



ユーザーガイド

AWS App Studio



AWS App Studio: ユーザーガイド

Copyright © 2026 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

AWS App Studio とは	1
App Studio を初めてお使いになる方向けの情報	1
概念	2
管理者ロール	2
アプリケーション (アプリ)	3
オートメーション	3
自動化アクション	3
ビルダーロール	3
コンポーネント	3
コネクタ	4
デベロッパー環境	4
エンティティ	4
インスタンス	4
ページ	5
Trigger トリガー)	5
App Studio の仕組み	6
アプリケーションを他の サービスに接続する	7
アプリケーションのデータモデルの設定	8
アプリケーションの UI の構築	9
アプリケーションのロジックまたは動作の実装	11
アプリケーションの開発ライフサイクル	13
詳細情報	14
App Studio のセットアップとサインイン	15
App Studio インスタンスを初めて作成して設定する	15
AWS アカウントにサインアップする	15
AWS リソースを管理するための管理ユーザーを作成する	16
で App Studio インスタンスを作成する AWS マネジメントコンソール	16
App Studio への招待の承諾	21
開始方法	22
チュートリアル: AI を使用してアプリを生成する	22
前提条件	23
ステップ 1: アプリケーションを作成する	23
ステップ 2: 新しいアプリケーションを調べる	24
ステップ 3: アプリケーションをプレビューする	26

次の手順	27
チュートリアル: 空のアプリから構築を開始する	27
前提条件	30
ステップ 1: アプリケーションを作成する	30
ステップ 2: エンティティを作成してアプリのデータを定義する	31
ステップ 3: ユーザーインターフェイス (UI) とロジックを設計する	33
ステップ 4: アプリケーションをプレビューする	37
ステップ 5: アプリケーションをテスト環境に公開する	37
次の手順	38
管理者向けドキュメント	39
グループとロールを使用したユーザーアクセスの管理	39
ロールとアクセス許可	39
グループの表示	40
ユーザーまたはグループの追加	40
グループロールの変更	42
ユーザーまたはグループの削除	42
コネクタを使用して他の サービスに接続する	43
AWS サービスに接続する	43
サードパーティーサービスに接続する	89
コネクタの表示、編集、削除	97
App Studio インスタンスの削除	98
Builder ドキュメント	99
チュートリアル	99
Amazon Bedrock を使用してテキストサマリアプリを構築する	99
Amazon S3 の操作	107
Lambda 関数の呼び出し	117
生成 AI を使用したアプリの構築	120
アプリの生成	120
アプリの構築または編集	120
データモデルの生成	121
サンプルデータの生成	121
AWS サービスのアクションの設定	121
レスポンスのモック	121
構築中に AI にヘルプを求める	121
アプリケーションの作成、編集、削除	122
Creating an application	122

アプリケーションのインポート	123
アプリケーションの複製	128
アプリケーションの編集または構築	129
以前に公開されたアプリバージョンを編集する	129
アプリケーションの名前変更	130
Deleting an application	131
アプリケーションのプレビュー、公開、共有	131
アプリケーションのプレビュー	131
アプリケーションの公開	132
公開されたアプリケーションの共有	137
以前に公開されたバージョンにロールバックする	138
アプリケーションのエクスポート	139
ページとコンポーネント: アプリケーションのユーザーインターフェイスを構築する	140
ページの管理	140
コンポーネントの管理	143
ページのロールベースの可視性の設定	144
アプリナビゲーションでのページの順序付けと整理	146
アプリテーマでアプリの色を変更する	147
コンポーネントリファレンス	148
自動化とアクション: アプリのビジネスロジックを定義する	194
オートメーションの概念	195
オートメーションの作成、編集、削除	196
自動化アクションの追加、編集、削除	198
オートメーションアクションのリファレンス	200
エンティティとデータアクション: アプリのデータモデルを設定する	219
データモデルを設計する際のベストプラクティス	219
エンティティの作成	221
エンティティの設定	224
エンティティの削除	232
マネージドデータエンティティ	232
ページとオートメーションのパラメータ	234
ページパラメータ	234
自動化パラメータ	235
JavaScript を使用して式を記述する	241
基本構文	241
Interpolation	242

連結	242
日付および時間	242
コードブロック	243
グローバル変数と関数	243
UI コンポーネント値を参照または更新する	243
テーブルデータの使用	245
オートメーションへのアクセス	246
データの依存関係とタイミングに関する考慮事項	249
例: 注文の詳細と顧客情報	249
データ依存関係とタイミングのベストプラクティス	249
複数のユーザーによるアプリの構築	251
ビルダーを招待してアプリを編集する	251
他のユーザーが編集中のアプリの編集を試みる	252
アプリのコンテンツセキュリティ設定の更新	252
トラブルシューティングとデバッグ	255
セットアップ、アクセス許可、オンボーディング	255
Create an account instance for me オプションを選択したときに App Studio の設定に失敗しました	255
設定後に App Studio にアクセスできない	256
App Studio にログインするときに使用するユーザー名またはパスワードがわからない	256
App Studio の設定時にシステムエラーが表示される	256
App Studio インスタンス URL が見つかりません	257
App Studio でグループまたはロールを変更できない	257
App Studio からオフボードする方法	255
アプリケーションのトラブルシューティングとデバッグ	257
AI ビルダーアシスタント	258
アプリスタジオ内	258
アプリケーションのプレビュー	259
テスト環境	260
CloudWatch でのログの使用	262
Connector	264
アプリケーションの公開と共有	267
共有ダイアログボックスに新しく作成されたアプリロールが表示されない	267
アプリの公開が完了したときに E メールが届かなかった	267
アプリのエンドユーザーが公開されたアプリにアクセスできない	267
セキュリティ	269

セキュリティに関する考慮事項と緩和策	270
セキュリティに関する考慮事項	270
セキュリティリスク軽減の推奨事項	271
データ保護	271
データ暗号化	272
転送中の暗号化	273
キー管理	273
ネットワーク間トラフィックのプライバシー	273
App Studio と Identity and Access Management	273
アイデンティティベースのポリシー	275
リソースベースのポリシー	276
ポリシーアクション	276
ポリシーリソース	277
ポリシー条件キー	277
ACL	277
ABAC	277
一時的な認証情報	277
プリンシパルアクセス権限	278
サービスロール	278
サービスリンクロール	278
AWS マネージドポリシー	279
サービスリンクロール	283
アイデンティティベースのポリシーの例	286
コンプライアンス検証	290
耐障害性	290
インフラストラクチャセキュリティ	291
設定と脆弱性の分析	291
サービス間での不分別な代理処理の防止	291
クロスリージョンデータ転送	292
サポートされるブラウザ	294
アプリケーションを構築するためにサポートされるブラウザと推奨ブラウザ	294
アプリケーションのエンドユーザーがサポートおよび推奨するブラウザ	294
ブラウザ設定を更新して App Studio でアプリケーションを構築する	295
クォータ	296
ドキュメント履歴	297
.....	cccviii

AWS App Studio とは

AWS App Studio は、自然言語を使用してエンタープライズグレードのアプリケーションを作成するのに役立つ生成 AI を活用したサービスです。App Studio は、IT プロジェクトマネージャー、データエンジニア、エンタープライズアーキテクトなどのソフトウェア開発スキルのない技術専門家にアプリケーション開発を提供します。App Studio を使用すると、運用上の専門知識を必要とせずに AWS、によって安全で完全に管理されたアプリケーションをすばやく構築できます。

ビルダーは App Studio を使用してアプリケーションを作成およびデプロイし、内部ビジネスプロセスをモダナイズできます。ユースケースの例としては、インベントリの管理と追跡、クレーム処理、従業員の生産性と顧客の成果を向上させるための複雑な承認などがあります。

トピック

- [App Studio を初めてお使いになる方向けの情報](#)

App Studio を初めてお使いになる方向けの情報

App Studio を初めて使用する場合は、まず以下のセクションを読むことをお勧めします。

- App Studio の設定、ユーザーとアクセスの管理、他の AWS またはサードパーティーのサービスとのコネクタの設定を行う管理者ロールを持つユーザーについては、[AWS App Studio の概念](#)「」および「」を参照してください[AWS App Studio のセットアップとサインイン](#)。
- アプリケーションを作成および開発するビルダーについては、[AWS App Studio の概念](#)「」および「」を参照してください[AWS App Studio の開始方法](#)。

AWS App Studio の概念

主要な App Studio の概念を理解して、チームのアプリケーションの作成とプロセスの自動化を高速化してください。これらの概念には、管理者とビルダーの両方に App Studio 全体で使用される用語が含まれます。

トピック

- [管理者ロール](#)
- [アプリケーション \(アプリ\)](#)
- [オートメーション](#)
- [自動化アクション](#)
- [ビルダーロール](#)
- [コンポーネント](#)
- [コネクタ](#)
- [デベロッパー環境](#)
- [エンティティ](#)
- [インスタンス](#)
- [ページ](#)
- [Trigger トリガー](#)

管理者ロール

Admin は、App Studio のグループに割り当てることができるロールです。管理者は、App Studio 内のユーザーとグループの管理、コネクタの追加と管理、ビルダーによって作成されたアプリケーションの管理を行うことができます。さらに、管理者ロールを持つユーザーには、ビルダーロールに含まれるすべてのアクセス許可があります。

Admin ロールを持つユーザーのみが Admin Hub にアクセスできます。Admin Hub には、ロール、データソース、アプリケーションを管理するためのツールが含まれています。

アプリケーション (アプリ)

アプリケーション (アプリ) は、エンドユーザーが特定のタスクを実行するために開発された単一のソフトウェアプログラムです。App Studio のアプリには、ユーザーが操作できる UI ページとコンポーネント、オートメーション、データソースなどのアセットが含まれます。

オートメーション

自動化は、アプリケーションのビジネスロジックを定義する方法です。オートメーションの主なコンポーネントは、オートメーションを開始するトリガー、1つ以上のアクションのシーケンス、オートメーションにデータを渡すために使用される入力パラメータ、および出力です。

自動化アクション

一般的にアクションと呼ばれるオートメーションアクションは、オートメーションを構成するロジックの個々のステップです。各アクションは、Eメールの送信、データレコードの作成、Lambda 関数の呼び出し、APIs呼び出しなど、特定のタスクを実行します。アクションはアクションライブラリのオートメーションに追加され、条件ステートメントまたはループにグループ化できます。

ビルダーロール

Builder は、App Studio のグループに割り当てることができるロールです。ビルダーはアプリケーションを作成および構築できます。ビルダーは、ユーザーまたはグループを管理したり、コネクタインスタンスを追加または編集したり、他のビルダーのアプリケーションを管理したりすることはできません。

Builder ロールを持つユーザーは Builder Hub にアクセスできます。これには、ビルダーがアクセスできるアプリケーションなどのリソースの詳細と、学習リソースなどの役立つ情報が含まれます。

コンポーネント

コンポーネントは、アプリケーションの UI 内の個々の機能項目です。コンポーネントはページに含まれており、一部のコンポーネントは他のコンポーネントのコンテナとして機能します。コンポーネントは、UI 要素とその UI 要素を実行するビジネスロジックを組み合わせます。たとえば、コンポーネントの1つのタイプは、ユーザーがフィールドに情報を入力し、送信するとその情報がデータベースレコードとして追加されるフォームです。

コネクタ

コネクタは、App Studio と、AWS Lambda や Amazon Redshift などの他の AWS サービス、またはサードパーティーのサービス間の接続です。コネクタを作成して設定すると、ビルダーはそのコネクタと、アプリケーションで App Studio に接続するリソースを使用できます。

管理者ロールを持つユーザーのみがコネクタを作成、管理、または削除できます。

デベロッパー環境

開発環境は、アプリケーションを構築するためのビジュアルツールです。この環境には、アプリケーションを構築するための以下のタブが含まれています。

- ページ: ビルダーが [ページ](#) と [コンポーネント](#) を使用してアプリケーションを設計する場所。
- 自動化: ビルダーが [自動化](#) を使用してアプリケーションのビジネスロジックを設計する場所。
- データ: ビルダーが [エンティティ](#) を使用してアプリケーションのデータモデルを設計する場所。

開発環境には、デバッグコンソールと、構築中にコンテキストに応じたヘルプを取得するための AI チャットウィンドウも含まれています。ビルダーは、開発環境から進行中のアプリケーションをプレビューできます。

エンティティ

エンティティは App Studio のデータテーブルです。エンティティはデータソースのテーブルと直接やり取りします。エンティティには、その中のデータを記述するフィールド、データを検索して返すクエリ、エンティティのフィールドをデータソースの列に接続するためのマッピングが含まれます。

インスタンス

インスタンスは、すべての App Studio リソースの論理コンテナです。ユーザー、会社、チーム、または組織を表し、ユーザーとグループのアプリケーション、コネクタ、ロールの割り当てなど、すべての App Studio リソースが含まれます。大規模な組織や企業には通常、サンドボックス、テスト、本番稼働用インスタンスなど、複数の App Studio インスタンスがあります。App Studio のセットアップの一環としてインスタンスを作成します。

ページ

ページは[コンポーネント](#)のコンテナであり、App Studio のアプリケーションの UI を構成します。各ページは、ユーザーが操作するアプリケーションのユーザーインターフェイス (UI) の画面を表します。ページは、アプリケーションスタジオのページタブで作成および編集されます。

Trigger トリガー)

トリガーは、オートメーションをいつ、どの条件で実行するかを決定します。トリガーの例としては、ボタン0n click用とテキスト入力0n select用があります。コンポーネントのタイプによって、そのコンポーネントの使用可能なトリガーのリストが決まります。トリガーは[コンポーネント](#)に追加され、アプリケーションスタジオで設定されます。

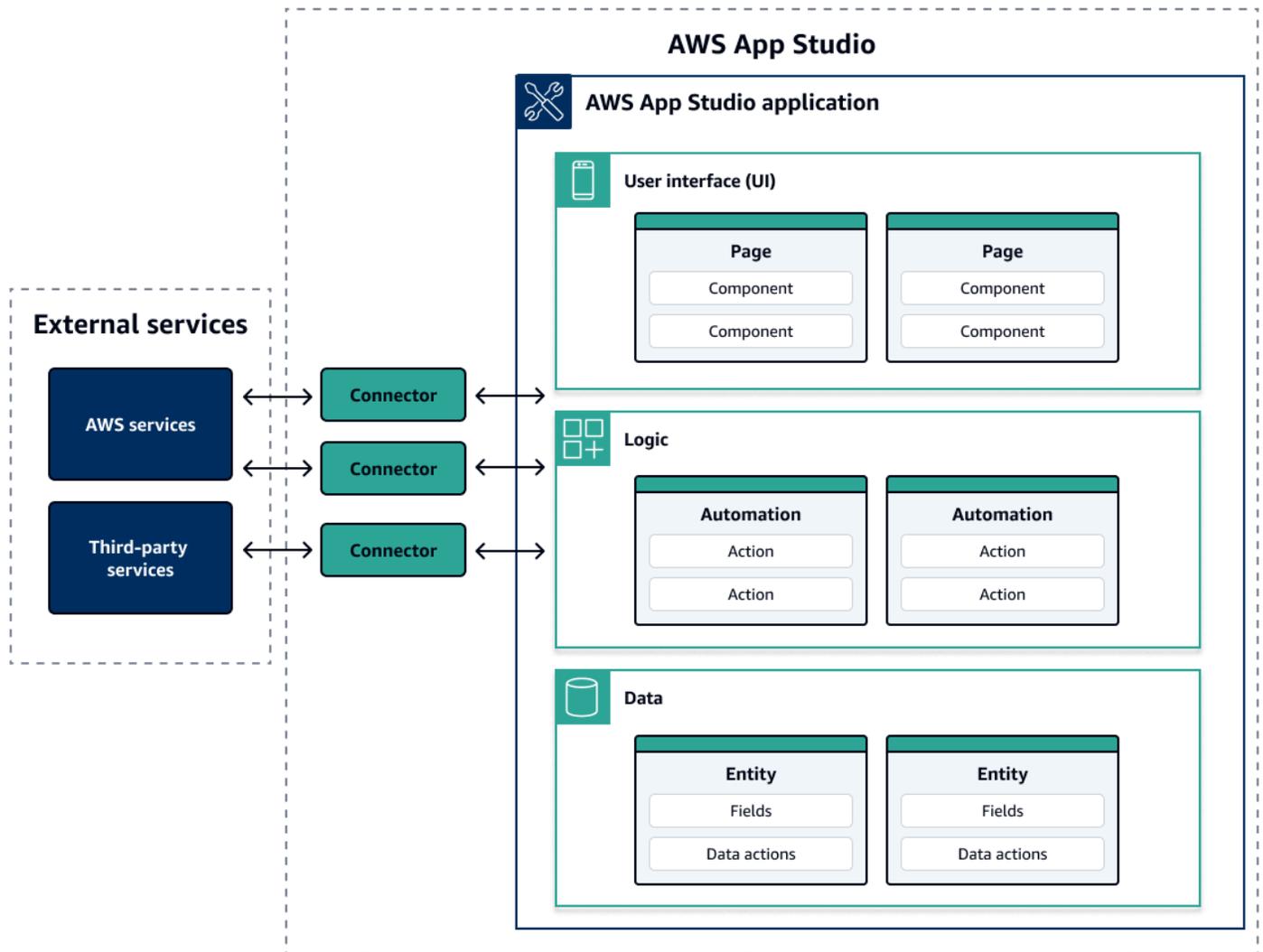
AWS App Studio の仕組み

AWS App Studio を使用してアプリケーションを構築する際に理解しておくべき重要な概念がいくつかあります。このトピックでは、以下の概念またはリソースの基本について説明します。

- コネクタを使用して他の サービスに接続し、アプリケーションでリソースまたは API コールを使用します。例えば、コネクタを使用してデータを保存してアクセスしたり、アプリから通知を送信したりできます。
- エンティティを使用してアプリケーションのデータモデルを設定し、アプリケーションを外部データソースに接続します。
- ページとコンポーネントを使用してアプリケーションのユーザーインターフェイス (UI) を構築します。
- オートメーションとアクションを使用して、アプリケーションのロジックまたは動作を実装します。
- App Studio でのアプリケーション開発ライフサイクル: 構築、テスト、公開。

App Studio の概念の詳細については、「」を参照してください [AWS App Studio の概念](#)。

次の図は、App Studio とそのリソースの整理方法の簡単な図です。



App Studio のアプリ内では、ページ、オートメーション、エンティティはすべて相互にやり取りします。コネクタを使用して、これらのリソースをデータ、ストレージ、通知プロバイダーなどの外部サービスに接続します。アプリを正常に構築するには、これらの概念とリソースが相互にどのように相互作用するかを理解することが重要です。

アプリケーションを他の サービスに接続する

App Studio を使用してアプリケーションを構築する最大の利点の 1 つは、アプリを他の サービスと簡単に統合できることです。App Studio では、サービスに固有のコネクタと、アプリケーションで使用するリソースまたは API コールを使用して、他の サービスに接続します。

コネクタは、個々のアプリケーションではなく、App Studio インスタンスレベルで作成します。コネクタを作成したら、接続されたサービスとアプリケーションに応じて、アプリケーションのさまざまな部分でコネクタを使用できます。

コネクタを使用して他の サービスに接続するアプリケーションの機能の例を次に示します。

- ほとんどのアプリケーションで使用される最も一般的なユースケースは、Amazon Redshift、Amazon DynamoDB、Amazon Aurora などのデータサービスに接続して、アプリケーションで使用される AWS データを保存してアクセスすることです。
- 受信などのイメージのアップロードと表示を許可するアプリケーションは、Amazon S3 を使用してイメージファイルを保存し、アクセスできます。
- テキストサマリアプリは、Amazon Bedrock にテキスト入力を送信し、返された概要を表示できません。

Note

コネクタを作成するには、App Studio に管理者ロールが必要です。コネクタを作成するときは、使用するリソースまたは API コールに関する適切な認証情報と情報を含める必要があります。

アプリケーションのデータモデルの設定

アプリケーションのデータは、アプリケーションを強化する情報です。App Studio では、保存して操作するさまざまなタイプのデータを表すエンティティを作成して使用します。たとえば、顧客との会議の追跡アプリケーションには、顧客との会議、アジェンダ、参加者を表す 3 つのエンティティがあるとします。

エンティティには、保存されるデータを記述する整数や文字列などの型を持つフィールドが含まれます。エンティティを使用してデータモデルを定義しますが、エンティティを Amazon Redshift や Amazon DynamoDB などの外部データストレージサービスに接続してデータを保存する必要があります。エンティティは、App Studio アプリケーションと外部サービスのデータとの間の仲介と考えることができます。

データアクションを使用して、コンポーネントやオートメーションからアプリケーション内のデータを操作できます。使用する最も一般的な 2 つのデータアクションは、`getAll` アクションと `getByID` アクションです。たとえば、アプリケーションは `getAll` データアクションを使用して

テーブルにデータを入力し、詳細コンポーネントに特定のデータ入力に関する詳細情報を入力するgetByIDアクションを使用できます。

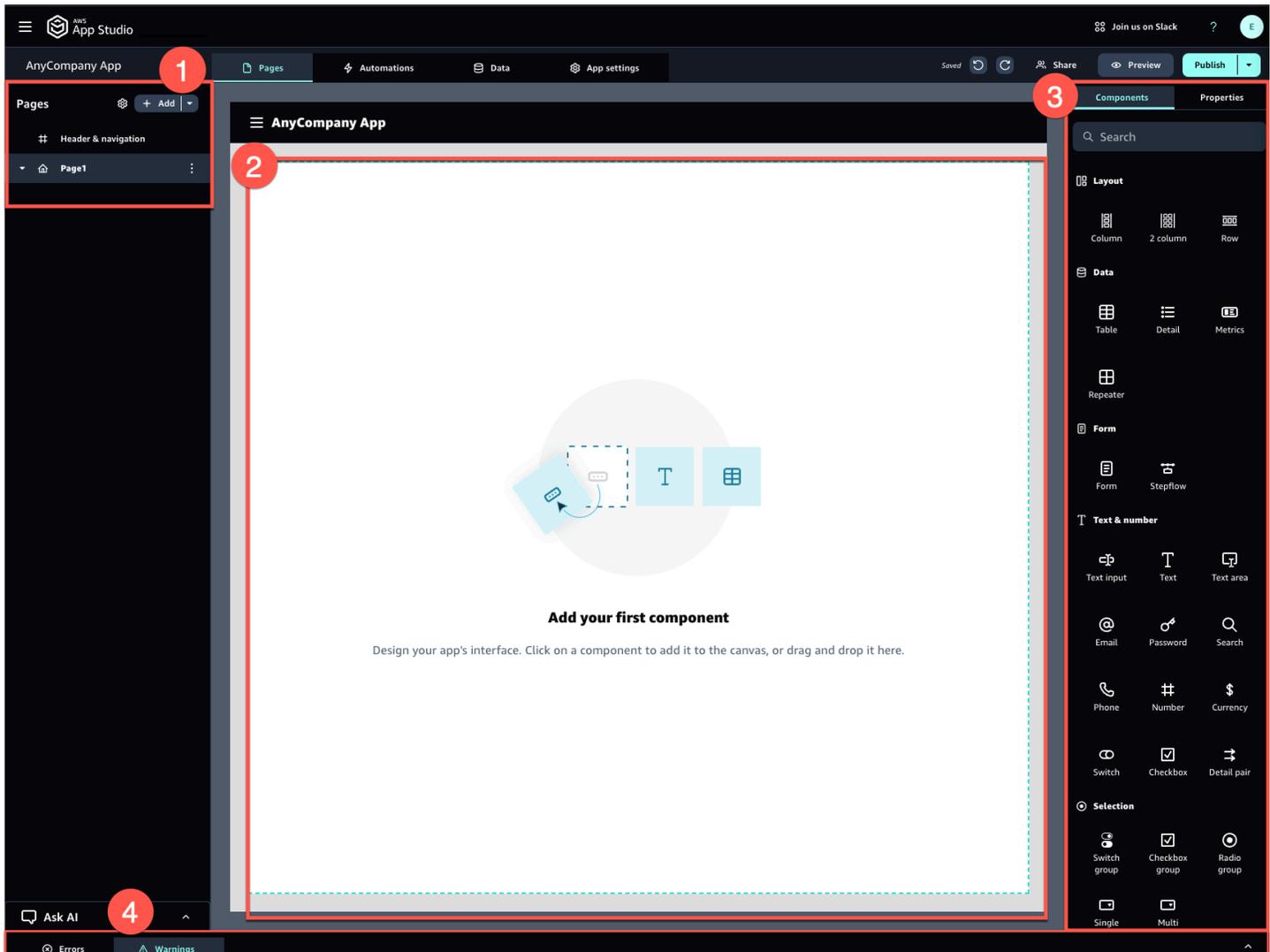
また、エンティティにサンプルデータを追加して、外部サービスを呼び出すことなく、アプリケーションをより簡単にテストすることもできます。

アプリケーションの UI の構築

App Studio では、ページとコンポーネントを使用してアプリケーションの UI を構築します。ページはアプリケーションの個々の画面であり、コンポーネントのコンテナです。コンポーネントは、アプリケーションの UI の構成要素です。テーブル、フォーム、イメージビューワー、ボタンなど、さまざまなタイプのコンポーネントがあります。

次の図は、アプリケーションスタジオのページタブを示しています。ここでは、アプリケーション内のページとコンポーネントを追加または設定します。次のキーエリアが強調表示され、番号が付けられます。

1. 左側のページパネル。ここでは、ページ、アプリケーションヘッダー、ナビゲーション設定を管理します。アプリケーションのすべてのページとコンポーネントを表示できます。
2. 現在のページのコンポーネントを表示するキャンバス。キャンバス内のコンポーネントを選択して、プロパティを設定できます。
3. 右側のコンポーネントまたはプロパティパネル。何も選択しない場合、コンポーネントパネルが表示され、ページに追加できるコンポーネントのリストが表示されます。ページまたはコンポーネントを選択すると、プロパティパネルが表示され、ページまたはコンポーネントを設定します。
4. 下部のエラーと警告パネル。これらのパネルには、アプリケーションのエラーや警告が表示されます。これは、最も一般的に設定の問題によるものです。パネルを選択して展開し、メッセージを表示できます。



たとえば、ユーザーが情報を入力する必要があるアプリケーションには、次のページとコンポーネントがあります。

- ユーザーが情報の入力と送信に使用するフォームコンポーネントを含む入力ページ。
- 各入力に関する情報を含むテーブルコンポーネントを含むリストビューページ。
- 各入力に関する詳細情報を含む詳細コンポーネントを含む詳細ビューページ。

コンポーネントには、フィールドが定義されたフォームなどの静的な情報またはデータを含めることができます。また、Amazon S3 バケットからイメージを取得してユーザーに表示するイメージビューワーなどのオートメーションを使用して、動的情報を含めることもできます。

ページパラメータの概念を理解することが重要です。ページパラメータを使用して、あるページから別のページに情報を送信します。ページパラメータのユースケースの一般的な例は、検索とフィルタ

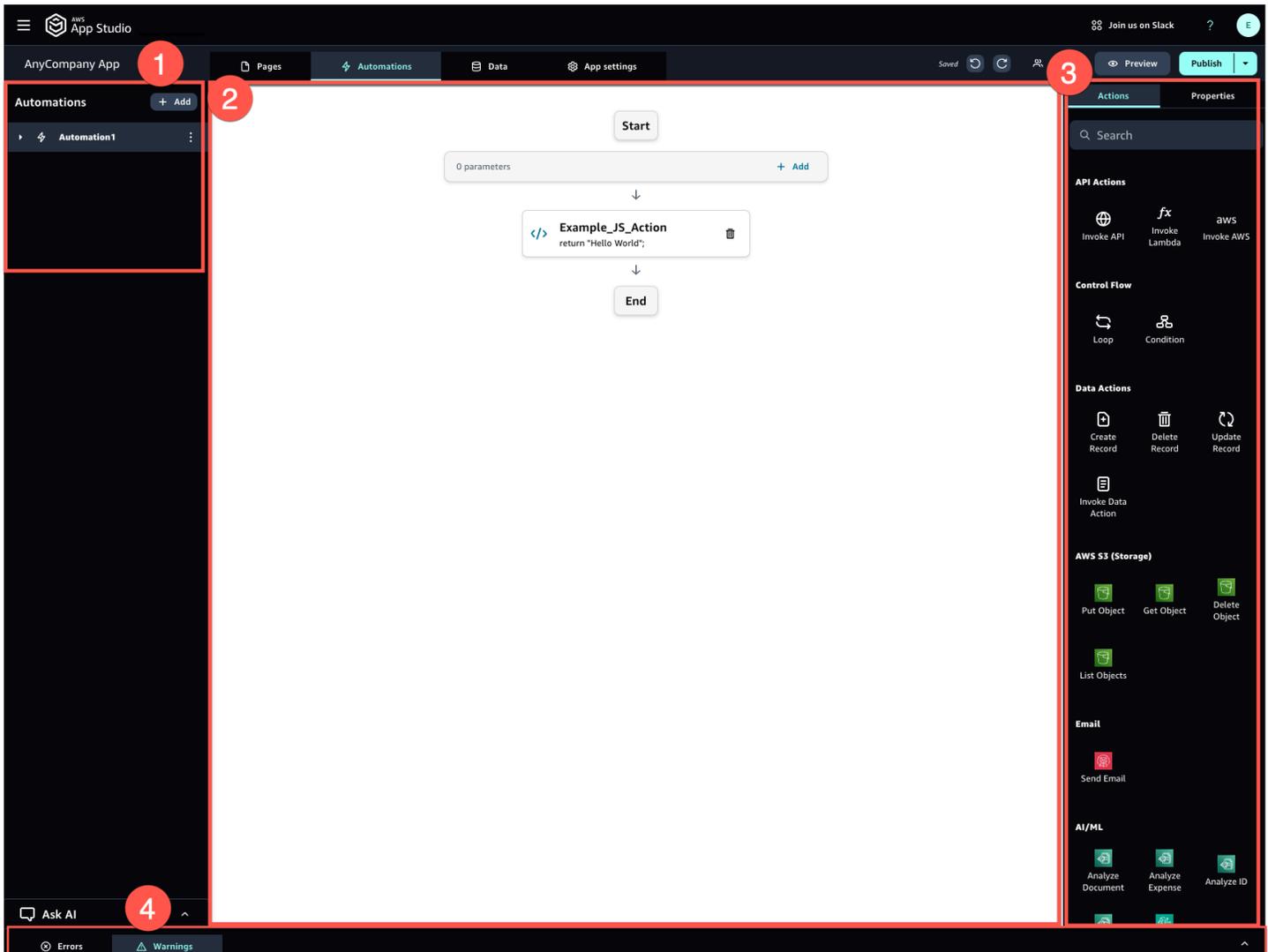
リングです。あるページの検索用語は、別のページのフィルタリングする項目のリストまたはテーブルに送信されます。もう1つのユースケースの例は、項目識別子が詳細なビューワーページに送信される項目の詳細の表示です。

アプリケーションのロジックまたは動作の実装

アプリケーションのロジックまたは動作は、アプリケーションの機能と考えることができます。ユーザーがボタンを選択したとき、情報を送信したとき、新しいページに移動したとき、または他の方法で操作したときに何が起こるかを定義できます。App Studio では、自動化とアクションを使用してアプリケーションのロジックを定義します。自動化はアクションのコンテナであり、自動化の機能の構成要素です。

次の図は、アプリケーションスタジオの Automations タブを示しています。ここでは、アプリケーションでオートメーションとそのアクションを追加または設定します。次のキーエリアが強調表示され、番号が付けられます。

- 左側の Automations パネル。ここで自動化を管理します。アプリケーションのすべてのオートメーションとアクションを表示できます。
- 現在のオートメーションを表示するキャンバス。設定されたオートメーションパラメータ (このセクションで後述) とアクションが表示されます。キャンバス内のコンポーネントを選択して、プロパティを設定できます。
- 右側の Actions and Properties パネル。何も選択しない場合、アクションパネルが表示されます。オートメーションに追加できるアクションのリストが表示されます。オートメーションを選択すると、オートメーションの入出力などのプロパティを表示および設定できます。アクションを選択すると、アクションのプロパティを表示および設定できます。
- 下部のエラーと警告パネル。このパネルには、アプリケーションのエラーまたは警告 (最も一般的には設定の問題による) が表示されます。パネルを選択して展開し、メッセージを表示できます。



自動化は、シンプル (数値の追加や結果の返すなど) でも、より強力 (別のサービスへの入力の送信や結果の返すなど) でもかまいません。オートメーションの主なコンポーネントは次のとおりです。

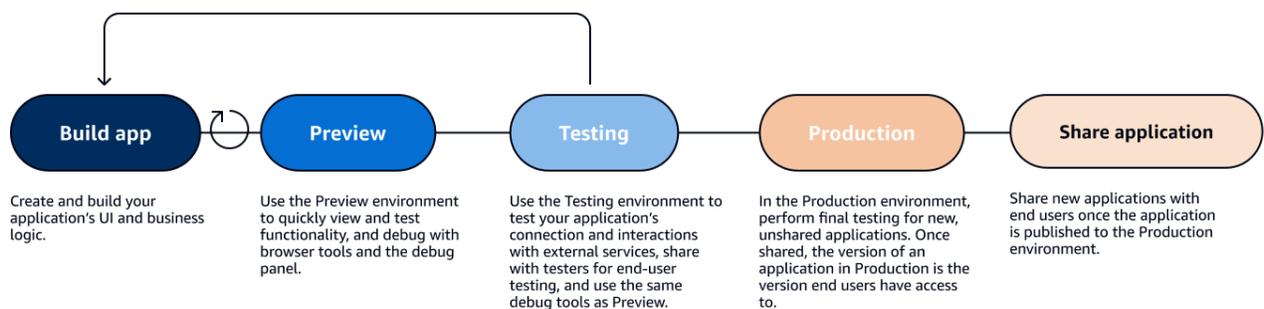
- オートメーションが実行されるタイミングを定義するトリガー。たとえば、ユーザーが UI のボタンを押す場合です。
- オートメーション入力。オートメーションに情報を送信します。オートメーションパラメータを使用してオートメーション入力を定義します。例えば、Amazon Bedrock を使用してテキストの概要をユーザーに返す場合は、自動化パラメータとして要約されるようにテキストを設定します。
- オートメーションの機能の構成要素であるアクション。各アクションは自動化のステップと考えることができます。アクションでは、APIs呼び出し、カスタム JavaScript の呼び出し、データレコードの作成、その他の関数の実行を行うことができます。アクションをループまたは条件にグループ化して、機能をさらにカスタマイズすることもできます。アクションを使用して他のオートメーションを呼び出すこともできます。

- オートメーション出力。コンポーネントや他のオートメーションでも使用できます。たとえば、オートメーション出力は、テキストコンポーネントに表示されるテキスト、イメージビューワーコンポーネントに表示されるイメージ、または別のオートメーションへの入力です。

アプリケーションの開発ライフサイクル

アプリケーションの開発ライフサイクルには、構築、テスト、公開のステージが含まれます。アプリケーションの作成時と反復時に、これらのステージ間を反復する可能性が高いため、サイクルと呼ばれます。

次の図は、App Studio でのアプリケーション開発ライフサイクルの簡略化されたタイムラインを示しています。



App Studio には、アプリケーションのライフサイクルをサポートするさまざまなツールが用意されています。これらのツールには、前の図に示す 3 つの異なる環境が含まれます。

- プレビュー環境では、アプリケーションをプレビューしてエンドユーザーがどのように見えるかを確認し、特定の機能をテストできます。プレビュー環境を使用すると、公開することなく、アプリケーションでテストと反復をすばやく実行できます。プレビュー環境のアプリケーションは、外部サービスと通信したり、データを転送したりしません。つまり、プレビュー環境で外部サービスに依存するインタラクションや機能をテストすることはできません。
- テスト環境。アプリケーションの接続と外部サービスとのインタラクションをテストできます。また、テスト環境に公開されたバージョンをテスターのグループと共有することで、エンドユーザーテストを実行することもできます。
- 本番稼働環境では、新しいアプリケーションをエンドユーザーと共有する前に最終テストを実行できます。アプリケーションが共有されると、本番環境に公開されるアプリケーションのバージョンは、エンドユーザーが表示および使用するバージョンになります。

詳細情報

App Studio でのアプリケーション開発の仕組みの基本を理解できたので、独自のアプリケーションの構築を開始するか、概念とリソースの詳細について深く掘り下げることができます。

構築を開始するには、入門チュートリアルのいずれかを試すことをお勧めします。

- AI ビルダーアシスタントを使用してアプリの構築を開始する方法については、[チュートリアル: AI を使用してアプリを生成する](#)「」を参照してください。
- [チュートリアル: 空のアプリから構築を開始する](#)「」に従って、基本を学習しながらアプリケーションをゼロから構築する方法を学習します。

このトピックで説明されているリソースまたは概念の詳細については、以下のトピックを参照してください。

- [コネクタを使用して App Studio を他の サービスに接続する](#)
- [エンティティとデータアクション: アプリのデータモデルを設定する](#)
- [ページとコンポーネント: アプリケーションのユーザーインターフェイスを構築する](#)
- [自動化とアクション: アプリのビジネスロジックを定義する](#)
- [アプリケーションのプレビュー、公開、共有](#)

AWS App Studio のセットアップとサインイン

AWS App Studio のセットアップは、ロールによって異なります。

- AWS または組織の管理者としての初回セットアップ: 管理者として App Studio を初めてセットアップするには、AWS アカウントがない場合はアカウントを作成し、App Studio インスタンスを作成し、IAM Identity Center グループを使用してユーザーアクセスを設定します。インスタンスが作成されると、App Studio の管理者ロールを持つすべてのユーザーが、他のサービス (データソースなど) を App Studio インスタンスに接続するようにコネクタを設定するなど、タスクをさらに設定できます。初回セットアップの詳細については、「」を参照してください[App Studio インスタンスを初めて作成して設定する](#)。
- ビルダーの開始方法: ビルダーとして App Studio に参加する招待を受け取ったら、招待を承諾し、パスワードを指定して IAM Identity Center ユーザー認証情報をアクティブ化する必要があります。その後、App Studio にサインインしてアプリケーションの構築を開始できます。招待の承諾と App Studio インスタンスへの参加については、「」を参照してください[App Studio への招待の承諾](#)。

App Studio インスタンスを初めて作成して設定する

AWS アカウントにサインアップする

App Studio をセットアップするには、AWS アカウントが必要です。App Studio を使用するには 1 つの AWS アカウントのみが必要です。アクセスは IAM Identity Center で管理されるため、ビルダーと管理者が App Studio AWS を使用するのに AWS アカウントは必要ありません。

を作成するには 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](#)

[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 リソースを管理するための管理ユーザーを作成する

AWS アカウントを初めて作成するときは、アカウントのすべての AWS リソースへの完全なアクセス権を持つデフォルトの認証情報セットから始めます。この ID は[AWS アカウントのルートユーザー](#)と呼ばれます。App Studio で使用する AWS ロールとリソースを作成するには、AWS アカウントのルートユーザーを使用しないことを強くお勧めします。代わりに、管理ユーザーを作成して使用することをお勧めします。

以下のトピックを使用して、App Studio で使用する AWS ロールとリソースを管理するための管理ユーザーを作成します。

- 単一のスタンドアロン AWS アカウントについては、[IAM ユーザーガイドの「最初の IAM ユーザーを作成する」](#)を参照してください。任意のユーザー名を指定できますが、アクセス AdministratorAccess 許可ポリシーが必要です。
- で管理される複数の AWS アカウントについては AWS Organizations、[「ユーザーガイド」の「IAM Identity Center 管理ユーザーの AWS アカウントアクセスを設定する」](#)を参照してください。AWS IAM アイデンティティセンター

で App Studio インスタンスを作成する AWS マネジメントコンソール

App Studio を使用するには、の App Studio ランディングページからインスタンスを作成する必要があります AWS マネジメントコンソール。App Studio インスタンスの作成に使用できる方法は 2 つあります。

1. 簡単な作成: この簡略化された方法では、設定の一部として App Studio にアクセスして使用するユーザーを 1 人だけ設定します。組織またはチームの App Studio を評価する場合、または App Studio を自分で使用する予定のみの場合は、この方法を使用する必要があります。セットアップ後に App Studio にユーザーまたはグループを追加できます。IAM Identity Center の組織インスタンスがある場合、このメソッドを使用することはできません。
2. 標準作成: この方法では、ユーザーまたはグループを追加し、セットアップの一環として App Studio でロールを割り当てます。設定時に複数のユーザーを App Studio に追加する場合は、この方法を使用する必要があります。

Note

App Studio のインスタンスは、すべての AWS リージョンで 1 つだけ作成できます。既存のインスタンスがある場合は、別のインスタンスを作成する前に削除する必要があります。詳細については、「[App Studio インスタンスの削除](#)」を参照してください。

Easy create

で AWS マネジメントコンソール App Studio インスタンスを簡単に作成するには

1. <https://console.aws.amazon.com/appstudio/> で App Studio コンソールを開きます。
2. App Studio インスタンスを作成する AWS リージョンに移動します。
3. [開始する] を選択します。
4. Easy create を選択し、Next を選択します。
5. App Studio をセットアップする次のステップは、IAM Identity Center アカウントインスタンスがあるかどうかによって決まります。IAM Identity Center インスタンスの詳細については、IAM Identity Center AWS ユーザーガイドの「[IAM Identity Center の組織インスタンスとアカウントインスタンスの管理](#)」を参照してください。
 - IAM Identity Center のアカウントインスタンスがある場合:
 - a. アカウントアクセス許可で、App Studio を有効にするために必要なアクセス許可を確認します。アカウントに必要なアクセス許可がない場合、App Studio を有効にすることはできません。必要なアクセス許可をアカウントに追加するか、アカウントに切り替える必要があります。
 - b. ユーザーの追加で、App Studio にアクセスする IAM Identity Center アカウントインスタンスでユーザーの E メールアドレスを検索して選択します。このユーザーは、App Studio インスタンスで管理者ロールを持ちます。App Studio へのアクセスを許可するユーザーが表示されない場合は、IAM Identity Center インスタンスに追加する必要がある場合があります。
 - IAM Identity Center のアカウントインスタンスがない場合:

Note

App Studio をセットアップすると、セットアッププロセス中に設定したユーザーを使用して IAM Identity Center アカウントインスタンスが自動的に作成されます。セットアップが完了したら、<https://console.aws.amazon.com/>

[singlesignon/](#) の IAM Identity Center コンソールでユーザーとグループを追加または管理できます。

- a. アカウントのアクセス許可で、App Studio を有効にするために必要なアクセス許可を確認します。アカウントに必要なアクセス許可がない場合、App Studio を有効にすることはできません。必要なアクセス許可をアカウントに追加するか、アカウントに切り替える必要があります。
 - b. ユーザーを追加する で、App Studio にアクセスするユーザーの E メールアドレス、名、姓、ユーザー名を指定します。このユーザーは、App Studio インスタンスで管理者ロールを持ちます。
6. サービスアクセスとロールで、必要なアクセス許可をサービスに提供するように App Studio をセットアップするときに自動的に作成されるサービスロールとサービスにリンクされたロールを確認します。アクセス許可を表示を選択してサービスロールに付与された正確なアクセス許可を表示するか、ポリシーを表示を選択してサービスにリンクされたロールにアタッチされたアクセス許可ポリシーを確認します。
 7. 確認 で、チェックボックスを選択してステートメントを確認します。
 8. セットアップを選択してインスタンスを作成します。

 Note

セットアップ後に App Studio インスタンスにユーザーまたはグループを追加するには、IAM Identity Center インスタンスに追加する必要があります。

Standard create

標準メソッド AWS マネジメントコンソール を使用して で App Studio インスタンスを作成するには

1. <https://console.aws.amazon.com/appstudio/> で App Studio コンソールを開きます。
2. App Studio インスタンスを作成する AWS リージョンに移動します。
3. [開始する] を選択します。
4. Standard create を選択し、Next を選択します。
5. App Studio をセットアップするステップは、IAM Identity Center インスタンスがあるかどうか、およびインスタンスのタイプによって決まります。IAM Identity Center インスタンスの

詳細については、IAM Identity Center AWS ユーザーガイドの「[IAM Identity Center の組織インスタンスとアカウントインスタンスの管理](#)」を参照してください。

- IAM Identity Center の組織インスタンスがある場合:
 - シングルサインオンを使用して App Studio へのアクセスを設定するで、既存の IAM Identity Center グループを選択して、App Studio へのアクセスを許可します。App Studio グループは、指定された設定に基づいて作成されます。Admin グループに追加されたグループのメンバーは Admin ロールを持ち、Builder グループに追加されたグループのメンバーは App Studio の Builder ロールを持ちます。ロールは次のように定義されます。
 - 管理者は、App Studio 内のユーザーとグループの管理、コネクタの追加と管理、ビルダーによって作成されたアプリケーションの管理を行うことができます。さらに、管理者ロールを持つユーザーには、ビルダーロールに含まれるすべてのアクセス許可があります。
 - ビルダーはアプリケーションを作成および構築できます。ビルダーは、ユーザーまたはグループを管理したり、コネクタインスタンスを追加または編集したり、他のビルダーのアプリケーションを管理したりすることはできません。
- IAM Identity Center インスタンスのアカウントインスタンスがある場合:
 - a. アカウントのアクセス許可で、App Studio を有効にするために必要なアクセス許可を確認します。アカウントに必要なアクセス許可がない場合、App Studio を有効にすることはできません。必要なアクセス許可をアカウントに追加するか、アカウントに切り替える必要があります。
 - b. シングルサインオンで App Studio へのアクセスを設定するで、IAM Identity Center アカウントで、既存のアカウントインスタンスを使用するを選択します。
 - c. AWS リージョンで、IAM Identity Center アカウントインスタンスがある Region を選択します。
 - d. 既存の IAM Identity Center グループを選択して、App Studio へのアクセスを提供します。App Studio グループは、指定された設定に基づいて作成されます。管理者グループに追加されたグループのメンバーには管理者ロールがあり、ビルダーグループに追加されたグループのメンバーには App Studio のビルダーロールがあります。ロールは次のように定義されます。
 - 管理者は、App Studio 内のユーザーとグループの管理、コネクタの追加と管理、ビルダーによって作成されたアプリケーションの管理を行うことができます。さ

らに、管理者ロールを持つユーザーには、ビルダーロールに含まれるすべてのアクセス許可があります。

- ビルダーはアプリケーションを作成および構築できます。ビルダーは、ユーザーまたはグループを管理したり、コネクタインスタンスを追加または編集したり、他のビルダーのアプリケーションを管理したりすることはできません。
- IAM Identity Center インスタンスがない場合:

 Note

App Studio をセットアップすると、セットアッププロセス中に設定したグループを持つ IAM Identity Center アカウントインスタンスが自動的に作成されます。セットアップが完了したら、<https://console.aws.amazon.com/singlesignon/> の IAM Identity Center コンソールでユーザーとグループを追加または管理できます。

- a. アカウントのアクセス許可で、App Studio を有効にするために必要なアクセス許可を確認します。アカウントに必要なアクセス許可がない場合、App Studio を有効にすることはできません。必要なアクセス許可をアカウントに追加するか、アカウントに切り替える必要があります。
- b. シングルサインオンで App Studio へのアクセスを設定するで、IAM Identity Center アカウントで、アカウントインスタンスの作成を選択します。
- c. 「ユーザーとグループを作成して App Studio に追加する」で、名前を指定し、管理者グループとビルダーグループにユーザーを追加します。管理者グループに追加されたユーザーは App Studio の管理者ロールを持ち、ビルダーグループに追加されたユーザーはビルダーロールを持ちます。ロールは次のように定義されます。
 - 管理者は、App Studio 内のユーザーとグループの管理、コネクタの追加と管理、ビルダーによって作成されたアプリケーションの管理を行うことができます。さらに、管理者ロールを持つユーザーには、ビルダーロールに含まれるすべてのアクセス許可があります。
 - ビルダーはアプリケーションを作成および構築できます。ビルダーは、ユーザーまたはグループを管理したり、コネクタインスタンスを追加または編集したり、他のビルダーのアプリケーションを管理したりすることはできません。

⚠ Important

App Studio をセットアップし、セットアップ後に管理者アクセス権を持つには、管理者グループのユーザーとして自分自身を追加する必要があります。

6. サービスアクセスとロールで、必要なアクセス許可をサービスに提供するように App Studio をセットアップするときに自動的に作成されるサービスロールとサービスにリンクされたロールを確認します。アクセス許可を表示を選択してサービスロールに付与された正確なアクセス許可を表示するか、ポリシーを表示を選択してサービスにリンクされたロールにアタッチされたアクセス許可ポリシーを確認します。
7. 確認 で、チェックボックスを選択してステートメントを確認します。
8. セットアップを選択してインスタンスを作成します。

App Studio への招待の承諾

App Studio へのアクセスは、IAM Identity Center によって管理されます。つまり、App Studio を使用する各ユーザーは、IAM Identity Center でユーザーを設定し、管理者によって App Studio に追加されたグループに属している必要があります。管理者が IAM Identity Center への参加を招待すると、招待を受け入れてユーザー認証情報をアクティブ化するように求める E メールが届きます。アクティブ化されたら、これらの認証情報を使用して App Studio にサインインできます。

App Studio にアクセスするための IAM Identity Center への招待を受け入れるには

1. 招待メールを受け取ったら、手順に従ってパスワードを指定し、IAM Identity Center でユーザー認証情報をアクティブ化します。詳細については、[「IAM Identity Center への参加の招待の承諾」](#)を参照してください。
2. ユーザー認証情報をアクティブ化したら、それを使用して App Studio インスタンスにサインインします。

AWS App Studio の開始方法

以下の入門チュートリアルでは、App Studio で最初のアプリケーションを構築する方法について説明します。

- 推奨: 生成 AI を使用して作成するアプリを記述し、そのアプリとそのリソースを自動的に作成するには、「」を参照してください[チュートリアル: AI を使用してアプリを生成する](#)。
- 空のアプリケーションから構築を開始するには、「」を参照してください[チュートリアル: 空のアプリから構築を開始する](#)。

チュートリアル: AI を使用してアプリを生成する

AWS App Studio には、アプリケーション構築を高速化するために、サービス全体で生成 AI 機能が含まれています。このチュートリアルでは、自然言語を使用してアプリを記述することで、AI を使用してアプリを生成する方法について説明します。

AI を使用してアプリケーションを生成することは、アプリケーションのリソースの多くが自動的に作成されるため、構築を開始するのに最適な方法です。通常、空のアプリケーションから開始するよりも、既存のリソースを使用して生成されたアプリケーションから構築を開始する方がはるかに簡単です。

Note

ブログ記事「[Build enterprise-grade applications with natural language using AWS App Studio \(preview\)](#)」を参照してください。このブログ記事には、管理者関連のリソースの設定と設定に関する情報も含まれていますが、必要に応じてアプリケーションの構築に関する部分に進むことができます。

App Studio が AI でアプリを生成すると、説明したアプリに合わせた以下のリソースでアプリが作成されます。

- ページとコンポーネント: コンポーネントは、アプリケーションのユーザーインターフェイスの構成要素です。テーブル、フォーム、ボタンなどのビジュアル要素を表します。各コンポーネントには独自のプロパティセットがあり、特定の要件に合わせてコンポーネントをカスタマイズできます。ページはコンポーネントのコンテナです。

- **自動化:** 自動化を使用して、アプリケーションの動作を管理するロジックとワークフローを定義します。たとえば、オートメーションを使用して、データテーブル内の行を作成、更新、読み取り、または削除したり、Amazon S3 バケット内のオブジェクトとやり取りしたりできます。これらを使用して、データ検証、通知、他のシステムとの統合などのタスクを処理することもできます。
- **エンティティ:** データは、アプリケーションを強化する情報です。生成されたアプリケーションは、テーブルに似たエンティティを作成します。エンティティは、顧客、製品、注文など、保存して操作する必要があるさまざまなタイプのデータを表します。App Studio コネクタを使用して、これらのデータモデルを AWS サービスや外部 APIs などのさまざまなデータソースに接続できます。

目次

- [前提条件](#)
- [ステップ 1: アプリケーションを作成する](#)
- [ステップ 2: 新しいアプリケーションを調べる](#)
 - [ページとコンポーネントを調べる](#)
 - [オートメーションとアクションを調べる](#)
 - [エンティティを使用してデータを探索する](#)
- [ステップ 3: アプリケーションをプレビューする](#)
- [次の手順](#)

前提条件

開始する前に、以下の前提条件を確認して完了してください。

- AWS App Studio へのアクセス。詳細については、「[AWS App Studio のセットアップとサインイン](#)」を参照してください。
- オプション: を確認して[AWS App Studio の概念](#)、App Studio の重要な概念を理解します。

ステップ 1: アプリケーションを作成する

アプリを生成する最初のステップは、作成するアプリを App Studio の AI アシスタントに記述することです。生成されるアプリケーションを確認し、必要に応じて反復処理してから生成できます。

AI を使用してアプリを生成するには

1. App Studio にサインインします。
2. 左側のナビゲーションで、ビルダーハブを選択し、+ アプリの作成を選択します。
3. AI でアプリを生成するを選択します。
4. アプリ名 フィールドで、アプリの名前を指定します。
5. 「データソースの選択」ダイアログボックスで、「スキップ」を選択します。
6. 生成するアプリの定義を開始するには、テキストボックスに記述するか、サンプルプロンプトでカスタマイズを選択します。アプリを記述すると、App Studio はアプリの要件と詳細を生成して確認できるようにします。これには、ユースケース、ユーザーフロー、データモデルが含まれます。
7. テキストボックスを使用して、要件と詳細に満足するまで、必要に応じてアプリを反復処理します。
8. アプリを生成して構築を開始する準備ができたなら、アプリの生成を選択します。
9. オプションで、新しいアプリ内を移動する方法を詳しく説明した短い動画を表示できます。
10. アプリの編集 を選択して、アプリの開発環境に入ります。

ステップ 2: 新しいアプリケーションを調べる

開発環境には、次のリソースがあります。

- アプリケーションを表示または編集するために使用するキャンバス。キャンバスは、選択したリソースに応じて変わります。
- キャンバスの上部にあるナビゲーションタブ。タブについては、次のリストで説明します。
 - ページ: ページとコンポーネントを使用してアプリケーションの UI を設計する場所。
 - 自動化: 自動化でアクションを使用してアプリケーションのビジネスロジックを定義する場所。
 - データ: エンティティ、そのフィールド、サンプルデータ、データアクションを定義して、アプリケーションのデータモデルを定義する場所。
 - アプリ設定: エンドユーザーのページのロールベースの可視性を定義するために使用するアプリロールなど、アプリの設定を定義する場所。
- 表示しているタブに基づくリソースを含む左側のナビゲーションメニュー。
- Pages and Automations タブで、選択したリソースのリソースとプロパティを一覧表示する右側メニュー。

- ビルダーの下部に警告とエラーを表示するデバッグコンソール。生成されたアプリにエラーが存在する可能性があります。これは、Amazon Simple Email Service で E メールを送信するなどのアクションを実行するために設定されたコネクタを必要とするオートメーションが原因である可能性があります。
- AI チャットウィンドウで AI ビルダーアシスタントからコンテキストに応じたヘルプを取得します。

ページ、オートメーション、データタブを詳しく見てみましょう。

ページとコンポーネントを調べる

ページタブには、生成されたページとそのコンポーネントが表示されます。

各ページは、ユーザーが操作するアプリケーションのユーザーインターフェイス (UI) の画面を表します。これらのページには、必要なレイアウトと機能を作成するためのさまざまなコンポーネント (テーブル、フォーム、ボタンなど) があります。

左側のナビゲーションメニューを使用して、ページとそのコンポーネントを表示する時間を取ります。ページまたはコンポーネントを選択すると、右側のメニューでプロパティを選択できます。

オートメーションとアクションを調べる

Automations タブには、生成されたオートメーションとそのアクションが表示されます。

自動化は、データエントリの作成、表示、更新、削除、Eメールの送信、APIs や Lambda 関数の呼び出しなど、アプリのビジネスロジックを定義します。

左側のナビゲーションメニューを使用して、自動化を表示する時間を取ります。オートメーションを選択すると、右側の Properties メニューでそのプロパティを表示できます。オートメーションには、次のリソースが含まれています。

- 自動化は、アプリのビジネスロジックの構成要素である個々のアクションで構成されます。オートメーションのアクションは、左側のナビゲーションメニュー、または選択したオートメーションのキャンバスで表示できます。アクションを選択すると、右側の Properties メニューでそのプロパティを表示できます。
- オートメーションパラメータは、データをオートメーションに渡す方法です。パラメータはプレースホルダーとして機能し、自動化の実行時に実際の値に置き換えられます。これにより、毎回異なる入力で同じオートメーションを使用できます。

- 自動化出力は、自動化の結果を設定する場所です。デフォルトでは、オートメーションには出力がないため、オートメーションの結果をコンポーネントや他のオートメーションで使用するには、ここで定義する必要があります。

詳細については、「[オートメーションの概念](#)」を参照してください。

エンティティを使用してデータを探索する

データタブには、生成されたエンティティが表示されます。

エンティティは、データベース内のテーブルと同様に、アプリケーションのデータを保持するテーブルを表します。アプリケーションのユーザーインターフェイス (UI) とオートメーションをデータソースに直接接続する代わりに、最初にエンティティに接続します。エンティティは、実際のデータソースと App Studio アプリの間の仲介として機能します。これにより、データを 1 か所で管理してアクセスできます。

左側のナビゲーションメニューからエンティティを選択して、生成されたエンティティを表示します。以下の詳細を確認できます。

- 設定タブには、エンティティ名とそのフィールドが表示され、エンティティの列を表します。
- データアクションタブには、エンティティで生成されたデータアクションが表示されます。コンポーネントとオートメーションは、データアクションを使用してエンティティからデータを取得できます。
- サンプルデータタブには、開発環境でアプリをテストするために使用できるサンプルデータが表示されます (外部サービスと通信しません)。環境の詳細については、「[アプリケーション環境](#)」を参照してください。
- Connection タブには、エンティティが接続されている外部データソースに関する情報が表示されます。App Studio は、DynamoDB テーブルを使用するマネージドデータストレージソリューションを提供します。詳細については、「[AWS App Studio のマネージドデータエンティティ](#)」を参照してください。

ステップ 3: アプリケーションをプレビューする

App Studio でアプリケーションをプレビューして、ユーザーにどのように表示されるかを確認できます。また、この機能を使用してデバッグパネルでログを確認することで、その機能をテストすることもできます。

アプリケーションプレビュー環境は、ライブデータの表示や、データソースなどのコネクタを使用した外部リソースへの接続をサポートしていません。代わりに、サンプルデータとモック出力を使用して機能をテストできます。

テスト用にアプリケーションをプレビューするには

1. App Builder の右上隅で、プレビューを選択します。
2. アプリのページを操作します。

次の手順

最初のアプリを作成したら、次のステップをいくつか示します。

- イメージを含む別の入門チュートリアルについては、ブログ記事 [「Build enterprise-grade applications with natural language using AWS App Studio \(preview\)」](#) を参照してください。
- アプリは、コネクタを使用してデータを送受信したり、外部サービス (サービスとサードパーティー AWS サービスの両方) と通信したりします。コネクタの詳細と、アプリケーションを構築するようにコネクタを設定する方法を学ぶ必要があります。コネクタを管理するには、管理者ロールが必要です。詳細については [コネクタを使用して App Studio を他の サービスに接続する](#) を参照してください。
- アプリケーションをエンドユーザーにプレビュー、公開、最終的に共有する方法の詳細については、「」を参照してください [アプリケーションのプレビュー、公開、共有](#)。
- 実践的な経験のために生成したアプリを探索して更新し続けます。
- アプリケーションの構築の詳細については、「」を参照してください [Builder ドキュメント](#)。具体的には、以下のトピックを参考にしてください。
 - [オートメーションアクションのリファレンス](#)
 - [コンポーネントリファレンス](#)
 - [Amazon Simple Storage Service とコンポーネントおよびオートメーションの操作](#)
 - [セキュリティに関する考慮事項と緩和策](#)

チュートリアル: 空のアプリから構築を開始する

このチュートリアルでは、AWS App Studio を使用して内部の Customer Meeting Request アプリケーションを構築します。App Studio でアプリケーションを構築する方法について学び、実際の

ユースケースと実践的な例に焦点を当てます。また、データ構造、UI 設計、アプリケーションのデプロイの定義についても説明します。

Note

このチュートリアルでは、空のアプリケーションから始めて、アプリケーションをゼロから構築する方法について詳しく説明します。通常、作成するアプリの説明を提供することで、AI を使用してアプリとそのリソースを生成する方がはるかに迅速かつ簡単です。詳細については、「[チュートリアル: AI を使用してアプリを生成する](#)」を参照してください。

App Studio でアプリケーションを構築する方法を理解するための鍵は、次の 4 つの主要な概念と、それらがどのように連携するかを理解することです。コンポーネント、オートメーション、データ、コネクタです。

- **コンポーネント:** コンポーネントは、アプリケーションのユーザーインターフェイスの構成要素です。テーブル、フォーム、ボタンなどのビジュアル要素を表します。各コンポーネントには独自のプロパティセットがあり、特定の要件に合わせてカスタマイズできます。
- **自動化:** 自動化を使用すると、アプリケーションの動作を管理するロジックとワークフローを定義できます。オートメーションを使用して、データテーブル内の行を作成、更新、読み取り、または削除したり、Amazon S3 バケット内のオブジェクトとやり取りしたりできます。また、データ検証、通知、他のシステムとの統合などのタスクを処理するためにも使用できます。
- **データ:** データはアプリケーションを強化する情報です。App Studio では、エンティティと呼ばれるデータモデルを定義できます。エンティティは、カスタマーミーティングのリクエスト、アジェンダ、参加者など、保存して操作する必要があるさまざまなタイプのデータを表します。App Studio コネクタを使用して、これらのデータモデルを AWS サービスや外部 APIs などのさまざまなデータソースに接続できます。
- **コネクタ:** App Studio は、Aurora、DynamoDB、Amazon Redshift などの AWS サービスを含む、幅広いデータソースとの接続を提供します。データソースには、Salesforce などのサードパーティーサービス、または OpenAPI や汎用 API コネクタを使用するその他の多くのサービスも含まれます。App Studio コネクタを使用すると、これらのエンタープライズグレードのサービスや外部アプリケーションのデータと機能をアプリケーションに簡単に組み込むことができます。

チュートリアルを進める中で、コンポーネント、データ、自動化の主要な概念がどのように組み合わせられて内部の Customer Meeting Request アプリケーションを構築するかを説明します。

以下は、このチュートリアルで実行する内容を説明する大まかなステップです。

1. データから始める: 多くのアプリケーションはデータモデルから始まるため、このチュートリアルもデータから始めます。Customer Meeting Request アプリを構築するには、まず MeetingRequests エンティティを作成します。このエンティティは、顧客名、会議日、アジェンダ、参加者など、関連するすべての会議リクエスト情報を保存するためのデータ構造を表します。このデータモデルはアプリケーションの基盤として機能し、構築するさまざまなコンポーネントとオートメーションを強化します。
2. ユーザーインターフェイス (UI) を作成する: データモデルが整ったら、チュートリアルでユーザーインターフェイス (UI) の構築をガイドします。App Studio では、ページを追加し、コンポーネントを追加して UI を構築します。テーブル、詳細ビュー、カレンダーなどのコンポーネントを会議リクエストダッシュボードページに追加します。これらのコンポーネントは、MeetingRequests エンティティに保存されているデータを表示して操作するように設計されています。これにより、ユーザーはカスタマー会議を表示、管理、スケジュールできます。また、会議リクエストの作成ページを作成します。このページには、データを収集するためのフォームコンポーネントと、送信するためのボタンコンポーネントが含まれています。
3. 自動化を使用してビジネスロジックを追加する: アプリケーションの機能を強化するには、ユーザーインタラクションを有効にするように一部のコンポーネントを設定します。例としては、ページに移動したり、MeetingRequests エンティティに新しい会議リクエストレコードを作成したりすることが挙げられます。
4. 検証と式を強化: データの整合性と正確性を確保するために、検証ルールをフォームコンポーネントに追加します。これにより、新しい会議リクエストレコードを作成するときに、ユーザーが完全で有効な情報を提供できるようになります。また、式を使用してアプリケーション内のデータを参照および操作し、ユーザーインターフェイス全体で動的およびコンテキスト情報を表示できるようにします。
5. プレビューとテスト: アプリケーションをデプロイする前に、完全にプレビューしてテストする機会があります。これにより、コンポーネント、データ、オートメーションがすべてシームレスに連携していることを確認できます。これにより、ユーザーはスムーズで直感的なエクスペリエンスを得られます。
6. アプリケーションを公開する: 最後に、完了した内部 Customer Meeting Request アプリケーションをデプロイし、ユーザーがアクセスできるようにします。App Studio のローコードアプローチを活用すると、プログラミングに関する広範な専門知識を必要とせずに、組織の特定のニーズを満たすカスタムアプリケーションを構築できます。

目次

- [前提条件](#)
- [ステップ 1: アプリケーションを作成する](#)

- [ステップ 2: エンティティを作成してアプリのデータを定義する](#)
 - [マネージドエンティティを作成する](#)
 - [エンティティにフィールドを追加する](#)
- [ステップ 3: ユーザーインターフェイス \(UI\) とロジックを設計する](#)
 - [会議リクエストダッシュボードページを追加する](#)
 - [会議リクエストの作成ページを追加する](#)
- [ステップ 4: アプリケーションをプレビューする](#)
- [ステップ 5: アプリケーションをテスト環境に公開する](#)
- [次の手順](#)

前提条件

開始する前に、以下の前提条件を確認して完了してください。

- AWS App Studio へのアクセス。詳細については、「[AWS App Studio のセットアップとサインイン](#)」を参照してください。
- オプション: を確認して[AWS App Studio の概念](#)、App Studio の重要な概念を理解します。
- オプション: JavaScript 構文などの基本的なウェブ開発概念の理解。
- オプション: AWS サービスに精通していること。

ステップ 1: アプリケーションを作成する

1. App Studio にサインインします。
2. 左側のナビゲーションで、ビルダーハブを選択し、+ アプリの作成を選択します。
3. [最初から開始] を選択します。
4. アプリ名フィールドに、などのアプリの名前を入力します **Customer Meeting Requests**。
5. データソースまたはコネクタを選択するように求められた場合は、このチュートリアルの目的でスキップを選択します。
6. 次へ をクリックして先に進みます。
7. (オプション): App Studio でアプリケーションを構築する方法の概要については、ビデオチュートリアルをご覧ください。
8. 「アプリの編集」を選択すると、App Studio アプリビルダーに移動します。

ステップ 2: エンティティを作成してアプリのデータを定義する

エンティティは、データベース内のテーブルと同様に、アプリケーションのデータを保持するテーブルを表します。アプリケーションのユーザーインターフェイス (UI) とオートメーションがデータソースに直接接続する代わりに、最初にエンティティに接続します。エンティティは、実際のデータソースと App Studio アプリの間の仲介として機能し、データを管理およびアクセスするための単一の場所を提供します。

エンティティを作成するには 4 つの方法があります。このチュートリアルでは、App Studio マネージドエンティティを使用します。

マネージドエンティティを作成する

マネージドエンティティを作成すると、App Studio が管理する対応する DynamoDB テーブルも作成されます。App Studio アプリでエンティティが変更されると、DynamoDB テーブルが自動的に更新されます。このオプションを使用すると、サードパーティーのデータソースを手動で作成、管理、接続したり、エンティティフィールドからテーブル列へのマッピングを指定したりする必要はありません。

エンティティを作成するときは、プライマリキーフィールドを定義する必要があります。プライマリキーは、エンティティ内の各レコードまたは行の一意の識別子として機能します。これにより、各レコードをあいまいさなく簡単に識別して取得できます。プライマリキーは、次のプロパティで構成されます。

- **プライマリキー名:** エンティティのプライマリキーフィールドの名前。
- **プライマリキーデータ型:** プライマリキーフィールドのタイプ。App Studio では、サポートされているプライマリキータイプはテキストの場合は文字列、数値の場合は浮動小数点です。テキストプライマリキー (*meetingName* など) には String のタイプがあり、数値プライマリキー (*meetingId* など) には Float のタイプがあります。

プライマリキーは、データの整合性を強制し、データの重複を防ぎ、効率的なデータの取得とクエリを可能にするため、エンティティの重要なコンポーネントです。

マネージドエンティティを作成するには

1. 上部のバーメニューからデータを選択します。
2. + エンティティの作成を選択します。
3. App Studio マネージドエンティティの作成を選択します。

4. エンティティ名 フィールドで、エンティティの名前を指定します。このチュートリアルでは、**MeetingRequests** と入力します。
5. プライマリキーフィールドに、エンティティのプライマリキー列に渡すプライマリキー名ラベルを入力します。このチュートリアルでは、**requestID** と入力します。
6. プライマリキーのデータ型で、浮動小数点を選択します。
7. [エンティティの作成] を選択します。

エンティティにフィールドを追加する

フィールドごとに、アプリケーションユーザーに表示されるラベルである表示名を指定します。表示名にはスペースと特殊文字を含めることができますが、エンティティ内で一意である必要があります。表示名は、フィールドのわかりやすいラベルとして機能し、ユーザーがその目的を簡単に特定して理解するのに役立ちます。

次に、フィールドを参照するためにアプリケーションが内部で使用する一意の識別子であるシステム名を指定します。システム名は簡潔で、スペースや特殊文字は使用できません。システム名により、アプリケーションはフィールドのデータに変更を加えることができます。これは、アプリケーション内のフィールドの一意の参照ポイントとして機能します。

最後に、文字列 (テキスト)、ブール値 (true/false)、日付、10 進数、浮動小数点数、整数、DateTimeなど、フィールドに保存するデータの種類を最もよく表すデータ型を選択します。適切なデータ型を定義すると、データの整合性が確保され、フィールドの値の適切な処理と処理が可能になります。たとえば、会議リクエストに顧客名を保存する場合は、テキスト値に対応するStringデータ型を選択します。

MeetingRequests エンティティにフィールドを追加するには

- + 追加フィールドを選択して、次の 4 つのフィールドを追加します。
 - a. 顧客の名前を表すフィールドを次の情報とともに追加します。
 - 表示名: **Customer name**
 - システム名: **customerName**
 - データ型: **String**
 - b. 次の情報を含む会議の日付を表すフィールドを追加します。
 - 表示名: **Meeting date**
 - システム名: **meetingDate**

- データ型: **DateTime**
- c. 会議アジェンダを表すフィールドを以下の情報とともに追加します。
- 表示名: **Agenda**
 - システム名: **agenda**
 - データ型: **String**
- d. 次の情報を使用して、会議の参加者を表すフィールドを追加します。
- 表示名: **Attendees**
 - システム名: **attendees**
 - データ型: **String**

サンプルデータをエンティティに追加して、公開する前にアプリケーションをテストおよびプレビューできます。最大 500 行のモックデータを追加することで、実際のデータを信頼したり、外部サービスに接続したりすることなく、実際のシナリオをシミュレートし、アプリケーションがさまざまなタイプのデータを処理および表示する方法を調べることができます。これにより、開発プロセスの早い段階で問題や不整合を特定して解決できます。これにより、実際のデータを処理するときにアプリケーションが意図したとおりに機能します。

サンプルデータをエンティティに追加するには

1. バナーでサンプルデータタブを選択します。
2. サンプルデータをさらに生成を選択します。
3. [保存] を選択します。

必要に応じて、バナーで接続を選択して、コネクタと作成された DynamoDB テーブルの詳細を確認します。

ステップ 3: ユーザーインターフェイス (UI) とロジックを設計する

会議リクエストダッシュボードページを追加する

App Studio では、各ページは、ユーザーが操作するアプリケーションのユーザーインターフェイス (UI) の画面を表します。これらのページ内で、テーブル、フォーム、ボタンなどのさまざまなコンポーネントを追加して、必要なレイアウトと機能を作成できます。

新しく作成されたアプリケーションにはデフォルトのページがあるため、シンプルな会議リクエストダッシュボードページとして使用する新しいアプリケーションを追加する代わりに、そのアプリケーションの名前を変更します。

デフォルトページの名前を変更するには

1. 上部のバーナビゲーションメニューで、ページを選択します。
2. 左側のパネルで PagePage1 をダブルクリックし、名前を `MeetingRequestsDashboard` に変更して **MeetingRequestsDashboard** Enter キーを押します。

次に、会議リクエストの表示に使用されるテーブルコンポーネントをページに追加します。

会議リクエストダッシュボードページにテーブルコンポーネントを追加するには

1. 右側のコンポーネントパネルで、テーブルコンポーネントを見つけてキャンバスにドラッグします。
2. キャンバス内のテーブルを選択して選択します。
3. 右側のプロパティパネルで、次の設定を更新します。
 - a. 鉛筆アイコンを選択して、テーブルの名前を `meetingRequestsTable` に変更します。
 - b. ソースドロップダウンで、エンティティを選択します。
 - c. データアクションドロップダウンで、作成したエンティティ (**MeetingRequests**) を選択し、+ データアクションの追加を選択します。
4. プロンプトが表示されたら、`getAll` を選択します。

Note

`getAll` データアクションは、指定されたエンティティからすべてのレコード (行) を取得する特定のタイプのデータアクションです。たとえば、`getAll` データアクションをテーブルコンポーネントに関連付けると、テーブルは接続されたエンティティのすべてのデータを自動的に入力し、各レコードをテーブルの行として表示します。

会議リクエストの作成ページを追加する

次に、エンドユーザーが会議リクエストの作成に使用するフォームを含むページを作成します。また、MeetingRequestsエンティティにレコードを作成する送信ボタンを追加し、エンドユーザーをMeetingRequestsDashboardページに移動します。

会議リクエストの作成ページを追加するには

1. 上部のバナーで、ページを選択します。
2. 左側のパネルで + 追加 を選択します。
3. 右側のプロパティパネルで、鉛筆アイコンを選択し、ページの名前を に変更します **CreateMeetingRequest**。

ページが追加されたら、エンドユーザーが情報を入力してMeetingRequestsエンティティに会議リクエストを作成するために使用するフォームをページに追加します。App Studio は、既存のエンティティからフォームを生成する方法を提供します。これにより、エンティティのフィールドに基づいてフォームフィールドが自動的に入力され、フォーム入力を使用してエンティティにレコードを作成するための送信ボタンも生成されます。

会議リクエストの作成ページでエンティティからフォームを自動的に生成するには

1. 右側のコンポーネントメニューで、フォームコンポーネントを見つけてキャンバスにドラッグします。
2. フォームの生成を選択します。
3. ドロップダウンからエンティティを選択しますMeetingRequests。
4. [Generate] (生成) を選択します。
5. キャンバスの送信ボタンを選択して選択します。
6. 右側のプロパティパネルの「トリガー」セクションで + 追加を選択します。
7. 「ナビゲート」を選択します。
8. 右側のプロパティパネルで、アクション名を などのわかりやすい名前に変更します **Navigate to MeetingRequestsDashboard**。
9. ナビゲーションタイプをページに変更します。ドロップダウンに移動で、 を選択します **MeetingRequestsDashboard**。

会議リクエストの作成ページとフォームができたので、MeetingRequestsDashboardこのページからこのページに簡単に移動できるようにして、ダッシュボードを確認するエンドユーザーが会議リクエストを簡単に作成できるようにしたいと考えています。次の手順に従って、MeetingRequestsDashboardページに移動するボタンを作成しますCreateMeetingRequest。

から **MeetingRequestsDashboard**に移動するボタンを追加するには **CreateMeetingRequest**

1. 上部のバナーで、ページを選択します。
2. MeetingRequestsDashboard ページを選択します。
3. 右側のコンポーネントパネルで、ボタンコンポーネントを見つけてキャンバスにドラッグし、テーブルの上に置きます。
4. 新しく追加されたボタンを選択して選択します。
5. 右側のプロパティパネルで、次の設定を更新します。
 - a. 鉛筆アイコンを選択して、ボタンの名前を に変更します **createMeetingRequestButton**。
 - b. ボタンラベル: **Create Meeting Request**。これは、エンドユーザーに表示される名前です。
 - c. アイコンドロップダウンで + Plus を選択します。
 - d. エンドユーザーをMeetingRequestsDashboardページに移動するトリガーを作成します。
 1. Triggers セクションで + Add を選択します。
 2. アクションタイプで、ナビゲートを選択します。
 3. 先ほど作成したトリガーを選択して設定します。
 4. アクション名で、 などのわかりやすい名前を指定します **NavigateToCreateMeetingRequest**。
 5. 移動タイプのドロップダウンで、ページを選択します。
 6. ドロップダウンに移動で、CreateMeetingRequestページを選択します。

ステップ 4: アプリケーションをプレビューする

App Studio でアプリケーションをプレビューして、ユーザーにどのように表示されるかを確認できます。また、この機能を使用してデバッグパネルでログを確認することで、機能をテストすることもできます。

アプリケーションプレビュー環境は、ライブデータの表示をサポートしていません。また、データソースなどのコネクタを持つ外部リソースとの接続もサポートしていません。代わりに、サンプルデータとモック出力を使用して機能をテストできます。

テスト用にアプリケーションをプレビューするには

1. App Builder の右上隅で、プレビュー を選択します。
2. MeetingRequestsDashboard ページを操作し、テーブル、フォーム、ボタンをテストします。

ステップ 5: アプリケーションをテスト環境に公開する

アプリケーションの作成、設定、テストが完了したら、テスト環境に公開して最終テストを実行し、ユーザーと共有します。

アプリケーションをテスト環境に公開するには

1. App Builder の右上隅で、発行を選択します。
2. テスト環境のバージョンの説明を追加します。
3. SLA に関するチェックボックスを確認して選択します。
4. [開始] を選択します。発行には最大 15 分かかる場合があります。
5. (オプション) 準備ができたなら、共有を選択し、プロンプトに従って他のユーザーにアクセスを許可できます。

Note

アプリを共有するには、管理者がエンドユーザーグループを作成している必要があります。

テスト後、再度発行を選択してアプリケーションを本番環境に昇格させます。さまざまなアプリケーション環境の詳細については、「」を参照してください[アプリケーション環境](#)。

次の手順

最初のアプリを作成したら、次のステップをいくつか示します。

1. チュートリアルアプリの構築を続けます。データ、一部のページ、オートメーションが設定されたので、追加のページを追加したり、コンポーネントを追加したりして、アプリケーションの構築の詳細を確認したりできます。
2. アプリケーションの構築の詳細については、「」を参照してください[Builder ドキュメント](#)。具体的には、以下のトピックが役立つ場合があります。
 - [オートメーションアクションのリファレンス](#)
 - [コンポーネントリファレンス](#)
 - [Amazon Simple Storage Service とコンポーネントおよびオートメーションの操作](#)
 - [セキュリティに関する考慮事項と緩和策](#)

さらに、以下のトピックには、チュートリアルで説明されている概念に関する詳細が含まれています。

- [アプリケーションのプレビュー、公開、共有](#)
- [App Studio アプリでのエンティティの作成](#)

管理者向けドキュメント

以下のトピックには、App Studio でサードパーティーのサービス接続とアクセスを管理するユーザー、ユーザー、ロールに役立つ情報が含まれています。

トピック

- [App Studio でのアクセスとロールの管理](#)
- [コネクタを使用して App Studio を他の サービスに接続する](#)
- [App Studio インスタンスの削除](#)

App Studio でのアクセスとロールの管理

App Studio の管理者の責任の 1 つは、アクセス、ロール、アクセス許可を管理することです。以下のトピックでは、App Studio のロールに関する情報と、ユーザーの追加、ユーザーの削除、ロールの変更方法について説明します。

AWS App Studio へのアクセスは、IAM Identity Center グループを使用して管理されます。App Studio インスタンスにユーザーを追加するには、次のいずれかを行う必要があります。

- App Studio に追加された既存の IAM Identity Center グループに追加します。
- App Studio に追加されていない新規または既存の IAM Identity Center グループに追加し、App Studio に追加します。

ロールはグループに適用されるため、IAM Identity Center グループは、グループのメンバーに割り当てるアクセス権限 (またはロール) を表す必要があります。ユーザーとグループの管理に関する情報など、IAM Identity Center の詳細については、[IAM Identity Center ユーザーガイド](#)を参照してください。

ロールとアクセス許可

App Studio には 3 つのロールがあります。次のリストには、各ロールとその説明が含まれています。

- **管理者:** 管理者は、App Studio 内のユーザーとグループの管理、コネクタの追加と管理、ビルダーによって作成されたアプリケーションの管理を行うことができます。さらに、管理者ロールを持つユーザーには、ビルダーロールに含まれるすべてのアクセス許可があります。

- **ビルダー:** ビルダーはアプリケーションを作成および構築できます。ビルダーは、ユーザーまたはグループを管理したり、コネクタインスタンスを追加または編集したり、他のビルダーのアプリケーションを管理したりすることはできません。
- **アプリユーザー:** アプリユーザーは公開されたアプリにアクセスして使用できますが、App Studio インスタンスにアクセスしてアプリを構築したり、リソースを管理したりすることはできません。

App Studio では、ロールはグループに割り当てられるため、追加された IAM Identity Center グループの各メンバーには、グループに割り当てられたロールが割り当てられます。

グループの表示

App Studio インスタンスに追加されたグループを表示するには、次の手順を実行します。

Note

App Studio インスタンスでグループを表示するには、管理者である必要があります。

App Studio インスタンスに追加されたグループを表示するには

- ナビゲーションペインで、「管理」セクションの「ロール」を選択します。既存のグループと各グループに割り当てられたロールのリストを表示するページが表示されます。

グループの管理の詳細については、[ユーザーまたはグループの追加](#)「」、[グループロールの変更](#)「」、または「」を参照してください[App Studio からユーザーまたはグループを削除する](#)。

ユーザーまたはグループの追加

App Studio にユーザーを追加するには、IAM Identity Center グループに追加し、そのグループを App Studio に追加する必要があります。IAM Identity Center グループを追加してロールを割り当てることで、App Studio にユーザーを追加するには、次の手順を実行します。

Note

App Studio インスタンスにユーザーを追加するには、管理者である必要があります。

App Studio インスタンスにユーザーまたはグループを追加するには

1. App Studio インスタンスにユーザーを追加するには、App Studio に追加された既存の IAM アイデンティティセンターグループにユーザーを追加するか、新しい IAM アイデンティティセンターグループを作成し、新しいユーザーを追加して、新しいグループを App Studio に追加する必要があります。

IAM アイデンティティセンターのユーザーとグループの管理については、AWS IAM アイデンティティセンター「ユーザーガイド」の[「IAM アイデンティティセンターでの ID の管理」](#)を参照してください。

2. App Studio に既に追加されている既存の IAM アイデンティティセンターグループにユーザーを追加した場合、新しいユーザーは IAM アイデンティティセンターのアクセス許可の設定を完了した後、指定されたアクセス許可で App Studio にアクセスできます。新しい IAM Identity Center グループを作成した場合は、次の手順を実行してグループを App Studio に追加し、グループのメンバーのロールを指定します。
3. ナビゲーションペインで、「管理」セクションの「ロール」を選択します。
4. ロールページで、+グループの追加を選択します。これにより、グループの追加ダイアログボックスが開き、グループに関する情報を入力できます。
5. グループの追加ダイアログボックスに、次の情報を入力します。
 - ドロップダウンで既存の IAM Identity Center グループを選択します。
 - グループのロールを選択します。
 - 管理者: 管理者は、App Studio 内のユーザーとグループの管理、コネクタの追加と管理、ビルダーによって作成されたアプリケーションの管理を行うことができます。さらに、管理者ロールを持つユーザーには、ビルダーロールに含まれるすべてのアクセス許可があります。
 - ビルダー: ビルダーはアプリケーションを作成および構築できます。ビルダーは、ユーザーまたはグループを管理したり、コネクタインスタンスを追加または編集したり、他のビルダーのアプリケーションを管理したりすることはできません。
 - アプリユーザー: アプリユーザーは公開されたアプリにアクセスして使用できますが、App Studio インスタンスにアクセスしてアプリを構築したり、リソースを管理したりすることはできません。
6. 割り当てを選択して、グループを App Studio に追加し、メンバーに設定されたロールを提供します。

グループロールの変更

App Studio のグループに割り当てられたロールを変更するには、次の手順に従います。グループのロールを変更すると、そのグループ内のすべてのメンバーのロールが変更されます。

Note

App Studio でグループのロールを変更するには、管理者である必要があります。

グループのロールを変更するには

1. ナビゲーションペインで、「管理」セクションの「ロール」を選択します。既存のグループと各グループに割り当てられたロールのリストを表示するページが表示されます。
2. 省略記号アイコン (...) を選択し、ロールの変更を選択します。
3. ロールの変更ダイアログボックスで、グループの新しいロールを選択します。
 - 管理者: 管理者は、App Studio 内のユーザーとグループの管理、コネクタの追加と管理、ビルダーによって作成されたアプリケーションの管理を行うことができます。さらに、管理者ロールを持つユーザーには、ビルダーロールに含まれるすべてのアクセス許可があります。
 - ビルダー: ビルダーはアプリケーションを作成および構築できます。ビルダーは、ユーザーまたはグループを管理したり、コネクタインスタンスを追加または編集したり、他のビルダーのアプリケーションを管理したりすることはできません。
 - アプリユーザー: アプリユーザーは公開されたアプリにアクセスして使用できますが、App Studio インスタンスにアクセスしてアプリを構築したり、リソースを管理したりすることはできません。
4. グループロールの変更を選択します。

App Studio からユーザーまたはグループを削除する

App Studio から IAM Identity Center グループを削除することはできません。次の手順を実行すると、代わりにグループのロールが App User にダウングレードされます。グループのメンバーは、公開された App Studio アプリに引き続きアクセスできます。

App Studio とそのアプリケーションへのすべてのアクセスを削除するには、AWS IAM アイデンティティセンター コンソールで IAM Identity Center グループまたはユーザーを削除する必要があります。IAM アイデンティティセンターのユーザーとグループの管理については、AWS IAM アイデン

アイデンティティセンター「ユーザーガイド」の「IAM アイデンティティセンターでの ID の管理」を参照してください。

Note

App Studio でのグループのアクセスをダウングレードするには、管理者である必要があります。

グループを削除するには

1. ナビゲーションペインで、「管理」セクションの「ロール」を選択します。既存のグループと各グループに割り当てられたロールのリストを表示するページが表示されます。
2. 省略記号アイコン (...) を選択し、ロールの取り消しを選択します。
3. ロールの取り消しダイアログボックスで、Revoke を選択してグループのロールを App User にダウングレードします。

コネクタを使用して App Studio を他の サービスに接続する

コネクタは、App Studio と、AWS Lambda や Amazon Redshift などの他の AWS サービス、またはサードパーティーのサービス間の接続です。コネクタを作成して設定すると、ビルダーはそのコネクタと、アプリケーションで App Studio に接続するリソースを使用できます。

管理者ロールを持つユーザーのみがコネクタを作成、管理、または削除できます。

トピック

- [AWS サービスに接続する](#)
- [サードパーティーサービスに接続する](#)
- [コネクタの表示、編集、削除](#)

AWS サービスに接続する

トピック

- [Amazon Redshift に接続する](#)
- [Amazon DynamoDB に接続する](#)

- [に接続する AWS Lambda](#)
- [Amazon Simple Storage Service \(Amazon S3\) に接続する](#)
- [Amazon Aurora に接続する](#)
- [Amazon Bedrock に接続する](#)
- [Amazon Simple Email Service に接続する](#)
- [その他の AWS サービスコネクタを使用して AWS サービスに接続する](#)
- [CMKs](#)

Amazon Redshift に接続する

App Studio を Amazon Redshift に接続して、ビルダーがアプリケーションで Amazon Redshift リソースにアクセスして使用できるようにするには、次の手順を実行する必要があります。

1. [ステップ 1: Amazon Redshift リソースを作成して設定する](#)
2. [ステップ 2: 適切な Amazon Redshift アクセス許可を持つ IAM ポリシーとロールを作成する](#)
3. [ステップ 3: Amazon Redshift コネクタを作成する](#)

ステップ 1: Amazon Redshift リソースを作成して設定する

App Studio で使用する Amazon Redshift リソースを作成および設定するには、次の手順に従います。

App Studio で使用する Amazon Redshift を設定するには

1. にサインイン AWS マネジメントコンソール し、<https://console.aws.amazon.com/redshiftv2/> で Amazon Redshift コンソールを開きます。

で作成した管理ユーザーを使用することをお勧めします [AWS リソースを管理するための管理ユーザーを作成する](#)。

2. Redshift Serverless データウェアハウスまたはプロビジョニングされたクラスターを作成します。詳細については、「[Amazon Redshift ユーザーガイド](#)」の「[Redshift Serverless を使用したデータウェアハウスの作成](#)」または「[クラスターの作成](#)」を参照してください。
3. プロビジョニングが完了したら、クエリデータを選択してクエリエディタを開きます。データベースに接続します。
4. 次の設定を変更します。

1. 分離されたセッションの切り替えを に設定しますOFF。これは、実行中の App Studio アプリケーションからなど、他のユーザーが行ったデータ変更を確認できるようにするために必要です。
2. 「歯車」アイコンを選択します。[Account settings] (アカウント設定) を選択します。への最大同時接続数を増やします10。これは、Amazon Redshift データベースに接続できるクエリエディタセッションの数の制限です。App Studio アプリケーションなどの他のクライアントには適用されません。
5. public スキーマの下にデータテーブルを作成します。これらのテーブルに初期データINSERTを入力します。
6. クエリエディタで次のコマンドを実行します。

次のコマンドは、データベースユーザーを作成し、App Studio で使用される *AppBuilderDataAccessRole* という名前の IAM ロールに接続します。IAM ロールは後のステップで作成します。ここでの名前は、そのロールに指定された名前と一致する必要があります。

```
CREATE USER "IAMR:AppBuilderDataAccessRole" WITH PASSWORD DISABLE;
```

次のコマンドは、すべてのテーブルに対するすべてのアクセス許可を App Studio に付与します。

Note

ベストプラクティスとして、ここでのアクセス許可の範囲を、適切なテーブルに必要な最小限のアクセス許可に絞り込む必要があります。GRANT コマンドの詳細については、「Amazon Redshift データベースデベロッパーガイド」の「[GRANT](#)」を参照してください。

```
GRANT ALL ON ALL TABLES IN SCHEMA public to "IAMR:AppBuilderDataAccessRole";
```

ステップ 2: 適切な Amazon Redshift アクセス許可を持つ IAM ポリシーとロールを作成する

App Studio で Amazon Redshift リソースを使用するには、管理者は IAM ポリシーとロールを作成して、リソースへのアクセス許可を App Studio に付与する必要があります。IAM ポリシーは、ビル

ダーが使用できるデータの範囲と、作成、読み取り、更新、削除など、そのデータに対して呼び出すことができるオペレーションを制御します。IAM ポリシーは、App Studio で使用される IAM ロールにアタッチされます。

サービスおよびポリシーごとに少なくとも 1 つの IAM ロールを作成することをお勧めします。たとえば、ビルダーが Amazon Redshift の異なるテーブルにバックアップされた 2 つのアプリケーションを作成する場合、管理者は Amazon Redshift のテーブルごとに 1 つずつ、2 つの IAM ポリシーとロールを作成する必要があります。

ステップ 2a: 適切な Amazon Redshift アクセス許可を持つ IAM ポリシーを作成する

App Studio で作成して使用する IAM ポリシーには、アプリケーションがベストプラクティスに従うための適切なリソースに対する最小限のアクセス許可のみを含める必要があります。

適切な Amazon Redshift アクセス許可を持つ IAM ポリシーを作成するには

1. [IAM ポリシーを作成する権限を持つユーザーを使用して IAM コンソールにサインイン](#)します。で作成した管理ユーザーを使用することをお勧めします[AWS リソースを管理するための管理ユーザーを作成する](#)。
2. 左側のナビゲーションペインで、ポリシーを選択します。
3. [Create policy] (ポリシーの作成) を選択します。
4. [ポリシーエディタ] セクションで、[JSON] オプションを選択します。
5. JSON ポリシードキュメントに入力または貼り付けます。次のタブには、プロビジョニングされた Amazon Redshift とサーバーレス Amazon Redshift の両方のポリシーの例が含まれています。

Note

以下のポリシーは、ワイルドカード (*) を使用するすべての Amazon Redshift リソースに適用されます*。セキュリティのベストプラクティスとして、ワイルドカードを App Studio で使用するリソースの Amazon リソースネーム (ARN) に置き換える必要があります。

Provisioned

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ProvisionedRedshiftForAppStudio",
      "Effect": "Allow",
      "Action": [
        "redshift:DescribeClusters",
        "redshift:GetClusterCredentialsWithIAM",
        "redshift-data:ListDatabases",
        "redshift-data:ListTables",
        "redshift-data:DescribeTable",
        "redshift-data:DescribeStatement",
        "redshift-data:ExecuteStatement",
        "redshift-data:GetStatementResult"
      ],
      "Resource": "*"
    }
  ]
}
```

Serverless

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ServerlessRedshiftForAppStudio",
      "Effect": "Allow",
      "Action": [
        "redshift-serverless:ListNamespaces",
        "redshift-serverless:GetCredentials",
        "redshift-serverless:ListWorkgroups",

```

```
        "redshift-data:ListDatabases",
        "redshift-data:ListTables",
        "redshift-data:DescribeTable",
        "redshift-data:DescribeStatement",
        "redshift-data:ExecuteStatement",
        "redshift-data:GetStatementResult"
    ],
    "Resource": "*"
}
]
```

6. [次へ] を選択します。
7. 確認と作成ページで、**RedshiftServerlessForAppStudio**や などのポリシー名**RedshiftProvisionedForAppStudio**、および説明 (オプション) を指定します。
8. [ポリシーの作成]を選択し、ポリシーを作成します。

ステップ 2b: App Studio に Amazon Redshift リソースへのアクセスを許可する IAM ロールを作成する

次に、以前に作成したポリシーを使用する IAM ロールを作成します。App Studio はこのポリシーを使用して、設定された Amazon Redshift リソースにアクセスします。

App Studio に Amazon Redshift リソースへのアクセスを許可する IAM ロールを作成するには

1. [IAM ロールを作成する権限を持つユーザーを使用して IAM コンソールにサインイン](#)します。で作成した管理ユーザーを使用することをお勧めします[AWS リソースを管理するための管理ユーザーを作成する](#)。
2. 左側のナビゲーションペインで、ロールを選択します。
3. [ロールの作成] を選択してください。
4. 信頼されたエンティティタイプで、カスタム信頼ポリシーを選択します。
5. デフォルトポリシーを次のポリシーに置き換えて、App Studio アプリケーションがアカウントでこのロールを引き受けることを許可します。

ポリシーで次のプレースホルダーを置き換える必要があります。使用する値は、App Studio のアカウント設定ページにあります。

- **111122223333** を、App Studio インスタンスのセットアップに使用したアカウントの AWS アカウント番号に置き換えます。これは、App Studio インスタンスのアカウント設定に AWS アカウント ID として表示されます。
- **11111111-2222-3333-4444-5555555555** を、App Studio インスタンスのアカウント設定にインスタンス ID としてリストされている App Studio インスタンス ID に置き換えます。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::111122223333:root"
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "StringEquals": {
          "aws:PrincipalTag/IsAppStudioAccessRole": "true",
          "sts:ExternalId":
"11111111-2222-3333-4444-5555555555"
        }
      }
    }
  ]
}
```

[次へ] を選択します。

6. アクセス許可の追加で、前のステップで作成したポリシー (**RedshiftServerlessForAppStudio** または) を検索して選択します **RedshiftProvisionedForAppStudio**。ポリシーの横にある + を選択すると、ポリシーが展開され、ポリシーによって付与されたアクセス許可が表示され、チェックボックスを選択するとポリシーが選択されます。

[次へ] を選択します。

7. 名前、レビュー、および作成ページで、ロール名と説明を指定します。

⚠ Important

ここでのロール名は、[ステップ 1: Amazon Redshift リソースを作成して設定する \(AppBuilderDataAccessRole\)](#) の GRANT コマンドで使用されるロール名と一致する必要があります。

- ステップ 3: タグを追加するで、新しいタグを追加を選択して次のタグを追加して App Studio アクセスを提供します。
 - キー: IsAppStudioDataAccessRole
 - 値: true
- ロールの作成を選択し、生成された Amazon リソース名前 (ARN) を書き留めます。[App Studio で Amazon Redshift コネクタを作成する](#) ときに必要になります。

ステップ 3: Amazon Redshift コネクタを作成する

Amazon Redshift リソースと IAM ポリシーとロールを設定したら、その情報を使用して、ビルダーがアプリケーションを Amazon Redshift に接続するために使用できるコネクタを App Studio に作成します。

📘 Note

コネクタを作成するには、App Studio に管理者ロールが必要です。

Amazon Redshift のコネクタを作成するには

- App Studio に移動します。
- 左側のナビゲーションペインで、「管理」セクションの「コネクタ」を選択します。既存のコネクタのリストとそれぞれの詳細を示すページが表示されます。
- + コネクタの作成を選択します。
- Amazon Redshift コネクタを選択します。
- 次のフィールドに入力してコネクタを設定します。
 - 名前: コネクタの名前を指定します。
 - 説明: コネクタの説明を入力します。

- IAM ロール: で作成した IAM ロールから Amazon リソースネーム (ARN) を入力します [ステップ 2b: App Studio に Amazon Redshift リソースへのアクセスを許可する IAM ロールを作成する](#)。IAM の詳細については、[IAM ユーザーガイド](#) を参照してください。
 - リージョン: Amazon Redshift リソースがある AWS リージョンを選択します。
 - コンピューティングタイプ: Amazon Redshift Serverless を使用しているか、プロビジョニングされたクラスターを使用しているかを選択します。
 - クラスターまたはワークグループの選択: プロビジョニング済み を選択した場合は、App Studio に接続するクラスターを選択します。Serverless を選択した場合は、ワークグループを選択します。
 - データベースの選択: App Studio に接続するデータベースを選択します。
 - 使用可能なテーブル: App Studio に接続するテーブルを選択します。
6. [次へ] を選択します。接続情報を確認し、作成を選択します。
 7. 新しく作成されたコネクタがコネクタリストに表示されます。

Amazon DynamoDB に接続する

App Studio を DynamoDB に接続して、ビルダーがアプリケーションで DynamoDB リソースにアクセスして使用できるようにするには、次の手順を実行する必要があります。

1. [ステップ 1: DynamoDB リソースを作成して設定する](#)
2. [ステップ 2: 適切な DynamoDB アクセス許可を持つ IAM ポリシーとロールを作成する](#)
3. [DynamoDB コネクタを作成する](#)

ステップ 1: DynamoDB リソースを作成して設定する

App Studio で使用する DynamoDB リソースを作成および設定するには、次の手順に従います。

App Studio で使用する DynamoDB をセットアップするには

1. にサインイン AWS マネジメントコンソール し、<https://console.aws.amazon.com/dynamodb/> で DynamoDB コンソールを開きます。

で作成した管理ユーザーを使用することをお勧めします [AWS リソースを管理するための管理ユーザーを作成する](#)。

2. 左のナビゲーションペインで、[テーブル] を選択します。

3. [テーブルの作成] を選択します。
4. テーブルの名前とキーを入力します。
5. [テーブルの作成] を選択します。
6. テーブルを作成したら、テーブルが App Studio に接続された後に表示されるように、いくつかの項目を追加します。
 - a. テーブルを選択し、アクションを選択し、項目を探索を選択します。
 - b. 返されたアイテムで、アイテムの作成 を選択します。
 - c. (オプション): 新しい属性を追加を選択して、テーブルに属性を追加します。
 - d. 各属性の値を入力し、項目の作成を選択します。

ステップ 2: 適切な DynamoDB アクセス許可を持つ IAM ポリシーとロールを作成する

App Studio で DynamoDB リソースを使用するには、管理者は IAM ポリシーとロールを作成して、リソースへのアクセス許可を App Studio に付与する必要があります。IAM ポリシーは、ビルダーが使用できるデータの範囲と、作成、読み取り、更新、削除など、そのデータに対して呼び出すことができるオペレーションを制御します。IAM ポリシーは、App Studio で使用される IAM ロールにアタッチされます。

サービスおよびポリシーごとに少なくとも 1 つの IAM ロールを作成することをお勧めします。たとえば、ビルダーが DynamoDB の同じテーブルにバックアップされた 2 つのアプリケーションを作成する場合、1 つは読み取りアクセスのみが必要で、もう 1 つは読み取り、作成、更新、削除が必要です。管理者は 2 つの IAM ロールを作成し、1 つは読み取り専用アクセス許可を使用し、もう 1 つは DynamoDB の該当するテーブルに対する完全な CRUD アクセス許可を持っている必要があります。

ステップ 2a: 適切な DynamoDB アクセス許可を持つ IAM ポリシーを作成する

App Studio で作成して使用する IAM ポリシーには、アプリケーションがベストプラクティスに従うための適切なリソースに対する最小限のアクセス許可のみを含める必要があります。

適切な DynamoDB アクセス許可を持つ IAM ポリシーを作成するには

1. [IAM ポリシーを作成する権限を持つユーザーを使用して IAM コンソールにサインイン](#)します。で作成した管理ユーザーを使用することをお勧めします [AWS リソースを管理するための管理ユーザーを作成する](#)。
2. 左側のナビゲーションペインで、ポリシーを選択します。
3. [Create policy] (ポリシーの作成) を選択します。

4. [ポリシーエディタ] セクションで、[JSON] オプションを選択します。
5. JSON ポリシードキュメントに入力または貼り付けます。次のタブには、DynamoDB テーブルへの読み取り専用およびフルアクセス用のポリシーの例と、AWS KMS カスタマーマネージドキー (CMK) で暗号化された DynamoDB テーブルの AWS KMS アクセス許可を含むポリシーの例が含まれています。

Note

以下のポリシーは、ワイルドカード (*) を使用するすべての DynamoDB リソースに適用されます*。ベストプラクティスとして、ワイルドカードを App Studio で使用するリソースの Amazon リソースネーム (ARN) に置き換える必要があります。

Read only

次のポリシーは、設定された DynamoDB リソースへの読み取りアクセスを許可します。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ReadOnlyDDBForAppStudio",
      "Effect": "Allow",
      "Action": [
        "dynamodb:ListTables",
        "dynamodb:DescribeTable",
        "dynamodb: PartiQLSelect"
      ],
      "Resource": "*"
    }
  ]
}
```

Full access

次のポリシーは、設定された DynamoDB リソースへの作成、読み取り、更新、削除アクセスを許可します。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "FullAccessDDBForAppStudio",
      "Effect": "Allow",
      "Action": [
        "dynamodb:ListTables",
        "dynamodb:DescribeTable",
        "dynamodb:PartiQLSelect",
        "dynamodb:PartiQLInsert",
        "dynamodb:PartiQLUpdate",
        "dynamodb:PartiQLDelete"
      ],
      "Resource": "*"
    }
  ]
}
```

Read only - KMS encrypted

次のポリシーは、AWS KMS アクセス許可を提供することで、設定された暗号化された DynamoDB リソースへの読み取りアクセスを許可します。ARN は AWS KMS キーの ARN に置き換える必要があります。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ReadOnlyDDBForAppStudio",
      "Effect": "Allow",
      "Action": [
        "dynamodb:ListTables",
        "dynamodb:DescribeTable",
        "dynamodb:PartiQLSelect"
      ]
    }
  ]
}
```

```
    ],
    "Resource": "*"
  },
  {
    "Sid": "KMSPermissionsForEncryptedTable",
    "Effect": "Allow",
    "Action": [
      "kms:Decrypt",
      "kms:DescribeKey"
    ],
    "Resource": "arn:aws:kms:us-  
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
  }
]
```

Full access - KMS encrypted

次のポリシーは、AWS KMS アクセス許可を提供することで、設定された暗号化された DynamoDB リソースへの読み取りアクセスを許可します。ARN は AWS KMS キーの ARN に置き換える必要があります。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ReadOnlyDDBForAppStudio",
      "Effect": "Allow",
      "Action": [
        "dynamodb:ListTables",
        "dynamodb:DescribeTable",
        "dynamodb: PartiQLSelect",
        "dynamodb: PartiQLInsert",
        "dynamodb: PartiQLUpdate",
        "dynamodb: PartiQLDelete"
      ],
      "Resource": "*"
    },
  ]
}
```

```
    "Sid": "KMSPermissionsForEncryptedTable",
    "Effect": "Allow",
    "Action": [
        "kms:Decrypt",
        "kms:DescribeKey"
    ],
    "Resource": "arn:aws:kms:us-  
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
  }
]
```

6. [次へ] を選択します。
7. 確認と作成ページで、 **ReadOnlyDDBForAppStudio** や などのポリシー名 **FullAccessDDBForAppStudio**、 および説明 (オプション) を指定します。
8. [ポリシーの作成] を選択し、ポリシーを作成します。

ステップ 2b: App Studio に DynamoDB リソースへのアクセスを許可する IAM ロールを作成する

次に、以前に作成したポリシーを使用する IAM ロールを作成します。App Studio はこのポリシーを使用して、設定された DynamoDB リソースにアクセスします。

App Studio に DynamoDB リソースへのアクセスを許可する IAM ロールを作成するには

1. [IAM ロールを作成する権限を持つユーザーを使用して IAM コンソールにサインインします。](#) で作成した管理ユーザーを使用することをお勧めします [AWS リソースを管理するための管理ユーザーを作成する](#)。
2. コンソールのナビゲーションペインで、[ロール]、[ロールの作成] の順に選択します。
3. 信頼されたエンティティタイプで、カスタム信頼ポリシーを選択します。
4. デフォルトポリシーを次のポリシーに置き換えて、App Studio アプリケーションがアカウントでこのロールを引き受けることを許可します。

ポリシーで次のプレースホルダーを置き換える必要があります。使用する値は、App Studio のアカウント設定ページにあります。

- **111122223333** を、App Studio インスタンスのセットアップに使用したアカウントの AWS アカウント番号に置き換えます。これは、App Studio インスタンスのアカウント設定に AWS アカウント ID として表示されます。

- **11111111-2222-3333-4444-5555555555** を、App Studio インスタンスのアカウント設定でインスタンス ID としてリストされている App Studio インスタンス ID に置き換えます。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::111122223333:root"
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "StringEquals": {
          "aws:PrincipalTag/IsAppStudioAccessRole": "true",
          "sts:ExternalId":
            "11111111-2222-3333-4444-5555555555"
        }
      }
    }
  ]
}
```

[次へ] を選択します。

5. アクセス許可の追加で、前のステップで作成したポリシー (**ReadOnlyDDBForAppStudio** または) を検索して選択します **FullAccessDDBForAppStudio**。ポリシーの横にある + を選択すると、ポリシーが展開され、ポリシーによって付与されたアクセス許可が表示され、チェックボックスを選択するとポリシーが選択されます。

[次へ] を選択します。

6. 名前、レビュー、および作成ページで、ロール名と説明を指定します。
7. ステップ 3: タグを追加するで、新しいタグを追加を選択して次のタグを追加して App Studio アクセスを提供します。
 - キー: IsAppStudioDataAccessRole

- 値: true

8. ロールの作成を選択し、生成された Amazon リソースネーム (ARN) を書き留めます。[App Studio で DynamoDB コネクタを作成する](#) ときに必要になります。

DynamoDB コネクタを作成する

DynamoDB リソースと IAM ポリシーとロールを設定したら、その情報を使用して、ビルダーがアプリケーションを DynamoDB に接続するために使用できるコネクタを App Studio に作成します。

Note

コネクタを作成するには、App Studio に管理者ロールが必要です。

DynamoDB のコネクタを作成するには

1. App Studio に移動します。
2. 左側のナビゲーションペインで、「管理」セクションの「コネクタ」を選択します。既存のコネクタのリストとそれぞれの詳細を示すページが表示されます。
3. + コネクタの作成を選択します。
4. コネクタタイプのリストから Amazon DynamoDB を選択します。
5. 次のフィールドに入力してコネクタを設定します。
 - 名前: DynamoDB コネクタの名前を入力します。
 - 説明: DynamoDB コネクタの説明を入力します。
 - IAM ロール: で作成した IAM ロールから Amazon リソースネーム (ARN) を入力します。[ステップ 2b: App Studio に DynamoDB リソースへのアクセスを許可する IAM ロールを作成する](#)。IAM の詳細については、[IAM ユーザーガイド](#) を参照してください。
 - リージョン: DynamoDB リソースがある AWS リージョンを選択します。
 - 使用可能なテーブル: App Studio に接続するテーブルを選択します。
6. [次へ] を選択します。接続情報を確認し、作成を選択します。
7. 新しく作成されたコネクタがコネクタリストに表示されます。

に接続する AWS Lambda

App Studio を Lambda に接続して、ビルダーがアプリケーションで Lambda リソースにアクセスして使用できるようにするには、次の手順を実行する必要があります。

1. [ステップ 1: Lambda 関数を作成して設定する](#)
2. [ステップ 2: App Studio に Lambda リソースへのアクセスを許可する IAM ロールを作成する](#)
3. [ステップ 3: Lambda コネクタを作成する](#)

ステップ 1: Lambda 関数を作成して設定する

既存の Lambda 関数がない場合は、まず作成する必要があります。Lambda 関数の作成の詳細については、「[AWS Lambda デベロッパーガイド](#)」を参照してください。

ステップ 2: App Studio に Lambda リソースへのアクセスを許可する IAM ロールを作成する

App Studio で Lambda リソースを使用するには、管理者は IAM ロールを作成して、リソースへのアクセス許可を App Studio に付与する必要があります。IAM ロールは、アプリケーションが Lambda からアクセスできるリソースまたはオペレーションを制御します。

サービスおよびポリシーごとに少なくとも 1 つの IAM ロールを作成することをお勧めします。

App Studio に Lambda リソースへのアクセスを許可する IAM ロールを作成するには

1. [IAM ロールを作成する権限を持つユーザーを使用して IAM コンソールにサインインします。](#)で作成した管理ユーザーを使用することをお勧めします。[AWS リソースを管理するための管理ユーザーを作成する。](#)
2. コンソールのナビゲーションペインで、[ルール]、[ルールの作成] の順に選択します。
3. 信頼されたエンティティタイプで、カスタム信頼ポリシーを選択します。
4. デフォルトポリシーを次のポリシーに置き換えて、App Studio アプリケーションがアカウントでこのルールを引き受けることを許可します。

ポリシーで次のプレースホルダーを置き換える必要があります。使用する値は、App Studio のアカウント設定ページにあります。

- **111122223333** を、App Studio インスタンスのセットアップに使用したアカウントの AWS アカウント番号に置き換えます。これは、App Studio インスタンスのアカウント設定に AWS アカウント ID として表示されます。

- **11111111-2222-3333-4444-5555555555** を、App Studio インスタンスのアカウント設定でインスタンス ID としてリストされている App Studio インスタンス ID に置き換えます。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::111122223333:root"
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "StringEquals": {
          "aws:PrincipalTag/IsAppStudioAccessRole": "true",
          "sts:ExternalId":
"11111111-2222-3333-4444-5555555555"
        }
      }
    }
  ]
}
```

[次へ] を選択します。

5. アクセス許可の追加で、ロールに適切なアクセス許可を付与するポリシーを検索して選択します。ポリシーの横にある + を選択すると、ポリシーが展開され、ポリシーによって付与されたアクセス許可が表示され、チェックボックスを選択するとポリシーが選択されます。Lambda では、Lambda 関数を呼び出すアクセス許可を付与する `AWSLambdaRole` ポリシーの追加を検討できます。

マネージドポリシーのリストとその説明など、Lambda での IAM ポリシーの使用の詳細については、「[AWS Lambda デベロッパーガイド](#)」の「[Identity and Access Management AWS Lambda](#)」を参照してください。

[次へ] を選択します。

6. 名前、レビュー、および作成ページで、ロール名と説明を指定します。

7. ステップ 3: タグを追加するで、新しいタグを追加を選択して次のタグを追加して App Studio アクセスを提供します。
 - キー: IsAppStudioDataAccessRole
 - 値: true
8. ロールの作成を選択し、生成された Amazon リソースネーム (ARN) を書き留めます。[App Studio で Lambda コネクタを作成する](#)ときに必要になります。

ステップ 3: Lambda コネクタを作成する

Lambda リソースと IAM ポリシーとロールを設定したら、その情報を使用して、ビルダーがアプリケーションを Lambda に接続するために使用できるコネクタを App Studio に作成します。

Note

コネクタを作成するには、App Studio に管理者ロールが必要です。

Lambda のコネクタを作成するには

1. App Studio に移動します。
2. 左側のナビゲーションペインで、「管理」セクションの「コネクタ」を選択します。既存のコネクタのリストとそれぞれの詳細を示すページが表示されます。
3. + コネクタの作成を選択します。
4. コネクタタイプのリストからその他の AWS サービスを選択します。
5. 次のフィールドに入力してコネクタを設定します。
 - 名前: Lambda コネクタの名前を入力します。
 - 説明: Lambda コネクタの説明を入力します。
 - IAM ロール: で作成した IAM ロールから Amazon リソースネーム (ARN) を入力します [ステップ 2: App Studio に Lambda リソースへのアクセスを許可する IAM ロールを作成する](#)。IAM の詳細については、[IAM ユーザーガイド](#) を参照してください。
 - サービス: Lambda を選択します。
 - リージョン: Lambda リソースがある AWS リージョンを選択します。
6. [作成] を選択します。
7. 新しく作成されたコネクタがコネクタリストに表示されます。

Amazon Simple Storage Service (Amazon S3) に接続する

App Studio を Amazon S3 に接続して、ビルダーがアプリケーションで Amazon S3 リソースにアクセスして使用できるようにするには、次の手順を実行します。

1. [ステップ 1: Amazon S3 リソースを作成して設定する](#)
2. [ステップ 2: 適切な Amazon S3 アクセス許可を持つ IAM ポリシーとロールを作成する](#)
3. [ステップ 3: Amazon S3 コネクタを作成する](#)

ステップを完了し、適切なアクセス許可を持つコネクタを作成したら、ビルダーはコネクタを使用して Amazon S3 リソースとやり取りするアプリケーションを作成できます。App Studio アプリケーションで Amazon S3 を操作する方法の詳細については、「」を参照してください [Amazon Simple Storage Service とコンポーネントおよびオートメーションの操作](#)。

ステップ 1: Amazon S3 リソースを作成して設定する

アプリのニーズと既存のリソースによっては、アプリの書き込みと読み取りを行う Amazon S3 バケットを作成する必要がある場合があります。バケットを含む Amazon S3 リソースの作成については、Amazon Simple Storage Service ユーザーガイドの [Amazon S3 の開始方法](#)」を参照してください。

アプリケーションで [S3 アップロード](#) コンポーネントを使用するには、アップロード先の Amazon S3 バケットに Cross-Origin Resource Sharing (CORS) 設定を追加する必要があります。CORS 設定は、オブジェクトをバケットにプッシュするアクセス許可を App Studio に付与します。次の手順では、コンソールを使用して Amazon S3 バケットに CORS 設定を追加する方法について詳しく説明します。CORS とその設定の詳細については、「Amazon Simple Storage Service ユーザーガイド」の [「Cross-Origin Resource Sharing \(CORS\) の使用](#)」を参照してください。

コンソールで Amazon S3 バケットに CORS 設定を追加するには

1. <https://console.aws.amazon.com/s3/> でバケットに移動します。
2. [アクセス許可] タブを選択します。
3. Cross-Origin Resource Sharing (CORS) で、Edit を選択します。
4. 次のスニペットを追加します。

```
[
  {
    "AllowedHeaders": [
```

```
        "*"
    ],
    "AllowedMethods": [
        "PUT",
        "POST"
    ],
    "AllowedOrigins": [
        "*"
    ],
    "ExposeHeaders": []
}
]
```

5. [Save changes] (変更の保存) をクリックします。

ステップ 2: 適切な Amazon S3 アクセス許可を持つ IAM ポリシーとロールを作成する

App Studio で Amazon S3 リソースを使用するには、管理者は IAM ポリシーとロールを作成して、リソースへのアクセス許可を App Studio に付与する必要があります。IAM ポリシーは、ビルダーが使用できるデータの範囲と、作成、読み取り、更新、削除など、そのデータに対して呼び出すことができるオペレーションを制御します。IAM ポリシーは、App Studio で使用される IAM ロールにアタッチされます。

サービスおよびポリシーごとに少なくとも 1 つの IAM ロールを作成することをお勧めします。例えば、ビルダーが Amazon S3 の異なるバケットにバックアップされた 2 つのアプリケーションを作成している場合、管理者はバケットごとに 1 つずつ、2 つの IAM ポリシーとロールを作成する必要があります。

ステップ 2a: 適切な Amazon S3 アクセス許可を持つ IAM ポリシーを作成する

App Studio で作成して使用する IAM ポリシーには、アプリケーションがベストプラクティスに従うための適切なリソースに対する最小限のアクセス許可のみを含める必要があります。

適切な Amazon S3 アクセス許可を持つ IAM ポリシーを作成するには

1. [IAM ポリシーを作成する権限を持つユーザーを使用して IAM コンソールにサインイン](#)します。で作成した管理ユーザーを使用することをお勧めします[AWS リソースを管理するための管理ユーザーを作成する](#)。
2. 左側のナビゲーションペインで、ポリシーを選択します。
3. [Create policy] (ポリシーの作成) を選択します。

4. [ポリシーエディタ] セクションで、[JSON] オプションを選択します。
5. JSON ポリシードキュメントに入力または貼り付けます。次のタブには、Amazon S3 リソースへの読み取り専用およびフルアクセスのポリシー例が含まれています。

 Note

以下のポリシーは、ワイルドカード (*) を使用するすべての Amazon S3 リソースに適用されます*。セキュリティのベストプラクティスとして、ワイルドカードを App Studio で使用するバケットやフォルダなどのリソースの Amazon リソースネーム (ARN) に置き換える必要があります。

Read only

次のポリシーは、設定された Amazon S3 バケットまたはフォルダへの読み取り専用アクセス (取得および一覧表示) を許可します。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "S3ReadOnlyForAppStudio",
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:ListBucket"
      ],
      "Resource": "*"
    }
  ]
}
```

Full access

次のポリシーは、設定された Amazon S3 バケットまたはフォルダへのフルアクセス (入力、取得、一覧表示、削除) を許可します。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "S3FullAccessForAppStudio",
      "Effect": "Allow",
      "Action": [
        "s3:PutObject",
        "s3:GetObject",
        "s3:ListBucket",
        "s3:DeleteObject"
      ],
      "Resource": "*"
    }
  ]
}
```

6. [次へ] を選択します。
7. 確認と作成ページで、**AWSAppStudioS3FullAccess**、説明 (オプション) などのポリシー名を指定します。
8. [ポリシーの作成] を選択し、ポリシーを作成します。

ステップ 2b: App Studio に Amazon S3 リソースへのアクセスを許可する IAM ロールを作成する

App Studio で Amazon S3 リソースを使用するには、管理者は IAM ロールを作成して、リソースへのアクセス許可を App Studio に付与する必要があります。IAM ロールは、ビルダーが使用できるデータの範囲と、作成、読み取り、更新、削除など、そのデータに対して呼び出すことができるオペレーションを制御します。

サービスおよびポリシーごとに少なくとも 1 つの IAM ロールを作成することをお勧めします。

App Studio に Amazon S3 リソースへのアクセスを許可する IAM ロールを作成するには

1. [IAM ロールを作成する権限を持つユーザーを使用して IAM コンソールにサインインします。](#) で作成した管理ユーザーを使用することをお勧めします [AWS リソースを管理するための管理ユーザーを作成する](#)。
2. コンソールのナビゲーションペインで、[ルール]、[ルールの作成] の順に選択します。

3. 信頼されたエンティティタイプで、カスタム信頼ポリシーを選択します。
4. デフォルトポリシーを次のポリシーに置き換えて、App Studio アプリケーションがアカウントでこのロールを引き受けることを許可します。

ポリシーで次のプレースホルダーを置き換える必要があります。使用する値は、App Studio のアカウント設定ページにあります。

- **111122223333** を、App Studio インスタンスのセットアップに使用したアカウントの AWS アカウント番号に置き換えます。これは、App Studio インスタンスのアカウント設定にアカウント AWS ID として表示されます。
- **11111111-2222-3333-4444-5555555555** を、App Studio インスタンスのアカウント設定でインスタンス ID としてリストされている App Studio インスタンス ID に置き換えます。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::111122223333:root"
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "StringEquals": {
          "aws:PrincipalTag/IsAppStudioAccessRole": "true",
          "sts:ExternalId":
            "11111111-2222-3333-4444-5555555555"
        }
      }
    }
  ]
}
```

[次へ] を選択します。

5. アクセス許可の追加で、前のステップで作成したポリシー (**S3ReadOnlyForAppStudio** または) を検索して選択します **S3FullAccessForAppStudio**。ポリシーの横にある + を選択する

と、ポリシーが展開され、ポリシーによって付与されたアクセス許可が表示され、チェックボックスを選択するとポリシーが選択されます。

[次へ] を選択します。

- 名前、レビュー、および作成ページで、ロール名と説明を指定します。
- ステップ 3: タグを追加するで、新しいタグを追加を選択して次のタグを追加して App Studio アクセスを提供します。
 - キー: IsAppStudioDataAccessRole
 - 値: true
- ロールの作成を選択し、生成された Amazon リソースネーム (ARN) を書き留めます。次のステップで App Studio で Amazon S3 コネクタを作成する必要があります。

ステップ 3: Amazon S3 コネクタを作成する

Amazon S3 リソースと IAM ポリシーとロールを設定したら、その情報を使用して、ビルダーがアプリケーションを Amazon S3 に接続するために使用できるコネクタを App Studio に作成します。

Note

コネクタを作成するには、App Studio に管理者ロールが必要です。

Amazon S3 のコネクタを作成するには

- App Studio に移動します。
- 左側のナビゲーションペインで、「管理」セクションの「コネクタ」を選択します。既存のコネクタのリストとそれぞれの詳細を示すページが表示されます。
- + コネクタの作成を選択します。
- Amazon S3 コネクタを選択します。
- 次のフィールドに入力してコネクタを設定します。
 - 名前: Amazon S3 コネクタの名前を入力します。
 - 説明: Amazon S3 コネクタの説明を入力します。
 - IAM ロール: で作成した IAM ロールから Amazon リソースネーム (ARN) を入力します [ステップ 2b: App Studio に Amazon S3 リソースへのアクセスを許可する IAM ロールを作成する](#)。IAM の詳細については、[IAM ユーザーガイド](#) を参照してください。

- リージョン: Amazon S3 リソースがある AWS リージョンを選択します。
6. [作成] を選択します。
 7. 新しく作成されたコネクタがコネクタリストに表示されます。

Amazon Aurora に接続する

App Studio を Aurora に接続して、ビルダーがアプリケーションで Aurora リソースにアクセスして使用できるようにするには、次の手順を実行する必要があります。

1. [ステップ 1: Aurora リソースを作成して設定する](#)
2. [ステップ 2: 適切な Aurora アクセス許可を持つ IAM ポリシーとロールを作成する](#)
3. [ステップ 3: App Studio で Aurora コネクタを作成する](#)

App Studio は、次の Aurora バージョンをサポートしています。

- Aurora MySQL Serverless V1: 5.72
- Aurora PostgreSQL Serverless V1: 11.18、13.9
- Aurora MySQL Serverless V2: 13.11 以降、14.8 以降、および 15.3 以降
- Aurora PostgreSQL Serverless V2: 13.11 以降、14.8 以降、および 15.3 以降

ステップ 1: Aurora リソースを作成して設定する

App Studio で Aurora データベースを使用するには、まずデータベースを作成して適切に設定する必要があります。App Studio では、Aurora PostgreSQL と Aurora MySQL の 2 つの Aurora データベースタイプがサポートされています。タイプを比較するには、[MySQL と PostgreSQL の違いは何ですか?](#)」を参照してください。適切なタブを選択し、手順に従って App Studio アプリで使用する Aurora を設定します。

Aurora PostgreSQL

App Studio で使用する Aurora PostgreSQL データベースクラスターを作成および設定するには、次の手順に従います。

App Studio で使用する Aurora を設定するには

1. にサインイン AWS マネジメントコンソール し、<https://console.aws.amazon.com/rds/> で Amazon RDS コンソールを開きます。

2. [データベースの作成] を選択します。
3. Aurora (PostgreSQL 互換) を選択します。
4. 使用可能なバージョンで、バージョン 13.11、14.8 および 以降のバージョンを選択します15.3。
5. 設定で、DB クラスター識別子を入力します。
6. インスタンス設定で、Serverless v2 を選択し、適切な容量を選択します。
7. Connectivity で、RDS Data API を有効にする を選択します。
8. データベース認証で、IAM データベース認証を選択します。
9. 追加設定の「初期データベース名」に、データベースの初期データベース名を入力します。

Aurora MySQL

App Studio で使用する Aurora MySQL データベースクラスターを作成および設定するには、次の手順に従います。

Aurora MySQL は、Data API または Serverless v1 をサポートするバージョンの UI からの作成をサポートしていません。Data API をサポートする Aurora MySQL クラスターを作成するには、を使用する必要があります AWS CLI。

Note

App Studio で Aurora MySQL データベースを使用するには、仮想プライベートクラウド (VPC) に存在する必要があります。詳細については、「Amazon Aurora [ユーザーガイド](#)」の「[VPC での DB クラスターの使用](#)」を参照してください。

App Studio で使用する Aurora MySQL を設定するには

1. 必要に応じて、「AWS Command Line Interface ユーザーガイド」の「[のインストールまたは最新バージョンへの更新](#)」の AWS CLI 手順に従って、をインストールします。 [AWS CLI](#)
2. にサインイン AWS マネジメントコンソールし、 <https://console.aws.amazon.com/rds/> で Amazon RDS コンソールを開きます。
3. 左側のナビゲーションで、サブネットグループを選択します。
4. [Create DB subnet group] (DB サブネットグループの作成) を選択します。
5. 情報を入力し、sunbnet グループを作成します。サブネットグループと VPCs [「VPC での DB クラスターの使用」](#) を参照してください。

6. 次の AWS CLI コマンドを実行します。

```
aws rds create-db-cluster --database-name db_name \  
  --db-cluster-identifier db_cluster_identifier \  
  --engine aurora-mysql \  
  --engine-version 5.7.mysql_aurora.2.08.3 \  
  --engine-mode serverless \  
  --scaling-configuration  
  MinCapacity=4,MaxCapacity=32,SecondsUntilAutoPause=1000,AutoPause=true \  
  --master-username userName \  
  --master-user-password userPass \  
  --availability-zones us-west-2b us-west-2c \  
  --db-subnet-group-name subnet-group-name
```

次のフィールドを変更します。

- *db_name* を目的のデータベース名に置き換えます。
- *db_cluster_identifier* を目的のデータベースクラスター識別子に置き換えます。
- (オプション) 必要に応じて *scaling-configuration* フィールドの数値を置き換えます。
- *userName* を目的のユーザー名に置き換えます。
- *userPass* を目的のパスワードに置き換えます。
- *availability-zones*、作成したサブネットグループからアベイラビリティゾーンを追加します。
- *subnet-group-name* を、作成したサブネットグループの名前に置き換えます。

ステップ 2: 適切な Aurora アクセス許可を持つ IAM ポリシーとロールを作成する

App Studio で Aurora リソースを使用するには、管理者は IAM ポリシーを作成し、設定されたリソースへのアクセス許可を App Studio に付与するために使用される IAM ロールにアタッチする必要があります。IAM ポリシーとロールは、ビルダーが使用できるデータの範囲と、作成、読み取り、更新、削除など、そのデータに対して呼び出すことができるオペレーションを制御します。

サービスおよびポリシーごとに少なくとも 1 つの IAM ロールを作成することをお勧めします。

ステップ 2a: 適切な Aurora アクセス許可を持つ IAM ポリシーを作成する

App Studio で作成して使用する IAM ポリシーには、アプリケーションがベストプラクティスに従うための適切なリソースに対する最小限のアクセス許可のみを含める必要があります。

適切な Aurora アクセス許可を持つ IAM ポリシーを作成するには

1. [IAM ロールを作成する権限を持つユーザーを使用して IAM コンソールにサインインします。](#)で作成した管理ユーザーを使用することをお勧めします。[AWS リソースを管理するための管理ユーザーを作成する。](#)
2. 左側のナビゲーションペインで、ポリシーを選択します。
3. [Create policy] (ポリシーの作成) を選択します。
4. [ポリシーエディタ] セクションで、[JSON] オプションを選択します。
5. 既存のスニペットを次のスニペットに置き換え、**111122223333** を AWS Amazon Redshift および Aurora リソースが含まれているアカウント番号に置き換えます。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "BaselineAuroraForAppStudio",
      "Effect": "Allow",
      "Action": [
        "rds-data:ExecuteStatement",
        "secretsmanager:GetSecretValue"
      ],
      "Resource": [
        "arn:aws:rds:*:111122223333:cluster:*",
        "arn:aws:secretsmanager:*:111122223333:secret:rds*"
      ]
    }
  ]
}
```

6. [次へ] を選択します。
7. 確認と作成ページで、**Aurora_AppStudio**や説明 (オプション) などのポリシー名を指定します。
8. [ポリシーの作成]を選択し、ポリシーを作成します。

ステップ 2b: App Studio に Aurora リソースへのアクセスを許可する IAM ロールを作成する

次に、以前に作成したポリシーを使用する IAM ロールを作成します。App Studio はこのポリシーを使用して、設定された Aurora リソースにアクセスします。

App Studio に Aurora リソースへのアクセスを許可する IAM ロールを作成するには

1. [IAM ロールを作成する権限を持つユーザーを使用して IAM コンソールにサインインします。](#) で作成した管理ユーザーを使用することをお勧めします[AWS リソースを管理するための管理ユーザーを作成する。](#)
2. コンソールのナビゲーションペインで、[ロール]、[ロールの作成] の順に選択します。
3. 信頼されたエンティティタイプで、カスタム信頼ポリシーを選択します。
4. デフォルトポリシーを次のポリシーに置き換えて、App Studio アプリケーションがアカウントでこのロールを引き受けることを許可します。

ポリシーで次のプレースホルダーを置き換える必要があります。使用する値は、App Studio のアカウント設定ページにあります。

- **111122223333** を、App Studio インスタンスのセットアップに使用したアカウントの AWS アカウント番号に置き換えます。これは、App Studio インスタンスのアカウント設定にアカウント AWS ID として表示されます。
- **11111111-2222-3333-4444-5555555555** を、App Studio インスタンスのアカウント設定でインスタンス ID としてリストされている App Studio インスタンス ID に置き換えます。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::111122223333:root"
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "StringEquals": {
          "aws:PrincipalTag/IsAppStudioAccessRole": "true",
```

```
        "sts:ExternalId":  
          "11111111-2222-3333-4444-555555555555"  
        }  
      }  
    }  
  ]  
}
```

[次へ] を選択します。

5. アクセス許可を追加で、前に作成したポリシーを検索して選択します (**Aurora_AppStudio**)。ポリシーの横にある + を選択すると、ポリシーが展開され、ポリシーによって付与されたアクセス許可が表示され、チェックボックスを選択するとポリシーが選択されます。

[次へ] を選択します。

6. 名前、レビュー、および作成ページで、ロール名と説明を指定します。
7. ステップ 3: タグを追加するで、新しいタグを追加を選択して次のタグを追加して App Studio アクセスを提供します。
 - キー: IsAppStudioDataAccessRole
 - 値: true
8. ロールの作成を選択し、生成された Amazon リソースネーム (ARN) を書き留めます。 [App Studio で Aurora コネクタを作成する](#) ときに必要になります。

ステップ 3: App Studio で Aurora コネクタを作成する

Aurora リソースと IAM ポリシーとロールを設定したら、その情報を使用して、ビルダーがアプリを Aurora に接続するために使用できるコネクタを App Studio に作成します。

Note

コネクタを作成するには、App Studio に管理者ロールが必要です。

Aurora のコネクタを作成するには

1. App Studio に移動します。

2. 左側のナビゲーションペインで、「管理」セクションの「コネクタ」を選択します。既存のコネクタのリストとそれぞれの詳細を示すページが表示されます。
3. + コネクタの作成を選択します。
4. Amazon Aurora コネクタを選択します。
5. 次のフィールドに入力してコネクタを設定します。
 - 名前: Aurora コネクタの名前を入力します。
 - 説明: Aurora コネクタの説明を入力します。
 - IAM ロール: で作成した IAM ロールから Amazon リソースネーム (ARN) を入力します [ステップ 2b: App Studio に Aurora リソースへのアクセスを許可する IAM ロールを作成する](#)。IAM の詳細については、[IAM ユーザーガイド](#) を参照してください。
 - シークレット ARN: データベースクラスターのシークレット ARN を入力します。シークレット ARN の場所については、[「Amazon Aurora ユーザーガイド」の「DB cluser のシークレットの詳細の表示」](#) を参照してください。
 - リージョン: Aurora リソースがある AWS リージョンを選択します。
 - データベース ARN: データベースクラスターの ARN を入力します。ARN は、シークレット ARN と同様に、データベースクラスターの設定タブにあります。
 - データベースタイプ: で作成されたデータベースのタイプと一致するデータベースタイプ MySQL または PostgreSQL を選択します [ステップ 1: Aurora リソースを作成して設定する](#)。
 - データベース名: データベースの名前を入力します。データベースの名前は、データベースクラスターの設定タブでも確認できます。
 - 使用可能なテーブル: このコネクタを使用して App Studio で使用するテーブルを選択します。
6. Next を選択して、エンティティマッピングを確認または定義します。
7. Create を選択して Aurora コネクタを作成します。新しく作成されたコネクタがコネクタリストに表示されます。

Amazon Bedrock に接続する

App Studio を Amazon Bedrock に接続して、ビルダーがアプリケーションで Amazon Bedrock にアクセスして使用できるようにするには、次の手順を実行する必要があります。

1. [ステップ 1: Amazon Bedrock モデルを有効にする](#)
2. [ステップ 2: 適切な Amazon Bedrock アクセス許可を持つ IAM ポリシーとロールを作成する](#)
3. [ステップ 3: Amazon Bedrock コネクタを作成する](#)

ステップ 1: Amazon Bedrock モデルを有効にする

Amazon Bedrock モデルを有効にするには、次の手順に従います。

Amazon Bedrock モデルを有効にするには

1. にサインイン AWS マネジメントコンソール し、<https://console.aws.amazon.com/bedrock/> で Amazon Bedrock コンソールを開きます。
2. 左側のナビゲーションペインで、モデルアクセスを選択します。
3. 使用するモデルを有効にします。詳細については、「[Amazon Bedrock 基盤モデルへのアクセスを管理する](#)」を参照してください。

ステップ 2: 適切な Amazon Bedrock アクセス許可を持つ IAM ポリシーとロールを作成する

App Studio で Amazon Bedrock リソースを使用するには、管理者は IAM ポリシーとロールを作成して、リソースへのアクセス許可を App Studio に付与する必要があります。IAM ポリシーは、などのリソースに対して呼び出すことができるリソースとオペレーションを制御します InvokeModel。IAM ポリシーは、App Studio で使用される IAM ロールにアタッチされます。

ステップ 2a: 適切な Amazon Bedrock アクセス許可を持つ IAM ポリシーを作成する

App Studio で作成して使用する IAM ポリシーには、アプリケーションがベストプラクティスに従うための適切なリソースに対する最小限のアクセス許可のみを含める必要があります。

適切な Amazon Bedrock アクセス許可を持つ IAM ポリシーを作成するには

1. [IAM ポリシーを作成する権限を持つユーザーを使用して IAM コンソールにサインイン](#)します。で作成した管理ユーザーを使用することをお勧めします [AWS リソースを管理するための管理ユーザーを作成する](#)。
2. 左側のナビゲーションペインで、ポリシーを選択します。
3. [Create policy] (ポリシーの作成) を選択します。
4. [ポリシーエディタ] セクションで、[JSON] オプションを選択します。
5. JSON ポリシードキュメントに入力または貼り付けます。次のポリシー例では、ワイルドカード () を使用して、InvokeModel すべての Amazon Bedrock リソースで を提供します*。

セキュリティのベストプラクティスとして、ワイルドカードを App Studio で使用するリソースの Amazon リソースネーム (ARN) に置き換える必要があります。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "BedrockAccessForAppStudio",
      "Effect": "Allow",
      "Action": [
        "bedrock:InvokeModel"
      ],
      "Resource": "*"
    }
  ]
}
```

6. [次へ] を選択します。
7. 確認と作成ページで、**BedrockAccessForAppStudio**説明 (オプション) などのポリシー名を指定します。
8. [ポリシーの作成]を選択し、ポリシーを作成します。

ステップ 2b: App Studio に Amazon Bedrock へのアクセスを許可する IAM ロールを作成する

App Studio で Amazon Bedrock を使用するには、管理者は IAM ロールを作成して、リソースへのアクセス許可を App Studio に付与する必要があります。IAM ロールは、使用する App Studio アプリのアクセス許可の範囲を制御し、コネクタを作成するときに使用します。サービスおよびポリシーごとに少なくとも 1 つの IAM ロールを作成することをお勧めします。

App Studio に Amazon Bedrock へのアクセスを許可する IAM ロールを作成するには

1. [IAM ロールを作成する権限を持つユーザーを使用して IAM コンソールにサインインします。](#)で作成した管理ユーザーを使用することをお勧めします[AWS リソースを管理するための管理ユーザーを作成する](#)。
2. コンソールのナビゲーションペインで、[ロール]、[ロールの作成] の順に選択します。
3. 信頼されたエンティティタイプで、カスタム信頼ポリシーを選択します。
4. デフォルトポリシーを次のポリシーに置き換えて、App Studio アプリケーションがアカウントでこのロールを引き受けることを許可します。

ポリシーで次のプレースホルダーを置き換える必要があります。使用する値は、App Studio のアカウント設定ページにあります。

- **111122223333** を、App Studio インスタンスのセットアップに使用したアカウントの AWS アカウント番号に置き換えます。これは、App Studio インスタンスのアカウント設定に AWS アカウント ID として表示されます。
- **11111111-2222-3333-4444-5555555555** を、App Studio インスタンスのアカウント設定でインスタンス ID としてリストされている App Studio インスタンス ID に置き換えます。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::111122223333:root"
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "StringEquals": {
          "aws:PrincipalTag/IsAppStudioAccessRole": "true",
          "sts:ExternalId":
            "11111111-2222-3333-4444-5555555555"
        }
      }
    }
  ]
}
```

[次へ] を選択します。

5. アクセス許可の追加で、前のステップで作成したポリシー () を検索して選択します **BedrockAccessForAppStudio**。ポリシーの横にある + を選択すると、ポリシーが展開され、ポリシーによって付与されたアクセス許可が表示され、チェックボックスを選択するとポリシーが選択されます。

[次へ] を選択します。

- 名前、レビュー、および作成ページで、ロール名と説明を指定します。
- ステップ 3: タグを追加するで、新しいタグを追加を選択して次のタグを追加して App Studio アクセスを提供します。
 - キー: `IsAppStudioDataAccessRole`
 - 値: `true`
- ロールの作成を選択し、生成された Amazon リソースネーム (ARN) を書き留めます。次のステップで App Studio で Amazon Bedrock コネクタを作成するときに必要になります。

ステップ 3: Amazon Bedrock コネクタを作成する

Amazon Bedrock リソースと IAM ポリシーとロールを設定したら、その情報を使用して、ビルダーがアプリケーションを Amazon Bedrock に接続するために使用できるコネクタを App Studio に作成します。

Note

コネクタを作成するには、App Studio に管理者ロールが必要です。

Amazon Bedrock のコネクタを作成するには

- App Studio に移動します。
- 左側のナビゲーションペインで、「管理」セクションの「コネクタ」を選択します。既存のコネクタのリストとそれぞれの詳細を示すページが表示されます。
- + コネクタの作成を選択します。
- コネクタタイプのリストからその他の AWS サービスを選択します。
- 次のフィールドに入力してコネクタを設定します。
 - 名前: Amazon Bedrock コネクタの名前を入力します。
 - 説明: Amazon Bedrock コネクタの説明を入力します。
 - IAM ロール: で作成した IAM ロールから Amazon リソースネーム (ARN) を入力します [ステップ 2b: App Studio に Amazon Bedrock へのアクセスを許可する IAM ロールを作成する](#)。IAM の詳細については、[IAM ユーザーガイド](#) を参照してください。

- サービス: Bedrock ランタイムを選択します。

Note

Bedrock ランタイムは Amazon Bedrock でホストされているモデルの推論リクエストを行うために使用されますが、Bedrock はモデルの管理、トレーニング、デプロイに使用されます。

- リージョン: Amazon Bedrock リソースがある AWS リージョンを選択します。
6. [作成] を選択します。
 7. 新しく作成されたコネクタがコネクタリストに表示されます。

Amazon Simple Email Service に接続する

App Studio を Amazon SES に接続して、ビルダーがアプリから E メール通知を送信できるようにするには、次の手順を実行する必要があります。

1. [ステップ 1: Amazon SES リソースを設定する](#)
2. [ステップ 2: 適切な Amazon SES アクセス許可を持つ IAM ポリシーとロールを作成する](#)
3. [ステップ 3: Amazon SES コネクタを作成する](#)

ステップ 1: Amazon SES リソースを設定する

まだ設定していない場合は、まず Amazon SES を使用して E メールを送信するように設定する必要があります。Amazon SES の設定の詳細については、[「Amazon Simple Email Service デベロッパーガイド」](#)の「Amazon Simple Email Service の開始方法」を参照してください。

ステップ 2: 適切な Amazon SES アクセス許可を持つ IAM ポリシーとロールを作成する

App Studio で Amazon SES リソースを使用するには、管理者は IAM ロールを作成して、リソースへのアクセス許可を App Studio に付与する必要があります。IAM ロールは、App Studio アプリで使用できる Amazon SES 関数またはリソースを制御します。

サービスおよびポリシーごとに少なくとも 1 つの IAM ロールを作成することをお勧めします。

ステップ 2a: 適切な Amazon SES アクセス許可を持つ IAM ポリシーを作成する

App Studio で作成して使用する IAM ポリシーには、アプリケーションがベストプラクティスに従うための適切なリソースに対する最小限のアクセス許可のみを含める必要があります。

適切な Amazon SES アクセス許可を持つ IAM ポリシーを作成するには

1. [IAM ポリシーを作成する権限を持つユーザーを使用して IAM コンソールにサインインします。](#)で作成した管理ユーザーを使用することをお勧めします[AWS リソースを管理するための管理ユーザーを作成する](#)。
2. 左側のナビゲーションペインで、ポリシーを選択します。
3. [Create policy] (ポリシーの作成) を選択します。
4. [ポリシーエディタ] セクションで、[JSON] オプションを選択します。
5. 次の JSON ポリシードキュメントに入力または貼り付けます。

Note

以下のポリシーは、ワイルドカード (*) を使用するすべての Amazon SES リソースに適用されます*。ベストプラクティスとして、ワイルドカードを App Studio で使用するリソースの Amazon リソースネーム (ARN) に置き換える必要があります。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "VisualEditor0",
      "Effect": "Allow",
      "Action": "ses:SendEmail",
      "Resource": "*"
    }
  ]
}
```

6. [次へ] を選択します。
7. 確認と作成ページで、**SESTForAppStudioPolicy**、説明 (オプション) などのポリシー名を指定します。
8. [ポリシーの作成] を選択し、ポリシーを作成します。

ステップ 2b: App Studio に Amazon SES へのアクセスを許可する IAM ロールを作成する

次に、以前に作成したポリシーを使用する IAM ロールを作成します。App Studio は、このポリシーを使用して Amazon SES にアクセスします。

App Studio に Amazon SES へのアクセスを許可する IAM ロールを作成するには

1. [IAM ロールを作成する権限を持つユーザーを使用して IAM コンソールにサインイン](#)します。で作成した管理ユーザーを使用することをお勧めします[AWS リソースを管理するための管理ユーザーを作成する](#)。
2. 左側のナビゲーションペインで、ロールを選択します。
3. [ロールの作成] を選択してください。
4. 信頼されたエンティティタイプで、カスタム信頼ポリシーを選択します。
5. デフォルトポリシーを次のポリシーに置き換えて、App Studio アプリケーションがアカウントでこのロールを引き受けることを許可します。

ポリシーで次のプレースホルダーを置き換える必要があります。使用する値は、App Studio のアカウント設定ページにあります。

- **111122223333** を、App Studio インスタンスのセットアップに使用したアカウントの AWS アカウント番号に置き換えます。これは、App Studio インスタンスのアカウント設定に AWS アカウント ID として表示されます。
- **11111111-2222-3333-4444-5555555555** を、App Studio インスタンスのアカウント設定でインスタンス ID としてリストされている App Studio インスタンス ID に置き換えます。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::111122223333:root"
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "StringEquals": {
```

```
        "aws:PrincipalTag/IsAppStudioAccessRole": "true",
        "sts:ExternalId":
            "11111111-2222-3333-4444-555555555555"
    }
}
]
```

[次へ] を選択します。

6. アクセス許可の追加で、前のステップで作成したポリシー () を検索して選択します **SESTForAppStudioPolicy**。ポリシーの横にある + を選択すると、ポリシーが展開され、ポリシーによって付与されたアクセス許可が表示され、チェックボックスを選択するとポリシーが選択されます。

[次へ] を選択します。

7. 名前、レビュー、および作成ページで、ロール名と説明を指定します。
8. ステップ 3: タグを追加するで、新しいタグを追加を選択して次のタグを追加して App Studio アクセスを提供します。
 - キー: IsAppStudioDataAccessRole
 - 値: true
9. ロールの作成を選択し、生成された Amazon リソースネーム (ARN) を書き留めます。 [App Studio で Amazon SES コネクタを作成する](#) ときに必要になります。

ステップ 3: Amazon SES コネクタを作成する

Amazon SES と IAM ポリシーとロールを設定したら、その情報を使用して、ビルダーがアプリで Amazon SES を使用するために使用できるコネクタを App Studio に作成します。

Note

コネクタを作成するには、App Studio に管理者ロールが必要です。

Amazon SES のコネクタを作成するには

1. App Studio に移動します。

2. 左側のナビゲーションペインで、「管理」セクションの「コネクタ」を選択します。既存のコネクタのリストとそれぞれの詳細を示すページが表示されます。
3. + コネクタの作成を選択します。
4. コネクタタイプのリストからその他の AWS サービスを選択します。
5. 次のフィールドに入力してコネクタを設定します。
 - 名前: Amazon SES コネクタの名前を入力します。
 - 説明: Amazon SES コネクタの説明を入力します。
 - IAM ロール: で作成した IAM ロールから Amazon リソースネーム (ARN) を入力します [ステップ 2b: App Studio に Amazon SES へのアクセスを許可する IAM ロールを作成する](#)。IAM の詳細については、[IAM ユーザーガイド](#) を参照してください。
 - サービス: Simple Email Service を選択します。
 - リージョン: Amazon SES リソースがある AWS リージョンを選択します。
6. [作成] を選択します。
7. 新しく作成されたコネクタがコネクタリストに表示されます。

その他の AWS サービスコネクタを使用して AWS サービスに接続する

App Studio には、特定の AWS サービスに固有のコネクタが用意されていますが、その他の AWS サービスコネクタを使用して他の AWS サービスに接続することもできます。

Note

利用可能な場合は、AWS サービスに固有のコネクタを使用することをお勧めします。

App Studio を AWS サービスに接続して、ビルダーがアプリケーションのサービスのリソースにアクセスして使用できるようにするには、次の手順を実行する必要があります。

1. [App Studio に AWS リソースへのアクセスを許可する IAM ロールを作成する](#)
2. [その他の AWS サービスコネクタを作成する](#)

App Studio に AWS リソースへのアクセスを許可する IAM ロールを作成する

App Studio で AWS サービスとリソースを使用するには、管理者は IAM ロールを作成して、リソースへのアクセス許可を App Studio に付与する必要があります。IAM ロールは、ビルダーがアクセス

できるリソースの範囲と、リソースに対して呼び出すことができるオペレーションを制御します。サービスおよびポリシーごとに少なくとも 1 つの IAM ロールを作成することをお勧めします。

App Studio に AWS リソースへのアクセスを許可する IAM ロールを作成するには

1. [IAM ロールを作成する権限を持つユーザーを使用して IAM コンソールにサインイン](#)します。で作成した管理ユーザーを使用することをお勧めします[AWS リソースを管理するための管理ユーザーを作成する](#)。
2. コンソールのナビゲーションペインで、[ロール]、[ロールの作成] の順に選択します。
3. 信頼されたエンティティタイプで、カスタム信頼ポリシーを選択します。
4. デフォルトポリシーを次のポリシーに置き換えて、App Studio アプリケーションがアカウントでこのロールを引き受けることを許可します。

ポリシーで次のプレースホルダーを置き換える必要があります。使用する値は、App Studio のアカウント設定ページにあります。

- **111122223333** を、App Studio インスタンスのセットアップに使用したアカウントの AWS アカウント番号に置き換えます。これは、App Studio インスタンスのアカウント設定に AWS アカウント ID として表示されます。
- **11111111-2222-3333-4444-5555555555** を、App Studio インスタンスのアカウント設定でインスタンス ID としてリストされている App Studio インスタンス ID に置き換えます。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::111122223333:root"
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "StringEquals": {
          "aws:PrincipalTag/IsAppStudioAccessRole": "true",
          "sts:ExternalId":
            "11111111-2222-3333-4444-5555555555"
        }
      }
    }
  ]
}
```

```
}  
    }  
  }  
}
```

[次へ] を選択します。

5. アクセス許可の追加で、ロールに適切なアクセス許可を付与するポリシーを検索して選択します。ポリシーの横にある + を選択すると、ポリシーが展開され、ポリシーによって付与されたアクセス許可が表示され、チェックボックスを選択するとポリシーが選択されます。IAM の詳細については、[IAM ユーザーガイド](#) を参照してください。

[次へ] を選択します。

6. ロールの詳細で、名前と説明を入力します。
7. ステップ 3: タグを追加する で、新しいタグを追加 を選択して次のタグを追加して App Studio アクセスを提供します。
 - キー: IsAppStudioDataAccessRole
 - 値: true
8. ロールの作成を選択し、生成された Amazon リソースネーム (ARN) を書き留めます。[App Studio でその他の AWS サービスコネクタを作成する](#) ときに必要になります。

その他の AWS サービスコネクタを作成する

IAM ロールを設定したら、その情報を使用して、ビルダーがアプリをサービスやリソースに接続するために使用できるコネクタを App Studio に作成します。

Note

コネクタを作成するには、App Studio に管理者ロールが必要です。

その他の AWS サービスコネクタを使用して サービスに接続するには AWS

1. App Studio に移動します。
2. 左側のナビゲーションペインで、「管理」セクションの「コネクタ」を選択します。
3. + コネクタの作成を選択します。

- サポートされている AWS サービスリストのコネクタセクションでその他のサービスを選択します。AWS
- 次のフィールドに入力して、AWS サービスコネクタを設定します。
 - 名前: コネクタの名前を指定します。
 - 説明: コネクタの説明を入力します。
 - IAM ロール: で作成された IAM ロールから Amazon リソースネーム (ARN) を入力します [App Studio に AWS リソースへのアクセスを許可する IAM ロールを作成する](#)。
 - サービス: App Studio に接続する AWS サービスを選択します。
 - リージョン: AWS リソースが配置されている AWS リージョンを選択します。
- [作成] を選択します。新しく作成されたコネクタがコネクタリストに表示されます。

CMKs

このトピックでは、[AWS KMS カスタマーマネージドキー \(CMK\)](#) を使用して暗号化されたデータソースへの App Studio のセットアップと接続について説明します。

目次

- [暗号化されたマネージドデータストレージテーブルの使用](#)
- [暗号化された DynamoDB テーブルの使用](#)

暗号化されたマネージドデータストレージテーブルの使用

App Studio アプリのマネージドストレージエンティティが使用する DynamoDB テーブルを暗号化するには、次の手順に従います。マネージドデータエンティティの詳細については、「」を参照してください [AWS App Studio のマネージドデータエンティティ](#)。

暗号化されたマネージドデータストレージテーブルを使用するには

- 必要に応じて、App Studio のアプリケーションにマネージドデータエンティティを作成します。詳細については、「[App Studio マネージドデータソースを使用したエンティティの作成](#)」を参照してください。
- 次の手順を実行して、CMK を使用してテーブルデータを暗号化および復号するアクセス許可を持つポリシーステートメントを AppStudioManagedStorageDDBAccess IAM ロールに追加します。

- a. IAM コンソール (<https://console.aws.amazon.com/iam/>) を開きます。

⚠ Important

App Studio インスタンスの作成に使用したのと同じアカウントを使用する必要があります。

- b. IAM コンソールのナビゲーションペインで [ロール] を選択します。
 - c. AppStudioManagedStorageDDBAccess を選択してください。
 - d. アクセス許可ポリシーで、アクセス許可の追加を選択し、インラインポリシーの作成を選択します。
 - e. JSON を選択し、内容を次のポリシーに置き換えます。
 - **111122223333** を、App Studio インスタンスのセットアップに使用したアカウントの AWS アカウント番号に置き換えます。これは、App Studio インスタンスのアカウント設定に AWS アカウント ID として表示されます。
 - **CMK_id** を CMK ID に置き換えます。これを見つけるには、[「キー ID とキー ARN を検索する」](#) を参照してください。
3. 次の手順を実行して、App Studio マネージドデータエンティティで使用される DynamoDB テーブルを暗号化します。
 - a. <https://console.aws.amazon.com/dynamodbv2/> で Amazon DynamoDB コンソールを開きます。
 - b. 暗号化するテーブルを選択します。テーブル名は、App Studio の対応するエンティティの接続タブにあります。
 - c. [Additional settings] (追加設定) を選択します。
 - d. 暗号化で、暗号化の管理を選択します。
 - e. アカウントに保存され、ユーザーが所有および管理している を選択し、CMK を選択します。
 4. アプリを再公開し、データの読み取りと書き込みがテスト環境と本番稼働環境の両方で機能し、別のエンティティでこのテーブルが期待どおりに機能することを確認して、変更をテストします。

Note

新しく追加されたマネージドデータエンティティはデフォルトで DynamoDB マネージドキーを使用し、前のステップに従って CMK を使用してに更新する必要があります。

暗号化された DynamoDB テーブルの使用

App Studio アプリで使用する暗号化された DynamoDB テーブルを設定するには、次の手順に従います。

暗号化された DynamoDB テーブルを使用するには

1. 以下の変更[ステップ 1: DynamoDB リソースを作成して設定する](#)を加えて、「」の手順に従います。
 - 暗号化するテーブルを設定します。詳細については、「[Amazon DynamoDB デベロッパーガイド](#)」の「[新しいテーブルの暗号化キーの指定](#)」を参照してください。 DynamoDB
2. 「」の手順に従って[ステップ 2: 適切な DynamoDB アクセス許可を持つ IAM ポリシーとロールを作成する](#)、新しいポリシーステートメントを追加して新しいロールのアクセス許可ポリシーを更新し、次の手順を実行して CMK を使用してテーブルデータを暗号化および復号できるようにします。
 - a. 必要に応じて、IAM コンソールでロールに移動します。
 - b. アクセス許可ポリシーで、アクセス許可の追加を選択し、インラインポリシーの作成を選択します。
 - c. JSON を選択し、内容を次のポリシーに置き換え、以下を置き換えます。
 - `team_account_id` を、アカウント設定にある App Studio チーム ID に置き換えます。
 - `CMK_id` を CMK ID に置き換えます。これを見つけるには、「[キー ID とキー ARN を検索する](#)」を参照してください。
3. 「」の手順に従って[DynamoDB コネクタを作成する](#)、前に作成したロールを使用してコネクタを作成します。
4. DynamoDB コネクタとテーブルを使用するアプリをテストまたは本番稼働用に公開して、設定をテストします。データの読み取りと書き込みが機能し、このテーブルを使用して別のエンティティを作成することも機能します。

Note

新しい DynamoDB テーブルが作成されたら、前の手順に従って CMK を使用して暗号化されるように設定する必要があります。

サードパーティーサービスに接続する

トピック

- [OpenAPI Connector と API Connector](#)
- [サードパーティーAPIs に接続する \(汎用\)](#)
- [OpenAPI を使用して サービスに接続する](#)
- [Salesforce に接続する](#)

OpenAPI Connector と API Connector

App Studio アプリケーションからサードパーティーのサービスに API リクエストを送信するには、アプリケーションでサービスでの認証と API コールの設定に使用するコネクタを作成して設定する必要があります。App Studio には、これを実現するために API Connector と OpenAPI Connector コネクタタイプの両方が用意されています。以下に説明します。

- API Connector: 任意のタイプの REST API の認証とリクエスト情報を設定するために使用されます。
- OpenAPI Connector: OpenAPI 仕様 (OAS) OpenAPI を採用している APIs の認証とリクエスト情報を設定するために使用されます。OAS に準拠する APIs には、標準化、セキュリティ、ガバナンス、ドキュメントなど、いくつかの利点があります。

App Studio では、OAS に準拠する APIs OpenAPI Connector に を使用し、OpenAPI 仕様ファイルを提供することをお勧めします。OpenAPI の詳細については、Swagger ドキュメントの[OpenAPI とは](#)」を参照してください。

サードパーティーAPIs に接続する (汎用)

App Studio で汎用 API Connector を作成するには、次の手順に従います。API Connector は、App Studio アプリにサードパーティーのサービス、リソース、またはオペレーションへのアクセスを提供するために使用されます。

API Connector を使用してサードパーティーのサービスに接続するには

1. 左側のナビゲーションペインで、管理セクションでコネクタを選択します。既存のコネクタのリストとそれぞれの詳細を示すページが表示されます。
2. + コネクタの作成を選択します。
3. API Connector を選択します。次に、次のフィールドに入力してコネクタを設定します。
4. コネクタ名: コネクタの名前を指定します。
5. コネクタの説明: コネクタの説明を入力します。
6. ベース URL: サードパーティー接続のウェブサイトまたはホスト。例えば、`www.slack.com`。
7. 認証方法: ターゲットサービスで認証する方法を選択します。
 - なし: 認証なしでターゲットサービスにアクセスします。
 - 基本: 接続先のサービスから取得したユーザー名とパスワードを使用して、ターゲットサービスにアクセスします。
 - ベアラートークン: サービスのユーザーアカウントまたは API 設定から取得した認証トークンのトークン値を使用して、ターゲットサービスにアクセスします。
 - OAuth 2.0: OAuth 2.0 プロトコルを使用してターゲットサービスにアクセスします。これにより、App Studio は認証情報や ID を共有せずにサービスとリソースにアクセスできるようになります。OAuth 2.0 認証方法を使用するには、まず App Studio を表す接続先のサービスからアプリケーションを作成して、必要な情報を取得する必要があります。この情報を使用して、次のフィールドに入力します。
 - a. クライアント認証情報フロー: アプリケーションがユーザーとやり取りすることなく独自に動作するsystem-to-systemやり取りに最適です。たとえば、ユーザーが追加した新しいレコードに基づいて Salesforce レコードを自動的に更新する CRM アプリや、トランザクションデータを取得してレポートに表示するアプリなどです。
 1. クライアント ID で、ターゲットサービスで作成された OAuth アプリから取得した ID を入力します。
 2. クライアントシークレットで、ターゲットサービスで作成された OAuth アプリから取得したシークレットを入力します。
 3. アクセストークン URL で、ターゲットサービスで作成された OAuth アプリから取得したトークン URL を入力します。
 4. 必要に応じて、スコープにアプリケーションのスコープを入力します。スコープは、アプリケーションに必要なアクセス許可またはアクセスレベルです。ターゲッ

トサービスの API ドキュメントを参照してスコープを理解し、App Studio アプリに必要なスコープのみを設定します。

接続の検証を選択して、認証と接続をテストします。

- b. 認可コードフロー: ユーザーに代わって動作する必要があるアプリケーションに最適です。たとえば、ユーザーがログインしてサポートチケットを表示および更新するカスタマーサポートアプリ、または各チームメンバーがログインして販売データを表示および管理するセールスアプリなどです。
 1. クライアント ID に、ターゲットサービスで作成された OAuth アプリから取得した ID を入力します。
 2. クライアントシークレットで、ターゲットサービスで作成された OAuth アプリから取得したシークレットを入力します。
 3. 認可 URL に、ターゲットサービスの認可 URL を入力します。
 4. アクセストークン URL で、ターゲットサービスで作成された OAuth アプリから取得したトークン URL を入力します。
 5. 必要に応じて、スコープにアプリケーションのスコープを入力します。スコープは、アプリケーションに必要なアクセス許可またはアクセスレベルです。ターゲットサービスの API ドキュメントを参照してスコープを理解し、App Studio アプリに必要なスコープのみを設定します。
8. ヘッダー: リクエストまたはレスポンスに関するメタデータを提供するために使用される HTTP ヘッダーを追加します。キーと値の両方を追加することも、ビルダーがアプリケーションで値を提供できるキーのみを指定することもできます。
9. クエリパラメータ: リクエスト URL の一部としてオプション、フィルター、またはデータを渡すために使用されるクエリパラメータを追加します。ヘッダーと同様に、キーと値の両方を指定することも、ビルダーがアプリケーションで値を提供できるキーのみを指定することもできます。
10. [作成] を選択します。新しく作成されたコネクタがコネクタリストに表示されます。

コネクタが作成されると、ビルダーはそれをアプリケーションで使用できます。

OpenAPI を使用して サービスに接続する

OpenAPI を使用して App Studio をサービスに接続し、ビルダーがリクエストを送信し、サービスからレスポンスを受信するアプリケーションを構築できるようにするには、次の手順を実行します。

1. [OpenAPI 仕様ファイルを取得し、サービス情報を収集する](#)
2. [OpenAPI コネクタを作成する](#)

OpenAPI 仕様ファイルを取得し、サービス情報を収集する

OpenAPI を使用してサービスを App Studio に接続するには、次の手順を実行します。

1. App Studio に接続するサービスに移動し、OpenAPI 仕様 JSON ファイルを見つけます。

Note

App Studio は、バージョン OpenAPI 仕様バージョン 3.0.0 以降に準拠した OpenAPI 仕様ファイルをサポートしています。

2. 以下を含む OpenAPI コネクタの設定に必要なデータを収集します。
 - サービスに接続するためのベース URL。
 - トークンやユーザー名/パスワードなどの認証認証情報。
 - 該当する場合は、任意のヘッダー。
 - 該当する場合は、任意のクエリパラメータ。

OpenAPI コネクタを作成する

OpenAPI 用のコネクタを作成するには

1. App Studio に移動します。
2. 左側のナビゲーションペインで、「管理」セクションの「コネクタ」を選択します。既存のコネクタのリストとそれぞれの詳細を示すページが表示されます。
3. + コネクタの作成を選択します。
4. コネクタタイプのリストから OpenAPI Connector を選択します。次に、次のフィールドに入力してコネクタを設定します。
5. 名前: OpenAPI コネクタの名前を入力します。
6. 説明: OpenAPI コネクタの説明を入力します。
7. ベース URL: サービスに接続するためのベース URL を入力します。
8. 認証方法: ターゲットサービスで認証する方法を選択します。

- なし: 認証なしでターゲットサービスにアクセスします。
- 基本: 接続先のサービスから取得したユーザー名とパスワードを使用して、ターゲットサービスにアクセスします。
- ベアラートークン: サービスのユーザーアカウントまたは API 設定から取得した認証トークンのトークン値を使用して、ターゲットサービスにアクセスします。
- OAuth 2.0: OAuth 2.0 プロトコルを使用してターゲットサービスにアクセスします。これにより、App Studio は認証情報や ID を共有せずにサービスやリソースにアクセスできます。OAuth 2.0 認証方法を使用するには、まず App Studio を表す接続先のサービスからアプリケーションを作成して、必要な情報を取得する必要があります。この情報を使用して、次のフィールドに入力します。
 - a. クライアント認証情報フロー:
 1. クライアント ID で、ターゲットサービスの ID を入力します。
 2. クライアントシークレットで、ターゲットサービスのシークレットを入力します。
 3. アクセストークン URL で、ターゲットサービスのトークン URL を入力します。
 4. 必要に応じて、スコープにアプリケーションのスコープを入力します。スコープは、アプリケーションに必要なアクセス許可またはアクセスレベルです。ターゲットサービスの API ドキュメントを参照してスコープを理解し、App Studio アプリに必要なスコープのみを設定します。

呼び出しごとにサービスで送信される変数を追加し、接続の検証を選択して認証と接続をテストします。

- b. 認可コードフロー:
 1. クライアント ID で、ターゲットサービスの ID を入力します。
 2. クライアントシークレットで、ターゲットサービスのシークレットを入力します。
 3. 認可 URL に、ターゲットサービスの認可 URL を入力します。
 4. アクセストークン URL で、ターゲットサービスのトークン URL を入力します。
 5. 必要に応じて、スコープにアプリケーションのスコープを入力します。スコープは、アプリケーションに必要なアクセス許可またはアクセスレベルです。ターゲットサービスの API ドキュメントを参照してスコープを理解し、App Studio アプリに必要なスコープのみを設定します。
9. 変数: 呼び出しごとにサービスに送信する変数を追加します。設定中に追加された変数は安全に保存され、接続を使用するアプリケーションのランタイム中にのみアクセスされます。

10. ヘッダー: リクエストまたはレスポンスに関するメタデータを提供するために使用される HTTP ヘッダーを追加します。キーと値の両方を追加することも、ビルダーがアプリケーションで値を提供できるキーのみを指定することもできます。
11. クエリパラメータ: リクエスト URL の一部としてオプション、フィルター、またはデータを渡すために使用されるクエリパラメータを追加します。ヘッダーと同様に、キーと値の両方を指定することも、ビルダーがアプリケーションで値を提供できるキーのみを指定することもできます。
12. OpenAPI 仕様ファイル: OpenAPI 仕様 JSON ファイルをドラッグアンドドロップしてアップロードするか、ファイルを選択してローカルファイルシステムを移動し、アップロードするファイルを選択します。

追加すると、ファイルが処理され、使用可能なオプションのリストが表示されます。コネクタに必要なオペレーションを選択します。
13. [作成] を選択します。新しく作成されたコネクタがコネクタリストに表示されます。

コネクタが作成されると、ビルダーはそれをアプリケーションで使用できます。

Salesforce に接続する

App Studio を Salesforce に接続して、ビルダーがアプリケーションで Salesforce リソースにアクセスして使用できるようにするには、Salesforce で接続されたアプリケーションを作成して設定し、App Studio で Salesforce コネクタを作成する必要があります。

Salesforce を App Studio に接続するには

1. App Studio のナビゲーションペインで、「管理」セクションの「コネクタ」を選択します。既存のコネクタのリストとそれぞれの詳細を示すページが表示されます。
2. + コネクタの作成を選択します。
3. コネクタタイプのリストから Salesforce を選択して、コネクタ作成ページを開きます。
4. リダイレクト URL を書き留めておきます。この URL を使用して、次の手順で Salesforce を設定します。
5. 次のステップでは、Salesforce で接続されたアプリケーションを作成します。別のタブまたはウィンドウで、Salesforce インスタンスに移動します。
6. クイック検索ボックスで、App Manager を検索 **App Manager** して選択します。
7. 新しい接続されたアプリを選択します。

8. 接続されたアプリケーション名と API 名に、アプリケーションの名前を入力します。App Studio アプリ名と一致する必要はありません。
9. 必要に応じて連絡先情報を入力します。
10. API (OAuth 設定の有効化) セクションで、OAuth 設定の有効化を有効にします。
11. コールバック URL に、先ほど App Studio から書き留めたリダイレクト URL を入力します。
12. 選択した OAuth スコープで、リストから必要なアクセス許可スコープを追加します。App Studio は Salesforce REST APIs とやり取りして、アカウント、ケース、連絡先、リード、オポチュニティの 5 つのオブジェクトに対して CRUD オペレーションを実行できます。フルアクセス (フルアクセス) を追加して、App Studio アプリに関連するすべてのアクセス許可またはスコープがあることを確認することをお勧めします。
13. サポートされている認可フローのコード交換 (PKCE) 拡張に必要な証明キーオプションを無効にします。PKCE は App Studio ではサポートされていません。
14. ウェブサーバーフローのシークレットを要求し、更新トークンフローのシークレットを有効にして、のベストプラクティスに従います。
15. App Studio は、次の認証フローの両方をサポートしています。
 - クライアント認証情報フロー: アプリケーションがユーザーとのやり取りなしで独自に動作する server-to-server やり取りに最適です。たとえば、Salesforce アクセスを持たない臨時従業員のチームのすべてのリード情報を一覧表示します。
 - 認可コードフロー: 個人データへのアクセスやアクションなど、ユーザーに代わって動作するアプリケーションに適しています。例えば、このアプリを通じて他のタスクを実行するために、販売マネージャーが調達または所有する各セールスマネージャーのリードを一覧表示します。
 - クライアント認証情報フローの場合:
 - a. クライアント認証情報フローの有効化を有効にします。メッセージを確認して確認します。
 - b. アプリを保存します。
 - c. フローにはユーザー操作はありませんが、実行ユーザーを選択する必要があります。実行ユーザーを選択すると、Salesforce はユーザーに代わってアクセストークンを返します。
 1. App Manager で、アプリのリストから App Studio アプリの矢印を選択し、管理を選択します。

2. ポリシーの編集を選択する
3. クライアント認証情報フローで、適切なユーザーを追加します。
 - 認可コードフローで、認可コードと認証情報フローを有効にする
16. Salesforce はクライアント ID とクライアントシークレットを提供します。これは、次の手順で App Studio でコネクタを設定するために使用する必要があります。
 - a. App Manager で、App Studio アプリの矢印を選択し、表示を選択します。
 - b. API (OAuth 設定の有効化) セクションで、コンシューマーの詳細の管理 を選択します。これにより、確認キーの E メールが送信される場合があります、確認のために入力する必要があります。
 - c. コンシューマーキー (クライアント ID) とコンシューマーシークレット (クライアントシークレット) を書き留めます。
17. App Studio に戻り、次のフィールドに入力してコネクタを設定および作成します。
18. Name に、Salesforce コネクタの名前を入力します。
19. 説明 に、Salesforce コネクタの説明を入力します。
20. ベース URL に、Salesforce インスタンスのベース URL を入力します。ホスト名を Salesforce インスタンス名に置き換えると `https://hostname.salesforce.com/services/data/v60.0`、 のようになります。
21. 認証方法で、OAuth 2.0 が選択されていることを確認します。
22. OAuth 2.0 フローで、OAuth 認証方法を選択し、関連フィールドに入力します。
 - system-to-system統合のために、独自に動作するアプリケーションで使用するために、クライアント認証情報フローを選択します。
 - a. クライアント ID に、Salesforce から以前に取得したコンシューマーキーを入力します。
 - b. クライアントシークレットに、Salesforce から以前に取得したコンシューマーシークレットを入力します。
 - c. アクセストークン URL に、OAuth 2.0 トークンエンドポイントを入力します。ホスト名を Salesforce インスタンス名に置き換えると `https://hostname/services/oauth2/token`、 のようになります。詳細については、[Salesforce OAuth Endpoints](#) ドキュメントを参照してください。
 - d. 接続の検証を選択して、認証と接続をテストします。
 - ユーザーに代わって動作するアプリケーションで使用するために認可コードフローを選択します。

- a. クライアント ID に、Salesforce から以前に取得したコンシューマーキーを入力します。
 - b. クライアントシークレットに、Salesforce から以前に取得したコンシューマーシークレットを入力します。
 - c. 認可 URL に認可エンドポイントを入力します。ホスト名を Salesforce インスタンス名に置き換えると `https://hostname/services/oauth2/authorize`、のようになります。詳細については、[Salesforce OAuth Endpoints](#) ドキュメントを参照してください。
 - d. アクセストークン URL に、OAuth 2.0 トークンエンドポイントを入力します。ホスト名を Salesforce インスタンス名に置き換えると `https://hostname/services/oauth2/token`、のようになります。詳細については、[Salesforce OAuth Endpoints](#) ドキュメントを参照してください。
23. オペレーションで、コネクタがサポートする Salesforce オペレーションを選択します。このリストのオペレーションは事前定義されており、共通オブジェクトからのレコードの作成、取得、更新、削除など、Salesforce 内の一般的なタスクを表します。
24. [作成] を選択します。新しく作成されたコネクタがコネクタリストに表示されます。

コネクタの表示、編集、削除

既存のコネクタを表示、編集、または削除するには

1. ナビゲーションペインで、「管理」セクションの「コネクタ」を選択します。既存のコネクタのリストと各コネクタの以下の詳細を示すページが表示されます。
 - 名前: 作成時に提供されたコネクタの名前。
 - 説明: 作成時に提供されたコネクタの説明。
 - 接続先: コネクタが App Studio に接続しているサービス。API の値は、サードパーティーのサービスへの接続を表します。
 - 作成者: コネクタを作成したユーザー。
 - 作成日: コネクタが作成された日付。
2. コネクタの詳細を表示したり、コネクタを編集または削除したりするには、次の手順を使用します。
 - 特定のコネクタの詳細については、そのコネクタの表示 を選択します。

- コネクタを編集するには、表示の横にあるドロップダウンメニューを選択し、編集を選択します。
- コネクタを削除するには、表示の横にあるドロップダウンメニューを選択し、削除を選択します。

App Studio インスタンスの削除

このトピックの手順を使用して、App Studio インスタンスを削除します。App Studio で使用する他のサービスでリソースを作成した場合は、料金が発生しないように、必要に応じてリソースを確認して削除します。

App Studio インスタンスは、次の理由で削除できます。

- App Studio が不要になりました。
- 別の AWS リージョンに App Studio インスタンスを作成する場合。App Studio は一度に 1 つのリージョンにのみインスタンスを持つことをサポートしているため、既存のインスタンスを削除して別のインスタンスを作成する必要があります。

Warning

App Studio インスタンスを削除すると、アプリケーションやコネクタなどのすべての App Studio リソースも削除されます。インスタンスの削除は元に戻すことができません。

App Studio インスタンスを削除するには

1. <https://console.aws.amazon.com/appstudio/> で App Studio コンソールを開きます。
2. App Studio インスタンスが存在するリージョンを選択します。
3. ナビゲーションペインで、インスタンスを選択します。
4. アクション を選択して、追加のインスタンスアクションを含むドロップダウンを開きます。
5. App Studio インスタンスの削除を選択します。
6. 「**confirm**」と入力し、[削除] を選択します。
7. インスタンスの削除が処理されるまでに時間がかかる場合があります。削除されると、確認メールが届きます。E メールを受信したら、必要に応じて別のインスタンスを作成できます。

Builder ドキュメント

以下のトピックには、アプリケーションを作成、編集、公開する App Studio のユーザーに役立つ情報が含まれています。

トピック

- [チュートリアル](#)
- [生成 AI を使用した App Studio アプリの構築](#)
- [アプリケーションの作成、編集、削除](#)
- [アプリケーションのプレビュー、公開、共有](#)
- [ページとコンポーネント: アプリケーションのユーザーインターフェイスを構築する](#)
- [自動化とアクション: アプリのビジネスロジックを定義する](#)
- [エンティティとデータアクション: アプリのデータモデルを設定する](#)
- [ページとオートメーションのパラメータ](#)
- [JavaScript を使用して App Studio で式を記述する](#)
- [データの依存関係とタイミングに関する考慮事項](#)
- [複数のユーザーによるアプリの構築](#)
- [アプリのコンテンツセキュリティ設定の表示または更新](#)

チュートリアル

トピック

- [Amazon Bedrock を使用して AI テキストサマリアプリを構築する](#)
- [Amazon Simple Storage Service とコンポーネントおよびオートメーションの操作](#)
- [App Studio アプリでの Lambda 関数の呼び出し](#)

Amazon Bedrock を使用して AI テキストサマリアプリを構築する

このチュートリアルでは、Amazon Bedrock を使用してエンドユーザーからのテキスト入力の簡潔な概要を提供するアプリケーションを App Studio で構築します。アプリケーションには、ユーザーが要約したいテキストを入力できるシンプルなユーザーインターフェイスが含まれています。これには、会議メモ、記事の内容、調査結果、またはその他のテキスト情報が含まれます。ユーザーがテキ

ストを入力したら、ボタンを押してテキストを Amazon Bedrock に送信できます。Amazon Bedrock は Claude 3 Sonnet モデルを使用してテキストを処理し、要約されたバージョンを返します。

目次

- [前提条件](#)
- [ステップ 1: IAM ロールと App Studio コネクタを作成して設定する](#)
- [ステップ 2: アプリケーションを作成する](#)
- [ステップ 3: オートメーションを作成して設定する](#)
- [ステップ 4: ページとコンポーネントを作成する](#)
 - [デフォルトページの名前を変更する](#)
 - [ページにコンポーネントを追加する](#)
 - [ページコンポーネントを設定する](#)
- [ステップ 5: アプリケーションをテスト環境に公開する](#)
- [\(オプション\) クリーンアップする](#)

前提条件

開始する前に、以下の前提条件を確認して完了してください。

- AWS App Studio へのアクセス。このチュートリアルでは、コネクタを作成するには管理者ロールが必要です。
- オプション: [AWS App Studio の概念](#)とを確認して[チュートリアル: 空のアプリから構築を開始する](#)、App Studio の重要な概念を理解します。

ステップ 1: IAM ロールと App Studio コネクタを作成して設定する

App Studio に Amazon Bedrock モデルへのアクセスを提供するには、以下を行う必要があります。

1. アプリで使用する Amazon Bedrock モデルを有効にします。このチュートリアルでは Claude 3 Sonnet を使用するため、そのモデルを有効にしてください。
2. Amazon Bedrock への適切なアクセス許可を持つ IAM ロールを作成します。
3. アプリで使用する IAM ロールを使用して App Studio コネクタを作成します。

詳細な手順 [Amazon Bedrock に接続する](#) については、「」を参照してください。ステップに従ってコネクタを作成したら、このチュートリアルに戻ります。

ステップ 2: アプリケーションを作成する

テキストサマリアプリに組み込む App Studio で空のアプリを作成するには、次の手順に従います。

1. App Studio にサインインします。
2. ビルダーハブに移動し、+ アプリケーションの作成を選択します。
3. [最初から開始] を選択します。
4. アプリ名フィールドに、などのアプリの名前を入力します **Text Summarizer**。
5. データソースまたはコネクタを選択するように求められた場合は、このチュートリアルの目的でスキップを選択します。
6. 次へ をクリックして先に進みます。
7. (オプション): App Studio でアプリケーションを構築する方法の概要については、ビデオチュートリアルをご覧ください。
8. 「アプリの編集」を選択すると、アプリケーションスタジオに移動します。

ステップ 3: オートメーションを作成して設定する

App Studio アプリのロジックと動作をオートメーションで定義します。自動化は、アクションと呼ばれる個々のステップ、他のリソースからアクションにデータを渡すために使用されるパラメータ、および他の自動化やコンポーネントで使用できる出力で構成されます。このステップでは、以下を使用して Amazon Bedrock とのやり取りを処理するオートメーションを作成します。

- 入力: ユーザーからオートメーションにテキスト入力を渡すパラメータ。
- アクション: Amazon Bedrock にテキスト入力を送信し、出力テキストの概要を返す 1 つの GenAI プロンプトアクション。
- 出力: アプリで使用できる Amazon Bedrock から処理された概要で構成される自動化出力。

Amazon Bedrock にプロンプトを送信し、要約を処理して返すオートメーションを作成して設定するには

1. キャンバスの上部にあるオートメーションタブを選択します。
2. 「+ オートメーションの追加」を選択します。

3. 右側のパネルで、プロパティを選択します。
4. 鉛筆アイコンを選択してオートメーション名を更新します。**InvokeBedrock** を入力して Enter を押します。
5. 以下のステップを実行して、ユーザーからのテキストプロンプト入力を Amazon Bedrock へのリクエストで使用するオートメーションに渡すために使用されるパラメータをオートメーションに追加します。
 - a. キャンバスのパラメータボックスで + 追加 を選択します。
 - b. [名前] に「**input**」と入力します。
 - c. 説明 に、 などの説明を入力します **Text to be sent to Amazon Bedrock.**
 - d. Type で、String を選択します。
 - e. 追加 を選択してパラメータを作成します。
6. 次の手順を実行して GenAI プロンプトアクションを追加します。
 - a. 右側のパネルで、アクションを選択します。
 - b. GenAI プロンプトを選択してアクションを追加します。
7. 次の手順を実行して、アクションを設定します。
 - a. キャンバスからアクションを選択して、右側のプロパティメニューを開きます。
 - b. 鉛筆アイコン **PromptBedrock** を選択し、名前を入力し、Enter キーを押して、アクションの名前を に変更します。
 - c. Connector で、 で作成されたコネクタを選択します [ステップ 1: IAM ロールと App Studio コネクタを作成して設定する](#)。
 - d. Model で、プロンプトの処理に使用する Amazon Bedrock モデルを選択します。このチュートリアルでは、Claude 3.5 Sonnet を選択します。
 - e. ユーザープロンプトで、 と入力します `{{params.input}}`。これは、前に作成した input パラメータを表し、アプリユーザーによるテキスト入力が含まれます。
 - f. システムプロンプトで、Amazon Bedrock に送信するシステムプロンプトの指示を入力します。このチュートリアルでは、次のように入力します。

You are a highly efficient text summarizer. Provide a concise summary of the prompted text, capturing the key points and main ideas.

- g. リクエスト設定を選択して展開し、次のフィールドを更新します。

- 「温度」に「」と入力します0。テクスチャは、出力のランダム性または創造性を 0~10 のスケールで決定します。数値が大きいほど、レスポンスはよりクリエイティブになります。
 - 最大トークンで、 と入力4096してレスポンスの長さを制限します。
8. この自動化の出力は要約されたテキストになりますが、デフォルトでは自動化は出力を作成しません。次の手順を実行して、オートメーション出力を作成するようにオートメーションを設定します。
- a. 左側のナビゲーションで、InvokeBedrock オートメーションを選択します。
 - b. 右側のプロパティメニューの出力で + 追加を選択します。
 - c. Output に と入力します `{{results.PromptBedrock.text}}`。この式は、processResultsアクションの内容を返します。

ステップ 4: ページとコンポーネントを作成する

App Studio では、各ページは、ユーザーが操作するアプリケーションのユーザーインターフェイス (UI) の画面を表します。これらのページ内で、テーブル、フォーム、ボタンなどのさまざまなコンポーネントを追加して、必要なレイアウトと機能を作成できます。

デフォルトページの名前を変更する

このチュートリアルテキストサマリアプリには 1 ページしか含まれません。新しく作成されたアプリケーションにはデフォルトのページがあるため、追加するのではなく名前を変更します。

デフォルトページの名前を変更するには

1. 上部のバーナビゲーションメニューで、ページを選択します。
2. 左側のパネルで PagePage1 を選択し、右側のパネルで Properties パネルを選択します。
3. 鉛筆アイコンを選択し、 と入力し `TextSummarizationTool`、Enter キーを押します。
4. ナビゲーションラベルに「」と入力します `TextSummarizationTool`。

ページにコンポーネントを追加する

このチュートリアルでは、テキストサマリアプリに次のコンポーネントを含む 1 ページがあります。

- エンドユーザーが要約されるプロンプトの入力に使用するテキスト入力コンポーネント。

- Amazon Bedrock にプロンプトを送信するために使用されるボタンコンポーネント。
- Amazon Bedrock の概要を表示するテキストエリアコンポーネント。

要約されるテキストプロンプトの入力にユーザーが使用するページにテキスト入力コンポーネントを追加します。

テキスト入力コンポーネントを追加するには

1. 右側のコンポーネントパネルで、テキスト入力コンポーネントを見つけ、キャンバスにドラッグします。
2. キャンバス内のテキスト入力を選択して選択します。
3. 右側のプロパティパネルで、次の設定を更新します。
 - a. 鉛筆アイコンを選択して、テキスト入力の名前を `inputPrompt` に変更します。
 - b. Label に「」と入力します `Prompt`。
 - c. プレースホルダーに「」と入力します `Enter text to be summarized`。

次に、ユーザーがプロンプトを Amazon Bedrock に送信することを選択するボタンコンポーネントを追加します。

ボタンコンポーネントを追加するには

1. 右側のコンポーネントパネルで、ボタンコンポーネントを見つけてキャンバスにドラッグします。
2. キャンバス内のボタンを選択して選択します。
3. 右側のプロパティパネルで、次の設定を更新します。
 - a. 鉛筆アイコンを選択して、ボタンの名前を `sendButton` に変更します。
 - b. 「ボタンラベル」に「」と入力します `Send`。

ここで、Amazon Bedrock から返された概要を表示するテキストエリアコンポーネントを追加します。

テキストエリアコンポーネントを追加するには

1. 右側のコンポーネントパネルで、テキストエリアコンポーネントを見つけ、キャンバスにドラッグします。
2. キャンバス内のテキスト領域を選択して選択します。
3. 右側のプロパティパネルで、次の設定を更新します。
 - a. 鉛筆アイコンを選択して、ボタンの名前を `textSummary` に変更します。
 - b. Label に「」と入力します `Summary`。

ページコンポーネントを設定する

アプリケーションにコンポーネントを含むページが含まれているので、次のステップは、適切な動作を実行するようにコンポーネントを設定することです。ボタンなどのコンポーネントが操作されたときにアクションを実行するように設定するには、トリガーを追加する必要があります。このチュートリアルアプリでは、`sendButton` ボタンに 2 つのトリガーを追加して、以下を実行します。

- 最初のトリガーは、分析する `textPrompt` コンポーネント内のテキストを Amazon Bedrock に送信します。
- 2 番目のトリガーでは、Amazon Bedrock から返されたサマリーが `textSummary` コンポーネントに表示されます。

プロンプトを Amazon Bedrock に送信するトリガーを追加するには

1. キャンバス内のボタンを選択して選択します。
2. 右側の Properties パネルの Triggers セクションで + Add を選択します。
3. オートメーションの呼び出し を選択します。
4. 設定するために作成された `InvokeAutomation1` トリガーを選択します。
5. 「アクション名」に「」と入力します `invokeBedrockAutomation`。
6. Invoke Automation で、前に作成した `InvokeBedrock` オートメーションを選択します。
7. パラメータボックスに、前に作成した入力パラメータに、`inputPrompt` テキスト入力コンポーネントのコンテンツを `{{ui.inputPrompt.value}}` と入力します。
8. パネルの上部にある左矢印を選択して、コンポーネントプロパティメニューに戻ります。

これで、ボタンがクリックされたときに Amazon Bedrock にリクエストを送信する自動化を呼び出すトリガーを設定しました。次のステップは、textSummary コンポーネントに結果を表示する 2 番目のトリガーを設定することです。

Amazon Bedrock の結果をテキストエリアコンポーネントに表示するトリガーを追加するには

1. ボタンの右側にある Properties パネルの Triggers セクションで + Add を選択します。
2. コンポーネントの実行アクションを選択します。
3. 設定するために作成された Runcomponentaction1 トリガーを選択します。
4. 「アクション名」に「」と入力します `setTextSummary`。
5. コンポーネントで、textSummary コンポーネントを選択します。
6. Action で、Set value を選択します。
7. 「Set value to」に「」と入力します `{{results.invokeBedrockAutomation}}`。

ステップ 5: アプリケーションをテスト環境に公開する

通常、アプリケーションの構築中は、プレビューして外観を確認し、その機能の初期テストを実行することをお勧めします。ただし、アプリケーションはプレビュー環境で外部サービスとやり取りしないため、代わりにアプリケーションをテスト環境に公開して、Amazon Bedrock からの送信リクエストと受信レスポンスをテストできるようにします。

アプリケーションをテスト環境に公開するには

1. App Builder の右上隅で、発行を選択します。
2. テスト環境のバージョンの説明を追加します。
3. SLA に関するチェックボックスを確認して選択します。
4. [開始] を選択します。公開には最大 15 分かかる場合があります。
5. (オプション) 準備ができたなら、共有を選択し、プロンプトに従って他のユーザーにアクセスを許可できます。App Studio アプリの共有の詳細については、「」を参照してください [公開されたアプリケーションの共有](#)。

アプリケーションをテストしたら、再度発行を選択してアプリケーションを本番環境に昇格させます。本番稼働環境のアプリは、エンドユーザーが共有されるまで利用できないことに注意してください。さまざまなアプリケーション環境の詳細については、「」を参照してください [アプリケーション環境](#)。

(オプション) クリーンアップする

これでチュートリアルは正常に完了し、Amazon Bedrock を使用した App Studio でテキスト要約アプリケーションを構築しました。アプリを引き続き使用することも、このチュートリアルで作成したリソースをクリーンアップすることもできます。次のリストには、クリーンアップするリソースのリストが含まれています。

- App Studio で作成された Amazon Bedrock コネクタ。詳細については、「[コネクタの表示、編集、削除](#)」を参照してください。
- App Studio のテキストサマリアプリ。詳細については、「[Deleting an application](#)」を参照してください。
- IAM コンソールで作成された IAM ロール。詳細については、「AWS Identity and Access Management ユーザーガイド」の「[ロールまたはインスタンスプロファイルの削除](#)」を参照してください。
- Claude 3 Sonnet を使用するようにモデルアクセスをリクエストし、アクセスを元に戻す場合は、「[Amazon Bedrock ユーザーガイド](#)」の「[Amazon Bedrock 基盤モデルへのアクセスの管理](#)」を参照してください。

Amazon Simple Storage Service とコンポーネントおよびオートメーションの操作

App Studio アプリからさまざまな Amazon S3 オペレーションを呼び出すことができます。たとえば、シンプルな管理パネルを作成して、ユーザーと注文を管理し、Amazon S3 からメディアを表示できます。Invoke アクションを使用して任意の Amazon S3 オペレーションを呼び出す AWS ことができますが、Amazon S3 バケットとオブジェクトで一般的なオペレーションを実行するために、アプリのオートメーションに追加できる専用の Amazon S3 アクションが 4 つあります。4 つのアクションとそのオペレーションは次のとおりです。

- Put Object: Amazon S3 PutObject オペレーションを使用して、Amazon S3 バケットにオブジェクトを追加します。
- オブジェクトの取得: Amazon S3 GetObject オペレーションを使用して Amazon S3 バケットからオブジェクトを取得します。
- オブジェクトの一覧表示: Amazon S3 ListObjects オペレーションを使用して、Amazon S3 バケット内のオブジェクトを一覧表示します。
- オブジェクトの削除: Amazon S3 DeleteObject オペレーションを使用して、Amazon S3 バケットからオブジェクトを削除します。

アクションに加えて、アプリケーションのページに追加できる S3 アップロードコンポーネントがあります。ユーザーはこのコンポーネントを使用してアップロードするファイルを選択し、コンポーネントは Amazon S3 PutObject を呼び出して設定済みのバケットとフォルダにファイルをアップロードします。このチュートリアルでは、スタンドアロンの Put Object オートメーションアクションの代わりにこのコンポーネントを使用します。(スタンドアロンアクションは、アップロード前またはアップロード後に実行する追加のロジックまたはアクションを含む、より複雑なシナリオで使用する必要があります)。

前提条件

このガイドでは、次の前提条件を満たしていることを前提としています。

1. Amazon S3 を App Studio と正常に統合するために、Amazon S3 バケット、IAM ロールとポリシー、Amazon S3 コネクタを作成および設定しました。コネクタを作成するには、管理者ロールが必要です。詳細については、「[Amazon Simple Storage Service \(Amazon S3\) に接続する](#)」を参照してください。

空のアプリケーションを作成する

以下のステップを実行して、このガイド全体で使用する空のアプリケーションを作成します。

空のアプリケーションを作成するには

1. ナビゲーションペインで、アプリケーションを選択します。
2. + アプリの作成 を選択します。
3. アプリの作成ダイアログボックスで、アプリケーションに名前を付け、最初から開始を選択し、次へを選択します。
4. 既存のデータに接続ダイアログボックスで、スキップを選択してアプリケーションを作成します。
5. 新しいアプリケーションのキャンバスに移動するアプリの編集を選択します。ここでは、コンポーネント、オートメーション、データを使用して、アプリケーションのルックと機能を設定できます。

ページの作成

アプリケーションに 3 つのページを作成して、情報を収集または表示します。

ページを作成するには

1. 必要に応じて、キャンバスの上部にあるページタブを選択します。
2. 左側のナビゲーションには、アプリで作成されたページが 1 つあります。+ 追加を 2 回選択して、さらに 2 つのページを作成します。ナビゲーションペインには合計 3 ページが表示されます。
3. 次の手順を実行して、Page1 ページの名前を更新します。
 - a. 楕円アイコンを選択し、ページプロパティを選択します。
 - b. 右側のプロパティメニューで、鉛筆アイコンを選択して名前を編集します。
 - c. **FileList** を入力して Enter を押します。
4. 前のステップを繰り返して、2 ページ目と 3 ページ目を次のように更新します。
 - Page2 の名前を **UploadFile** に変更します。
 - Page3 の名前を **FailUpload** に変更します。

これで、アプリには FileList、UploadFile、FailUpload という 3 つのページがあり、左側の Pages パネルに表示されます。

次に、Amazon S3 とやり取りするオートメーションを作成して設定します。

オートメーションの作成と設定

Amazon S3 とやり取りするアプリケーションの自動化を作成します。次の手順を使用して、次のオートメーションを作成します。

- Amazon S3 バケット内のオブジェクトを一覧表示する **getFiles** オートメーション。テーブルコンポーネントを埋めるために使用されます。
- Amazon S3 バケットからオブジェクトを削除する **deleteFile** オートメーション。テーブルコンポーネントに削除ボタンを追加するために使用されます。
- Amazon S3 バケットからオブジェクトを取得して表示する **viewFile** オートメーション。テーブルコンポーネントから選択された 1 つのオブジェクトに関する詳細を表示するために使用されます。

getFiles オートメーションを作成する

指定した Amazon S3 バケット内のファイルを一覧表示するオートメーションを作成します。

1. キャンバスの上部にあるオートメーションタブを選択します。
2. 「+ オートメーションの追加」を選択します。
3. 右側のパネルで、プロパティを選択します。
4. 鉛筆アイコンを選択してオートメーション名を更新します。**getFiles** を入力して Enter を押します。
5. 次の手順を実行して、オブジェクトのリストアクションを追加します。
 - a. 右側のパネルで、アクションを選択します。
 - b. オブジェクトのリストを選択してアクションを追加します。アクションには **という名前を付ける必要がありますListObjects1**。
6. 次の手順を実行して、アクションを設定します。
 - a. キャンバスからアクションを選択して、右側のプロパティメニューを開きます。
 - b. Connector では、前提条件から作成した Amazon S3 コネクタを選択します。
 - c. 設定 に次のテキストを入力し、**bucket_name** を前提条件で作成したバケットに置き換えます。

```
{
  "Bucket": "bucket_name",
  "Prefix": ""
}
```

Note

Prefix フィールドを使用して、指定された文字列で始まるオブジェクトにレスポンスを制限できます。

7. この自動化の出力は、Amazon S3 バケットのオブジェクトをテーブルコンポーネントに入力するために使用されます。ただし、デフォルトでは、オートメーションは出力を作成しません。次の手順を実行して、オートメーション出力を作成するようにオートメーションを設定します。
 - a. 左側のナビゲーションで、getFiles オートメーションを選択します。
 - b. 右側の Properties メニューの Automation 出力で + Add output を選択します。
 - c. Output には、と入力します **{{results.ListObjects1.Contents}}**。この式は アクションの内容を返し、テーブルコンポーネントの入力に使用できるようになりました。

deleteFile オートメーションを作成する

指定された Amazon S3 バケットからオブジェクトを削除するオートメーションを作成します。

1. 左側の Automations パネルで、+ 追加 を選択します。
2. 「 」 + 「オートメーションの追加」を選択します。
3. 右側のパネルで、プロパティを選択します。
4. 鉛筆アイコンを選択してオートメーション名を更新します。deleteFile を入力して Enter を押します。
5. 以下のステップを実行して、オートメーションにデータを渡すために使用されるオートメーションパラメータを追加します。
 - a. 右側のプロパティメニューで、オートメーションパラメータで + 追加を選択します。
 - b. 鉛筆アイコンを選択して、オートメーションパラメータを編集します。パラメータ名を `params.fileName` に更新し、Enter キーを押します。
6. 次の手順を実行して、オブジェクトの削除アクションを追加します。
 - a. 右側のパネルで、アクションを選択します。
 - b. オブジェクトの削除を選択してアクションを追加します。アクションには `DeleteObject1` という名前を付ける必要があります。
7. 次の手順を実行して、アクションを設定します。
 - a. キャンバスからアクションを選択して、右側のプロパティメニューを開きます。
 - b. Connector では、前提条件から作成した Amazon S3 コネクタを選択します。
 - c. 設定 に次のテキストを入力し、`bucket_name` を前提条件で作成したバケットに置き換えます。

```
{
  "Bucket": "bucket_name",
  "Key": params.fileName
}
```

viewFile オートメーションを作成する

指定された Amazon S3 バケットから単一のオブジェクトを取得するオートメーションを作成します。後で、この自動化をファイルビューワーコンポーネントで設定して、オブジェクトを表示します。

1. 左側の Automations パネルで、+ 追加 を選択します。
2. 「」 + 「オートメーションの追加」を選択します。
3. 右側のパネルで、プロパティを選択します。
4. 鉛筆アイコンを選択してオートメーション名を更新します。**viewFile** を入力して Enter を押します。
5. 以下のステップを実行して、オートメーションにデータを渡すために使用されるオートメーションパラメータを追加します。
 - a. 右側の Properties メニューの Automation パラメータで + Add を選択します。
 - b. 鉛筆アイコンを選択して、オートメーションパラメータを編集します。パラメータ名を **fileName** に更新し、Enter キーを押します。
6. 次の手順を実行して、オブジェクトの取得アクションを追加します。
 - a. 右側のパネルで、アクションを選択します。
 - b. オブジェクトの取得 を選択してアクションを追加します。アクションには という名前を付ける必要があります `GetObject1`。
7. 次の手順を実行して、アクションを設定します。
 - a. キャンバスからアクションを選択して、右側のプロパティメニューを開きます。
 - b. Connector では、前提条件から作成した Amazon S3 コネクタを選択します。
 - c. 設定 に次のテキストを入力し、**bucket_name** を前提条件で作成したバケットに置き換えます。

```
{
  "Bucket": "bucket_name",
  "Key": params.fileName
}
```
8. デフォルトでは、オートメーションは出力を作成しません。次の手順を実行して、オートメーション出力を作成するようにオートメーションを設定します。
 - a. 左側のナビゲーションで、viewFile オートメーションを選択します。
 - b. 右側の Properties メニューの Automation 出力で + Add output を選択します。
 - c. Output には、 と入力します **{{results.GetObject1.Body.transformToWebStream()}}**。この式は、アクションの内容を返します。

Note

のレスポンスはS3 `GetObject`、次の方法で読み取ることができます。

- `transformToWebStream`: データを取得するために消費する必要があるストリームを返します。自動化出力として使用すると、自動化によって処理され、出力はイメージまたは PDF ビューワーコンポーネントのデータソースとして使用できます。また、などの別のオペレーションへの入力としても使用できますS3 `PutObject`。
- `transformToString`: オートメーションの raw データを返します。ファイルに JSON データなどのテキストコンテンツが含まれている場合は、JavaScript アクションで使用する必要があります。次の例のように、待機している必要があります。 `await results.GetObject1.Body.transformToString();`
- `transformToByteArray`: 8 ビットの符号なし整数の配列を返します。このレスポンスは、バイナリデータのストレージと操作を可能にするバイト配列の目的を果たします。次の例のように、待機している必要があります。 `await results.GetObject1.Body.transformToByteArray();`

次に、前に作成したページにコンポーネントを追加し、ユーザーがアプリを使用してファイルを表示および削除できるようにオートメーションでコンポーネントを設定します。

ページコンポーネントの追加と設定

アプリケーションのビジネスロジックと機能を定義するオートメーションを作成したら、コンポーネントを作成し、両方を接続します。

FileList ページにコンポーネントを追加する

前に作成した FileList ページは、設定された Amazon S3 バケット内のファイルのリストと、リストから選択されたファイルの詳細を表示するために使用されます。これを行うには、以下を実行します。

1. ファイルのリストを表示するテーブルコンポーネントを作成します。テーブルの行を、以前に作成した `getFiles` オートメーションの出力で埋めるように設定します。

- PDF ビューワーコンポーネントを作成して、1 つの PDF を表示します。バケットからファイルを取得するために以前に作成した viewFile オートメーションを使用して、テーブルから選択されたファイルを表示するようにコンポーネントを設定します。

FileList ページにコンポーネントを追加するには

- キャンバスの上部にあるページタブを選択します。
- 左側のページパネルで、FileList ページを選択します。
- 右側のコンポーネントページで、テーブルコンポーネントを見つけ、キャンバスの中央までドラッグします。
- ページに追加したテーブルコンポーネントを選択します。
- 右側のプロパティメニューで、ソースドロップダウンを選択し、オートメーションを選択します。
- 自動化ドロップダウンを選択し、getFiles 自動化を選択します。このテーブルは、getFiles オートメーションの出力をコンテンツとして使用します。
- ファイルの名前を入力する列を追加します。
 - 右側のプロパティメニューで、列の横にある + 追加を選択します。
 - 追加した Column1 列の右側にある矢印アイコンを選択します。
 - 列ラベルの場合は、列の名前を に変更します **Filename**。
 - [Value] (値) に 「**{{currentRow.Key}}**」 と入力します。
 - パネルの上部にある矢印アイコンを選択して、メインのプロパティパネルに戻ります。
- テーブルアクションを追加して、行内のファイルを削除します。
 - 右側のプロパティメニューで、アクションの横にある + 追加 を選択します。
 - アクションで、ボタンの名前を に変更します **Delete**。
 - 先ほど名前を変更した削除アクションの右側にある矢印アイコンを選択します。
 - クリック時に + アクションを追加 を選択し、オートメーションを呼び出す を選択します。
 - 追加したアクションを選択して設定します。
 - [アクション名] に 「**DeleteRecord**」 と入力します。
 - オートメーションを呼び出すで、 を選択します **deleteFile**。
 - パラメータテキストボックスに、 と入力します **{{currentRow.Key}}**。
 - [Value] (値) に 「**{{currentRow.Key}}**」 と入力します。

9. 右側のパネルで、コンポーネントを選択してコンポーネントメニューを表示します。ファイルを表示するには、次の2つの選択肢があります。

- .png、.jpegまたは.jpg拡張子を持つファイルを表示するイメージビューワー。
- PDF ファイルを表示する PDF ビューワーコンポーネント。

このチュートリアルでは、PDF ビューワーコンポーネントを追加および設定します。

10. PDF ビューワーコンポーネントを追加します。

- 右側のコンポーネントページで、PDF ビューワーコンポーネントを見つけ、テーブルコンポーネントの下のキャンバスにドラッグします。
- 追加した PDF ビューワーコンポーネントを選択します。
- 右側のプロパティメニューで、ソースドロップダウンを選択し、オートメーションを選択します。
- 自動化ドロップダウンを選択し、viewFile 自動化を選択します。テーブルは、viewFile オートメーションの出力をコンテンツとして使用します。
- パラメータテキストボックスに、と入力します `{{ui.table1.selectedRow["Filename"]}}`。
- 右側のパネルには、ファイル名フィールドもあります。このフィールドの値は、PDF ビューワーコンポーネントのヘッダーとして使用されます。前のステップと同じテキストを入力します: `{{ui.table1.selectedRow["Filename"]}}`。

UploadFile ページにコンポーネントを追加する

UploadFile ページには、設定された Amazon S3 バケットへのファイルの選択とアップロードに使用できるファイルセレクタが含まれます。S3 アップロードコンポーネントをページに追加します。ユーザーはこれを使用してファイルを選択してアップロードできます。

- 左側のページパネルで、UploadFile ページを選択します。
- 右側のコンポーネントページで、S3 アップロードコンポーネントを見つけ、キャンバスの中央までドラッグします。
- ページに追加した S3 アップロードコンポーネントを選択します。
- 右側のプロパティメニューで、コンポーネントを設定します。
 - Connector ドロップダウンで、前提条件で作成された Amazon S3 コネクタを選択します。

- b. バケットには、Amazon S3 バケットの名前を入力します。
- c. [ファイル名]に `{{ui.s3Upload1.files[0]?.nameWithExtension}}` と入力します。
- d. 最大ファイルサイズの場合は、テキストボックス5に を入力し、ドロップダウンで **MB** が選択されていることを確認します。
- e. トリガーセクションで、次の手順を実行して、アップロードが成功または失敗した後に実行されるアクションを追加します。

アップロードが成功した後に実行されるアクションを追加するには:

1. 成功したら、**+ アクションを追加** を選択し、**ナビゲート** を選択します。
2. 追加したアクションを選択して設定します。
3. ナビゲーションタイプで、**ページ**を選択します。
4. 「ナビゲート」で、「」を選択します**FileList**。
5. パネルの上部にある矢印アイコンを選択すると、メインのプロパティパネルに戻ります。

アップロードに失敗した後に実行されるアクションを追加するには:

1. 「失敗時」で、「アクションの追加」を選択し、「ナビゲート」を選択します。
2. 追加したアクションを選択して設定します。
3. ナビゲーションタイプで、**ページ**を選択します。
4. 「ナビゲート」で、「」を選択します**FailUpload**。
5. パネルの上部にある矢印アイコンを選択すると、メインのプロパティパネルに戻ります。

FailUpload ページにコンポーネントを追加する

FailUpload ページは、アップロードが失敗したことをユーザーに知らせるテキストボックスを含むシンプルなページです。

1. 左側のページパネルで、FailUpload ページを選択します。
2. 右側のコンポーネントページで、テキストコンポーネントを見つけ、キャンバスの中央までドラッグします。
3. ページに追加したテキストコンポーネントを選択します。

4. 右側の「プロパティ」メニューの「値」に「」と入力します**Failed to upload, try again**。

アプリのセキュリティ設定を更新する

App Studio のすべてのアプリケーションには、外部メディアやリソースを制限するために使用できるコンテンツセキュリティ設定、またはオブジェクトをアップロードできる Amazon S3 のドメインがあります。デフォルト設定では、すべてのドメインをブロックします。アプリケーションから Amazon S3 にオブジェクトをアップロードするには、オブジェクトをアップロードするドメインを許可するように設定を更新する必要があります。

Amazon S3 へのオブジェクトのアップロードをドメインに許可するには

1. アプリ設定タブを選択します。
2. コンテンツセキュリティ設定タブを選択します。
3. Connect source で、すべての接続を許可する を選択します。
4. [保存] を選択します。

次のステップ: テスト用にアプリケーションをプレビューして公開する

これで、アプリケーションをテストする準備が整いました。アプリケーションのプレビューと公開の詳細については、「」を参照してください[アプリケーションのプレビュー、公開、共有](#)。

App Studio アプリでの Lambda 関数の呼び出し

このチュートリアルでは、App Studio を Lambda に接続し、アプリから Lambda 関数を呼び出す方法を示します。

前提条件

このガイドでは、次の前提条件を満たしていることを前提としています。

1. App Studio アプリを作成しました。アプリがない場合は、チュートリアルで使用する空のアプリを作成できます。詳細については、「[Creating an application](#)」を参照してください。

Note

このチュートリアルに従って設定する方法を学ぶために Lambda 関数は必要ありませんが、アプリを正しく設定するために Lambda 関数を用意すると役立つ場合があります。このチュートリアルには、Lambda 関数の作成に関する情報は含まれていません。詳細については、「[AWS Lambda デベロッパーガイド](#)」を参照してください。

Lambda コネクタを作成する

App Studio アプリで Lambda 関数を使用するには、コネクタを使用して App Studio を Lambda に接続し、関数へのアクセスを提供する必要があります。App Studio でコネクタを作成するには、管理者である必要があります。Lambda コネクタを作成する手順など、Lambda コネクタの作成の詳細については、「」を参照してください [に接続する AWS Lambda](#)。

オートメーションの作成と設定

自動化はアプリケーションのロジックを定義するために使用され、アクションで構成されます。アプリで Lambda 関数を呼び出すには、まず Lambda 呼び出しアクションをオートメーションに追加して設定します。次のステップを使用してオートメーションを作成し、Lambda 呼び出しアクションを追加します。

1. アプリの編集中に、オートメーションタブを選択します。
2. 「」と「オートメーションの追加」を選択します。
3. 右側のアクションメニューで、Lambda の呼び出しを選択して、オートメーションにステップを追加します。
4. キャンバスで新しい Lambda ステップを選択して、そのプロパティを表示および設定します。
5. 右側のプロパティメニューで、次の手順を実行してステップを設定します。
 - a. Connector で、App Studio を Lambda 関数に接続するために作成されたコネクタを選択します。
 - b. 関数名に、Lambda 関数の名前を入力します。
 - c. 関数イベントで、Lambda 関数に渡されるイベントを入力します。一般的なユースケースの例を次のリストに示します。
 - ファイル名やその他の文字列などのオートメーションパラメータの値を渡す: `varName: params.paramName`

- 前のアクションの結果を渡す: `varName: results.actionName1.data[0].fieldName`
 - ループアクション内に Lambda アクションを呼び出すを追加すると、パラメータのような反復された各項目からフィールドを送信できます。 `varName: currentItem.fieldName`
- d. モック出力フィールドは、コネクタがアクティブでないプレビュー中にアプリをテストするモック出力を提供するために使用できます。

オートメーションを実行するように UI 要素を設定する

これで、Lambda 関数を呼び出すアクションで設定されたオートメーションができたので、オートメーションを実行するように UI 要素を設定できます。このチュートリアルでは、クリックしたときにオートメーションを実行するボタンを作成します。

Tip

オートメーションの呼び出しアクションを使用して、他のオートメーションからオートメーションを実行することもできます。

ボタンからオートメーションを実行するには

1. アプリの編集集中に、ページタブを選択します。
2. 右側のメニューで、ボタンコンポーネントを選択してページにボタンを追加します。
3. 新しいボタンを選択して設定します。
4. 右側の Properties メニューの Triggers で + Add を選択し、Invoke automation を選択します。
5. 新しいオートメーション呼び出しトリガーを選択して設定します。
6. Invoke Automation で、Lambda 関数を呼び出すオートメーションを選択し、オートメーションに送信するパラメータを設定します。

これで、アプリでこのボタンを選択すると、設定されたオートメーションが実行されます。

次のステップ: テスト用にアプリケーションをプレビューして公開する

これで、アプリケーションをテストする準備が整いました。開発環境でアプリケーションをプレビューする場合、コネクタはアクティブではないため、コネクタを使用して接続するため、プレ

ビュー中にオートメーションをテストすることはできません AWS Lambda。コネクタに依存するアプリケーションの機能をテストするには、アプリケーションをテスト環境に公開する必要があります。アプリケーションのプレビューと公開の詳細については、「」を参照してください[アプリケーションのプレビュー、公開、共有](#)。

生成 AI を使用した App Studio アプリの構築

AWS App Studio は、開発を加速し、一般的なタスクを合理化するための統合された生成 AI 機能を提供します。生成 AI を活用して、アプリケーション、データモデル、サンプルデータを生成および編集したり、アプリケーションの構築中にコンテキストに応じたヘルプを取得したりできます。

アプリの生成

高速スタートでは、AI を活用した自然言語プロンプトを使用してアプリケーション全体を生成できます。この機能を使用すると、必要なアプリ機能を記述でき、AI はデータモデル、ユーザーインターフェイス、ワークフロー、コネクタを自動的に構築します。AI でアプリを生成する方法の詳細については、「」を参照してください[Creating an application](#)。

アプリの構築または編集

アプリケーションの編集に、チャットを使用して変更を記述すると、アプリケーションは自動的に更新されます。既存のサンプルプロンプトから選択することも、独自のプロンプトを入力することもできます。チャットを使用して、サポートされているコンポーネントを追加、編集、削除したり、オートメーションとアクションを作成、設定したりできます。AI を使用してアプリケーションを編集または構築するには、次の手順に従います。

AI でアプリを編集するには

1. 必要に応じて、アプリケーションを編集してアプリケーションスタジオに移動します。
2. (オプション) AI で編集するページまたはコンポーネントを選択します。
3. チャットを開くには、左下の AI で構築を選択します。
4. 変更を入力するか、サンプルプロンプトから選択します。
5. 変更内容を確認します。変更を行う場合は、確認を選択します。それ以外の場合は、別のプロンプトを入力します。
6. 変更の概要を確認します。

データモデルの生成

指定されたエンティティ名に基づいて、フィールド、データ型、データアクションを含むエンティティを自動的に生成できます。GenAi を使用したエンティティの作成など、エンティティの作成の詳細については、「」を参照してください[App Studio アプリでのエンティティの作成](#)。

以下の方法で既存のエンティティを更新することもできます。

- エンティティにフィールドを追加します。詳細については、「[エンティティフィールドの追加、編集、または削除](#)」を参照してください。
- エンティティにデータアクションを追加します。詳細については、「[データアクションの作成](#)」を参照してください。

サンプルデータの生成

エンティティのフィールドに基づいて、エンティティのサンプルデータを生成できます。これは、外部データソースを接続する前にアプリケーションをテストしたり、外部データソースと通信しない開発環境でアプリケーションをテストしたりする場合に便利です。詳細については、「[サンプルデータの追加または削除](#)」を参照してください。

アプリケーションをテストまたは本稼働環境に公開すると、ライブデータソースとコネクタがそれらの環境で使用されます。

AWS サービスのアクションの設定

Amazon Simple Email Service などの AWS サービスと統合する場合、AI を使用して、選択したサービスに基づいて事前入力されたフィールドを含む設定例を生成できます。これを試すには、AWS 自動化アクションの呼び出しのプロパティメニューで、両側矢印を選択して設定フィールドを展開します。次に、「サンプル設定の生成」を選択します。

レスポンスのモック

AWS サービスアクションのモックレスポンスを生成できます。これは、外部データソースと通信しない開発環境でアプリケーションをテストするのに役立ちます。

構築中に AI にヘルプを求める

アプリケーションスタジオ内には、サポートされているリソースまたはプロパティに関するヘルプボタンとして Ask AI があります。これを使用して、現在のビューまたは選択したコンポーネントに関

連するコンテキストの提案、ドキュメント、ガイダンスを取得します。App Studio、アプリ構築のベストプラクティス、または特定のアプリケーションのユースケースに関する一般的な質問をして、カスタマイズされた情報と推奨事項を受け取ります。

アプリケーションの作成、編集、削除

目次

- [Creating an application](#)
- [アプリケーションのインポート](#)
 - [App Studio が提供するインポート可能なアプリ](#)
- [アプリケーションの複製](#)
- [アプリケーションの編集または構築](#)
- [以前に公開されたアプリバージョンを編集する](#)
- [アプリケーションの名前変更](#)
- [Deleting an application](#)

Creating an application

App Studio でアプリケーションを作成するには、次の手順に従います。

アプリケーションを作成するには

1. ナビゲーションペインで、ビルドセクションのマイアプリケーションを選択して、アプリケーションのリストに移動します。
2. + アプリの作成を選択します。
3. アプリの作成ダイアログボックスで、アプリケーションに名前を付け、次のいずれかのアプリ作成方法を選択します。
 - AI を使用してアプリを生成する: 自然言語でアプリを記述し、AI がアプリとそのリソースを自動的に生成するようにするには、このオプションを選択します。
 - 最初から開始: 空のアプリから構築を開始するには、このオプションを選択します。
4. [次へ] を選択します。
5. AI を使用してアプリを生成するを選択した場合:
 - a. 既存のデータに接続ダイアログボックスで、App Studio にデータソースへのアクセスを提供するコネクタを選択し、Tablese を選択し、次へ を選択して、既存のデータソースをアプ

りに追加します。ここにデータソースを追加すると、AI が最適化されたアプリケーションを生成するのに役立ちます。このステップをスキップし、後で「スキップ」を選択してデータソースを追加できます。

- b. 短い遅延 (数分) の後、AI を使用してアプリを生成するページに移動し、作成するアプリを記述できます。
 - c. チャットでアプリの説明を開始するか、提供されたサンプルプロンプトを選択してカスタマイズできます。
 - d. プロンプトが分析されたら、アプリの要件と概要を確認します。チャットを使用して変更をリクエストするか、「やり直す」を選択して空のプロンプトから開始します。
 - e. 準備ができたら、アプリの生成を選択します。
 - f. 生成したら、アプリのプレビューを選択して、別のタブでアプリをプレビューします。編集を開始する準備ができたら、アプリの編集を選択できます。アプリケーションのページ、オートメーション、データを参照して、理解を深めます。下部のデバッグパネルでエラーや警告を確認します。AI を使用してアプリを生成する方法については、「」を参照してください [チュートリアル: AI を使用してアプリを生成する](#)。App Studio でのビルドの仕組みに関する一般的な情報については、「」を参照してください [AWS App Studio の仕組み](#)。
6. 最初から開始を選択した場合:
- a. 既存のデータへの接続ダイアログボックスで、データソースへのアクセスを App Studio に許可するコネクタを選択し、Tablese を選択し、Next を選択して、既存のデータソースをアプリに追加します。スキップを選択して、このステップをスキップし、後でデータソースを追加できます。
 - b. アプリを作成したら、アプリの編集を選択してアプリの編集を開始します。空のアプリケーションから構築する方法については、「」を参照してください [チュートリアル: 空のアプリから構築を開始する](#)。App Studio でのビルドの仕組みに関する一般的な情報については、「」を参照してください [AWS App Studio の仕組み](#)。

アプリケーションのインポート

エクスポートしたアプリケーションのコピーを App Studio インスタンスにインポートできます。他の App Studio インスタンスからエクスポートされたアプリ、または App Studio が提供するカタログからアプリをインポートできます。App Studio アプリカタログからアプリをインポートすると、同様の機能を持つアプリの使用を開始したり、インポートされたアプリを調べて App Studio でのアプリ構築について学習したりできます。

インスタンスにアプリをインポートすると、元のアプリのコピーがインスタンスに作成されます。新しいアプリが作成されると、アプリの開発環境に移動し、プレビューしてアプリの機能を参照できます。

Warning

アプリケーションをインポートすると、アプリケーションからすべてのロジックをインポートするため、望ましくない動作や予期しない動作が発生する可能性があります。たとえば、アプリケーションに接続するデータベースからデータを削除する破壊的なクエリが存在する可能性があります。本番稼働用データを接続する前に、アプリケーションとその設定を徹底的に確認し、非本番稼働用アセットでテストすることをお勧めします。

アプリケーションをインポートするには

1. ナビゲーションペインで、ビルドセクションのマイアプリケーションを選択して、アプリケーションのリストに移動します。
2. + アプリの作成の横にあるドロップダウン矢印を選択します。
3. アプリのインポート を選択します。
4. アプリケーションのインポートダイアログボックスのインポートコードに、インポートするアプリケーションのインポートコードを入力します。アプリケーションの説明やインポートコードなど、インポートできる App Studio が提供するアプリケーションのリストについては、「」を参照してください[App Studio が提供するインポート可能なアプリ](#)。
5. インポートを選択してアプリをインポートし、インポートしたアプリの開発環境に移動して表示または編集します。App Studio でのアプリの構築については、「」を参照してください。 [AWS App Studio の仕組み](#)

App Studio が提供するインポート可能なアプリ

App Studio には、アプリケーション構築について学ぶのに役立つ、インスタンスにインポートできるアプリケーションが用意されています。App Studio で特定のアプリケーション機能がどのように実装されているかを確認するには、アプリケーションをプレビューし、開発環境で設定を参照します。

次の表に、アプリケーションのリスト、インポートコード、およびアプリケーションの簡単な説明を示します。各アプリには、インポート後に表示できるアプリに関する情報を含むREADMEページが含まれています。

アプリ名	説明	コードのインポート
スワップリクエストアンケート	従業員がブランド会社の商品を注文できるように設計された内部スワッグリクエストアプリケーション。従業員はアイテムを選択し、サイズを指定して、シンプルな形式でリクエストを送信できます。このアプリケーションは、組み込みストレージを介してすべてのデータを処理するため、外部接続が不要になります。	スワップリクエストSurvey/ec4f5faf-e2f8-42ee-ab8d-6723d8ca21b2
スプリントの追跡	チームがソフトウェア開発作業の整理と追跡に使用できるスプリント管理アプリケーション。ユーザーは、専用のスプリント、トラック、タスクビューを使用して、スプリントの作成、タスクの追加、作業の割り当て、進行状況のモニタリングを行うことができます。このアプリケーションは、組み込みストレージを介してすべてのデータを処理するため、外部接続が不要になります。	スプリントTracking/8f31e160-771f-48d7-87b0-374e285e2fbc
Amazon Review 感情トラッカー	このアプリケーションは、製品レビューから感情スコアを生成して、企業が顧客満足を理解するのに役立つ顧客フィードバック分析ツールです。このアプリケーションには、クイックテスト用のサンプルデータ生成ユーティリ	Amazon Review Sentiment Tracker/60f0dae4-f8e2-4c20-9583-fa456f5ebfab

アプリ名	説明	コードのインポート
	<p>ティが含まれており、AI を活用したインサイトを得るには Amazon Bedrock コネクタが必要ですが、他のすべてのデータは組み込みストレージシステム内に維持されます。</p>	
請求書と領収書の処理	<p>この受信処理アプリケーションは、手動データ入力を自動化し、ドキュメント承認ワークフローを合理化することで、時間を節約し、エラーを削減します。このソリューションには、Amazon Textract、Amazon S3、Amazon SES コネクタが必要です。Amazon Textract を使用して、Amazon S3 に保存されている受信からデータを分析および抽出し、抽出された情報を処理して Amazon SES を使用して承認者に Eメールで送信します。</p>	請求書と領収書Processing/ 98bde3ae-e454-4b18- a1e6-6f23e8b2a4f1

アプリ名	説明	コードのインポート
検査とインベントリ監査	<p>倉庫検査と機器追跡を管理するアプリケーション。ユーザーは、部屋の場所ごとに合格/不合格の機器評価を行い、インベントリレベルをモニタリングし、検査履歴を表示できます。アプリケーションは、施設検査と機器ステータスの両方を追跡するための一元化されたシステムを提供します。このアプリケーションは、組み込みストレージを介してすべてのデータを処理するため、外部接続が不要になります。</p>	検査とインベントリのAudit/ cf570a06-1c5e-4dd7-9 ea8-5c04723d687f
製品導入トラッカー	<p>顧客のフィードバック、機能リクエスト、顧客との会議メモを一元化する、製品開発を管理するための包括的なアプリケーション。チームは、顧客とのやり取りを追跡し、要件を整理し、四半期ごとのロードマップ計画のための AI を活用したレポートを生成できます。アプリケーションにはサンプルデータユーティリティが含まれており、AI インサイトには Amazon Bedrock、データ管理には Amazon Aurora PostgreSQL を活用します。</p>	製品導入Tracker/9b3a4437- bb50-467f-ae9e-d108776b7 ca1

アプリ名	説明	コードのインポート
クイックスイートの埋め込み	基盤となるデータの操作中にユーザーが分析を表示できるようにするデモアプリケーション。このアプリには、App Studio に Amazon Quick Suite ダッシュボードを埋め込む 2 つの方法があります。登録済みユーザーと匿名ユーザー向けの API ベースのアプローチ (Quick Suite コネクタが必要) と、パブリックダッシュボード用の iFrame 統合です。	Quicksight Embedding /0cdc15fc-ca8b-41b7-869e-ed13c9072bc8
キッチンシンク	高度な App Studio 開発のヒントとベストプラクティスを示すリファレンスアプリケーション。ビルダーが独自のアプリケーションで調査および実装できる状態管理、CSV データ処理、ブラウザ API 統合、UI パターンの実例が含まれています。いずれの例も外部接続を必要としません。	App Studio Kitchen Sink/1cfe6b2f-544c-4611-b82c-80eadc76a0c8

アプリケーションの複製

アプリケーション所有者と共同所有者は、アプリケーションを複製して、アプリケーションの正確なコピーを作成できます。アプリの複製は、テスト目的で現在の状態を保持する場合や、複製されたアプリをスターターとして使用して新しいアプリを作成する場合に便利です。

アプリケーションを複製するには

1. ナビゲーションペインで、ビルドセクションのマイアプリケーションを選択します。アクセス可能なアプリケーションのリストを表示するページが表示されます。

- 複製するアプリケーションのアクション列でドロップダウンを選択します。
- [複製] を選択します。重複オプションが利用できない場合、アプリケーションの所有者または共同所有者ではない可能性があります。
- 必要に応じて、複製されたアプリケーションの名前を指定します。デフォルト名は `Current_App_Name COPY` です。
- [複製] を選択します。

アプリケーションの編集または構築

App Studio でアプリケーションを編集するには、次の手順に従います。

アプリケーションを編集 (ビルド) するには

- ナビゲーションペインで、ビルドセクションのマイアプリケーションを選択します。アクセス可能なアプリケーションのリストを表示するページが表示されます。
- 編集するアプリケーションのアクション列で、編集を選択します。これにより、開発環境に移動し、コンポーネント、オートメーション、データを使用してアプリケーションのルックと機能を設定できます。アプリケーションの構築の詳細については、「」を参照してください [AWS App Studio の開始方法](#)。

以前に公開されたアプリバージョンを編集する

次の手順を使用して、以前に公開されたバージョンの App Studio アプリケーションを編集します。以前に公開されたバージョンを編集することを選択したら、開発環境でアプリを編集するか、テストしてから本稼働環境に公開できます。

Warning

以前に公開されたバージョンは、開発環境で進行中のアプリケーションのバージョンを置き換えます。アプリに対する未公開の変更はすべて失われます。

以前に公開されたバージョンを編集すると、ユーザーのアプリケーションを壊す不要な変更や変更を誤って公開し、以前のアプリケーションバージョンからさらにビルドまたは編集する場合に便利です。

Note

公開されたアプリの問題を検出し、以前に動作したバージョンをすぐに公開する必要がある場合、または以前のバージョンを公開して開発環境でアプリへの最新の更新を保持する場合は、代わりにアプリをロールバックする必要があります。詳細については、「[以前に公開されたバージョンにロールバックする](#)」を参照してください。

以前に公開されたアプリバージョンを編集するには

1. 必要に応じて、アプリケーションのアプリケーションスタジオに移動します。
2. Publish ボタンの横にあるドロップダウン矢印を選択し、Publish Center を選択します。
3. バージョン履歴を選択すると、以前に公開されたアプリケーションのバージョンのリストが表示されます。
4. 編集するバージョンを見つけ、編集を選択します。
5. 情報を確認し、元に戻すを選択します。
6. 編集することを選択したバージョンは、開発環境の現在のバージョンになりました。Publish を選択すると、変更を加えることも、テスト環境にそのまま公開することもできます。テストに公開されると、必要に応じて本番環境に再公開できます。

アプリケーションの名前変更

App Studio でアプリケーションの名前を変更するには、次の手順に従います。アプリケーションの名前は、アプリケーションのリストから変更することも、アプリケーションの構築中に開発環境から変更することもできます。

アプリケーションの名前を変更するには

1. ナビゲーションペインで、ビルドセクションのマイアプリケーションを選択します。アクセス可能なアプリケーションのリストを表示するページが表示されます。
2. このリストから、または編集中に開発環境からアプリケーションの名前を変更できます。
 - このリストから名前を変更するには:
 - a. 名前を変更するアプリケーションのアクション列のドロップダウンを選択し、名前の変更を選択します。
 - b. アプリケーションに新しい名前を付け、名前の変更を選択します。

- 開発環境から名前を変更するには:
 - a. 編集するアプリケーションのアクション列で、編集を選択します。
 - b. 開発環境で、アプリケーション名を選択して更新し、Enter キーを押すか、テキストフィールドから移動して変更を保存します。

Deleting an application

App Studio でアプリケーションを削除するには、次の手順に従います。

アプリケーションを削除するには

1. ナビゲーションペインで、ビルドセクションのマイアプリケーションを選択します。アクセス可能なアプリケーションのリストを表示するページが表示されます。
2. 削除するアプリケーションのアクション列でドロップダウンを選択します。
3. [削除] を選択します。
4. アプリケーションの削除ダイアログボックスで、アプリケーションの削除に関する情報を注意深く確認します。アプリケーションを削除する場合は、削除を選択します。

アプリケーションのプレビュー、公開、共有

トピック

- [アプリケーションのプレビュー](#)
- [アプリケーションの公開](#)
- [公開されたアプリケーションの共有](#)
- [以前に公開されたバージョンにロールバックする](#)
- [アプリケーションのエクスポート](#)

アプリケーションのプレビュー

App Studio でアプリケーションをプレビューして、ユーザーにどのように表示されるかを確認し、それを使用してデバッグパネルでログをチェックすることでその機能をテストすることもできます。

アプリケーションプレビュー環境は、ライブデータの表示や、データソースなどのコネクタを使用した外部リソースへの接続をサポートしていません。プレビュー環境で機能をテストするには、オー

トメーションのモック出力とエンティティのサンプルデータを使用できます。リアルタイムデータでアプリケーションを表示するには、アプリケーションを公開する必要があります。詳細については、「[アプリケーションの公開](#)」を参照してください。

プレビュー環境または開発環境は、他の環境で公開されているアプリケーションを更新しません。アプリケーションが公開されていない場合、ユーザーは公開および共有されるまでアクセスできません。アプリケーションがすでに公開および共有されている場合、ユーザーはプレビュー環境で使用されているバージョンではなく、公開されたバージョンに引き続きアクセスできます。

アプリケーションをプレビューするには

1. 必要に応じて、プレビューするアプリケーションのアプリケーションスタジオに移動します。
 - a. ナビゲーションペインで、ビルドセクションのマイアプリケーションを選択します。
 - b. アプリケーションの編集を選択します。
2. プレビューを選択して、アプリケーションのプレビュー環境を開きます。
3. (オプション) 画面の下部にあるヘッダーを選択して、デバッグパネルを展開します。メッセージのタイプ別にパネルをフィルタリングするには、「ログのフィルタリング」セクションでメッセージのタイプを選択します。コンソールのクリアを選択すると、パネルのログをクリアできます。
4. プレビュー環境では、ページ内を移動し、コンポーネントを使用して、そのボタンを選択してデータを転送するオートメーションを開始することで、アプリケーションをテストできます。プレビュー環境はライブデータや外部ソースへの接続をサポートしていないため、転送されるデータの例をデバッグパネルで表示できます。

アプリケーションの公開

アプリケーションの作成と設定が完了したら、次のステップとして公開してデータ転送をテストするか、エンドユーザーと共有します。App Studio でのアプリケーションの公開を理解するには、利用可能な環境を理解することが重要です。App Studio には、次のリストで説明されている 3 つの個別の環境が用意されています。

1. **開発:** アプリケーションをビルドしてプレビューする場所。アプリケーションの最新バージョンは自動的にホストされるため、開発環境に発行する必要はありません。この環境では、ライブデータ、サードパーティーのサービスやリソースは利用できません。
2. **テスト:** アプリケーションの包括的なテストを実行できる場所。テスト環境では、他のサービスに接続し、データを送受信できます。

3. 本番環境: エンドユーザーが使用するためのライブ運用環境。

すべてのアプリケーション構築は、開発環境で行われます。次に、テスト環境に公開して他のサービス間のデータ転送をテストし、エンドユーザーにアクセス URL を指定してユーザー承認テスト (UAT) を行います。その後、アプリケーションを本番環境に公開して、ユーザーと共有する前に最終テストを実行します。アプリケーション環境の詳細については、「」を参照してください[アプリケーション環境](#)。

アプリケーションを公開すると、共有されるまでユーザーは使用できなくなります。これにより、ユーザーがアプリケーションにアクセスする前に、テスト環境と本番稼働環境でアプリケーションを使用およびテストできます。以前に公開および共有されたアプリケーションを Production に公開すると、ユーザーが利用できるバージョンが更新されます。

アプリケーションの公開

App Studio アプリケーションをテスト環境または本番稼働環境に公開するには、次の手順に従います。

アプリケーションをテスト環境または本番稼働環境に公開するには

1. ナビゲーションペインで、ビルドセクションのマイアプリケーションを選択します。アクセス可能なアプリケーションのリストを表示するページが表示されます。
2. 発行するアプリケーションの編集を選択します。
3. 右上隅にある発行を選択します。
4. 更新の発行ダイアログボックスで、次の操作を行います。
 - a. アプリケーションの公開に関する情報を確認します。
 - b. (オプション) バージョンの説明に、このバージョンのアプリケーションの説明を含めます。
 - c. 環境に関する情報を承認するボックスを選択します。
 - d. [開始] を選択します。アプリケーションがライブ環境で更新されるまでに最大 15 分かかる場合があります。
5. テスト環境または本番稼働環境でアプリケーションを表示する方法については、「」を参照してください[公開されたアプリケーションの表示](#)。

Note

テスト環境または本番稼働環境でアプリケーションを使用すると、コネクタに接続されたデータソースのテーブルにレコードを作成するなど、ライブデータ転送が発生します。

一度も共有されていない公開アプリケーションは、ユーザーや他のビルダーでは使用できません。ユーザーがアプリケーションを使用できるようにするには、公開後にアプリケーションを共有する必要があります。詳細については、「[公開されたアプリケーションの共有](#)」を参照してください。

公開されたアプリケーションの表示

テスト環境と本番環境に公開されたアプリケーションを表示して、エンドユーザーや他のビルダーと共有する前にアプリケーションをテストできます。

テスト環境または本番稼働環境で公開されたアプリケーションを表示するには

1. 必要に応じて、プレビューするアプリケーションのアプリケーションスタジオに移動します。
 - a. ナビゲーションペインで、ビルドセクションのマイアプリケーションを選択します。
 - b. アプリケーションの編集を選択します。
2. 右上隅の Publish の横にあるドロップダウン矢印を選択し、Publish Center を選択します。
3. 公開センターから、アプリケーションが公開されている環境を表示できます。アプリケーションがテスト環境または本番環境に公開されている場合は、各環境の URL リンクを使用してアプリケーションを表示できます。

Note

テスト環境または本番稼働環境でアプリケーションを使用すると、コネクタに接続されたデータソースのテーブルにレコードを作成するなど、ライブデータ転送が発生します。

アプリケーション環境

AWS App Studio は、開発、テスト、本番稼働の 3 つの異なる環境でアプリケーションライフサイクル管理 (ALM) 機能を提供します。これにより、アプリケーションのライフサイクル全体で個別の環

境、バージョン管理、共有、モニタリングを維持するなどのベストプラクティスをより簡単に行うことができます。

デベロッパー環境

開発環境は、アプリケーションスタジオとサンプルデータを使用して、ライブデータソースやサービスに接続せずにアプリケーションを構築できる独立したサンドボックスです。開発環境では、本番データを侵害することなく、アプリケーションをプレビューしてアプリケーションを表示およびテストできます。

アプリは開発環境の他のサービスに接続しませんが、ライブデータコネクタとオートメーションを模倣するようにアプリでさまざまなリソースを設定できます。

開発環境のアプリケーションスタジオの下部にエラーと警告を含む折りたたみ可能なデバッグパネルがあり、ビルド時にアプリケーションを検査およびデバッグするのに役立ちます。アプリケーションのトラブルシューティングとデバッグの詳細については、「」を参照してください [App Studio のトラブルシューティングとデバッグ](#)。

環境のテスト

最初のアプリケーション開発が完了したら、次のステップとしてテスト環境に公開します。テスト環境では、アプリは他のサービスに接続したり、データを送信したり、他のサービスからデータを受信したりできます。したがって、この環境を使用して、エンドユーザーにアクセス URL を提供することで、ユーザー受け入れテスト (UAT) を含む包括的なテストを実行できます。

Note

テスト環境への最初の発行には最大 15 分かかる場合があります。

テスト環境に公開されたアプリのバージョンは、エンドユーザーが非アクティブになってから 3 時間後に削除されます。ただし、すべてのバージョンは保持され、バージョン履歴タブから復元できます。

テスト環境の主な機能は次のとおりです。

- ライブデータソースと APIs との統合テスト
- 管理されたアクセスを通じて容易になるユーザー受け入れテスト (UAT)
- フィードバックを集め、問題に対処するための環境

- ブラウザコンソールと開発者ツールを使用して、クライアント側とサーバー側のアクティビティの両方を検査およびデバッグする機能。

アプリケーションのトラブルシューティングとデバッグの詳細については、「」を参照してください [App Studio のトラブルシューティングとデバッグ](#)。

本番環境

テストして問題を修正したら、アプリケーションのバージョンをテスト環境から本番環境に昇格させて、運用をライブで使用できます。本番稼働環境はエンドユーザーが使用するライブ運用環境ですが、公開されたバージョンをユーザーと共有する前にテストできます。

本番稼働環境で公開されたバージョンは、エンドユーザーが 14 日間操作しないと削除されます。ただし、すべてのバージョンは保持され、バージョン履歴タブから復元できます。

本番稼働環境の主な機能は次のとおりです。

- エンドユーザーが使用するためのライブ運用環境
- きめ細かなロールベースのアクセスコントロール
- バージョン管理とロールバック機能
- クライアント側のアクティビティのみを検査およびデバッグする機能
- ライブコネクタ、データ、オートメーション、APIs

バージョンニングとリリース管理

App Studio は、パブリッシュセンターのバージョンニングシステムを通じてバージョン管理とリリース管理機能を提供します。

主要なバージョンニング機能:

- テスト環境に発行すると、新しいバージョン番号 (1.0、2.0、3.0...) が生成されます。
- テスト環境から本番環境に昇格しても、バージョン番号は変更されません。
- バージョン履歴から以前のバージョンにロールバックできます。
- テスト環境に公開されたアプリケーションは、非アクティブ状態が 3 時間続くと一時停止されます。バージョンは保持され、バージョン履歴から復元できます。
- 本番環境に公開されたアプリケーションは、14 日間の非アクティブ状態後に削除されます。バージョンは保持され、バージョン履歴から復元できます。

このバージョンニングモデルにより、アプリケーションの開発とテストサイクル全体でトレーサビリティ、ロールバック機能、最適なパフォーマンスを維持しながら、迅速なイテレーションが可能になります。

メンテナンスとオペレーション

App Studio は、特定のメンテナンスタスク、運用アクティビティに対処し、新しいソフトウェアライブラリを組み込むために、アプリケーションを自動的に再公開する必要がある場合があります。ビルダーは何もする必要はありませんが、エンドユーザーはアプリケーションにログインし直す必要がある場合があります。状況によっては、自動的に追加できない新機能とライブラリを組み込むために、アプリケーションを再公開する必要がある場合があります。再発行する前に、エラーを解決し、警告を確認する必要があります。

公開されたアプリケーションの共有

まだ公開されていないアプリケーションを公開すると、共有されるまでユーザーは使用できなくなります。公開されたアプリケーションが共有されると、ユーザーは使用でき、別のバージョンが公開された場合は再度共有する必要はありません。

Note

このセクションでは、公開されたアプリケーションをエンドユーザーまたはテスターと共有します。他のユーザーを招待してアプリを構築する方法については、「[」を参照してください](#)
[複数のユーザーによるアプリの構築](#)。

公開されたアプリケーションを共有するには

1. 以下の手順に従って、アプリケーションリストまたはアプリケーションのアプリケーションスタジオから共有ダイアログボックスにアクセスします。
 - アプリケーションリストから共有ダイアログボックスにアクセスするには: ナビゲーションペインで、ビルドセクションでアプリケーションを選択します。共有するアプリケーションのアクション列のドロップダウンを選択し、共有を選択します。
 - アプリケーションスタジオから共有ダイアログボックスにアクセスするには: アプリケーションのアプリケーションスタジオから、上部のヘッダーで共有を選択します。
2. 共有ダイアログボックスで、共有する環境のタブを選択します。テストタブまたは本番稼働用タブが表示されない場合、アプリは対応する環境に公開されない可能性があります。公開の詳細については、「[アプリケーションの公開](#)」を参照してください。

3. 適切なタブで、ドロップダウンメニューからグループを選択して環境を共有します。
4. (オプション) 条件付きページの可視性をテストまたは設定するために、アプリケーションレベルのロールをグループに割り当てます。詳細については、「[ページのロールベースの可視性の設定](#)」を参照してください。
5. [共有] を選択します。
6. (オプション) リンクをコピーしてユーザーと共有します。アプリケーションと環境が共有されているユーザーのみが、対応する環境内のアプリケーションにアクセスできます。

以前に公開されたバージョンにロールバックする

App Studio アプリの本番稼働環境を以前に公開されたバージョンにロールバックするには、次の手順に従います。アプリケーションのエンドユーザーが影響を受け、デプロイ後にアプリケーションのロールバックバージョンが表示されます。アプリケーションをロールバックすると、コンポーネントコードも前回の公開時刻のバージョンにロールバックされ、アプリケーションデプロイスタック全体 (ユーザーコード、コンポーネント設定状態) に影響します。つまり、フィールドやその他の設定の変更など、App Studio がコンポーネントコードに対して行った更新はロールバックされ、ロールバックされたアプリケーションバージョンが最初に公開されたときと同じように動作します。

開発環境で進行中のアプリケーションのバージョンは、公開されたバージョンをロールバックしても影響を受けません。

公開されたアプリケーションのバージョンをロールバックすると、公開されたアプリケーションの問題を検出し、以前に動作したバージョンをすぐに公開する必要がある場合や、以前のバージョンを公開して開発環境でアプリケーションに最新の更新を保持する場合に便利です。

Note

アプリケーションの開発環境を以前に公開されたバージョンに戻す場合は、アプリケーションを元に戻す必要があります。詳細については、「[以前に公開されたアプリバージョンを編集する](#)」を参照してください。

本番稼働環境バージョンを以前に公開されたアプリケーションバージョンにロールバックするには

1. 必要に応じて、アプリケーションを編集して開発環境に移動します。詳細については、「[アプリケーションの編集または構築](#)」を参照してください。

2. 本番稼働用環境タイルの上部にあるバージョンドロップダウン矢印を選択すると、ロールバックに使用できるバージョンが表示されます。ドロップダウンには、過去 30 日以内に発行されたバージョンが含まれます。このドロップダウンが無効になっているのは、アプリの発行がすでに進行中であり、同時に実行できる発行は 1 つだけである可能性があります。
3. ロールバックするバージョンを選択します。
4. ロールバックの理由を入力し、ロールバックを選択します。ロールバック公開が開始され、完了すると、アプリケーションの本番稼働環境が選択したバージョンに更新されます。

Note

ロールバック後に、以前に公開されたアプリバージョンにロールフォワードすることもできます。

アプリケーションのエクスポート

アプリケーションのスナップショットをエクスポートして、他の App Studio インスタンスと共有できます。アプリをエクスポートすると、アプリの開発環境からスナップショットが作成され、インポートコードが生成されます。その後、インポートコードを使用して、アプリケーションを表示および構築できる他の App Studio インスタンスにアプリケーションをインポートできます。

エクスポートされたアプリは、App Studio で AWS リージョン サポートされている のインスタンスにインポートできます。

アプリケーションをエクスポートするには

1. ナビゲーションペインで、ビルドセクションのマイアプリケーションを選択して、アプリケーションのリストに移動します。
2. エクスポートするアプリケーションのアクション列でドロップダウンを選択します。
3. [エクスポート] を選択します。
4. インポートコードを生成して共有する手順は、アプリのインポートコードが既に作成されているかどうかによって異なります。
 - インポートコードが作成されていない場合:
 - a. アプリケーションのインポート許可で、エクスポートされたアプリケーションをインポートできるインスタンスを指定します。すべてのインスタンスにインポートアクセス

許可を付与することも、インスタンス IDs を入力して特定の App Studio インスタンスを追加することもできます。複数のインスタンス IDs で区切ります。

インスタンス ID を検索するには、App Studio コンソールでアカウント設定を選択して、インスタンスのアカウント設定に移動します。

- b. インポートコードの生成 を選択します。
 - c. 生成されたインポートコードをコピーして共有します。
- インポートコードがすでに作成されている場合:
 - 現在エクスポートされているアプリを共有するには、既存のインポートコードをコピーして共有します。アプリに最新の変更を加えた新しいエクスポート済みアプリを作成するには、「新しいコードの生成」を選択します。必要に応じてインポート権限を更新することもできます。

ページとコンポーネント: アプリケーションのユーザーインターフェイスを構築する

トピック

- [ページの管理](#)
- [コンポーネントの管理](#)
- [ページのロールベースの可視性の設定](#)
- [アプリナビゲーションでのページの順序付けと整理](#)
- [アプリテーマでアプリの色を変更する](#)
- [コンポーネントリファレンス](#)

ページの管理

AWS App Studio アプリケーションからページを作成、編集、または削除するには、次の手順に従います。

ページは [コンポーネント](#) のコンテナであり、App Studio のアプリケーションの UI を構成します。各ページは、ユーザーが操作するアプリケーションのユーザーインターフェイス (UI) の画面を表します。ページは、アプリケーションスタジオのページタブで作成および編集されます。

ページの作成

App Studio のアプリケーションにページを作成するには、次の手順に従います。既存のページの複製については、「」を参照してください[ページの複製](#)。

ページを作成するには

1. 必要に応じて、アプリケーションを編集して開発環境に移動します。
2. Pages タブに移動します。
3. 左側のページメニューで、+ 追加 を選択します。

ページの複製

App Studio のアプリケーションでページを複製するには、次の手順に従います。

ページを複製するには

1. 必要に応じて、アプリケーションを編集して開発環境に移動します。
2. Pages タブに移動します。
3. 左側のページメニューで、複製するページの名前の横にある楕円メニューを選択し、重複を選択します。複製されたページは、元のページの直後に追加されます。

ページプロパティの表示と編集

App Studio のアプリケーションのページを編集するには、次の手順に従います。ページの名前、パラメータ、レイアウトなどのプロパティを編集できます。

ページのプロパティを表示または編集するには

1. 必要に応じて、アプリケーションを編集して開発環境に移動します。
2. Pages タブに移動します。
3. 左側のページメニューで、編集するページの名前の横にある楕円メニューを選択し、ページプロパティを選択します。これにより、右側のプロパティメニューが開きます。
4. ページ名を編集するには:

Note

有効なページ名文字: A~Z、a~z、0~9、_、\$

- a. プロパティメニューの上部にある名前の横にある鉛筆アイコンを選択します。
 - b. ページの新しい名前を入力し、Enter キーを押します。
5. ページパラメータを作成、編集、または削除するには:
- a. ページパラメータを作成するには、「ページパラメータ」セクションで + 新規追加を選択します。
 - b. ページパラメータのキーまたは説明の値を編集するには、変更するプロパティの入力フィールドを選択し、新しい値を入力します。変更は編集時に保存されます。
 - c. ページパラメータを削除するには、削除するページパラメータのごみ箱アイコンを選択します。
6. ページのロゴまたはバナーを追加、編集、または削除するには:
- a. ページロゴまたはバナーを追加するには、スタイルセクションでそれぞれのオプションを有効にします。イメージのソースを設定し、オプションで代替テキストを指定します。
 - b. ページロゴまたはバナーを編集するには、スタイルセクションのフィールドを更新します。
 - c. ページロゴまたはバナーを削除するには、スタイルセクションでそれぞれのオプションを無効にします。
7. ページのレイアウトを編集するには:
- レイアウトセクションのフィールドを更新します。

ページの削除

App Studio のアプリケーションからページを削除するには、次の手順に従います。

ページを削除するには

1. 必要に応じて、アプリケーションを編集して開発環境に移動します。
2. Pages タブに移動します。

3. 左側のページメニューで、削除するページの名前の横にある楕円メニューを選択し、削除を選択します。

コンポーネントの管理

次の手順を使用して、App Studio アプリケーションスタジオのページとの間でコンポーネントを追加、編集、削除し、アプリケーションに必要なユーザーインターフェイスを作成します。

ページへのコンポーネントの追加

App Studio のページにコンポーネントを追加するには、次の手順に従います。既存のコンポーネントを複製する方法については、「」を参照してください[コンポーネントの複製](#)。

1. 必要に応じて、アプリケーションを編集して開発環境に移動します。
2. Pages タブに移動します。
3. コンポーネントパネルは、使用可能なコンポーネントを含む右側のメニューにあります。
4. パネルから目的のコンポーネントをキャンバスにドラッグアンドドロップします。または、パネル内のコンポーネントをダブルクリックして、現在のページの中央に自動的に追加することもできます。
5. コンポーネントを追加したので、右側のプロパティパネルを使用して、データソース、レイアウト、動作などの設定を調整します。各コンポーネントタイプの設定の詳細については、「」を参照してください[コンポーネントリファレンス](#)。

コンポーネントの複製

App Studio アプリでコンポーネントを複製するには、次の手順に従います。複製されたコンポーネントには、元のコンポーネントのリンクされたオートメーションまたはエンティティが含まれます。

1. 必要に応じて、アプリケーションを編集して開発環境に移動します。
2. Pages タブに移動します。
3. コンポーネントを複製する方法は 2 つあります。
 - 左側のページメニューで、複製するコンポーネントを含むページを展開します。複製するコンポーネントの名前の横にある楕円メニューを選択し、複製を選択します。
 - 複製するコンポーネントを選択し、複製アイコンを選択します。

複製されたコンポーネントは、元のコンポーネントの直後に追加されます。

i Tip

CTRL+Z または CMD+Z キーボードショートカットを使用して、開発環境の他の多くのアクションとともに、コンポーネントの重複を元に戻すことができます。

コンポーネントプロパティの表示と編集

1. 必要に応じて、アプリケーションを編集して開発環境に移動します。
2. Pages タブに移動します。
3. 左側のページメニューで、コンポーネントを含むページを展開し、表示または編集するコンポーネントを選択します。または、ページを選択し、キャンバスからコンポーネントを選択することもできます。
4. 右側のプロパティパネルには、選択したコンポーネントの設定内容が表示されます。
5. 使用可能なさまざまなプロパティとオプションを調べ、必要に応じて更新してコンポーネントの外観と動作を設定します。たとえば、データソースを変更したり、レイアウトを設定したり、追加機能を有効にしたりできます。

各コンポーネントタイプの設定の詳細については、「」を参照してください [コンポーネントリファレンス](#)。

コンポーネントの削除

1. 必要に応じて、アプリケーションを編集して開発環境に移動します。
2. Pages タブに移動します。
3. 左側のページメニューで、削除するコンポーネントを選択して選択します。
4. 右側のプロパティメニューで、ごみ箱アイコンを選択します。
5. 確認ダイアログボックスで、[削除] を選択します。

ページのロールベースの可視性の設定

App Studio アプリ内でロールを作成し、それらのロールに基づいてページの可視性を設定できます。たとえば、プロジェクトの承認やクレーム処理などの機能を提供し、特定のページを特定のロールに表示するアプリケーションの管理者、マネージャー、ユーザーなど、ユーザーのニーズやアクセ

スレベルに基づいてロールを作成できます。この例では、管理者はフルアクセス、マネージャーはレポートダッシュボードを表示するアクセス、ユーザーは入力フォームを含むタスクページにアクセスできる場合があります。

App Studio アプリのページのロールベースの可視性を設定するには、次の手順に従います。

1. 必要に応じて、アプリケーションのアプリケーションスタジオに移動します。左側のナビゲーションメニューから、マイアプリケーションを選択し、アプリケーションを検索して編集を選択します。
2. アプリケーションスタジオでアプリケーションレベルのロールを作成します。
 - a. アプリケーションスタジオの上部にあるアプリケーション設定タブを選択します。
 - b. 選択 + ロールの追加
 - c. ロール名で、ロールを識別する名前を指定します。名前を使用してページの可視性を設定するため、グループのアクセスレベルまたは職務を説明する名前を使用することをお勧めします。
 - d. 必要に応じて、説明でロールの説明を追加します。
 - e. これらのステップを繰り返して、必要な数のロールを作成します。
3. ページの可視性を設定する
 - a. アプリケーションスタジオの上部にあるページタブを選択します。
 - b. 左側のページメニューから、ロールベースの可視性を設定するページを選択します。
 - c. 右側のメニューで、プロパティタブを選択します。
 - d. 可視性で、すべてのエンドユーザーに対して Open を無効にします。
 - e. 前のステップで作成したロールのリストから選択するロールを選択したままにします。カスタムを選択して、より複雑な可視性設定のための JavaScript 式を記述します。
 1. ロールを選択し、ページが表示されるアプリロールのチェックボックスをオンにします。
 2. Custom を選択し、true または false に解決される JavaScript 式を入力します。次の例を使用して、現在のユーザーがマネージャーのロールを持っているかどうかを確認します。

```
{{currentUser.roles.includes('manager')}}
```
4. 可視性が設定されたら、アプリをプレビューしてページの可視性をテストできます。
 - a. プレビュー を選択して、アプリのプレビューを開きます。

- b. プレビューの右上で、プレビューをメニューとして選択し、テストするロールのチェックボックスをオンにします。表示されるページには、選択したロールが反映されている必要があります。
5. 次に、公開されたアプリのアプリロールにグループを割り当てます。グループとロールの割り当ては、環境ごとに個別に設定する必要があります。アプリケーション環境の詳細については、「」を参照してください[アプリケーション環境](#)。

Note

App Studio グループを作成および設定したロールに割り当てるには、アプリケーションをテスト環境または本番稼働環境に公開する必要があります。必要に応じて、アプリを公開してロールにグループを割り当てます。公開の詳細については、「[アプリケーションの公開](#)」を参照してください。

- a. アプリケーションスタジオの右上で、共有を選択します。
- b. ページの可視性を設定する環境のタブを選択します。
- c. 検索グループ入力ボックスを選択し、アプリバージョンを共有するグループを選択します。テキストを入力してグループを検索できます。
- d. ドロップダウンメニューで、グループに割り当てるロールを選択します。アプリケーションバージョンを共有するロールなしを選択し、グループにロールを割り当てません。ロールのないグループには、すべてのユーザーに表示されるページのみが表示されます。
- e. [共有] を選択します。これらのステップを繰り返して、必要な数のグループを追加します。

アプリナビゲーションでのページの順序付けと整理

このトピックでは、App Studio アプリケーションのページの順序変更と整理について説明します。製品には、アプリケーションページが表示される 2 つの領域があります。アプリケーションスタジオでアプリケーションを編集する際の左側のページメニューと、公開されたアプリケーションのプレビューの左側のナビゲーションです。

アプリの編集時の左側のページメニューでのページの順序付け

アプリケーションスタジオでアプリケーションを編集する際、ページは左側のページメニューで作成時刻順に並べられます。このメニューのページの順序を変更することはできません。

プレビューまたは公開アプリケーションのナビゲーションでのページの順序付け、表示、非表示

プレビューまたは公開されたアプリケーションの左側のナビゲーションの次の設定を編集できます。

- ナビゲーション全体の可視性
- ナビゲーション内の特定のページの可視性
- ナビゲーション内のページの順序

プレビューまたは公開されたアプリケーションの左側のナビゲーションを編集するには

1. 必要に応じて、アプリケーションのアプリケーションスタジオに移動して編集します。
2. 左側のページメニューで、ヘッダーとナビゲーションを選択します。
3. 右側のヘッダーとナビゲーションメニューで、以下を表示または編集します。
 - a. アプリでナビゲーションを非表示または表示するには、アプリナビゲーショントグルを使用します。
 - b. アプリのナビゲーションからページを非表示にするには、リンクされていないページセクションにページをドラッグします。
 - c. アプリのナビゲーションでページを並べ替えるには、リンクされたページセクションで目的の順序にドラッグします。

アプリテーマでアプリの色を変更する

アプリケーションテーマを設定してアプリケーションの色を更新するには、次の手順に従います。

1. 必要に応じて、アプリケーションのアプリケーションスタジオに移動して編集します。
2. アプリケーションスタジオで、ページタブに移動します。
3. 左側のナビゲーションで、アプリテーマを選択して右側のアプリテーマ設定を開きます。
4. ベーステーマで、ライトモードまたはダークモードを選択します。
5. アプリケーションにカスタム色を追加するには、カスタマイズトグルを有効にし、次の設定を更新します。
 - a. プライマリ色で、特定のコンポーネントに適用される色とアプリのナビゲーションを選択します。カラーピッカー、RGB、HSL、または HEX コードを使用して色を選択できます。

Note

App Studio は、色に自動的にアクセスできるようにします。例えば、ライトモードで明るい色を選択すると、よりアクセスしやすいように更新されます。

- b. ヘッダーの色で、アプリのヘッダーに適用される色を選択します。カラーピッカー、RGB、HSL、または HEX コードを使用して色を選択できます。
 - c. デフォルトのテーマを選択して事前定義されたテーマを表示および選択するか、ランダム化を選択してランダムなプライマリとヘッダーの色を生成します。
6. 変更の保存を選択して、アプリのテーマを更新します。

コンポーネントリファレンス

このトピックでは、App Studio の各コンポーネントとそのプロパティについて詳しく説明し、設定例を示します。

一般的なコンポーネントプロパティ

このセクションでは、アプリケーションスタジオの複数のコンポーネント間で共有される一般的なプロパティと機能の概要を説明します。各プロパティタイプの具体的な実装の詳細とユースケースはコンポーネントによって異なりますが、これらのプロパティの一般的な概念は App Studio 全体で一貫しています。

名前

デフォルト名はコンポーネントごとに生成されますが、編集して各コンポーネントに一意の名前を変更できます。この名前を使用して、コンポーネントと、同じページ内の他のコンポーネントまたは式からのデータを参照します。制限: コンポーネント名にはスペースを含めないでください。文字、数字、アンダースコア、ドル記号のみを使用できます。例: `userNameInput`、`ordersTable`、`metricCard1`。

プライマリ値、セカンダリ値、および値

アプリケーションスタジオの多くのコンポーネントには、コンポーネント内に表示されるコンテンツまたはデータを決定する値または式を指定するためのフィールドが用意されています。これらのフィールドは、多くの場合、コンポーネントタイプと目的に応じて `Value`、`Primary valueSecondary value`、または単に というラベルが付けられます。

Primary value フィールドは通常、コンポーネント内で目立つように表示する必要があるメイン値、データポイント、またはコンテンツを定義するために使用されます。

Secondary value フィールドは、利用可能な場合、プライマリ値とともに追加またはサポートする値または情報を表示するために使用されます。

Value フィールドでは、コンポーネントに表示する値または式を指定できます。

これらのフィールドは、静的テキスト入力式と動的式の両方をサポートします。式を使用すると、アプリケーション内の他のコンポーネント、データソース、または変数のデータを参照できるため、動的でデータ駆動型のコンテンツ表示が可能になります。

式の構文

これらのフィールドに式を入力するための構文は、一貫したパターンに従います。

```
{{expression}}
```

ここで、*expression* は、表示する値またはデータに評価される有効な式です。

例: 静的テキスト

- プライマリ値: "123"や などの静的な数値を直接入力できます"\$1,999.99"。
- セカンダリ値: "Goal"や などの静的テキストラベルを入力できます"Projected Revenue"。
- 値: "since last month"や などの静的文字列を入力できます"Total Quantity"。

例: 式

- Hello, {{currentUser.firstName}}: 現在ログインしているユーザーの名で挨拶を表示します。
- {{currentUser.role === 'Admin' ? 'Admin Dashboard' : 'User Dashboard'}}: 条件付きで、ユーザーのロールに基づいて異なるダッシュボードタイトルを表示します。
- {{ui.componentName.data?.[0]?.fieldName}}: ID のコンポーネントのデータの最初の項目から fieldName フィールドの値を取得します componentName。
- {{ui.componentName.value * 100}}: ID のコンポーネントの値に対して計算を実行します componentName。
- {{ui.componentName.value + ' items'}}: コンポーネントの値を ID componentName と文字列 と連結します ' items'。

- `{{ui.ordersTable.data?.[0]?.orderNumber}}`: ordersTableコンポーネント内のデータの最初の行から注文番号を取得します。
- `{{ui.salesMetrics.data?.[0]?.totalRevenue * 1.15}}`: salesMetricsコンポーネントのデータの最初の行から総収益を 15% 増やして、予測収益を計算します。
- `{{ui.customerProfile.data?.[0]?.firstName + ' ' + ui.customerProfile.data?.lastName}}`: customerProfileコンポーネント内のデータから名と姓を連結します。
- `{{new Date(ui.orderDetails.data?.orderDate).toLocaleDateString()}}`: orderDetailsコンポーネントからの注文日をより読みやすい日付文字列にフォーマットします。
- `{{ui.productList.data?.length}}`: productListコンポーネントに接続されたデータ内の製品の合計数を表示します。
- `{{ui.discountPercentage.value * ui.orderTotal.value}}`: 割引率と注文合計に基づいて割引額を計算します。
- `{{ui.cartItemCount.value + ' items in cart'}}`: ショッピングカート内の項目数をラベルとともに表示します items in cart。

これらの式フィールドを使用すると、アプリケーション内で動的コンテンツとデータ駆動型コンテンツを作成し、ユーザーのコンテキストまたはアプリケーションの状態に合わせた情報を表示できます。これにより、よりパーソナライズされたインタラクティブなユーザーエクスペリエンスが可能になります。

ラベル

Label プロパティを使用すると、コンポーネントの字幕またはタイトルを指定できます。このラベルは通常、コンポーネントの横または上に表示され、ユーザーがその目的を理解するのに役立ちます。

静的テキストと式の両方を使用してラベルを定義できます。

例: 静的テキスト

Label フィールドにテキスト「First Name」を入力すると、コンポーネントはラベルとして「First Name」を表示します。

例: 式

例: 小売店

次の例では、各ユーザーのラベルをパーソナライズし、インターフェイスを個々のユーザーに合わせてカスタマイズします。

```
{{currentUser.firstName}} {{currentUser.lastName}}'s Account
```

例: SaaS プロジェクト管理

次の例では、選択したプロジェクトからデータを取得してコンテキスト固有のラベルを提供し、ユーザーがアプリケーション内で向きを維持できるようにします。

```
Project {{ui.projectsTable.selectedRow.id}} - {{ui.projectsTable.selectedRow.name}}
```

例: 医療施設

次の例では、現在のユーザーのプロファイルと医師の情報を参照し、患者によりパーソナライズされたエクスペリエンスを提供します。

```
Dr. {{ui.doctorProfileTable.data.firstName}}  
    {{ui.doctorProfileTable.data.lastName}}
```

Placeholder

Placeholder プロパティを使用すると、コンポーネントが空の場合に表示されるヒントまたはガイダンステキストを指定できます。これにより、ユーザーは予想される入力形式を理解したり、追加のコンテキストを提供したりできます。

静的テキストと式の両方を使用してプレースホルダーを定義できます。

例: 静的テキスト

Enter your name Placeholder フィールドにテキストを入力すると、コンポーネントはプレースホルダーテキスト Enter your name として表示されます。

例: 式

例: 金融サービス

Enter the amount you'd like to deposit into your
{{ui.accountsTable.selectedRow.balance}} account これらの例では、選択したアカウ

ントからデータを取得して関連するプロンプトを表示し、銀行のお客様にインターフェイスを直感的に使用できるようにします。

例: e コマース

Enter the coupon code for `{{ui.cartTable.data.currency}}` total ここでのプレースホルダーは、ユーザーのカートの内容に基づいて動的に更新され、シームレスなチェックアウトエクスペリエンスを提供します。

例: 医療施設

Enter your `{{ui.patientProfile.data.age}}`-year-old patient's symptoms 患者の年齢を参照する式を使用することで、アプリケーションはよりパーソナライズされた便利なプレースホルダーを作成できます。

ソース

Source プロパティを使用すると、コンポーネントのデータソースを選択できます。選択時に、`entity`、`expression`または `automation` のデータソースタイプから選択できます。

エンティティ

Entity をデータソースとして選択すると、コンポーネントをアプリケーションの既存のデータエンティティまたはモデルに接続できます。これは、アプリケーション全体で活用したい明確に定義されたデータ構造またはスキーマがある場合に便利です。

エンティティデータソースを使用するタイミング:

- コンポーネントに表示する情報を含むデータモデルまたはエンティティがある場合 (たとえば、「名前」、「説明」、「価格」などのフィールドを持つ「製品」エンティティ)。
- データベース、API、またはその他の外部データソースからデータを動的に取得し、コンポーネントに表示する必要がある場合。
- アプリケーションのデータモデルで定義されている関係と関連付けを活用する場合。

エンティティでのクエリの選択

場合によっては、エンティティ全体ではなく、エンティティからデータを取得する特定のクエリにコンポーネントを接続することができます。エンティティデータソースでは、既存のクエリから選択するか、新しいクエリを作成するかを選択できます。

クエリを選択すると、次のことができます。

- 特定の条件に基づいて、コンポーネントに表示されるデータをフィルタリングします。
- クエリにパラメータを渡して、データを動的にフィルタリングまたはソートします。
- クエリで定義された複雑な結合、集約、またはその他のデータ操作手法を活用します。

たとえば、アプリケーションに Name、 、 などのフィールドを持つ Customers エンティティがある場合 Email です PhoneNumber。テーブルコンポーネントをこのエンティティに接続し、ステータスに基づいて顧客をフィルタリングする事前定義された Active Customers データアクションを選択できます。これにより、顧客データベース全体ではなく、アクティブな顧客のみをテーブルに表示できます。

エンティティデータソースへのパラメータの追加

エンティティをデータソースとして使用する場合は、コンポーネントにパラメータを追加することもできます。これらのパラメータを使用して、コンポーネントに表示されるデータをフィルタリング、ソート、または変換できます。

たとえば、 、 Name、 、 Description Price などのフィールドを持つ Products エンティティがある場合です Category。製品リストを表示するテーブルコンポーネント category に という名前のパラメータを追加できます。ユーザーがドロップダウンからカテゴリを選択すると、データアクションの `{{params.category}}` 式を使用して、選択したカテゴリに属する製品のみを表示するようにテーブルが自動的に更新されます。

式

カスタム式または計算を入力するデータソースとして式を選択し、コンポーネントのデータを動的に生成します。これは、変換を実行したり、複数のソースからのデータを結合したり、特定のビジネスロジックに基づいてデータを生成したりする必要がある場合に便利です。

Expression データソースを使用するタイミング:

- データモデルで直接利用できないデータを計算または取得する必要がある場合 (例: 数量と価格に基づいて合計注文価格を計算する)。
- 複数のエンティティまたはデータソースのデータを組み合わせて複合ビューを作成する場合 (顧客の注文履歴と連絡先情報の表示など)。
- 特定のルールまたは条件に基づいてデータを生成する必要がある場合 (ユーザーの閲覧履歴に基づいて「推奨製品」リストを表示するなど)。

たとえば、現在の月の合計収益を表示する必要がある#####コンポーネントがある場合、次のような式を使用して月別収益を計算して表示できます。

```
{{ui.table1.orders.concat(ui.table1.orderDetails).filter(o => o.orderDate.getMonth()  
=== new Date().getMonth()).reduce((a, b) => a + (b.quantity * b.unitPrice), 0)}}
```

オートメーション

Automation をデータソースとして選択し、コンポーネントをアプリケーションの既存のオートメーションまたはワークフローに接続します。これは、コンポーネントのデータまたは機能が特定のプロセスまたはワークフローの一部として生成または更新される場合に便利です。

Automation データソースを使用するタイミング:

- コンポーネントに表示されるデータが特定の自動化またはワークフローの結果である場合 (承認プロセスの一部として更新される「承認保留中」テーブルなど)。
- オートメーション内のイベントまたは条件に基づいて、コンポーネントに対するアクションまたは更新をトリガーする場合 (例: SKU の最新の売上数値でメトリクスを更新する)。
- オートメーション (サードパーティー API からデータを取得してテーブルに表示するなど) を使用して、コンポーネントをアプリケーションの他のサービスまたはシステムと統合する必要がある場合。

たとえば、ジョブアプリケーションプロセスを通じてユーザーをガイドするステップフローコンポーネントがある場合です。ステップフローコンポーネントは、ジョブアプリケーションの送信、バックグラウンドチェック、オファー生成を処理するオートメーションに接続できます。自動化がこれらのステップを進めるにつれて、ステップフローコンポーネントはアプリケーションの現在のステータスを反映するように動的に更新できます。

各コンポーネントに適したデータソースを慎重に選択することで、アプリケーションのユーザーインターフェイスが適切なデータとロジックを利用できるようにし、ユーザーにシームレスで魅力的なエクスペリエンスを提供できます。

次の場合に表示

Visible if プロパティを使用して、特定の条件またはデータ値に基づいてコンポーネントまたは要素を表示または非表示にします。これは、アプリケーションのユーザーインターフェイスの特定部分の可視性を動的に制御する場合に便利です。

プロパティが次の構文を使用している場合に表示されます。

```
{{expression ? true : false}}
```

または

```
{{expression}}
```

ここで、*expression* は、trueまたは のいずれかに評価されるブール式ですfalse。

式が trueと評価されるとtrue、コンポーネントが表示されます。式が falseに評価されるとfalse、コンポーネントは非表示になります。式は、アプリケーション内の他のコンポーネント、データソース、または変数の値を参照できます。

式の例の場合に表示されます

例: E メール入力に基づいてパスワード入力フィールドを表示または非表示にする

E メール入力フィールドとパスワード入力フィールドを含むログインフォームがあるとします。ユーザーが E メールアドレスを入力した場合にのみ、パスワード入力フィールドを表示します。式の場合、次の表示を使用できます。

```
{{ui.emailInput.value !== ""}}
```

この式は、emailInputコンポーネントの値が空の文字列ではないかどうかを確認します。ユーザーが E メールアドレスを入力した場合、式は trueと評価されtrue、パスワード入力フィールドが表示されます。E メールフィールドが空の場合、式は falseに評価されfalse、パスワード入力フィールドは非表示になります。

例: ドロップダウン選択に基づいて追加のフォームフィールドを表示する

ユーザーがドロップダウンリストからカテゴリを選択できるフォームがあるとします。選択したカテゴリに応じて、追加のフォームフィールドを表示または非表示にして、より具体的な情報を収集します。

たとえば、ユーザーが **##**カテゴリを選択した場合、次の式を使用して追加の **####**フィールドを表示できます。

```
{{ui.categoryDropdown.value === "Products"}}
```

ユーザーが **####**または **#####**カテゴリを選択した場合、この式を使用して、別の追加フィールドのセットを表示できます。

```
{{ui.categoryDropdown.value === "Services" || ui.categoryDropdown.value === "Consulting"}}
```

例: その他

コンポーネントの値が空の文字列でない場合にtextInput1コンポーネントを表示するには:

```
{{ui.textInput1.value === "" ? false : true}}
```

コンポーネントを常に表示するには:

```
{{true}}
```

コンポーネントの値が空の文字列でない場合にemailInputコンポーネントを表示するには:

```
{{ui.emailInput.value !== ""}}
```

次の場合に無効

Disabled if 機能を使用すると、特定の条件またはデータ値に基づいてコンポーネントを条件付きで有効または無効にできます。これは、コンポーネントを有効または無効にするかどうかを決定するブール式を受け入れる Disabled if プロパティを使用して実現されます。

Disabled if プロパティは次の構文を使用します。

```
{{expression ? true : false}}
```

または

```
{{expression}}
```

式の例の場合、無効

例: フォームの検証に基づいて送信ボタンを無効にする

複数の入力フィールドを持つフォームがあり、すべての必須フィールドが正しく入力されるまで送信ボタンを無効にしたい場合は、次の Disabled If 式を使用できます。

```
{{ui.nameInput.value === "" || ui.emailInput.value === "" || ui.passwordInput.value === ""}}
```

この式は、必須の入力フィールド (nameInput、emailInput、passwordInput) のいずれかが空かどうかを確認します。いずれかのフィールドが空の場合、式は `true` に評価され、送信ボタンは無効になります。必須フィールドがすべて入力されると、式は `false` と評価され、送信ボタンが有効になります。

これらのタイプの条件式を「Visible if」および「Disabled if properties」で使用すると、ユーザー入力に適応する動的で応答性の高いユーザーインターフェイスを作成し、アプリケーションのユーザーにより合理化され、関連するエクスペリエンスを提供できます。

ここで、*expression* は `true` または `false` のいずれかに評価されるブール式です。

例:

```
{{ui.textInput1.value === "" ? true : false}}: The component will be Disabled if the textInput1 component's value is an empty string.  
{!!ui.nameInput.isValid || !ui.emailInput.isValid || !ui.passwordInput.isValid}}: The component will be Disabled if any of the named input fields are invalid.
```

コンテナレイアウト

レイアウトプロパティは、コンポーネント内のコンテンツまたは要素がどのように配置および配置されるかを決定します。複数のレイアウトオプションがあり、それぞれがアイコンで表されます。

- 列レイアウト: このレイアウトは、コンテンツまたは要素を 1 つの列に垂直に配置します。
- 2 列レイアウト: このレイアウトでは、コンポーネントを 2 つの等幅列に分割し、コンテンツまたは要素を並べて配置できます。
- 行レイアウト: このレイアウトは、コンテンツまたは要素を 1 行に水平に配置します。

[Orientation] (向き)

- 水平: このレイアウトは、コンテンツまたは要素を 1 行に水平に配置します。
- 垂直: このレイアウトは、コンテンツまたは要素を 1 つの列に垂直に配置します。
- インラインラップ: このレイアウトはコンテンツまたは要素を水平に配置しますが、要素が使用可能な幅を超えた場合は次の行にラップします。

Alignment

- 左: コンテンツまたは要素をコンポーネントの左側に揃えます。

- 中心: コンテンツまたは要素をコンポーネント内で水平にセンタリングします。
- 右: コンテンツまたは要素をコンポーネントの右側に揃えます。

Width

Width プロパティは、コンポーネントの水平サイズを指定します。親コンテナまたは使用可能なスペースに対するコンポーネントの幅を表す 0% ~ 100% のパーセンテージ値を入力できます。

Height

Height プロパティは、コンポーネントの垂直サイズを指定します。「自動」値は、コンポーネントの内容または使用可能なスペースに基づいてコンポーネントの高さを自動的に調整します。

間のスペース

プロパティ間のスペースは、コンポーネント内のコンテンツまたは要素間の間隔またはギャップを決定します。0px (間隔なし) から 64px までの値を 4px (4px、8px、12px など) 単位で選択できます。

[Padding] (パディング)

Padding プロパティは、コンテンツまたは要素とコンポーネントのエッジの間のスペースを制御します。0px (パディングなし) から 64px までの値を 4px (4px、8px、12px など) 単位で選択できます。

背景

Background は、コンポーネントの背景色またはスタイルを有効または無効にします。

これらのレイアウトプロパティは、コンポーネント内のコンテンツの配置と配置に柔軟性を提供し、コンポーネント自体のサイズ、間隔、外観を制御します。

データコンポーネント

このセクションでは、テーブル、詳細、メトリクス、フォーム、リピータコンポーネントなど、アプリケーションスタジオで使用できるさまざまなデータコンポーネントについて説明します。これらのコンポーネントは、アプリケーション内のデータを表示、収集、操作するために使用されます。

[テーブル]

Table コンポーネントは、行と列を含む表形式のデータを表示します。これは、データベースの項目やレコードのリストなどの構造化データを、整理された easy-to-read 方法で提示するために使用されます。

テーブルプロパティ

Table コンポーネントは、、、、などの他のコンポーネントといくつかの共通プロパティを共有NameSourceしますActions。これらのプロパティの詳細については、「」を参照してください[一般的なコンポーネントプロパティ](#)。

テーブルコンポーネントには、一般的なプロパティに加えて、、、、などの特定のプロパティColumnsSearch and exportと設定オプションがありますExpressions。

列

このセクションでは、テーブルに表示する列を定義できます。各列は、次のプロパティで設定できます。

- 形式: フィールドのデータ型。テキスト、数値、日付など。
- 列ラベル: 列のヘッダーテキスト。
- 値: この列に表示するデータソースのフィールド。

このフィールドでは、列セルに表示する値または式を指定できます。式を使用して、接続されたソースまたは他のコンポーネントからのデータを参照できます。

例: `{{currentRow.title}}` - この式は、列セルの現在の行の####フィールドの値を表示します。

- ソートを有効にする: このトグルを使用すると、特定の列のソート機能を有効または無効にできます。有効にすると、ユーザーはこの列の値に基づいてテーブルデータをソートできます。

検索とエクスポート

テーブルコンポーネントには、検索およびエクスポート機能を有効または無効にするための次のトグルが用意されています。

- 検索を表示する 有効にすると、このトグルはテーブルに検索入力フィールドを追加し、ユーザーは表示されたデータを検索してフィルタリングできます。
- エクスポートを表示する 有効にすると、このトグルによってテーブルにエクスポートオプションが追加され、ユーザーは CSV などのさまざまな形式でテーブルデータをダウンロードできるようになります。

Note

デフォルトでは、検索機能はテーブルにロードされたデータに制限されます。検索を完全に使用するには、データのすべてのページをロードする必要があります。

ページあたりの行数

テーブルのページごとに表示する行数を指定できます。その後、ユーザーはページ間を移動してデータセット全体を表示できます。

プリフェッチの制限

各クエリリクエストでプリフェッチするレコードの最大数を指定します。最大数は 3000 です。

アクション

アクションセクションで、次のプロパティを設定します。

- アクションの場所: Pin to right が有効になっている場合、追加されたアクションは、ユーザーのスクロールに関係なく、常にテーブルの右側に表示されます。
- アクション: テーブルにアクションボタンを追加します。これらのボタンは、ユーザーがクリックしたときに指定されたアクションを実行するように設定できます。次に例を示します。
 - コンポーネントアクションを実行する
 - 別のページに移動する
 - データアクションを呼び出す
 - カスタム JavaScript を実行する
 - オートメーションを呼び出す

表現

テーブルコンポーネントには、式と行レベルのアクション機能を使用する複数の領域があり、テーブルの機能やインタラクティブ性をカスタマイズして強化できます。これにより、テーブル内のデータを動的に参照および表示できます。これらの式フィールドを活用することで、動的列を作成し、行レベルのアクションにデータを渡し、アプリケーション内の他のコンポーネントまたは式からテーブルデータを参照できます。

例: 行値を参照する

`{{currentRow.columnName}}` または `{{currentRow["Column Name"]}}` これらの式を使用すると、レンダリングする現在の行の特定の列の値を参照できます。`columnName` または `Column Name` を、参照する列の実際の名前に置き換えます。

例:

- `{{currentRow.productName}}` 現在の行の製品名を表示します。
- `{{currentRow["Supplier Name"]}}` 現在の行のサプライヤー名を表示します。列ヘッダーは#####です。
- `{{currentRow.orderDate}}` 現在の行の注文日を表示します。

例: 選択した行を参照する

`{{ui.table1.selectedRow["columnName"]}}` この式を使用すると、ID `table1` を使用して、テーブルで現在選択されている行の特定の列の値を参照できます。`table1` はテーブルコンポーネントの実際の ID に、`columnName` は参照する列の名前に置き換えます。

例:

- `{{ui.ordersTable.selectedRow["totalAmount"]}}` ID `ordersTable` を使用して、テーブル内の現在選択されている行の合計金額を表示します。
- `{{ui.customersTable.selectedRow["email"]}}` ID `customersTable` を持つテーブルで現在選択されている行の E メールアドレスを表示します。
- `{{ui.employeesTable.selectedRow["department"]}}` ID `employeesTable` を使用して、テーブルで現在選択されている行の部門を表示します。

例: カスタム列の作成

基になるデータアクション、オートメーション、または式から返されたデータに基づいて、テーブルにカスタム列を追加できます。既存の列値と JavaScript 式を使用して、新しい列を作成できます。

例:

- `{{currentRow.quantity * currentRow.unitPrice}}` 数量と単価の列を乗算して、合計価格を表示する新しい列を作成します。

- `{{new Date(currentRow.orderDate).toLocaleDateString()}}` 注文日をより読みやすい形式で表示する新しい列を作成します。
- `{{currentRow.firstName + ' ' + currentRow.lastName + ' (' + currentRow.email + ')'}}` 各行のフルネームと E メールアドレスを表示する新しい列を作成します。

例: 列の表示値のカスタマイズ:

列マッピングのフィールドを設定することで、テーブル列内の Value フィールドの表示値をカスタマイズできます。これにより、表示されたデータにカスタムフォーマットまたは変換を適用できます。

例:

- `{{ currentRow.rating >= 4 ? '##'.repeat(currentRow.rating) : currentRow.rating }}` 各行の評価値に基づいて星の絵文字を表示します。
- `{{ currentRow.category.toLowerCase().replace(/\b\w/g, c => c.toUpperCase()) }}` 行ごとに大文字化された各単語を含むカテゴリ値を表示します。
- `{{ currentRow.status === 'Active' ? '# Active' : '# Inactive' }}`: 各行のステータス値に基づいて色付きの円の絵文字とテキストを表示します。

行レベルのボタンアクション

`{{currentRow.columnName}}` または `{{currentRow["Column Name"]}}`、これらの式を使用して、選択した行のデータを使用して別のページに移動する、行のデータを使用してオートメッセージをトリガーするなど、参照された行のコンテンツを行レベルのアクション内で渡すことができます。

例:

- 行アクション列に編集ボタンがある場合は、パラメータ `{{currentRow.orderId}}` として を渡して、選択した注文の ID を持つ注文編集ページに移動できます。
- 行アクション列に削除ボタンがある場合は、注文を削除する前に顧客に確認メールを送信するオートメッセージ `{{currentRow.customerName}}` に渡すことができます。
- 行アクション列にビューの詳細ボタンがある場合は、注文の詳細を表示した従業員をログに記録するオートメッセージ `{{currentRow.employeeId}}` に渡すことができます。

これらの式フィールドと行レベルのアクション機能を活用することで、特定の要件に基づいてデータを表示および操作する高度にカスタマイズされたインタラクティブなテーブルを作成できます。さらに、行レベルのアクションをアプリケーション内の他のコンポーネントやオートメーションに接続して、シームレスなデータフローと機能を実現できます。

[Detail] (詳細)

詳細コンポーネントは、特定のレコードまたは項目に関する詳細情報を表示するように設計されています。単一のエンティティまたは行に関連する包括的なデータを表示するための専用スペースを提供するため、詳細な詳細を表示したり、データ入力や編集タスクを容易にしたりするのに最適です。

詳細プロパティ

詳細コンポーネントは、、、などの他のコンポーネントといくつかの共通プロパティを共有NameSourceしますActions。これらのプロパティの詳細については、「」を参照してください[一般的なコンポーネントプロパティ](#)。

詳細コンポーネントには、、、などの特定のプロパティFieldsLayoutと設定オプションもありますExpressions。

[レイアウト]

レイアウトセクションでは、詳細コンポーネント内のフィールドの配置と表示をカスタマイズできます。次のようなオプションを設定できます。

- 列の数: フィールドを表示する列の数を指定します。
- フィールドの順序: フィールドをドラッグアンドドロップして、外観の順序を変更します。
- 間隔と整列: コンポーネント内のフィールドの間隔と整列を調整します。

式と例

詳細コンポーネントには、コンポーネント内のデータを動的に参照および表示できるさまざまな式フィールドが用意されています。これらの式を使用すると、アプリケーションのデータとロジックにシームレスに接続する、カスタマイズされたインタラクティブな Detail コンポーネントを作成できます。

例: データを参照する

`{{ui.details.data[0]?.["colName"]}}`: この式を使用すると、Detail コンポーネントに接続されたデータ配列の最初の項目 (インデックス 0) のcolName」という名前の列の値を ID 「details」

で参照できます。"colName" を、参照する列の実際の名前に置き換えます。たとえば、次の式は、「details」コンポーネントに接続されたデータ配列の最初の項目の「customerName」列の値を表示します。

```
{{ui.details.data[0]?."customerName"}}
```

Note

この式は、Detail コンポーネントが参照されているテーブルと同じページにあり、Detail コンポーネント内のテーブルの最初の行のデータを表示する場合に便利です。

例: 条件付きレンダリング

`{{ui.table1.selectedRow["colName"]}}`: ID *table1* を持つテーブル内の選択した行に *colName* という名前の列のデータがある場合、この式は true を返します。これは、テーブルの選択した行が空かどうかに基づいて、詳細コンポーネントを条件付きで表示または非表示にするために使用できます。

例:

この式を Detail コンポーネントの Visible ifプロパティで使用して、テーブル内の選択した行に基づいて条件付きで表示または非表示にすることができます。

```
{{ui.table1.selectedRow["customerName"]}}
```

この式が true と評価された場合 (*table1* コンポーネントの選択した行に *customerName* 列の値がある場合)、Detail コンポーネントが表示されます。式が false と評価された場合 (選択した行が空であるか、customerName」の値がない場合)、Detail コンポーネントは非表示になります。

例: 条件付き表示

`{{(ui.Component.value === "green" ? "#" : ui.Component.value === "yellow" ? "#" : ui.detail1.data?.[0]?."CustomerStatus")}}`: この式は、コンポーネントまたはデータフィールドの値に基づいて絵文字を条件付きで表示します。

内訳:

- `ui.Component.value`: ID コンポーネントを持つ#####値を参照します。

- `=== "green"`: コンポーネントの値が文字列「green」と等しいかどうかを確認します。
- `? "#"`: 条件が true の場合、は緑色の円絵文字を表示します。
- `: ui.Component.value === "yellow" ? "#"`: 最初の条件が false の場合、はコンポーネントの値が文字列「黄色」と等しいかどうかを確認します。
- `? "#"`: 2 番目の条件が true の場合、は黄色の四角形絵文字を表示します。
- `: ui.detail1.data?.[0]?.CustomerStatus`: 両方の条件が false の場合、Detail コンポーネントに接続されたデータ配列の最初の項目の CustomerStatus 値を ID 「detail1」で参照します。

この式を使用して、詳細コンポーネント内のコンポーネントまたはデータフィールドの値に基づいて絵文字または特定の値を表示できます。

メトリクス

メトリクスコンポーネントは、キーメトリクスまたはデータポイントをカードのような形式で表示するビジュアル要素です。重要な情報やパフォーマンス指標を視覚的にわかりやすく表示できるように設計されています。

メトリクスのプロパティ

Metrics コンポーネントは、、、などの他のコンポーネントといくつかの共通プロパティを共有NameSourceしますActions。これらのプロパティの詳細については、「」を参照してください[一般的なコンポーネントプロパティ](#)。

トレンド

メトリクスのトレンド機能を使用すると、表示されるメトリクスのパフォーマンスや経時的な変化を視覚的に表示できます。

トレンド値

このフィールドでは、トレンドの方向と大きさを決定するために使用される値または式を指定できます。通常、これは特定の期間における変更またはパフォーマンスを表す値になります。

例:

```
{{ui.salesMetrics.data?.[0]?.monthOverMonthRevenue}}
```

この式は、「salesMetrics」メトリクスに接続されたデータの最初の項目からmonth-over-monthの収益値を取得します。

肯定的な傾向

このフィールドでは、正の傾向の条件を定義する式を入力できます。式は true または false と評価されます。

例:

```
{{ui.salesMetrics.data?.[0]?.monthOverMonthRevenue > 0}}
```

この式は、month-over-month収益値が 0 より大きいかどうかをチェックし、正の傾向を示します。

負の傾向

このフィールドでは、負の傾向の条件を定義する式を入力できます。式は true または false と評価されます。

例:

```
{{ui.salesMetrics.data?.[0]?.monthOverMonthRevenue < 0}}
```

この式は、month-over-month収益値が 0 未満であるかどうかをチェックし、負の傾向を示します。

カラーバー

この切り替えにより、色付きのバーの表示を有効または無効にして、トレンドステータスを視覚的に示すことができます。

カラーバーの例:

例: 売上メトリクスの傾向

- トレンド値: `{{ui.salesMetrics.data?.[0]?.monthOverMonthRevenue}}`
- 肯定的な傾向: `{{ui.salesMetrics.data?.[0]?.monthOverMonthRevenue > 0}}`
- 負の傾向: `{{ui.salesMetrics.data?.[0]?.monthOverMonthRevenue < 0}}`
- カラーバー: 有効

例: インベントリメトリクスの傾向

- トレンド値: `{{ui.inventoryMetrics.data?.[0]?.currentInventory - ui.inventoryMetrics.data?.[1]?.currentInventory}}`

- 肯定的な傾向: `{{ui.inventoryMetrics.data?.[0]?.currentInventory > ui.inventoryMetrics.data?.[1]?.currentInventory}}`
- 負の傾向: `{{ui.inventoryMetrics.data?.[0]?.currentInventory < ui.inventoryMetrics.data?.[1]?.currentInventory}}`
- カラーバー: 有効

例: 顧客満足度の傾向

- トレンド値: `{{ui.customerSatisfactionMetrics.data?.[0]?.npsScore}}`
- 肯定的な傾向: `{{ui.customerSatisfactionMetrics.data?.[0]?.npsScore >= 8}}`
- 負の傾向: `{{ui.customerSatisfactionMetrics.data?.[0]?.npsScore < 7}}`
- カラーバー: 有効

これらのトレンド関連のプロパティを設定することで、表示されるメトリクスのパフォーマンスまたは経時的な変化を視覚的に表現するメトリクスコンポーネントを作成できます。

これらの式を活用することで、データを動的に参照および表示する高度にカスタマイズされたインタラクティブなメトリクスコンポーネントを作成できるため、アプリケーション内で主要なメトリクス、パフォーマンス指標、データ駆動型の視覚化を表示できます。

メトリクス式の例

プロパティパネルでは、式を入力してタイトル、プライマリ値、セカンダリ値、値キャプションを表示し、値を動的に表示できます。

例: プライマリ値を参照する

`{{ui.metric1.primaryValue}}`: この式を使用すると、同じページ内の他のコンポーネントまたは式から ID *metric1* を持つ Metrics コンポーネントのプライマリ値を参照できます。

例: `{{ui.salesMetrics.primaryValue}}` は *salesMetrics* メトリクスコンポーネントのプライマリ値を表示します。

例: セカンダリ値を参照する

`{{ui.metric1.secondaryValue}}`: この式を使用すると、同じページ内の他のコンポーネントまたは式から ID *metric1* を持つ Metrics コンポーネントのセカンダリ値を参照できます。

例: `{{ui.revenueMetrics.secondaryValue}}` は `revenueMetrics` メトリクスコンポーネントのセカンダリ値を表示します。

例: データを参照する

`{{ui.metric1.data}}`: この式を使用すると、同じページ内の他のコンポーネントまたは式から ID `metric1` を持つ Metrics コンポーネントのデータを参照できます。

例: `{{ui.kpiMetrics.data}}` は `kpiMetrics` メトリクスコンポーネントに接続されているデータを参照します。

例: 特定のデータ値の表示:

`{{ui.metric1.data?.[0]?.id}}`: この式は、ID `metric1` を使用して Metrics コンポーネントに接続されたデータから特定の情報を表示する方法の例です。これは、データの最初の項目の特定のプロパティを表示する場合に便利です。

内訳:

- `ui.metric1`: ID `metric1` を持つ Metrics コンポーネントを参照します。
- `data`: そのコンポーネントに接続されている情報またはデータセットを指します。
- `?.[0]`: そのデータセットの最初の項目またはエントリを意味します。
- `?.id`: その最初の項目またはエントリの **ID** 値または識別子を表示します。

例: `{{ui.orderMetrics.data?.[0]?.orderId}}` は `orderMetrics` メトリクスコンポーネントに接続されたデータの最初の項目の `orderId` `orderId` 値を表示します。

例: データ長の表示

`{{ui.metric1.data?.length}}`: この式は、ID `metric1` を使用して Metrics コンポーネントに接続されたデータの長さ (項目数) を表示する方法を示しています。データ内の項目数を表示する場合に便利です。

内訳:

- `ui.metric1.data`: コンポーネントに接続されているデータセットを参照します。
- `?.length`: そのデータセット内の項目またはエントリの合計数または数にアクセスします。

例: `{{ui.productMetrics.data?.length}}` は、`productMetrics` メトリクスコンポーネントに接続されたデータ内の項目の数を表示します。

リピータ

Repeater コンポーネントは、提供されたデータソースに基づいて要素のコレクションを生成および表示できる動的コンポーネントです。これは、アプリケーションのユーザーインターフェイス内でリスト、グリッド、または繰り返しパターンの作成を容易にするように設計されています。ユースケースには次のようなものがあります。

- アカウント内の各ユーザーのカードの表示
- イメージとカートに追加するボタンを含む製品のリストを表示する
- ユーザーがアクセスできるファイルのリストの表示

Repeater コンポーネントは、豊富なコンテンツを持つテーブルコンポーネントと区別されます。テーブルコンポーネントには、厳密な行と列の形式があります。Repeater は、より柔軟にデータを表示できます。

リピータプロパティ

Repeater コンポーネントは、、、などの他のコンポーネントといくつかの共通プロパティを共有NameSourceしますActions。これらのプロパティの詳細については、「」を参照してください [一般的なコンポーネントプロパティ](#)。

一般的なプロパティに加えて、Repeater コンポーネントには次の追加のプロパティと設定オプションがあります。

項目テンプレート

項目テンプレートは、データソース内の項目ごとに繰り返される構造とコンポーネントを定義できるコンテナです。テキスト、イメージ、ボタン、または各項目を表すために必要なその他のコンポーネントなど、他のコンポーネントをこのコンテナにドラッグアンドドロップできます。

項目テンプレートでは、形式の式を使用して、現在の項目のプロパティまたは値を参照できます `{{currentItem.propertyName}}`。

たとえば、データソースに `itemName` プロパティが含まれている場合、`{{currentItem.itemName}}` を使用して現在の項目の項目名 (複数可) を表示できます。

[レイアウト]

レイアウトセクションでは、リピータコンポーネント内の繰り返し要素の配置を設定できます。

[Orientation] (向き)

- リスト: 繰り返し要素を 1 つの列に垂直に配置します。
- グリッド: グリッドレイアウト内の反復要素を複数の列で配置します。

1 ページあたりの行数

リストレイアウトでページごとに表示する行数を指定します。ページ分割は、指定された行数をオーバーフローする項目に対して提供されます。

ページあたりの列と行 (グリッド)

- 列: グリッドレイアウトの列数を指定します。
- ページあたりの行数: グリッドレイアウトでページごとに表示する行数を指定します。ページ分割は、指定されたグリッドディメンションをオーバーフローする項目に対して提供されます。

式と例

Repeater コンポーネントには、コンポーネント内のデータを動的に参照および表示できるさまざまな式フィールドが用意されています。これらの式を使用すると、アプリケーションのデータとロジックにシームレスに接続するカスタマイズされたインタラクティブな Repeater コンポーネントを作成できます。

例: 項目を参照する

- `{{currentItem.propertyName}}`: 項目テンプレート内の現在の項目のプロパティまたは値を参照します。
- `{{ui.repeaterID[index]}}`: リピータコンポーネントの特定の項目をインデックスで参照します。

例: 製品リストのレンダリング

- ソース: データソースとして *Products* エンティティを選択します。
- 項目テンプレート: 中にテキストコンポーネントを含むコンテナコンポーネントを追加して製品名 (`{{currentItem.productName}}`) を表示し、イメージコンポーネントを追加して製品イメージ (`{{currentItem.productImageUrl}}`) を表示します。
- レイアウト: `Orientation` を に設定し、`Rows per Page` 必要に応じて を調整します。

例: ユーザーアバターのグリッドの生成

- ソース: 式を使用して、ユーザーデータの配列 (例: `[{name: 'John', avatarUrl: '...'}, {...}, {...}]`) を生成します。
- 項目テンプレート: イメージコンポーネントを追加し、そのSourceプロパティを に設定します `{{currentItem.avatarUrl}}`。
- レイアウト: Orientationを に設定しGrid、Columnsと の数を指定しRows per Page、Padding必要に応じて Space Between と を調整します。

Repeater コンポーネントを使用することで、動的でデータ駆動型のユーザーインターフェイスを作成し、要素のコレクションをレンダリングするプロセスを合理化し、手動の繰り返しやハードコーディングの必要性を軽減できます。

フォーム

フォームコンポーネントは、ユーザー入力をキャプチャし、アプリケーション内のデータ入力タスクを容易にするように設計されています。入力フィールド、ドロップダウン、チェックボックス、その他のフォームコントロールを表示するための構造化されたレイアウトを提供し、ユーザーはデータをシームレスに入力または変更できます。テーブルなど、フォームコンポーネント内に他のコンポーネントをネストできます。

フォームプロパティ

Form コンポーネントは、、、、などの他のコンポーネントといくつかの共通プロパティを共有NameSourceしますActions。これらのプロパティの詳細については、「」を参照してください [一般的なコンポーネントプロパティ](#)。

フォームの生成

フォームの生成機能を使用すると、選択したデータソースに基づいてフォームフィールドを自動的に入力することで、フォームフィールドをすばやく作成できます。これにより、多数のフィールドを表示する必要があるフォームを構築する際の時間と労力を節約できます。

フォームの生成機能を使用するには:

1. フォームコンポーネントのプロパティで、フォームの生成セクションを見つけます。
2. フォームフィールドの生成に使用するデータソースを選択します。これは、アプリケーションで使用できるエンティティ、ワークフロー、またはその他のデータソースです。

3. フォームフィールドは、フィールドラベル、タイプ、データマッピングなど、選択したデータソースに基づいて自動的に生成されます。
4. 生成されたフィールドを確認し、検証ルールの追加やフィールドの順序の変更など、必要なカスタマイズを行います。
5. フォーム設定に満足したら、送信を選択して生成されたフィールドをフォームコンポーネントに適用します。

フォームの生成機能は、アプリケーションにユーザー入力をキャプチャする必要がある明確に定義されたデータモデルまたはエンティティのセットがある場合に特に便利です。フォームフィールドを自動的に生成することで、時間を節約し、アプリケーションのフォーム間で整合性を確保できます。

フォームの生成機能を使用した後、特定の要件に合わせてフォームコンポーネントのレイアウト、アクション、式をさらにカスタマイズできます。

式と例

他のコンポーネントと同様に、式を使用して Form コンポーネント内のデータを参照および表示できます。例えば、次のようになります。

- `{{ui.userForm.data.email}}`: フォームコンポーネントに接続されたデータソースの email フィールドの値を ID で参照します userForm。

Note

一般的なプロパティのその他の式の例 [一般的なコンポーネントプロパティ](#) については、「」を参照してください。

これらのプロパティを設定し、式を活用することで、アプリケーションのデータソースとロジックとシームレスに統合するカスタマイズされたインタラクティブな Form コンポーネントを作成できます。これらのコンポーネントを使用して、ユーザー入力をキャプチャし、事前入力されたデータを表示し、フォームの送信またはユーザーインタラクションに基づいてアクションをトリガーできます。

ステップフロー

Stepflow コンポーネントは、アプリケーション内の複数ステップのプロセスまたはワークフローを通じてユーザーをガイドするように設計されています。一連のステップを表示するための構造化され

た直感的なインターフェイスを提供します。各ステップには独自の入力、検証、アクションのセットがあります。

Stepflow コンポーネントは、、、などの他のコンポーネントといくつかの共通プロパティを共有NameSourceしますActions。これらのプロパティの詳細については、「」を参照してください[一般的なコンポーネントプロパティ](#)。

Stepflow コンポーネントには、、、などの追加のプロパティStep NavigationValidationと設定オプションがありますExpressions。

AI コンポーネント

生成 AI

Gen AI コンポーネントは、コンポーネントとその付随するロジックをグループ化して、アプリケーションスタジオ内のチャットを使用して AI で簡単に編集するために使用されるグループ化コンテナです。チャットを使用してコンポーネントを作成すると、コンポーネントは Gen AI コンテナにグループ化されます。このコンポーネントの編集または使用については、「」を参照してください[アプリの構築または編集](#)。

テキストと数値のコンポーネント

テキスト入力

テキスト入力コンポーネントを使用すると、ユーザーはアプリケーション内でテキストデータを入力および送信できます。名前、住所、その他のテキスト情報など、ユーザー入力を簡単かつ直感的にキャプチャできます。

- `{{ui.inputTextID.value}}`: 入力フィールドに指定された値を返します。
- `{{ui.inputTextID.isValid}}`: 入力フィールドに指定された値の有効性を返します。

テキスト

Text コンポーネントは、アプリケーション内でテキスト情報を表示するために使用されます。静的テキスト、動的値、または式から生成されたコンテンツを表示するために使用できます。

テキスト領域

テキストエリアコンポーネントは、ユーザーからの複数行のテキスト入力をキャプチャするように設計されています。これにより、ユーザーが説明、メモ、コメントなどの長いテキストエントリを入力するための入力フィールド領域が大きくなります。

- `{{ui.textAreaID.value}}`: テキスト領域に指定された値を返します。
- `{{ui.textAreaID.isValid}}`: テキスト領域で指定された値の有効性を返します。

E メール

E メールコンポーネントは、ユーザーの E メールアドレスをキャプチャするように設計された特殊な入力フィールドです。特定の検証ルールを適用して、入力した値が正しい E メール形式に従っていることを確認できます。

- `{{ui.emailID.value}}`: E メール入力フィールドに指定された値を返します。
- `{{ui.emailID.isValid}}`: E メール入力フィールドに指定された値の有効性を返します。

[パスワード]

パスワードコンポーネントは、ユーザーがパスワードや PIN コードなどの機密情報を入力できるように特別に設計された入力フィールドです。プライバシーとセキュリティを維持するために、入力した文字をマスクします。

- `{{ui.passwordID.value}}`: パスワード入力フィールドに指定された値を返します。
- `{{ui.passwordID.isValid}}`: パスワード入力フィールドに指定された値の有効性を返します。

検索

検索コンポーネントは、検索クエリを実行したり、アプリケーション内の入力データ内に検索語を入力したりするための専用の入力フィールドをユーザーに提供します。

- `{{ui.searchID.value}}`: 検索フィールドに指定された値を返します。

電話

電話コンポーネントは、ユーザーの電話番号やその他の連絡先情報をキャプチャするためにカスタマイズされた入力フィールドです。特定の検証ルールとフォーマットオプションを含めることで、入力した値が正しい電話番号形式に準拠していることを確認できます。

- `{{ui.phoneID.value}}`: 電話入力フィールドに指定された値を返します。
- `{{ui.phoneID.isValid}}`: 電話入力フィールドに指定された値の有効性を返します。

Number

数値コンポーネントは、ユーザーが数値を入力できるように特別に設計された入力フィールドです。検証ルールを適用して、入力された値が指定された範囲または形式の有効な数値であることを確認できます。

- `{{ui.numberID.value}}`: 数値入力フィールドに指定された値を返します。
- `{{ui.numberID.isValid}}`: 数値入力フィールドに指定された値の有効性を返します。

通貨

通貨コンポーネントは、金額または金額を取得するための特殊な入力フィールドです。これには、通貨記号、小数点区切り記号を表示し、通貨入力に固有の検証ルールを適用する書式設定オプションを含めることができます。

- `{{ui.currencyID.value}}`: 通貨入力フィールドに指定された値を返します。
- `{{ui.currencyID.isValid}}`: 通貨入力フィールドに指定された値の有効性を返します。

詳細ペア

詳細ペアコンポーネントは、キーと値のペアまたは関連情報のペアを構造化された読み取り可能な形式で表示するために使用されます。一般的に、特定の項目またはエンティティに関連付けられた詳細またはメタデータを表示するために使用されます。

選択コンポーネント

[Switch] (スイッチ)

Switch コンポーネントは、ユーザーがオン/オフ、true/false、有効/無効などの 2 つの状態またはオプションを切り替えることができるユーザーインターフェイスコントロールです。現在の状態を視覚的に表現し、ワンクリックまたはタップで変更できます。

グループを切り替える

スイッチグループコンポーネントは、ユーザーが事前定義されたセットから 1 つ以上のオプションを選択できるようにする個々のスイッチコントロールのコレクションです。選択したオプションと選択されていないオプションを視覚的に表示できるため、ユーザーは利用可能な選択肢を理解し、操作しやすくなります。

グループ式のフィールドを切り替える

- `{{ui.switchGroupID.value}}`: アプリケーションユーザーが有効にした各スイッチの値を含む文字列の配列を返します。

チェックボックスグループ

チェックボックスグループコンポーネントは、ユーザーにチェックボックスのグループを表示し、複数のオプションを同時に選択できるようにします。これは、オプションリストから 1 つ以上の項目を選択する機能をユーザーに提供する場合に便利です。

チェックボックスグループの式フィールド

- `{{ui.checkboxGroupID.value}}`: アプリケーションユーザーが選択した各チェックボックスの値を含む文字列の配列を返します。

無線グループ

無線グループコンポーネントは、ユーザーが複数の相互に排他的な選択肢から 1 つのオプションを選択できるようにする一連のラジオボタンです。これにより、一度に選択できるオプションは 1 つだけになり、ユーザーが選択できる明確であいまいな方法が提供されます。

無線グループ式フィールド

以下のフィールドは式で使用できます。

- `{{ui.radioGroupID.value}}`: アプリユーザーが選択したラジオボタンの値を返します。

単一選択

Single Select コンポーネントは、1 つの項目を選択できるオプションのリストをユーザーに示します。これは、カテゴリ、場所、設定の選択など、ユーザーが事前定義された一連のオプションから選択する必要があるシナリオでよく使用されます。

シングル選択式フィールド

- `{{ui.singleSelectID.value}}`: アプリユーザーが選択したリスト項目の値を返します。

マルチ選択

マルチ選択コンポーネントはシングル選択コンポーネントに似ていますが、ユーザーは選択肢のリストから複数のオプションを同時に選択できます。これは、ユーザーが複数のタグ、関心、設定の選択など、事前定義された一連のオプションから複数の選択を行う必要がある場合に便利です。

複数選択式のフィールド

- `{{ui.multiSelectID.value}}`: アプリケーションユーザーが選択した各リスト項目の値を含む文字列の配列を返します。

ボタンとナビゲーションコンポーネント

アプリケーションスタジオには、ユーザーがアクションをトリガーし、アプリケーション内を移動できるように、さまざまなボタンとナビゲーションコンポーネントが用意されています。

ボタンコンポーネント

使用可能なボタンコンポーネントは次のとおりです。

- ボタン
- アウトラインされたボタン
- アイコンボタン
- テキストボタン

これらのボタンコンポーネントは、次の一般的なプロパティを共有します。

コンテンツ

- ボタンラベル: ボタンに表示されるテキスト。

タイプ

- ボタン: 標準ボタン。
- アウトライン: アウトラインされたスタイルのボタン。
- アイコン: アイコンの付いたボタン。
- テキスト: テキストのみのボタン。

サイズ

ボタンのサイズ。指定できる値は Small、Medium、および Large です。

アイコン

ボタンに表示するさまざまなアイコンから選択できます。

- エンベロープが閉じられました
- Bell
- 個人
- ハンバーガーメニュー
- 検索
- 丸で囲まれた情報
- 歯車
- シェブロン左
- シェブロン右
- 水平ドット
- ごみ箱
- 編集
- チェック
- 閉じる
- ホーム
- Plus

トリガ

ボタンをクリックすると、トリガーされるアクションを 1 つ以上設定できます。使用可能なアクションタイプは次のとおりです。

- ベーシック
 - コンポーネントアクションの実行: コンポーネント内で特定のアクションを実行します。
 - 移動: 別のページまたはビューに移動します。

- データアクションの呼び出し: レコードの作成、更新、削除など、データ関連のアクションをトリガーします。
- アドバンスト
 - JavaScript: カスタム JavaScript コードを実行します。
 - 自動化を呼び出す: 既存の自動化またはワークフローを開始します。

JavaScript アクションボタンのプロパティ

ボタンをクリックしたときにカスタム JavaScript コードを実行する JavaScript アクションタイプを選択します。

ソースコード

Source code フィールドに、JavaScript 式または関数を入力できます。例えば、次のようになります。

```
return "Hello World";
```

これにより、ボタンがクリックされた Hello World ときに文字列が返されます。

条件: 次の場合に を実行します。

JavaScript アクションを実行するかどうかを決定するブール式を指定することもできます。次の構文を使用します。

```
{{ui.textinput1.value !== ""}}
```

この例では、JavaScript アクションは、textinput1 コンポーネントの値が空の文字列でない場合にのみ実行されます。

これらの高度なトリガーオプションを使用すると、アプリケーションのロジックやデータと直接統合する高度にカスタマイズされたボタン動作を作成できます。これにより、ボタンの組み込み機能を拡張し、特定の要件に合わせてユーザーエクスペリエンスを調整できます。

Note

JavaScript アクションが期待どおりに機能していることを常に徹底的にテストします。

ハイパーリンク

ハイパーリンクコンポーネントは、外部 URLs または内部アプリケーションルートに移動するためのクリック可能なリンクを提供します。

ハイパーリンクプロパティ

コンテンツ

- **ハイパーリンクラベル:** ハイパーリンクラベルとして表示されるテキスト。

[URL]

ハイパーリンクの送信先 URL。外部ウェブサイトまたは内部アプリケーションルートです。

トリガ

ハイパーリンクをクリックすると、トリガーされる 1 つ以上のアクションを設定できます。使用可能なアクションタイプは、ボタンコンポーネントのアクションタイプと同じです。

日付と時刻のコンポーネント

日付

Date コンポーネントを使用すると、ユーザーは日付を選択して入力できます。

Date コンポーネントは、、、などの他のコンポーネントといくつかの共通プロパティを共有NameSourceしますValidation。これらのプロパティの詳細については、「」を参照してください[一般的なコンポーネントプロパティ](#)。

Date コンポーネントには、共通のプロパティに加えて、次の特定のプロパティがあります。

日付プロパティ

形式

- YYYY/MM/DD、DD/MM/YYYY、YYYY/MM/DD、YYYY/DD/MM、MM/DD、DD/MM: 日付を表示する形式。

値

- YYYY-MM-DD: 日付値が内部に保存される形式。

最小日付

- YYYY-MM-DD: 選択できる最小日付。

Note

この値は、 の形式と一致する必要がありますYYYY-MM-DD。

最大日付

- YYYY-MM-DD: 選択できる最大日付。

Note

この値は、 の形式と一致する必要がありますYYYY-MM-DD。

カレンダータイプ

- 1 か月、2 か月: 表示するカレンダー UI のタイプ。

無効になった日付

- ソース: 無効にする必要がある日付のデータソース。例: None、式。
- 無効日付: 無効にする日付を決定する式。次に例を示します。
 - `{{currentRow.column}}`: この式が評価する日付を無効にします。
 - `{{new Date(currentRow.dateColumn) < new Date("2023-01-01")}}`: 2023 年 1 月 1 日より前の日付を無効にします
 - `{{new Date(currentRow.dateColumn).getDay() === 0 || new Date(currentRow.dateColumn).getDay() === 6}}`: 週末を無効にします。

行動

- 表示可能: Date コンポーネントの可視性を決定する式。
- Disable if: Date コンポーネントを無効にするかどうかを決定する式。

検証

検証セクションでは、日付入力に追加のルールと制約を定義できます。これらの検証ルールを設定することで、ユーザーが入力した日付値がアプリケーションの特定の要件を満たしていることを確認できます。次のタイプの検証を追加できます。

- 必須: このトグルにより、ユーザーはフォームを送信する前に日付値を入力する必要があります。
- カスタム: JavaScript 式を使用してカスタム検証ルールを作成できます。例えば、次のようになります。

```
{{new Date(ui.dateInput.value) < new Date("2023-01-01")}}
```

この式は、入力された日付が 2023 年 1 月 1 日より前であるかどうかをチェックします。条件が true の場合、検証は失敗します。

検証が満たされない場合に表示されるカスタム検証メッセージを指定することもできます。

```
"Validation not met. The date must be on or after January 1, 2023."
```

これらの検証ルールを設定することで、ユーザーが入力した日付値がアプリケーションの特定の要件を満たしていることを確認できます。

式と例

Date コンポーネントには、次の式フィールドが用意されています。

- `{{ui.dateID.value}}`: ユーザーが入力した日付値を形式で返しますYYYY-MM-DD。

Time

Time コンポーネントを使用すると、ユーザーは時間値を選択して入力できます。Time コンポーネントのさまざまなプロパティを設定することで、選択可能な時間範囲の制限、特定の時間の無効化、コンポーネントの可視性とインタラクティブ性の制御など、アプリケーションの特定の要件を満たす時間入力フィールドを作成できます。

時間プロパティ

Time コンポーネントは、、、などの他のコンポーネントといくつかの共通プロパティを共有NameSourceしますValidation。これらのプロパティの詳細については、「」を参照してください[一般的なコンポーネントプロパティ](#)。

共通プロパティに加えて、Time コンポーネントには次の特定のプロパティがあります。

時間間隔

- 5 分、10 分、15 分、20 分、25 分、30 分、60 分: 時間の選択に使用できる間隔。

値

- HH:MM AA: 時間値が内部に保存される形式。

Note

この値は、 の形式と一致する必要がありますHH:MM AA。

Placeholder

- カレンダー設定: 時間フィールドが空の場合に表示されるプレースホルダーテキスト。

最小時間

- HH:MM AA: 選択できる最小時間。

Note

この値は、 の形式と一致する必要がありますHH:MM AA。

最大時間

- HH:MM AA: 選択できる最大時間。

Note

この値は、 の形式と一致する必要がありますHH:MM AA。

無効時間

- ソース: 無効にする必要がある時間のデータソース (例: なし、式)。
- 無効時間: など、無効にする時間を決定する式`{{currentRow.column}}`。

無効時間設定

Disabled Times セクションを使用して、選択できない時間値を指定できます。

ソース

- なし: 時間が無効になることはありません。
- 式: JavaScript 式を使用して、 など、無効にする時間を決定できます`{{currentRow.column}}`。

式の例:

```
{{currentRow.column === "Lunch Break"}}
```

この式は、現在の行の「昼食休憩」列が true のときはいつでも無効にします。

これらの検証ルールと無効な時間式を設定することで、ユーザーが入力した時間値がアプリケーションの特定の要件を満たしていることを確認できます。

行動

- 表示可能: Time コンポーネントの可視性を決定する式。
- Disable if: Time コンポーネントを無効にするかどうかを決定する式。

検証

- 必須: フォームを送信する前にユーザーが時間値を入力するようにするトグル。

- カスタム: JavaScript 式を使用してカスタム検証ルールを作成できます。

カスタム検証メッセージ: カスタム検証が満たされない場合に表示されるメッセージ。

例えば、次のようになります。

```
{{ui.timeInput.value === "09:00 AM" || ui.timeInput.value === "09:30 AM"}}
```

この式は、入力された時刻が午前 9 時または午前 9 時 30 分であるかどうかをチェックします。条件が true の場合、検証は失敗します。

検証が満たされない場合に表示されるカスタム検証メッセージを指定することもできます。

```
Validation not met. The time must be 9:00 AM or 9:30 AM.
```

式と例

Time コンポーネントには、次の式フィールドが用意されています。

- `{{ui.timeID.value}}`: ユーザーが HH:MM AA 形式で入力した時間値を返します。

例: 時間値

- `{{ui.timeID.value}}`: ユーザーが 形式で入力した時間値を返します HH:MM AA。

例: 時間比較

- `{{ui.timeInput.value > "10:00 AM"}}`: 時刻値が午前 10:00 より大きいかどうかを確認します。
- `{{ui.timeInput.value < "05:00 PM"}}`: 時刻値が午後 5 時未満かどうかを確認します。

日付範囲

日付範囲コンポーネントを使用すると、ユーザーは日付の範囲を選択して入力できます。日付範囲コンポーネントのさまざまなプロパティを設定することで、選択可能な日付範囲の制限、特定の日付の無効化、コンポーネントの可視性とインタラクティブ性の制御など、アプリケーションの特定の要件を満たす日付範囲入力フィールドを作成できます。

日付範囲プロパティ

日付範囲コンポーネントは、、、などの他のコンポーネントといくつかの共通プロパティを共有NameSourceしますValidation。これらのプロパティの詳細については、「」を参照してください[一般的なコンポーネントプロパティ](#)。

共通のプロパティに加えて、日付範囲コンポーネントには次の特定のプロパティがあります。

形式

- MM/DD/YYYY: 日付範囲を表示する形式。

[開始日]

- YYYY-MM-DD: 範囲の開始として選択できる最小日付。

Note

この値は の形式と一致する必要がありますYYYY-MM-DD。

[終了日]

- YYYY-MM-DD: 範囲の最後に選択できる最大日付。

Note

この値は の形式と一致する必要がありますYYYY-MM-DD。

Placeholder

- カレンダー設定: 日付範囲フィールドが空の場合に表示されるプレースホルダーテキスト。

最小日付

- YYYY-MM-DD: 選択できる最小日付。

Note

この値は の形式と一致する必要がありますYYYY-MM-DD。

最大日付

- YYYY-MM-DD: 選択できる最大日付。

Note

この値は の形式と一致する必要がありますYYYY-MM-DD。

カレンダータイプ

- 1 か月: 表示するカレンダー UI のタイプ。たとえば、1 か月です。
- 2 か月: 表示するカレンダー UI のタイプ。たとえば、2 か月です。

選択した必須日数

- 0: 日付範囲内で選択する必要がある必須日数。

無効になった日付

- ソース: 無効にする必要がある日付のデータソース (None、Expression、Entity、Automation など)。
- 無効日付: など、無効にする日付を決定する式`{{currentRow.column}}`。

検証

検証セクションでは、日付範囲入力に追加のルールと制約を定義できます。

式と例

日付範囲コンポーネントには、次の式フィールドが用意されています。

- `{{ui.dateRangeID.startDate}}`: 選択した範囲の開始日を 形式で返しますYYYY-MM-DD。

- `{{ui.dateRangeID.endDate}}`: 選択した範囲の終了日を形式で返しますYYYY-MM-DD。

例: 日付の差の計算

- `{{(new Date(ui.dateRangeID.endDate) - new Date(ui.dateRangeID.startDate)) / (1000 * 60 * 60 * 24)}}` 開始日から終了日までの日数を計算します。

例: 日付範囲に基づく条件付き可視性

- `{{new Date(ui.dateRangeID.startDate) < new Date("2023-01-01") || new Date(ui.dateRangeID.endDate) > new Date("2023-12-31")}}` 選択した日付範囲が2023年の外側にあるかどうかを確認します。

例: 現在の行データに基づいて日付を無効にする

- `{{currentRow.isHoliday}}` 現在の行の「isHoliday」列が true である日付を無効にします。
- `{{new Date(currentRow.dateColumn) < new Date("2023-01-01")}}` 現在の行の「dateColumn」に基づいて、2023年1月1日より前の日付を無効にします。
- `{{new Date(currentRow.dateColumn).getDay() === 0 || new Date(currentRow.dateColumn).getDay() === 6}}` 現在の行の「dateColumn」に基づいて週末を無効にします。

カスタム検証

- `{{new Date(ui.dateRangeID.startDate) > new Date(ui.dateRangeID.endDate)}}` 開始日が終了日より後であるかどうかをチェックし、カスタム検証が失敗します。

メディアコンポーネント

アプリケーションスタジオには、アプリケーション内にさまざまなメディアタイプを埋め込んで表示するためのコンポーネントがいくつか用意されています。

iFrame 埋め込み

iFrame 埋め込みコンポーネントを使用すると、iFrame を使用してアプリケーション内に外部ウェブコンテンツまたはアプリケーションを埋め込むことができます。

iFrame 埋め込みプロパティ

[URL]

Note

このコンポーネントに表示されるメディアのソースは、アプリケーションのコンテンツセキュリティ設定で許可する必要があります。詳細については、「[アプリのコンテンツセキュリティ設定の表示または更新](#)」を参照してください。

埋め込む外部コンテンツまたはアプリケーションの URL。

[レイアウト]

- 幅: パーcentage (%) または固定ピクセル値 (300px など) で指定された iFrame の幅。
- 高さ: パーcentage (%) または固定ピクセル値で指定された iFrame の高さ。

S3 アップロード

S3 アップロードコンポーネントを使用すると、ユーザーは Amazon S3 バケットにファイルをアップロードできます。S3 アップロードコンポーネントを設定することで、ユーザーはアプリケーションの Amazon S3 ストレージにファイルを簡単にアップロードし、アップロードされたファイル情報をアプリケーションのロジックとユーザーインターフェイス内で活用できます。

Note

アプリケーションのファイルアップロードとストレージ要件をサポートするために、必要なアクセス許可と Amazon S3 バケット設定が設定されていることを確認してください。

S3 アップロードプロパティ

S3 設定

- コネクタ: ファイルのアップロードに使用する事前設定済みの Amazon S3 コネクタを選択します。
- バケット: ファイルをアップロードする Amazon S3 バケット。

- フォルダ: ファイルを保存する Amazon S3 バケット内のフォルダ。
- ファイル名: アップロードされたファイルの命名規則。

ファイルのアップロード設定

- ラベル: ファイルのアップロード領域の上に表示されるラベルまたは手順。
- 説明: ファイルのアップロードに関する追加の手順または情報。
- ファイルタイプ: アップロードできるファイルのタイプ。例: イメージ、ドキュメント、ビデオ。
- サイズ: アップロードできる個々のファイルの最大サイズ。
- ボタンラベル: ファイル選択ボタンに表示されるテキスト。
- ボタンスタイル: ファイル選択ボタンのスタイル。例えば、概要や入力などです。
- ボタンサイズ: ファイル選択ボタンのサイズ。

検証

- ファイルの最大数: 一度にアップロードできるファイルの最大数。
- 最大ファイルサイズ: 個々のファイルで許可される最大サイズ。

トリガ

- 成功時: ファイルのアップロードが成功したときにトリガーされるアクション。
- 失敗時: ファイルのアップロードが失敗した場合にトリガーされるアクション。

S3 アップロード式フィールド

S3 アップロードコンポーネントには、次の式フィールドが用意されています。

- `{{ui.s3uploadID.files}}`: アップロードされたファイルの配列を返します。
- `{{ui.s3uploadID.files[0]?.size}}`: 指定されたインデックスのファイルのサイズを返します。
- `{{ui.s3uploadID.files[0]?.type}}`: 指定されたインデックスのファイルのタイプを返します。
- `{{ui.s3uploadID.files[0]?.nameOnly}}`: 指定されたインデックスで拡張子のサフィックスなしでファイルの名前を返します。

- `{{ui.s3uploadID.files[0]?.nameWithExtension}}`: 指定されたインデックスに拡張子のサフィックスが付いたファイルの名前を返します。

式と例

例: アップロードされたファイルへのアクセス

- `{{ui.s3uploadID.files.length}}`: アップロードされたファイルの数を返します。
- `{{ui.s3uploadID.files.map(f => f.name).join(', ')}}`: アップロードされたファイル名のカンマ区切りリストを返します。
- `{{ui.s3uploadID.files.filter(f => f.type.startsWith('image/'))}}`: アップロードされたイメージファイルのみの配列を返します。

例: ファイルのアップロードの検証

- `{{ui.s3uploadID.files.some(f => f.size > 5 * 1024 * 1024)}}`: アップロードされたファイルのサイズが 5 MB を超えているかどうかを確認します。
- `{{ui.s3uploadID.files.every(f => f.type === 'image/png')}}`: アップロードされたすべてのファイルが PNG イメージかどうかを確認します。
- `{{ui.s3uploadID.files.length > 3}}`: 3 つ以上のファイルがアップロードされたかどうかを確認します。

例: アクションのトリガー

- `{{ui.s3uploadID.files.length > 0 ? 'Upload Successful' : 'No files uploaded'}}`: 少なくとも 1 つのファイルがアップロードされると、成功メッセージが表示されます。
- `{{ui.s3uploadID.files.some(f => f.type.startsWith('video/')) ? triggerVideoProcessing() : null}}`: 動画ファイルがアップロードされると、動画処理の自動化をトリガーします。
- `{{ui.s3uploadID.files.map(f => f.url)}}`: アップロードされたファイルの URLs を取得します。この URL は、ファイルの表示や処理に使用できます。

これらの式を使用すると、アップロードされたファイルにアクセスし、ファイルのアップロードを検証し、ファイルのアップロード結果に基づいてアクションをトリガーできます。これらの式を使用す

ることで、アプリケーションのファイルアップロード機能内でより動的でインテリジェントな動作を作成できます。

Note

`s3uploadID` を S3 アップロードコンポーネントの ID に置き換えます。

PDF ビューワーコンポーネント

PDF ビューワーコンポーネントを使用すると、ユーザーはアプリケーション内の PDF ドキュメントを表示して操作できます。App Studio は PDF ソースに対してこれらのさまざまな入力タイプをサポートしています。PDF ビューワーコンポーネントは、静的 URL、インラインデータ URI、動的に生成されたコンテンツのいずれからでも、PDF ドキュメントをアプリケーションに統合する方法に柔軟性を提供します。

PDF ビューワーのプロパティ

ソース

Note

このコンポーネントに表示されるメディアのソースは、アプリケーションのコンテンツセキュリティ設定で許可する必要があります。詳細については、「[アプリのコンテンツセキュリティ設定の表示または更新](#)」を参照してください。

PDF ドキュメントのソース。式、エンティティ、URL、またはオートメーションです。

式

式を使用して PDF ソースを動的に生成します。

エンティティ

PDF ビューワーコンポーネントを PDF ドキュメントを含むデータエンティティに接続します。

[URL]

PDF ドキュメントの URL を指定します。

[URL]

表示する PDF ドキュメントを指す URL を入力できます。これは、パブリックウェブ URL でも、独自のアプリケーション内の URL でもかまいません。

例: `https://example.com/document.pdf`

データ URI

データ URI は、アプリケーション内に小さなデータファイル (画像や PDFs) をインラインで含めるコンパクトな方法です。PDF ドキュメントは base64 文字列としてエンコードされ、コンポーネントの設定に直接含まれます。

Blob または ArrayBuffer

PDF ドキュメントを Blob または ArrayBuffer オブジェクトとして提供することもできます。これにより、アプリケーション内のさまざまなソースから PDF データを動的に生成または取得できます。

オートメーション

PDF ビューワーコンポーネントを PDF ドキュメントを提供するオートメーションに接続します。

アクション

- **ダウンロード:** ユーザーが PDF ドキュメントをダウンロードできるようにするボタンまたはリンクを追加します。

[レイアウト]

- **幅:** PDF ビューワーの幅。パーセンテージ (%) または固定ピクセル値 (600px など) で指定します。
- **高さ:** 固定ピクセル値として指定された PDF ビューワーの高さ。

イメージビューワー

イメージビューワーコンポーネントを使用すると、ユーザーはアプリケーション内のイメージファイルを表示して操作できます。

イメージビューワーのプロパティ

ソース

Note

このコンポーネントに表示されるメディアのソースは、アプリケーションのコンテンツセキュリティ設定で許可する必要があります。詳細については、「[アプリのコンテンツセキュリティ設定の表示または更新](#)」を参照してください。

- Entity: Image Viewer コンポーネントを、イメージファイルを含むデータエンティティに接続します。
- URL: イメージファイルの URL を指定します。
- 式: 式を使用してイメージソースを動的に生成します。
- 自動化: イメージビューワーコンポーネントを、イメージファイルを提供する自動化に接続します。

Alt テキスト

アクセシビリティの目的で使用されるイメージの代替テキストの説明。

[レイアウト]

- イメージフィット: イメージのサイズを変更してコンポーネント内に表示する方法を決定します。たとえば、Contain、Cover、または Fill などです。
- 幅: パーセンテージ (%) または固定ピクセル値 (300px など) で指定されたイメージビューワーコンポーネントの幅。
- 高さ: 固定ピクセル値として指定されたイメージビューワーコンポーネントの高さ。
- 背景: イメージビューワーコンポーネントの背景色またはイメージを設定できます。

自動化とアクション: アプリのビジネスロジックを定義する

自動化は、アプリケーションのビジネスロジックを定義する方法です。オートメーションの主なコンポーネントは、オートメーションを開始するトリガー、1 つ以上のアクションのシーケンス、オートメーションにデータを渡すために使用される入力パラメータ、および出力です。

トピック

- [オートメーションの概念](#)
- [オートメーションの作成、編集、削除](#)
- [自動化アクションの追加、編集、削除](#)
- [オートメーションアクションのリファレンス](#)

オートメーションの概念

App Studio のオートメーションを使用してアプリのビジネスロジックを定義および設定するとき、知っておくべき概念と用語をいくつか紹介します。

自動化

自動化は、アプリケーションのビジネスロジックを定義する方法です。オートメーションの主なコンポーネントは、オートメーションを開始するトリガー、1つ以上のアクションのシーケンス、オートメーションにデータを渡すために使用される入力パラメータ、および出力です。

アクション

一般的にアクションと呼ばれるオートメーションアクションは、オートメーションを構成するロジックの個々のステップです。各アクションは、Eメールの送信、データレコードの作成、Lambda 関数の呼び出し、APIs呼び出しなど、特定のタスクを実行します。アクションはアクションライブラリのオートメーションに追加され、条件ステートメントまたはループにグループ化できます。

自動化入力パラメータ

自動化入力パラメータは、コンポーネントから自動化に渡すことができる動的入力値であり、柔軟で再利用可能なものにします。パラメータをオートメーションの変数と考えると、値をオートメーションにハードコーディングする代わりに、パラメータを定義し、必要に応じて異なる値を指定できます。パラメータを使用すると、実行されるたびに異なる入力で同じオートメーションを使用できます。

モック出力

一部のアクションは、コネクタを使用して外部リソースまたはサービスとやり取りします。プレビュー環境を使用する場合、アプリケーションは外部サービスとやり取りしません。プレビュー環境でコネクタを使用するアクションをテストするには、モック出力を使用してコネクタの動作と出力をシミュレートできます。モック出力は JavaScript を使用して設定され、コネクタのレスポンスが公開されたアプリケーションに保存されているのと同様に、結果はアクションの結果に保存されます。

モックを使用すると、プレビュー環境を使用して、コネクタを介して外部サービスを呼び出すことなく、さまざまな結果値、エラーシナリオ、エッジケース、不満足なパスのシミュレーションなど、自動化を使用してさまざまなシナリオと他のアクションへの影響をテストできます。

自動化出力

オートメーション出力は、あるオートメーションから、コンポーネントやその他のオートメーションなど、アプリケーションの他のリソースに値を渡すために使用されます。自動化出力は式として設定され、式は自動化パラメータとアクションから計算された静的値または動的値を返すことができます。デフォルトでは、オートメーションは、オートメーション内のアクションの結果を含むデータを返しません。

自動化出力の使用方法の例をいくつか示します。

- オートメーション出力を設定して配列を返し、その配列を渡してデータコンポーネントを入力できます。
- オートメーションを使用して値を計算し、ビジネスロジックを一元化して再利用する方法として、その値を他の複数のオートメーションに渡すことができます。

トリガ

トリガーは、オートメーションをいつ、どの条件で実行するかを決定します。トリガーの例としては、ボタン0n click用とテキスト入力0n select用があります。コンポーネントのタイプによって、そのコンポーネントの使用可能なトリガーのリストが決まります。トリガーは[コンポーネント](#)に追加され、アプリケーションスタジオで設定されます。

オートメーションの作成、編集、削除

目次

- [オートメーションの作成](#)
- [自動化プロパティの表示または編集](#)
- [オートメーションの削除](#)

オートメーションの作成

App Studio アプリケーションでオートメーションを作成するには、次の手順に従います。作成したオートメーションは、プロパティを編集してアクションを追加して設定する必要があります。

オートメーションを作成するには

1. 必要に応じて、アプリケーションのアプリケーションスタジオに移動します。
2. [自動化] タブを選択します。
3. 自動化がない場合は、キャンバスで + 自動化を追加する を選択します。それ以外の場合は、左側のオートメーションメニューで + 追加を選択します。
4. 新しいオートメーションが作成され、そのプロパティの編集を開始するか、アクションを追加および設定してアプリケーションのビジネスロジックを定義できます。

自動化プロパティの表示または編集

オートメーションプロパティを表示または編集するには、次の手順に従います。

オートメーションプロパティを表示または編集するには

1. 必要に応じて、アプリケーションのアプリケーションスタジオに移動します。
2. [自動化] タブを選択します。
3. 左側のオートメーションメニューで、プロパティを表示または編集するオートメーションを選択して、右側のプロパティメニューを開きます。
4. プロパティメニューでは、次のプロパティを表示できます。
 - 自動化識別子: 自動化の一意の名前。編集するには、テキストフィールドに新しい識別子を入力します。
 - 自動化パラメータ: 自動化パラメータは、アプリケーションの UI から自動化アクションとデータアクションに動的値を渡すために使用されます。パラメータを追加するには、+ 追加を選択します。鉛筆アイコンを選択して、パラメータの名前、説明、またはタイプを変更します。パラメータを削除するには、ごみ箱アイコンを選択します。

Tip

キャンバスから直接オートメーションパラメータを追加することもできます。

- オートメーション出力: オートメーション出力は、オートメーションのどのデータを他のオートメーションまたはコンポーネントで参照できるかを設定するために使用されます。デフォルトでは、オートメーションは出力を作成しません。自動化出力を追加するには、+ 追加を選択します。出力を削除するには、ごみ箱アイコンを選択します。

5. オートメーションの動作を定義するには、アクションを追加および設定します。アクションの詳細については、[自動化アクションの追加、編集、削除](#)「」および「」を参照してください。[オートメーションアクションのリファレンス](#)。

オートメーションの削除

App Studio アプリケーションでオートメーションを削除するには、次の手順に従います。

オートメーションを削除するには

1. 必要に応じて、アプリケーションのアプリケーションスタジオに移動します。
2. [自動化] タブを選択します。
3. 左側のオートメーションメニューで、削除するオートメーションの省略形メニューを選択し、削除を選択します。または、自動化の右側にあるプロパティメニューからごみ箱アイコンを選択することもできます。
4. 確認ダイアログボックスで、[削除] を選択します。

自動化アクションの追加、編集、削除

一般的にアクションと呼ばれるオートメーションアクションは、オートメーションを構成するロジックの個々のステップです。各アクションは、Eメールの送信、データレコードの作成、Lambda 関数の呼び出し、APIs呼び出しなど、特定のタスクを実行します。アクションはアクションライブラリのオートメーションに追加され、条件ステートメントまたはループにグループ化できます。

目次

- [オートメーションアクションの追加](#)
- [オートメーションアクションのプロパティの表示と編集](#)
- [オートメーションアクションの削除](#)

オートメーションアクションの追加

App Studio アプリケーションのオートメーションにアクションを追加するには、次の手順に従います。

オートメーションアクションを追加するには

1. 必要に応じて、アプリケーションのアプリケーションスタジオに移動します。

2. [自動化] タブを選択します。
3. 左側のオートメーションメニューで、アクションを追加するオートメーションを選択します。
4. 右側のアクションメニューで、追加するアクションを選択するか、アクションをキャンバスにドラッグアンドドロップします。アクションを作成したら、アクションを選択してアクションのプロパティを設定して、アクションの機能を定義できます。アクションプロパティとその設定の詳細については、「」を参照してください[オートメーションアクションのリファレンス](#)。

オートメーションアクションのプロパティの表示と編集

App Studio アプリケーションでオートメーションアクションのプロパティを表示または編集するには、次の手順に従います。

オートメーションアクションのプロパティを表示または編集するには

1. 必要に応じて、アプリケーションのアプリケーションスタジオに移動します。
2. [自動化] タブを選択します。
3. 左側のオートメーションメニューで、プロパティを表示または編集するアクションを選択します。または、キャンバスに含まれるオートメーションを表示するときに、キャンバス内のアクションを選択することもできます。
4. 右側のプロパティメニューでアクションプロパティを表示または編集できます。アクションのプロパティは、アクションタイプごとに異なります。アクションプロパティとその設定の詳細については、「」を参照してください[オートメーションアクションのリファレンス](#)。

オートメーションアクションの削除

App Studio アプリケーションのオートメーションからアクションを削除するには、次の手順に従います。

オートメーションアクションを削除するには

1. 必要に応じて、アプリケーションのアプリケーションスタジオに移動します。
2. [自動化] タブを選択します。
3. 左側のオートメーションメニューで、削除するアクションを含むオートメーションを選択します。
4. キャンバスで、削除するアクションのごみ箱アイコンを選択し、削除を選択します。

オートメーションアクションのリファレンス

App Studio で使用されるオートメーションアクションのリファレンスドキュメントを次に示します。

一般的にアクションと呼ばれるオートメーションアクションは、オートメーションを構成するロジックの個々のステップです。各アクションは、Eメールの送信、データレコードの作成、Lambda 関数の呼び出し、APIs呼び出しなど、特定のタスクを実行します。アクションはアクションライブラリのオートメーションに追加され、条件ステートメントまたはループにグループ化できます。

オートメーションとそのアクションの作成と設定については、「」のトピックを参照してください [自動化とアクション: アプリのビジネスロジックを定義する](#)。

API を呼び出す

HTTP REST API リクエストを呼び出します。ビルダーはこのアクションを使用して、APIs を使用して App Studio から他のシステムまたはサービスにリクエストを送信できます。例えば、サードパーティーのシステムや自社開発のアプリケーションに接続してビジネスクリティカルなデータにアクセスしたり、専用の App Studio アクションでは呼び出せない API エンドポイントを呼び出すことができます。

REST APIs [RESTful API とは](#)」を参照してください。

プロパティ

コネクタ

このアクションによって行われた API リクエストに使用するコネクタ。コネクタドロップダウンには、API Connector および タイプのコネクタのみが含まれます OpenAPI Connector。コネクタの設定方法に応じて、認証情報やデフォルトのヘッダー、クエリパラメータなどの重要な情報を含めることができます。

API Connector と の使用の比較など、API コネクタの詳細については OpenAPI Connector、「」を参照してください [サードパーティーサービスに接続する](#)。

API リクエスト設定プロパティ

プロパティパネルから API リクエストの設定 を選択し、リクエスト設定ダイアログボックスを開きます。API コネクタが選択されている場合、ダイアログボックスにはコネクタ情報が含まれます。

メソッド: API コールのメソッド。可能な値は以下のとおりです。

- DELETE: 指定されたリソースを削除します。
- GET: 情報またはデータを取得します。
- HEAD: 本文のないレスポンスのヘッダーのみを取得します。
- POST: 処理するデータを送信します。
- PUSH: 処理するデータを送信します。
- PATCH: 指定されたリソースを部分的に更新します。

パス: リソースへの相対パス。

ヘッダー: API リクエストで送信されるキーと値のペアの形式のヘッダー。コネクタが選択されている場合、設定されたヘッダーは自動的に追加され、削除することはできません。設定されたヘッダーは編集できませんが、同じ名前の別のヘッダーを追加して上書きできます。

クエリパラメータ: API リクエストで送信されるキーと値のペアの形式のクエリパラメータ。コネクタが選択されている場合、設定されたクエリパラメータは自動的に追加され、編集または削除することはできません。

本文: API リクエストとともに JSON 形式で送信される情報。GET リクエストの本文はありません。

モック出力

アクションは、プレビュー環境の外部サービスまたはリソースとやり取りしません。モック出力フィールドは、テスト目的でプレビュー環境のコネクタの動作をシミュレートする JSON 式を提供するために使用されます。このスニペットは、ライブ環境で公開されたアプリケーションに対するコネクタレスポンスと同様に、アクションの `results` マップに保存されます。

このフィールドを使用すると、コネクタを介して外部サービスと通信することなく、さまざまな結果値、エラーシナリオ、エッジケース、不満のあるパスのシミュレートなど、オートメーション内の他のアクションに対するさまざまなシナリオとその影響をテストできます。

呼び出し AWS

AWS サービスから オペレーションを呼び出します。これはサービス AWS またはオペレーションを呼び出すための一般的なアクションであり、目的の AWS サービスまたはオペレーション専用のアクションがない場合に使用する必要があります。

プロパティ

サービス

実行するオペレーションを含む AWS サービス。

運用

実行するオペレーション。

コネクタ

このアクションによって実行されるオペレーションに使用されるコネクタ。設定済みコネクタは、オペレーションを実行するための適切な認証情報と、オペレーションで参照されるリソースを含む AWS リージョンなどのその他の設定情報を使用して設定する必要があります。

設定

指定されたオペレーションを実行するときの JSON 入力。AWS オペレーションの入力の設定の詳細については、「」を参照してください [AWS SDK for JavaScript](#)。

Lambda を呼び出す

既存の Lambda 関数を呼び出します。

プロパティ

コネクタ

このアクションによって実行される Lambda 関数に使用されるコネクタ。設定済みコネクタは、Lambda 関数にアクセスするための適切な認証情報と、Lambda 関数を含む AWS リージョンなどのその他の設定情報を使用して設定する必要があります。Lambda のコネクタの設定の詳細については、「」を参照してください [ステップ 3: Lambda コネクタを作成する](#)。

関数名

実行する Lambda 関数の名前。これは関数名であり、関数 ARN (Amazon リソースネーム) ではないことに注意してください。

関数イベント

イベントペイロードとして Lambda 関数に渡されるキーと値のペア。

モック出力

アクションは、プレビュー環境の外部サービスまたはリソースとやり取りしません。モック出力フィールドは、テスト目的でプレビュー環境のコネクタの動作をシミュレートする JSON 式を提供するために使用されます。このスニペットは、ライブ環境で公開されたアプリケーションに対するコネクタレスポンスと同様に、アクションの `results` マップに保存されます。

このフィールドを使用すると、コネクタを介して外部サービスと通信することなく、さまざまな結果値、エラーシナリオ、エッジケース、不満のあるパスのシミュレートなど、オートメーション内の他のアクションに対するさまざまなシナリオとその影響をテストできます。

ループ

ネストされたアクションを繰り返し実行して、項目のリストを一度に 1 つずつ反復処理します。たとえば、[レコードの作成](#) アクションをループアクションに追加して、複数のレコードを作成します。

ループアクションは、他のループまたは条件アクション内にネストできます。ループアクションは、並列ではなく順番に実行されます。ループ内の各アクションの結果は、同じループ反復内の後続のアクションにのみアクセスできます。ループの外部またはループの異なる反復でアクセスすることはできません。

プロパティ

ソース

反復処理する項目のリスト。一度に 1 つの項目。ソースは、前のアクションの結果でも、JavaScript 式を使用して指定できる文字列、数値、またはオブジェクトの静的リストの結果でもかまいません。

例

次のリストには、ソース入力の例が含まれています。

- 前のアクションの結果: `{{results.actionName.data}}`
- 数値のリスト: `{{[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]}}`
- 文字列のリスト: `{{["apple", "banana", "orange", "grape", "kiwi"]}}`
- 計算された値: `{{params.actionName.split("\n")}}`

現在の項目名

反復される現在の項目を参照するために使用できる変数の名前。現在の項目名は、2 つ以上のループをネストし、各ループから変数にアクセスできるように設定できます。たとえば、2 つのループを持つ国と都市をループする場合は、`currentCountry` を設定して参照できます `currentCity`。

Condition

オートメーションの実行時に評価される 1 つ以上の指定された論理条件の結果に基づいてアクションを実行します。条件アクションは、次のコンポーネントで構成されます。

- `true` または `false` と評価される JavaScript 式を提供するために使用される条件フィールド `false`。
- 条件が `true` に評価された場合に実行されるアクションを含む `true` ブランチ `true`。
- 条件が `false` に評価された場合に実行されるアクションを含む `false` ブランチ `false`。

条件アクションにドラッグして、アクションを `true` ブランチと `false` ブランチに追加します。

プロパティ

Condition

アクションの実行時に評価される JavaScript 式。

レコードの作成

既存の App Studio エンティティに 1 つのレコードを作成します。

プロパティ

エンティティ

レコードを作成するエンティティ。エンティティを選択したら、レコードを作成するエンティティのフィールドに値を追加する必要があります。フィールドのタイプ、およびフィールドが必須かオプションかは、エンティティで定義されます。

レコードの更新

App Studio エンティティの既存のレコードを更新します。

プロパティ

エンティティ

更新するレコードを含むエンティティ。

条件

アクションによって更新されるレコードを定義する基準。条件をグループ化して、1つの論理ステートメントを作成できます。グループまたは条件を AND または OR ステートメントと組み合わせることができます。

フィールド

条件によって指定されたレコードで更新されるフィールド。

値

指定されたフィールドで更新する値。

レコードの削除

App Studio エンティティからレコードを削除します。

プロパティ

エンティティ

削除するレコードを含むエンティティ。

条件

アクションによって削除するレコードを定義する基準。条件をグループ化して、1つのロジックステートメントを作成できます。グループまたは条件を AND または OR ステートメントと組み合わせることができます。

データアクションを呼び出す

オプションのパラメータを使用してデータアクションを実行します。

プロパティ

データアクション

アクションによって実行されるデータアクション。

パラメータ

データアクションで使用されるデータアクションパラメータ。データアクションパラメータは、データアクションの入力として使用される値を送信するために使用されます。データアクションパラメータはオートメーションアクションを設定するときに追加できますが、データタブで編集する必要があります。

詳細設定

Invoke data action アクションには、次の詳細設定が含まれます。

- ページサイズ: 各クエリで取得するレコードの最大数。デフォルト値は 500 で、最大値は 3000 です。
- ページ分割トークン: クエリから追加のレコードを取得するために使用されるトークン。たとえば、Page sizeが 500 に設定されている場合、500 を超えるレコードがある場合、ページ分割トークンを後続のクエリに渡すと、次の 500 が取得されます。これ以上レコードやページが存在しない場合、トークンは未定義になります。

Amazon S3: オブジェクトを配置する

Amazon S3 PutObject オペレーションを使用して、キー (ファイルパス) によって識別されるオブジェクトを指定された Amazon S3 バケットに追加します。

プロパティ

コネクタ

このアクションによって実行されるオペレーションに使用されるコネクタ。設定済みコネクタは、オペレーションを実行するための適切な認証情報と、オペレーションで参照されるリソースを含む AWS リージョンなどのその他の設定情報を使用して設定する必要があります。

設定

PutObject コマンドで使用する必須オプション。オプションは以下のとおりです:

Note

Amazon S3 PutObject オペレーションの詳細については、Amazon Simple Storage Service API リファレンスの[PutObject](#)を参照してください。

- **バケット:** オブジェクトを配置する Amazon S3 バケットの名前。
- **キー:** Amazon S3 バケットに入れるオブジェクトの一意的名前。
- **本文:** Amazon S3 バケットに入れるオブジェクトの内容。

モック出力

アクションは、プレビュー環境の外部サービスまたはリソースとやり取りしません。モック出力フィールドは、テスト目的でプレビュー環境のコネクタの動作をシミュレートする JSON 式を提供するために使用されます。このスニペットは、ライブ環境で公開されたアプリのコネクタレスポンスと同様に、アクションの `results` マップに保存されます。

このフィールドを使用すると、コネクタを介して外部サービスと通信することなく、さまざまな結果値、エラーシナリオ、エッジケース、不満のあるパスのシミュレートなど、オートメーション内の他のアクションに対するさまざまなシナリオとその影響をテストできます。

Amazon S3: オブジェクトを削除する

Amazon S3 `DeleteObject` オペレーションを使用して、指定された Amazon S3 バケットからキー (ファイルパス) によって識別されるオブジェクトを削除します。

プロパティ

コネクタ

このアクションによって実行されるオペレーションに使用されるコネクタ。設定済みコネクタは、オペレーションを実行するための適切な認証情報と、オペレーションで参照されるリソースを含む AWS リージョンなどのその他の設定情報を使用して設定する必要があります。

設定

`DeleteObject` コマンドで使用する必須オプション。オプションは以下のとおりです:

Note

Amazon S3 `DeleteObject` オペレーションの詳細については、Amazon Simple Storage Service API リファレンスの [DeleteObject](#) を参照してください。

- **バケット:** オブジェクトを削除する Amazon S3 バケットの名前。

- キー: Amazon S3 バケットから削除するオブジェクトの一意の名前。

モック出力

アクションは、プレビュー環境の外部サービスまたはリソースとやり取りしません。モック出力フィールドは、テスト目的でプレビュー環境のコネクタの動作をシミュレートする JSON 式を提供するために使用されます。このスニペットは、ライブ環境で公開されたアプリのコネクタレスポンスと同様に、アクションの results マップに保存されます。

このフィールドを使用すると、コネクタを介して外部サービスと通信することなく、さまざまな結果値、エラーシナリオ、エッジケース、不満のあるパスのシミュレートなど、オートメーション内の他のアクションに対するさまざまなシナリオとその影響をテストできます。

Amazon S3: オブジェクトの取得

Amazon S3 GetObject オペレーションを使用して、指定された Amazon S3 バケットからキー (ファイルパス) によって識別されるオブジェクトを取得します。

プロパティ

コネクタ

このアクションによって実行されるオペレーションに使用されるコネクタ。設定済みコネクタは、オペレーションを実行するための適切な認証情報と、オペレーションで参照されるリソースを含む AWS リージョンなどのその他の設定情報を使用して設定する必要があります。

設定

GetObject コマンドで使用する必須オプション。オプションは以下のとおりです:

Note

Amazon S3 GetObject オペレーションの詳細については、Amazon Simple Storage Service API リファレンスの [GetObject](#) を参照してください。

- バケット: オブジェクトを取得する Amazon S3 バケットの名前。
- キー: Amazon S3 バケットから取得するオブジェクトの一意の名前。

モック出力

アクションは、プレビュー環境の外部サービスまたはリソースとやり取りしません。モック出力フィールドは、テスト目的でプレビュー環境のコネクタの動作をシミュレートする JSON 式を提供するために使用されます。このスニペットは、ライブ環境で公開されたアプリのコネクタレスポンスと同様に、アクションのresultsマップに保存されます。

このフィールドを使用すると、コネクタを介して外部サービスと通信することなく、さまざまな結果値、エラーシナリオ、エッジケース、不満のあるパスのシミュレートなど、オートメーション内の他のアクションに対するさまざまなシナリオとその影響をテストできます。

Amazon S3: オブジェクトを一覧表示する

Amazon S3 ListObjects オペレーションを使用して、指定された Amazon S3 バケット内のオブジェクトを一覧表示します。

プロパティ

コネクタ

このアクションによって実行されるオペレーションに使用されるコネクタ。設定済みコネクタは、オペレーションを実行するための適切な認証情報と、オペレーションで参照されるリソースを含む AWS リージョンなどのその他の設定情報を使用して設定する必要があります。

設定

ListObjects コマンドで使用する必須オプション。オプションは以下のとおりです:

Note

Amazon S3 ListObjects オペレーションの詳細については、Amazon Simple Storage Service API リファレンスの[ListObjects](#)を参照してください。

- **バケット:** オブジェクトを一覧表示する Amazon S3 バケットの名前。

モック出力

アクションは、プレビュー環境の外部サービスまたはリソースとやり取りしません。モック出力フィールドは、テスト目的でプレビュー環境のコネクタの動作をシミュレートする JSON 式を提

供するために使用されます。このスニペットは、ライブ環境で公開されたアプリのコネクタレスポンスと同様に、アクションのresultsマップに保存されます。

このフィールドを使用すると、コネクタを介して外部サービスと通信することなく、さまざまな結果値、エラーシナリオ、エッジケース、不満のあるパスのシミュレートなど、オートメーション内の他のアクションに対するさまざまなシナリオとその影響をテストできます。

Amazon Textract: ドキュメントを分析する

Amazon Textract AnalyzeDocument オペレーションを使用して、検出された項目間の関係について入力ドキュメントを分析します。

プロパティ

コネクタ

このアクションによって実行されるオペレーションに使用されるコネクタ。設定済みコネクタは、オペレーションを実行するための適切な認証情報と、オペレーションで参照されるリソースを含む AWS リージョンなどのその他の設定情報を使用して設定する必要があります。

設定

AnalyzeDocument コマンドで使用するリクエストの内容。オプションは以下のとおりです:

Note

Amazon Textract AnalyzeDocument オペレーションの詳細については、「Amazon Textract デベロッパーガイド」の [AnalyzeDocument](#) を参照してください。

- ドキュメント/S3Object/バケット: Amazon S3 バケットの名前。S3 アップロードコンポーネントを使用してアクションにファイルが渡された場合、このパラメータは空のままにすることができます。
- ドキュメント/S3Object/名前: 入力ドキュメントのファイル名。S3 アップロードコンポーネントを使用してアクションにファイルが渡された場合、このパラメータは空のままにすることができます。
- ドキュメント/S3Object/バージョン: Amazon S3 バケットでバージョンングが有効になっている場合は、オブジェクトのバージョンを指定できます。S3 アップロードコンポーネントを使用してアクションにファイルが渡された場合、このパラメータは空のままにすることができます。

- **FeatureTypes**: 実行する分析のタイプのリスト。有効な値は、TABLES、FORMS、QUERIES、SIGNATURES、LAYOUT です。

モック出力

アクションは、プレビュー環境の外部サービスまたはリソースとやり取りしません。モック出力フィールドは、テスト目的でプレビュー環境のコネクタの動作をシミュレートする JSON 式を提供するために使用されます。このスニペットは、ライブ環境で公開されたアプリのコネクタレスポンスと同様に、アクションのresultsマップに保存されます。

このフィールドを使用すると、コネクタを介して外部サービスと通信することなく、さまざまな結果値、エラーシナリオ、エッジケース、不満のあるパスのシミュレートなど、オートメーション内の他のアクションに対するさまざまなシナリオとその影響をテストできます。

Amazon Textract: 経費を分析する

Amazon Textract AnalyzeExpense オペレーションを使用して、テキスト間の財務関係について入力ドキュメントを分析します。

プロパティ

コネクタ

このアクションによって実行されるオペレーションに使用されるコネクタ。設定済みコネクタは、オペレーションを実行するための適切な認証情報と、オペレーションで参照されるリソースを含む AWS リージョンなどのその他の設定情報を使用して設定する必要があります。

設定

AnalyzeExpense コマンドで使用するリクエストの内容。オプションは以下のとおりです:

Note

Amazon Textract AnalyzeExpense オペレーションの詳細については、「Amazon Textract デベロッパーガイド」の[AnalyzeExpense](#)を参照してください。

- **ドキュメント/S3Object/バケット**: Amazon S3 バケットの名前。S3 アップロードコンポーネントを使用してアクションにファイルが渡された場合、このパラメータは空のままにすることができます。

- Document / S3Object / Name: 入力ドキュメントのファイル名。S3 アップロードコンポーネントを使用してアクションにファイルが渡された場合、このパラメータは空のままにすることができます。
- ドキュメント/S3Object/バージョン: Amazon S3 バケットでバージョンングが有効になっている場合は、オブジェクトのバージョンを指定できます。S3 アップロードコンポーネントを使用してアクションにファイルが渡された場合、このパラメータは空のままにすることができます。

モック出力

アクションは、プレビュー環境の外部サービスまたはリソースとやり取りしません。モック出力フィールドは、テスト目的でプレビュー環境のコネクタの動作をシミュレートする JSON 式を提供するために使用されます。このスニペットは、ライブ環境で公開されたアプリケーションに対するコネクタレスポンスと同様に、アクションの results マップに保存されます。

このフィールドを使用すると、コネクタを介して外部サービスと通信することなく、さまざまな結果値、エラーシナリオ、エッジケース、不満のあるパスのシミュレートなど、オートメーション内の他のアクションに対するさまざまなシナリオとその影響をテストできます。

Amazon Textract: ID を分析する

Amazon Textract AnalyzeID オペレーションを使用して、ID ドキュメントの関連情報を分析します。

プロパティ

コネクタ

このアクションによって実行されるオペレーションに使用されるコネクタ。設定済みコネクタは、オペレーションを実行するための適切な認証情報と、オペレーションで参照されるリソースを含む AWS リージョンなどのその他の設定情報を使用して設定する必要があります。

設定

AnalyzeID コマンドで使用するリクエストの内容。オプションは以下のとおりです:

Note

Amazon Textract AnalyzeID オペレーションの詳細については、「Amazon Textract デベロッパーガイド」の [AnalyzeID](#) を参照してください。

- ドキュメント/S3Object/バケット: Amazon S3 バケットの名前。S3 アップロードコンポーネントを使用してアクションにファイルが渡された場合、このパラメータは空のままにすることができます。
- ドキュメント/S3Object/名前: 入力ドキュメントのファイル名。S3 アップロードコンポーネントを使用してアクションにファイルが渡された場合、このパラメータは空のままにすることができます。
- ドキュメント/S3Object/バージョン: Amazon S3 バケットでバージョンングが有効になっている場合は、オブジェクトのバージョンを指定できます。S3 アップロードコンポーネントを使用してアクションにファイルが渡された場合、このパラメータは空のままにすることができます。

モック出力

アクションは、プレビュー環境の外部サービスまたはリソースとやり取りしません。モック出力フィールドは、テスト目的でプレビュー環境のコネクタの動作をシミュレートする JSON 式を提供するために使用されます。このスニペットは、ライブ環境で公開されたアプリケーションに対するコネクタレスポンスと同様に、アクションの results マップに保存されます。

このフィールドを使用すると、コネクタを介して外部サービスと通信することなく、さまざまな結果値、エラーシナリオ、エッジケース、不満のあるパスのシミュレートなど、オートメーション内の他のアクションに対するさまざまなシナリオとその影響をテストできます。

Amazon Textract: ドキュメントテキストを検出する

Amazon Textract DetectDocumentText オペレーションを使用して、入力ドキュメント内のテキスト行とテキスト行を構成する単語を検出します。

プロパティ

コネクタ

このアクションによって実行されるオペレーションに使用されるコネクタ。設定済みコネクタは、オペレーションを実行するための適切な認証情報と、オペレーションで参照されるリソースを含む AWS リージョンなどのその他の設定情報を使用して設定する必要があります。

設定

DetectDocumentText コマンドで使用するリクエストの内容。オプションは以下のとおりです:

Note

Amazon Textract DetectDocumentText オペレーションの詳細については、Amazon Textract [DetectDocumentText](#)」を参照してください。

- ドキュメント/S3Object/バケット: Amazon S3 バケットの名前。S3 アップロードコンポーネントを使用してアクションにファイルが渡された場合、このパラメータは空のままにすることができます。
- ドキュメント/S3Object/名前: 入力ドキュメントのファイル名。S3 アップロードコンポーネントを使用してアクションにファイルが渡された場合、このパラメータは空のままにすることができます。
- ドキュメント/S3Object/バージョン: Amazon S3 バケットでバージョンニングが有効になっている場合は、オブジェクトのバージョンを指定できます。S3 アップロードコンポーネントを使用してアクションにファイルが渡された場合、このパラメータは空のままにすることができます。

モック出力

アクションは、プレビュー環境の外部サービスまたはリソースとやり取りしません。モック出力フィールドは、テスト目的でプレビュー環境のコネクタの動作をシミュレートする JSON 式を提供するために使用されます。このスニペットは、ライブ環境で公開されたアプリケーションに対するコネクタレスポンスと同様に、アクションの results マップに保存されます。

このフィールドを使用すると、コネクタを介して外部サービスと通信することなく、さまざまな結果値、エラーシナリオ、エッジケース、不満のあるパスのシミュレートなど、オートメーション内の他のアクションに対するさまざまなシナリオとその影響をテストできます。

Amazon Bedrock: GenAI プロンプト

[Amazon Bedrock InvokeModel](#) オペレーションを使用して、アクションプロパティで指定されたプロンプトパラメータと推論パラメータを使用して推論を実行します。アクションは、テキスト、イメージ、埋め込みを生成できます。

プロパティ

コネクタ

このアクションによって実行されるオペレーションに使用されるコネクタ。このアクションを正常に使用するには、Amazon Bedrock ランタイムをサービスとしてコネクタを設定する必要があります。

設定済みコネクタは、オペレーションを実行するための適切な認証情報と、オペレーションで参照されるリソースを含む AWS リージョンなどのその他の設定情報を使用して設定する必要があります。

モデル

Amazon Bedrock がリクエストを処理するために使用される基盤モデル。Amazon Bedrock のモデルの詳細については、[「Amazon Bedrock ユーザーガイド」の「Amazon Bedrock 基盤モデル情報」](#)を参照してください。

入力タイプ

Amazon Bedrock モデルに送信する入力の入力タイプ。指定できる値は、テキスト、ドキュメント、イメージです。入力タイプを選択できない場合、設定されたモデルではサポートされていない可能性があります。

ユーザープロンプト

レスポンスを生成するために処理する Amazon Bedrock モデルに送信するプロンプト。静的テキストを入力するか、パラメータ、オートメーションの以前のアクション、または別のオートメーションを使用するコンポーネントなど、アプリケーションの別の部分から入力を渡すことができます。次の例は、コンポーネントまたは以前のアクションから値を渡す方法を示しています。

- パラメータを使用してコンポーネントから値を渡すには: `{{params.paramName}}`
- 前のアクションから値を渡すには: `{{results.actionName}}`

システムプロンプト (Claude モデル)

リクエストの処理時に Amazon Bedrock モデルで使用されるシステムプロンプト。システムプロンプトは、Claude モデルにコンテキスト、手順、またはガイドラインを提供するために使用されます。

リクエスト設定

さまざまなリクエスト設定とモデル推論パラメータを設定します。以下の設定を設定できます。

- 温度: リクエストの処理時に Amazon Bedrock モデルで使用される温度。温度によって、Bedrock モデルの出力のランダム性または創造性が決まります。温度が高いほど、レスポンスはよりクリエイティブで分析性が低くなります。指定できる値は `0-10` です。
- 最大トークン: Amazon Bedrock モデルの出力の長さを制限します。

- TopP: nucleus サンプリングでは、モデルは後続の各トークンのすべてのオプションに対する累積分布を低い確率順で計算し、TopP で指定された特定の確率に達すると切り捨てます。温度または TopP のいずれかを変更する必要がありますが、両方を変更することはできません
- Stop Sequences: モデルがリクエストの処理と出力の生成を停止するシーケンス。

詳細については、「Amazon Bedrock [ユーザーガイド](#)」の「[基盤モデルの推論リクエストパラメータとレスポンスフィールド](#)」を参照してください。

停止シーケンス

Amazon Bedrock ガードレール ID とバージョンを入力します。ガードレールは、ユースケースと責任ある AI ポリシーに基づいて保護を実装するために使用されます。詳細については、「[Amazon Bedrock ユーザーガイド](#)」の「[Amazon Bedrock ガードレールを使用したモデルの有害なコンテンツの停止](#)」を参照してください。

モック出力

アクションは、プレビュー環境の外部サービスまたはリソースとやり取りしません。モック出力フィールドは、テスト目的でプレビュー環境のコネクタの動作をシミュレートする JSON 式を提供するために使用されます。このスニペットは、ライブ環境で公開されたアプリケーションに対するコネクタレスポンスと同様に、アクションの results マップに保存されます。

このフィールドを使用すると、コネクタを介して外部サービスと通信することなく、さまざまな結果値、エラーシナリオ、エッジケース、不満のあるパスのシミュレートなど、オートメーション内の他のアクションに対するさまざまなシナリオとその影響をテストできます。

Amazon Bedrock: モデルを呼び出す

[Amazon Bedrock InvokeModel](#) オペレーションを使用して、リクエスト本文で指定されたプロンプトパラメータと推論パラメータを使用して推論を実行します。モデル推論を使用して、テキスト、イメージ、埋め込みを生成します。

プロパティ

コネクタ

このアクションによって実行されるオペレーションに使用されるコネクタ。このアクションを正常に使用するには、Amazon Bedrock ランタイムをサービスとしてコネクタを設定する必要があります。設定済みコネクタは、オペレーションを実行するための適切な認証情報と、オペレーションで参照されるリソースを含む AWS リージョンなどのその他の設定情報を使用して設定する必要があります。

設定

InvokeModel コマンドで使用するリクエストの内容。

Note

コマンドの例を含む Amazon Bedrock InvokeModel オペレーションの詳細については、「Amazon Bedrock API [InvokeModel](#)」を参照してください。

モック出力

アクションは、プレビュー環境の外部サービスまたはリソースとやり取りしません。モック出力フィールドは、テスト目的でプレビュー環境のコネクタの動作をシミュレートする JSON 式を提供するために使用されます。このスニペットは、ライブ環境で公開されたアプリケーションに対するコネクタレスポンスと同様に、アクションの results マップに保存されます。

このフィールドを使用すると、コネクタを介して外部サービスと通信することなく、さまざまな結果値、エラーシナリオ、エッジケース、不満のあるパスのシミュレートなど、オートメーション内の他のアクションに対するさまざまなシナリオとその影響をテストできます。

JavaScript

カスタム JavaScript 関数を実行して、指定された値を返します。

Important

App Studio は、サードパーティーまたはカスタム JavaScript ライブラリの使用をサポートしていません。

プロパティ

ソースコード

アクションによって実行される JavaScript コードスニペット。

Tip

AI を使用して、次の手順を実行して JavaScript を生成できます。

1. 展開アイコンを選択して、展開された JavaScript エディタを開きます。

2. (オプション): コードの変更トグルを有効にして、既存の JavaScript を変更します。それ以外の場合、AI は既存の JavaScript を置き換えます。
3. JavaScript の生成で、JavaScript で何をするかを記述します。例: **Add two numbers**。
4. 送信アイコンを選択して JavaScript を生成します。

オートメーションを呼び出す

指定されたオートメーションを実行します。

プロパティ

自動化を呼び出す

アクションによって実行されるオートメーション。

Eメールの送信

Amazon SES SendEmail オペレーションを使用して E メールを送信します。

プロパティ

コネクタ

このアクションによって実行されるオペレーションに使用されるコネクタ。設定済みコネクタは、オペレーションを実行するための適切な認証情報と、オペレーションで参照されるリソースを含む AWS リージョンなどのその他の設定情報を使用して設定する必要があります。

設定

SendEmail コマンドで使用するリクエストの内容。オプションは以下のとおりです:

Note

Amazon SES SendEmail オペレーションの詳細については、Amazon Simple Email Service API リファレンスの [SendEmail](#) を参照してください。

モック出力

アクションは、プレビュー環境の外部サービスまたはリソースとやり取りしません。モック出力フィールドは、テスト目的でプレビュー環境のコネクタの動作をシミュレートする JSON 式を提

供するために使用されます。このスニペットは、ライブ環境で公開されたアプリのコネクタレスポンスと同様に、アクションのresultsマップに保存されます。

このフィールドを使用すると、コネクタを介して外部サービスと通信することなく、さまざまな結果値、エラーシナリオ、エッジケース、不満のあるパスのシミュレートなど、オートメーション内の他のアクションに対するさまざまなシナリオとその影響をテストできます。

エンティティとデータアクション: アプリのデータモデルを設定する

エンティティは App Studio のデータテーブルです。エンティティはデータソースのテーブルと直接やり取りします。エンティティには、その中のデータを記述するフィールド、データを検索して返すクエリ、エンティティのフィールドをデータソースの列に接続するためのマッピングが含まれます。

トピック

- [データモデルを設計する際のベストプラクティス](#)
- [App Studio アプリでのエンティティの作成](#)
- [App Studio アプリでエンティティを設定または編集する](#)
- [エンティティの削除](#)
- [AWS App Studio のマネージドデータエンティティ](#)

データモデルを設計する際のベストプラクティス

以下のベストプラクティスを使用して、アプリケーションの要件を満たし、データインフラストラクチャの長期的な信頼性とパフォーマンスを保証する App Studio アプリケーションで AWS 使用するために、堅牢でスケーラブル、安全なリレーショナルデータモデルを作成します。

- 適切な AWS データサービスを選択する: 要件に応じて、適切な AWS データサービスを選択します。例えば、オンライントランザクション処理 (OLTP) アプリケーションの場合、MySQL や PostgreSQL などのさまざまなデータベースエンジンをサポートするクラウドネイティブ、リレーショナル、フルマネージド型のデータベースサービスである Amazon Aurora などのデータベース (DB) を検討できます。App Studio でサポートされている Aurora バージョンの完全なリストについては、「」を参照してください[Amazon Aurora に接続する](#)。一方、オンライン分析処理 (OLAP) のユースケースでは、非常に大きなデータセットに対して複雑なクエリを実行できるクラウドデータウェアハウスである Amazon Redshift の使用を検討してください。これらのクエリの完了には

多くの場合 (数秒) がかかるため、Amazon Redshift は低レイテンシーのデータアクセスを必要とする OLTP アプリケーションには適していません。

- スケーラビリティの設計: 将来の成長とスケーラビリティを念頭に置いてデータモデルを計画します。適切なデータサービス、データベースインスタンスのタイプと設定 (プロビジョニングされた容量など) を選択するときは、予想されるデータ量、アクセスパターン、パフォーマンス要件などの要因を考慮してください。
- Aurora サーバーレスでのスケーリングの詳細については、[「Aurora Serverless V2 のパフォーマンスとスケーリング」](#)を参照してください。
- データの正規化: データベースの正規化の原則に従って、データの冗長性を最小限に抑え、データの整合性を向上させます。これには、適切なテーブルの作成、プライマリキーと外部キーの定義、エンティティ間の関係の確立が含まれます。App Studio では、あるエンティティからデータをクエリするときに、クエリで join句を指定することで、別のエンティティから関連データを取得できます。
- 適切なインデックス作成を実装する: 最も重要なクエリとアクセスパターンを特定し、パフォーマンスを最適化するための適切なインデックスを作成します。
- AWS データサービス機能を活用する: 自動バックアップ、マルチ AZ 配置、自動ソフトウェア更新など、選択した AWS データサービスが提供する機能を活用します。
- データを保護する: IAM () ポリシー、テーブルとスキーマへのアクセス許可が制限されたデータベースユーザーの作成、保管中および転送中の暗号化など、堅牢なセキュリティ対策を実装します。AWS Identity and Access Management
- パフォーマンスのモニタリングと最適化: データベースのパフォーマンスを継続的にモニタリングし、リソースのスケーリング、クエリの最適化、データベース設定の調整など、必要に応じて調整を行います。
- データベース管理の自動化: Aurora Autoscaling、Performance Insights for Aurora、AWS Database Migration Service などの AWS サービスを活用して、データベース管理タスクを自動化し、運用オーバーヘッドを削減します。
- デザスタリカバリとバックアップ戦略を実装する: Aurora 自動バックアップ、point-in-timeリカバリ、クロスリージョンレプリカ設定などの機能を活用して、明確に定義されたバックアップとリカバリ計画があることを確認します。
- AWS ベストプラクティスとドキュメントに従う: 選択したデータサービスに関する AWS 最新のベストプラクティス、ガイドライン、ドキュメント up-to-dateし、データモデルと実装が AWS 推奨事項と一致していることを確認します。

各 AWS データサービスの詳細なガイダンスについては、以下のトピックを参照してください。

- [Amazon Aurora のベストプラクティス](#)
- [Amazon Aurora MySQL のベストプラクティス](#)
- [Amazon Redshift クエリのパフォーマンス調整](#)
- [Amazon DynamoDB でデータをクエリおよびスキャンするためのベストプラクティス](#)

App Studio アプリでのエンティティの作成

App Studio アプリでエンティティを作成するには、4つの方法があります。次のリストには、各メソッドとその利点、およびそのメソッドを使用してエンティティを作成および設定する手順へのリンクが含まれています。

- [既存のデータソースからエンティティを作成する](#): 既存のデータソーステーブルからエンティティとそのフィールドを自動的に作成し、そのフィールドをデータソーステーブルの列にマッピングします。このオプションは、App Studio アプリで使用する既存のデータソースがある場合に推奨されます。
- [App Studio マネージドデータソースを使用したエンティティの作成](#): App Studio が管理するエンティティと DynamoDB テーブルを作成します。DynamoDB テーブルは、エンティティを更新すると自動的に更新されます。このオプションを使用すると、サードパーティーのデータソースを手動で作成、管理、接続したり、エンティティフィールドからテーブル列へのマッピングを指定したりする必要はありません。アプリのデータモデリングと設定はすべて App Studio で行われます。このオプションは、独自のデータソースを管理したくない場合や、DynamoDB テーブルとその機能がアプリに十分である場合に適しています。
- [空のエンティティの作成](#): 空のエンティティを最初から完全に作成します。このオプションは、管理者によって作成された既存のデータソースやコネクタがなく、外部データソースに制約されることなくアプリケーションのデータモデルを柔軟に設計する場合に適しています。エンティティは、作成後にデータソースに接続できます。
- [AI を使用したエンティティの作成](#): 指定されたエンティティ名に基づいてエンティティ、フィールド、データアクション、サンプルデータを生成します。このオプションは、アプリケーションのデータモデルについて考えているが、エンティティへの変換を支援したい場合に推奨されます。

既存のデータソースからエンティティを作成する

データソースのテーブルを使用して、エンティティとそのフィールドを自動的に作成し、エンティティフィールドをテーブルの列にマッピングします。このオプションは、App Studio アプリで使用する既存のデータソースがある場合に推奨されます。

1. 必要に応じて、アプリケーションに移動します。
2. キャンバスの上部にあるデータタブを選択します。
3. アプリにエンティティがない場合は、+ エンティティの作成を選択します。それ以外の場合は、左側のエンティティメニューで + 追加を選択します。
4. 既存のデータソースからテーブルを使用するを選択します。
5. Connector で、エンティティの作成に使用するテーブルを含むコネクタを選択します。
6. Table で、エンティティの作成に使用するテーブルを選択します。
7. データアクションの作成チェックボックスを選択して、データアクションを作成します。
8. [エンティティの作成] を選択します。これでエンティティが作成され、左側のエンティティパネルに表示されます。
9. 「」の手順に従って、新しいエンティティを設定します [App Studio アプリでエンティティを設定または編集する](#)。エンティティは既存のデータソースで作成されたため、フィールド、接続されたデータソース、フィールドマッピングなど、一部のプロパティまたはリソースが既に作成されていることに注意してください。また、作成時にデータアクションの作成チェックボックスを選択した場合、エンティティにはデータアクションが含まれます。

App Studio マネージドデータソースを使用したエンティティの作成

App Studio によって管理されるマネージドエンティティと対応する DynamoDB テーブルを作成します。DynamoDB テーブルが関連付けられた AWS アカウントに存在する間、App Studio アプリのエンティティに変更を加えると、DynamoDB テーブルは自動的に更新されます。このオプションを使用すると、サードパーティーのデータソースを手動で作成、管理、接続したり、エンティティフィールドからテーブル列へのマッピングを指定したりする必要はありません。このオプションは、独自のデータソースを管理したくない場合や、DynamoDB テーブルとその機能がアプリに十分である場合に適しています。マネージドエンティティの詳細については、「」を参照してください [AWS App Studio のマネージドデータエンティティ](#)。

複数のアプリケーションで同じマネージドエンティティを使用できます。手順については、「[既存のデータソースからエンティティを作成する](#)」を参照してください。

1. 必要に応じて、アプリケーションに移動します。
2. キャンバスの上部にあるデータタブを選択します。
3. アプリにエンティティがない場合は、+ エンティティの作成を選択します。それ以外の場合は、左側のエンティティメニューで + 追加を選択します。
4. App Studio マネージドエンティティの作成を選択します。

5. エンティティ名で、エンティティの名前を指定します。
6. プライマリキーで、エンティティのプライマリキーの名前を指定します。プライマリキーはエンティティの一意的識別子であり、エンティティの作成後に変更することはできません。
7. プライマリキーのデータ型で、エンティティのプライマリキーのデータ型を選択します。エンティティの作成後にデータ型を変更することはできません。
8. [エンティティの作成] を選択します。これでエンティティが作成され、左側のエンティティパネルに表示されます。
9. 「」の手順に従って、新しいエンティティを設定します [App Studio アプリでエンティティを設定または編集する](#)。エンティティはマネージドデータで作成されたため、プライマリキーフィールドや接続されたデータソースなど、一部のプロパティまたはリソースがすでに作成されていることに注意してください。

空のエンティティの作成

空のエンティティを最初から完全に作成します。このオプションは、管理者によって作成された既存のデータソースまたはコネクタがない場合に推奨されます。空のエンティティを作成すると、外部データソースに制約されることなく App Studio アプリ内でエンティティを設計できるため、柔軟性が得られます。アプリケーションのデータモデルを設計し、それに応じてエンティティを設定した後も、後で外部データソースに接続できます。

1. 必要に応じて、アプリケーションに移動します。
2. キャンバスの上部にあるデータタブを選択します。
3. アプリにエンティティがない場合は、+ エンティティの作成を選択します。それ以外の場合は、左側のエンティティメニューで + 追加を選択します。
4. エンティティの作成を選択します。
5. [エンティティの作成] を選択します。これでエンティティが作成され、左側のエンティティパネルに表示されます。
6. 「」の手順に従って、新しいエンティティを設定します [App Studio アプリでエンティティを設定または編集する](#)。

AI を使用したエンティティの作成

指定されたエンティティ名に基づいて、エンティティ、フィールド、データアクション、サンプルデータを生成します。このオプションは、アプリケーションのデータモデルについて考えているが、エンティティへの変換を支援したい場合に推奨されます。

1. 必要に応じて、アプリケーションに移動します。
2. キャンバスの上部にあるデータタブを選択します。
3. アプリにエンティティがない場合は、+ エンティティの作成を選択します。それ以外の場合は、左側のエンティティメニューで + 追加を選択します。
4. AI を使用してエンティティを作成するを選択します。
5. エンティティ名で、エンティティの名前を指定します。この名前は、エンティティのフィールド、データアクション、およびサンプルデータを生成するために使用されます。
6. データアクションの作成チェックボックスを選択して、データアクションを作成します。
7. エンティティの生成を選択します。これでエンティティが作成され、左側のエンティティパネルに表示されます。
8. 「」の手順に従って、新しいエンティティを設定します [App Studio アプリでエンティティを設定または編集する](#)。エンティティは AI で作成されたため、エンティティにはすでに生成されたフィールドが含まれていることに注意してください。また、作成時にデータアクションの作成チェックボックスを選択した場合、エンティティにはデータアクションが含まれます。

App Studio アプリでエンティティを設定または編集する

次のトピックを使用して、App Studio アプリケーションでエンティティを設定します。

トピック

- [エンティティ名の編集](#)
- [エンティティフィールドの追加、編集、または削除](#)
- [データアクションの作成、編集、または削除](#)
- [サンプルデータの追加または削除](#)
- [接続されたデータソースフィールドとマップフィールドを追加または編集する](#)

エンティティ名の編集

1. 必要に応じて、編集するエンティティに移動します。
2. 設定タブのエンティティ名でエンティティ名を更新し、テキストボックスの外部を選択して変更を保存します。

エンティティフィールドの追加、編集、または削除

Tip

CTRL+Z キーを押して、エンティティに対する最新の変更を元に戻すことができます。

1. 必要に応じて、編集するエンティティに移動します。
2. 設定タブのフィールドで、エンティティのフィールドのテーブルを表示します。エンティティフィールドには次の列があります。
 - 表示名: 表示名はテーブルヘッダーまたはフォームフィールドに似ており、アプリケーションユーザーが表示できます。スペースと特殊文字を含めることができますが、エンティティ内で一意である必要があります。
 - システム名: システム名は、フィールドを参照するためにコードで使用される一意の識別子です。Amazon Redshift テーブルの列にマッピングする場合、Amazon Redshift テーブルの列名と一致する必要があります。
 - データ型: Integer、Booleanなど、このフィールドに保存されるデータのタイプString。
3. フィールドを追加するには:
 - a. AI を使用してエンティティ名と接続されたデータソースに基づいてフィールドを生成するには、追加フィールドの生成を選択します。
 - b. 1つのフィールドを追加するには、+ 追加フィールドを選択します。
4. フィールドを編集するには:
 - a. 表示名を編集するには、表示名テキストボックスに目的の値を入力します。フィールドのシステム名が編集されていない場合は、表示名の新しい値に更新されます。
 - b. システム名を編集するには、システム名テキストボックスに目的の値を入力します。
 - c. データ型を編集するには、データ型ドロップダウンメニューを選択し、リストから目的の型を選択します。
 - d. フィールドのプロパティを編集するには、フィールドの歯車アイコンを選択します。次のリストは、フィールドプロパティの詳細を示しています。
 - 必須: フィールドがデータソースで必要な場合は、このオプションを有効にします。

- **プライマリキー:** フィールドがデータソースのプライマリキーにマッピングされている場合は、このオプションを有効にします。
- **Unique:** このフィールドの値が一意でなければならない場合は、このオプションを有効にします。
- **データソースのデフォルトを使用する:** 自動増分やイベントタイムスタンプを使用するなど、フィールドの値がデータソースによって提供される場合は、このオプションを有効にします。
- **データ型オプション:** 特定のデータ型のフィールドは、最小値や最大値などのデータ型オプションで設定できます。

5. フィールドを削除するには、削除するフィールドのごみ箱アイコンを選択します。

データアクションの作成、編集、または削除

データアクションは、すべてのレコードの取得や ID によるレコードの取得など、エンティティのデータに対してアクションを実行するためにアプリケーションで使用されます。データアクションを使用して、テーブルや詳細ビューなどのコンポーネントに表示される、指定された条件に一致するデータを検索して返すことができます。

目次

- [データアクションの作成](#)
- [データアクションの編集または設定](#)
- [データアクション条件演算子と例](#)
 - [データベースによる条件演算子のサポート](#)
 - [データアクション条件の例](#)
- [データアクションの削除](#)

データアクションの作成

Tip

CTRL+Z キーを押して、エンティティに対する最新の変更を元に戻すことができます。

1. 必要に応じて、データアクションを作成するエンティティに移動します。

2. データアクションタブを選択します。
3. データアクションを作成するには、次の 2 つの方法があります。
 - (推奨) AI を使用して、エンティティ名、フィールド、接続されたデータソースに基づいてデータアクションを生成するには、データアクションの生成を選択します。次のアクションが生成されます。
 1. getAll: エンティティからすべてのレコードを取得します。このアクションは、レコードのリストを表示したり、複数のレコードに対して一度にオペレーションを実行したりする必要がある場合に便利です。
 2. getById: 一意の識別子 (ID またはプライマリキー) に基づいてエンティティから 1 つのレコードを取得します。このアクションは、特定のレコードに対してオペレーションを表示または実行する必要がある場合に便利です。
 - 単一のデータアクションを追加するには、+ データアクションの追加を選択します。
4. 新しいデータアクションを表示または設定するには、次のセクション「」を参照してください。[データアクションの編集または設定](#)。

データアクションの編集または設定

1. 必要に応じて、データアクションを作成するエンティティに移動します。
2. データアクションタブを選択します。
3. フィールドで、クエリによって返されるフィールドを設定します。デフォルトでは、エンティティ内のすべての設定済みフィールドが選択されます。

次の手順を実行して、データアクションに Joins を追加することもできます。

1. + 結合を追加 を選択してダイアログボックスを開きます。
2. 関連エンティティで、現在のエンティティと結合するエンティティを選択します。
3. エイリアスで、必要に応じて関連エンティティの一時エイリアス名を入力します。
4. 結合タイプで、目的の結合タイプを選択します。
5. 各エンティティからフィールドを選択して、結合句を定義します。
6. 追加 を選択して結合を作成します。

作成されると、結合が結合セクションに表示され、フィールドを返すドロップダウンに追加のフィールドが使用可能になります。エンティティ間の連鎖結合など、複数の結合を追加できます。結合されたエンティティのフィールドでフィルタリングおよびソートすることもできます。

結合を削除するには、結合の横にあるごみ箱アイコンを選択します。これにより、それらのフィールドを使用して、その結合からすべてのフィールドが削除され、依存結合または制約が破棄されます。

4. 条件で、クエリの出力をフィルタリングするルールを追加、編集、または削除します。ルールをグループに整理し、複数のルールを AND または OR ステートメントと連鎖させることができます。使用できる演算子の詳細については、「」を参照してください [データアクション条件演算子と例](#)。
5. ソートでは、属性を選択し、昇順または降順を選択して、クエリ結果のソート方法を設定します。ソートルールの横にあるごみ箱アイコンを選択して、ソート設定を削除できます。
6. 変換結果では、カスタム JavaScript を入力して、結果を表示またはオートメーションに送信する前に結果を変更またはフォーマットできます。
7. 出力プレビューで、設定されたフィールド、フィルター、ソート、JavaScript に基づいてクエリ出力のプレビューテーブルを表示します JavaScript 。

データアクション条件演算子と例

条件演算子を使用して、設定された式値をエンティティ列と比較し、データベースオブジェクトのサブセットを返すことができます。使用できる演算子は、列のデータ型と、Amazon Redshift、Amazon Aurora、Amazon DynamoDB など、エンティティが接続されているデータベースのタイプによって異なります。

次の条件演算子は、すべてのデータベースサービスで使用できます。

- = および !=: すべてのデータ型 (プライマリキー列を除く) で使用できます。
- <=、>=、<、および >=: 数値列でのみ使用できます。
- IS NULL および IS NOT NULL: null または空の値を持つ列を一致させるために使用されます。Null 値はデータベースごとに解釈が異なることがよくありますが、App Studio では、NULL オペレーターは接続されたデータベーステーブルに Null 値を持つレコードを照合して返します。

次の条件演算子は、それらをサポートするデータベースサービスに接続されているエンティティでのみ使用できます。

- LIKE および NOT LIKE (Redshift、Aurora): 接続されたデータベースでパターンベースのクエリを実行するために使用されます。LIKE 演算子は、指定されたパターンに一致するレコードを検索して返すため、検索機能の柔軟性を提供します。パターンは、パターン内の任意の文字または文字の

シーケンスに一致するワイルドカード文字を使用して定義します。各データベース管理システムには一意のワイルドカード文字のセットがありますが、最も一般的な 2 つは任意の文字数 (0 を含む) を表し%、1 文字_を表すことです。

- Contains および Not Contains (DynamoDB): 大文字と小文字を区別する検索を実行して、指定されたテキストが列値内で見つかったかどうかを判断するために使用されます。
- Starts With および Not Starts With (DynamoDB): 大文字と小文字を区別する検索を実行して、指定されたテキストが列値の先頭で見つかったかどうかを判断するために使用されます。

データベースによる条件演算子のサポート

次の表は、App Studio に接続できる各データベースでサポートされているデータアクション条件演算子を示しています。

	=, !=, <, >, <=, >=	LIKE, NOT LIKE	含まれる、含まれない	で始まるがで始まらない	IS NULL, IS NOT NULL
DynamoDB	はい	なし	はい	はい	はい
Aurora	はい	あり	なし	なし	はい
Redshift	はい	あり	なし	なし	はい

データアクション条件の例

、namecityおよび hireDateフィールドを持つ複数の項目を含む次のデータベーステーブルを検討してください。

名前	city	hireDate
アダム	Seattle	2025-03-01
アドリエンヌ	Boston	2025-03-05
Bob	アルバカーキ	2025-03-06
Carlos	シカゴ	2025-03-10
キャロライン	NULL	2025-03-12

名前	city	hireDate
リタ	マイアミ	2025-03-15

ここで、指定した条件に一致する項目の name フィールドを返すデータアクションを App Studio で作成することを検討してください。次のリストには、条件の例と、それぞれに対してテーブルが返す値が含まれています。

Note

例は SQL の例としてフォーマットされています。App Studio では表示されない場合がありますが、演算子の動作を説明するために使用されます。

- WHERE name LIKE 'Adam': を返します Adam。
- WHERE name LIKE 'A%': Adam と を返します Adrienne。
- WHERE name NOT LIKE 'B_B': Adam、Adrienne、Carlos、Caroline、および を返しません Rita。
- WHERE contains(name, 'ita'): を返します Rita。
- WHERE begins_with(name, 'Car'): Carlos と を返します Caroline。
- WHERE city IS NULL: を返しません Caroline。
- WHERE hireDate < "2025-03-06": Adam と を返しません Adrienne。
- WHERE hireDate >= DateTime.now().toISODate(): は現在の日付 DateTime.now().toISODate() を返します。現在の日付が 2025-03-10 のシナリオでは、式は Carlos、Caroline、および を返しません Rita。

Tip

式で日付と時刻を比較する方法の詳細については、「」を参照してください [日付および時間](#)。

データアクションの削除

App Studio エンティティからデータアクションを削除するには、次の手順に従います。

1. 必要に応じて、データアクションを削除するエンティティに移動します。
2. データアクションタブを選択します。
3. 削除するデータアクションごとに、編集の横にあるドロップダウンメニューを選択し、削除を選択します。
4. ダイアログボックスで確認 を選択します。

サンプルデータの追加または削除

App Studio アプリケーションのエンティティにサンプルデータを追加できます。アプリケーションのは公開されるまで外部サービスと通信しないため、サンプルデータを使用してプレビュー環境でアプリケーションとエンティティをテストできます。

1. 必要に応じて、編集するエンティティに移動します。
2. サンプルデータタブを選択します。
3. サンプルデータを生成するには、「より多くのサンプルデータを生成する」を選択します。
4. サンプルデータを削除するには、削除するデータのチェックボックスを選択し、Delete キーまたは Backspace キーを押します。保存 を選択して変更を保存します。

接続されたデータソースフィールドとマップフィールドを追加または編集する

Tip

CTRL+Z キーを押して、エンティティに対する最新の変更を元に戻すことができます。

1. 必要に応じて、編集するエンティティに移動します。
2. 接続タブを選択して、アプリケーションの公開時にデータが保存されるエンティティとデータソーステーブル間の接続を表示または管理します。データソーステーブルが接続されたら、エンティティフィールドをテーブルの列にマッピングできます。
3. Connector で、目的のデータソーステーブルへの接続を含むコネクタを選択します。コネクタの詳細については、「」を参照してください [コネクタを使用して App Studio を他の サービスに接続する](#)。
4. Table で、エンティティのデータソースとして使用するテーブルを選択します。
5. この表は、エンティティのフィールドと、それらがマッピングされているデータソース列を示しています。自動マップを選択すると、エンティティフィールドがデータソース列に自動的にマッ

ピングされます。各エンティティフィールドのドロップダウンでデータソース列を選択して、テーブル内のフィールドを手動でマッピングすることもできます。

エンティティの削除

App Studio アプリケーションからエンティティを削除するには、次の手順に従います。

Note

App Studio アプリからエンティティを削除しても、マネージドエンティティの対応する DynamoDB テーブルを含む、接続されたデータソーステーブルは削除されません。データソーステーブルは関連付けられた AWS アカウントに残り、必要に応じて対応するサービスから削除する必要があります。

エンティティを削除するには

1. 必要に応じて、アプリケーションに移動します。
2. [データ] タブを選択します。
3. 左側のエンティティメニューで、削除するエンティティの横にある楕円メニューを選択し、削除を選択します。
4. ダイアログボックスの情報を確認し、「削除」と入力 **confirm** して選択し、エンティティを削除します。

AWS App Studio のマネージドデータエンティティ

通常、外部データベーステーブルへの接続を使用して App Studio でエンティティを設定し、各エンティティフィールドを作成して、接続されたデータベーステーブルの列にマッピングする必要があります。データモデルを変更するときは、外部データベーステーブルとエンティティの両方を更新し、変更されたフィールドを再マッピングする必要があります。この方法は柔軟で、さまざまなタイプのデータソースを使用できますが、より事前の計画と継続的なメンテナンスが必要です。

マネージドエンティティは、App Studio がデータストレージと設定プロセス全体を管理するエンティティのタイプです。マネージドエンティティを作成すると、対応する DynamoDB テーブルが関連する AWS アカウントに作成されます。これにより、内の安全で透過的なデータ管理が保証されます AWS。マネージドエンティティでは、App Studio でエンティティのスキーマを設定すると、対応する DynamoDB テーブルも自動的に更新されます。

複数のアプリケーションでマネージドエンティティを使用する

App Studio アプリでマネージドエンティティを作成すると、そのエンティティを他の App Studio アプリで使用できます。これは、維持する単一の基盤となるリソースを提供することで、同一のデータモデルとスキーマを持つアプリケーションのデータストレージを設定するのに役立ちます。

複数のアプリケーションでマネージドエンティティを使用する場合、対応する DynamoDB テーブルに対するすべてのスキーマ更新は、マネージドエンティティが作成された元のアプリケーションを使用して行う必要があります。他のアプリケーションのエンティティに対してスキーマを変更しても、対応する DynamoDB テーブルは更新されません。

マネージドエンティティの制限

プライマリキーの更新制限: 作成後にエンティティのプライマリキー名またはタイプを変更することはできません。これは DynamoDB の破壊的な変更であり、既存のデータが失われる可能性があるためです。

列の名前変更: DynamoDB で列の名前を変更すると、元の列が元のデータのまま新しい列が実際に作成されます。元のデータは、新しい列に自動的にコピーされたり、元の列から削除されたりすることはありません。システム名と呼ばれるマネージドエンティティフィールドの名前は変更できますが、元の列とそのデータにアクセスできなくなります。表示名の変更に制限はありません。

データ型の変更: DynamoDB では、テーブルの作成後に列データ型を柔軟に変更できますが、このような変更が既存のデータやクエリロジック、精度に重大な影響を与える可能性があります。データ型の変更では、既存のすべてのデータを変換して新しい形式に準拠する必要があります。これは、大規模でアクティブなテーブルでは複雑です。さらに、データアクションは、データ移行が完了するまで予期しない結果を返す可能性があります。フィールドのデータ型を切り替えることはできますが、既存のデータは新しいデータ型に移行されません。

ソート列: DynamoDB では、ソートキーを使用してソートされたデータを取得できます。ソートキーは、パーティションキーとともに複合プライマリキーの一部として定義する必要があります。制限には、必須のソートキー、1つのパーティション内のソート、パーティション間のグローバルソートが含まれます。ホットパーティションを回避するには、ソートキーの慎重なデータモデリングが必要です。Sorting for Preview のマイルストーンはサポートされません。

Joins: 結合は DynamoDB ではサポートされていません。テーブルは、高価な結合操作を避けるために、設計によって非正規化されます。one-to-many関係をモデル化するために、子テーブルには親テーブルのプライマリキーを参照する属性が含まれています。マルチテーブルデータクエリでは、親テーブルから項目を検索して詳細を取得します。プレビューマイルストーンの一環として、マネージ

ドエンティティのネイティブ結合はサポートされません。回避策として、2つのエンティティのデータマージを実行できる自動化ステップを紹介します。これは、1レベルのルックアップと非常によく似ています。Sorting for Preview のマイルストーンはサポートされません。

Env ステージ: 公開はテストできますが、両方の環境で同じマネージドストアを使用できます

ページとオートメーションのパラメータ

パラメータは、アプリケーション内のさまざまなコンポーネント、ページ、オートメーション間で動的な値を渡すために使用される AWS App Studio の強力な機能です。パラメータを使用すると、柔軟でコンテキスト対応のエクスペリエンスを実現し、アプリケーションの応答性とパーソナライズを強化できます。この記事では、ページパラメータとオートメーションパラメータの2種類のパラメータについて説明します。

トピック

- [ページパラメータ](#)
- [自動化パラメータ](#)

ページパラメータ

ページパラメータはページ間で情報を送信する方法であり、App Studio アプリ内であるページから別のページに移動してコンテキストを維持したりデータを渡すときによく使用されます。ページパラメータは通常、名前と値で構成されます。

ページパラメータのユースケース

ページパラメータは、App Studio アプリケーション内の異なるページとコンポーネント間でデータを渡すために使用されます。これらは、次のユースケースで特に役立ちます。

1. 検索とフィルタリング: ユーザーがアプリのホームページで検索すると、検索用語を結果ページにパラメータとして渡すことができ、関連するフィルタリングされた項目のみを表示できます。たとえば、ユーザーが `#####` を検索する場合、`#####` の値を持つパラメータを製品出品ページに渡すことができます。
2. 項目の詳細の表示: ユーザーが製品などのリストをクリックすると、その項目の一意の識別子をパラメータとして詳細ページに渡すことができます。これにより、詳細ページに特定の項目に関するすべての情報を表示できます。たとえば、ユーザーがヘッドフォン製品をクリックすると、製品の一意の ID がパラメータとして製品の詳細ページに渡されます。

3. ページナビゲーションでユーザーコンテキストを渡す: ユーザーがページ間を移動するとき、パラメータはユーザーの場所、優先製品カテゴリ、ショッピングカートの内容、その他の設定などの重要なコンテキストを渡すことができます。たとえば、ユーザーがアプリでさまざまな製品カテゴリを閲覧すると、その場所と優先カテゴリがパラメータとして保持され、パーソナライズされた一貫したエクスペリエンスが提供されます。
4. ディープリンク: ページパラメータを使用して、アプリ内の特定のページへのリンクを共有またはブックマークします。
5. データアクション: パラメータ値を受け入れるデータアクションを作成して、渡されたパラメータに基づいてデータソースをフィルタリングおよびクエリできます。たとえば、製品出品ページで、関連する製品を取得するためのcategoryパラメータを受け入れるデータアクションを作成できます。

ページパラメータのセキュリティ上の考慮事項

ページパラメータはページ間でデータを渡す強力な方法ですが、正しく使用しないと機密情報が公開される可能性があるため、注意して使用する必要があります。留意すべき重要なセキュリティ上の考慮事項は次のとおりです。

1. URLs

- a. リスク: データアクションパラメータを含む URLs は、多くの場合、サーバーログ、ブラウザ履歴、その他の場所に表示されます。そのため、ユーザー認証情報、個人を特定できる情報 (PII)、またはその他の機密データなどの機密データをページパラメータ値に公開しないようにすることが重要です。
- b. 緩和策: 機密データに安全にマッピングできる識別子の使用を検討してください。たとえば、ユーザー名または E メールアドレスをパラメータとして渡す代わりに、ユーザー名または E メールを取得するために使用できるランダムな一意の識別子を渡すことができます。

自動化パラメータ

オートメーションパラメータは App Studio の強力な機能であり、UI、他のオートメーション、データアクションなどのさまざまなソースから動的値を渡すことで、柔軟で再利用可能なオートメーションを作成できます。自動化の実行時に実際の値に置き換えられるプレースホルダーとして機能し、毎回異なる入力で同じ自動化を使用できます。

オートメーション内では、パラメータには一意の名前があり、パラメータ変数の後にパラメータの名前が続く などを使用してパラメータの値を参照できます `{{params.customerId}}`。

この記事では、自動化パラメータの基本概念、使用法、ベストプラクティスなどについて詳しく説明します。

自動化パラメータの利点

自動化パラメータには、次のリストを含むいくつかの利点があります。

1. **再利用性:** パラメータを使用すると、異なる入力値でカスタマイズできる再利用可能なオートメーションを作成できるため、同じオートメーションロジックを異なる入力で再利用できます。
2. **柔軟性:** 値をオートメーションにハードコーディングする代わりに、パラメータを定義し、必要に応じて異なる値を指定できるため、オートメーションをより動的に適応させることができます。
3. **懸念の分離:** パラメータは、使用される特定の値から自動化ロジックを分離し、コードの整理と保守性を促進します。
4. **検証:** 各パラメータには文字列、数値、ブール値などのデータ型があり、実行時に検証されます。これにより、誤ったデータ型のリクエストは、カスタム検証コードなしで拒否されます。
5. **オプションパラメータと必須パラメータ:** オートメーションパラメータをオプションまたは必須として指定できます。オートメーションを実行するときは必要なパラメータを指定する必要がありますが、オプションのパラメータにはデフォルト値を含めることも、省略することもできます。この柔軟性により、提供されたパラメータに基づいてさまざまなシナリオを処理できる、より汎用性の高いオートメーションを作成できます。

シナリオとユースケース

シナリオ: 製品の詳細を取得する

製品 ID に基づいてデータベースから製品の詳細を取得するオートメーションがあるとします。この自動化には、`productId` というパラメータを含めることができます。

`productId` パラメータは、オートメーションの実行時に実際の製品 ID 値を入力できるプレースホルダーとして機能します。特定の製品 ID をオートメーションにハードコーディングする代わりに、`productId` パラメータを定義し、オートメーションを実行するたびに異なる製品 ID 値を渡すことができます。

コンポーネントのデータソースからこの自動化を呼び出し、二重中括弧構文を使用して、選択した製品の ID を `productId` パラメータとして渡すことができます `{{ui.productsTable.selectedRow.id}}`。これにより、ユーザーがテーブル (`ui.productsTable`) から製品を選択すると、オートメーションは選択した行の ID を `productId` パラメータとして渡すことで、選択した製品の詳細を取得します。

または、製品のリストをループし、製品の ID を `productId` パラメータとして渡すことで各製品の詳細を取得する別のオートメーションからこのオートメーションを呼び出すこともできます。このシナリオでは、`productId` パラメータ値はループの各反復で `{{product.id}}` 式から動的に提供されます。

`productId` パラメータと二重中括弧構文を使用することで、この自動化をより柔軟に再利用できます。製品ごとに個別のオートメーションを作成する代わりに、UI コンポーネントやその他のオートメーションなど、さまざまなソースからパラメータ値として適切な製品 ID を指定するだけで、任意の製品の詳細を取得できる単一のオートメーションを作成できます。

シナリオ: フォールバック値を使用したオプションのパラメータの処理

必須の「所有者」列を持つ「タスク」エンティティがあるが、このフィールドをオートメーションでオプションにし、所有者が選択されていない場合はフォールバック値を指定するシナリオを考えてみましょう。

1. Task エンティティの `Owner` フィールドにマッピング `Owner` するという名前のパラメータを使用してオートメーションを作成します。
2. エンティティには `Owner` フィールドが必要なため、`Owner` パラメータは必要な設定と同期します。
3. オートメーションで `Owner` パラメータをオプションにするには、このパラメータ `required` の設定をオフに切り替えます。
4. 自動化ロジックでは、のような式を使用できます `{{params.Owner || currentUser.userId}}`。この式は、`Owner` パラメータが指定されているかどうかを確認します。指定しない場合、現在のユーザーの所有者としての ID にフォールバックされます。
5. これにより、ユーザーがフォームまたはコンポーネントで所有者を選択しない場合、オートメーションは現在のユーザーをタスクの所有者として自動的に割り当てます。

`Owner` パラメータ `required` の設定を切り替え、フォールバック式を使用することで、エンティティフィールド要件から切り離し、オートメーションではオプションにし、パラメータが指定されていない場合はデフォルト値を指定できます。

オートメーションパラメータタイプの定義

パラメータタイプを使用してデータ型を指定し、要件を設定することで、オートメーションの入力を制御できます。これにより、予想される入力でオートメーションを確実に実行できます。

エンティティからの型の同期

エンティティフィールド定義からパラメータタイプと要件を動的に同期することで、エンティティデータを操作するオートメーションの構築が効率化され、パラメータに常に最新のエンティティフィールドタイプと要件が反映されます。

次の手順では、エンティティからパラメータタイプを同期するための一般的な手順について詳しく説明します。

1. 型付きフィールド (ブール値、数値など) を持つエンティティを作成し、必要に応じてフィールドをマークします。
2. 新しいオートメーションを作成します。
3. オートメーションにパラメータを追加し、タイプを選択するときは、同期するエンティティフィールドを選択します。データ型と必要な設定は、マッピングされたエンティティフィールドから自動的に同期されます。
4. 必要に応じて、パラメータごとにオン/オフを切り替えることで、「必須」設定を上書きできます。つまり、必要なステータスはエンティティフィールドと同期されませんが、それ以外の場合は同期されたままになります。

型を手動で定義する

エンティティから同期せずにパラメータタイプを手動で定義することもできます。

カスタムパラメータタイプを定義することで、エンティティフィールドマッピングに依存することなく、特定の入力タイプを受け入れ、必要に応じてオプションまたは必須のパラメータを処理するオートメーションを作成できます。

1. 型付きフィールド (ブール値、数値など) を持つエンティティを作成し、必要に応じてフィールドをマークします。
2. 新しいオートメーションを作成します。
3. オートメーションにパラメータを追加し、タイプを選択するときは、目的のタイプを選択します。

自動化パラメータに渡される動的値の設定

オートメーションのパラメータを定義したら、オートメーションを呼び出すときにパラメータに値を渡すことができます。パラメータ値は、次の2つの方法で渡すことができます。

1. コンポーネントトリガー: ボタンクリックなどのコンポーネントトリガーからオートメーションを呼び出す場合は、JavaScript 式を使用してコンポーネントコンテキストから値を渡すことができます。たとえば、 という名前のテキスト入力フィールドがある場合emailInput、その値を という式で E メールパラメータに渡すことができますui.emailInput.value。
2. その他のオートメーション: 別のオートメーションからオートメーションを呼び出す場合は、JavaScript 式を使用してオートメーションコンテキストから値を渡すことができます。たとえば、別のパラメータの値を渡すか、前のアクションステップの結果を渡すことができます。

型の安全性

文字列、数値、ブール値など、特定のデータ型を持つパラメータを定義することで、オートメーションに渡される値が期待される型であることを確認します。

Note

App Studio では、date (s) は ISO 文字列の日付であり、それらも検証されます。

このタイプの安全は、オートメーションロジックでエラーや予期しない動作につながる可能性のあるタイプの不一致を防ぐのに役立ちます。例えば、パラメータをとして定義した場合Number、そのパラメータに渡される値は数値であり、オートメーション内で追加の型チェックや変換を実行する必要はありません。

検証

パラメータに検証ルールを追加して、オートメーションに渡される値が特定の基準を満たすようにすることができます。

App Studio はパラメータの組み込み検証設定を提供していませんが、特定の制約に違反した場合にエラーをスローする JavaScript アクションをオートメーションに追加することで、カスタム検証を実装できます。

エンティティフィールドでは、最小値/最大値などの検証ルールのサブセットがサポートされています。ただし、レコードCreate/Update/Deleteアクションを実行する場合、これらはデータレイヤーでのみ自動化レベルで検証されません。

自動化パラメータのベストプラクティス

自動化パラメータが適切に設計され、保守可能で、使いやすいようにするには、次のベストプラクティスに従ってください。

1. わかりやすいパラメータ名を使用する: パラメータの目的またはコンテキストを明確に説明するパラメータ名を選択します。
2. パラメータの説明を指定する: パラメータを定義するときは、説明フィールドを活用して、その目的、制約、期待値を説明します。これらの説明は、パラメータを参照するときの JSDoc コメントと、オートメーションを呼び出すときにユーザーがパラメータの値を指定する必要があるユーザーインターフェイスに表示されます。
3. 適切なデータ型を使用する: 文字列、数値、ブール値、オブジェクトなど、予想される入力値に基づいて各パラメータのデータ型を慎重に検討します。
4. パラメータ値を検証する: 自動化内で適切な検証チェックを実装して、パラメータ値が特定の要件を満たしていることを確認してから、さらなるアクションに進みます。
5. フォールバック値またはデフォルト値を使用する: App Studio は現在パラメータのデフォルト値の設定をサポートしていませんが、オートメーションロジックでパラメータを使用するときにフォールバック値またはデフォルト値を実装できます。たとえば、param1パラメータが指定されていない場合、または false 値がある場合は、`のような式を使用してデフォルト値``{ params.param1 || "default value" }`を指定できます。
6. パラメータの一貫性を維持する: 同様のパラメータを必要とするオートメーションが複数ある場合は、それらのオートメーション全体でパラメータ名とデータ型の一貫性を維持してください。
7. ドキュメントパラメータの使用: 各パラメータの説明、目的、想定値、関連する例やエッジケースなど、オートメーションに関する明確なドキュメントを維持します。
8. 頻繁なレビューとリファクタリング: オートメーションとそのパラメータを定期的を確認し、必要に応じてパラメータをリファクタリングまたは統合して、明確性、保守性、再利用性を向上させます。
9. パラメータの数を制限する: パラメータには柔軟性がありますが、パラメータが多すぎるとオートメーションが複雑になり、使用が困難になる可能性があります。パラメータの数を必要なものみに制限することで、柔軟性とシンプルさのバランスを取ります。
10. パラメータのグループ化を検討する: 複数の関連パラメータを定義している場合は、それらを 1 つの#####パラメータにグループ化することを検討してください。
11. 個別の懸念事項: 1 つのパラメータを複数の目的に使用したり、無関係な値を 1 つのパラメータに結合したりすることは避けてください。各パラメータは、個別の懸念事項またはデータを表す必要があります。

12パラメータエイリアスを使用する: 長い名前または複雑な名前のパラメータがある場合は、読みやすさと保守性を高めるために、自動化ロジック内でエイリアスまたは短縮バージョンを使用することを検討してください。

これらのベストプラクティスに従うことで、自動化パラメータの設計、保守性、使いやすさを確保し、最終的に自動化の全体的な品質と効率を向上させることができます。

JavaScript を使用して App Studio で式を記述する

AWS App Studio では、JavaScript 式を使用して、アプリケーションの動作と外観を動的に制御できます。単一行の JavaScript 式は、二重中括弧内に記述され `{{ }}`、オートメーション、UI コンポーネント、データクエリなどのさまざまなコンテキストで使用できます。これらの式は実行時に評価され、計算の実行、データの操作、アプリケーションロジックの制御に使用できます。

App Studio は、Luxon、UUID、Lodash、および SDK 統合の 3 つの JavaScript オープンソースライブラリをネイティブにサポートし、アプリの設定内の JavaScript 構文と型チェックエラーを検出します。

Important

App Studio は、サードパーティーまたはカスタム JavaScript ライブラリの使用をサポートしていません。

基本構文

JavaScript 式には、変数、リテラル、演算子、関数呼び出しを含めることができます。式は、計算の実行や条件の評価によく使用されます。

以下の例を参照してください。

- `{{ 2 + 3 }}` は 5 と評価されます。
- `{{ "Hello, " + "World!" }}` は「Hello, World!」と評価されます。
- `{{ Math.max(5, 10) }}` は 10 に評価されます。
- `{{ Math.random() * 10 }}` は、[0 ~ 10) の乱数 (小数点以下) を返します。

Interpolation

JavaScript を使用して、静的テキスト内の動的値を補間することもできます。これは、次の例のように、JavaScript 式を二重中括弧で囲むことで実現されます。

```
Hello {{ currentUser.firstName }}, welcome to App Studio!
```

この例では、`currentUser.firstName`は現在のユーザーの名を取得する JavaScript 式で、挨拶メッセージに動的に挿入されます。

連結

次の例のように、JavaScript の `+`演算子を使用して文字列と変数を連結できます。

```
{{ currentRow.FirstName + " " + currentRow.LastName }}
```

この式は、`currentRow.FirstName`と `currentRow.LastName`の値をその間のスペースと組み合わせ、現在の行のフルネームになります。たとえば、`currentRow.FirstName`が `John`で、`currentRow.LastName`が `Doe`の場合、式は `John Doe` に解決されます。

日付および時間

JavaScript には、日付と時刻を操作するためのさまざまな関数とオブジェクトが用意されています。例えば、次のようになります。

- `{{ new Date().toLocaleDateString() }}`: ローカライズされた形式で現在の日付を返します。
- `{{ DateTime.now().toISODate() }}`: Date コンポーネントで使用する現在の日付を YYYY-MM-DD 形式で返します。

日付と時刻の比較

`=`、`>`、`<`、などの演算子を使用して `>=`、日付または時刻の値を比較 `<=` します。例えば、次のようになります。

- `{{ui.timeInput.value > "10:00 AM"}}`: 時刻が午前 10 時より後かどうかを確認します。
- `{{ui.timeInput.value <= "5:00 PM"}}`: 時刻が午後 5 時以前かどうかを確認します。

- `{{ui.timeInput.value > DateTime.now().toISOTime()}}`: 時刻が現在の時刻より後かどうかを確認します。
- `{{ui.dateInput.value > DateTime.now().toISODate()}}`: 日付が現在の日付より前かどうかを確認します。
- `{{ DateTime.fromISO(ui.dateInput.value).diff(DateTime.now(), "days").days >= 5 }}`: 日付が現在の日付から 5 日以上経過しているかどうかを確認します。

コードブロック

式に加えて、複数行の JavaScript コードブロックを記述することもできます。式とは異なり、コードブロックには中括弧は必要ありません。代わりに、JavaScript コードをコードブロックエディタ内で直接記述できます。

Note

式が評価され、その値が表示される間、コードブロックが実行され、出力 (存在する場合) が表示されます。

グローバル変数と関数

App Studio は、JavaScript 式とコードブロックで使用できる特定のグローバル変数と関数へのアクセスを提供します。たとえば、`currentUser` は、現在ログインしているユーザーを表すグローバル変数であり、`currentUser.role` のプロパティにアクセスしてユーザーのロールを取得できます。

UI コンポーネント値を参照または更新する

コンポーネントとオートメーションアクションで式を使用して、UI コンポーネント値を参照および更新できます。コンポーネント値をプログラムで参照して更新することで、ユーザーの入力とデータの変更に対応する動的でインタラクティブなユーザーインターフェイスを作成できます。

UI コンポーネント値を参照する

UI コンポーネントから値にアクセスして動的な動作を実装することで、インタラクティブでデータ駆動型のアプリケーションを作成できます。

式の名前ui空間を使用して、同じページで UI コンポーネントの値とプロパティにアクセスできます。コンポーネントの名前を参照することで、その値を取得したり、その状態に基づいてオペレーションを実行したりできます。

Note

コンポーネントはそれぞれのページにスコープされているため、ui名前空間には現在のページにのみコンポーネントが表示されます。

App Studio アプリのコンポーネントを参照するための基本的な構文は、`ui.componentName` です。

次のリストには、ui名前空間を使用して UI コンポーネント値にアクセスする例が含まれています。

- `ui.textInputName.value`: `textInputName` という名前のテキスト入力コンポーネントの値を表します。
- `ui.formName.isValid`: `formName` という名前のフォームのすべてのフィールドが、指定した検証基準に基づいて有効かどうかを確認します。
- `ui.tableName.currentRow.columnName`: `tableName` という名前のテーブルコンポーネントの現在の行にある特定の列の値を表します。
- `ui.tableName.selectedRowData.fieldName`: `tableName` という名前のテーブルコンポーネントの選択した行から指定されたフィールドの値を表します。その後、ID (`ui.tableName.selectedRowData.ID`) などのフィールド名を追加して、選択した行からそのフィールドの値を参照できます。

次のリストには、コンポーネント値を参照するより具体的な例が含まれています。

- `ui.inputText1.value.trim().length > 0`: `inputText1` コンポーネントの値に空でない文字列があるかどうかを確認します。これは、ユーザー入力の検証や、入力テキストフィールドの値に基づく他のコンポーネントの有効化/無効化に役立ちます。
- `ui.multiSelect1.value.join(", ")`: `multiSelect1` という名前の複数選択コンポーネントの場合、この式は選択したオプション値の配列をカンマ区切りの文字列に変換します。これは、選択したオプションをユーザーフレンドリーな形式で表示したり、選択を別のコンポーネントやオートメーションに渡すのに役立ちます。

- `{{ui.multiSelect1.value.includes("option1")}}`: この式は、値 `option1` が `multiSelect1` コンポーネントの選択したオプションの配列に含まれているかどうかを確認します。`option1` が選択されている場合は `true` を返し、それ以外の場合は `false` を返します。これは、条件付きでコンポーネントをレンダリングしたり、特定のオプション選択に基づいてアクションを実行したりするのに役立ちます。
- `{{ui.s3Upload1.files.length > 0}}`: `s3Upload1` という名前の Amazon S3 ファイルアップロードコンポーネントの場合、この式は `files` 配列の長さをチェックして、ファイルがアップロードされたかどうかを確認します。これは、ファイルがアップロードされたかどうかに基づいて、他のコンポーネントやアクションを有効または無効にする場合に便利です。
- `{{ui.s3Upload1.files.filter(file => file.type === "image/png").length}}`: この式は、`s3Upload1` コンポーネントにアップロードされたファイルのリストをフィルタリングして PNG イメージファイルのみを含め、それらのファイルの数を返します。これは、アップロードされたファイルの種類に関する情報を検証または表示するのに役立ちます。

UI コンポーネント値の更新

コンポーネントの値を更新または操作するには、オートメーション `RunComponentAction` 内で使用します。以下は、`RunComponentAction` アクションを使用して `myInput` という名前のテキスト入力コンポーネントの値を更新するために使用できる構文の例です。

```
RunComponentAction(ui.myInput, "setValue", "New Value")
```

この例では、`RunComponentAction` ステップは `myInput` コンポーネントの `setValue` アクションを呼び出し、新しい値 `New Value` を渡します。

テーブルデータの使用

テーブルデータと値にアクセスして、オペレーションを実行できます。次の式を使用して、テーブルデータにアクセスできます。

- `currentRow`: テーブル内の現在の行からテーブルデータにアクセスするために使用します。たとえば、テーブルアクションの名前の設定、行からアクションから開始されたオートメーションへの値の送信、テーブル内の既存の列の値を使用した新しい列の作成などです。
- `ui.tableName.selectedRow` と `ui.tableName.selectedRowData` はどちらも、ページの他のコンポーネントからテーブルデータにアクセスするために使用されます。例えば、選択した行に基づいてテーブルの外部にボタンの名前を設定します。返される値は同じですが、`selectedRow` との違い `selectedRowData` は次のとおりです。

- `selectedRow`: この名前空間には、各フィールドの列ヘッダーに表示される名前が含まれます。テーブル内の表示可能な列から値を参照する場合は、`selectedRow`を使用する必要があります。たとえば、テーブルにカスタム列または計算列があり、エンティティのフィールドとして存在しない場合です。
- `selectedRowData`: この名前空間には、テーブルのソースとして使用されるエンティティのフィールドが含まれます。`selectedRowData`を使用して、テーブルに表示されないエンティティの値を参照する必要がありますが、アプリの他のコンポーネントやオートメーションに役立ちます。

次のリストには、式内のテーブルデータにアクセスする例が含まれています。

- `{{ui.tableName.selectedRow.columnNameWithNoSpace}}`: テーブル内の選択した行から `columnNameWithNoSpace` 列の値を返します。
- `{{ui.tableName.selectedRow['Column Name With Space']}}`: テーブル内の選択した行の#####の列の値を返します。
- `{{ui.tableName.selectedRowData.fieldName}}`: テーブル内の選択した行から `fieldName` エンティティフィールドの値を返します。
- `{{ui.tableName.selectedRows[0].columnMappingName}}`: 同じページの他のコンポーネントまたは式から選択した行の列名を参照します。
- `{{currentRow.firstName + ' ' + currentRow.lastNamecolumnMapping}}`: 複数の列の値を連結して、テーブルに新しい列を作成します。
- `{{ { "Blocked": "#", "Delayed": "#", "On track": "#" } [currentRow.statuscolumnMapping] + " " + currentRow.statuscolumnMapping}}`: 保存されたステータス値に基づいて、テーブル内のフィールドの表示値をカスタマイズします。
- `{{currentRow.colName}}`、`{{currentRow["First Name"]}}`、`{{currentRow}}`、または `{{ui.tableName.selectedRows[0]}}`: 参照された行のコンテキストを行アクションに渡します。

オートメーションへのアクセス

オートメーションを使用して、App Studio でサーバー側のロジックとオペレーションを実行できます。自動化アクション内では、式を使用してデータを処理し、動的値を生成し、以前のアクションの結果を組み込むことができます。

オートメーションパラメータへのアクセス

UI コンポーネントやその他のオートメーションからオートメーションに動的な値を渡すことができるため、再利用可能で柔軟性があります。これは、次のように `params` 名前空間のオートメーションパラメータを使用して行われます。

`{{params.parameterName}}`: UI コンポーネントまたは他のソースからオートメーションに渡される値を参照します。たとえば、は `ID` という名前のパラメータを参照 `{{params.ID}}` します。

オートメーションパラメータの操作

JavaScript を使用してオートメーションパラメータを操作できます。以下の例を参照してください。

- `{{params.firstName}} {{params.lastName}}`: パラメータとして渡された値を連結します。
- `{{params.numberParam1 + params.numberParam2}}`: 2 つの数値パラメータを追加します。
- `{{params.valueProvided?.length > 0 ? params.valueProvided : 'Default'}}`: パラメータが `null` または未定義ではなく、長さがゼロでないかどうかを確認します。true の場合は、指定された値を使用します。それ以外の場合は、デフォルト値を設定します。
- `{{params.rootCause || "No root cause provided"}}`: `params.rootCause` パラメータが `false` (`null`、`undefined`、または空の文字列) の場合、指定されたデフォルト値を使用します。
- `{{Math.min(params.numberOfProducts, 100)}}`: パラメータの値を最大値 (この場合は 100) に制限します。
- `{{ DateTime.fromISO(params.startDate).plus({ days: 7 }).toISO() }}`: `params.startDate` パラメータが の場合 "2023-06-15T10:30:00.000Z"、この式は開始日の 1 週間後の日付 "2023-06-22T10:30:00.000Z" である に評価されます。

前のアクションからのオートメーション結果へのアクセス

自動化により、アプリケーションはデータベースのクエリ、APIs とのやり取り、データ変換の実行など、サーバー側のロジックとオペレーションを実行できます。results 名前空間は、同じオートメーション内の以前のアクションによって返された出力とデータへのアクセスを提供します。自動化結果へのアクセスに関する以下の点に注意してください。

1. 前のオートメーションステップの結果には、同じオートメーション内でのみアクセスできます。

2. *action1* と *action2* という名前のアクションがその順序で存在する場合、*action1* は結果を参照できず、*action2* は `results.action1` にのみアクセスできます。
3. これはクライアント側のアクションでも機能します。たとえば、`InvokeAutomation` アクションを使用してオートメーションをトリガーするボタンがある場合です。その後、オートメーションによってファイルが PDF であることが示された場合 `results.myInvokeAutomation1.fileType === "pdf"`、PDF ビューワーのあるページに移動するなどの Run If 条件を含むナビゲーションステップを使用できます。

次のリストには、`results` 名前空間を使用して前のアクションの自動化結果にアクセスするための構文が含まれています。

- `{{results.stepName.data}}`: *stepName* という名前のオートメーションステップからデータ配列を取得します。
- `{{results.stepName.output}}`: *stepName* という名前のオートメーションステップの出力を取得します。

自動化ステップの結果にアクセスする方法は、アクションのタイプと返されるデータによって異なります。異なるアクションは、異なるプロパティまたはデータ構造を返す場合があります。以下は一般的な例です。

- データアクションの場合、`results.stepName.data` を使用して返されたデータ配列にアクセスできます。
- API コールアクションの場合、`results.stepName.body` を使用してレスポンス本文にアクセスできます。
- Amazon S3 アクションの場合、`results.stepName.Body.transformToWebStream()` を使用してファイルコンテンツにアクセスできます。

返されるデータの形状と `results`、名前空間内でデータにアクセスする方法については、使用している特定のアクションタイプのドキュメントを参照してください。次のリストには、いくつかの例が含まれています。

- `{{results.getDataStep.data.filter(row => row.status === "pending").length}}`: *getDataStep* がデータ行の配列を返す Invoke Data Action 自動化アクションであると仮定すると、この式はデータ配列をフィルタリングして、ステータスフィールドが `pending` と等しい行のみを含め、フィルタリングされた配列の長さ (カウント) を返します。これは、特定の条件に基づいてデータをクエリまたは処理する場合に便利です。

- `{{params.email.split("@")[0]}}`: `params.email` パラメータに E メールアドレスが含まれている場合、この式は @ 記号で文字列を分割し、@ 記号の前に部分を返します。これにより、E メールアドレスのユーザー名部分が効果的に抽出されます。
- `{{new Date(params.timestamp * 1000)}}`: この式は Unix タイムスタンプパラメータ (`params.timestamp`) を受け取り、JavaScript Date オブジェクトに変換します。タイムスタンプは秒単位であることを前提としているため、1000 を掛けてミリ秒に変換します。これは `new Date` ストラクチャが想定する形式です。これは、オートメーションで日付と時刻の値を操作するのに役立ちます。
- `{{results.stepName.Body}}`: `stepName` という名前の Amazon S3 GetObject オートメーションアクションの場合、この式はファイルコンテンツを取得し、取得されたファイルを表示するために Image や PDF Viewer などの UI コンポーネントで使用できます。この式は、コンポーネントで使用するオートメーションの Automation 出力で設定する必要があることに注意してください。

データの依存関係とタイミングに関する考慮事項

App Studio で複雑なアプリケーションを構築する場合、フォーム、詳細ビュー、オートメーションを活用したコンポーネントなど、さまざまなデータコンポーネント間のデータ依存関係を理解し、管理することが重要です。データコンポーネントとオートメーションは、データの取得または実行を同時に完了しない可能性があるため、タイミングの問題、エラー、予期しない動作が発生する可能性があります。潜在的なタイミングの問題を認識し、ベストプラクティスに従うことで、App Studio アプリケーションでより信頼性が高く一貫性のあるユーザーエクスペリエンスを作成できます。

潜在的な問題には、次のようなものがあります。

1. レンダリングタイミングの競合: データコンポーネントは、データの依存関係と一致しない順序でレンダリングされ、視覚的な不整合やエラーが発生する可能性があります。
2. 自動化実行のタイミング: 自動化タスクは、コンポーネントが完全にロードされる前に完了し、ランタイム実行エラーが発生する可能性があります。
3. コンポーネントのクラッシュ: オートメーションを搭載したコンポーネントは、無効なレスポンスやオートメーションの実行が完了していない場合にクラッシュする可能性があります。

例: 注文の詳細と顧客情報

この例では、データコンポーネント間の依存関係がどのようにタイミングの問題やデータ表示の潜在的なエラーにつながるかを示します。

同じページに次の 2 つのデータコンポーネントがあるアプリケーションを考えてみましょう。

- 注文データを取得する詳細コンポーネント (orderDetails)。
- 注文に関連する顧客の詳細を表示する詳細コンポーネント (customerDetails)。

このアプリケーションでは、orderDetails 詳細コンポーネントに次の値で設定された 2 つのフィールドがあります。

```
// 2 text fields within the orderDetails detail component

// Info from orderDetails Component
{{ui.orderDetails.data[0].name}}

// Info from customerDetails component
{{ui.customerDetails.data[0].name}} // Problematic reference
```

この例では、orderDetails コンポーネントは customerDetails コンポーネントからのデータを参照して顧客名を表示しようとしています。これは問題です。orderDetails コンポーネントがデータを取得する前に customerDetails コンポーネントがレンダリングされる可能性があるためです。customerDetails コンポーネントデータの取得が遅延または失敗した場合、orderDetails コンポーネントは不完全または誤った情報を表示します。

データ依存関係とタイミングのベストプラクティス

次のベストプラクティスを使用して、App Studio アプリのデータ依存関係とタイミングの問題を軽減します。

1. 条件付きレンダリングを使用する: コンポーネントをレンダリングするか、データが利用可能であることを確認した場合にのみデータを表示します。条件ステートメントを使用して、データを表示する前にデータの存在を確認します。次のスニペットは、条件ステートメントの例を示しています。

```
{{ui.someComponent.data ? ui.someComponent.data.fieldName : "Loading..."}}
```

2. 子コンポーネントの可視性の管理: データがロードされる前に子をレンダリングする
Stepflow、Form、Detail などのコンポーネントについては、子コンポーネントの可視性を手動で設定します。次のスニペットは、親コンポーネントのデータの可用性に基づいて可視性を設定する例を示しています。

```
{{ui.parentComponent.data ? true : false}}
```

3. 結合クエリを使用する: 可能であれば、結合クエリを使用して 1 つのクエリで関連データを取得します。これにより、個別のデータフェッチの数が減少し、データコンポーネント間のタイミングの問題が最小限に抑えられます。
4. オートメーションにエラー処理を実装する: オートメーションに堅牢なエラー処理を実装して、予想されるデータが利用できない、または無効なレスポンスが受信されるシナリオを適切に管理します。
5. オプションの連鎖を使用する: ネストされたプロパティにアクセスするときは、オプションの連鎖を使用して、親プロパティが定義されていない場合のエラーを防止します。次のスニペットは、オプションの連鎖の例を示しています。

```
{{ui.component.data?.[0]?.fieldSystemName}}
```

複数のユーザーによるアプリの構築

複数のユーザーが 1 つの App Studio アプリで作業できますが、一度に 1 つのアプリを編集できるのは 1 人のユーザーのみです。他のユーザーを招待してアプリを編集すること、および複数のユーザーが同時にアプリを編集しようとする場合の動作については、以下のセクションを参照してください。

ビルダーを招待してアプリを編集する

次の手順を使用して、他のビルダーを招待して App Studio アプリを編集します。

他のビルダーを招待してアプリを編集するには

1. 必要に応じて、アプリケーションのアプリケーションスタジオに移動します。
2. [共有] を選択します。
3. 開発タブで、テキストボックスを使用して、アプリの編集に招待するグループまたは個々のユーザーを検索して選択します。
4. ユーザーまたはグループごとにドロップダウンを選択し、そのユーザーまたはグループに付与するアクセス許可を選択します。
 - 共同所有者: 共同所有者には、アプリ所有者と同じアクセス許可があります。

- 編集のみ: 編集のみのロールを持つユーザーは、以下を除き、所有者および共同所有者と同じアクセス許可を持ちます。
- 他のユーザーを招待してアプリを編集することはできません。
- テスト環境または本番稼働環境にアプリを公開することはできません。
- アプリにデータソースを追加することはできません。
- アプリを削除または複製することはできません。

他のユーザーが編集中のアプリの編集を試みる

1 つの App Studio アプリは、一度に 1 人のユーザーのみが編集できます。複数のユーザーが同時にアプリを編集しようとするとうなるかについては、次の例を参照してください。

この例では、User A は現在アプリを編集しており、はそれをと共有しています User B。User B は、によって編集されているアプリの編集を試みます User A。

アプリを編集 User B しようとする、現在アプリを編集 User A していることを示すダイアログボックスが表示され、継続するとアプリケーションスタジオ User A から開始され、すべての変更が保存されます。User B はキャンセルして User A 続行するか、続行してアプリケーションスタジオに入り、アプリを編集できます。この例では、アプリの編集を選択します。

がアプリの編集 User B を選択すると、User B はアプリの編集を開始した通知 User A を受け取り、セッションは終了します。アプリ User A が非アクティブなブラウザタブで開いた場合、通知を受信しない可能性があることに注意してください。この場合、アプリに戻って編集しようとする、エラーメッセージが表示され、ページを更新するように指示され、アプリケーションのリストに戻ります。

アプリのコンテンツセキュリティ設定の表示または更新

App Studio のすべてのアプリケーションには、イメージ、iFramesPDFs などの外部メディアやリソースのロードを制限したり、指定されたドメインや URLs (Amazon S3 バケットを含む) からのみ許可したりするために使用できるコンテンツセキュリティ設定があります。アプリが Amazon S3 にオブジェクトをアップロードできるドメインを指定することもできます。

すべてのアプリケーションのデフォルトのコンテンツセキュリティ設定は、Amazon S3 バケットを含む外部ソースからのすべてのメディアのロードをブロックし、Amazon S3 へのオブジェクトのアップロードをブロックすることです。したがって、イメージ、iFramesPDFs、または同様のメ

メディアをロードするには、メディアのソースを許可するように設定を編集する必要があります。また、Amazon S3 へのオブジェクトのアップロードを許可するには、[がアップロードできるドメインを許可する設定を編集する必要があります。](#)

Note

コンテンツセキュリティ設定は、アプリケーションでコンテンツセキュリティポリシー (CSP) ヘッダーを設定するために使用されます。CSP は、クロスサイトスクリプティング (XSS)、クリックジャック、その他のコードインジェクション攻撃からアプリを保護するのに役立つセキュリティ標準です。CSP の詳細については、MDN ウェブドキュメントの「[コンテンツセキュリティポリシー \(CSP\)](#)」を参照してください。

アプリのコンテンツセキュリティ設定を更新するには

1. 必要に応じて、アプリケーションリストから編集を選択して、アプリケーションのアプリケーションスタジオに移動します。
2. アプリ設定を選択します。
3. コンテンツセキュリティ設定タブを選択すると、次の設定が表示されます。
 - フレームソース: アプリがフレームや iframe (インタラクティブコンテンツや PDFs など) をロードできるドメインを管理するために使用されます。この設定は、以下のコンポーネントまたはアプリケーションリソースに影響します。
 - iFrame 埋め込みコンポーネント
 - PDF ビューワーコンポーネント
 - イメージソース: アプリがイメージをロードできるドメインを管理するために使用されます。この設定は、以下のコンポーネントまたはアプリケーションリソースに影響します。
 - アプリのロゴとバナー
 - イメージビューワーコンポーネント
 - Connect source: アプリが Amazon S3 オブジェクトをアップロードできるドメインを管理するために使用されます。
4. 設定ごとに、ドロップダウンから目的の設定を選択します。
 - すべてのframes/images/connectionsブロックする: メディア (イメージ、フレーム、PDFs) をロードしたり、オブジェクトを Amazon S3 にアップロードしたりしないでください。

- すべてのframes/images/connectionsを許可する: すべてのドメインのすべてのメディア (イメージ、フレーム、PDFs) のロードを許可するか、すべてのドメインの Amazon S3 へのオブジェクトのアップロードを許可します。
- 特定のドメインを許可する: 指定したドメインへのメディアのロードまたはアップロードを許可します。ドメインまたは URLs は、式のスペース区切りリストとして指定されます。ここで、ワイルドカード (*) をサブドメイン、ホストアドレス、またはポート番号に使用して、それぞれのすべての法的値が有効であることを示すことができます。の指定httpも と一致しますhttps。次のリストには、有効なエントリの例が含まれています。
- blob:: Amazon S3 バケットから項目GetObjectを返す、Amazon Bedrock によって生成されたイメージなど、自動化アクションによって返されるファイルデータを含むすべてのBLOB に一致します。

 Important

指定された式blob:に を含めて、アクションによって返されるファイルデータを許可する必要があります。式が であっても*、 に更新する必要があります。 * blob:

- http://*.example.com: の任意のサブドメインからロードするすべての試行に一致しますexample.com。 https リソースとも一致します。
 - https://source1.example.com https://source2.example.com: https://source1.example.comと の両方からロードするすべての試行に一致 https://source2.example.com
 - https://example.com/subdirectory/: サブディレクトリディレクトリにファイルをロードするすべての試行に一致します。例えば、https://example.com/subdirectory/path/to/file.jpeg。これは と一致しませんhttps://example.com/path/to/file.jpeg。
5. [保存] を選択して変更を保存します。

App Studio のトラブルシューティングとデバッグ

トピック

- [App Studio のセットアップ、アクセス許可、オンボーディングのトラブルシューティング](#)
- [アプリケーションのトラブルシューティングとデバッグ](#)
- [アプリケーションの公開と共有のトラブルシューティング](#)

App Studio のセットアップ、アクセス許可、オンボーディングのトラブルシューティング

このトピックでは、App Studio をセットアップまたはオンボーディングする際の一般的な問題のトラブルシューティングと、アクセス許可の管理について説明します。

Create an account instance for me オプションを選択したときに App Studio の設定に失敗しました

問題: アカウントインスタンスの作成を使用して App Studio を設定すると、IAM Identity Center が 1 つのインスタンスのみをサポートしているため、任意の AWS リージョンにアカウントレベルの IAM Identity Center インスタンスがある場合、失敗します。

解決策: <https://console.aws.amazon.com/singlesignon/> の IAM Identity Center コンソールに移動して、IAM Identity Center インスタンスがあるかどうかを確認します。インスタンスが見つかるまで、サポートされているすべての AWS リージョンを確認します。App Studio をセットアップするときにそのインスタンスを使用するか、IAM Identity Center インスタンスを削除して、Create an account instance for me オプションで再試行できます。

Warning

IAM Identity Center インスタンスを削除すると、既存のユースケースに影響します。削除する前にインスタンスが使用されていないことを確認するか、インスタンスを使用して App Studio をセットアップします。

設定後に App Studio にアクセスできない

問題: App Studio を設定するときに、自分がメンバーではない IAM Identity Center グループを指定している可能性があります。App Studio にアクセスするには、少なくとも 1 つのグループのメンバーである必要があります。

解決策: <https://console.aws.amazon.com/singlesignon/> で IAM Identity Center コンソールに移動し、セットアップ時に App Studio に追加されたグループに自分自身を追加します。

App Studio にログインするときに使用するユーザー名またはパスワードがわからない

問題: IAM アイデンティティセンターの認証情報を設定していないか、IAM アイデンティティセンターのユーザー名またはパスワードを忘れていたため、App Studio にログインする方法がわからない場合があります。

解決策: IAM Identity Center インスタンスを使用せずに App Studio を設定する場合、IAM Identity Center ユーザーの作成に使用される E メールとユーザー名がユーザーごとに指定されました。提供された各 E メールアドレスには、IAM Identity Center への招待が記載された E メールが送信されました。各ユーザーは招待を受け入れ、IAM Identity Center ユーザー認証情報のパスワードを作成する必要があります。その後、各ユーザーは IAM Identity Center のユーザー名とパスワードを使用して App Studio にログインできます。

認証情報をすでに設定していて、ユーザー名またはパスワードを忘れた場合は、IAM Identity Center コンソールを使用してユーザー名を表示および入力するか、パスワードをリセットするように管理者に依頼する必要があります。

App Studio の設定時にシステムエラーが表示される

問題: App Studio の設定時に次のエラーが表示されます。

```
System error. We encountered a problem. Report the issue and the App Studio service team will get back to you.
```

このエラーは、サービスで不明なエラーが発生した場合に発生します。

解決策: 左側のナビゲーションの「学習」セクションで「Slack で参加」を選択するか、アプリの編集集中に上部のバナーを選択して、コミュニティ Slack に参加してサポートチームに連絡してください。

App Studio インスタンス URL が見つかりません

App Studio インスタンスにアクセスするための URL が見つからない場合は、App Studio をセットアップする管理者にお問い合わせください。管理者は、 の App Studio コンソールで URL を表示できます AWS マネジメントコンソール。

App Studio でグループまたはロールを変更できない

問題: 左側のナビゲーションにロールリンクが表示されません。これは、管理者ロールを持つユーザーのみが App Studio のグループとロールを変更できるためです。

解決策: 管理者ロールを持つユーザーに連絡してグループまたはロールを変更するか、管理者に連絡して管理者グループに追加してもらいます。

App Studio からオフボードする方法

現時点では、App Studio からオフボードすることはできません。アプリケーションやコネクタなどのすべてのリソースを削除し、アクセスや使用を防ぐためにグループのロールを App User に変更することをお勧めします。また、IAM ロールやデータベーステーブルなど、App Studio 専用に使われるサードパーティリソースも削除する必要があります。

アプリケーションのトラブルシューティングとデバッグ

以下のトピックでは、App Studio アプリケーションのトラブルシューティングとデバッグについて説明します。

トピック

- [AI ビルダーアシスタントとチャットのトラブルシューティング](#)
- [アプリケーションスタジオでのトラブルシューティング](#)
- [プレビューアプリケーションのトラブルシューティング](#)
- [テスト環境でのトラブルシューティング](#)
- [Amazon CloudWatch Logs で公開されたアプリケーションからのログを使用したデバッグ](#)
- [コネクタのトラブルシューティング](#)

AI ビルダーアシスタントとチャットのトラブルシューティング

このトピックには、AI ビルダーアシスタントを使用する際の一般的な問題のトラブルシューティングガイドが含まれています。

AI でアプリケーションを作成するときのエラー

AI プロンプトを使用してアプリケーションを作成すると、次のエラーが発生する可能性があります。

```
We apologize, but we cannot proceed with your request. The request may contain content that violates our policies and guidelines. Please revise your prompt before trying again.
```

問題: 潜在的に有害なコンテンツのため、リクエストがブロックされます。

解決策: プロンプトを言い換えて、もう一度試してください。

AI を使用して生成されたアプリが空であるか、コンポーネントがありません。

問題: これは、予期しないサービスエラーが原因である可能性があります。

解決策: AI を使用してアプリの作成を再試行するか、生成されたアプリでコンポーネントを手動で作成します。

アプリケーションスタジオでのトラブルシューティング

このトピックには、アプリケーションを構築する際の問題のトラブルシューティングとデバッグのガイドが含まれています。

デバッグパネルの使用

アプリケーションの構築中にライブデバッグをサポートするために、App Studio には、アプリケーションスタジオのページ、オートメーション、データタブにまたがる折りたたみ可能なビルダーデバッグパネルが用意されています。このパネルには、エラーと警告の両方が表示されます。警告は、設定されていないリソースなど、実用的な提案として機能しますが、エラーを解決してアプリを正常にプレビューまたは公開する必要があります。各エラーまたは警告には、問題の場所に移動するために使用できるビューリンクが含まれています。

デバッグパネルは、新しいエラーまたは警告が発生すると自動的に更新され、エラーまたは警告は解決されると自動的に消えます。これらの警告メッセージとエラーメッセージの状態は、ビルダーを離れても保持されます。

JavaScript 式の構文とデータ型処理

App Studio には JavaScript エラー検出機能があり、コードに赤い線を付け、エラーを強調表示します。これらのコンパイルエラーは、アプリが正常に構築されないようにし、入力ミス、無効な参照、無効なオペレーション、必要なデータ型の誤った出力などの問題を示します。一般的な問題については、次のリストを参照してください。

1. リソースの名前変更によるエラー: App Studio で JavaScript 式がリソース名を参照する場合、それらの名前を変更すると式が正しくなくなり、エラーが発生します。これらのエラーは、デバッグパネルで表示できます。
2. データ型の問題: データ型の不一致により、アプリにエラーが発生します。たとえば、オートメーションが型のパラメータを受け入れるように設定されていてもString、コンポーネントが型の値を送信するように設定されている場合Integer、エラーが発生します。データ型が、コンポーネント、オートメーション、データエンティティ、アクションなど、適切なリソース間で一致していることを確認します。JavaScript 式の値のタイプを変更する必要がある場合があります。

プレビューアプリケーションのトラブルシューティング

このトピックには、アプリケーションをプレビューする際の問題のトラブルシューティングに関する情報が含まれています。

プレビューは次のエラーでロードに失敗します。 **Your app failed to build and cannot be previewed**

問題: プレビューするには、アプリが正常にビルドされている必要があります。このエラーは、アプリが正常に構築できないコンパイルエラーがある場合に発生します。

解決策: アプリケーションスタジオのデバッグパネルを使用してエラーを確認して解決します。

プレビューのロードに時間がかかる

問題: 特定のタイプのアプリケーション更新では、コンパイルと構築に長い時間がかかります。

解決策: タブを開いたままにして、更新が構築されるまで待ちます。アプリのアプリケーションスタジオの右上に保存されたと表示され、プレビューが再ロードされます。

プレビューには最新の変更が反映されていない

問題: これは、アプリ編集セッションが別のユーザーによって引き継がれたが、通知されなかった場合に発生する可能性があります。これにより、アプリが編集されてプレビュー環境と一致しない可能性があります。

解決策: Application Studio ブラウザタブを更新し、必要に応じて編集セッションを引き継ぎます。

テスト環境でのトラブルシューティング

このトピックには、テスト環境に公開されたアプリケーションのトラブルシューティングに関する情報が含まれています。

Note

オートメーションまたはデータアクションからの HTTP 500 レスポンスは、式でのランタイムクラッシュ、コネクタの障害、またはアプリケーションに接続されているデータソースからのスロットリングが原因で発生する可能性があります。の手順に従って、基盤となるエラーの詳細を示すデバッグログ [ブラウザコンソールを使用したデバッグ](#) を表示します。

デバッグパネルの使用

App Studio は、アプリケーションの構築時に使用するビルドデバッグパネルと同様に、テスト環境で折りたたみ可能なデバッグパネルを提供します。このパネルには、ページのロード時間、ユーザーナビゲーション、アプリイベントなどの情報メッセージが表示されます。また、エラーと警告も含まれています。デバッグパネルは、イベントが発生すると新しいメッセージで自動的に更新されます。

ブラウザコンソールを使用したデバッグ

アプリのプレビュー中にアクションが呼び出されないため、呼び出しとレスポンスの処理をテストするために、アプリをテスト環境に公開する必要があります。オートメーションの実行中にエラーが発生した場合、またはアプリケーションが特定の動作をする理由を把握したい場合は、ブラウザのコンソールを使用してリアルタイムのデバッグを行うことができます。

ブラウザコンソールを使用してテスト環境でアプリケーションをデバッグするには

1. URL の末尾?debug=trueに追加し、Enter キーを押します。URL に既にクエリ文字列 (を含む?) がある場合は、代わりに URL の末尾&debug=trueに を追加します。

2. ブラウザコンソールを開き、アクションまたは API の入力と出力を調べてデバッグを開始します。
 - Chrome で: ブラウザで右クリックし、Inspect を選択します。Chrome DevTools でのデバッグの詳細については、[Chrome DevTools ドキュメント](#)を参照してください。
 - Firefox の場合: ウェブページ要素を長押しまたは右クリックし、要素の検査を選択します。Firefox DevTools でのデバッグの詳細については、[Firefox DevTools ユーザードキュメント](#)を参照してください。

次のリストには、エラーを生成する一般的な問題が含まれています。

- ランタイムエラー
 - 問題: オートメーションまたは式が正しく設定されていない場合、オートメーションの実行時にエラーが発生する可能性があります。一般的なエラーは、アセットの名前変更です。その結果、式が正しくなくなり、他の JavaScript コンパイルエラーが発生したり、 のデータやアセットを使用したりしますundefined。
 - 解決策: カスタムコード入力 (式、JavaScript、JSON) の各使用状況を確認し、コードエディタまたはデバッグパネルにコンパイルエラーがないことを確認します。
- コネクタの問題
 - 問題: App Studio アプリは、公開されるまでコネクタと外部サービスと通信しないため、プレビュー中に発生しなかったテスト環境でエラーが発生する可能性があります。コネクタを使用するオートメーションのアクションが失敗した場合、リクエストをコネクタに送信するアクションの設定ミス、またはコネクタ設定自体のアクションが原因である可能性があります。
 - 解決策: これらのエラーを防ぐために、モック出力を使用してプレビュー環境の早い段階でオートメーションをテストする必要があります。コネクタが正しく設定されていることを確認します。詳細については、「」を参照してください[コネクタのトラブルシューティング](#)。最後に、CloudWatch を使用してログを確認できます。詳細については、「[Amazon CloudWatch Logs で公開されたアプリケーションからのログを使用したデバッグ](#)」を参照してください。ConnectorService 名前空間ログには、コネクタから発信されたエラーメッセージまたはメタデータがあるはずです。

Amazon CloudWatch Logs で公開されたアプリケーションからのログを使用したデバッグ

Amazon CloudWatch Logs は、AWS リソースと で AWS 実行されるアプリケーションをリアルタイムでモニタリングします。CloudWatch Logs を使用して、リソースとアプリケーションに対して測定できる変数であるメトリクスを収集および追跡できます。

App Studio アプリのデバッグでは、CloudWatch Logs は、アプリの実行中に発生するエラーの追跡、情報の監査、ユーザーアクションと独自のインタラクションに関するコンテキストの提供に役立ちます。ログには履歴データが用意されており、これを使用してアプリケーションの使用状況とアクセスパターンを監査したり、ユーザーが遭遇したエラーを確認したりできます。

Note

CloudWatch Logs は、アプリケーションの UI から渡されたパラメータ値のリアルタイムトレースを提供しません。

CloudWatch Logs の App Studio アプリからログにアクセスするには、次の手順に従います。

1. アプリの App Studio アプリケーションスタジオで、URL を参照してアプリ ID を見つけてメモします。アプリ ID は次のようになります: 802a3bd6-ed4d-424c-9f6b-405aa42a62c5。
2. CloudWatch コンソールの <https://console.aws.amazon.com/cloudwatch/> を開いてください。
3. ナビゲーションペインで、[Log groups] (ロググループ) を選択します。
4. ここでは、アプリケーションごとに 5 つのロググループがあります。関心のある情報のタイプに応じて、グループを選択し、検出するデータのクエリを書き込みます。

次のリストには、ロググループと、それぞれをいつ使用するかに関する情報が含まれています。

1. `/aws/appstudio/teamId/appId/TEST/app`: テスト環境に現在公開されているアプリのバージョンに関連する自動化レスポンス、コンポーネントエラー、または JavaScript コードをデバッグするために使用します。
2. `/aws/appstudio/teamId/appId/TEST/audit`: 条件付き可視性または変換、クエリの失敗、テスト環境に現在公開されているアプリのバージョンに関連するログインまたはアクセス許可ユーザーエラーなどの JavaScript コードエラーをデバッグするために使用します。
3. `/aws/appstudio/teamId/setup`: ビルダーまたは管理者アクションをモニタリングするために使用します。

4. `/aws/appstudio/teamId/appId/PRODUCTION/app`: を使用して、自動化レスポンス、クエリの失敗、コンポーネントエラー、または本番環境に現在公開されているアプリのバージョンに関連する JavaScript コードをデバッグします。
5. `/aws/appstudio/teamId/appId/PRODUCTION/audit`: 条件付き可視性や変換などの JavaScript コードエラー、および本番環境に現在公開されているアプリのバージョンに関連するログインまたはアクセス許可ユーザーエラーをデバッグするために使用します。

Note

デバッグに使用されるログのほとんどは、`DebugLogClient` 名前空間に分類されます。

5. ロググループに入ったら、最新のログストリームを選択するか、目的の時刻に最も近い最後のイベント時刻を持つログストリームを選択するか、すべてのログストリームを検索して、そのロググループのすべてのイベントを検索できます。CloudWatch Logs でのログデータの表示の詳細については、[CloudWatch Logs に送信されたログデータの表示](#) を参照してください。

CloudWatch Logs Insights クエリを使用したログのフィルタリングとソート

CloudWatch Logs Insights を使用して、複数のロググループを一度にクエリできます。セッション情報を含むロググループのリストを特定したら、CloudWatch Logs Insights に移動し、ロググループを選択します。次に、クエリをカスタマイズしてターゲットログエントリをさらに絞り込みます。サンプルクエリを次に示します。

キーワードを含むログのリスト: **error**

```
fields @timestamp, @message
| filter @message like 'error'
| sort @timestamp desc
```

テスト環境からログをデバッグします。

```
fields @timestamp, @message
| filter namespace = "DebugLogClient"
| sort @timestamp desc
```

5 分間隔での全体的な 504/404/500 エラー数:

```
filter @message like '/api/automation' and (@message like ': 404' or @message like ':
500' or @message like ': 504')
| fields @timestamp, method, path, statusCode
| stats count(*) as errorCount by bin(5m)
```

CloudWatch Logs Insights の詳細については、「Amazon [CloudWatch Logs ユーザーガイド](#)」の「[CloudWatch Logs Insights を使用したログデータの分析](#)」を参照してください。Amazon CloudWatch

コネクタのトラブルシューティング

このトピックには、一般的なコネクタの問題に関するトラブルシューティングガイダンスが含まれています。コネクタを表示または編集するには、管理者グループのメンバーである必要があります。

IAM ロールに正しいカスタム信頼ポリシーとタグがあることを確認します。

コネクタの IAM ロールを設定するときは、カスタム信頼ポリシーが App Studio へのアクセスを提供するように適切に設定されていることを確認します。このカスタム信頼ポリシーは、AWS リソースが App Studio のセットアップに使用したのと同じ AWS アカウントにある場合でも必要です。

- Principal セクションの AWS アカウント番号が、App Studio のセットアップに使用されるアカウントのアカウント AWS ID であることを確認します。このアカウント番号は、リソースが配置されているアカウントであるとは限りません。
- "aws:PrincipalTag/IsAppStudioAccessRole": "true" が sts:AssumeRole セクションに正しく追加されていることを確認します。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::111122223333:root"
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "StringEquals": {
```

```
        "aws:PrincipalTag/IsAppStudioAccessRole": "true"
    }
}
}
]
```

また、次のキーと値を持つタグが IAM ロールに追加されていることを確認します。タグの追加の詳細については、[「IAM ロールのタグ付け」](#)を参照してください。

Note

タグの値は `IsAppStudioDataAccessRole`、カスタム信頼ポリシー (`IsAppStudioAccessRole`) の値とは若干異なることに注意してください。

- [Key] (キー): `IsAppStudioDataAccessRole`
- 値: `true`

コネクタが接続されている製品またはサービス内のリソースの設定を確認します。Amazon Redshift テーブルなどの一部のリソースでは、App Studio で使用する追加の設定が必要です。

コネクタの設定を確認します。AWS サービスの場合は、App Studio のコネクタに移動し、正しい Amazon リソースネーム (ARN) が含まれ、指定された AWS リージョンがリソースを含むリージョンであることを確認します。

IAM ロールに正しいアクセス許可があることを確認します。

App Studio に AWS リソースへのアクセスを提供するには、コネクタで使用される IAM ロールに適切なアクセス許可を割り当てる必要があります。必要なアクセス許可は、実行するサービス、リソース、およびアクションに固有です。例えば、Amazon Redshift テーブルからデータを読み取るには、Amazon S3 バケットにオブジェクトをアップロードするアクセス許可とは異なるアクセス許可が必要です。詳細については、[AWS サービスに接続する](#)を参照してください。

Amazon Redshift コネクタのトラブルシューティング

このセクションでは、Amazon Redshift コネクタの一般的な問題のトラブルシューティングガイドンスについて説明します。Amazon Redshift コネクタとリソースの設定については、「[Amazon Redshift に接続する](#)」を参照してください。

1. Amazon Redshift エディタOFFでIsolated Sessionトグルが に設定されていることを確認します。この設定は、App Studio アプリなどの他のユーザーによるデータ変更を可視化するために必要です。
2. Amazon Redshift テーブルに適切なアクセス許可が付与されていることを確認します。
3. コネクタ設定で、Amazon Redshift テーブルタイプと一致する適切なコンピューティングタイプ (Provisioned または Serverless) が選択されていることを確認します。

Aurora コネクタのトラブルシューティング

このセクションでは、Aurora コネクタの一般的な問題のトラブルシューティングガイダンスについて説明します。Aurora コネクタとリソースの設定については、「」を参照してください[Amazon Aurora に接続する](#)。

1. テーブルの作成時に、サポートされている適切な Aurora バージョンが選択されていることを確認します。
2. Amazon RDS Data API が有効になっていることを確認します。これは、App Studio が Aurora テーブルでオペレーションを実行できるようにするための要件です。詳細については、「[Amazon RDS Data API の有効化](#)」を参照してください。
3. アクセス AWS Secrets Manager 許可が付与されていることを確認します。

DynamoDB コネクタのトラブルシューティング

このセクションでは、DynamoDB コネクタに関する一般的な問題のトラブルシューティングガイダンスについて説明します。DynamoDB コネクタとリソースの設定については、「」を参照してください[Amazon DynamoDB に接続する](#)。

コネクタの作成時に DynamoDB テーブルスキーマが表示されない場合は、DynamoDB テーブルがカスタマーマネージドキー (CMK) で暗号化されており、キーを記述してテーブルを復号するアクセス許可がないとテーブルデータにアクセスできない可能性があります。CMK で暗号化されたテーブルを使用して DynamoDB コネクタを作成するには、IAM ロールに kms:decrypt および アクセス kms:describeKey 許可を追加する必要があります。

Amazon S3 コネクタのトラブルシューティング

このセクションでは、Amazon S3 コネクタの一般的な問題のトラブルシューティングガイダンスについて説明します。Amazon S3 コネクタとリソースの設定については、「」を参照してください[Amazon Simple Storage Service \(Amazon S3\) に接続する](#)。

一般的なトラブルシューティングガイダンスには、以下のチェックが含まれます。

1. Amazon S3 コネクタが Amazon S3 リソースがある AWS リージョンで設定されていることを確認します。
2. IAM ロールが正しく設定されていることを確認します。
3. Amazon S3 バケットで、CORS 設定が適切なアクセス許可を付与していることを確認します。詳細については、「[ステップ 1: Amazon S3 リソースを作成して設定する](#)」を参照してください。

Amazon S3 ファイルのアップロードエラー: 署名付き URL の計算に失敗しました

S3 アップロードコンポーネントを使用して Amazon S3 バケットにファイルをアップロードしようとすると、次のエラーが発生することがあります。

```
Error while uploading file to S3: Failed to calculate presigned URL.
```

このエラーは通常、Amazon S3 バケットの IAM ロール設定が正しくないか、CORS 設定が正しくないために発生します。これらの設定を の情報で修正することで解決できます [Amazon Simple Storage Service \(Amazon S3\) に接続する](#)。

アプリケーションの公開と共有のトラブルシューティング

このトピックには、App Studio アプリケーションを公開または共有する際の一般的な問題のトラブルシューティングガイダンスが含まれています。

共有ダイアログボックスに新しく作成されたアプリロールが表示されない

新しく作成されたアプリレベルのロールは、アプリが再公開された後にのみ共有ダイアログボックスに表示されます。新しいロールが作成された後にアプリを公開して使用します。

アプリの公開が完了したときに E メールが届かなかった

アプリが公開されると、アプリ所有者のみが E メールを受信します。

アプリのエンドユーザーが公開されたアプリにアクセスできない

エンドユーザーが公開されたアプリケーションにアクセスできず、アクセスしようとする とForbiddenメッセージが表示される場合、公開されたアプリケーションはアクセスしようとして

いるユーザーと共有されていない可能性があります。グループ内のユーザーにアクセス権を付与するには、公開されたアプリをグループと共有する必要があります。

アプリケーションの共有の詳細については、「[公開されたアプリケーションの共有](#)」を参照してください。

AWS App Studio のセキュリティ

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 お客様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して説明しています。

- クラウドのセキュリティ – AWS クラウドで 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 AWS 責任があります。AWS また、では、安全に使用できるサービスも提供しています。セキュリティの有効性は、[AWS コンプライアンスプログラム](#)の一環として、サードパーティーの審査機関によって定期的にテストおよび検証されています。App Studio に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 「コンプライアンスプログラムによる対象範囲内のサービス」](#)を参照してください。
- クラウドのセキュリティ – 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、お客様は、お客様のデータの機密性、組織の要件、および適用可能な法律および規制などの他の要因についても責任を担います。

このドキュメントは、App Studio 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。以下のトピックでは、セキュリティおよびコンプライアンスの目的を達成するように App Studio を設定する方法について説明します。また、App Studio 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

- [セキュリティに関する考慮事項と緩和策](#)
- [AWS App Studio でのデータ保護](#)
- [AWS App Studio と AWS Identity and Access Management \(IAM\)](#)
- [AWS App Studio のコンプライアンス検証](#)
- [AWS App Studio の耐障害性](#)
- [AWS App Studio のインフラストラクチャセキュリティ](#)
- [AWS App Studio の設定と脆弱性の分析](#)
- [サービス間での不分別な代理処理の防止](#)
- [AWS App Studio でのクロスリージョンデータ転送](#)

セキュリティに関する考慮事項と緩和策

セキュリティに関する考慮事項

データコネクタ、データモデル、公開されたアプリケーションを処理する場合、データ漏洩、アクセスコントロール、潜在的な脆弱性に関連して、いくつかのセキュリティ上の懸念が発生します。次のリストには、主なセキュリティ上の懸念が含まれています。

IAM ロールの設定が正しくない

データコネクタの IAM ロールの設定が正しくない場合、不正アクセスやデータリークにつながる可能性があります。データコネクタの IAM ロールへのアクセスを過度に許可すると、権限のないユーザーが機密データにアクセスして変更できるようになります。

IAM ロールを使用してデータオペレーションを実行する

App Studio アプリのエンドユーザーは、コネクタ設定で提供される IAM ロールを引き受けてアクションを実行するため、これらのエンドユーザーは通常アクセスできないデータにアクセスする可能性があります。

公開アプリケーションのデータコネクタの削除

データコネクタが削除されても、関連付けられているシークレット認証情報は、そのコネクタを既に使用している公開アプリケーションから自動的に削除されません。このシナリオでは、アプリケーションが特定のコネクタで公開され、それらのコネクタの 1 つが App Studio から削除された場合、公開されたアプリケーションは、以前に保存されたコネクタ認証情報を使用して引き続き機能します。コネクタを削除しても、公開されたアプリは影響を受けず、動作し続けることに注意してください。

公開されたアプリケーションでのデータコネクタの編集

データコネクタを編集すると、そのコネクタを使用している公開アプリケーションに変更は自動的に反映されません。アプリケーションが特定のコネクタで公開されており、それらのコネクタの 1 つが App Studio で変更されている場合、公開されたアプリケーションは、以前に保存されたコネクタ設定と認証情報を引き続き使用します。更新されたコネクタの変更を組み込むには、アプリケーションを再公開する必要があります。アプリが再公開されるまでは、正しくない非運用状態、または影響を受けない運用状態のままですが、最新のコネクタの変更は反映されません。

セキュリティリスク軽減の推奨事項

このセクションでは、前のセキュリティ上の考慮事項セクションで説明したセキュリティリスクを回避するための緩和策の推奨事項を示します。

1. 適切な IAM ロール設定: データコネクタの IAM ロールが、不正アクセスやデータリークを防ぐために最小特権の原則で正しく設定されていることを確認します。
2. 制限されたアプリケーションアクセス: アプリケーションデータを表示または実行する権限を持つユーザーとのみ、アプリケーションを共有します。
3. アプリの発行: コネクタが更新または削除されるたびにアプリが再発行されていることを確認します。

AWS App Studio でのデータ保護

AWS App Studio でのデータ保護には、AWS [責任共有モデル](#)が適用されます。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 アイデンティティセンターまたは 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 は必須ですが、TLS 1.3 を推奨します。
- 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、API、または SDK を使用して AWS App Studio 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。

データ暗号化

App Studio は、保管中および転送中のデータを暗号化することで、データを安全に保存および転送します。

保管中の暗号化

保存時の暗号化とは、保存中にデータを暗号化することで、不正なアクセスからデータを保護することです。App Studio は、デフォルトで AWS KMS キーを使用して保管時の暗号化を提供します。保管時のデータ暗号化に追加の設定を行う必要はありません。

App Studio は、アプリケーションのソースコード、ビルドアーティファクト、メタデータ、アクセス許可情報などのデータを安全に保存します。

Customer AWS KMS Managed Key (CMK) で暗号化されたデータソースを使用する場合、App Studio リソースは引き続き AWS マネージドキーを使用して暗号化されますが、暗号化されたデータソース内のデータは CMK によって暗号化されます。App Studio アプリケーションで暗号化されたデータソースを使用する方法の詳細については、「」を参照してください [CMKs](#)。

App Studio は Amazon CloudFront を使用してユーザーにアプリを提供します。CloudFront は、エッジロケーション POP (Point Of Presence) 用に暗号化された SSD、およびリージョナルエッジキャッシュ (REC) 用に暗号化された EBS ボリュームを使用します。CloudFront Functions の関数コードと設定は、エッジロケーション POP の暗号化された SSD や、CloudFront で使用されるその他のストレージロケーションに、常に暗号化された形式で保存されます。

転送中の暗号化

転送中の暗号化とは、通信エンドポイント間の移動中にデータが傍受されるのを防ぐことです。App Studio は、デフォルトで転送中のデータの暗号化を提供します。顧客と App Studio 間、および App Studio とそのダウンストリームの依存関係間のすべての通信は、署名バージョン 4 の署名プロセスを使用して署名された TLS 接続を使用して保護されます。すべての App Studio エンドポイントは AWS Certificate Manager、プライベート認証局によって管理される SHA-256 証明書を使用します。

キー管理

App Studio は暗号化キーの管理をサポートしていません。

ネットワーク間トラフィックのプライバシー

App Studio でインスタンスを作成するときは、そのインスタンスのデータとリソースが保存される AWS リージョンを選択します。アプリケーションビルドアーティファクトとメタデータがその AWS リージョンを離れることはありません。

ただし、次の情報に注意してください。

- App Studio は Amazon CloudFront を使用してアプリケーションを提供し、Lambda@Edge を使用してアプリケーションへの認証を管理するため、CloudFront エッジロケーションからアクセスされる認証データ、認可データ、アプリケーションメタデータのセットは限られており、別のリージョンにある可能性があります。
- AWS App Studio は、AWS リージョン間でデータを転送して、サービス内の特定の生成 AI 機能を有効にします。クロスリージョンデータ転送で有効になる機能、ガリージョン間を移動するデータの種類、オプトアウト方法の詳細については、「」を参照してください[AWS App Studio でのクロスリージョンデータ転送](#)。

AWS App Studio と AWS Identity and Access Management (IAM)

AWS App Studio では、IAM Identity Center のグループを App Studio の適切なロールに割り当てることで、サービスのアクセスとアクセス許可を管理します。グループメンバーのアクセス許可は、AWS Identity and Access Management (IAM) でユーザー、ロール、またはアクセス許可を直接設定するのではなく、割り当てられたロールによって決定されます。App Studio でのアクセスとアクセス許可の管理の詳細については、「」を参照してください[App Studio でのアクセスとロールの管理](#)。

App Studio は、請求目的でインスタンスを検証するとき、および AWS アカウントに接続してその AWS アカウントでリソースを作成および使用する場合、IAM と統合されます。App Studio をアプリケーションで使用する他の AWS サービスに接続する方法については、「」を参照してください [AWS サービスに接続する](#)。

App Studio でインスタンスを作成するときは、インスタンスの請求および管理 AWS アカウントとしてアカウントを接続する必要があります。主要な機能を有効にするために、App Studio はユーザーに代わってタスクを実行するために必要なアクセス許可をサービスに提供する [IAM サービスロール](#) も作成します。

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 App Studio 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

- [App Studio のアイデンティティベースのポリシー](#)
- [App Studio 内のリソースベースのポリシー](#)
- [App Studio のポリシーアクション](#)
- [App Studio のポリシーリソース](#)
- [App Studio のポリシー条件キー](#)
- [App Studio ACLs](#)
- [App Studio での ABAC](#)
- [App Studio での一時的な認証情報の使用](#)
- [App Studio のクロスサービスプリンシパルアクセス許可](#)
- [App Studio のサービスロール](#)
- [App Studio のサービスにリンクされたロール](#)
- [AWS App Studio の マネージドポリシー](#)
- [App Studio のサービスにリンクされたロール](#)
- [AWS App Studio のアイデンティティベースのポリシーの例](#)

IAM を使用して App Studio へのアクセスを管理する前に、App Studio で使用できる IAM 機能を確認してください。

AWS App Studio で使用できる IAM 機能

IAM の特徴量	App Studio のサポート
アイデンティティベースのポリシー	あり
リソースベースのポリシー	なし
ポリシーアクション	あり
ポリシーリソース	あり
ポリシー条件キー	いいえ
ACL	なし
ABAC (ポリシー内のタグ)	いいえ
一時的な認証情報	あり
プリンシパルアクセス権限	あり
サービスロール	あり
サービスリンクロール	はい

App Studio およびその他の AWS のサービスがほとんどの IAM 機能と連携する方法の概要については、「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。

App Studio の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、ユーザーグループ、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。

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。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。

App Studio のアイデンティティベースのポリシーの例

App Studio の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。[AWS App Studio のアイデンティティベースのポリシーの例](#)。

App Studio 内のリソースベースのポリシー

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コントロールできます。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、[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、全体のアカウント、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。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。

App Studio のポリシーアクション

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。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。

App Studio 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[AWS 「App Studio で定義されるアクション」](#)を参照してください。

App Studio 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を使用します。

```
appstudio
```

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": [  
  "appstudio:action1",  
  "appstudio:action2"  
]
```

次のステートメントは、App Studio のすべてのアクションを一覧表示します。

App Studio のポリシーリソース

ポリシーリソースのサポート: あり

App Studio のアクセス許可は、ポリシーの Resource要素でワイルドカード (*) のみをサポートします。

App Studio のポリシー条件キー

サービス固有のポリシー条件キーへのサポート: なし

App Studio はポリシー条件キーをサポートしていません。

App Studio ACLs

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アクセス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。

App Studio での ABAC

ABAC (ポリシー内のタグ) のサポート: なし

App Studio は、属性ベースのアクセスコントロール (ABAC) をサポートしていません。

App Studio での一時的な認証情報の使用

一時的な認証情報のサポート: あり

一時的な認証情報は、AWS リソースへの短期的なアクセスを提供し、フェデレーションまたはスイッチロールの使用時に自動的に作成されます。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 を行う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「[IAM の一時的な認証情報](#)」および「[AWS のサービスと IAM との連携](#)」を参照してください。

App Studio のクロスサービスプリンシパルアクセス許可

転送アクセスセッション (FAS) のサポート: あり

転送アクセスセッション (FAS) は、 を呼び出すプリンシパルのアクセス許可と 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 を使用します。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。

App Studio 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。

AWS App Studio は、一部の機能に [IAM サービスロール](#) を使用して、ユーザーに代わってタスクを実行するアクセス許可を App Studio に付与します。App Studio をセットアップすると、コンソールはサポートされている機能のサービスロールを自動的に作成します。

Warning

サービスロールのアクセス許可を変更すると、App Studio の機能が破損する可能性があります。App Studio が指示する場合にのみ、サービスロールを編集します。

App Studio のサービスにリンクされたロール

サービスリンクロールのサポート: あり

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。

AWS AWS App Studio の マネージドポリシー

ユーザー、グループ、ロールにアクセス許可を追加するには、自分でポリシーを記述するよりも AWS 管理ポリシーを使用する方が簡単です。チームに必要な権限のみを提供する [IAM カスタマー マネージドポリシーを作成する](#) には時間と専門知識が必要です。すぐに開始するには、AWS マネージドポリシーを使用できます。これらのポリシーは、一般的なユースケースをターゲット範囲に含めており、AWS アカウントで利用できます。AWS 管理ポリシーの詳細については、IAM ユーザーガイドの「[AWS 管理ポリシー](#)」を参照してください。

AWS サービスは、AWS 管理ポリシーを維持および更新します。AWS 管理ポリシーのアクセス許可は変更できません。サービスでは新しい機能を利用できるようにするために、AWS マネージドポリシーに権限が追加されることがあります。この種類の更新はポリシーがアタッチされている、すべてのアイデンティティ (ユーザー、グループおよびロール) に影響を与えます。新しい機能が立ち上げられた場合や、新しいオペレーションが使用可能になった場合に、各サービスが AWS マネージドポリシーを更新する可能性が最も高くなります。サービスは AWS マネージドポリシーからアクセス許可を削除しないため、ポリシーの更新によって既存のアクセス許可が損なわれることはありません。

さらに、 は、複数のサービスにまたがるジョブ関数の マネージドポリシー AWS をサポートしています。例えば、ReadOnlyAccess AWS 管理ポリシーは、すべての AWS サービスとリソースへの読み取り専用アクセスを提供します。サービスが新機能を起動すると、 は新しいオペレーションとリソースの読み取り専用アクセス許可 AWS を追加します。ジョブ機能のポリシーの一覧および詳細については、「IAM ユーザーガイド」の「[AWS のジョブ機能のマネージドポリシー](#)」を参照してください。

AWS マネージドポリシー: AppStudioServiceRolePolicy

IAM エンティティに AppStudioServiceRolePolicy をアタッチすることはできません。このポリシーは、App Studio がユーザーに代わってアクションを実行できるようにするサービスにリンクされたロールにアタッチされます。詳細については、「[App Studio のサービスにリンクされたロール](#)」を参照してください。

このポリシーは、サービスにリンクされたロールが AWS リソースを管理できるようにするアクセス許可を付与します。

許可の詳細

このポリシーには以下を実行するための許可が含まれています。

- logs - CloudWatch ロググループとログストリームを作成します。また、これらのロググループとストリームにログイベントを作成するアクセス許可も付与します。
- secretsmanager - App Studio によって管理されるマネージドシークレットを作成、読み取り、更新、削除します。
- sso - アプリケーションインスタンスを取得します。
- sso-directory - ユーザーに関する情報を取得し、グループ内のメンバーのリストを取得します。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AppStudioResourcePermissionsForCloudWatch",
      "Effect": "Allow",
      "Action": [
        "logs:CreateLogGroup",
        "logs:CreateLogStream",
        "logs:PutLogEvents"
      ],
      "Resource": [
        "arn:aws:logs:*:*:log-group:/aws/appstudio/*"
      ],
      "Condition": {
```

```

        "StringEquals": {
            "aws:ResourceAccount": "${aws:PrincipalAccount}"
        }
    },
    {
        "Sid": "AppStudioResourcePermissionsForSecretsManager",
        "Effect": "Allow",
        "Action": [
            "secretsmanager:CreateSecret",
            "secretsmanager>DeleteSecret",
            "secretsmanager:DescribeSecret",
            "secretsmanager:GetSecretValue",
            "secretsmanager:PutSecretValue",
            "secretsmanager:UpdateSecret",
            "secretsmanager:TagResource"
        ],
        "Resource": "arn:aws:secretsmanager:*:*:secret:appstudio-*",
        "Condition": {
            "ForAllValues:StringEquals": {
                "aws:TagKeys": [
                    "IsAppStudioSecret"
                ]
            },
            "StringEquals": {
                "aws:ResourceAccount": "${aws:PrincipalAccount}",
                "aws:ResourceTag/IsAppStudioSecret": "true"
            }
        }
    },
    {
        "Sid": "AppStudioResourcePermissionsForManagedSecrets",
        "Effect": "Allow",
        "Action": [
            "secretsmanager>DeleteSecret",
            "secretsmanager:DescribeSecret",
            "secretsmanager:GetSecretValue",
            "secretsmanager:PutSecretValue",
            "secretsmanager:UpdateSecret"
        ],
        "Resource": "arn:aws:secretsmanager:*:*:secret:appstudio!*",
        "Condition": {
            "StringEquals": {
                "aws:ResourceAccount": "${aws:PrincipalAccount}",

```

```
    "secretsmanager:ResourceTag/aws:secretsmanager:owningService":
      "appstudio"
        }
      }
    },
    {
      "Sid": "AppStudioResourceWritePermissionsForManagedSecrets",
      "Effect": "Allow",
      "Action": [
        "secretsmanager:CreateSecret"
      ],
      "Resource": "arn:aws:secretsmanager:*:*:secret:appstudio!*",
      "Condition": {
        "StringEquals": {
          "aws:ResourceAccount": "${aws:PrincipalAccount}"
        }
      }
    },
    {
      "Sid": "AppStudioResourcePermissionsForSSO",
      "Effect": "Allow",
      "Action": [
        "sso:GetManagedApplicationInstance",
        "sso-directory:DescribeUsers",
        "sso-directory:ListMembersInGroup"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "aws:ResourceAccount": "${aws:PrincipalAccount}"
        }
      }
    }
  ]
}
```

AWS マネージドポリシーに対する App Studio の更新

このサービスがこれらの変更の追跡を開始してからの App Studio の AWS マネージドポリシーの更新に関する詳細を表示します。

変更	説明	日付
AppStudioServiceRolePolicy – 既存のポリシーの更新	App Studio に、App Studio マネージドシークレットの管理を許可する新しいアクセス許可が追加されました AWS Secrets Manager。	2025 年 3 月 14 日
App Studio が変更の追跡を開始しました	App Studio は、AWS 管理ポリシーの変更の追跡を開始しました。	2024 年 6 月 28 日

App Studio のサービスにリンクされたロール

App Studio は [AWS Identity and Access Management \(IAM\) サービスにリンクされたロール](#)を使用します。サービスにリンクされたロールは、App Studio に直接リンクされた一意のタイプの IAM ロールです。サービスにリンクされたロールは App Studio によって事前定義されており、サービスがユーザーに代わって他の AWS サービスを呼び出すために必要なすべてのアクセス許可が含まれています。

サービスにリンクされたロールを使用すると、必要なアクセス許可を手動で追加する必要がなくなるため、App Studio の設定が簡単になります。App Studio は、サービスにリンクされたロールのアクセス許可を定義します。特に定義されている場合を除き、App Studio のみがそのロールを引き受けることができます。定義される許可は信頼ポリシーと許可ポリシーに含まれており、その許可ポリシーを他の IAM エンティティにアタッチすることはできません。

サービスリンクロールを削除するには、最初に関連リソースを削除する必要があります。これにより、リソースへのアクセス許可が誤って削除されないため、App Studio リソースが保護されます。

内容

- [App Studio のサービスにリンクされたロールのアクセス許可](#)
- [App Studio のサービスにリンクされたロールの作成](#)
- [App Studio のサービスにリンクされたロールの編集](#)
- [App Studio のサービスにリンクされたロールの削除](#)

App Studio のサービスにリンクされたロールのアクセス許可

App Studio は、 という名前のサービスにリンクされたロールを使用します `AWSServiceRoleForAppStudio`。これは、アプリケーション構築エクスペリエンスを維持するために、App Studio が AWS サービスを永続的に管理するために必要なサービスにリンクされたロールです。

`AWSServiceRoleForAppStudio` サービスにリンクされたロールは、サービスのみを信頼する次の信頼ポリシーを使用します `appstudio-service.amazonaws.com`。

JSON

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

アクセス許可の場合、`AWSServiceRoleForAppStudio` サービスにリンクされたロールは次のサービスへのアクセス許可を提供します。

- Amazon CloudWatch: App Studio の使用に関するログとメトリクスを送信します。
- AWS Secrets Manager: App Studio でコネクタの認証情報を管理するために、アプリケーションを他のサービスに接続するために使用されます。
- IAM アイデンティティセンター: ユーザーアクセスを管理するための読み取り専用アクセス。

具体的には、で付与されるアクセス許可 `AWSServiceRoleForAppStudio` は、アタッチされた `AppStudioServiceRolePolicy` 管理ポリシーによって定義されます。含まれるアクセス許可を含む管理ポリシーの詳細については、「」を参照してください [AWS マネージドポリシー: AppStudioServiceRolePolicy](#)。

App Studio のサービスにリンクされたロールの作成

サービスリンクロールを手動で作成する必要はありません。App Studio インスタンスを作成すると、App Studio によってサービスにリンクされたロールが作成されます。

このサービスにリンクされたロールを削除する場合は、App Studio インスタンスを作成して、別のインスタンスを自動的に作成することをお勧めします。

必須ではありませんが、前述の信頼ポリシーニペットのよう、サービス名でサービスにリンクされたロールを作成することで、IAM コンソールまたは `awscli` を使用して `appstudio-service.amazonaws.com` サービスにリンクされたロール `AWS CLI` を作成することもできます。詳細については、「IAM ユーザーガイド」の「[サービスリンクロールの作成](#)」を参照してください。

App Studio のサービスにリンクされたロールの編集

App Studio では、`AWSServiceRoleForAppStudio` サービスにリンクされたロールを編集することはできません。サービスリンクロールを作成すると、多くのエンティティによってロールが参照される可能性があるため、ロール名を変更することはできません。ただし、IAM を使用してロールの説明の編集はできます。詳細については、「IAM ユーザーガイド」の「[サービスリンクロールの編集](#)」を参照してください。

App Studio のサービスにリンクされたロールの削除

`AWSServiceRoleForAppStudio` ロールを削除する必要はありません。App Studio インスタンスを削除すると、App Studio はリソースをクリーンアップし、サービスにリンクされたロールを自動的に削除します。

推奨されませんが、IAM コンソールまたは `awscli` を使用して、サービスにリンクされたロールを削除できます。これを行うには、まずサービスにリンクされたロールのリソースをクリーンアップしてから、削除する必要があります。

Note

リソースを削除しようとしたときに App Studio がロールを使用している場合は、削除が失敗する可能性があります。その場合は、数分待ってからオペレーションを再試行してください。

IAM を使用してサービスリンクロールを手動で削除するには

1. App Studio インスタンスからアプリケーションとコネクタを削除します。
2. AWSServiceRoleForAppStudio サービスにリンクされたロールを削除するには、IAM コンソール、IAM CLI、または IAM API を使用します。詳細については、「IAM ユーザーガイド」の「[サービスにリンクされたロールの削除](#)」を参照してください。

AWS App Studio のアイデンティティベースのポリシーの例

デフォルトでは、ユーザーとロールには App Studio リソースを作成または変更するアクセス許可はありません。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。

これらのサンプルの 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 ARNs 「サービス認可リファレンス」の[AWS 「App Studio のアクション、リソース、および条件キー」](#)を参照してください。

トピック

- [ポリシーに関するベストプラクティス](#)
- [App Studio コンソールの使用](#)
- [自分の権限の表示をユーザーに許可する](#)
- [例 1: ユーザーに App Studio インスタンスの設定を許可する](#)
- [例 2: App Studio インスタンスの設定をユーザーに拒否する](#)

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、アカウント内で誰かが App Studio 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では、AWS アカウントに費用が発生する場合があります。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 ポリシーでアクセス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。たとえば、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 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 アクセスアナライザーを使用して IAM ポリシーを検証し、安全で機能的な権限を確保する - IAM アクセスアナライザーは、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 – 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。

App Studio コンソールの使用

AWS App Studio 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 App Studio 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引き続き App Studio 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 App Studio *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:ListGroupForUser",
        "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": "*"
  }
]
}
```

例 1: ユーザーに App Studio インスタンスの設定を許可する

次の例は、ロールに App Studio インスタンスの設定を許可するアイデンティティベースのポリシーを示しています。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Allow",
    "Action": [
      "appstudio:GetAccountStatus",
      "appstudio:GetEnablementJobStatus",
      "appstudio:StartEnablementJob",
      "appstudio:StartRollbackEnablementJob",
      "appstudio:StartTeamDeployment"
    ],
    "Resource": "*"
  }]
}
```

例 2: App Studio インスタンスの設定をユーザーに拒否する

次の例は、ロールが App Studio インスタンスをセットアップすることを拒否するアイデンティティベースのポリシーを示しています。

JSON

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

```
"Action": [  
  "appstudio:*"  
],  
"Resource": "*" ]  
}
```

AWS App Studio のコンプライアンス検証

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対象範囲内のコンプライアンス」](#)を参照し、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「コンプライアンスプログラム」](#)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、適用可能な法律および規制によって決まります。を使用する際のコンプライアンス責任の詳細については AWS のサービス、[AWS「セキュリティドキュメント」](#)を参照してください。

AWS App Studio の耐障害性

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ゾーン AWS リージョン を提供します。アベイラビリティゾーンでは、ゾーン間で中断することなく自動的にフェー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、フォールトトレランス、および拡張性が優れています。

AWS リージョン およびアベイラビリティ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。

AWS グローバルインフラストラクチャに加えて、AWS App Studio には、データの耐障害性とバックアップのニーズをサポートするのに役立つ機能がいくつか用意されています。

AWS App Studio のインフラストラクチャセキュリティ

マネージドサービスである AWS App Studio は、ホワイトペーパー「[Amazon Web Services: セキュリティプロセスの概要](#)」に記載されている AWS グローバルネットワークセキュリティ手順で保護されています。

AWS 公開された API コールを使用して、ネットワーク経由で App Studio にアクセスします。クライアントは少なくとも Transport Layer Security (TLS) 1.2 をサポートする必要がありますが、TLS 1.3 が推奨されます。また、DHE (Ephemeral Diffie-Hellman) や ECDHE (Elliptic Curve Ephemeral Diffie-Hellman) などの Perfect Forward Secrecy (PFS) を使用した暗号スイートもクライアントでサポートされている必要があります。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。

AWS App Studio の設定と脆弱性の分析

設定と IT コントロールは、AWS とお客様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。

サービス間での不分別な代理処理の防止

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、より特権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、1 つのサービス (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は、本来ならアクセスすることが許可されるべきではない方法でその許可を使用して、別のお客様のリソースに対する処理を実行するように操作される場合があります。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権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、ガリソースに別のサービス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。クロスサービス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。そのアカウント内のリソースをクロスサービスの使用に関連付けることを許可する場合は、`aws:SourceAccount` を使用します。

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。リソースの完全な ARN が不明な場合や、複数のリソースを指定する場合には、グローバルコンテキスト条件キー `aws:SourceArn` で、ARN の未知部分を示すためにワイルドカード文字 (*) を使用します。例えば、`arn:aws:service:*:123456789012:*`。

`aws:SourceArn` の値に Amazon S3 バケット ARN などのアカウント ID が含まれていない場合は、両方のグローバル条件コンテキストキーを使用して、アクセス許可を制限する必要があります。

`aws:SourceArn` の値は `ResourceDescription` である必要があります。

次の例では、`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、混乱した代理問題を回避する方法を示します。

AWS App Studio でのクロスリージョンデータ転送

AWS App Studio は、AWS リージョン間でデータを転送して、サービス内の特定の生成 AI 機能を有効にします。このトピックでは、クロスリージョンデータ転送で有効になる機能、ガリージョン間を移動するデータの種類、オプトアウト方法について説明します。

次の機能はクロスリージョンデータ転送によって有効になり、オプトアウトするとインスタンスでアクセスできなくなります。

1. AI を使用してアプリを作成する。自然言語でアプリを記述し、リソースを作成することで、アプリ構築を開始するために使用します。
2. アプリケーションスタジオの AI チャット。アプリケーションの構築、公開、共有について質問するために使用されます。

次のデータはリージョン間で転送されます。

1. 前述の機能からのプロンプトまたはユーザー入力。

クロスリージョンデータ転送をオプトアウトし、それが有効にする機能を使用するには、次の手順を使用してコンソールからオプトアウトリクエストフォームに入力します。

1. <https://console.aws.amazon.com/appstudio/> で App Studio コンソールを開きます。

2. データ転送のオプトアウトを選択します。
3. AWS アカウント ID を入力し、E メールアドレスを入力します。
4. [Submit] を選択してください。
5. 送信すると、クロスリージョンデータ転送のオプトアウトリクエストが処理されます。これには最大 60 日かかる場合があります。

AWS App Studio でサポートされているブラウザ

このトピックには、公開されたアプリケーションにアクセスするエンドユーザーとアプリケーションビルダーの両方に対するブラウザのサポートなど、AWS App Studio でサポートされているブラウザと推奨されるブラウザに関する情報が含まれています。

アプリケーションを構築するためにサポートされるブラウザと推奨ブラウザ

最適なアプリケーション構築エクスペリエンスを実現するために、App Studio は Google Chrome をサポートしており、使用を強くお勧めします。

Note

推奨されませんが、Mozilla Firefox、Microsoft Edge、Apple Safari for MacOS などの他の一般的なウェブブラウザを使用してアプリケーションを構築することもできますが、これらのブラウザは公式にサポートまたは検証されていないため、一部のビルダー機能にアクセスするために設定を更新する必要がある場合があります。詳細については、「[ブラウザ設定を更新して App Studio でアプリケーションを構築する](#)」を参照してください。

App Studio は、モバイルプラットフォームからのアプリケーションの構築をサポートしていません。

アプリケーションのエンドユーザーがサポートおよび推奨するブラウザ

公開されたアプリケーションにアクセスするエンドユーザーの場合、App Studio では Google Chrome または Mozilla Firefox を使用することを強くお勧めします。これらは推奨ブラウザですが、エンドユーザーは Microsoft Edge や Apple Safari for MacOS など、他の一般的なウェブブラウザを使用して公開されたアプリケーションにアクセスすることもできます。

エンドユーザーは、モバイルプラットフォームから公開されたアプリケーションにアクセスすることもできます。

ブラウザ設定を更新して App Studio でアプリケーションを構築する

App Studio は、Google Chrome を使用してアプリケーションを構築することを公式にサポートおよび推奨しています。ただし、他のブラウザを使用してアプリケーションを構築する場合は、クロスサイト追跡に関連する特定の設定または Cookie を更新して、App Studio の特定のページにアクセスする必要がある場合があります。

Mozilla Firefox の場合: アプリケーションをプレビューするには、次の設定を Firefox Settings > Privacy & Security > Enhanced Tracking Protection に更新します。Custom > Cookies > Cross-site tracking cookies

MacOS 用 Apple Safari の場合: アプリケーションを構築またはプレビューするには、次の設定を無効にします: Settings > Privacy > Prevent cross-site tracking。

AWS App Studio のクォータ

次の表に、AWS App Studio のクォータと制限を示します。

App Studio インスタンス内のアプリの最大数	20
App Studio インスタンスのテスト環境または本番稼働環境に公開されるアプリケーションの最大数。テストと本番稼働の両方に公開された 1 つのアプリケーションは、公開された 2 つのアプリケーションとしてカウントされます。	6
アプリケーションあたりのマネージドエンティティの最大数	20
クエリごとに返される行の最大数	3000
エンティティあたりのサンプルデータの最大行数	500
オートメーションの最大実行時間	2 分。2 分以上実行されるオートメーションは失敗します。
自動化の最大入力サイズと出力サイズ	入力または出力あたり 5GB。
オートメーションまたはデータアクションで使用される最大データサイズ	オートメーションまたはデータアクションの実行あたり 450MB。
ページ名とコンポーネント名	空ではなく一意である必要があります。文字、数字のアンダースコア (_)、ドル記号 (\$) のみを含める必要があります。スペースを含めることはできません。

AWS App Studio ユーザーガイドのドキュメント履歴

次の表に、AWS App Studio のドキュメントリリースを示します。

変更	説明	日付
新しいトピック: アプリケーションの本番稼働環境バージョンをロールバックする	問題が検出された場合に、公開されたバージョンのアプリを以前に公開されたバージョンにロールバックする情報を追加しました。詳細については、 「以前に公開されたバージョンにロールバックする」 を参照してください。	2025 年 4 月 10 日
新しいトピック: 重複するアプリケーション、ページ、コンポーネント	App Studio でのアプリケーション、ページ、コンポーネントの複製に関する情報を含む新しいトピックを追加しました。詳細については、 「アプリケーションの複製」 、 「ページの複製」 、および 「コンポーネントの複製」 を参照してください。	2025 年 4 月 7 日
新しいトピック: App Studio インスタンス間でアプリケーションをインポートおよびエクスポートする	App Studio インスタンス間でのアプリケーションのインポートとエクスポートに関する情報を含む新しいトピックを追加しました。これには、App Studio が提供する、アプリケーション構築の概念を学習するために使用できるインポート可能なアプリケーションのリストが含まれません。詳細については、 「アプリケーションのインポート」	2025 年 3 月 30 日

と「[アプリケーションのエクスポート](#)」を参照してください。

[更新されたトピック: AWS 管理ポリシー: AppStudio ServiceRolePolicy](#)

各サービスのAppStudio ServiceRolePolicy アクセス許可を更新し、ポリシーの説明情報を追加しました。詳細については、「[AWS マネージドポリシー: AppStudio ServiceRolePolicy](#)」を参照してください。

2025 年 3 月 14 日

[更新されたトピック: データアクションの編集または設定](#)

データアクション条件で 사용되는既存および新規の演算子に関する情報を追加しました。これは、条件に一致するデータベーステーブルからデータのサブセットを取得するために使用されます。詳細については、「[データアクションの編集または設定](#)」を参照してください。

2025 年 2 月 27 日

[更新されたトピック: JavaScript を使用して式を記述する](#)

アプリケーションの式を使用した UI コンポーネントとテーブルデータの参照または更新に関する情報を再編成し、追加しました。詳細については、「[Using JavaScript to write expressions in App Studio](#)」を参照してください。

2025 年 2 月 18 日

[トピックの更新: アプリコンテンツのセキュリティ設定](#)

コンテンツソースアプリのコンテンツセキュリティ設定に関する情報を追加しました。この設定を使用して、アプリが Amazon S3 にオブジェクトをアップロードできるドメインを制限できます。詳細については、[「アプリのコンテンツセキュリティ設定の表示または更新」](#)を参照してください。

2025 年 2 月 14 日

[新しいトピック: App Studio アプリでの Lambda 関数の呼び出し](#)

App Studio アプリで Lambda 関数を呼び出す方法を説明する短いチュートリアルを追加しました。詳細については、[「Lambda 関数の呼び出し」](#)を参照してください。

2025 年 1 月 24 日

[新しいトピック: Amazon SES に接続する](#)

App Studio アプリケーションでサービスを使用するための Amazon SES コネクタを作成する手順を追加しました。詳細については、[「Amazon Simple Email Service に接続する」](#)を参照してください。

2025 年 1 月 16 日

[更新されたトピック: App Studio インスタンスを初めて作成して設定する](#)

Easy Create メソッドを使用して App Studio インスタンスを作成し、より迅速に開始する手順を追加しました。詳細については、[「App Studio インスタンスを初めて作成して設定する」](#)を参照してください。

2024 年 12 月 13 日

[新しいトピック: データの依存関係とタイミングの問題を管理するためのベストプラクティス](#)

App Studio アプリでのデータ依存関係とタイミング問題の適切な管理に関するドキュメントを追加しました。詳細については、[「データ依存関係とタイミングに関する考慮事項」](#)を参照してください。

2024 年 11 月 20 日

[更新されたトピック: AI を使用したアプリの編集](#)

アプリケーションスタジオの AI チャットを使用したアプリケーションの編集に関する情報を含むドキュメントを追加しました。詳細については、[「生成 AI を使用した App Studio アプリの構築」](#)を参照してください。

2024 年 11 月 18 日

[更新されたトピック: AI を使用して JavaScript を生成する](#)

JavaScript 自動化アクションリファレンスを更新し、AI を使用して JavaScript を生成する方法に関する情報を追加しました。詳細については、[「JavaScript オートメーションアクション」](#)を参照してください。

2024 年 11 月 18 日

[更新されたトピック: Amazon Bedrock を使用して AI テキストサマリアプリを構築する](#)

新しくリリースされた GenAI プロンプトアクションを使用するように Amazon Bedrock プロンプトチュートリアルを更新しました。詳細については、[「Amazon Bedrock で AI テキストサマリアプリを構築する」](#)を参照してください。

2024 年 11 月 18 日

[新しいトピック: アプリテーマでアプリの色を変更する](#)

アプリテーマを使用したアプリの色の変更に関する情報を含むトピックを追加しました。詳細については、[「アプリテーマでアプリの色を変更する」](#)を参照してください。

2024 年 11 月 18 日

[新しいトピック: データモデルのベストプラクティス](#)

App Studio アプリケーションで使用するために、安全で堅牢、スケーラブルなデータモデルを作成するためのベストプラクティスを含むトピックを追加しました。詳細については、[「データモデルを設計する際のベストプラクティス」](#)を参照してください。

2024 年 11 月 15 日

[更新されたトピック: AWS サービスへの接続](#)

AWS サービスへのコネクタを作成するために使用される IAM ロール `sts:ExternalId` に必要な `ExternalId` を含むように信頼ポリシーを更新しました。詳細については、[「AWS サービスに接続する」](#)を参照してください。

2024 年 11 月 13 日

[新しいトピック: 以前に公開されたアプリバージョンにロールバックまたは元に戻す](#)

アプリケーションを以前に公開されたバージョンにロールバックまたは元に戻す方法に関する情報を含むトピックを追加しました。詳細については、[「以前に公開されたバージョンにロールバックする」](#)を参照してください。

2024 年 11 月 13 日

[新しいトピック: App Studio インスタンスを削除する](#)

App Studio インスタンスの削除方法など、App Studio インスタンスの削除に関する情報を含むトピックを追加しました。詳細については、[「App Studio インスタンスの削除」](#)を参照してください。

2024 年 11 月 12 日

[新しいトピック: アプリコンテンツのセキュリティ設定の更新](#)

App Studio のアプリコンテンツセキュリティ設定の更新方法などに関する情報を含むトピックを追加しました。詳細については、[「アプリのコンテンツセキュリティ設定の表示または更新」](#)を参照してください。

2024 年 11 月 8 日

[更新されたトピック: AWS App Studio のセキュリティ](#)

データ保護に関する情報や App Studio が IAM とやり取りする方法など、セキュリティドキュメントを拡張しました。詳細については、[AWS 「App Studio のセキュリティ」](#)を参照してください。

2024 年 11 月 6 日

[更新されたトピック: AWS App Studio のクォータ](#)

App Studio サービスクォータを更新し、ドキュメントを制限して誤った値を修正し、クォータを削除しました。詳細については、App [Studio の「クォータ AWS」](#)を参照してください。

2024 年 10 月 21 日

[更新されたトピック: App Studio を他の AWS サービスに接続する](#)

App Studio に AWS サービスまたはリソースに必要な最小限のアクセス許可を付与する手順と例を提供することで、サービスに接続してベストプラクティスに準拠するためのドキュメントを更新しました。詳細については、[「AWS サービスに接続する」](#)を参照してください。

2024 年 10 月 18 日

[更新されたトピック: Aurora コネクタドキュメントにバージョンサポートを追加](#)

Aurora コネクタドキュメントにサポートされているバージョンのリストを追加しました。詳細については、[「Amazon Aurora に接続する」](#)を参照してください。

2024 年 10 月 16 日

[新しいトピック: App Studio でサポートされているブラウザ](#)

App Studio を使用するためのブラウザのサポートと推奨事項を含むトピックを追加しました。詳細については、[「サポートされているブラウザ」](#)を参照してください。

2024 年 10 月 10 日

[新しいトピック: AWS App Studio の仕組み](#)

図やスクリーンショットなど、App Studio でのアプリケーション開発の主要な概念を説明するトピックを追加しました。詳細については、[「App Studio の仕組み」](#)を参照してください。

2024 年 10 月 10 日

[新しいトピック: ページの順序付けと整理](#)

プレビューまたは公開アプリケーションのナビゲーションでページを並べ替えたり非表示にしたり表示したりする情報を含むトピックを追加しました。詳細については、[「ページの順序付けと整理」](#)を参照してください。

2024 年 9 月 24 日

[新しいトピック: AWS App Studio のクォータ](#)

App Studio に関連するサービスクォータと制限を含むトピックを追加しました。詳細については、App [Studio の「クォータ AWS」](#)を参照してください。

2024 年 9 月 11 日

[更新されたトピック: 暗号化された DynamoDB テーブルに接続する](#)

App Studio で AWS KMS カスタマーマネージドキー (CMKs) で暗号化された DynamoDB テーブルを使用するために必要なアクセス許可などの情報を追加しました。詳細については、[DynamoDB に接続する](#)」を参照してください。

2024 年 9 月 6 日

[更新されたトピック: DynamoDB、Amazon Redshift、Aurora に接続する](#)

App Studio アプリケーションで DynamoDB、Amazon Redshift、および Aurora リソースを使用するために必要な最小限のアクセス許可を IAM ロールに追加しました。詳細については、[「AWS サービスに接続する」](#)を参照してください。

2024 年 9 月 5 日

[トピックの更新: Amazon Aurora への接続](#)

App Studio アプリで使用する Amazon Aurora データベースとテーブルを作成および設定するためのドキュメントを更新しました。詳細については、[「Amazon Aurora に接続する」](#)を参照してください。

2024 年 9 月 5 日

[新規および更新されたトピック: トラブルシューティングとデバッグ](#)

アプリケーションを構築するためのデバッグ情報など、App Studio の一般的な問題の解決に役立つトラブルシューティングとデバッグのドキュメントを拡張しました。詳細については、[「App Studio のトラブルシューティングとデバッグ」](#)を参照してください。

2024 年 8 月 26 日

[新しいトピック: チュートリアル: Amazon Bedrock で AI テキストサマリアプリを構築する](#)

チュートリアルのステップに従って、エンドユーザーからの入力プロンプトを受け取り、Amazon Bedrock に送信して要約バージョンを返して表示するアプリを構築します。詳細については、[「Amazon Bedrock で AI テキストサマリアプリを構築する」](#)を参照してください。

2024 年 8 月 20 日

[更新されたトピック: App Studio アプリのプレビュー、公開、共有](#)

ドキュメントのプレビュー、公開、共有を拡張して、わかりやすくし、サービスのエクスペリエンスに一致させ、公開環境とそこでのアプリの表示に関する追加情報を提供しました。詳細については、[「アプリケーションのプレビュー、公開、共有」](#)を参照してください。

2024 年 8 月 2 日

[新しいトピック: 複数のユーザーによるアプリの構築](#)

ドキュメントのプレビュー、公開、共有を拡張して、わかりやすくし、サービスのエクスペリエンスに一致させ、公開環境とそこでのアプリの表示に関する追加情報を提供しました。詳細については、[「複数のユーザーによるアプリの構築」](#)を参照してください。

2024 年 8 月 2 日

[トピックの更新: App Studio のサービスへの接続 AWS](#)

その他の AWS サービスコネクタの作成時に AWS リソースへのアクセスを提供するための IAM ロールの作成と提供に関する情報を追加しました。詳細については、[「その他の AWS サービスコネクタを使用して AWS サービスに接続する」](#)を参照してください。

2024 年 7 月 29 日

[更新されたトピック: 設定の環境として AWS 管理ユーザーを作成する手順を追加](#)

AWS リソースを管理するための管理ユーザーを作成する手順が [App Studio](#) ドキュメントの設定に追加されました。また、コネクタドキュメント全体で更新を行い、そのユーザーの使用を推奨しました。

2024 年 7 月 24 日

[新しいトピック: Amazon Bedrock に接続する](#)

Amazon Bedrock のコネクタを作成する手順を含むトピックを追加しました。ビルダーは、コネクタを使用して Amazon Bedrock を使用するアプリケーションを構築できます。詳細については、[「Amazon Bedrock に接続する」](#)を参照してください。

2024 年 7 月 24 日

[初回リリース](#)

AWS App Studio ユーザーガイドの初回リリース

2024 年 7 月 10 日

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。