSpace Weaver アプリケーション SDK 配布パッケージの一部として含まれています。Python による開発は、サポートされている他の言語での開発と同様に機能します。

> **Important**
>
> SimSpace Weaver は Python バージョン 3.9 のみをサポートしています。

> **Important**
>
> SimSpace Weaver Python のサポートには SimSpace Weaver バージョン 1.15.0 以降が必要です。

トピック

- Python プロジェクトの作成
- Python シミュレーションを開始する
- Python のサンプルクライアント
- Python の使用に関するよくある質問
- Python に関連する問題のトラブルシューティング

Python プロジェクトの作成

Python カスタムコンテナ

で Python ベースの SimSpace Weaver シミュレーションを実行するには AWS クラウド、必要な依存関係を含むカスタムコンテナを作成できます。詳細については、「カスタムコンテナ」を参照してください。

Python カスタムコンテナには以下が含まれている必要があります。

- gcc
- openssl-devel
- bzip2-devel
- libffi-devel
- wget
- tar
- gzip
- make
- Python (バージョン 3.9)

`PythonBubblesSample` テンプレートを使用してプロジェクトを作成する場合は、(プロジェクトの `tools` フォルダにある) `quick-start.py` スクリプトを実行して、必要な依存関係を含む

Docker イメージを作成できます。このスクリプトは、Amazon Elastic Container Registry (Amazon ECR) にイメージをアップロードします。

`quick-start.py` スクリプトは以下の `Dockerfile` を使用します。

```
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
```

独自の依存関係を、`Dockerfile` に追加できます。

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

`requirements.txt` ファイルには、`PythonBubblesSample` サンプルシミュレーションに必要な Python パッケージのリストが含まれています。

```
Flask==2.1.1
```

独自の Python パッケージの依存関係を `requirements.txt` に追加できます。

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

`Dockerfile` および `requirements.txt` はプロジェクトの `tools` フォルダにあります。

> **Important**
>
> 技術的には Python シミュレーションでカスタムコンテナを使用する必要はありませんが、カスタムコンテナを使用することを強くお勧めします。Amazon Linux 2 (AL2) の標準コンテナは、Python を提供していません。したがって、Python を含むカスタムコンテナを使用し

ない場合は、アップロードする各アプリの zip ファイルに Python と必要な依存関係を含める必要があります SimSpace Weaver。

Python シミュレーションを開始する

Python ベースのシミュレーションは、通常の SimSpace Weaver シミュレーションと同じ方法で、の SimSpace Weaver Localと SimSpace Weaver の両方で開始できます AWS クラウド。詳細については、「」のチュートリアルを参照してください[の開始方法 SimSpace Weaver](#)。

`PythonBubblesSample` には独自の Python サンプルクライアントが含まれています。詳細については、「[Python のサンプルクライアント](#)」を参照してください。

Python のサンプルクライアント

`PythonBubblesSample` テンプレートを使用してプロジェクトを作成する場合、プロジェクトには Python サンプルクライアントが含まれます。サンプルクライアントを使用して `PythonBubblesSample` シミュレーションを表示できます。サンプルクライアントを起点として使用して、独自の Python クライアントを作成することもできます。

以下の手順は、`PythonBubblesSample` プロジェクトを作成し、シミュレーションを開始済みであることを前提としています。

Python クライアントを起動する方法

1. コマンドプロンプトウィンドウで、`PyBubbleClient`サンプルプロジェクトフォルダに移動します。

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

2. Python クライアントを実行します。

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

   パラメータ

   host

   シミュレーションの IP アドレス。で開始されたシミュレーションについては AWS クラウド、 [SimSpace Weaver コンソール](#)でシミュレーションの IP アドレスを確認するか、ク

イックスタートチュートリアルカスタムアプリケーションの IP アドレスとポート番号を取得するの の手順を使用します。ローカルシミュレーションの場合は、IP アドレスとして `127.0.0.1` を使用します。

port

シミュレーションのポート番号。で開始されたシミュレーションの場合 AWS クラウド、これは`Actual`ポート番号です。シミュレーションのポート番号は [SimSpace Weaver] コンソールで確認するか、「クイックスタートチュートリアル」の「カスタムアプリケーションの IP アドレスとポート番号を取得する」手順を使用して確認できます。ローカルシミュレーションの場合は、ポート番号として `7000` を使用します。

simsize

クライアントに表示するエンティティの最大数。

Python の使用に関するよくある質問

Q1. どのバージョンの Python がサポートされていますか?

SimSpace Weaver は Python バージョン 3.9 のみをサポートしています。

Python に関連する問題のトラブルシューティング

トピック

- カスタムコンテナ作成中の失敗
- Python シミュレーションが開始されない
- Python シミュレーションまたはビュークライアントに ModuleNotFound エラーが表示される

カスタムコンテナ作成中の失敗

`quick-start.py` の実行後にエラー `no basic auth credentials` が発生した場合は、Amazon ECR の一時的な認証情報に問題がある可能性があります。 AWS リージョン ID と AWS アカウント番号を使用して、次のコマンドを実行します。

```
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 リージョン 指定する がシミュレーションに使用するものと同じであることを確認します。が SimSpace Weaver サポート AWS リージョン する のいずれかを使用します。詳細については、「SimSpace Weaver エンドポイントとクォータ」を参照してください。

`aws ecr` コマンドを実行したら、もう一度 `quick-start.py` を実行してください。

確認すべきその他のトラブルシューティングリソース

- カスタムコンテナのトラブルシューティング
- 「Amazon ECR ユーザーガイド」の「Amazon ECR トラブルシューティング」
- 詳細については、「Amazon ECR ユーザーガイド」の「Amazon ECR での設定」を参照してください

Python シミュレーションが開始されない

シミュレーションの管理ログに `Unable to start app` エラーが表示される場合があります。これは、カスタムコンテナの作成に失敗した場合に発生する可能性があります。詳細については、「カスタムコンテナ作成中の失敗」を参照してください。ログの詳細については、SimSpace Weaver Amazon CloudWatch Logs の ログ を参照してください。

コンテナに問題がないと確信できる場合は、アプリケーションの Python ソースコードを確認します。SimSpace Weaver Local を使用してアプリケーションをテストできます。詳細については、「でのローカル開発 SimSpace Weaver」を参照してください。

Python シミュレーションまたはビュークライアントに ModuleNotFound エラーが表示される

必要な Python パッケージが見つからない場合、Python は `ModuleNotFound` エラーを返します。

シミュレーションが にある場合は AWS クラウド、カスタムコンテナに必要な依存関係がすべて にリストされていることを確認しますrequirements.txt。requirements.txt を編集する場合は、必ずもう一度 quick-start.py を実行してください。

PythonBubblesSample クライアント側でエラーが発生した場合は、pip を使用して指定のパッケージをインストールします。

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

他のエンジンのサポート

で独自のカスタムC++エンジンを使用できます SimSpace Weaver。現在、以下のエンジンのサポートを開発中です。これらのエンジンにはそれぞれ個別のドキュメントが用意されています。

> **Important**
> ここに記載されているエンジンとの統合は実験段階です。これらはプレビューできます。

エンジン

- Unity (最小バージョン 2022.3.19.F1)
- Unreal Engine (最小バージョン 5.0)

Unity

Unity で SimSpace Weaver シミュレーションを構築する前に、Unity開発環境が既にインストールされている必要があります。詳細については、以下の個別の指示を参照してください。

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

Unreal Engine

ソースコードから Unreal Engine 専有サーバーを構築する必要があります。SimSpaceWeaverAppSDKDistributable には、Unreal Engine の PathfindingSample のバージョンが含まれています。詳細については、以下の個別の指示を参照してください。

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

でのライセンスソフトウェアの使用 AWS SimSpace Weaver

AWS SimSpace Weaver では、選択したシミュレーションエンジンとコンテンツを使用してシミュレーションを構築できます。の使用に関連して SimSpace Weaver、シミュレーションで使用するソフトウェアまたはコンテンツのライセンス条項を取得、維持、遵守する責任はお客様にあります。仮想化ホスト環境でのソフトウェアとコンテンツの展開がライセンス契約書上で許可されていることを確認してください。

を使用した リソースの管理 AWS CloudFormation

AWS CloudFormation を使用して AWS SimSpace Weaver リソースを管理できます。 AWS CloudFormation は、 AWS インフラストラクチャをコードとして指定、プロビジョニング、管理するのに役立つ別の AWS サービスです。 AWS CloudFormation では、テンプレートと呼ばれる JSON または YAML ファイルを作成します。テンプレートはインフラストラクチャの詳細を指定します。 AWS CloudFormation　はテンプレートを使用して、スタックと呼ばれる単一のユニットとしてインフラストラクチャをプロビジョニングします。スタックを削除すると、スタック内のすべてのものを同時に AWS CloudFormation 削除できます。標準のソースコード管理プロセスを使用してテンプレートを管理できます (例えば、Git などのバージョン管理システムでテンプレートを追跡するなどです)。詳細については AWS CloudFormation、AWS CloudFormation 「 ユーザーガイド」を参照してください。

シミュレーションリソース

では AWS、リソースはユーザーが操作できるエンティティです。例として、Amazon EC2 インスタンス、Amazon S3 バケット、IAM ロールなどがあります。 SimSpace Weaver シミュレーションはリソースです。設定では、通常、 形式で AWS リソースを指定しますAWS::*service*::resource。では SimSpace Weaver、シミュレーションリソースを として指定しますAWS::SimSpaceWeaver::Simulation。のシミュレーションリソースの詳細については AWS CloudFormation、AWS CloudFormation 「 ユーザーガイド」のSimSpace Weaver「」セクションを参照してください。

AWS CloudFormation で を使用するにはどうすればよいですか SimSpace Weaver？

プロビジョニングする AWS リソースを指定する AWS CloudFormation テンプレートを作成できます。テンプレートでは、アーキテクチャ全体、アーキテクチャの一部、または小規模なソリューションを指定できます。例えば、Amazon S3 バケット、IAM アクセス許可、Amazon Relational

Database Service または Amazon DynamoDB のサポートデータベース、`Simulation`リソースを含む SimSpace Weaver ソリューションのアーキテクチャを指定できます。その後、 AWS CloudFormation を使用して、これらのすべてのリソースをユニットとして同時にプロビジョニングできます。

Example IAM リソースを作成してシミュレーションを開始するテンプレート

以下のテンプレート例は、 SimSpace Weaver がアカウントでアクションを実行するために必要な IAM ロールとアクセス許可を作成します。 SimSpace Weaver アプリケーション SDK スクリプトは、プロジェクトの作成 AWS リージョン 時に特定の にロールとアクセス許可を作成しますが、 AWS CloudFormation テンプレートを使用して、スクリプトを再度実行せずにシミュレーションを別の AWS リージョン にデプロイできます。例えば、ディザスタリカバリを目的としたバックアップシミュレーションを設定できます。

この例では、元のシミュレーション名は `MySimulation` です。スキーマのバケットは、 AWS リージョン AWS CloudFormation がスタックを構築する に既に存在します。バケットには、その AWS リージョンでシミュレーションを実行するように適切に設定されたバージョンのスキーマが含まれています。スキーマはアプリケーションのの場所を指定していることに留意します。この zip ファイルは、シミュレーションと同じ AWS リージョン にある Amazon S3 バケットです。アプリケーション zip バケットとファイルは、 がスタック AWS CloudFormation を構築する AWS リージョン ときに に既に存在している必要があります。存在しない場合、シミュレーションは開始されません。この例のバケット名には が含まれていますが AWS リージョン、バケットが実際にどこにあるかは決定されません。バケットが実際にその にあることを確認する必要があります AWS リージョン （バケットのプロパティは、Amazon S3 コンソール、Amazon S3 APIs、または の Amazon S3 コマンドで確認できます AWS CLI ）。

この例では、 の組み込み関数とパラメータ AWS CloudFormation を使用して変数置換を実行します。詳細については、「AWS CloudFormation ユーザーガイド」の「組み込み関数リファレンス」と「疑似パラメータリファレンス」を参照してください。

```
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'
```

でのスナップショットの使用 AWS CloudFormation

スナップショットはシミュレーションのバックアップです。以下の例では、スキーマからではなくスナップショットから新しいシミュレーションを開始します。この例のスナップショットは、

SimSpace Weaver アプリケーション SDK プロジェクト simulation から作成されました。 は新しいシミュレーションリソース AWS CloudFormation を作成し、スナップショットのデータを使用して初期化します。新しいシミュレーションは、元のシミュレーションとは異なる `MaximumDuration` を持つ可能性があります。

元のシミュレーションのアプリケーションロールのコピーを作成して使用することをお勧めします。元のシミュレーションのアプリケーションロールは、そのシミュレーションの AWS CloudFormation スタックを削除すると削除される可能性があります。

```
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"
```

スナップショット

スナップショットを作成して、シミュレーションエンティティデータをいつでもバックアップできます。 SimSpace Weaver は Amazon S3 バケットに .zip ファイルを作成します。snapshot を使用して新しいシミュレーションを作成できます。 は、スナップショットに保存されているエンティティデータを使用して新しいシミュレーションのステートファブリックを SimSpace Weaver 初期化し、スナップショットの作成時に実行されていた空間アプリケーションとサービスアプリケーションを起動し、クロックを適切な tick に設定します。 は、スキーマファイルからではなく、スナップショットからシミュレーションの設定 SimSpace Weaver を取得します。アプリケーションの .zip ファイルは、Amazon S3 内の元のシミュレーションと同じ場所にある必要があります。カスタムアプリケーションはすべて個別に起動する必要があります。

トピック

- スナップショットのユースケース
- SimSpace Weaver コンソールを使用してスナップショットを操作する

- AWS CLI を使用してスナップショットを操作する
- でのスナップショットの使用 AWS CloudFormation
- スナップショットに関するよくある質問

スナップショットのユースケース

前の状態に戻り、分岐シナリオを検討する

シミュレーションのスナップショットを作成して、特定の状態に保存できます。その後、そのスナップショットから複数の新しいシミュレーションを作成し、その状態から分岐する可能性のあるさまざまなシナリオを検討できます。

ディザスタリカバリとセキュリティのベストプラクティス

特に 1 時間以上実行されるシミュレーションや複数のワーカーを使用するシミュレーションでは、定期的にシミュレーションをバックアップすることをお勧めします。バックアップは、障害やセキュリティインシデントからの回復に役立ちます。スナップショットは、シミュレーションのバックアップを提供する方法を提供します。スナップショットでは、アプリケーションの .zip ファイルが以前と同じ Amazon S3 の場所に存在している必要があります。アプリケーションの.zip ファイルを別の場所に移動できるようにする必要がある場合は、カスタムバックアップソリューションを使用する必要があります。

その他のベストプラクティスの詳細については、「を使用する際のベストプラクティス SimSpace Weaver」および「のセキュリティのベストプラクティス SimSpace Weaver」を参照してください。

シミュレーション時間を延長する

シミュレーションリソースは、 SimSpace Weaverでのシミュレーションの表現です。すべてのシミュレーションリソースには `MaximumDuration` 設定があります。シミュレーションは、`MaximumDuration` に達すると自動的に停止します。`MaximumDuration` の最大値は `14D` (14 日間) です。

シミュレーションをそのシミュレーションリソースの `MaximumDuration` より長く持続させる必要がある場合は、シミュレーションリソースがその `MaximumDuration` に達する前にスナップショットを作成できます。スナップショットを使用して新しいシミュレーションを開始 (新しいシミュレーションリソースを作成) できます。 SimSpace Weaver はスナップショットからエンティティデータを初期化し、以前に実行したのと同じ空間アプリケーションとサービスアプリケーションを起動して、クロックを復元します。カスタムアプリケーションを起動して、その他のカスタム初期化を実行

できます。新しいシミュレーションリソースの `MaximumDuration` は、起動時に別の値に設定できます。

SimSpace Weaver コンソールを使用してスナップショットを操作する

SimSpace Weaver コンソールを使用してシミュレーションのスナップショットを作成できます。

トピック

- コンソールでスナップショットを作成する
- コンソールを使用してスナップショットからシミュレーションを開始する

コンソールでスナップショットを作成する

スナップショットを作成する方法

1. にサインイン AWS Management Console し、 SimSpace Weaver コンソールに接続します。
2. ナビゲーションペインで、[シミュレーション] を選択します。
3. シミュレーション名の横にあるラジオボタンを選択します。シミュレーションの [ステータス] は [開始] である必要があります。
4. ページの上部で、[スナップショットの作成] を選択します。
5. スナップショット設定のスナップショット送信先で、スナップショット SimSpace Weaver を作成するバケットまたはバケットとフォルダの Amazon S3 URI を入力します。使用可能なバケットをブラウズして場所を選択する場合は、[S3 を参照] を選択できます。

   > **Important**
   >
   > Amazon S3 バケットはシミュレーションと同じ AWS リージョン にある必要があります。

   > **Note**
   >
   > SimSpace Weaver は、選択したスナップショット送信先内に`snapshot`フォルダを作成します。 はその`snapshot`フォルダにスナップショット .zip ファイル SimSpace Weaver を作成します。

6. [スナップショットを作成] を選択します。

コンソールを使用してスナップショットからシミュレーションを開始する

スナップショットからシミュレーションを開始するには、スナップショットの .zip ファイルが、シミュレーションがアクセスできる Amazon S3 バケットに存在している必要があります。シミュレーションでは、シミュレーションの開始時に選択したアプリケーションロールで定義されているアクセス許可を使用します。元のシミュレーションのアプリケーションの .zip ファイルはすべて、スナップショットが作成されたときと同じ場所に存在している必要があります。

スナップショットからシミュレーションを開始する方法

1. にサインイン AWS Management Console し、 SimSpace Weaver コンソールに接続します。
2. ナビゲーションペインで、[シミュレーション] を選択します。
3. ページ上部の [シミュレーションを開始] を選択します。
4. [シミュレーション設定] で、シミュレーションの名前と説明 (オプション) を入力します。シミュレーション名は AWS アカウント内で一意である必要があります。
5. [シミュレーション開始方法] で [Amazon S3 のスナップショットを使用する] を選択します。
6. スナップショットの Amazon S3 URI には、スナップショットファイルの Amazon S3 URI を入力するか、[S3 を参照] を選択してファイルを参照して選択します。

   > **Important**
   >
   > Amazon S3 バケットはシミュレーションと同じ AWS リージョン にある必要があります。

7. [IAM ロール] には、シミュレーションで使用するアプリケーションロールを選択します。
8. [最大期間] には、シミュレーションリソースを実行する最大期間を入力します。最大値は 14D です。最大期間の詳細については、「https://docs.aws.amazon.com/simspaceweaver/latest/APIReference/API_StartSimulation.html」を参照してください
9. タグを追加する場合は、[タグ - オプション] で [新しいタグを追加] を選択します。
10. [シミュレーションを開始] を選択します。

AWS CLI を使用してスナップショットを操作する

を使用して AWS CLI 、コマンドプロンプトから SimSpace Weaver APIsを呼び出すことができます。が正しく AWS CLI インストールおよび設定されている必要があります。詳細について

は、「 バージョン 2 AWS ユーザーガイド」の「 CLI の最新バージョンのインストールまたは更新」を参照してください。 AWS Command Line Interface

トピック

- を使用してスナップショット AWS CLI を作成する
- を使用してスナップショットからシミュレーション AWS CLI を開始する

を使用してスナップショット AWS CLI を作成する

スナップショットを作成する方法

- コマンドプロンプトで `CreateSnapshot` API を呼び出します。

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

パラメータ

シミュレーション

開始したシミュレーション名。`aws simspaceweaver list-simulations` を使用して、シミュレーション名およびステータスを確認できます。

宛先

スナップショットファイルの送信先の Amazon S3 バケットとオプションのオブジェクトキーのプレフィックス指定する文字列。オブジェクトキープレフィックスは通常、バケット内のフォルダです。 は、この宛先の`snapshot`フォルダ内にスナップショット SimSpace Weaver を作成します。

> **Important**
> Amazon S3 バケットはシミュレーションと同じ AWS リージョン にある必要があります。

例

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

`CreateSnapshot` API の詳細については、「AWS SimSpace Weaver API リファレンス」の「CreateSnapshot」を参照してください。

を使用してスナップショットからシミュレーション AWS CLI を開始する

スナップショットからシミュレーションを開始する方法

- コマンドプロンプトで `StartSimulation` API を呼び出します。

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

パラメータ

名前

新しいシミュレーションの名前。シミュレーション名は 内で一意である必要があります AWS アカウント。`aws simspaceweaver list-simulations` を使用して、既存のシミュレーション名を確認できます。

role-arn

シミュレーションが使用するアプリケーションロールの Amazon リソースネーム (ARN)。

snapshot-s3-location

スナップショットファイルの Amazon S3 バケットとオブジェクトキーを指定する文字列。

> **Important**
> Amazon S3 バケットはシミュレーションと同じ AWS リージョン にある必要があります。

例

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

`StartSimulation` API の詳細については、「AWS SimSpace Weaver API リファレンス」の「StartSimulation」を参照してください。

スナップショットに関するよくある質問

スナップショットの作成中もシミュレーションは実行され続けますか?

シミュレーションリソースはスナップショット中も引き続き実行され、その間も引き続き料金が発生します。この時間はシミュレーションの最大期間に加算されます。スナップショットの進行中は、アプリケーションにティックを受け取りません。スナップショットの作成開始時にクロックのステータスが `STARTED` であった場合、クロックは `STARTED` ステータスのままになります。スナップショットの終了後、アプリケーションには再びティックを受け取ります。クロックのステータスが `STOPPED` であった場合、クロックの状態は `STOPPED` のままになります。`STARTED` ステータスのシミュレーションは、クロックのステータスが `STOPPED` であっても実行されることに注意してください。

スナップショットが進行中に、シミュレーションが最大期間に達した場合はどうなりますか?

シミュレーションはスナップショットを終了し、スナップショット処理が終了すると (成功か失敗かにかかわらず) すぐに停止します。所要時間、予測できるスナップショットファイルのサイズ、および正常に完了するかどうかを、スナップショット処理の前にテストして、確認することをお勧めします。

スナップショットが進行中のシミュレーションを停止するとどうなりますか?

進行中のスナップショットは、シミュレーションを停止するとすぐに停止します。スナップショットファイルは作成されません。

実行中のスナップショットを停止する方法を教えてください。

進行中のスナップショットを停止する唯一の方法は、シミュレーションを停止することです。停止したシミュレーションは再開できません。

スナップショットの完了にはどれくらいの時間がかかりますか?

スナップショットの作成に必要な時間は、シミュレーションによって異なります。シミュレーションにどれくらいの時間がかかるかをスナップショット処理の前にテストして、確認することをお勧めします。

スナップショットファイルのサイズはどれくらいになりますか?

スナップショットファイルのサイズは、シミュレーションによって異なります。シミュレーション用のファイルのサイズをスナップショット処理の前にテストして、確認することをお勧めします。

メッセージング

メッセージング API は、シミュレーション内のアプリケーション間の通信を簡素化します。メッセージを送受信するための APIsは、 SimSpace Weaver アプリケーション SDK の一部です。メッセージングは現在、メッセージの送受信にベストエフォートアプローチを使用しています。SimSpace Weaver は、次のシミュレーションティックでメッセージの送受信を試みますが、配信、注文、到着時間の保証はありません。

トピック

- メッセージングのユースケース
- メッセージング APIs
- メッセージングを使用するタイミング
- メッセージングを使用する際のヒント
- メッセージングエラーとトラブルシューティング

メッセージングのユースケース

シミュレーションアプリケーション間の通信

メッセージング API を使用して、シミュレーション内のアプリケーション間で通信します。これを使用して、遠くのエンティティの状態を変更したり、エンティティの動作を変更したり、シミュレーション全体に情報をブロードキャストしたりできます。

メッセージ受信の確認

送信メッセージには、メッセージヘッダーの送信者に関する情報が含まれます。この情報を使用して、メッセージの受信時に確認応答を送り返します。

カスタムアプリケーションによって受信したデータをシミュレーション内の他のアプリケーションに転送する

メッセージングは、クライアントが で実行されているカスタムアプリケーションに接続する方法に代わるものではありません SimSpace Weaver。ただし、メッセージングを使用すると、クライアントデータを受信するカスタムアプリケーションから、外部接続を持たない他のアプリケーションにデータを転送できます。メッセージフローは逆に機能するため、外部接続のないアプリはカスタムアプリにデータを転送し、次にクライアントに転送できます。

メッセージング APIs

メッセージング APIs は、 SimSpace Weaver アプリケーション SDK (最小バージョン 1.16.0) に含まれています。メッセージングは、C++、Python、および Unreal Engine 5 および Unity との統合でサポートされています。

メッセージトランザクションを処理する関数は `SendMessage`と の 2 つです`ReceiveMessages`。送信されたすべてのメッセージには、送信先とペイロードが含まれます。`ReceiveMessages` API は、アプリのインバウンドメッセージキューに現在あるメッセージのリストを返します。

C++

メッセージの送信

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

メッセージを受信する

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

Python

メッセージの送信

```
api.send_message(
```

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

メッセージを受信する

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

トピック

- メッセージの送信
- メッセージの受信
- 送信者への返信

メッセージの送信

メッセージは、トランザクション (他の Weaver API コールと同様) 、ペイロード、および送信先で構成されます。

メッセージペイロード

メッセージペイロードは、最大 256 バイトの柔軟なデータ構造です。メッセージペイロードを作成するためのベストプラクティスとして、以下をお勧めします。

メッセージペイロードを作成するには

1. メッセージの内容を定義するデータ構造 (C++ `struct` の など) を作成します。
2. メッセージで送信する値を含むメッセージペイロードを作成します。
3. `MessagePayload` オブジェクトを作成します。

メッセージの送信先

メッセージの送信先は、 `MessageEndpoint` オブジェクトによって定義されます。これには、エンドポイントタイプとエンドポイント ID の両方が含まれます。現在サポートされているエンドポイン

トタイプは のみです。これによりPartition、シミュレーション内の他のパーティションへのメッセージに対処できます。エンドポイント ID は、ターゲット送信先のパーティション ID です。

メッセージには 1 つの送信先アドレスしか指定できません。複数のパーティションに同時にメッセージを送信する場合は、複数のメッセージを作成して送信します。

位置からメッセージエンドポイントを解決する方法については、「」を参照してください[メッセージングを使用する際のヒント](#)。

メッセージを送信する

送信先オブジェクトとペイロードオブジェクトを作成した後、 SendMessage API を使用できます。

C++

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

Python

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

メッセージを送信する完全な例

次の例は、汎用メッセージを作成して送信する方法を示しています。この例では、16 個の個別のメッセージを送信します。各メッセージには、0 と 15 の間の値を持つペイロードと、現在のシミュレーションティックが含まれます。

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
```

メッセージの受信

SimSpace Weaver は、パーティションのインバウンドメッセージキューにメッセージを配信します。`ReceiveMessages` API を使用して、キューからのメッセージを含む`MessageList`オブジェクトを取得します。`ExtractMessage` API を使用して各メッセージを処理し、メッセージデータを取得します。

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
```

送信者への返信

受信したすべてのメッセージには、メッセージの元の送信者に関する情報を含むメッセージヘッダーが含まれています。message.header.source_endpoint を使用して返信を送信できます。

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
```

メッセージングを使用するタイミング

のメッセージング SimSpace Weaver は、シミュレーションアプリケーション間で情報を交換するための別のパターンを提供します。サブスクリプションは、シミュレーションの特定のアプリケーションまたは領域からデータを読み取るプルメカニズムを提供します。メッセージは、シミュレーションの特定のアプリケーションまたは領域にデータを送信するプッシュメカニズムを提供します。

以下は、サブスクリプションを通じてデータを取得または読み取るよりも、メッセージングを使用してデータをプッシュする方が役立つ 2 つのユースケースです。

Example 1: エンティティの位置を変更するコマンドを別のアプリケーションに送信する

```
// 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);
```

受信側では、アプリケーションはエンティティの位置を更新し、ステートファブリックに書き込みます。

```
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: 空間アプリケーションへのエンティティ作成メッセージの送信

```
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);
```

受信側では、アプリケーションはステートファブリックに新しいエンティティを作成し、その位置を更新します。

```
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)));
}
```

メッセージングを使用する際のヒント

位置またはアプリケーション名からエンドポイントを解決する

`AllPartitions` 関数を使用して、メッセージパーティション ID とメッセージ送信先を決定するために必要な空間境界とドメイン IDsを取得できます。ただし、メッセージする位置がわかっていてもパーティション ID がわかっていない場合は、MessageEndpointResolver 関数を使用できます。

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

メッセージペイロードのシリアル化と逆シリアル化

次の関数を使用して、メッセージペイロードを作成および読み取ることができます。詳細については、ローカルシステムのアプリケーション SDK ライブラリの MessagingUtils.h を参照してください。

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

メッセージングエラーとトラブルシューティング

メッセージング APIs を使用すると、次のエラーが発生することがあります。

エンドポイント解決エラー

これらのエラーは、アプリケーションがメッセージを送信する前に発生する可能性があります。

ドメイン名チェック

無効なエンドポイントにメッセージを送信すると、次のエラーが発生します。

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

これは、カスタムアプリにメッセージを送信しようとして、そのカスタムアプリがまだシミュレーションに参加していない場合に発生する可能性があります。DescribeSimulation API を使用して、メッセージを送信する前にカスタムアプリが起動していることを確認します。この動作は、SimSpace Weaver Localと で同じです AWS クラウド。

位置チェック

有効なドメイン名のエンドポイントを解決しようとしたが、無効な位置があると、次のエラーが発生します。

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

SimSpace Weaver アプリケーション SDK に含まれるMessageUtilsライブラリMessageEndpointResolverで を使用することをお勧めします。

メッセージ送信エラー

アプリケーションがメッセージを送信すると、次のエラーが発生する可能性があります。

アプリ、ティック、超過あたりのメッセージ送信制限

シミュレーションティックごとにアプリごとに送信できるメッセージ数の現在の制限は 128 です。同じティックに対するそれ以降の呼び出しは、次のエラーで失敗します。

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

SimSpace Weaver は、次のティックで未送信メッセージを送信しようとします。この問題を解決するには、送信頻度を下げます。256 バイトの制限より小さいメッセージペイロードを組み合わせて、アウトバウンドメッセージの数を減らします。

この動作は、 SimSpace Weaver Localと で同じです AWS クラウド。

メッセージペイロードのサイズ制限を超えました

メッセージペイロードサイズの現在の制限は、 SimSpace Weaver Localと の両方で 256 バイトです AWS クラウド。ペイロードが 256 バイトを超えるメッセージを送信すると、次のエラーが発生します。

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

SimSpace Weaver は各メッセージをチェックし、制限を超えたメッセージのみを拒否します。例えば、アプリが 10 件のメッセージを送信しようとして 1 件がチェックに失敗した場合、その 1 件のみが拒否されます。 SimSpace Weaver は他の 9 件のメッセージを送信します。

この動作は、 SimSpace Weaver Localと で同じです AWS クラウド。

送信先は送信元と同じです

アプリケーションは、所有するパーティションにメッセージを送信できません。アプリケーションが所有しているパーティションにメッセージを送信すると、次のエラーが発生します。

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

この動作は、 SimSpace Weaver Localと で同じです AWS クラウド。

ベストエフォートメッセージング

SimSpace Weaver はメッセージの配信を保証するものではありません。サービスは、後続のシミュレーションティックでメッセージの配信を完了しようとしますが、メッセージが失われたり遅延したりする可能性があります。

を使用する際のベストプラクティス SimSpace Weaver

を使用するときは、次のベストプラクティスをお勧めします SimSpace Weaver。

トピック

- 請求アラームの設定
- SimSpace Weaver Local を使用する
- 不要なシミュレーションを停止します
- 不要なリソースの削除
- バックアップの取得

請求アラームの設定

リソースが不要になった場合でも、 でリソースをプロビジョニング AWS し、常に実行しておくことは簡単です。その結果、コストが跳ね上がり、請求書を受け取って驚くことがあるかもしれません。Amazon CloudWatch では、コストが設定したしきい値を超えた場合にトリガーして通知するアラームを設定できます。コスト管理ツールを使用してコストを調べることができます。詳細については、以下を参照してください。

- 請求アラームを作成して推定 AWS 請求額をモニタリングする
- とは AWS Cost Management

SimSpace Weaver Local を使用する

SimSpace Weaver Local で SimSpace Weaver サービスにアップロードする前に、 を使用してシミュレーションを開発およびテストすることをお勧めします AWS クラウド。SimSpace Weaver Local を使用して開発することには以下のようなメリットがあります。

- 大量のアップロードを待つ必要はありません
- 作成できるローカルシミュレーションの数に制限はありません
- ローカルコンピューターの計算時間には料金は発生しません
- アプリケーションからコンソール出力に直接アクセスできます
- で再作成することなく、ローカルシミュレーションを変更、再構築、再起動する AWS クラウド

不要なシミュレーションを停止します

シミュレーションの実行中は、料金が発生します。料金が発生しないようにするには、シミュレーションを停止する必要があります。実行中のシミュレーションもシミュレーションの最大数のクォータにカウントされます。ロギングが設定されたシミュレーションを実行すると大量のログが生成される場合があり、それに対しても料金が発生します。追加料金が発生しないようにするには、必要のないシミュレーションをすべて中止する必要があります。

> **Important**
>
> シミュレーションクロックを停止してもシミュレーションは停止しません。クロックはアプリケーションへのティックの公開を停止するだけです。シミュレーションを停止すると、再開することはできません。

不要なリソースの削除

で作成する各シミュレーションは、他の AWS サービスのリソース SimSpace Weaver も作成します。これらの他のサービスのリソースやデータに対して料金が発生することがあります。実行中のシミュレーションと失敗したシミュレーションは、最大シミュレーション回数のクォータにカウントされます。新しいシミュレーションを開始するために、失敗した不要なシミュレーションを削除する必要があります。シミュレーションを削除すると、他の AWS サービスに存在するシミュレーションのリソースが削除されない場合があります。例えば、Amazon CloudWatch Logs のシミュレーションログデータは、削除するまでそのまま残ります。そのログデータには料金が発生します。不要になったシミュレーションに関連するリソースをすべて削除する必要があります。

バックアップの取得

すべてをバックアップし、バックアップする計画を立てることをお勧めします。データをバックアップ AWS する必要がないという理由だけで、 を前提とすべきではありません。シミュレーションの状態をバックアップする必要がある場合は、独自のシステムを作成する必要があります。必要に応じて本番環境のワークロードを別のワークロードにすばやく切り替えられるように、複数の を使用し AWS リージョン 、計画を立て AWS リージョン ることを検討してください。このサポートの詳細については SimSpace Weaver、 AWS リージョン 「」を参照してください[SimSpace Weaver エンドポイントとクォータ](#)。

のセキュリティ AWS SimSpace Weaver

のクラウドセキュリティが最優先事項 AWS です。お客様は AWS 、セキュリティを最も重視する組織の要件を満たすように構築されたデータセンターとネットワークアーキテクチャを活用できます。

セキュリティは、 AWS とお客様の間の責任共有です。責任共有モデルでは、これをクラウドのセキュリティおよびクラウド内のセキュリティと説明しています。

- クラウドのセキュリティ – AWS は、 で AWS サービスを実行するインフラストラクチャを保護する責任を担います AWS クラウド。 は、お客様が安全に使用できるサービス AWS も提供します。サードパーティーの監査者は、AWS コンプライアンスプログラムコンプライアンスプログラムの一環として、当社のセキュリティの有効性を定期的にテストおよび検証。が適用されるコンプライアンスプログラムの詳細については AWS SimSpace Weaver、「コンプライアンスプログラムAWS による対象範囲内のサービスコンプライアンスプログラム」を参照してください。
- クラウド内のセキュリティ – お客様の責任は、使用する AWS サービスによって決まります。また、ユーザーは、データの機密性、会社の要件、適用される法律や規制など、その他の要因についても責任を負います。

このドキュメントは、 を使用する際の責任共有モデルの適用方法を理解するのに役立ちます SimSpace Weaver。以下のトピックでは、セキュリティとコンプライアンスの目的を達成する SimSpace Weaver ように を設定する方法を示します。また、 SimSpace Weaver リソースのモニタリングや保護に役立つ他の AWS サービスの使用方法についても説明します。

トピック

- でのデータ保護 AWS SimSpace Weaver
- の Identity and Access Management AWS SimSpace Weaver
- のセキュリティイベントのログ記録とモニタリング AWS SimSpace Weaver
- のコンプライアンス検証 AWS SimSpace Weaver
- の耐障害性 AWS SimSpace Weaver
- のインフラストラクチャセキュリティ AWS SimSpace Weaver
- での設定と脆弱性の分析 AWS SimSpace Weaver
- のセキュリティのベストプラクティス SimSpace Weaver

でのデータ保護 AWS SimSpace Weaver

責任 AWS 共有モデル、 でのデータ保護に適用されます AWS SimSpace Weaver。このモデルで説明されているように、 AWS はすべての を実行するグローバルインフラストラクチャを保護する責任があります AWS クラウド。ユーザーは、このインフラストラクチャでホストされるコンテンツに対する管理を維持する責任があります。また、使用する「 AWS のサービス 」のセキュリティ設定と管理タスクもユーザーの責任となります。データプライバシーの詳細については、データプライバシーに関するよくある質問を参照してください。欧州でのデータ保護の詳細については、AWS セキュリティブログに投稿された AWS 責任共有モデルおよび GDPR のブログ記事を参照してください。

データ保護の目的で、認証情報を保護し AWS アカウント 、 AWS IAM Identity Center または AWS Identity and Access Management (IAM) を使用して個々のユーザーを設定することをお勧めします。この方法により、それぞれのジョブを遂行するために必要な権限のみが各ユーザーに付与されます。また、次の方法でデータを保護することもお勧めします:

- 各アカウントで多要素認証 (MFA) を使用します。
- SSL/TLS を使用して AWS リソースと通信します。TLS 1.2 が必須で、TLS 1.3 をお勧めします。
- で API とユーザーアクティビティのログ記録を設定します AWS CloudTrail。CloudTrail 証跡を使用して AWS アクティビティをキャプチャする方法については、「 AWS CloudTrail ユーザーガイド」のCloudTrail 証跡の使用」を参照してください。
- AWS 暗号化ソリューションと、 内のすべてのデフォルトのセキュリティコントロールを使用します AWS のサービス。
- Amazon Macie などの高度な管理されたセキュリティサービスを使用します。これらは、Amazon S3 に保存されている機密データの検出と保護を支援します。
- コマンドラインインターフェイスまたは API AWS を介して にアクセスするときに FIPS 140-3 検証済み暗号化モジュールが必要な場合は、FIPS エンドポイントを使用します。利用可能な FIPS エンドポイントの詳細については、「連邦情報処理規格 (FIPS) 140-3」を参照してください。

お客様の E メールアドレスなどの極秘または機密情報を、タグ、または [名前] フィールドなどの自由形式のテキストフィールドに含めないことを強くお勧めします。これは、コンソール SimSpace Weaver 、API、または SDK を使用して AWS CLIまたは他の AWS のサービス を操作する場合も同様です。 AWS SDKs タグ、または名前に使用される自由記述のテキストフィールドに入力したデータは、請求または診断ログに使用される場合があります。外部サーバーに URL を提供する場合、そのサーバーへのリクエストを検証できるように、認証情報を URL に含めないことを強くお勧めします。

保管中の暗号化

データがディスクなどの不揮発性 (永続的) データストレージに格納されている場合、データは保管中と見なされます。メモリやレジスタなどの揮発性データストレージにあるデータは、保管中とは見なされません。

を使用する場合 SimSpace Weaver、保管中のデータは次のとおりです。

- Amazon Simple Storage Service (Amazon S3) にアップロードするアプリケーションとスキーマ
- Amazon CloudWatch に保存されているシミュレーションログデータ

が内部的 SimSpace Weaver に使用する他のデータは、シミュレーションを停止した後も保持されません。

保管中のデータの暗号化の方法については、以下を参照してください。

- Amazon S3 でのデータの暗号化
- ログデータの暗号化

転送中の暗号化

AWS Command Line Interface（AWS CLI）、 AWS SDK、および SimSpace Weaver アプリケーション SDK を介した SimSpace Weaver API への接続では、署名バージョン 4 の署名プロセスで TLS 暗号化を使用します。 は、接続に使用するセキュリティ認証情報の IAM 定義のアクセスポリシーを使用して認証 AWS を管理します。

内部的には、TLS SimSpace Weaver を使用して、使用する他の AWS サービスに接続します。

> **Important**
>
> アプリケーションとそのクライアント間の通信には関係ありません SimSpace Weaver。必要に応じてシミュレーションクライアントとの通信を暗号化する責任は、お客様にあります。クライアント接続間でのすべての転送中のデータの暗号化を作成することをお勧めします。

暗号化ソリューションをサポートできる AWS サービスの詳細については、 AWS セキュリティブログを参照してください。

ネットワーク間トラフィックのプライバシー

SimSpace Weaver コンピューティングリソースは、すべての SimSpace Weaver お客様が共有する 1 つの Amazon VPC 内に存在します。すべての内部 SimSpace Weaver サービストラフィックは AWS ネットワーク内にとどまり、インターネットを経由しません。シミュレーションクライアントとアプリケーション間の通信はインターネットを経由します。

の Identity and Access Management AWS SimSpace Weaver

AWS Identity and Access Management (IAM) は、管理者が AWS リソースへのアクセスを安全に制御 AWS のサービス するのに役立つ です。IAM 管理者は、誰を認証 (サインイン) し、誰に SimSpace Weaver リソースの使用を許可する (アクセス許可を付与する) かを制御します。IAM は、追加料金なしで使用できる AWS のサービス です。

トピック

- 対象者
- アイデンティティを使用した認証
- ポリシーを使用したアクセスの管理
- が IAM と AWS SimSpace Weaver 連携する方法
- のアイデンティティベースのポリシーの例 AWS SimSpace Weaver
- がユーザーに代わって SimSpace Weaver 作成するアクセス許可
- サービス間の混乱した代理の防止
- AWS SimSpace Weaver ID とアクセスのトラブルシューティング

対象者

AWS Identity and Access Management (IAM) の使用方法は、作業内容によって異なります SimSpace Weaver。

サービスユーザー – SimSpace Weaver サービスを使用してジョブを実行する場合、管理者から必要な認証情報とアクセス許可が提供されます。さらに多くの SimSpace Weaver 機能を使用して作業を行う場合は、追加のアクセス許可が必要になることがあります。アクセスの管理方法を理解すると、管理者から適切な権限をリクエストするのに役に立ちます。 SimSpace Weaver機能にアクセスでき

ない場合は、「AWS SimSpace Weaver ID とアクセスのトラブルシューティング」を参照してください。

サービス管理者 – 社内の SimSpace Weaver リソースを担当している場合は、通常、 へのフルアクセスがあります SimSpace Weaver。サービスユーザーがどの SimSpace Weaver 機能やリソースにアクセスするかを決めるのは管理者の仕事です。その後、IAM 管理者にリクエストを送信して、サービスユーザーの権限を変更する必要があります。このページの情報を点検して、IAM の基本概念を理解してください。会社で IAM を使用する方法の詳細については SimSpace Weaver、「」を参照してくださいが IAM と AWS SimSpace Weaver 連携する方法。

IAM 管理者 - 管理者は、 SimSpace Weaverへのアクセスを管理するポリシーの書き込み方法の詳細について確認する場合があります。IAM で使用できる SimSpace Weaver アイデンティティベースのポリシーの例を表示するには、「」を参照してくださいのアイデンティティベースのポリシーの例 AWS SimSpace Weaver。

アイデンティティを使用した認証

認証とは、ID 認証情報 AWS を使用して にサインインする方法です。として、IAM ユーザーとして AWS アカウントのルートユーザー、または IAM ロールを引き受けることによって認証 (にサインイン AWS) される必要があります。

ID ソースを介して提供された認証情報を使用して、フェデレーティッド ID AWS として にサインインできます。 AWS IAM Identity Center （ IAM Identity Center) ユーザー、会社のシングルサインオン認証、Google または Facebook 認証情報は、フェデレーション ID の例です。フェデレーティッド ID としてサインインする場合、IAM ロールを使用して、前もって管理者により ID フェデレーションが設定されています。フェデレーション AWS を使用して にアクセスすると、間接的にロールを引き受けることになります。

ユーザーの種類に応じて、 AWS Management Console または AWS アクセスポータルにサインインできます。へのサインインの詳細については AWS、「 AWS サインイン ユーザーガイド」の「 へのサインイン AWS アカウント方法」を参照してください。

AWS プログラムで にアクセスする場合、 はソフトウェア開発キット (SDK) とコマンドラインインターフェイス (CLI) AWS を提供し、認証情報を使用してリクエストを暗号化して署名します。AWS ツールを使用しない場合は、自分でリクエストに署名する必要があります。リクエストに自分で署名する推奨方法の使用については、「IAM ユーザーガイド」の「API リクエストに対するAWS Signature Version 4」を参照してください。

使用する認証方法を問わず、追加セキュリティ情報の提供をリクエストされる場合もあります。たとえば、 では、アカウントのセキュリティを高めるために多要素認証 (MFA) を使用する AWS ことを

お勧めします。詳細については、「AWS IAM Identity Center ユーザーガイド」の「多要素認証」および「IAM ユーザーガイド」の「IAM のAWS 多要素認証」を参照してください。

AWS アカウント ルートユーザー

を作成するときは AWS アカウント、アカウント内のすべての およびリソースへの AWS のサービス 完全なアクセス権を持つ 1 つのサインインアイデンティティから始めます。この ID は AWS アカウント ルートユーザーと呼ばれ、アカウントの作成に使用した E メールアドレスとパスワードでサインインすることでアクセスできます。日常的なタスクには、ルートユーザーを使用しないことを強くお勧めします。ルートユーザーの認証情報は保護し、ルートユーザーでしか実行できないタスクを実行するときに使用します。ルートユーザーとしてサインインする必要があるタスクの完全なリストについては、「IAM ユーザーガイド」の「ルートユーザー認証情報が必要なタスク」を参照してください。

フェデレーティッドアイデンティティ

ベストプラクティスとして、管理者アクセスを必要とするユーザーを含む人間のユーザーに、ID プロバイダーとのフェデレーションを使用して一時的な認証情報 AWS のサービス を使用して にアクセスすることを要求します。

フェデレーティッド ID は、エンタープライズユーザーディレクトリ、ウェブ ID プロバイダー、、AWS Directory Serviceアイデンティティセンターディレクトリ、または ID ソースを介して提供された認証情報 AWS のサービス を使用して にアクセスするユーザーです。フェデレーティッド ID がアクセスすると AWS アカウント、ロールを引き受け、ロールは一時的な認証情報を提供します。

アクセスを一元管理する場合は、 AWS IAM Identity Centerを使用することをお勧めします。IAM Identity Center でユーザーとグループを作成するか、独自の ID ソースのユーザーとグループのセットに接続して同期し、すべての AWS アカウント とアプリケーションで使用できます。IAM Identity Center の詳細については、「AWS IAM Identity Center ユーザーガイド」の「What is IAM Identity Center?」(IAM Identity Center とは) を参照してください。

IAM ユーザーとグループ

IAM ユーザーは、単一のユーザーまたはアプリケーションに対して特定のアクセス許可 AWS アカウント を持つ 内の ID です。可能であれば、パスワードやアクセスキーなどの長期的な認証情報を保有する IAM ユーザーを作成する代わりに、一時的な認証情報を使用することをお勧めします。ただし、IAM ユーザーでの長期的な認証情報が必要な特定のユースケースがある場合は、アクセスキーをローテーションすることをお勧めします。詳細については、「IAM ユーザーガイド」の「長期的な認証情報を必要とするユースケースのためにアクセスキーを定期的にローテーションする」を参照してください。

IAM グループは、IAM ユーザーの集団を指定するアイデンティティです。グループとしてサインインすることはできません。グループを使用して、複数のユーザーに対して一度に権限を指定できます。多数のユーザーグループがある場合、グループを使用することで権限の管理が容易になります。例えば、IAMAdmins という名前のグループを設定して、そのグループに IAM リソースを管理する許可を与えることができます。

ユーザーは、ロールとは異なります。ユーザーは 1 人の人または 1 つのアプリケーションに一意に関連付けられますが、ロールはそれを必要とする任意の人が引き受けるようになっています。ユーザーには永続的な長期の認証情報がありますが、ロールでは一時認証情報が提供されます。詳細については、「IAM ユーザーガイド」の「IAM ユーザーに関するユースケース」を参照してください。

IAM ロール

IAM ロールは、特定のアクセス許可 AWS アカウント を持つ 内の ID です。これは IAM ユーザーに似ていますが、特定のユーザーには関連付けられていません。で IAM ロールを一時的に引き受けるには AWS Management Console、ユーザーから IAM ロール (コンソール) に切り替えることができます。ロールを引き受けるには、 または AWS API オペレーションを AWS CLI 呼び出すか、カスタム URL を使用します。ロールを使用する方法の詳細については、「IAM ユーザーガイド」の「ロールを引き受けるための各種方法」を参照してください。

IAM ロールと一時的な認証情報は、次の状況で役立ちます:

- フェデレーションユーザーアクセス – フェデレーティッド ID に許可を割り当てるには、ロールを作成してそのロールの許可を定義します。フェデレーティッド ID が認証されると、その ID はロールに関連付けられ、ロールで定義されている許可が付与されます。フェデレーションのロールについては、「IAM ユーザーガイド」の「サードパーティー ID プロバイダー (フェデレーション) 用のロールを作成する」を参照してください。IAM Identity Center を使用する場合は、許可セットを設定します。アイデンティティが認証後にアクセスできるものを制御するため、IAM Identity Center は、権限セットを IAM のロールに関連付けます。アクセス許可セットの詳細については、「AWS IAM Identity Center User Guide」の「Permission sets」を参照してください。
- 一時的な IAM ユーザー権限 - IAM ユーザーまたはロールは、特定のタスクに対して複数の異なる権限を一時的に IAM ロールで引き受けることができます。
- クロスアカウントアクセス - IAM ロールを使用して、自分のアカウントのリソースにアクセスすることを、別のアカウントの人物 (信頼済みプリンシパル) に許可できます。クロスアカウントアクセス権を付与する主な方法は、ロールを使用することです。ただし、一部の では AWS のサービス、 (プロキシとしてロールを使用する代わりに) リソースに直接ポリシーをアタッチできます。クロスアカウントアクセスにおけるロールとリソースベースのポリシーの違いについては、

「IAM ユーザーガイド」の「IAM でのクロスアカウントのリソースへのアクセス」を参照してください。

- クロスサービスアクセス — 一部の は他の の機能 AWS のサービス を使用します AWS のサービス。例えば、あるサービスで呼び出しを行うと、通常そのサービスによって Amazon EC2 でアプリケーションが実行されたり、Amazon S3 にオブジェクトが保存されたりします。サービスでは、呼び出し元プリンシパルの許可、サービスロール、またはサービスリンクロールを使用してこれを行う場合があります。
  - 転送アクセスセッション (FAS) – IAM ユーザーまたはロールを使用してアクションを実行すると AWS、プリンシパルと見なされます。一部のサービスを使用する際に、アクションを実行することで、別のサービスの別のアクションがトリガーされることがあります。FAS は、 を呼び出すプリンシパルのアクセス許可を AWS のサービス、ダウンストリームサービス AWS のサービス へのリクエストをリクエストする と組み合わせて使用します。FAS リクエストは、サービスが他の AWS のサービス またはリソースとのやり取りを完了する必要があるリクエストを受け取った場合にのみ行われます。この場合、両方のアクションを実行するためのアクセス許可が必要です。FAS リクエストを行う際のポリシーの詳細については、「転送アクセスセッション」を参照してください。
  - サービスロール - サービスがユーザーに代わってアクションを実行するために引き受ける IAM ロールです。IAM 管理者は、IAM 内からサービスロールを作成、変更、削除することができます。詳細については、「IAM ユーザーガイド」の「AWS のサービスに許可を委任するロールを作成する」を参照してください。
  - サービスにリンクされたロール – サービスにリンクされたロールは、 にリンクされたサービスロールの一種です AWS のサービス。サービスは、ユーザーに代わってアクションを実行するロールを引き受けることができます。サービスにリンクされたロールは に表示され AWS アカウント 、サービスによって所有されます。IAM 管理者は、サービスリンクロールのアクセス許可を表示できますが、編集することはできません。
- Amazon EC2 で実行されているアプリケーション – IAM ロールを使用して、EC2 インスタンスで実行され、 AWS CLI または AWS API リクエストを行うアプリケーションの一時的な認証情報を管理できます。これは、EC2 インスタンス内でのアクセスキーの保存に推奨されます。EC2 インスタンスに AWS ロールを割り当て、そのすべてのアプリケーションで使用できるようにするには、インスタンスにアタッチされたインスタンスプロファイルを作成します。インスタンスプロファイルにはロールが含まれ、EC2 インスタンスで実行されるプログラムは一時的な認証情報を取得できます。詳細については、「IAM ユーザーガイド」の「Amazon EC2 インスタンスで実行されるアプリケーションに IAM ロールを使用して許可を付与する」を参照してください。

ポリシーを使用したアクセスの管理

でアクセスを制御する AWS には、ポリシーを作成し、ID AWS またはリソースにアタッチします。ポリシーは AWS 、アイデンティティまたはリソースに関連付けられているときにアクセス許可を定義する のオブジェクトです。 は、プリンシパル (ユーザー、ルートユーザー、またはロールセッション) がリクエストを行うときに、これらのポリシー AWS を評価します。ポリシーでの権限により、リクエストが許可されるか拒否されるかが決まります。ほとんどのポリシーは JSON ドキュメント AWS として に保存されます。JSON ポリシードキュメントの構造と内容の詳細については、「IAM ユーザーガイド」の「JSON ポリシー概要」を参照してください。

管理者は JSON AWS ポリシーを使用して、誰が何にアクセスできるかを指定できます。つまり、どのプリンシパルがどのリソースに対してどのような条件下でアクションを実行できるかということです。

デフォルトでは、ユーザーやロールに権限はありません。IAM 管理者は、リソースで必要なアクションを実行するための権限をユーザーに付与する IAM ポリシーを作成できます。その後、管理者はロールに IAM ポリシーを追加し、ユーザーはロールを引き受けることができます。

IAM ポリシーは、オペレーションの実行方法を問わず、アクションの許可を定義します。例えば、`iam:GetRole` アクションを許可するポリシーがあるとします。そのポリシーを持つユーザーは、 AWS Management Console、、 AWS CLIまたは AWS API からロール情報を取得できます。

アイデンティティベースのポリシー

アイデンティティベースポリシーは、IAM ユーザーグループ、ユーザーのグループ、ロールなど、アイデンティティにアタッチできる JSON 許可ポリシードキュメントです。これらのポリシーは、ユーザーとロールが実行できるアクション、リソース、および条件をコントロールします。アイデンティティベースポリシーの作成方法については、「IAM ユーザーガイド」の「カスタマー管理ポリシーでカスタム IAM アクセス許可を定義する」を参照してください。

アイデンティティベースのポリシーは、さらにインラインポリシーまたはマネージドポリシーに分類できます。インラインポリシーは、単一のユーザー、グループ、またはロールに直接埋め込まれています。管理ポリシーは、 内の複数のユーザー、グループ、ロールにアタッチできるスタンドアロンポリシーです AWS アカウント。管理ポリシーには、 AWS 管理ポリシーとカスタマー管理ポリシーが含まれます。マネージドポリシーまたはインラインポリシーのいずれかを選択する方法については、「IAM ユーザーガイド」の「管理ポリシーとインラインポリシーのいずれかを選択する」を参照してください。

リソースベースのポリシー

リソースベースのポリシーは、リソースに添付する JSON ポリシードキュメントです。リソースベースのポリシーには例として、IAM ロールの信頼ポリシーや Amazon S3 バケットポリシーがあげられます。リソースベースのポリシーをサポートするサービスでは、サービス管理者はポリシーを使用して特定のリソースへのアクセスを制御できます。ポリシーがアタッチされているリソースの場合、指定されたプリンシパルがそのリソースに対して実行できるアクションと条件は、ポリシーによって定義されます。リソースベースのポリシーでは、プリンシパルを指定する必要があります。プリンシパルには、アカウント、ユーザー、ロール、フェデレーティッドユーザー、または を含めることができます AWS のサービス。

リソースベースのポリシーは、そのサービス内にあるインラインポリシーです。リソースベースのポリシーでは、IAM の AWS マネージドポリシーを使用できません。

アクセスコントロールリスト (ACL)

アクセスコントロールリスト (ACL) は、どのプリンシパル (アカウントメンバー、ユーザー、またはロール) がリソースにアクセスするための許可を持つかを制御します。ACL はリソースベースのポリシーに似ていますが、JSON ポリシードキュメント形式は使用しません。

Amazon S3、および Amazon VPC は AWS WAF、ACLs。ACL の詳細については、「Amazon Simple Storage Service デベロッパーガイド」の「アクセスコントロールリスト (ACL) の概要」を参照してください。

その他のポリシータイプ

AWS は、一般的でない追加のポリシータイプをサポートしています。これらのポリシータイプでは、より一般的なポリシータイプで付与された最大の権限を設定できます。

- アクセス許可の境界 - アクセス許可の境界は、アイデンティティベースポリシーによって IAM エンティティ (IAM ユーザーまたはロール) に付与できる権限の上限を設定する高度な機能です。エンティティにアクセス許可の境界を設定できます。結果として得られる権限は、エンティティのアイデンティティベースポリシーとそのアクセス許可の境界の共通部分になります。`Principal` フィールドでユーザーまたはロールを指定するリソースベースのポリシーでは、アクセス許可の境界は制限されません。これらのポリシーのいずれかを明示的に拒否した場合、権限は無効になります。アクセス許可の境界の詳細については、「IAM ユーザーガイド」の「IAM エンティティのアクセス許可の境界」を参照してください。
- サービスコントロールポリシー (SCPs) – SCPsは、 の組織または組織単位 (OU) の最大アクセス許可を指定する JSON ポリシーです AWS Organizations。 AWS Organizations は、ビジネスが所

有する複数の をグループ化して一元管理するためのサービス AWS アカウント です。組織内のすべての機能を有効にすると、サービスコントロールポリシー (SCP) を一部またはすべてのアカウントに適用できます。SCP は、各 を含むメンバーアカウントのエンティティのアクセス許可を制限します AWS アカウントのルートユーザー。Organizations と SCP の詳細については、「AWS Organizations ユーザーガイド」の「サービスコントロールポリシー (SCP)」を参照してください。

- リソースコントロールポリシー (RCP) – RCP は、所有する各リソースにアタッチされた IAM ポリシーを更新することなく、アカウント内のリソースに利用可能な最大数のアクセス許可を設定するために使用できる JSON ポリシーです。RCP は、メンバーアカウントのリソースのアクセス許可を制限し、組織に属しているかどうかにかかわらず AWS アカウントのルートユーザー、 を含む ID の有効なアクセス許可に影響を与える可能性があります。RCP をサポートする のリストを含む Organizations と RCP の詳細については、AWS Organizations RCPs「リソースコントロールポリシー (RCPs」を参照してください。 AWS のサービス
- セッションポリシー - セッションポリシーは、ロールまたはフェデレーションユーザーの一時的なセッションをプログラムで作成する際にパラメータとして渡す高度なポリシーです。結果としてセッションの権限は、ユーザーまたはロールのアイデンティティベースポリシーとセッションポリシーの共通部分になります。また、リソースベースのポリシーから権限が派生する場合もあります。これらのポリシーのいずれかを明示的に拒否した場合、権限は無効になります。詳細については、「IAM ユーザーガイド」の「セッションポリシー」を参照してください。

複数のポリシータイプ

1 つのリクエストに複数のタイプのポリシーが適用されると、結果として作成される権限を理解するのがさらに難しくなります。が複数のポリシータイプが関与する場合にリクエストを許可するかどうか AWS を決定する方法については、「IAM ユーザーガイド」の「ポリシー評価ロジック」を参照してください。

が IAM と AWS SimSpace Weaver 連携する方法

IAM を使用して へのアクセスを管理する前に SimSpace Weaver、 で使用できる IAM 機能を確認してください SimSpace Weaver。

で使用できる IAM 機能 AWS SimSpace Weaver

IAM 機能	SimSpace Weaver サポート
アイデンティティベースポリシー	はい
リソースベースのポリシー	いいえ
ポリシーアクション	はい
ポリシーリソース	はい
ポリシー条件キー (サービス固有)	はい
ACL	いいえ
ABAC (ポリシー内のタグ)	あり
一時的な認証情報	はい
プリンシパル権限	はい
サービスロール	はい
サービスリンクロール	いいえ

SimSpace Weaver およびその他の AWS のサービスがほとんどの IAM 機能とどのように連携するかの概要については、「IAM ユーザーガイド」のAWS 「IAM と連携する のサービス」を参照してください。

のアイデンティティベースのポリシー SimSpace Weaver

アイデンティティベースのポリシーのサポート: あり

アイデンティティベースポリシーは、IAM ユーザーグループ、ユーザーのグループ、ロールなど、アイデンティティにアタッチできる JSON 許可ポリシードキュメントです。これらのポリシーは、ユーザーとロールが実行できるアクション、リソース、および条件をコントロールします。ID ベースのポリシーの作成方法については、「IAM ユーザーガイド」の「カスタマー管理ポリシーでカスタム IAM アクセス許可を定義する」を参照してください。

IAM アイデンティティベースのポリシーでは、許可または拒否するアクションとリソース、およびアクションを許可または拒否する条件を指定できます。プリンシパルは、それが添付されているユーザーまたはロールに適用されるため、アイデンティティベースのポリシーでは指定できません。JSON ポリシーで使用できるすべての要素について学ぶには、「IAM ユーザーガイド」の「IAM JSON ポリシーの要素のリファレンス」を参照してください。

のアイデンティティベースのポリシーの例 SimSpace Weaver

SimSpace Weaver アイデンティティベースのポリシーの例を表示するには、「」を参照してくださいのアイデンティティベースのポリシーの例 AWS SimSpace Weaver。

内のリソースベースのポリシー SimSpace Weaver

リソースベースのポリシーのサポート: なし

リソースベースのポリシーは、リソースに添付する JSON ポリシードキュメントです。リソースベースのポリシーには例として、IAM ロールの信頼ポリシーや Amazon S3 バケットポリシーがあげられます。リソースベースのポリシーをサポートするサービスでは、サービス管理者はポリシーを使用して特定のリソースへのアクセスを制御できます。ポリシーがアタッチされているリソースの場合、指定されたプリンシパルがそのリソースに対して実行できるアクションと条件は、ポリシーによって定義されます。リソースベースのポリシーでは、プリンシパルを指定する必要があります。プリンシパルには、アカウント、ユーザー、ロール、フェデレーティッドユーザー、または を含めることができます AWS のサービス。

クロスアカウントアクセスを有効にするには、アカウント全体、または別のアカウントの IAM エンティティをリソースベースのポリシーのプリンシパルとして指定します。リソースベースのポリシーにクロスアカウントのプリンシパルを追加しても、信頼関係は半分しか確立されない点に注意してください。プリンシパルとリソースが異なる場合 AWS アカウント、信頼されたアカウントの IAM 管理者は、プリンシパルエンティティ (ユーザーまたはロール) にリソースへのアクセス許可も付与する必要があります。IAM 管理者は、アイデンティティベースのポリシーをエンティティにアタッチすることで権限を付与します。ただし、リソースベースのポリシーで、同じアカウントのプリンシパルへのアクセス権が付与されている場合は、アイデンティティベースのポリシーをさらに付与する必要はありません。詳細については、「IAM ユーザーガイド」の「IAM でのクロスアカウントリソースアクセス」を参照してください。

のポリシーアクション SimSpace Weaver

ポリシーアクションのサポート:あり

管理者は JSON AWS ポリシーを使用して、誰が何にアクセスできるかを指定できます。つまり、どのプリンシパルがどのリソースに対してどのような条件下でアクションを実行できるかということです。

JSON ポリシーの `Action` 要素にはポリシー内のアクセスを許可または拒否するために使用できるアクションが記述されます。ポリシーアクションの名前は通常、関連付けられた AWS API オペレーションと同じです。一致する API オペレーションのない許可のみのアクションなど、いくつかの例外があります。また、ポリシーに複数のアクションが必要なオペレーションもあります。これらの追加アクションは依存アクションと呼ばれます。

このアクションは関連付けられたオペレーションを実行するためのアクセス許可を付与するポリシーで使用されます。

SimSpace Weaver アクションのリストを確認するには、「サービス認可リファレンス」の「 で定義されるアクション AWS SimSpace Weaver」を参照してください。

のポリシーアクションは、アクションの前に次のプレフィックス SimSpace Weaver を使用します。

```
simspaceweaver
```

単一のステートメントで複数のアクションを指定するには、アクションをカンマで区切ります。

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

SimSpace Weaver アイデンティティベースのポリシーの例を表示するには、「」を参照してください のアイデンティティベースのポリシーの例 AWS SimSpace Weaver。

のポリシーリソース SimSpace Weaver

ポリシーリソースのサポート: あり

管理者は JSON AWS ポリシーを使用して、誰が何にアクセスできるかを指定できます。つまり、どのプリンシパルがどのリソースに対してどのような条件下でアクションを実行できるかということです。

`Resource` JSON ポリシー要素はアクションが適用されるオブジェクトを指定します。ステートメントには`Resource` または `NotResource` 要素を含める必要があります。ベストプラクティスとし

て、Amazon リソースネーム (ARN) を使用してリソースを指定します。これは、リソースレベルの許可と呼ばれる特定のリソースタイプをサポートするアクションに対して実行できます。

オペレーションのリスト化など、リソースレベルの権限をサポートしないアクションの場合は、ステートメントがすべてのリソースに適用されることを示すために、ワイルドカード (*) を使用します。

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

SimSpace Weaver リソースタイプとその ARNs「 で定義されるリソース AWS SimSpace Weaver」を参照してください。 どのアクションで各リソースの ARN を指定できるかについては、「AWS SimSpace Weaverで定義されるアクション」を参照してください。

SimSpace Weaver アイデンティティベースのポリシーの例を表示するには、「」を参照してくださいのアイデンティティベースのポリシーの例 AWS SimSpace Weaver。

のポリシー条件キー SimSpace Weaver

サービス固有のポリシー条件キーのサポート: あり

管理者は JSON AWS ポリシーを使用して、誰が何にアクセスできるかを指定できます。つまり、どのプリンシパルがどのリソースに対してどのような条件下でアクションを実行できるかということです。

`Condition` 要素 (または `Condition` ブロック) を使用すると、ステートメントが有効な条件を指定できます。`Condition` 要素はオプションです。イコールや未満などの 条件演算子 を使用して条件式を作成して、ポリシーの条件とリクエスト内の値を一致させることができます。

1 つのステートメントに複数の `Condition` 要素を指定する場合、または 1 つの `Condition` 要素に複数のキーを指定する場合、 AWS では `AND` 論理演算子を使用してそれらを評価します。1 つの条件キーに複数の値を指定すると、 は論理`OR`オペレーションを使用して条件 AWS を評価します。ステートメントの権限が付与される前にすべての条件が満たされる必要があります。

条件を指定する際にプレースホルダー変数も使用できます。例えば IAM ユーザーに、IAM ユーザー名がタグ付けされている場合のみリソースにアクセスできる権限を付与することができます。詳細については、「IAM ユーザーガイド」の「IAM ポリシーの要素: 変数およびタグ」を参照してください。

AWS は、グローバル条件キーとサービス固有の条件キーをサポートしています。すべての AWS グローバル条件キーを確認するには、「IAM ユーザーガイド」のAWS 「グローバル条件コンテキストキー」を参照してください。

SimSpace Weaver 条件キーのリストを確認するには、「サービス認可リファレンス」の「 の条件キー AWS SimSpace Weaver」を参照してください。条件キーを使用できるアクションとリソースについては、「 で定義されるアクション AWS SimSpace Weaver」を参照してください。

SimSpace Weaver アイデンティティベースのポリシーの例を表示するには、「」を参照してください のアイデンティティベースのポリシーの例 AWS SimSpace Weaver。

SimSpace Weaverのアクセスコントロールリスト (ACL)

ACL のサポート: なし

アクセスコントロールリスト (ACL) は、どのプリンシパル (アカウントメンバー、ユーザー、またはロール) がリソースにアクセスするための許可を持つかを制御します。ACL はリソースベースのポリシーに似ていますが、JSON ポリシードキュメント形式は使用しません。

を使用した属性ベースのアクセスコントロール (ABAC) SimSpace Weaver

ABAC (ポリシー内のタグ) のサポート: あり

属性ベースのアクセス制御 (ABAC) は、属性に基づいてアクセス許可を定義する認可戦略です。では AWS、これらの属性はタグと呼ばれます。タグは、IAM エンティティ (ユーザーまたはロール) および多くの AWS リソースにアタッチできます。エンティティとリソースのタグ付けは、ABAC の最初の手順です。その後、プリンシパルのタグがアクセスしようとしているリソースのタグと一致した場合にオペレーションを許可するように ABAC ポリシーをします。

ABAC は、急成長する環境やポリシー管理が煩雑になる状況で役立ちます。

タグに基づいてアクセスを管理するには、`aws:ResourceTag/key-name`、`aws:RequestTag/key-name`、または `aws:TagKeys` の条件キーを使用して、ポリシーの条件要素でタグ情報を提供します。

サービスがすべてのリソースタイプに対して 3 つの条件キーすべてをサポートする場合、そのサービスの値はありです。サービスが一部のリソースタイプに対してのみ 3 つの条件キーのすべてをサポートする場合、値は「部分的」になります。

ABAC の詳細については、「IAM ユーザーガイド」の「ABAC 認可でアクセス許可を定義する」を参照してください。ABAC をセットアップする手順を説明するチュートリアルについては、「IAM ユーザーガイド」の「属性ベースのアクセスコントロール (ABAC) を使用する」を参照してください。

での一時的な認証情報の使用 SimSpace Weaver

一時的な認証情報のサポート: あり

一部の AWS のサービス は、一時的な認証情報を使用してサインインすると機能しません。一時的な認証情報 AWS のサービス を使用する場合などの詳細については、IAM ユーザーガイドAWS のサービス の「IAM と連携する 」を参照してください。

ユーザー名とパスワード以外の方法 AWS Management Console を使用して にサインインする場合、一時的な認証情報を使用します。たとえば、会社のシングルサインオン (SSO) リンク AWS を使用して にアクセスすると、そのプロセスによって一時的な認証情報が自動的に作成されます。また、ユーザーとしてコンソールにサインインしてからロールを切り替える場合も、一時的な認証情報が自動的に作成されます。ロールの切り替えに関する詳細については、「IAM ユーザーガイド」の「ユーザーから IAM ロールに切り替える (コンソール)」を参照してください。

一時的な認証情報は、 AWS CLI または AWS API を使用して手動で作成できます。その後、これらの一時的な認証情報を使用してアクセスすることができます AWS。長期的なアクセスキーを使用する代わりに、一時的な認証情報を動的に生成 AWS することをお勧めします。詳細については、「IAM の一時的セキュリティ認証情報」を参照してください。

のクロスサービスプリンシパルのアクセス許可 SimSpace Weaver

転送アクセスセッション (FAS) のサポート: あり

IAM ユーザーまたはロールを使用して でアクションを実行すると AWS、プリンシパルと見なされます。一部のサービスを使用する際に、アクションを実行することで、別のサービスの別のアクションがトリガーされることがあります。FAS は、 を呼び出すプリンシパルのアクセス許可を AWS のサービス、ダウンストリームサービス AWS のサービス へのリクエストをリクエストする と組み合わせて使用します。FAS リクエストは、サービスが他の AWS のサービス またはリソースとのやり取りを完了する必要があるリクエストを受け取った場合にのみ行われます。この場合、両方のアクションを実行するためのアクセス許可が必要です。FASリクエストを行う際のポリシーの詳細については、「転送アクセスセッション」を参照してください。

SimSpace Weaverのサービスロール

サービスロールのサポート: あり

サービスロールとは、サービスがユーザーに代わってアクションを実行するために引き受ける IAM ロールです。IAM 管理者は、IAM 内からサービスロールを作成、変更、削除できます。詳細については、「IAM ユーザーガイド」の「AWS のサービスに許可を委任するロールを作成する」を参照してください。

> **Warning**
>
> サービスロールのアクセス許可を変更すると、 SimSpace Weaver 機能が破損する可能性があります。 SimSpace Weaver が指示する場合にのみ、サービスロールを編集します。

SimSpace Weaver アプリケーション SDK スクリプトは、 AWS CloudFormation テンプレートを使用して、シミュレーションをサポートする他の AWS サービスにリソースを作成します。これらのリソースの 1 つはシミュレーションのアプリケーションロールです。 は、CloudWatch Logs にログデータを書き込むなど、 AWS アカウント ユーザーに代わって でアクションを実行するアプリケーションロール SimSpace Weaver を引き受けます。アプリケーションロールの詳細については、「がユーザーに代わって SimSpace Weaver 作成するアクセス許可」を参照してください。

のサービスにリンクされたロール SimSpace Weaver

サービスにリンクされたロールのサポート: なし

サービスにリンクされたロールは、 にリンクされたサービスロールの一種です AWS のサービス。サービスは、ユーザーに代わってアクションを実行するロールを引き受けることができます。サービスにリンクされたロールは に表示され AWS アカウント 、サービスによって所有されます。IAM 管理者は、サービスにリンクされたロールのアクセス許可を表示できますが、編集することはできません。

サービスにリンクされたロールの作成または管理の詳細については、「IAM と提携するAWS のサービス」を参照してください。表の「サービスリンクロール」列に Yes と記載されたサービスを見つけます。サービスにリンクされたロールに関するドキュメントをサービスで表示するには、[はい] リンクを選択します。

のアイデンティティベースのポリシーの例 AWS SimSpace Weaver

デフォルトでは、 ユーザーおよびロールには、 SimSpace Weaver リソースを作成または変更する権限はありません。また、、 AWS Command Line Interface （ AWS CLI ） AWS Management Console、または AWS API を使用してタスクを実行することはできません。IAM 管理者は、リソースで必要なアクションを実行するための権限をユーザーに付与する IAM ポリシーを作成できます。その後、管理者はロールに IAM ポリシーを追加し、ユーザーはロールを引き継ぐことができます。

これらサンプルの JSON ポリシードキュメントを使用して、IAM アイデンティティベースのポリシーを作成する方法については、「IAM ユーザーガイド」の「IAM ポリシーを作成する (コンソール)」を参照してください。

各リソースタイプの ARN の形式など SimSpace Weaver、 で定義されるアクションとリソースタイプの詳細については、「サービス認可リファレンス」の「 のアクション、リソース、および条件キー AWS SimSpace Weaver」を参照してください。 ARNs

トピック

- ポリシーに関するベストプラクティス
- SimSpace Weaver コンソールを使用する
- 自分の権限の表示をユーザーに許可する
- ユーザーにシミュレーションの作成と実行を許可する

ポリシーに関するベストプラクティス

ID ベースのポリシーは、アカウント内の SimSpace Weaver リソースを作成、アクセス、または削除できるかどうかを決定します。これらのアクションを実行すると、 AWS アカウントに料金が発生する可能性があります。アイデンティティベースポリシーを作成したり編集したりする際には、以下のガイドラインと推奨事項に従ってください:

- AWS 管理ポリシーを開始し、最小特権のアクセス許可に移行する – ユーザーとワークロードにアクセス許可の付与を開始するには、多くの一般的なユースケースにアクセス許可を付与するAWS 管理ポリシーを使用します。これらは で使用できます AWS アカウント。ユースケースに固有の AWS カスタマー管理ポリシーを定義することで、アクセス許可をさらに減らすことをお勧めします。詳細については、「IAM ユーザーガイド」の「AWS マネージドポリシー」または「ジョブ機能のAWS マネージドポリシー」を参照してください。
- 最小特権を適用する – IAM ポリシーで許可を設定する場合は、タスクの実行に必要な許可のみを付与します。これを行うには、特定の条件下で特定のリソースに対して実行できるアクションを定義します。これは、最小特権アクセス許可とも呼ばれています。IAM を使用して許可を適用する方法の詳細については、「IAM ユーザーガイド」の「IAM でのポリシーとアクセス許可」を参照してください。
- IAM ポリシーで条件を使用してアクセスをさらに制限する - ポリシーに条件を追加して、アクションやリソースへのアクセスを制限できます。例えば、ポリシー条件を記述して、すべてのリクエストを SSL を使用して送信するように指定できます。条件を使用して、サービスアクションが などの特定の を通じて使用されている場合に AWS のサービス、サービスアクションへのアクセスを許可することもできます AWS CloudFormation。詳細については、「IAM ユーザーガイド」の「IAM JSON ポリシー要素:条件」を参照してください。
- IAM Access Analyzer を使用して IAM ポリシーを検証し、安全で機能的な権限を確保する - IAM Access Analyzer は、新規および既存のポリシーを検証して、ポリシーが IAM ポリシー言語

(JSON) および IAM のベストプラクティスに準拠するようにします。IAM アクセスアナライザーは 100 を超えるポリシーチェックと実用的な推奨事項を提供し、安全で機能的なポリシーの作成をサポートします。詳細については、「IAM ユーザーガイド」の「IAM Access Analyzer でポリシーを検証する」を参照してください。

- 多要素認証 (MFA) を要求する – で IAM ユーザーまたはルートユーザーを必要とするシナリオがある場合は AWS アカウント、MFA をオンにしてセキュリティを強化します。API オペレーションが呼び出されるときに MFA を必須にするには、ポリシーに MFA 条件を追加します。詳細については、「IAM ユーザーガイド」の「MFA を使用した安全な API アクセス」を参照してください。

IAM でのベストプラクティスの詳細については、「IAM ユーザーガイド」の「IAM でのセキュリティのベストプラクティス」を参照してください。

SimSpace Weaver コンソールを使用する

AWS SimSpace Weaver コンソールにアクセスするには、最小限のアクセス許可のセットが必要です。これらのアクセス許可により、 の SimSpace Weaver リソースの詳細を一覧表示および表示できます AWS アカウント。最小限必要な許可よりも制限が厳しいアイデンティティベースのポリシーを作成すると、そのポリシーを持つエンティティ (ユーザーまたはロール) に対してコンソールが意図したとおりに機能しません。

AWS CLI または AWS API のみを呼び出すユーザーには、最小限のコンソールアクセス許可を付与する必要はありません。代わりに、実行しようとしている API オペレーションに一致するアクションのみへのアクセスが許可されます。

ユーザーとロールが SimSpace Weaver 引き続きコンソールを使用できるようにするには、エンティティに SimSpace Weaver *ConsoleAccess*または *ReadOnly* AWS 管理ポリシーもアタッチします。詳細については、「IAM ユーザーガイド」の「ユーザーへのアクセス許可の追加」を参照してください。

自分の権限の表示をユーザーに許可する

この例では、ユーザーアイデンティティにアタッチされたインラインおよびマネージドポリシーの表示を IAM ユーザーに許可するポリシーの作成方法を示します。このポリシーには、コンソールで、または AWS CLI または AWS API を使用してプログラムでこのアクションを実行するアクセス許可が含まれています。

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

ユーザーにシミュレーションの作成と実行を許可する

この IAM ポリシーの例には、 SimSpace Weaverでのシミュレーションの作成と実行に必要な基本のアクセス許可が含まれます。

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

がユーザーに代わって SimSpace Weaver 作成するアクセス許可

SimSpace Weaver プロジェクトを作成すると、サービスは という名前の AWS Identity and Access Management (IAM) ロール`weaver-`*`project-name`*`-app-role`と IAM 信頼ポリシーを作成します。

信頼ポリシーでは SimSpace Weaver 、 がユーザーに代わってオペレーションを実行できるようにロールを引き受けることができます。

アプリケーションロールのアクセス許可ポリシー

シミュレーションアプリケーションのロールは以下のアクセス許可ポリシーを使用します。

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

アプリケーションロール信頼ポリシー

SimSpace Weaver は、シミュレーションアプリケーションロールに信頼関係を信頼ポリシーとして追加します。 は、次の例のように、シミュレーションごとに信頼ポリシー SimSpace Weaver を作成します。

```
{
  "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**
>
> この例では、アカウント番号は 111122223333 で、シミュレーション名は MySimName です。これらの値は信頼ポリシーによって異なります。

サービス間の混乱した代理の防止

「混乱した代理」問題は、アクションを実行するためのアクセス許可を持たないエンティティが、より権限のあるエンティティにアクションの実行を強制できてしまう場合に生じる、セキュリティ上の問題です。では AWS、サービス間のなりすましにより、混乱した代理問題が発生する可能性があります。サービス間でのなりすましは、あるサービス (呼び出し元サービス) が、別のサービス (呼び出し対象サービス) を呼び出すときに発生する可能性があります。呼び出し元サービスが操作され、それ自身のアクセス許可を使用して、本来アクセス許可が付与されるべきではない方法で別の顧客のリソースに対して働きかけることがあります。これを防ぐため、 AWS では、アカウント内のリソース

へのアクセス許可が付与されたサービスプリンシパルですべてのサービスのデータを保護するために役立つツールを提供しています。

リソースポリシーで aws:SourceArnおよび aws:SourceAccount グローバル条件コンテキストキーを使用して、 がリソースに別のサービス AWS SimSpace Weaver に付与するアクセス許可を制限することをお勧めします。aws:SourceArn 値が、Amazon S3 バケットの Amazon リソースネーム (ARN) などのアカウント ID が含まれていない場合、アクセス許可を制限するため、両方のグローバル条件コンテキストキーを使用する必要があります。同じポリシーステートメントでこれらのグローバル条件コンテキストキーの両方を使用し、アカウント ID にaws:SourceArn の値が含まれていない場合、aws:SourceAccount 値と aws:SourceArn 値の中のアカウントには、同じアカウント ID を使用する必要があります。クロスサービスのアクセスにリソースを 1 つだけ関連付けたい場合は、aws:SourceArn を使用します。クロスサービスが使用できるように、アカウント内の任意のリソースを関連づける場合は、aws:SourceAccount を使用します。

aws:SourceArn の価は、拡張の ARN を使用する必要があります。

混乱した代理問題から保護するための最も効果的な方法は、リソースの完全な ARN を指定して aws:SourceArn グローバル条件コンテキストキーを使用することです。拡張の完全な ARN が不明な場合や、複数の拡張を指定する場合には、ARN の不明な部分にワイルドカード (*) を含む aws:SourceArn グローバルコンテキスト条件キーを使用します。例えば、arn:aws:*simspaceweaver*:*:*111122223333*:*。

次の例は、 で aws:SourceArnおよび aws:SourceAccount グローバル条件コンテキストキーを使用して、混乱した代理問題 SimSpace Weaver を防ぐ方法を示しています。このポリシーは、リクエストが指定されたソースアカウントから送信され、指定された ARN が提供された場合にのみ、 がロールを引き受け SimSpace Weaver ることを許可します。この場合、 SimSpace Weaver はリクエスタ自身のアカウント (*111122223333*) のシミュレーションからのリクエストに対してのみロールを引き受け、指定されたリージョン (*us-west-2)* でのみロールを引き受けることができます。

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

このポリシーの記述のより安全な方法は、以下の例に示すように、aws:SourceArn にシミュレーション名を含めることです。これにより、ポリシーが MyProjectSimulation_22-10-04_22_10_15 という名前のシミュレーションに制限されます。

```
{
  "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/
MyProjectSimulation_22-10-04_22_10_15"
        }
      }
    }
  ]
}
```

aws:SourceArn がアカウント番号を明示的に含める場合は、以下の簡略化されたポリシーのように、aws:SourceAccount の Condition 要素テストを省略できます (詳細については 「IAM ユーザーガイド」を参照してください)。

```
{
  "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/
MyProjectSimulation_22-10-04_22_10_15"
        }
      }
    }
  ]
}
```

AWS SimSpace Weaver ID とアクセスのトラブルシューティング

次の情報は、 および IAM の使用時に発生する可能性がある一般的な問題の診断 SimSpace Weaver と修正に役立ちます。

トピック

- でアクションを実行する権限がありません SimSpace Weaver
- iam:PassRole を実行する権限がない
- アクセスキーを表示したい
- 管理者として、他のユーザーにアクセスを許可したい SimSpace Weaver
- 自分の 以外のユーザーに自分の SimSpace Weaver リソース AWS アカウント へのアクセスを許可したい

でアクションを実行する権限がありません SimSpace Weaver

にアクションを実行する権限がないと AWS Management Console 通知された場合は、管理者に連絡してサポートを依頼する必要があります。管理者とは、ユーザーにユーザー名とパスワードを提供した人です。

以下のエラー例は、`mateojackson` IAM ユーザーがコンソールを使用して架空の *my-example-widget* リソースに関する詳細情報を表示しようとしているが、架空の `simspaceweaver:`*GetWidget* アクセス許可がないという場合に発生します。

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

この場合、Mateo は、`simspaceweaver:`*GetWidget* アクションを使用して *my-example-widget* リソースにアクセスできるように、ポリシーの更新を管理者に依頼します。

iam:PassRole を実行する権限がない

`iam:PassRole` アクションを実行する権限がないというエラーが表示された場合は、ポリシーを更新して SimSpace Weaverにロールを渡すことができるようにする必要があります。

一部の AWS のサービス では、新しいサービスロールまたはサービスにリンクされたロールを作成する代わりに、そのサービスに既存のロールを渡すことができます。そのためには、サービスにロールを渡す権限が必要です。

以下の例のエラーは、`marymajor` という IAM ユーザーがコンソールを使用して SimSpace Weaver でアクションを実行しようする場合に発生します。ただし、このアクションをサービスが実行するには、サービスロールから付与された権限が必要です。メアリーには、ロールをサービスに渡す許可がありません。

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

この場合、Mary のポリシーを更新してメアリーに `iam:PassRole` アクションの実行を許可する必要があります。

サポートが必要な場合は、 AWS 管理者にお問い合わせください。サインイン資格情報を提供した担当者が管理者です。

アクセスキーを表示したい

IAM ユーザーアクセスキーを作成した後は、いつでもアクセスキー ID を表示できます。ただし、シークレットアクセスキーを再表示することはできません。シークレットアクセスキーを紛失した場合は、新しいアクセスキーペアを作成する必要があります。

アクセスキーは、アクセスキー ID (例: `AKIAIOSFODNN7EXAMPLE`) とシークレットアクセスキー (例: `wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY`) の 2 つで構成されています。ユーザー名とパスワードと同様に、リクエストを認証するために、アクセスキー ID とシークレットアクセスキーの両方を使用する必要があります。ユーザー名とパスワードと同様に、アクセスキーは安全に管理してください。

> **Important**
>
> 正規のユーザー ID を確認するためであっても、アクセスキーを第三者に提供しないでください。これにより、 への永続的なアクセス権をユーザーに付与できます AWS アカウント。

アクセスキーペアを作成する場合、アクセスキー ID とシークレットアクセスキーを安全な場所に保存するように求めるプロンプトが表示されます。このシークレットアクセスキーは、作成時にのみ使用できます。シークレットアクセスキーを紛失した場合、IAM ユーザーに新規アクセスキーを追加する必要があります。アクセスキーは最大 2 つまで持つことができます。既に 2 つある場合は、新規キーペアを作成する前に、いずれかを削除する必要があります。手順を表示するには、IAM ユーザーガイドの「アクセスキーの管理」を参照してください。

管理者として、他のユーザーにアクセスを許可したい SimSpace Weaver

他のユーザーにアクセスを許可するには SimSpace Weaver、アクセスを必要とするユーザーまたはアプリケーションにアクセス許可を付与する必要があります。 AWS IAM Identity Center を使用してユーザーとアプリケーションを管理する場合は、アクセスレベルを定義するアクセス許可セットをユーザーまたはグループに割り当てます。アクセス許可セットは、ユーザーまたはアプリケーションに関連付けられている IAM ロールに自動的に IAM ポリシーを作成して割り当てます。詳細については、「AWS IAM Identity Center ユーザーガイド」の「アクセス許可セット」を参照してください。

IAM アイデンティティセンターを使用していない場合は、アクセスを必要としているユーザーまたはアプリケーションの IAM エンティティ (ユーザーまたはロール) を作成する必要があります。次に、 SimSpace Weaverの適切なアクセス許可を付与するポリシーを、そのエンティティにアタッチする必要があります。アクセス許可が付与されたら、ユーザーまたはアプリケーション開発者に認証情報を提供します。これらの認証情報を使用して AWSにアクセスします。IAM ユーザー、グルー

プ、ポリシー、アクセス許可の作成の詳細については、「IAM ユーザーガイド」の「IAM アイデンティティ」と「IAM のポリシーとアクセス許可」を参照してください。

自分の 以外のユーザーに自分の SimSpace Weaver リソース AWS アカウント へのアクセスを許可したい

他のアカウントのユーザーや組織外の人が、リソースにアクセスするために使用できるロールを作成できます。ロールの引き受けを委託するユーザーを指定できます。リソースベースのポリシーまたはアクセスコントロールリスト (ACL) をサポートするサービスの場合、それらのポリシーを使用して、リソースへのアクセスを付与できます。

詳細については、以下を参照してください:

- がこれらの機能 SimSpace Weaver をサポートしているかどうかを確認するには、「」を参照してくださいが IAM と AWS SimSpace Weaver 連携する方法。
- 所有 AWS アカウント している のリソースへのアクセスを提供する方法については、IAM ユーザーガイドの「所有 AWS アカウント している別の の IAM ユーザーへのアクセスを提供する」を参照してください。
- リソースへのアクセスをサードパーティーに提供する方法については AWS アカウント、IAM ユーザーガイドの「サードパーティー AWS アカウント が所有する へのアクセスを提供する」を参照してください。
- ID フェデレーションを介してアクセスを提供する方法については、「IAM ユーザーガイド」の「外部で認証されたユーザー (ID フェデレーション) へのアクセスの許可」を参照してください。
- クロスアカウントアクセスにおけるロールとリソースベースのポリシーの使用方法の違いについては、「IAM ユーザーガイド」の「IAM でのクロスアカウントのリソースへのアクセス」を参照してください。

のセキュリティイベントのログ記録とモニタリング AWS SimSpace Weaver

モニタリングは、 および SimSpace Weaver AWS ソリューションの信頼性、可用性、パフォーマンスを維持する上で重要な部分です。マルチポイント障害が発生した場合は、その障害をより簡単にデバッグできるように、 AWS ソリューションのすべての部分からモニタリングデータを収集する必要があります。

AWS と SimSpace Weaver には、シミュレーションリソースをモニタリングし、潜在的なインシデントに対応するためのツールがいくつか用意されています。

Amazon CloudWatch のログ

SimSpace Weaver は CloudWatch にログを保存します。これらのログは、シミュレーション内のイベント (アプリケーションの起動や停止など) のモニタリングやデバッグに使用できます。詳細については、「SimSpace Weaver Amazon CloudWatch Logs の ログ」を参照してください。

Amazon CloudWatch アラーム

Amazon CloudWatch アラームを使用して、指定した期間中、単一のメトリクスをモニタリングします。メトリクスが特定のしきい値を超えると、Amazon SNS トピックまたは AWS Auto Scaling ポリシーに通知が送信されます。CloudWatch アラームは、状態が変化したときにトリガーされ、特定の状態になるまでではなく、指定した期間だけ維持されます。詳細については、「Amazon CloudWatch SimSpace Weaver によるモニタリング」を参照してください。

AWS CloudTrail ログ

CloudTrail は、ユーザー、ロール、または AWS のサービスによって実行されたアクションの記録を提供します SimSpace Weaver。CloudTrail によって収集された情報を使用して、リクエストの実行元の IP アドレス SimSpace Weaver、リクエストの実行者、リクエストの実行日時などの詳細を確認できます。詳細については、「を使用した AWS SimSpace Weaver API コールのログ記録 AWS CloudTrail」を参照してください。

のコンプライアンス検証 AWS SimSpace Weaver

SimSpace Weaver は AWS コンプライアンスプログラムの対象ではありません。

サードパーティーの監査者は、複数のコンプライアンスプログラムの一環として、他の AWS サービスのセキュリティと AWS コンプライアンスを評価します。これらのプログラムには、SOC、PCI、FedRAMP、HIPAA などがあります。

AWS のサービス が特定のコンプライアンスプログラムの範囲内にあるかどうかを確認するには、AWS のサービス 「コンプライアンスプログラムによる範囲内」を参照して、関心のあるコンプライアンスプログラムを選択します。一般的な情報については、AWS 「 Compliance ProgramsAssurance」を参照してください。

を使用して、サードパーティーの監査レポートをダウンロードできます AWS Artifact。詳細については、「Downloading Reports in AWS Artifact」を参照してください。

を使用する際のお客様のコンプライアンス責任 AWS のサービス は、お客様のデータの機密性、貴社のコンプライアンス目的、適用される法律および規制によって決まります。 は、コンプライアンスに役立つ以下のリソース AWS を提供します。

- セキュリティのコンプライアンスとガバナンス – これらのソリューション実装ガイドでは、アーキテクチャ上の考慮事項について説明し、セキュリティとコンプライアンスの機能をデプロイする手順を示します。
- HIPAA 対応サービスのリファレンス – HIPAA 対応サービスの一覧が提供されています。すべてが HIPAA AWS のサービス の対象となるわけではありません。
- AWS コンプライアンスリソース – このワークブックとガイドのコレクションは、お客様の業界や地域に適用される場合があります。
- AWS カスタマーコンプライアンスガイド – コンプライアンスの観点から責任共有モデルを理解します。このガイドでは、複数のフレームワーク (米国国立標準技術研究所 (NIST)、Payment Card Industry Security Standards Council (PCI)、国際標準化機構 (ISO) など) にわたるセキュリティコントロールを保護および AWS のサービス マッピングするためのベストプラクティスをまとめています。
- 「 デベロッパーガイド」の「ルールによるリソースの評価」 – この AWS Config サービスは、リソース設定が内部プラクティス、業界ガイドライン、および規制にどの程度準拠しているかを評価します。 AWS Config
- AWS Security Hub – これにより AWS のサービス 、 内のセキュリティ状態を包括的に把握できます AWS。Security Hub では、セキュリティコントロールを使用して AWS リソースを評価し、セキュリティ業界標準とベストプラクティスに対するコンプライアンスをチェックします。サポートされているサービスとコントロールの一覧については、Security Hub のコントロールリファレンスを参照してください。
- Amazon GuardDuty – 不審なアクティビティや悪意のあるアクティビティがないか環境をモニタリングすることで AWS アカウント、、ワークロード、コンテナ、データに対する潜在的な脅威 AWS のサービス を検出します。GuardDuty を使用すると、特定のコンプライアンスフレームワークで義務付けられている侵入検知要件を満たすことで、PCI DSS などのさまざまなコンプライアンス要件に対応できます。
- AWS Audit Manager – これにより AWS のサービス 、 AWS 使用状況を継続的に監査し、リスクの管理方法と規制や業界標準への準拠を簡素化できます。

の耐障害性 AWS SimSpace Weaver

AWS グローバルインフラストラクチャは、 AWS リージョン およびアベイラビリティーゾーンを中心に構築されています。 は、低レイテンシー、高スループット、高度に冗長なネットワークで接続された、物理的に分離および分離された複数のアベイラビリティーゾーン AWS リージョン を提供します。アベイラビリティーゾーンでは、ゾーン間で中断することなく自動的にフェイルオーバーす

るアプリケーションとデータベースを設計および運用することができます。アベイラビリティーゾーンは、従来の単一または複数のデータセンターインフラストラクチャよりも可用性が高く、フォールトトレラントで、スケーラブルです。

AWS リージョン およびアベイラビリティーゾーンの詳細については、AWS 「 グローバルインフラストラクチャ」を参照してください。

のインフラストラクチャセキュリティ AWS SimSpace Weaver

マネージドサービスである AWS SimSpace Weaver は、 AWS グローバルネットワークセキュリティで保護されています。 AWS セキュリティサービスと がインフラストラクチャ AWS を保護する方法については、AWS 「 クラウドセキュリティ」を参照してください。インフラストラクチャセキュリティのベストプラクティスを使用して AWS 環境を設計するには、「Security Pillar AWS Well-Architected Framework」の「Infrastructure Protection」を参照してください。

AWS が発行した API コールを使用して、ネットワーク SimSpace Weaver 経由で にアクセスします。クライアントは以下をサポートする必要があります。

- Transport Layer Security (TLS)。TLS 1.2 が必須で、TLS 1.3 をお勧めします。
- DHE (楕円ディフィー・ ヘルマン鍵共有) や ECDHE (楕円曲線ディフィー・ ヘルマン鍵共有) などの完全前方秘匿性 (PFS) による暗号スイート。これらのモードはJava 7 以降など、ほとんどの最新システムでサポートされています。

また、リクエストにはアクセスキー ID と、IAM プリンシパルに関連付けられているシークレットアクセスキーを使用して署名する必要があります。または、AWS Security Token Service (AWS STS) を使用して、一時的なセキュリティ認証情報を生成し、リクエストに署名することもできます。

ネットワーク接続セキュリティモデル

シミュレーションは、選択した AWS リージョン内にある Amazon VPC 内のコンピューティングインスタンスで実行されます。Amazon VPC は AWS クラウドの仮想ネットワークであり、ワークロードまたは組織エンティティごとにインフラストラクチャを分離します。Amazon VPC 内のコンピューティングインスタンス間の通信は AWS ネットワーク内にとどまり、インターネットを経由することはありません。一部の内部サービス通信はインターネットを経由し、暗号化されます。同じ AWS リージョンで実行されているすべてのお客様のシミュレーションは、同じ Amazon VPC を共有します。異なるお客様のシミュレーションは、同じ Amazon VPC 内の別々のコンピューティングインスタンスを使用します。

シミュレーションクライアントと で実行されているシミュレーション間の通信は、インターネット経由で SimSpace Weaver 転送されます。 SimSpace Weaver はこれらの接続を処理しません。クライアント接続を保護するのはお客様の責任です。

SimSpace Weaver サービスへの接続はインターネットを経由し、暗号化されます。これには、、AWS Command Line Interface（AWS CLI）AWS Management Console、 AWS ソフトウェア開発キット (SDK)、および SimSpace Weaver アプリケーション SDK を使用した接続が含まれます。

での設定と脆弱性の分析 AWS SimSpace Weaver

設定と IT コントロールは、 AWS とユーザーの間で責任を共有します。詳細については、 AWS「 責任共有モデル」を参照してください。 は、コンピューティングインスタンスでのオペレーティングシステムのパッチ適用、ファイアウォール設定、インフラストラクチャディザスタリカバリなど、基盤となる AWS インフラストラクチャの基本的なセキュリティタスク AWS を処理します。これらの手順は適切なサードパーティーによって確認され、認証されています。詳細については、「セキュリティ、アイデンティティ、コンプライアンスのベストプラクティス」を参照してください。

以下のシミュレーションソフトウェアのセキュリティに関する責任はユーザーにあります。

- アップデートやセキュリティパッチを含むアプリケーションコードを管理します。
- シミュレーションクライアントと接続先のアプリケーション間の通信を認証して暗号化します。
- SDK や SimSpace Weaver アプリケーション SDK などの最新の AWS SDK バージョンを使用するようにシミュレーションを更新します。

> **Note**
>
> SimSpace Weaver は、実行中のシミュレーションでのアプリケーションの更新をサポートしていません。アプリケーションを更新する必要がある場合、シミュレーションを停止して削除し、更新されたアプリケーションコードを使用して新しいシミュレーションを作成する必要があります。シミュレーションを再作成する必要がある場合に復元できるように、シミュレーションの状態を外部データストアに保存することをお勧めします。

のセキュリティのベストプラクティス SimSpace Weaver

このセクションでは、 固有のセキュリティのベストプラクティスについて説明します SimSpace Weaver。のセキュリティのベストプラクティスの詳細については AWS、「セキュリティ、アイデンティティ、コンプライアンスのベストプラクティス」を参照してください。

トピック

- アプリケーションとクライアント間の通信を暗号化します。
- シミュレーション状態の定期的なバックアップ
- アプリケーションと SDK の維持

アプリケーションとクライアント間の通信を暗号化します。

SimSpace Weaver は、アプリケーションとそのクライアント間の通信を管理しません。クライアントセッションには何らかの認証と暗号化を実装する必要があります。

シミュレーション状態の定期的なバックアップ

SimSpace Weaver はシミュレーション状態を保存しません。(API コール、コンソールオプション、またはシステムクラッシュの結果として) 停止したシミュレーション状態は保存されず、元の状態に戻す方法もありません。停止したシミュレーションは再開できません。再起動と同等の処理を実行する唯一の方法は、同じ設定とデータを使用してシミュレーションを再作成することです。シミュレーション状態のバックアップを使用して、新しいシミュレーションを初期化できます。 AWS には、シミュレーション状態を保存できる、信頼性が高く利用可能なクラウドストレージとデータベースサービスがあります。

アプリケーションと SDK の維持

アプリケーション、 AWS Software Development Kit (SDKs) のローカルインストール、および SimSpace Weaver アプリケーション SDK を維持します。新しいバージョンの AWS SDKs をダウンロードしてインストールできます。非本番稼働 SimSpace Weaver 用アプリケーションビルドを使用して App SDK の新しいバージョンをテストし、アプリケーションが期待どおりに動作していることを確認します。実行中のシミュレーションではアプリケーションを更新できません。アプリケーションの更新方法:

1. アプリケーションコードをローカル (またはテスト環境) で更新してテストします。
2. シミュレーション状態の変更を停止して保存します (必要な場合)。

3. シミュレーションを停止します (一度停止すると、再開することはできません)。
4. シミュレーションを削除します (停止したシミュレーションのうち、削除されていないものはサービス制限にカウントされます)。
5. 同じ設定と更新したアプリケーションコードでシミュレーションを再作成します。
6. 保存した状態データを使用してシミュレーションを初期化します (使用可能な場合)。
7. 新しいシミュレーションを開始します。

> **Note**
>
> 同じ設定で作成された新しいシミュレーションは、古いシミュレーションとは別のものです。これには新しいシミュレーション ID が割り当てられ、Amazon CloudWatch の新しいログストリームにログが送信されます。

でのログ記録とモニタリング SimSpace Weaver

モニタリングは、 およびその他の SimSpace Weaver AWS ソリューションの信頼性、可用性、パフォーマンスを維持する上で重要な部分です。 AWS には、監視 SimSpace Weaver、問題発生時の報告、必要に応じて自動アクションを実行するための以下のモニタリングツールが用意されています。

- Amazon CloudWatch は、 AWS リソースと で実行されるアプリケーションを AWS リアルタイムでモニタリングします。メトリクスを収集および追跡し、カスタマイズされたダッシュボードを作成し、指定されたメトリックが指定したしきい値に達したときに通知またはアクションを実行するアラームを設定できます。詳細については、『Amazon CloudWatch ユーザーガイド』を参照してください。
- Amazon CloudWatch Logs を使用すると、 SimSpace Weaver ワーカー、CloudTrail、およびその他のソースからのログデータをモニタリング、保存、およびアクセスできます。CloudWatch Logs は、ログデータ内の情報をモニタリングし、特定のしきい値に達すると通知できます。高い耐久性を備えたストレージにログデータをアーカイブすることもできます。詳細については、『Amazon CloudWatch Logs ユーザーガイド』を参照してください。
- AWS CloudTrailは、 AWS アカウント により、またはそのアカウントに代わって行われた API コールや関連イベントを取得し、指定した Amazon S3 バケットにログファイルを配信します。AWSを呼び出したユーザーとアカウント、呼び出し元の IP アドレス、および呼び出しの発生日時を特定できます。詳細については、「AWS CloudTrail ユーザーガイド」を参照してください。

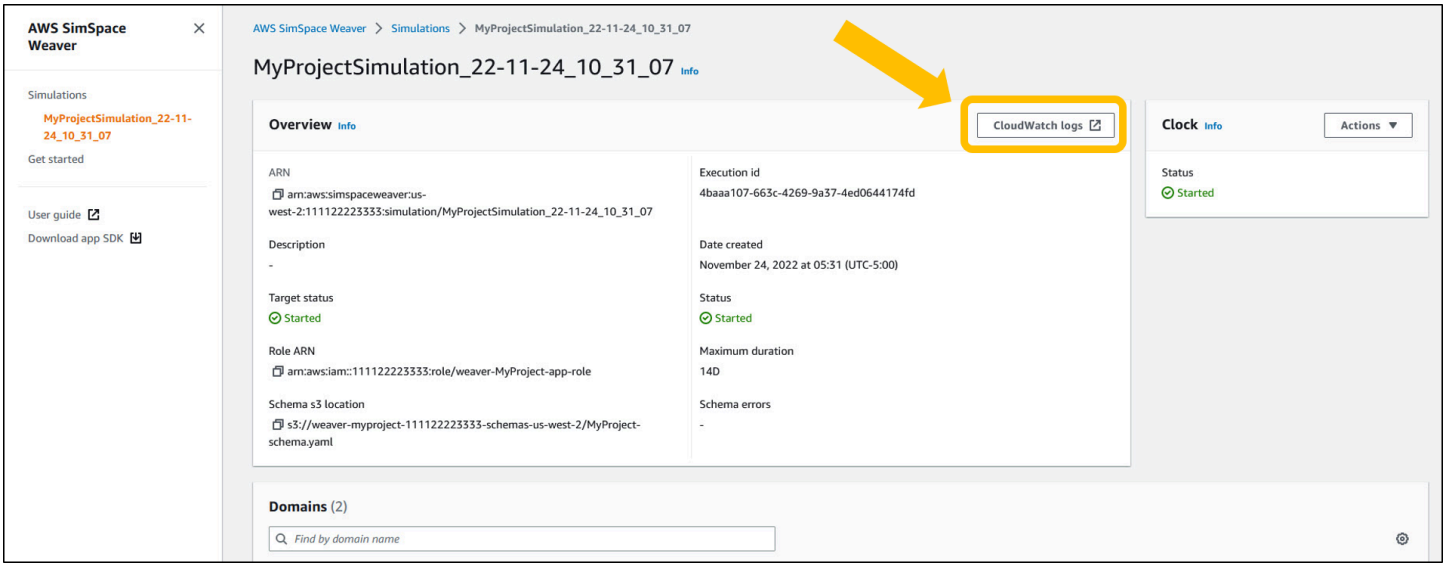
トピック

- SimSpace Weaver Amazon CloudWatch Logs の ログ
- Amazon CloudWatch SimSpace Weaver によるモニタリング
- を使用した AWS SimSpace Weaver API コールのログ記録 AWS CloudTrail

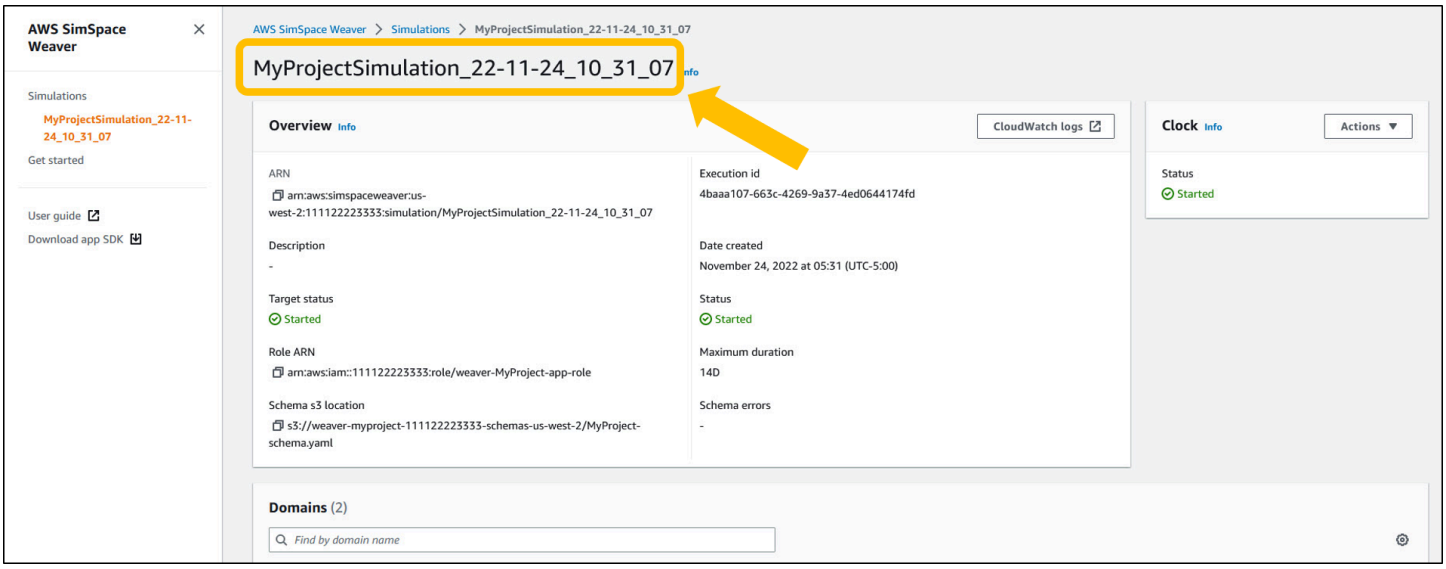
SimSpace Weaver Amazon CloudWatch Logs の ログ

SimSpace Weaver ログのアクセス

SimSpace Weaver シミュレーションから生成されたすべてのログは、Amazon CloudWatch Logs に保存されます。ログにアクセスするには、[SimSpace Weaver] コンソールのシミュレーションの [概要] ペインにある [CloudWatch logs] ボタンを使用できます。このボタンをクリックすると、その特定のシミュレーションのログに直接移動できます。



CloudWatch コンソールからログにアクセスすることもできます。ログを検索するには、シミュレーションの名前が必要です。



SimSpace Weaver ログ

SimSpace Weaver は、シミュレーション管理メッセージとコンソール出力をアプリから Amazon CloudWatch Logs に書き込みます。ログの使用の詳細については、「Amazon CloudWatch Logs ユーザーガイド」の「ロググループとログストリームの使用」を参照してください。

CloudWatch Logs には、作成した各シミュレーションに独自のロググループがあります。ロググループの名前は、シミュレーションスキーマで指定されます。以下のスキーマスニペットでは、`log_destination_service` の値は `logs` です。つま

り、log_destination_resource_name の値はロググループの名前です。この場合、ロググループは MySimulationLogs です。

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

DescribeSimulation API を使用して、シミュレーションを開始した後でシミュレーション用のロググループの名前を検索することもできます。

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

以下の例は、ロギング設定を説明する、DescribeSimulation から出力の一部を示しています。ロググループの名前は LogGroupArn の末尾に表示されます。

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

各シミュレーションロググループには、いくつかのログストリームが含まれます。

- 管理ログストリーム – SimSpace Weaver サービスによって生成されたシミュレーション管理メッセージ。

  ```
  /sim/management
  ```

- エラーログストリーム – SimSpace Weaver サービスによって生成されたエラーメッセージ。このログストリームは、エラーがある場合にのみ存在します。 は、アプリによって書き込まれたエラーを独自のアプリログストリームに SimSpace Weaver 保存します (次のログストリームを参照) 。

```
/sim/errors
```

- 空間アプリケーションログストリーム (各ワーカーの空間アプリケーションごとに 1 つ) — 空間アプリケーションによって生成されるコンソール出力。各空間アプリケーションは、独自のログストリームに書き込みます。*spatial-app-id* は、*worker-id* の末尾にあるスラッシュの後のすべての文字です。

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

- カスタムアプリケーションログストリーム (カスタムアプリケーションインスタンスごとに 1 つ) — カスタムアプリケーションによって生成されるコンソール出力。各カスタムアプリケーションインスタンスは、独自のログストリームに書き込みます。

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

- サービスアプリケーションログストリーム (サービスアプリケーションインスタンスごとに 1 つ) — サービスアプリケーションによって生成されるコンソール出力。各サービスアプリケーションは、独自のログストリームに書き込みます。*service-app-id* は、*service-app-name* の末尾にあるスラッシュの後のすべての文字です。

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

Amazon CloudWatch SimSpace Weaver によるモニタリング

Amazon CloudWatch SimSpace Weaver を使用してAmazon CloudWatch は raw データを収集し、読み取り可能なほぼリアルタイムのメトリクスに加工します。これらの統計は 15 か月間保持されるため、履歴情報にアクセスし、ウェブアプリケーションまたはサービスの動作をより的確に把握できます。また、特定のしきい値を監視するアラームを設定し、これらのしきい値に達したときに通知を送信したりアクションを実行したりできます。詳細については、「Amazon CloudWatch ユーザーガイド」を参照してください。

SimSpace Weaver サービスは、 `AWS/simspaceweaver`名前空間で次のメトリクスを報告します。

SimSpace Weaver アカウントレベルの メトリクス

SimSpace Weaver 名前空間には、 AWS アカウントレベルでのアクティビティに関連する以下のメトリクスが含まれます。

メトリクス	説明
`SimulationCount`	現在のアカウントのシミュレーションの数。 単位: カウント 次元: なし 統計: Average、Minimum、Maximum

を使用した AWS SimSpace Weaver API コールのログ記録 AWS CloudTrail

AWS SimSpace Weaver は、ユーザー AWS CloudTrail、ロール、または のサービスによって実行されたアクションを記録する AWS サービスである と統合されています SimSpace Weaver。CloudTrail は、 SimSpace Weaver のすべての API コールをイベントとしてキャプチャします。キャプチャされた呼び出しには、 SimSpace Weaver コンソールからの呼び出しと SimSpace Weaver API オペレーションへのコード呼び出しが含まれます。証跡を作成する場合は、イベントを含む Amazon S3 バケットへの CloudTrail イベントの継続的な配信を有効にすることができます SimSpace Weaver。証跡を設定しない場合でも、Event history の [CloudTrail] コンソールで最新のイベントを表示できます。CloudTrail によって収集された情報を使用して、リクエストの実行元の IP アドレス SimSpace Weaver、リクエストの実行者、リクエストの実行日時などの詳細を確認できます。

CloudTrail の詳細については、「AWS CloudTrail ユーザーガイド」を参照してください。

SimSpace Weaver CloudTrail の情報

CloudTrail は、アカウントの作成 AWS アカウント 時に で有効になります。アクティビティが発生すると SimSpace Weaver、そのアクティビティは の他の AWS サービスイベントとともに CloudTrail イベントに記録されますEvent history。で最近のイベントを表示、検索、ダウンロードできます AWS アカウント。詳細については、「CloudTrail イベント履歴でのイベントの表示」を参照してください。

のイベントを含む AWS アカウントのイベントの継続的な記録については SimSpace Weaver、証跡を作成します。証跡により、CloudTrail はログファイルを Amazon S3 バケットに配信できます。デフォルトでは、コンソールで証跡を作成するときに、証跡がすべての AWS リージョンに適用されます。証跡は、 AWS パーティション内のすべてのリージョンからのイベントをログに記録し、指定した Amazon S3 バケットにログファイルを配信します。さらに、CloudTrail ログで収集されたイベントデータをさらに分析して処理するように他の AWS サービスを設定できます。詳細については、次を参照してください:

- 追跡を作成するための概要
- 「CloudTrail がサポートされているサービスと統合」
- 「CloudTrail の Amazon SNS 通知の設定」
- 複数のリージョンから CloudTrail ログファイルを受け取るおよび複数のアカウントから CloudTrail ログファイルを受け取る

すべての SimSpace Weaver アクションは CloudTrail によってログに記録され、 AWS SimSpace Weaver API リファレンスに記載されています。例えば、`ListSimulations`、`DescribeSimulation`、`DeleteSimulation` の各アクションを呼び出すと、CloudTrail ログファイルにエントリが生成されます。

各イベントまたはログエントリには、誰がリクエストを生成したかという情報が含まれます。アイデンティティ情報は、以下を判別するのに役立ちます。

- リクエストがルートまたは AWS Identity and Access Management (IAM) ユーザー認証情報を使用して行われたかどうか。
- リクエストがロールまたはフェデレーションユーザーのテンポラリなセキュリティ認証情報を使用して行われたかどうか。
- リクエストが別の AWS サービスによって行われたかどうか。

詳細については、「CloudTrail userIdentity エレメント」を参照してください。

SimSpace Weaver ログファイルエントリについて

「トレイル」は、指定した Amazon S3 バケットにイベントをログファイルとして配信するように設定できます。CloudTrail のログファイルには、ログエントリが 1 つ以上あります。イベントは任意のソースからの単一のリクエストを表し、アクションの日時、リクエストのパラメータ、その他の詳細といったリクエストされたアクションに関する情報が含まれます。CloudTrail ログファイルは、パブ

リック API コールの順序付けられたスタックトレースではないため、特定の順序では表示されません。

以下の例は、`ListSimulations` アクションを示す CloudTrail ログエントリです。

```
{
    "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 エンドポイントとクォータ

以下の表で、SimSpace Weaver のサービスエンドポイントおよび Service Quotas を説明します。Service Quotas (制限とも呼ばれます) は、 AWS アカウントのサービスリソースまたはオペレーションの最大数です。詳細については、「AWS 全般のリファレンス」の「AWS Service Quotas」を参照してください。

サービスエンドポイント

リージョン名	リージョン	エンドポイント	プロトコル
米国東部 (バージニア北部)	us-east-1	simspaceweaver.us-east-1.amazonaws.com	HTTPS
米国東部 (オハイオ)	us-east-2	simspaceweaver.us-east-2.amazonaws.com	HTTPS
米国西部 (オレゴン)	us-west-2	simspaceweaver.us-west-2.amazonaws.com	HTTPS
アジアパシフィック (シンガポール)	ap-southeast-1	simspaceweaver.ap-southeast-1.amazonaws.com	HTTPS
アジアパシフィック (シドニー)	ap-southeast-2	simspaceweaver.ap-southeast-2.amazonaws.com	HTTPS
欧州 (ストックホルム)	eu-north-1	simspaceweaver.eu-north-1.amazonaws.com	HTTPS

リージョン名	リージョン	エンドポイント	プロトコル
欧州 (フランクフルト)	eu-central-1	simspaceweaver.eu-central-1.amazonaws.com	HTTPS
欧州 (アイルランド)	eu-west-1	simspaceweaver.eu-west-1.amazonaws.com	HTTPS
AWS GovCloud (米国東部)	us-gov-east-1	simspaceweaver.us-gov-east-1.amazonaws.com	HTTPS
AWS GovCloud (米国西部)	us-gov-west-1	simspaceweaver.us-gov-west-1.amazonaws.com	HTTPS

Service Quotas

名前	デフォルト	引き上げ可能	説明
各アプリケーションのコンピュートリソースユニット	サポートされている各リージョン: 4	いいえ	各アプリケーションに割り当てることができるコンピュートリソースユニットの最大数。
各ワーカーのコンピュートリソースユニット	サポートされている各リージョン: 17	いいえ	各ワーカーが利用できるコンピュートリソースユニットの数。

名前	デフォルト	引き上げ可能	説明
各エンティティのデータフィールド	サポートされている各リージョン: 7	いいえ	エンティティが保持できるデータ (非インデックス) フィールドの最大数。
パーティション内のエンティティ	サポートされている各リージョン: 8,192	いいえ	1 パーティションのエンティティの最大数。
エンティティのデータフィールドサイズ	サポートされている各リージョン: 1,024 バイト	いいえ	エンティティのデータ (非インデックス) フィールドの最大サイズ。
ワーカー間のエンティティ転送	サポートされている各リージョン: 25	いいえ	各パーティションおよび各ティックにおける、ワーカー間のエンティティ転送の最大数。
同じワーカーでのエンティティ転送	サポートされている各リージョン: 500	いいえ	同じワーカーでのパーティションごと、およびティックごとのエンティティ転送の最大数。
各エンティティのインデックスフィールド	サポートされている各リージョン: 1	いいえ	エンティティが保持できるインデックスフィールドの最大数。

名前	デフォルト	引き上げ可能	説明
シミュレーションの最大実行期間の最大日数	サポートされている各リージョン: 14	いいえ	シミュレーションの最大実行期間として指定できる最大日数。値を指定しなくても、すべてのシミュレーションには最大実行期間が設定されます。シミュレーションは、最大実行期間に達すると自動的に停止します。
各コンピュートリソースユニットのメモリ	サポートされている各リージョン: 1 GB	いいえ	コンピュートリソースユニットごとにアプリケーションが取得するランダムアクセスメモリ (RAM) の量。
各ワーカーのリモートサブスクリプション	サポートされている各リージョン: 24	いいえ	各ワーカーのリモートサブスクリプションの最大数。
シミュレーションカウント	サポートされている各リージョン: 2	可能	アカウントでターゲットステータスが STARTED のシミュレーションの最大数。最大 10 までクォータ引き上げをリクエストできます。

名前	デフォルト	引き上げ可能	説明
シミュレーション用のワーカー	サポートされている各リージョン: 2	可能	1 シミュレーションに割り当てることができるワーカーの最大数。最大 10 までクォータ引き上げをリクエストできます。
各コンピュートリソースユニットの vCPU	サポートされている各リージョン: 2	いいえ	アプリケーションが各コンピュートリソースユニットで取得する仮想化中央演算装置 (vCPU) の数。

メッセージングクォータ

以下のクォータは、 の SimSpace Weaver Localおよび のアプリケーション間メッセージングに適用されます AWS クラウド。

名前	デフォルト	引き上げ可能	説明
最大メッセージサイズ (MB)	サポートされている各リージョン: 256 バイト	いいえ	メッセージペイロードの最大サイズ。
最大メッセージ送信レート	サポートされている各リージョン: 128	いいえ	各アプリがティックごとに送信できるメッセージの最大数。

クロックレート

シミュレーションスキーマは、シミュレーションのクロックレート (ティックレートとも呼ばれます) を指定します。以下の表は、使用できる有効なクロックレートを指定しています。

名前	有効値	説明
クロックレート	サポートされている各リージョン：「10」、「15」、「30」、「無制限」	シミュレーションの有効なクロックレート。
クロックレート (バージョン 1.13 および 1.12)	サポートされている各リージョン: 10、15、30	シミュレーションの有効なクロックレート。

SimSpace Weaver Local の Service Quotas

以下の Service Quotas は SimSpace Weaver Local にのみ適用されます。他のすべてのクォータも SimSpace Weaver Local に適用されます。

名前	デフォルト	引き上げ可能	説明
最大パーティション数	SimSpace Weaver Local: 24	いいえ	シミュレーションのパーティションの最大数。
最大アプリケーション数	SimSpace Weaver Local: 24	いいえ	シミュレーションの (種類を問わず) アプリケーションの最大合計数。
最大ドメイン数	SimSpace Weaver Local: 24	いいえ	シミュレーションの (種類を問わず) ドメインの最大合計数。
パーティション内のエンティティ数	SimSpace Weaver Local: 4,096	いいえ	各パーティションのエンティティの最大数。

名前	デフォルト	引き上げ可能	説明
エンティティあたりのフィールド数	SimSpace Weaver Local: 8	いいえ	各エンティティのフィールドの最大数。
フィールドサイズ	SimSpace Weaver Local: 1024 バイト	いいえ	エンティティフィールドの最大サイズ。

でのトラブルシューティング SimSpace Weaver

トピック

- AssumeRoleAccessDenied
- InvalidBucketName
- ServiceQuotaExceededException
- TooManyBuckets
- シミュレーション開始中にアクセスが拒否される
- Docker を使用時の時間に関する問題
- PathfindingSample コンソールクライアントが接続に失敗する
- AWS CLI が認識しない simspaceweaver

AssumeRoleAccessDenied

シミュレーションが開始されない場合、以下のエラーが表示される場合があります。

```
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
```

このエラーは、シミュレーションの AWS Identity and Access Management (IAM) ロールについて次のいずれかに当てはまる場合に表示されることがあります。

- Amazon リソースネーム (ARN) が存在しない IAM ロールを指している。
- IAM ロールの信頼ポリシーが、新しいシミュレーション名がロールを引き受けることを許可しない。

ロールが存在することを確認します。ロールが存在する場合は、ロールの信頼ポリシーを確認します。以下の信頼ポリシーの例では、`aws:SourceArn` は名前が `MySimulation` で始まる (アカウント 111122223333 の) シミュレーションのみがロールを引き継ぐことを許可しています。

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

名前が MyOtherSimulation で始まる別のシミュレーションがロールを引き継ぐことを許可するには、以下の編集例のように信頼ポリシーを変更する必要があります。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "simspaceweaver.amazonaws.com"
            },
            "Action": "sts:AssumeRole",
            "Condition": {
                "ArnLike": {
                    "aws:SourceArn": [
                        aws:SourceArn": "arn:aws:simspaceweaver:us-
west-2:111122223333:simulation/MySimulation*",
                        aws:SourceArn": "arn:aws:simspaceweaver:us-
west-2:111122223333:simulation/MyOtherSimulation*"
                        ]
                }
            }
        }
    ]
}
```

InvalidBucketName

プロジェクトの作成中に、以下のエラーが表示される場合があります。

```
An error occurred (InvalidBucketName) when calling the CreateBucket operation: The
 specified bucket is not valid.
```

このエラーは、Amazon Simple Storage Service (Amazon S3) に渡 SimSpace Weaver された名前がバケットの命名規則に違反していたために発生しました (詳細については、Amazon Simple Storage Service ユーザーガイドの「バケットの命名規則」を参照してください) 。

SimSpace Weaver アプリケーション SDK の`create-project`スクリプトは、スクリプトに指定したプロジェクト名を使用してバケット名を作成します。バケット名は、以下のフォーマットを使用します。

- バージョン 1.13.x 以降
  - `weaver-`*`lowercase-project-name`*`-`*`account-number`*`-`*`region`*
- バージョン 1.12.x
  - `weaver-`*`lowercase-project-name`*`-`*`account-number`*`-app-zips-`*`region`*
  - `weaver-`*`lowercase-project-name`*`-`*`account-number`*`-schemas-`*`region`*

例えば、以下のプロジェクトプロパティがあるとします。

- プロジェクト名: `MyProject`
- AWS アカウント 番号: `111122223333`
- AWS リージョン: `us-west-2`

プロジェクトには以下のバケットがあります。

- バージョン 1.13.x 以降
  - `weaver-myproject-111122223333-us-west-2`
- バージョン 1.12.x
  - `weaver-myproject-111122223333-app-zips-us-west-2`
  - `weaver-myproject-111122223333-schemas-us-west-2`

プロジェクト名は Amazon S3 の命名規則に違反していない必要があります。また、`create-project` スクリプトによって作成されたバケット名は Amazon S3 バケットの名前の長さ制限を超えない長さのプロジェクト名を使用する必要があります。

ServiceQuotaExceededException

シミュレーションを開始すると、以下のエラーが表示されることがあります。

```
An error occurred (ServiceQuotaExceededException) when calling the StartSimulation
 operation: Failed to start simulation due to: simulation quota has already been
 reached.
```

このエラーは、新しいシミュレーションを開始しようとしたときに、アカウントのターゲットのステータスが STARTED のシミュレーションの数が最大数に達している場合に表示されます。これには、実行中のシミュレーション、失敗したシミュレーション、および最大時間に達したために停止したシミュレーションが含まれます。停止または失敗したシミュレーションを削除して、新しいシミュレーションを開始できます。すべてのシミュレーションが実行中の場合は、実行中のシミュレーションを停止して削除できます。まだリクエストの上限に達していない場合は、Service Quotas の引き上げをリクエストすることもできます。

TooManyBuckets

プロジェクトの作成中に、以下のエラーが表示される場合があります。

```
An error occurred (TooManyBuckets) when calling the CreateBucket operation: You have
 attempted to create more buckets than allowed.
```

Amazon Simple Storage Service (Amazon S3) には、 AWS アカウントに含めることができるバケットの数に制限があります (詳細については、「Amazon Simple Storage Service ユーザーガイド」の「バケットの制限と制限」を参照してください) 。

続ける前に、以下のいずれかを実行する必要があります。

- 不要な 2 つ以上の既存の Amazon S3 バケットを削除します。
- Amazon S3 の制限の引き上げをリクエストします (詳細については、「Amazon Simple Storage Service ユーザーガイド」の「バケットの制約および制限」を参照してください)。
- 別の AWS アカウントを使用します。

> **Note**
>
> の `DeleteSimulation` API SimSpace Weaver は、シミュレーションに関連付けられた Amazon S3 リソースを削除しません。不要になった場合は、シミュレーションに関連するすべてのリソースを削除することをお勧めします。

シミュレーション開始中にアクセスが拒否される

シミュレーションを開始すると、アクセスが拒否されたこと、またはアプリケーションのアーティファクトへのアクセス中にエラーが発生したことを示すエラーメッセージが表示される場合があります。この問題は、 (コンソールまたは SimSpace Weaver アプリ SDK スクリプトを介して) 作成されなかったシミュレーションの Amazon S3 バケットを指定した場合に発生する可能性があります。SimSpace Weaver

根本原因として最も考えられるのは以下の状況です。

- このサービスには、シミュレーションスキーマで指定した 1 つ以上の Amazon S3 バケットにアクセスする権限がない - アプリケーションロールのアクセス許可ポリシー、Amazon S3 バケットポリシー、および Amazon S3 バケットアクセス許可を確認して、`simspaceweaver.amazonaws.com` がバケットにアクセスするための正しい権限があることを確認します。アプリケーションロールアクセス許可ポリシーの詳細については、「がユーザーに代わって SimSpace Weaver 作成するアクセス許可」を参照してください。
- Amazon S3 バケットはシミュレーションとは異なる AWS リージョン にある可能性があります – シミュレーションアーティファクトの Amazon S3 バケットは、シミュレーション AWS リージョン と同じ にある必要があります。Amazon S3 コンソールで、 AWS リージョン バケットの内容を確認します。Amazon S3 バケットが別の にある場合は AWS リージョン、シミュレーション AWS リージョン と同じ にあるバケットを選択します。

Docker を使用時の時間に関する問題

を使用していてDocker、 SimSpace Weaver アプリケーション SDK からスクリプトを実行中に時間関連のエラーが発生した場合、Docker仮想マシンのクロックが正しくない可能性があります。これは、コンピューターが Docker を実行していて、その後にスリープ状態または休止状態から再開した場合に発生することがあります。

試すソリューション

- Docker を再起動します。
- Windows PowerShell で時刻同期を無効にしてから再度有効にします。

```
Get-VMIntegrationService -VMName DockerDesktopVM -Name "Time Synchronization" |
 Disable-VMIntegrationService
Get-VMIntegrationService -VMName DockerDesktopVM -Name "Time Synchronization" |
 Enable-VMIntegrationService
```

PathfindingSample コンソールクライアントが接続に失敗する

のチュートリアルで説明されているPathfindingSampleシミュレーションに接続すると、コンソールクライアントから次のエラーが表示されることがありますの開始方法 SimSpace Weaver。このエラーは、指定した IP アドレスとポート番号を組み合わせた ViewApp へのネットワーク接続を、クライアントが開くことができないために発生します。

```
Fatal error in function nng_dial. Error code: 268435577. Error message: no link
```

のシミュレーションの場合 AWS クラウド

- ネットワーク接続は正しく機能していますか? 動作するはずの他の IP アドレスまたはウェブサイトに接続できることを確認します。ウェブブラウザがキャッシュからウェブサイトを読み込んでいないことを確認します。
- シミュレーションは実行中ですか? ListSimulations API を使用してシミュレーションのステータスを取得できます。詳細については、「カスタムアプリケーションの IP アドレスとポート番号を取得する」を参照してください。[SimSpace Weaver] コンソールを使用してシミュレーションのステータスを確認することもできます。
- アプリケーションは実行中ですか? DescribeApp API を使用してアプリケーションのステータスを取得できます。詳細については、「カスタムアプリケーションの IP アドレスとポート番号を取得する」を参照してください。[SimSpace Weaver] コンソールを使用してシミュレーションのステータスを確認することもできます。
- アプリケーションは実行中ですか? DescribeApp API を使用してアプリケーションのステータスを取得できます。詳細については、「カスタムアプリケーションの IP アドレスとポート番号を取得する」を参照してください。[SimSpace Weaver] コンソールを使用してシミュレーションのステータスを確認することもできます。

- 正しい IP アドレスとポート番号を使用しましたか? インターネット経由で接続する場合は、`ViewApp` の IP アドレスおよび `Actual` ポート番号を使用する必要があります。IP `Address` および `Actual` ポート番号は DescribeApp API 出力の `EndpointInfo` ブロックで確認できます。[SimSpace Weaver] コンソールを使用して、`MyViewDomain` 詳細ページの `ViewApp` で IP アドレス (URI) とポート番号 (入力ポート) を確認することもできます。
- ネットワーク接続はファイアウォールを経由していますか? ファイアウォールによって IP アドレスまたはポート番号 (あるいはその両方) への接続がブロックされている可能性があります。ファイアウォールの設定を確認するか、ファイアウォール管理者に確認してください。

ローカルシミュレーション用

- ループバックアドレス (127.0.0.1) に接続できますか? Windows に `ping` コマンドラインツールがある場合は、コマンドプロンプトウィンドウを開いて 127.0.0.1 に ping を実行します。Ctrl-C を押して ping を終了します。

```
ping 127.0.0.1
```

Example ping 出力

```
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:\>
```

ping でパケットが失われたと表示される場合は、他のソフトウェア (ローカルファイアウォール、セキュリティ設定、マルウェア対策プログラムなど) が接続をブロックしている可能性があります。

- アプリケーションは実行中ですか? ローカルシミュレーションはアプリケーションごとに別々のウィンドウとして実行されます。空間アプリケーションおよび `ViewApp` のウィンドウが開いていることを確認します。詳細については、「でのローカル開発 SimSpace Weaver」を参照してください。
- 正しい IP アドレスとポート番号を使用しましたか? ローカルシミュレーションに接続するときには `tcp://127.0.0.1:7000` を使用する必要があります。詳細については、「でのローカル開発 SimSpace Weaver」を参照してください。
- 接続を遮断する可能性のあるローカルセキュリティソフトウェアはありますか? セキュリティ設定、ローカルファイアウォール、またはマルウェア対策プログラムを確認して、TCP ポート`7000` での `127.0.0.1` への接続がブロックされていないか確認します。

AWS CLI が認識しない `simspaceweaver`

が、 が について知らないことを示すエラー AWS CLI を表示した場合は SimSpace Weaver、次のコマンドを実行します。

```
aws simspaceweaver help
```

次の行で始まり、使用可能なすべての選択肢を一覧表示するエラーが発生した場合、 は古いバージョンである AWS CLI 可能性があります。

```
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:
```

次のコマンドを実行して、 のバージョンを確認します AWS CLI。

```
aws --version
```

バージョンが 2.9.19 以前の場合は、 AWS CLIを更新する必要があります。の現在のバージョン AWS CLI は、2.9.19「」より後であることに注意してください。

を更新するには AWS CLI、「 バージョン 2 ユーザーガイド」の「 の最新バージョンのインストールまたは更新 AWS CLI」を参照してください。 AWS Command Line Interface

SimSpace Weaver シミュレーションスキーマリファレンス

SimSpace Weaver は YAML ファイルを使用してシミュレーションのプロパティを設定します。このファイルはシミュレーションスキーマ (または単にスキーマ) と呼ばれます。 SimSpace Weaver アプリケーション SDK に含まれるサンプルシミュレーションには、独自のシミュレーション用にコピーおよび編集できるスキーマが含まれています。

トピック

- 完全なスキーマの例
- スキーマフォーマット

完全なスキーマの例

次の例は、シミュレーションを SimSpace Weaver 記述する YAML形式のテキストファイルを示しています。この例では、プロパティのダミー値が含まれています。ファイルのフォーマットは、ファイルに指定されている `sdk_version` の値によって異なります。プロパティとその有効値の詳細については、「スキーマフォーマット」を参照してください。

```
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"]
```

スキーマフォーマット

以下の例は、スキーマ構築全体を示しています。親子関係が同じであれば、スキーマの各レベルでのプロパティの順序は関係ありません。配列内の要素では順序が重要です。

```
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
```

セクション

- SDK のバージョン
- シミュレーションのプロパティ
- ワーカー
- クロック
- パーティショニング戦略
- ドメイン
- 配置の制約事項

SDK のバージョン

`sdk_version` セクション (必須) は、このスキーマをサポートする SimSpace Weaver アプリケーション SDK のバージョンを識別します。有効な値: `1.17`、`1.16`、`1.15`、`1.14`、`1.13`、`1.12`

> **Important**
>
> `sdk_version` の値には、メジャーバージョン番号と最初のマイナーバージョン番号のみが含まれます。例えば、値 `1.12` は、`1.12.0`、`1.12.1`、`1.12.2` などのすべてのバージョン `1.12.x` を指定します。

```
sdk_version: "1.17"
```

シミュレーションのプロパティ

`simulation_properties` セクション (必須) は、シミュレーションのさまざまなプロパティを指定します。このセクションを使用してロギングを設定し、既定のコンテナイメージを指定します。このセクションは、ロギングを設定していない場合や、デフォルトのコンテナイメージを指定する場合にも必須です。

```
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"
```

プロパティ

`log_destination_resource_name`

SimSpace Weaver ログを書き込む先のリソースを指定します。

必須: いいえ。このプロパティが含まれていない場合、 SimSpace Weaver にはシミュレーションのログは書き込まれません。

タイプ: 文字列

有効値:

- CloudWatch Logs ロググループ名 (例: `MySimulationLogs`)
- CloudWatch Logs ロググループの Amazon リソースネーム (ARN) (例: `arn:aws:logs:us-west-2:111122223333:log-group/MySimulationLogs`)

> **Note**
>
> SimSpace Weaver は、シミュレーション AWS リージョン と同じアカウントおよび の口グ送信先のみをサポートします。

`log_destination_service`

ARN ではない `logging_destination resource_name` を指定した場合のロギング先リソースのタイプを示します。

必須: `log_destination_resource_name` が指定されていて ARN ではない場合は、このプロパティを指定する必要があります。`log_destination_resource_name` が指定されていない場合、または ARN の場合、このプロパティは指定できません。

タイプ: 文字列

有効値:

- `logs`: ログ送信先リソースはロググループです。

`default_entity_index_key_type`

シミュレーションエンティティのインデックスキーフィールドのデータタイプを指定します。

必須: はい

タイプ: 文字列

有効な値: `Vector3<f32>`

`default_image`

シミュレーションのデフォルトコンテナイメージを指定します (バージョン `1.13` および `1.12` ではサポートされていません)。このプロパティを指定すると、`image` を指定していないドメインは `default_image` を使用します。

必須: いいえ

タイプ: 文字列

有効値:

- Amazon Elastic Container Registry (Amazon ECR) 内のリポジトリの URI (例: `111122223333.dkr.ecr.us-west-2.amazonaws.com/my-ecr-repository:latest`)

ワーカー

`workers` セクション (必須) は、ワーカーグループ (ワーカーのグループ) の設定を指定します。SimSpace Weaver はこの情報を `placement_constraints` と一緒に使用して、シミュレーションの基盤となるインフラストラクチャを構成します。現在、1 ワーカーグループのみがサポートされています。

ワーカーグループのプロパティを指定するには、*`worker-group-name`* を任意の名前に置き換えます。名前は 3 ~ 64 文字で、A ~ Z、a ~ z、0 ~ 9、_ - (ハイフン) を含むことができます。名前の後にワーカーグループのプロパティを指定します。

```
workers:
  worker-group-name:
    type: "sim.c5.24xlarge"
    desired: number-of-workers
```

プロパティ

`type`

ワーカータイプを指定します。

必須: はい

タイプ: 文字列

有効な値: `sim.c5.24xlarge`

`desired`

このワーカーグループに必要なワーカー数を指定します。

必須: はい

タイプ: 整数

有効値: 1-3。シミュレーションのワーカー数の Service Quotas (制限) によって、このプロパティの最大値が決まります。例えば、Service Quotas が 2 の場合、このプロパティの最大値は 2 です。Service Quotas の引き上げをリクエストすることもできます。詳細については、「SimSpace Weaver エンドポイントとクォータ」を参照してください。

クロック

`clock` セクション (必須) は、シミュレーションクロックのプロパティを指定します。

```
clock:
  tick_rate: tick-rate
```

プロパティ

`tick_rate`

クロックがアプリケーションに公開する 1 秒あたりのティック数を指定します。

必須: はい

タイプ:

- バージョン *1.14* および *1.15*: 文字列
- バージョン *1.13* および *1.12*: 整数

有効値:

- バージョン *1.14* および *1.15*: `"10"` | `"15"` | `"30"` | `"unlimited"`
  - `"unlimited"`: クロックは、すべてのアプリケーションが現在のティックのコミット操作を完了するとすぐに次のティックを送信します。
- バージョン *1.13* および *1.12*: `10` | `15` | `30`

パーティショニング戦略

`partitioning_strategies` セクション (必須) は、空間ドメインのパーティションの構成を指定します。

> **Note**
>
> SimSpace Weaver は 1 つのパーティショニング戦略のみをサポートします。

パーティショニング戦略のプロパティを指定するには、*partitioning-strategy-name* を任意の名前に置き換えます。名前は 3 ~ 64 文字で、A ~ Z、a ~ z、0 ~ 9、_ - (ハイフン) を含むことができます。名前の後にパーティショニング戦略のプロパティを指定します。

```
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
```

プロパティ

`topology`

このパーティショニング戦略のトポロジ (パーティション配置スキーマ) を指定します。

必須: はい

タイプ: 文字列

有効な値: `"Grid"`

`aabb_bounds`

シミュレーションの主軸に沿ったバウンディングボックス (AABB) の境界を指定します。範囲を、各軸 (x および y) の最小値と最大値を (この順序で) 記述する 2 要素の配列として指定します。

必須: 条件的。トポロジが `"Grid"` に設定されている場合、このプロパティは必須 (指定のみ可能) です。

タイプ: `Float` 配列 (各軸)

有効値: `-3.4028235e38-3.4028235e38`

`grid_placement_groups`

グリッドトポロジの各軸 (X および Y) に沿った配置グループの数を指定します。配置グループは、空間的に隣接する (同じドメイン内の) パーティションの集まりです。

必須: 条件的。トポロジが `"Grid"` に設定されている場合、このプロパティは必須 (指定のみ可能) です。配置グループ設定を指定しない場合、 SimSpace Weaver が自動的に計算します。配置グループ設定なしでパーティショニング戦略を使用するドメインでは、`grid_partition` を指定する必要があります (「空間ドメインのパーティショニング戦略」を参照してください)。

タイプ: 整数 (各軸)

有効値: `1-20`。`x * y` は必要なワーカー数と等しくすることをお勧めします。それ以外の場合、SimSpace Weaver は使用可能なワーカー間でプレイスメントグループのバランスを取ります。

ドメイン

`domains` セクション (必須) は、各ドメインのプロパティを指定します。すべてのシミュレーションには、空間領域用のセクションが少なくとも 1 つ必要です。追加ドメイン用に複数のセクションを作成できます。各タイプのドメインには、独自の構成フォーマットがあります。

> **Important**
>
> バージョン `1.13` および `1.12` は複数の空間ドメインをサポートしていません。

> **Important**
>
> SimSpace Weaver は、シミュレーションごとに最大 5 つのドメインをサポートします。これには、すべての空間ドメイン、カスタムドメイン、およびサービスドメインが含まれます。

```
domains:
  domain-name:
    domain-configuration
  domain-name:
```

```
    domain-configuration
  ...
```

ドメイン設定

- 空間ドメイン設定
- カスタムドメイン設定
- サービスドメイン設定

空間ドメイン設定

空間ドメインのプロパティを指定するには、*spatial-domain-name* を任意の名前に置き換えます。名前は 3 ~ 64 文字で、A ~ Z、a ~ z、0 ~ 9、_ - (ハイフン) を含むことができます。名前の後に空間ドメインのプロパティを指定します。

```
  spatial-domain-name:
    launch_apps_by_partitioning_strategy:
      partitioning_strategy: "partitioning-strategy-name"
      grid_partition:
        x: number-of-partitions-along-x-axis
        y: number-of-partitions-along-y-axis
    app_config:
      package: "app-package-s3-uri"
      launch_command: ["app-launch-command", "parameter1", ...]
      required_resource_units:
        compute: app-resource-units
    image: "ecr-repository-uri"
```

空間ドメインのパーティショニング戦略

`launch_apps_by_partitioning_strategy` セクション (必須) は、シミュレーション空間のパーティショニング戦略と次元 (パーティション数) を指定します。

```
    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
```

プロパティ

`partitioning_strategy`

この空間ドメインのパーティショニング戦略を指定します。

必須: はい

タイプ: 文字列

有効値: このプロパティの値は、`partitioning_strategies` セクションで定義されているパーティショニング戦略の名前と一致する必要があります。詳細については、「パーティショニング戦略」を参照してください。

`grid_partition`

グリッドトポロジーの各軸 (x および y) に沿ったパーティションの数を指定します。これらの次元は、このドメインのシミュレーションスペースの合計を表します。

必須: 条件的。このプロパティは、トポロジーが `"Grid"` に設定されている場合にのみ指定できます。このプロパティは、このドメインに指定されているパーティショニング戦略の `grid_placement_groups` プロパティによって異なります。

- このドメインのパーティショニング戦略で `grid_placement_groups` 設定が指定されていない場合、このプロパティは必須です。
- `grid_placement_groups` 設定があっても `grid_partition` を指定しない場合、SimSpace Weaver は `grid_placment_groups` 設定と同じ次元を使用します。
- `grid_placement_groups` と `grid_partition` の両方を指定する場合、`grid_partition` の次元は`grid_placement_groups`の次元の倍数である必要があります (例えば、`grid_placement_groups` の次元が 2x2 の場合、`grid_partition` で有効な次元は 2x2、4x4、6x6、8x8、10x10 です)。

タイプ: 整数 (各軸)

有効値: `1-20`

空間アプリケーションの設定

`app_config` セクション (必須) は、このドメイン内のアプリケーションのパッケージ、起動設定、およびリソース要件を指定します。

```
    app_config:
      package: "app-package-s3-uri"
      launch_command: ["app-launch-command", "parameter1", ...]
      required_resource_units:
        compute: app-resource-units
```

プロパティ

`package`

アプリケーションの実行ファイル/バイナリを含むパッケージ (zip ファイル) を指定します。パッケージは Amazon S3 バケットに保存されている必要があります。zip ファイルフォーマットのみがサポートされています。

必須: はい

タイプ: 文字列

有効値: Amazon S3 バケット内のパッケージの Amazon S3 URI。例えば、`s3://weaver-myproject-111122223333-app-zips-us-west-2/MySpatialApp.zip` と指定します。

`launch_command`

アプリケーションを起動するための実行ファイル/バイナリファイル名とコマンドラインパラメータを指定します。各コマンドライン文字列トークンは配列内の要素です。

必須: はい

タイプ: 文字列配列

`required_resource_units`

SimSpace Weaver がこのアプリケーションの各インスタンスに割り当てるリソースユニットの数を指定します。リソースユニットは、ワーカー上の固定量の仮想化中央演算装置 (vCPUs) とランダムアクセスメモリ (RAM) です。リソースユニットの詳細については、「エンドポイントと Service Quotas」を参照してください。この `compute` プロパティは、`compute` ワーカーファミリーのリソースユニット割り当てを指定するもので、現状での唯一の有効な割り当てタイプです。

必須: はい

タイプ: 整数

有効値: 1-4

カスタムコンテナイメージ

`image` プロパティ (オプション) は、 がこのドメインでアプリケーションを実行するために SimSpace Weaver 使用するコンテナイメージの場所を指定します (バージョン `1.13`および ではサポートされていません`1.12`) 。イメージを含む Amazon Elastic Container Registry (Amazon ECR) 内のリポジトリに URI を指定します。このプロパティは指定されていないが、最上位の `simulation_properties` セクションで `default_image` が指定されている場合、このドメインのアプリケーションは `default_image` を使用します。詳細については、「カスタムコンテナ」を参照してください。

```
image: "ecr-repository-uri"
```

プロパティ

`image`

このドメイン内のアプリケーションを実行するコンテナイメージの場所を指定します。

必須: いいえ

タイプ: 文字列

有効値:

- Amazon Elastic Container Registry (Amazon ECR) 内のリポジトリの URI (例: `111122223333.dkr.ecr.us-west-2.amazonaws.com/my-ecr-repository:latest`)

カスタムドメイン設定

カスタムドメインのプロパティを指定するには、*`custom-domain-name`* を任意の名前に置き換えます。名前は 3～64 文字で、A～Z、a～z、0～9、_ - (ハイフン) を含むことができます。名前の後にカスタムドメインのプロパティを指定します。カスタムドメインごとにこの手順を繰り返します。

```
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"
```

プロパティ

`launch_apps_via_start_app_call`

このプロパティは StartApp API を使用してカスタムアプリケーションを起動するために必要です。

必須: はい

タイプ: なし

有効な値: `{}`

カスタムアプリケーションの設定

`app_config section` (必須) は、このカスタムドメイン内のアプリケーションのパッケージ、起動設定、リソース要件、ネットワークポートを指定します。

```
    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, ...]
```

プロパティ

`package`

アプリケーションの実行ファイル/バイナリを含むパッケージ (zip ファイル) を指定します。パッケージは Amazon S3 バケットに保存されている必要があります。zip ファイルフォーマットのみがサポートされています。

必須: はい

タイプ: 文字列

有効値: Amazon S3 バケット内のパッケージの Amazon S3 URI。例えば、`s3://weaver-myproject-111122223333-app-zips-us-west-2/MyCustomApp.zip` と指定します。

`launch_command`

アプリケーションを起動するための実行ファイル/バイナリファイル名とコマンドラインパラメータを指定します。各コマンドライン文字列トークンは配列内の要素です。

必須: はい

タイプ: 文字列配列

`required_resource_units`

SimSpace Weaver がこのアプリケーションの各インスタンスに割り当てるリソースユニットの数を指定します。リソースユニットは、ワーカー上の固定量の仮想化中央演算装置 (vCPUs) とランダムアクセスメモリ (RAM) です。リソースユニットの詳細については、「エンドポイントと Service Quotas」を参照してください。この `compute` プロパティは、`compute` ワーカーファミリーのリソースユニット割り当てを指定するもので、現状での唯一の有効な割り当てタイプです。

必須: はい

タイプ: 整数

有効値: 1-4

`endpoint_config`

このドメイン内のアプリケーションのネットワークエンドポイントを指定します。の値は、受信クライアント接続のためにカスタムアプリケーションがバインドするポート`ingress_ports`を指定します。 は、動的に割り当てられたポートを指定されたイングレスポートに SimSpace Weaver マッピングします。入力ポートは TCP と UDP の両方です。DescribeApp API を使用して、クライアントに接続するための実際のポート番号を検索します。

必須: いいえ。 エンドポイント設定を指定しない場合、このドメインのカスタムアプリケーションにはネットワークエンドポイントがありません。

タイプ: 整数配列

有効値: `1024-49152`。値は一意である必要があります。

カスタムコンテナイメージ

`image` プロパティ (オプション) は、 がこのドメインでアプリケーションを実行するために SimSpace Weaver 使用するコンテナイメージの場所を指定します (バージョン `1.13`および ではサポートされていません`1.12`) 。イメージを含む Amazon Elastic Container Registry (Amazon ECR) 内のリポジトリに URI を指定します。このプロパティは指定されていないが、最上位の `simulation_properties` セクションで `default_image` が指定されている場合、このドメインのアプリケーションは `default_image` を使用します。詳細については、「カスタムコンテナ」を参照してください。

```
image: "ecr-repository-uri"
```

プロパティ

`image`

このドメイン内のアプリケーションを実行するコンテナイメージの場所を指定します。

必須: いいえ

タイプ: 文字列

有効値:

- Amazon Elastic Container Registry (Amazon ECR) 内のリポジトリの URI (例: `111122223333.dkr.ecr.us-west-2.amazonaws.com/my-ecr-repository:latest`)

サービスドメイン設定

サービスドメインのプロパティを指定するには、*`service-domain-name`* を任意の名前に置き換えます。名前は 3～64 文字で、A～Z、a～z、0～9、_ - (ハイフン) を含むことができます。名前の後にサービスドメインのプロパティを指定します。サービスドメインごとにこの手順を繰り返します。

```
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"
```

ワーカーごとにアプリケーションを起動する

launch_apps_per_worker セクション (必須) は、これがサービスドメイン設定であることを示し、ワーカーごとに起動するサービスアプリケーションの数を指定します。

```
    launch_apps_per_worker:
      count: number-of-apps-to-launch
```

プロパティ

count

このプロパティは、ワーカーごとに起動するサービスアプリケーションの数を指定します。

必須: はい

タイプ: 整数

有効値: {} | 1 | 2。{} の値はデフォルトの 1 の値を指定します。

サービスアプリケーションの設定

app_config section (必須) は、このサービスドメイン内のアプリケーションのパッケージ、起動設定、リソース要件、ネットワークポートを指定します。

```
    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, ...]
```

プロパティ

`package`

アプリケーションの実行ファイル/バイナリを含むパッケージ (zip ファイル) を指定します。パッケージは Amazon S3 バケットに保存されている必要があります。zip ファイルフォーマットのみがサポートされています。

必須: はい

タイプ: 文字列

有効値: Amazon S3 バケット内のパッケージの Amazon S3 URI。例えば、`s3://weaver-myproject-111122223333-app-zips-us-west-2/MyServiceApp.zip` と指定します。

`launch_command`

アプリケーションを起動するための実行ファイル/バイナリファイル名とコマンドラインパラメータを指定します。各コマンドライン文字列トークンは配列内の要素です。

必須: はい

タイプ: 文字列配列

`required_resource_units`

SimSpace Weaver がこのアプリケーションの各インスタンスに割り当てるリソースユニットの数を指定します。リソースユニットは、ワーカー上の固定量の仮想化中央演算装置 (vCPUs) とランダムアクセスメモリ (RAM) です。リソースユニットの詳細については、「エンドポイントと Service Quotas」を参照してください。この `compute` プロパティは、`compute` ワーカーファミリーのリソースユニット割り当てを指定するもので、現状での唯一の有効な割り当てタイプです。

必須: はい

タイプ: 整数

有効値: 1-4

`endpoint_config`

このドメイン内のアプリケーションのネットワークエンドポイントを指定します。`ingress_ports` の値は、サービスアプリケーションが受信クライアント接続にバインドするポートを指定します。 SimSpace Weaver は動的に割り当てられたポートを、指定した入力ポートにマッピングします。入力ポートは TCP と UDP の両方です。DescribeApp API を使用して、クライアントに接続するための実際のポート番号を検索します。

必須: いいえ。 エンドポイント設定を指定しない場合、このドメインのサービスアプリケーションにはネットワークエンドポイントがありません。

タイプ: 整数配列

有効値: `1024-49152`。値は一意である必要があります。

カスタムコンテナイメージ

`image` プロパティ (オプション) は、 がこのドメインでアプリケーションを実行するために SimSpace Weaver 使用するコンテナイメージの場所を指定します (バージョン `1.13`および ではサポートされていません`1.12`) 。イメージを含む Amazon Elastic Container Registry (Amazon ECR) 内のリポジトリに URI を指定します。このプロパティは指定されていないが、最上位の `simulation_properties` セクションで `default_image` が指定されている場合、このドメインのアプリケーションは `default_image` を使用します。詳細については、「カスタムコンテナ」を参照してください。

```
image: "ecr-repository-uri"
```

プロパティ

`image`

このドメイン内のアプリケーションを実行するコンテナイメージの場所を指定します。

必須: いいえ

タイプ: 文字列

有効値:

- Amazon Elastic Container Registry (Amazon ECR) 内のリポジトリの URI (例: `111122223333.dkr.ecr.us-west-2.amazonaws.com/my-ecr-repository:latest`)

配置の制約事項

`placement_constraints` セクション (オプション) では、 SimSpace Weaver がどの空間ドメインを同じワーカーにまとめて配置するかを指定します。詳細については、「空間ドメインの設定」を参照してください。

> **Important**
>
> バージョン 1.13 および 1.12 は placement_constraints をサポートしていません。

```
placement_constraints:
  - placed_together: ["spatial-domain-name", "spatial-domain-name", ...]
    on_workers: ["worker-group-name"]
```

プロパティ

placed_together

が一緒に配置 SimSpace Weaver する空間ドメインを指定します。

必須: はい

タイプ: 文字列配列

有効値: スキーマで指定されている空間ドメインの名前

on_workers

ドメインを配置 SimSpace Weaver するワーカーグループを指定します。

必須: はい

タイプ: 1 要素の文字列配列

有効値: スキーマで指定されたワーカーグループの名前

SimSpace Weaver API リファレンス

SimSpace Weaver には、2 つの異なるアプリケーションプログラミングインターフェイス (APIs セットがあります。

- サービス API — これらの API は、シミュレーション、クロック、アプリケーションなどのサービスとサービスリソースを制御します。これらはメイン AWS ソフトウェア開発キット (SDK) の一部であり、コマンドラインインターフェイス (CLI) AWS を使用して呼び出すことができます。サービス API の詳細については、「SimSpace Weaver API リファレンス」を参照してください。
- アプリケーション SDK API — これらの API はシミュレーション内のデータを制御します。これらをアプリケーションコードで使用して、エンティティフィールドデータの読み取りと書き込み、サブスクリプションの操作、シミュレーション内のイベントの監視などを行います。詳細については、 SimSpace Weaver App SDK フォルダの App SDK ドキュメントを参照してください。 *sdk-folder*`\SimSpaceWeaverAppSdk\documentation`

> **Note**
>
> *sdk-folder* は SimSpaceWeaverAppSDKDistributable パッケージを解凍したフォルダです。

AWS SimSpace Weaver バージョン

私たちは AWS SimSpace Weaverを継続的に改善しています。新機能や機能の更新を利用する場合は、新しいバージョンをリリースするときに最新の SimSpace Weaver アプリケーション SDK をダウンロードする必要があります。既存のシミュレーションを新しいバージョンで実行するには、スキーマとコードを更新してから、シミュレーションの新しいインスタンスを起動する必要がある場合があります。アップグレードする必要はなく、以前のバージョンで既存のシミュレーションを引き続き実行できます。このページでバージョン間の違いを確認できます。現在、すべてのバージョンがサポートされています。

> **Important**
>
> AWS SimSpace Weaver ユーザーガイドの最新バージョンは、サービスの最新バージョンのみを対象としています。以前のバージョンのドキュメントは、AWS SimSpace Weaver 「ガイドカタログ」にあります。メインドキュメントのランディングページから入手できます。

最新バージョン

最新バージョン: 1.17.0

現在のバージョンを検索する方法

SimSpace Weaver アプリケーション SDK でシミュレーションを作成した場合、`create-project`スクリプトは SDK ライブラリのバージョンを のサブドレクトリーにダウンロードします*`sdk-folder`*。SDK ライブラリを含むサブディレクトリには、SDK バージョン番号 `SimSpaceWeaverAppSdk-`*`sdk-version`* を含む名前が付いています。たとえば、バージョン 1.16.0 のライブラリは にあります`SimSpaceWeaverAppSdk-1.16.0`。

SimSpace Weaver アプリケーションの SDK 配布可能パッケージのバージョンは、 のテキストファイル`app_sdk_distributable_version.txt`でも確認できます*`sdk-folder`*。

最新バージョンをダウンロードする

最新バージョンをダウンロードするには、以下のリンクを使用します。

- アプリケーション SDK の配布可能パッケージ一式

- アプリケーション SDK ライブラリのみ

のSimSpace Weaver コンソールから完全な SimSpace Weaver App SDK 配布パッケージをダウンロードすることもできます AWS Management Console。ナビゲーションペインで [アプリ SDK をダウンロード] を選択します。

> **Warning**
>
> を使用して AWS CLI 、 SimSpace Weaver アプリケーション SDK 配布可能パッケージと思われるものをダウンロードしないでください。このページのダウンロードリンクまたはコンソールのダウンロードリンクのみを使用してください。その他のダウンロード方法や場所はサポートされていないため、古いコード、正しくないコード、または悪意のあるコードが含まれている可能性があります。

アプリケーション SDK のダウンロードに関するトラブルシューティング

私たちは、Amazon CloudFront (CloudFront) を使用してアプリケーション SDK の.zip ファイルを配布しています。以下のような状況が発生する可能性があります。

- ダウンロードしたパッケージが最新バージョンではない
  - ダウンロードした.zip ファイルに最新バージョンが含まれていない場合は、CloudFront エッジロケーションのキャッシュがまだ更新されていない可能性があります。24 時間後にもう一度ダウンロードしてください。
- ダウンロードリンクを使用すると、HTTP 4xx または 5xx エラーが表示される
  - 24 時間後にもう一度試してください。同じエラーが表示される場合は、[SimSpace Weaver] コンソールの下部にある [フィードバック] リンクを使用して問題を報告してください。フィードバックのタイプは、[問題を報告] を選択します。
- ブラウザがページを読み込めないと報告する
  - ローカルネットワークまたはブラウザの設定に問題がある可能性があります。他のページを読み込めるか確認してください。ブラウザのキャッシュをクリアして、もう一度試してください。ダウンロード URL をブロックする可能性のあるファイアウォールルールがないことを確認してください。
- ファイルを保存しようとするとエラーが発生する
  - ローカルファイルシステムのアクセス許可を確認して、ファイルを保存するための正しいアクセス許可を持っていることを確認してください。

- ブラウザに「AccessDenied」と表示される
  - URL をブラウザに手動で入力した場合は、URL が正しいことを確認してください。ダウンロードリンクを使用した場合は、ブラウザの URL に干渉するものがないことを確認し、リンクをもう一度使用してください。

最新バージョンをインストールする

最新バージョンをインストールするには

1. 最新バージョンをダウンロードします。
2. SimSpaceWeaverAppSdkDistributable.zip をフォルダに解凍します。
3. 解凍した最新バージョンの SimSpace Weaver アプリケーション SDK フォルダ`python setup.py`から を実行します。
4. 以前のバージョンではなく、解凍された最新バージョンの SimSpace Weaver アプリケーション SDK フォルダを使用します。

サービスバージョン

バージョン	メモ	リリース日	ドキュメント	アプリケーション SDK ダウンロード
1.17.0	SimSpace Weaver アプリケーション SDK 配布可能パッケージの主な変更点 • Windows バッチスクリプトと Linux Bash スクリプトを Python ベースのスクリプトに置き換えました。したがって、Pyt	2024 年 4 月 17 日	本ガイド	• パッケージ一式 • ライブラリのみ トラブルシューティングを参照してください。

バージョン	メモ	リリース日	ドキュメント	アプリケーション SDK ダウンロード
	hon SDK を使用しない (または使用する予定がない場合でも)、Python 3.9 はスクリプトとサンプルを使用する必要があるようになりました。 • このリリースでは、Amazon Linux 2 のサポートが強化されました。 • でいくつかのバグを修正しましたSimSpace Weaver Local。 詳細については、「release notes」を参照してください。 バグ修正 • リモートワーカー間の転送を完了しなかった場合にエンティティが所有しな			

バージョン	メモ	リリース日	ドキュメント	アプリケーション SDK ダウンロード
	くなるバグを修正しました。			
1.16.0	新機能: • SimSpace Weaver アプリケーション SDK でメッセージング APIs を使用して、アプリケーション間でメッセージを送受信できるようになりました。この機能は、C++、Python、Unity および Unreal Engine 5 の統合で使用できます。	2024 年 2 月 12 日	AWS SimSpace Weaver ガイドカタログを参照してください。	• パッケージ一式 • ライブラリのみ トラブルシューティングを参照してください。

バージョン	メモ	リリース日	ドキュメント	アプリケーション SDK ダウンロード
1.15.3	SimSpace Weaver Local の更新: • SimSpace Weaver Local を変更し、 AWS クラウドの開発とより密接に連携するようにしました。これらの変更は、SimSpace Weaver Local の C++、Python、Unity、および Unreal Engine のプロジェクトとワークフローに影響します。	2023 年 12 月 4 日	AWS SimSpace Weaver ガイドカタログを参照してください。	ダウンロードできません
1.15.2	App SDK 配布可能パッケージの更新: • cmake の特定の必要なバージョンを使用するように `Dockerfile` を更新しました。この変更を行わない場合、Docker コンテナのビルドに失敗する可能性があります。	2023 年 11 月 2 日	AWS SimSpace Weaver ガイドカタログを参照してください。	ダウンロードできません

バージョン	メモ	リリース日	ドキュメント	アプリケーション SDK ダウンロード
1.15.1	機能更新: • Python SDK: このリリースでは、 AWS クラウドで Python ベースのシミュレーションが失敗する原因となっていた問題が修正されています。	2023 年 9 月 22 日	AWS SimSpace Weaver ガイドカタログを参照してください。	ダウンロードできません
1.15.0	新機能: • Python SDK: Python を使用してシミュレーション開発ができるようになりました。 SimSpace Weaver アプリケーション SDK の配布可能パッケージには、サンプル Python プロジェクトとその Python ビュークライアントのテンプレートが含まれています。	2023 年 8 月 31 日	AWS SimSpace Weaver ガイドカタログを参照してください。	ダウンロードできません

バージョン	メモ	リリース日	ドキュメント	アプリケーション SDK ダウンロード
1.14.0	新機能: • カスタムコンテナ: 独自の Amazon Linux 2 (AL2) ベースのコンテナイメージを作成し、Amazon Elastic Container Registry (Amazon ECR) に保存し、それを使用して AWS クラウドで SimSpace Weaver アプリケーションを実行します。 • 複数の空間ドメイン: シミュレーションで複数の空間ドメインを作成します。シミュレーションロジックを、すべて単一の空間アプリケーションにまとめるのではなく、分離します。要件に基づいてさまざまなリソースを	2023 年 7 月 26 日	AWS SimSpace Weaver ガイドカタログを参照してください。	ダウンロードできません

バージョン	メモ	リリース日	ドキュメント	アプリケーション SDK ダウンロード
	空間ドメインに割り当てます。 • 無制限のティックレート: コード実行と同じ速さでシミュレーションを実行できます。すべてのアプリケーションが現在のティックのコミット操作を完了するとすぐに次のティックが送信されるように、シミュレーションのクロックを設定します。 SimSpace Weaver アプリケーション SDK: • `tick_rate` の値が文字列になりました。値には二重引用符 (") を含める必要があります。以前のバージョンのティックレート			

バージョン	メモ	リリース日	ドキュメント	アプリケーション SDK ダウンロード
	は整数のままです。			
1.13.1	SimSpace Weaver アプリケーション SDK: • 機能更新: プロジェクト作成が `PathfindingSampleUnreal` テンプレートで正しく機能するようになりました。	2023 年 6 月 7 日	AWS SimSpace Weaver ガイドカタログを参照してください。	ダウンロードできません

バージョン	メモ	リリース日	ドキュメント	アプリケーション SDK ダウンロード
1.13.0	SimSpace Weaver サービス APIs: • • 新しい `CreateSnapshot` アクション • `StartSimulation` アクションへの変更: • スナップショットから開始する `SnapshotS3Location` パラメータを追加しました。 • `SchemaS3Location` パラメータはオプションです。 • `DescribeSimulation` 出力への変更: • `SchemaError` を廃止しました。 • `StartError` フィール	2023 年 4 月 29 日	AWS SimSpace Weaver ガイドカタログを参照してください。	ダウンロードできません

バージョン	メモ	リリース日	ドキュメント	アプリケーション SDK ダウンロード
	ドを追加しました。 • SnapshotS3Location フィールドを追加しました。 • SNAPSHOT_IN_PROGRESS シミュレーションステータスを追加しました。 • 新しい S3Destination データタイプ SimSpace Weaver コンソール： • スナップショットを作成する新機能。 • スナップショットからシミュレーションを開始する新機能。			

バージョン	メモ	リリース日	ドキュメント	アプリケーション SDK ダウンロード

	SimSpace Weaver アプリケーション SDK:
	• スナップショットをサポートする新しいスクリプト
	  • `create-snapshot-`*proj*
	  name `.bat`
	  • `start-from-snapshot-`*project-*
	  name `.bat`
	  • `quick-start-from-snapshot-`*pro*
	  name `-cli.bat`
	  • `list-snapshots-`*proje*
	  name `.bat`
	• プロジェクトでは、プロジェクトごとに単一の Amazon S3 バケット (weaver-*lowercas*
	-project-
	name -account-
	n
	umber -region)			

バージョン	メモ	リリース日	ドキュメント	アプリケーション SDK ダウンロード
	を使用するようになりました			
1.12.3	SimSpace Weaver アプリケーション SDK: • 以下のスクリプトが `--maximum-duration` パラメータをサポートするようになりました。 • `quick-start-`*`project-name`*`-cli.bat` • `quick-start-`*`project-name`*`-cli.sh` • `start-simulation-`*`project-name`*`.bat` • `start-simulation-`*`project-name`*`.sh` • `run-`*`project-name`*`.bat` • `run-`*`project-name`*`.sh`	2023 年 3 月 27 日	AWS SimSpace Weaver ガイドカタログを参照してください。	ダウンロードできません

バージョン	メモ	リリース日	ドキュメント	アプリケーション SDK ダウンロード
1.12.2	SimSpace Weaver アプリケーション SDK: • バグ修正: `docker-create-image.bat` が正しく動作するようになりました。	2023 年 3 月 1 日	AWS SimSpace Weaver ガイドカタログを参照してください。	ダウンロードできません

バージョン	メモ	リリース日	ドキュメント	アプリケーション SDK ダウンロード
1.12.1	SimSpace Weaver アプリケーション SDK: • スクリプトは、AWS 認証に使用する AWS CLI プロファイルを受け入れるようになりました。 • スクリプトが AWS 認証 AWS IAM Identity Center をサポートするようになりました。 SimSpace Weaver Local: • バグ修正: すべての空間アプリケーションがシミュレーションに参加していない場合でも `Api::BeginUpdateWillBlock` が `true` を正しく返すようになりました。	2023 年 2 月 28 日	AWS SimSpace Weaver ガイドカタログを参照してください。	ダウンロードできません

バージョン	メモ	リリース日	ドキュメント	アプリケーション SDK ダウンロード
1.12.0	一般提供 (GA) のリリース	2022 年 11 月 29 日	AWS SimSpace Weaver ガイドカタログを参照してください。	ダウンロードできません

AWS SimSpace Weaver バージョン 1.17.0

このリリースは、 SimSpace Weaver アプリケーション SDK 配布可能パッケージの見直しです。古い Windows バッチスクリプトと Linux Bash スクリプトを Python ベースのスクリプトに置き換えました。

> **Important**
>
> Python 3.9 は、Python SDK だけでなく、スクリプトとサンプルを使用するための要件になりました。

目次

- 1.17.0 の主な変更点
- プロジェクトを 1.17.0 に更新する
- バージョン 1.17.0 に関するよくある質問

1.17.0 の主な変更点

- プロジェクト作成の簡素化
  - を実行したら`setup.py`、サンプルをコピーペーストするだけで独自のプロジェクトを作成できます。
- ワンクリックサンプル
  - ディストリビューション zip ファイルにはready-to-useサンプルが含まれるようになりました。
- 各 SDK は、、`cpp`、`python`、`unreal`および という独自のディレクトリに存在するようになりました`unity`。使用する SDK によっては、パスを更新する必要がある場合があります。

- ヘルパースクリプトの改善。
  - スクリプトには、柔軟性を最大化するための複数の AWS CLI オプションが含まれるようになりました。
  - クイックスタートの一部として統合されたコンソールクライアントの起動と接続。
  - コンソールの出力が改善されました。
  - Unreal および Unity サンプル構築が `quick-start`で機能するようになりました。手動ステップは必要ありません。
  - SimSpace Weaver Local は `quick-start`を呼び出すだけで機能するようになりました。手動による構築と起動はもうありません。
  - SimSpace Weaver Local `quick-start` には、アプリ出力のログ記録のサポートが統合されています。
  - SimSpace Weaver Local は、ssh セッションなど、GUI 以外の環境で起動できるようになりました。
  - 「カスタムコンテナ」機能が`quick-start`スクリプトに統合されました。
  - Amazon Linux 2 (AL2) のサポート強化: Windows と AL2 のスクリプトワークフローは同等になりました。以前は、AL2 プロジェクトではより多くの手動ステップが必要でしたが、AL2 ではサポートされていませんSimSpace Weaver Localでした。
- Unreal Engine プラグインと Unity プラグインが、 SimSpace Weaver アプリケーション SDK 配布可能パッケージの一部として含まれるようになりました。
- のバグ修正 SimSpace Weaver Local
  - エンティティに同じエンティティ ID を割り当てるバグを修正しました。
  - 2 つのパーティションに同じパーティション ID を割り当てることができるバグを修正しました。
  - 所有していないエンティティにアプリケーションが書き込もうとすることに関連するバグを修正しました。
  - メモリリークの問題を解決しました。

プロジェクトを 1.17.0 に更新する

1. 1.17.0 ディストリビューションをセットアップする: 1.17.0 用に変更したため、セットアップ手順を再度実行します。詳細については、「のセットアップ SimSpace Weaver」を参照してください。

2. 各 Weaver App SDK が独自のディレクトリに存在するようになりました。これを反映するようにビルドパスを更新します。

   a. C++ ディレクトリ: `SimSpaceWeaverAppSdk/cpp`

      - C++ アプリケーション SDK SimSpace Weaver は `FindSimSpaceWeaverAppSdk.cmake` ファイルを使用するようになりました。このファイルは、 にリンクされる`weaver`ターゲットを設定し、 で Weaver 用に構築する際の重要なバグ修正が含まれています AWS クラウド。バイナリに直接リンクする代わりに、これを使用する必要があります。

   b. Python ディレクトリ: `SimSpaceWeaverAppSdk/python`

   c. Unity プラグイン: `SimSpaceWeaverAppSdk/unity`

   d. Unreal Engine プラグイン: `SimSpaceWeaverAppSdk/unreal`

3. 前の`tools`スクリプトは新しい SimSpace Weaver ディストリビューションでは機能しません。プロジェクトで新しい`tools`スクリプトを使用するには：

   a. 古い `tools/windows`、`tools/linux`、および `tools/local` ディレクトリを削除します。

   b. プロジェクトと同じ SimSpace Weaver アプリケーション SDK を使用するサンプルプロジェクトの`tools`ディレクトリをコピーします。このディレクトリをコピー`setup.py`する前に、 が実行されていることを確認してください。

> **Important**
>
> ツールスクリプトは、サンプルプロジェクトでのみ動作することが保証されています。プロジェクトを操作するには、これらのスクリプト、特に `build.py`スクリプトを編集する必要がある場合があります。編集はプロジェクトに固有のため、ガイダンスを提供することはできません。

バージョン 1.17.0 に関するよくある質問

バージョン 1.17.0 に更新する必要がありますか？

SimSpace Weaver API または SimSpace Weaver アプリケーション SDK に変更がないため、これは必須の更新ではありません。複数のバグ修正を含む 1.17.0 を使用する場合はSimSpace Weaver Local、 を 1.17.0 に更新する必要があります。

必要な Python の最小バージョンは何ですか？

Python 3.9 は最小バージョンです。

最低限必要な CMake バージョンは何ですか？

CMake バージョン 3.13 が最小です。

Unreal Engine の最小バージョンは何が必要ですか？

Unreal Engine 5.0 が最小です。

Unity の最小バージョンは何が必要ですか？

Unity バージョン 2022.3.19.F1 が最小です。

AWS SimSpace Weaver バージョン 1.15.1

このリリースは、もともと SimSpace Weaver バージョン 1.15.0 でリリースされた Python SDK の必須アップデートです。Python ベースのシミュレーションが AWS クラウドで失敗する原因となっていたバージョン不一致の問題を修正しています。1.15.0 ではなく、このバージョンを使用してください。

既存の Python プロジェクトを 1.15.1 に更新します

バージョン 1.15.0 Python SDK で作成した既存の Python プロジェクトがある場合は、以下の手順を実行して 1.15.1 に更新し、 AWS クラウドで実行できるようにする必要があります。

この手順に従う代わりに、1.15.1 Python SDK を使用して新しい Python プロジェクトを作成し、カスタムコードを新しいプロジェクトに移動することもできます。

1.15.0 Python プロジェクトを 1.15.1 にアップデートする

1. Python プロジェクトのフォルダに移動します。
2. `src/PythonBubblesSample/bin/run-python` で以下の行を変更します。

   ```
   export PYTHONPATH=$PYTHONPATH:/roapp/lib
   ```

   項目の変更後:

```
export PYTHONPATH=$PYTHONPATH:$LD_LIBRARY_PATH:/roapp/lib
```

3. `CMakeLists.txt` で以下の行を削除します。

- ```
  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)
  ```

バージョン 1.15.1 のトラブルシューティング

1.15.0 Python シミュレーションを更新した後、 で開始できません。 AWS クラウド

症状: シミュレーションを開始してから約 5 ~ 10 分後に、シミュレーション管理ログに `internal error` が報告され、シミュレーションステータスは `FAILED` になります。

これは、1.15.0 Python SDK のライブラリファイルがアプリケーションの zip ファイルに含まれている場合に発生する可能性があります。プロジェクトを更新する手順を完了していることを確認し、`libweaver_app_sdk_python_v1.so` が zip ファイルに含まれていないこと、または何らかの方法で参照されていないことを確認してください。

バージョン 1.15.1 に関するよくある質問

このリリースは Python SDK 以外にも影響しますか？

いいえ。

バージョン 1.15.1 に更新する必要がありますか?

空間アプリケーションに Python を使用する予定がない場合は、1.15.1 に更新する必要はありません。を 1.15.0 に更新した場合、Python ベースのシミュレーションは では実行されません AWS クラウド。1.15.0 を使用している場合は 1.15.1 に更新することをお勧めします。

$LD_LIBRARY_PATHとは？

これは、 AWS クラウドでシミュレーションを実行するときの Python SDK の場所です。1.15.1 では新機能です。この変更は、将来発生する Python のバージョンに関する問題を避けるためのものです。このディレクトリへのリンクは、1.15.0 での `libweaver_app_sdk_python_v1.so` へのリンクと機能的には同じです。

のドキュメント履歴 AWS SimSpace Weaver

次の表に、 SimSpace Weaver ドキュメントの重要な変更点を示します。

日付	変更	ドキュメントの更新	API バージョンの更新
2025 年 5 月 20 日	サポート終了通知	サポート終了通知: 2026 年 5 月 20 日に、 AWS はサポートを終了します AWS SimSpace Weaver。2026 年 5 月 20 日以降、 SimSpace Weaver コンソールまたは SimSpace Weaver リソースにアクセスできなくなります。詳細については、AWS SimSpace Weaver 「サポート終了」を参照してください。	該当なし
2024 年 4 月 17 日	更新された内容	バージョン 1.17.0 リリースのユーザーガイド全体で更新されました。設定 章と入門チュートリアルの主な変更点。詳細については、リリースノートを参照してください。	該当なし
2024 年 2 月 12 日	更新された内容	バージョン 1.16.0 リリースのAWS SimSpace Weaver バージョン章を更新しました。	該当なし
2024 年 2 月 12 日	新しいコンテンツ	バージョン 1.16.0 リリースの一部として メッセージングセクションを追	該当なし

日付	変更	ドキュメントの更新	API バージョンの更新
		加しました。このセクションでは、 SimSpace Weaver アプリケーション SDK に追加されたメッセージング APIs について説明します。これらの APIs を使用して、アプリケーション間でメッセージを送受信できます。	
2024 年 2 月 12 日	更新された内容	バージョン 1.16.0 のSimSpace Weaver シミュレーションスキーマリファレンス章を更新しました。	該当なし
2024 年 2 月 12 日	更新された内容	メッセージングのサービスクォータを SimSpace Weaver エンドポイントとクォータ章に追加しました。	該当なし
2024 年 2 月 12 日	新規ガイド	1.16.0 より前のバージョンのコンテンツを別のガイドに分割します。以前のバージョンのAWS SimSpace Weaver ガイドにアクセスするためのガイドカタログ (メインドキュメントのランディングページから入手可能) を追加しました。	該当なし

日付	変更	ドキュメントの更新	API バージョンの更新
2023 年 12 月 4 日	更新された内容	バージョン 1.15.3 リリースの「AWS SimSpace Weaver バージョン」という章を更新しました。	該当なし
2023 年 12 月 4 日	更新された内容	最新バージョンのインストール手順を含むように「AWS SimSpace Weaver バージョン」という章を更新しました。	該当なし
2023 年 12 月 4 日	更新された内容	SimSpace Weaver Local の Service Quotas を更新しました。	該当なし
2023 年 12 月 4 日	新しいコンテンツと更新されたコンテンツ	「での ローカル開発 SimSpace Weaver」セクションを再構成し、バージョン 1.15.3 で導入された SimSpace Weaver Local との違いを説明する新しいページを追加しました。	該当なし
2023 年 11 月 7 日	更新されたコンテンツ	Docker と WSL がアプリケーション SDK の直接ダウンロードリンク/URL を使用するように設定する手順を更新しました。詳細については、「のローカル環境をセットアップする SimSpace Weaver」を参照してください。	該当なし

日付	変更	ドキュメントの更新	API バージョンの更新
2023 年 11 月 2 日	更新されたコンテンツ	1.15.2 リリースのサービスバージョンページを更新しました。詳細については、「サービスバージョン」を参照してください。	該当なし
2023 年 10 月 23 日	更新されたコンテンツ	サービスバージョンページを更新し、アプリケーション SDK 配布可能パッケージをダウンロードするための新しい手順を追加しました。お客様は、承認された直接ダウンロードリンクの 1 つのみを使用し、 AWS CLI を使用してアプリ SDK 頒布可能パッケージをダウンロードしないようにする必要があります。詳細については、「最新バージョンをダウンロードする」を参照してください。	該当なし
2023 年 9 月 22 日	新しいコンテンツ	1.15.1 リリースの更新手順が記載されたバージョンノートページを追加しました。詳細については、「AWS SimSpace Weaver バージョン 1.15.1」を参照してください。	該当なし

日付	変更	ドキュメントの更新	API バージョンの更新
2023 年 9 月 10 日	新しいコンテンツ	が を認識しない状況のトラブルシューティングセクションを追加 AWS CLI しました SimSpace Weaver。詳細については、「AWS CLI が認識しない `simspaceweaver` 」を参照してください。	該当なし
2023 年 9 月 10 日	更新された内容	WSL AWS CLI の のインストール手順を更新しました。詳細については、「で Amazon Linux 2 (AL2) の SimSpace Weaver ディストリビューションパッケージを設定する Windows Subsystem for Linux (WSL)」を参照してください。	該当なし
2023 年 9 月 7 日	API 更新	S3Location データタイプに BucketName と ObjectKey が必要になりました。S3Destination データタイプに BucketName が必要になりました。	AWS SDK: 2023-09-07
2023 年 8 月 31 日	新しいコンテンツ	リリース 1.15.0 の新しいセクションを追加しました: Python の使用。	該当なし

日付	変更	ドキュメントの更新	API バージョンの更新
2023 年 8 月 15 日	更新されたコンテンツ	公式の SimSpace Weaver Amazon S3 バケットのみを一覧表示するように AWS SimSpace Weaver バージョン のダウンロード手順を更新しました。その他のダウンロード場所は によって制御されず AWS 、悪意のあるコードが含まれている可能性があります。	該当なし
2023 年 7 月 26 日	更新されたコンテンツ	更新済み クロック。	該当なし
2023 年 7 月 26 日	更新されたコンテンツ	更新済み 空間ドメインの設定。	該当なし
2023 年 7 月 26 日	新しいコンテンツ	新しいセクション「カスタムコンテナ」を追加しました。	該当なし
2023 年 7 月 26 日	更新されたコンテンツ	AWS SimSpace Weaver バージョン をリリース 1.14.0 に更新しました。	該当なし
2023 年 7 月 6 日	新しいコンテンツ	新しいセクション「PathfindingSample コンソールクライアントが接続に失敗する」を追加しました。	該当なし
2023 年 6 月 7 日	更新されたコンテンツ	AWS SimSpace Weaver バージョン をリリース 1.13.1 に更新しました。	該当なし

日付	変更	ドキュメントの更新	API バージョンの更新
2023 年 5 月 15 日	新しいコンテンツ	新しいセクション「でのスナップショットの使用 AWS CloudFormation」を追加しました。	該当なし
2023 年 4 月 29 日	新しいコンテンツ	リリース 1.13.0 のコンテンツを追加しました。詳細については、「AWS SimSpace Weaver バージョン」を参照してください。	AWS SDK: 2023-04-28
2023 年 3 月 27 日	新しいコンテンツ	シミュレーションの最大時間について説明するセクションを追加しました。リリース 1.12.3 のチュートリアルに、SimSpace Weaver アプリケーション SDK スクリプトへの `--maximum-duration` パラメータのサポートが追加された注記を追加しました。	該当なし
2023 年 3 月 9 日	変更されたコンテンツ	Windows の Docker および Windows Subsystem for Linux (WSL) のみを説明対象としており、WSL (およびその他の Linux 環境) はサポートしていないことを明確にしました。	該当なし

日付	変更	ドキュメントの更新	API バージョンの更新
2023 年 2 月 28 日	新しいコンテンツ	SimSpace Weaver バージョンを説明する章を追加しました。	該当なし
2023 年 2 月 28 日	変更されたコンテンツ	() の AWS IAM Identity Center および名前付きプロファイルの使用を含めるように、認証に関するコンテンツを変更しました AWS Command Line Interface AWS CLI。	該当なし
2023 年 2 月 17 日	新しいコンテンツ	での リソースの管理に関するセクションを追加しました AWS CloudFormation。	該当なし
2023 年 1 月 23 日	新しいコンテンツ	ローカルシミュレーションをデバッグする手順を追加しました。	該当なし
2022 年 11 月 29 日	サービスの起動	SimSpace Weaverのユーザーガイドと API リファレンスをリリースしました。	AWS SDK: 2022-11-29

用語集

この用語集では、 固有の用語を定義します AWS SimSpace Weaver。

最新の AWS 用語については、 AWS 全般のリファレンスのAWS 用語集を参照してください。

A

アプリケーション	自分で作成する実行コード (バイナリとも呼ばれます)。アプリケーションという用語は、コードまたはそのコードの実行中のインスタンスを指す場合があります。アプリケーションはシミュレーションの動作をカプセル化したものです。アプリケーションはエンティティを作成、削除、読み取り、更新します。
アプリケーション SDK	アプリケーションを SimSpace Weaverと統合するために使用する Software Development Kit (SDK)。SDK には、エンティティデータの読み取りと書き込み、およびシミュレーション時間の追跡のための API が用意されています。詳細については、「SimSpace Weaver アプリケーション SDK」を参照してください。

C

クライアント	の外部に存在し SimSpace Weaver 、カスタムアプリまたはサービスアプリを介してシミュレーションとやり取りするプロセス (またはその定義) 。クライアントを使用してシミュレーションの状態を表示または変更できます。
クロック	SimSpace Weaverの内部スケジューリングプロセスの抽象化。クロックはアプリケーションにティックを表示して時間の同期を維持します。各シミュレーションには独自のクロックがあります。
クロックレート	クロックがアプリケーションに公開する 1 秒あたりのティック数。サポートされているクロックレートの詳細については、「SimSpace Weaver エンドポイントとクォータ」を参照してください。
クロックのティックレート	「クロックレート」を参照してください。

コンピュートリソースユニット	ワーカー上のコンピュートリソース (プロセッサとメモリ) のユニット。通常、アプリケーションの 単一のインスタンスには 1 つのコンピュートリソースユニットが割り当てられます。各アプリケーションには複数のコンピュートリソースユニットを割り当てることができます。
カスタムアプリケーション	シミュレーションの状態を読み取ったり操作したりするために使用するアプリケーションタイプ。カスタムアプリケーションはシミュレーションでエンティティを作成できますが、所有しているエンティティは作成できません。カスタムアプリケーションがエンティティを作成する場合、そのエンティティを空間ドメインに転送する必要があります。アプリケーション API を使用してカスタムアプリケーションのライフサイクルを制御します。 SimSpace Weaver APIs「」を参照してくださいSimSpace Weaver API リファレンス。
カスタムドメイン	カスタムアプリケーションを含むドメイン。
カスタムパーティション	カスタムアプリケーションのパーティション。

D

デッドライン	操作 (ティックの処理など) が完了するまでの実際の時間。
ドメイン	同じ実行コード (アプリケーションバイナリ) を実行し、同じ起動オプションを持つアプリケーションインスタンスのグループ。

E

エンドポイント (サービス)	プログラム (など) が SimSpace Weaver サービスへの接続に使用する完全修飾ドメイン名 (FQDN AWS Command Line Interface) 。
エンドポイント (シミュレーション)	クライアントがシミュレーションへの接続に使用する IP アドレスとポート番号。カスタムアプリケーションとサービスアプリケーションのエンドポイントを構成できます。
エンティティ	カスタマーデータオブジェクト (またはその定義)。エンティティは静的 (1 つの場所に留まる) でも動的 (シミュレーション空間内を移動する) でもかまいません。例えば、シミュレーション内の人や建物などです。

I

インデックス (シミュレーション)	空間境界や座標システムなど、シミュレーションの空間プロパティの説明。

L

(アプリケーションの) ライフサイクル	シミュレーション中にアプリケーションが行うと予想される論理的ステップの説明。ライフサイクルは、管理 (アプリSimSpace Weaver の開始と停止) または非管理 (アプリの開始と停止) のいずれかです。
ロード (エンティティフィールドデータ)	State Fabric からのエンティティフィールドデータの読み取り。

P

パーティション	ワーカーの共有メモリのセグメント。各パーティションには、ドメイン内のエンティティの個別のサブセットが含まれます。各アプリケーションにはパーティションが割り当てられています。アプリケーションは、そのパーティション内のすべてのエンティティを所有します。アプリケーションはエンティティを作成すると、そのパーティションにエンティティを作成します。エンティティがあるパーティションから別のパーティションに移動すると、所有権はソースパーティションのアプリケーションから転送先パーティションのアプリケーションに移ります。

R

リソースユニット	「???」を参照してください。

S

スキーマ	シミュレーションの設定を記述する YAML または JSON ドキュメント。SimSpace Weaver はスキーマを使用してシミュレーションリソースを作成します。
サービスアプリケーション	シミュレーションの状態を読み取ったり操作したりするために使用するアプリケーションタイプ。サービスアプリケーションはシミュレーションで

	エンティティを作成できますが、それらを空間ドメインに転送する必要があります。 SimSpace Weaver はサービスアプリケーションのライフサイクルを管理し、シミュレーションの各ワーカーで 1 つ (またはシミュレーションスキーマで指定されている場合はそれ以上) を起動します。
サービスドメイン	サービスアプリケーションを含むドメイン。
サービスパーティション	サービスアプリケーションのパーティション。
シミュレーション (リソース)	シミュレートされた仮想化空間を実行する計算クラスターを抽象化したもの。複数のシミュレーションを使用できます。シミュレーションはスキーマを使用して設定します。
空間アプリケーション	コアシミュレーションロジックをカプセル化するアプリケーションタイプ。各空間アプリケーションは 1 つ (1 つのみ) のパーティションを所有します。
空間ドメイン	空間アプリケーションを含むドメイン。
空間パーティション	空間アプリケーションのパーティション。
State Fabric	SimSpace Weaverのインメモリデータベース。は、エンティティや内部 SimSpace Weaver データを含むシミュレーションの状態State Fabricを保存します。
ストア (エンティティフィールドデータ)	State Fabric へのエンティティフィールドデータの書き込み。
サブスクリプション	特定のアプリケーションインスタンスがサブスクリプション領域からデータを受信するように要求する、長期にわたるリクエスト。サブスクライブしているアプリは、サブスクリプションを使用して、サブスクリプション領域内のエンティティへの変更を検出します。
サブスクリプション領域	シミュレーション空間の二次元領域。サブスクリプションはサブスクリプション領域を指します。サブスクリプション領域は複数のパーティションにまたがることができ、パーティションの一部を含むこともできます。サブスクリプション領域は定義された範囲内で連続しています。

T

ティック	時間の離散値 (ウォールクロック時間またはシミュレーション時間)。アプリケーションはティック時間よりも速く繰り返し処理できますが、特定のデッドライン内に指定されたティックを書き込むことが期待されます。特定のティックに対するすべてのアプリケーションのすべての操作は、次のティックが開始される前に完了する必要があります。
ティックレート	「クロックレート」を参照してください。
時間 (実際)	reality の観点から見た現在の時刻。 は、エポックからのナノ秒数である 64 ビット POSIX Unix タイムスタンプ SimSpace Weaver を使用します。 (January 1, 1970, 00:00:00 UTC).
時間 (シミュレーション)	simulation. の観点から見ると、現在の時刻は 64 ビット整数論理ティックカウンター SimSpace Weaver を使用します。これは実際の時刻に直接対応しない場合があります。

W

ワーカー	シミュレーションコードを実行する Amazon Elastic Compute Cloud (Amazon EC2) インスタンス。