実装ガイド

Virtual Waiting Room on AWS

Copyright © 2024 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Virtual Waiting Room on AWS: 実装ガイド

Copyright © 2024 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 が所有していない他のすべての商標は、それぞれの所有者の所有物であり、Amazon と提携、接続、または後援されている場合とされていない場合があります。

Table of Contents

Virtual Waiting Room on AWS でウェブサイトへの大量のトラフィックを吸収する

公開日: 2021 年 11 月 (最終更新日: 2024 年 11 月)

Virtual Waiting Room on AWS ソリューションは、トラフィックが急増しているときにウェブサイトへの着信ユーザーリクエストを制御するのに役立ちます。このソリューションでは、対象ウェブサイトへのトラフィックを一時的にオフロードするように設計されたクラウドインフラストラクチャを作成し、仮想待合室をカスタマイズおよび統合するためのオプションを提供します。このソリューションを新規または既存のウェブサイトと統合すると、シームレスなスケーリングにより、トラフィックの急激な増加に対応できます。

ウェブサイトでトラフィックの急増を引き起こす可能性のある大規模なイベントには、次のようなものがあります。

- コンサートやスポーツイベントのチケットの販売開始
- 特売や他の大規模なセール (ブラックフライデーなど)
- 多数の人々に向けたマーケティング関連の発表を伴う新製品のリリース
- オンライン試験へのアクセスや、オンライン授業への出席
- 診療予約枠の解放
- アカウントの作成と支払いを必要とする新しい D2C サービスの開始

このソリューションは、ウェブサイトへの訪問者が待機する場所としての役割を果たし、十分なキャパシティがあるときにトラフィックの通過を許可します。訪問者が使用するクライアントソフトウェアでは、ウェブサイトのキャパシティが上限に達するまでトラフィックが透過的に待合室を通過できるように設定できます。キャパシティが上限に達すると、訪問者は待合室に留まります。ウェブサイトに追加のトラフィックを受け入れる余裕ができると、ウェブサイトへのアクセスをユーザーに許可する JSON Web Token (JWT) が生成されます。例えば、2 時間続くイベントがあり、ウェブサイトで処理できるユーザー数が 1 秒あたり 50 人であるにもかかわらず、1 秒あたり 250 人分のトラフィック量が予想される場合に、このソリューションを使用すると、トラフィックを調節しながら、キューでユーザーの順序を維持することができます。

このソリューションには、主に次のような機能があります。

- 構造化されたキューイングでウェブサイトへのユーザーアクセスを処理

- きわめて大規模なイベントのトラフィックを制御するスケーラビリティ
- ターゲットサイトへのアクセスを許可する JSON Web Token の生成
- すべての機能を REST API 経由で制御
- クライアントソリューション用のターンキー API Gateway
- スタンドアロン統合または OpenID との併用

この実装ガイドでは、Amazon Web Services (AWS) クラウドに Virtual Waiting Room on AWS をデプロイするためのアーキテクチャ上の考慮事項と設定手順について説明します。これには、セキュリティと可用性に関する AWS ベストプラクティスを使用して、このソリューションをデプロイするために必要な AWS サービスを起動および設定する AWS CloudFormation テンプレートへのリンクが含まれています。

このガイドは、AWS クラウドにおけるアーキテクチャ設計の実務経験を持つ IT アーキテクト、デベロッパー、DevOps スタッフ、データアナリスト、マーケティング技術のプロフェッショナルを対象としています。

コスト

このソリューションの実行中に使用した AWS サービスのコストは、お客様の負担となります。この改訂の時点で、米国東部 (バージニア北部) リージョンでこのソリューションをデフォルト設定で実行する場合のコストは、概算でスタックあたり 10.00 USD/日に、イベントサイズに応じて API リクエスト数とデータトラフィック量に対する料金を加算した額になります。

ソリューションを維持するための 1 日のコスト (イベントがない場合)

AWS のサービス	リクエスト/時間	コスト [USD]
Amazon API Gateway	0	0.00 USD
Amazon CloudFront	0	0.00 USD
Amazon CloudWatch	0	0.00 USD
Amazon DynamoDB	0	0.00 USD
Amazon ElastiCache	コンピューティングノード時間 (Redis)	~6.00 USD
AWS Lambda	無料利用枠*	0.00 USD
AWS Secrets Manager	無料利用枠*	0.00 USD
Amazon Simple Storage Service (Amazon S3)	無料利用枠*	0.00 USD
Amazon Virtual Private Cloud (Amazon VPC)	VPC エンドポイント時間 NAT ゲートウェイ時間	~5.00 USD
合計:		~11.00 USD

*コストの見積もりは、クリーンな環境に基づいています。このソリューション以外でこの AWS のサービスを使用している場合は、無料利用枠のクォータを超える可能性があります。

次の表は、イベントの長さは 2～4 時間で、入るユーザー数が 500 ユーザー/秒、出るユーザー数が 1,000 ユーザー/分の場合の、50,000 ユーザーと 100,000 ユーザーの待合室の推定コストを示しています。料金は変更されることがあります。詳細については、このソリューションで使用する各 AWS のサービスの料金ウェブページを参照してください。

イベントの長さが 2 時間、待合室のユーザー数が 50,000 人の推定コスト

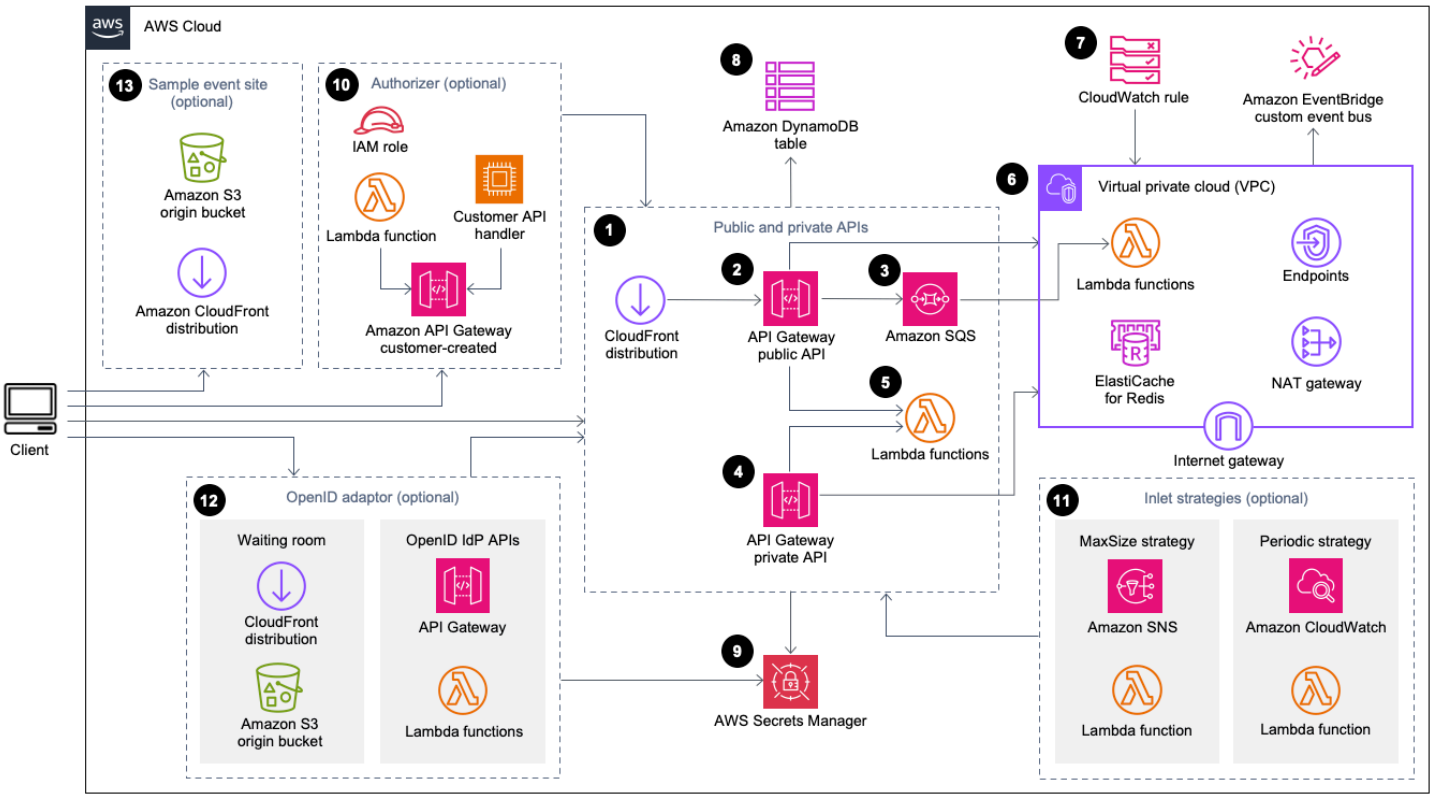
AWS のサービス	ディメンション	コスト [USD]
Amazon API Gateway	リクエスト	2.00 USD
CloudFront	リクエスト、帯域幅	75.00 USD
CloudWatch	メトリクス、アラーム、ストレージ	1.00 USD
Amazon CloudWatch Events	イベント	1.00 USD
DynamoDB	読み取り/書き込みユニット、ストレージ	1.00 USD
ElastiCache	ノード時間	8.00 USD
Lambda	リクエスト、計算時間	1.00 USD
AWS Secrets Manager	シークレット、リクエスト	1.00 USD
Amazon S3	リクエスト、ストレージ	1.00 USD
Amazon VPC	データ転送、エンドポイント時間	2.00 USD
合計		94.00 USD

イベントの長さが 2 時間、待合室のユーザー数が 100,000 人の推定コスト

AWS のサービス	ディメンション	コスト [USD]
Amazon API Gateway	リクエスト	4.00 USD
CloudFront	リクエスト、帯域幅	296.00 USD
CloudWatch	メトリクス、アラーム、ストレージ	1.00 USD
CloudWatch Events	イベント	1.00 USD
DynamoDB	読み取り/書き込みユニット、ストレージ	4.00 USD
ElastiCache	ノード時間	32.00 USD
Lambda	リクエスト、計算時間	1.00 USD
AWS Secrets Manager	シークレット、リクエスト	1.00 USD
Amazon Simple Queue Service (Amazon SQS)	リクエスト	1.00 USD
Amazon S3	リクエスト、ストレージ	1.00 USD
Amazon VPC	データ転送、エンドポイント時間	6.00 USD
合計		348.00 USD

アーキテクチャの概要

デフォルトのパラメータを使用して必須テンプレートとオプションのテンプレートでこのソリューションをデプロイすると、AWS クラウドに以下の環境が構築されます。



Virtual Waiting Room on AWS のアーキテクチャ

AWS CloudFormation テンプレートは次のインフラストラクチャをデプロイします。

1. クライアントにパブリック API コールを配信する Amazon CloudFront ディストリビューション。
2. 仮想待合室からのキューリクエストを処理し、キューの位置を追跡して、ターゲットウェブサイトへのアクセスを許可するトークンの検証をサポートする Amazon API Gateway のパブリック API リソース。
3. キューメッセージを処理する AWS Lambda 関数へのトラフィックを調整する Amazon Simple Queue Service (Amazon SQS) キュー。この SQS キューは、リクエストごとに Lambda 関数を呼び出す代わりに、受信したリクエストのバーストをバッチ処理します。
4. 管理機能をサポートする API Gateway のプライベート API リソース。
5. パブリック API リクエストとプライベート API リクエストを検証および処理し、適切なレスポンスを返す Lambda 関数。

6. Elasticache (Redis OSS) クラスターと直接やり取りする Lambda 関数をホストする Amazon Virtual Private Cloud (VPC)。VPC エンドポイントにより、VPC 内の Lambda 関数はソリューション内のサービスと通信できます。さらに、NAT ゲートウェイにより、VPC の Lambda 関数は CloudFront エンドポイントに接続し、必要に応じてキャッシュを無効にできます。
7. カスタムの Amazon EventBridge バスと連携して、ステータスアップデートを定期的にブロードキャストする Lambda 関数を呼び出す Amazon CloudWatch ルール。
8. トークン、キューの位置、サービングカウンターのデータを保存する Amazon DynamoDB テーブル。
9. トークンオペレーション用のキーやその他の機密データを保存する AWS Secrets Manager。
10. (オプション) API Gateway で使用する AWS Identity and Access Management (IAM) ロールと Lambda オーソライザー関数で構成されるオーソライザーコンポーネント。
11. (オプション) 2 つのインレットストラテジーをサポートする Amazon Simple Notification Service (Amazon SNS)、CloudWatch、Lambda 関数。
12. (オプション) OpenID プロバイダーがウェブサイトに対してユーザーを認証できるようにする API Gateway と Lambda 関数を備えた OpenID アダプターコンポーネント。このコンポーネントの待合室ページ用の Amazon Simple Storage Service (Amazon S3) バケットを含む CloudFront ディストリビューション。
13. (オプション) サンプル待合室ウェブアプリケーション用の Amazon S3 オリジンバケットを含む CloudFront ディストリビューション。

ソリューションの仕組み

このセクションでは、AWS の Virtual Waiting Room のワークフローについて、概要を説明します。ウェブサイトに待合室を構築し、カスタマイズや統合を行う方法の詳細については、GitHub 上のデベロッパーガイドを参照してください。

待合室のパブリック API は、今あるサイトで構築されているセキュリティ境界の背後に配置することも、認証なしで使用することもできます。待合室をウェブサイトと統合するために使用するアプローチによっては、ユーザーが待合室に移動してキュー内の位置を取得する前に、まずウェブサイトに対する認証が必要になる場合があります。

待合室に入って他のリクエストを行うには、クライアントソフトウェアが Event ID を取得する必要があります。Event ID は、パブリック API とプライベート API に対するほとんどのリクエストに必要な一意の ID です。Event ID は、コア API スタックのインストール時に設定されます。オペレーション中、Event ID は、待合室ページを介して URL パラメータまたは Cookie として提供できます。認証トークンのリクエストの一部として提供することも、別の方法を利用してクライアントに配布することもできます。

特定の API コールを行うために、クライアントが Event ID と Request ID の両方を必要とする場合があります。Request IDは、順番を待つ個々のクライアントを表す一意の ID であり、待合室から発行されます。

次の手順では、まずキューに入り、キューの進行を待ち、ウェブサイトへのアクセストークンを取得して待合室から出るという API リクエストの流れを説明しています。

ユーザーが待合室に入室する:

1. 待合室の入口を表す画面またはページがユーザーに表示されます。ユーザーがキューに入ることを選択し、クライアントソフトウェア (ブラウザ、モバイル、デバイス) によって、キュー位置をリクエストする `assign_queue_num` パブリック API が呼び出されます。
2. API リクエストは、API Gateway によってすぐに Amazon SQS キューに配信されます。
3. `assign_queue_num` API コールは、リクエストがキューに配置されると応答を返します。クライアントは一意の Request ID を受け取ります。この値は、後でキューの位置、リクエストの時刻、アクセストークンを取得するのに使用できます。
4. `AssignQueueNum` Lambda 関数は、SQS キューから最大 10 件のリクエストのバッチを受け取ります。Lambda サービスは呼び出しをファンアウトして、複数バッチのリクエストを処理します。

5. AssignQueueNum Lambda 関数は、バッチ内の各メッセージを検証し、Elasticache (Redis OSS) のキューカウンターを増分して、各リクエストおよび関連するキュー位置を Elasticach (Redis OSS) に保存します。
6. 各メッセージは、正常に処理されると削除されます。エラー状態に関連するメッセージは、後のバッチで 1 回再処理されます。2 回目に失敗すると、これらは CloudWatch アラームに接続されたデッドレターキューに送信されます。
7. クライアントは、assign_queue_num コールから Request ID を受け取ると、queue_num API のポーリングを開始できます。クライアントは、Event ID と Request ID を queue_num API に送信し、キュー位置の数値か、リクエストがまだ処理されていないことを示すレスポンスを受け取ります。大規模なイベント中には、クライアントによるこのコールが複数回必要になることもあります。GetQueueNum Lambda 関数は、API Gateway によって呼び出され、DynamoDB からキュー内のクライアントの位置を数値で返します。

ユーザーが待合室で待機する:

8. クライアントは、キュー内の位置を取得した後、一定の間隔で serving_num API のポーリングを開始できます。serving_num API は Event ID で呼び出され、キューの現在の処理待ち順序を返します。serving_num API からのレスポンスは、クライアントに、待合室から最終的なトランザクションが実行される実際のターゲットサイトに移動できるタイミングを伝えます。GetServingNum Lambda 関数は、待合室での現在の処理待ち順序を返します。
9. 処理待ちの順序がクライアントのキュー (リクエスト) 位置以上であれば、クライアントはパブリック API から JSON Web Token (JWT) をリクエストできます。このトークンをターゲットサイトで使用して、トランザクションを完了することができます。generate_token API は、Event ID と Request ID を使用して呼び出されます。API Gateway は、パラメータを使用して GenerateToken Lambda 関数を呼び出します。
10. GenerateToken Lambda 関数は、リクエストを検証し、このトークンが以前に生成されたかどうかを確認します。Lambda 関数は、一致するトークンを DynamoDB テーブルで照会します。見つかった場合、そのトークンは呼び出し元に返され、再生成されません。このプロセスにより、1 つの Request ID で、新しい有効期限を持つ複数の異なるトークンが生成されることを防ぎます。
11. トークンが DynamoDB で見つからない場合、Lambda 関数はトークンを作成するためのキーを取得し、Event ID とクライアントの Request ID を使用して DynamoDB にトークンを保存します。Lambda 関数は EventBridge にイベントを書き込み、新しいトークンが生成されたことを通知します。Lambda 関数は、イベントに対して生成されたトークンの数を追跡する Elasticache (Redis OSS) カウンターを増分します。

12.queue_pos_expiry がオンになっている場合は、クライアントは GetQueuePositionExpiryTime Lambda 関数を呼び出す queue_pos_expiry API を呼び出すことで、有効期限が切れるまでの残り時間をクエリすることができます。

ユーザーが待合室から退出する:

13.クライアントは、トークンを受け取ると、ターゲットサイトに入りトランザクションを開始します。インフラストラクチャで JWT との統合がどのようにサポートされるかによって、クライアントはリクエストヘッダー、cookie、またはその他の方法でトークンを提示する必要があります。API Gateway のオーソライザーを使用すると、クライアントのリクエストに含まれるトークンを検証できます。Virtual Waiting Room on AWS のトークンは、JWT を検証および管理するための商用またはオープンソースのあらゆるライブラリで使用できます。トークンが有効であれば、クライアントはトランザクションを続行できます。
14.クライアントがトランザクションを完了した後、クライアントのトークンのステータスを更新するためにプライベート API が呼び出され、DynamoDB 内で完了となります。

キュー位置の有効期限:

15.この機能をアクティブにすると、特定のキュー位置に対応する Request ID は、指定された時間間隔でのみトークンを生成できます。

キュー位置の有効期限が切れるとサービングカウンターを増分する:

16.この機能をアクティブにすると、トークンを生成できなかった有効期限切れのキュー位置に基づいて、サービングカウンターが自動的に増分されます。

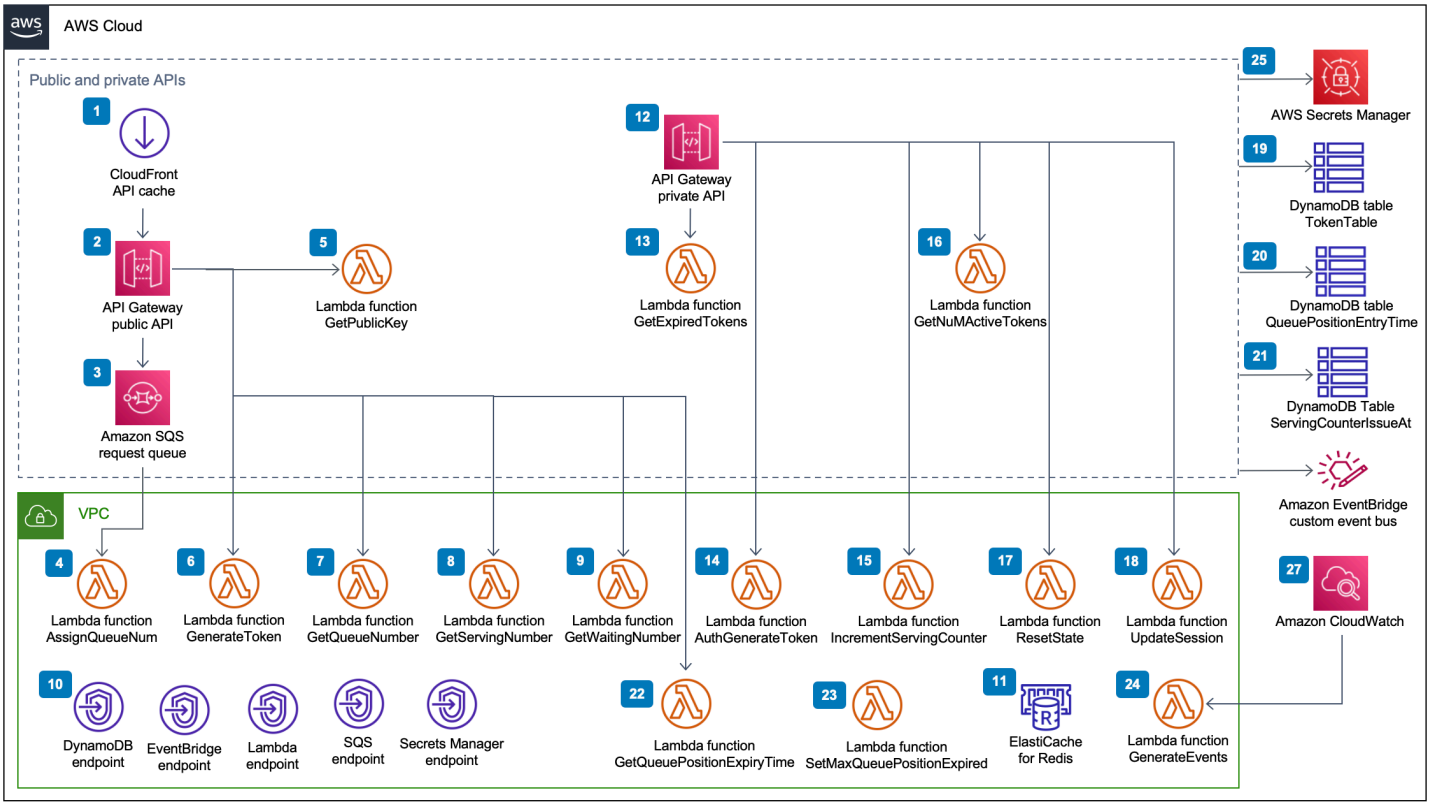
ソリューションのコンポーネント

待合室のパブリック API とプライベート API

Virtual Waiting Room on AWS ソリューションの主な目的は、クライアントに渡す JSON Web Token (JWT) の生成を制御して、新規ユーザーのアクセス急増により送信先のウェブサイトに過負荷がかかるのを防ぐことです。JWT は、待合室トークンを取得するまでの間、ウェブページへのアクセスを防止してサイトを保護するために利用されます。また、API アクセス認証にも利用できます。

コアテンプレートにより、Virtual Waiting Room on AWS のほとんどのオペレーションで使用されるパブリック API とプライベート (IAM で承認済み) API がインストールされます。パブリック API は、API のパスに基づく複数のキャッシュポリシーが設定された CloudFront ディストリビューションを使用して設定されます。DynamoDB テーブルと EventBridge イベントバスが作成されます。テンプレートは、新しい VPC を追加し、この VPC 内に 2 つのアベイラビリティーゾーン (AZ)、Elasticache (Redis OSS) クラスター (両方の AZ 内)、およびいくつかの Lambda 関数を含めます。Elasticache (Redis OSS) とやり取りする Lambda 関数は VPC 内のネットワークインターフェイスを使用し、その他すべての Lambda 関数はデフォルトのネットワーク接続を使用します。コア API は、ソリューションとのやり取りの最下層を担います。その他の Lambda 関数、Amazon Elastic Compute Cloud (Amazon EC2) インスタンス、およびコンテナは、拡張機能として使用できます。これらからコア API を呼び出して、待合室の構築、インレットトラフィックの制御、ソリューションから生成されたイベントへの対応を行うことができます。

さらに、コアスタックでは、Lambda 関数のすべてのエラーおよびスロットル条件に対するアラームと、各 API Gateway のデプロイに関する 4XX および 5XX ステータスコードのアラームが作成されます。



Virtual Waiting Room on AWS のパブリック API とプライベート API のコンポーネント

1. CloudFront ディストリビューションは、クライアント用にパブリック API コールを配信し、必要に応じて結果をキャッシュします。
2. Amazon API Gateway パブリック API は、仮想待合室からのキューリクエストを処理し、キューの位置を追跡して、ターゲットウェブサイトへのアクセスを許可するトークンの検証をサポートします。
3. SQS キューは、キューメッセージを処理する AWS Lambda 関数へのトラフィックを調整します。
4. `AssignQueueNum` Lambda 関数は、受信したバッチ内の各メッセージを検証し、Elasticache (Redis OSS) のキューカウンターを増分して、各リクエストおよび関連するキュー位置を Elasticache (Redis OSS) に保存します。
5. `GetPublicKey` Lambda 関数は、Secrets Manager からパブリックキー値を取得します。
6. `GenerateToken` Lambda 関数は、ターゲットサイトでのトランザクションの完了が許可された有効なリクエストに対して JWT を生成します。トークンの生成を示すイベントを、待合室のカスタムイベントバスに書き込みます。このリクエストに対して以前にトークンが生成されていれば、新しいトークンは生成されません。

7. GetQueueNumber Lambda 関数は、Elasticache (Redis OSS) からキュー内のクライアントの位置を数値で取得して返します。
8. GetServingNumber Lambda 関数は、Elasticache (Redis OSS) から、待合室で現在処理中の数を取得して返します。
9. GetWaitingNum Lambda 関数は、待合室で現在キューに入れられているものの、まだトークンが発行されていない数を返します。
10. VPC エンドポイントは、VPC 内の Lambda 関数がソリューション内のサービスと通信することを許可します。
11. Elasticache (Redis OSS) クラスターは、待合室に入るすべてのリクエストを有効な Event ID で保存します。また、キューに入れられたリクエストの数、現在処理中の数、生成されたトークンの数、完了したセッションの数、中止されたセッションの数などを示すカウンターも保存されます。
12. API Gateway のプライベート API リソースは、管理機能をサポートします。プライベート API は AWS IAM で認証されます。
13. GetExpiredTokens Lambda 関数は、有効期限切れのトークンを持つリクエスト ID のリストを返します。
14. AuthGenerateToken Lambda 関数は、ターゲットサイトでのトランザクションの完了が許可された有効なリクエストに対してトークンを生成します。コアスタックのデプロイ中に最初に設定されたトークンの発行者と有効期間は上書き可能です。トークンの生成を示すイベントを、待合室のカスタムイベントバスに書き込みます。このリクエストに対して以前にトークンが生成されていれば、新しいトークンは生成されません。
15. IncrementServingCounter Lambda 関数は、Elasticache (Redis OSS) に保存されている待合室のサービングカウンターに増分の値を加算します。
16. GetNumActiveTokens Lambda 関数は DynamoDB にクエリを実行して、まだ有効期限内で、トランザクションの完了に使用されておらず、中止とマークされていないトークンの数を確認します。
17. ResetState Lambda 関数は、Elasticache (Redis OSS) に保存されているすべてのカウンターをリセットします。また、TokenTable、QueuePositionEntryTime、ServingCounterIssuedAt の DynamoDB テーブルを削除して再作成します。さらに、CloudFront キャッシュの無効化を実行します。
18. UpdateSession Lambda 関数は、TokenTable DynamoDB テーブルに保存されているセッション (トークン) のステータスを更新します。セッションステータスは整数で示されます。ステータスが 1 に設定されたセッションは完了を示し、-1 は中止を示します。セッションの更新を示すイベントを、待合室のカスタムイベントバスに書き込みます。

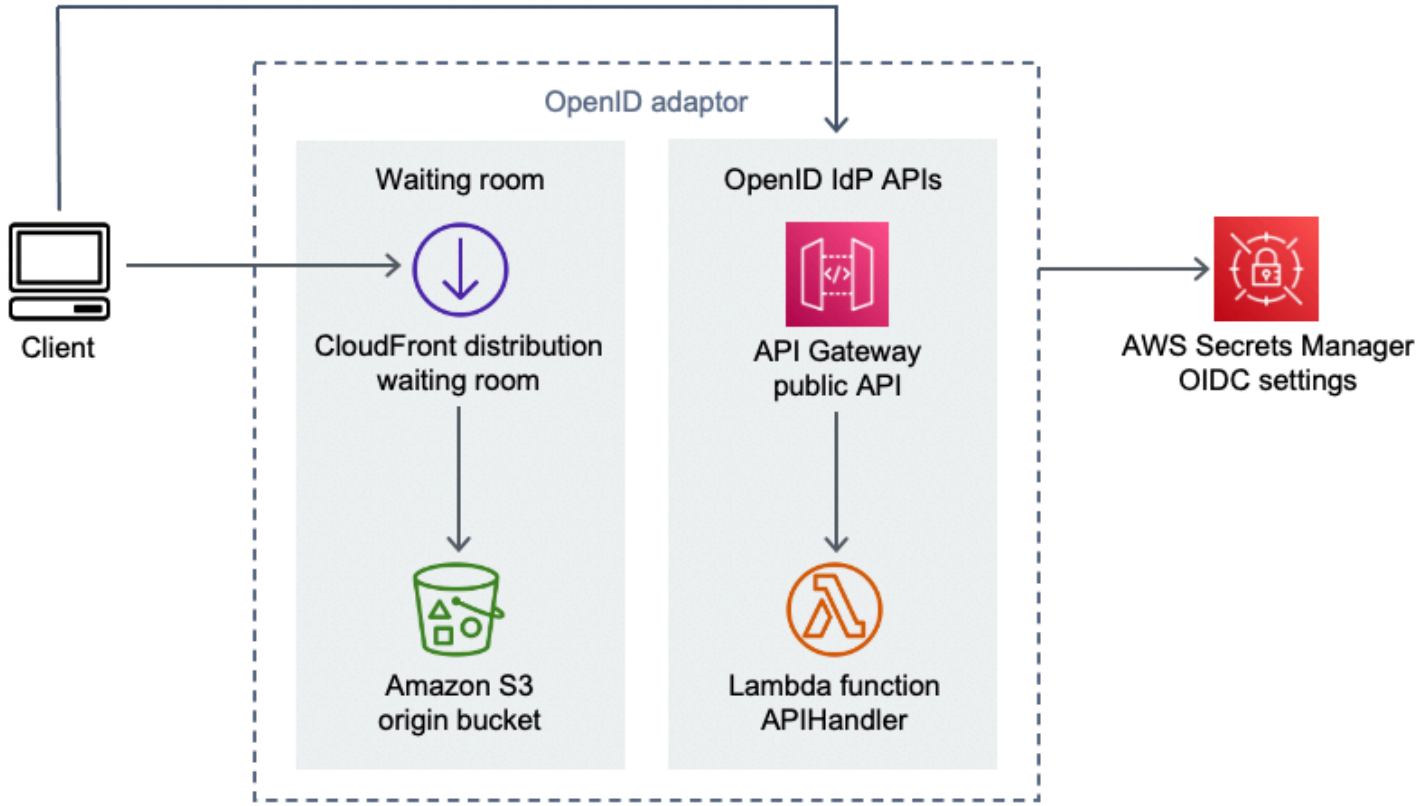
19. TokenTable DynamoDB テーブルには、トークンデータが保存されます。
20. QueuePositionEntryTime DynamoDB テーブルには、キューの位置とエントリ時間のデータが保存されます。
21. ServingCounterIssuedAt DynamoDB テーブルには、サービングカウンターの更新が保存されます。
22. GetQueuePositionExpireTime Lambda 関数は、クライアントがキュー位置の残りの有効期限をリクエストしたときに呼び出されます。
23. SetMaxQueuePositionExpired Lambda 関数は、ServingCounterIssuedAt テーブルの値に対応して有効期限切れになったキュー位置の最大値を設定します。コアスタックのデプロイ時に IncrSvcOnQueuePositionExpiry パラメータが true に設定されている場合は、1 分ごとに実行されます。
24. GenerateEvents Lambda 関数は、待合室のカスタムイベントバスに待合室のさまざまなメトリクスを書き込みます。コアスタックのデプロイ時に Enable Events Generation パラメータが true に設定されている場合は、1 分ごとに実行されます。
25. AWS Secrets Manager はトークンオペレーション用のキーやその他の機密データを保存します。
26. Amazon EventBridge カスタムイベントバスは、トークンが生成され、TokenTable DynamoDB テーブルでセッションが更新されるたびにイベントを受け取ります。また、サービングカウンターが SetMaxQueuePositionExpired Lambda で移動したときにもイベントを受け取ります。コアスタックのデプロイ時にアクティブになっている場合は、待合室のさまざまなメトリクスも書き込まれます。
27. Amazon CloudWatch イベントルールは、Enable Events Generation パラメータがコアスタックのデプロイ時に true に設定されている場合に作成されます。このイベントルールは、GenerateEvents Lambda 関数を 1 分ごとに起動します。

オーソライザー

このソリューションには、API Gateway の Lambda オーソライザースタックが含まれています。このスタックは、1 つの IAM ロールと 1 つの Lambda 関数で構成されています。APIGatewayAuthorizer Lambda 関数は、Virtual Waiting Room on AWS の API によって発行されたトークンの署名とリクエストを検証する API Gateway のオーソライザーです。スタックに付属する Lambda 関数を使用して、待合室で待機していたユーザーがアクセストークンを受け取るまで、クラウド API を保護することができます。オーソライザーは、トークン検証用にパブリックキーと設定をコア API から自動的に取得してキャッシュします。AWS Lambda をサポートする AWS リージョンであれば、変更することなく使用でき、インストールすることもできます。

OpenID アダプター

OpenID アダプターのスタックは、OpenID の ID プロバイダーとして機能する API Gateway と Lambda 関数をデプロイします。OpenID アダプターには OIDC 互換 API のセットが用意されており、AWS Elastic Load Balancing や WordPress などの OIDC の ID プロバイダーをサポートする既存のウェブホスティングソフトウェアで使用したり、Amazon Cognito や同様のサービスのフェデレーション ID プロバイダーとして使用したりすることもできます。このアダプターを使用すると、統合オプションが限られている市販のウェブホスティングソフトウェアを使用する場合でも、AuthN/AuthZ のフローで待合室を使用できます。このスタックでは、1 つの Amazon S3 バケットをオリジンとして、もう 1 つの Amazon S3 バケットをロギングリクエストとして使用する CloudFront ディストリビューションもインストールされます。OpenID アダプターは、サンプル待合室スタックで提供されているものと似ていますが、OpenID 認証フロー用に設計されたサンプル待合室ページを提供します。認証を受けるプロセスには、待合室でキュー内の位置を取得して、処理待ち順序がクライアントのキュー位置と同じかそれ以上になるまで待機することが含まれています。OpenID の待合室ページはターゲットサイトにリダイレクトされ、OpenID API を使用してクライアントのトークンの取得とセッションの設定を完了します。このソリューションの API エンドポイントは、正式な OpenID Connect 1.0 フロー仕様に直接 (名前に対応して) マッピングされています。詳細については、「OpenID Connect Core 1.0 Authentication」を参照してください。



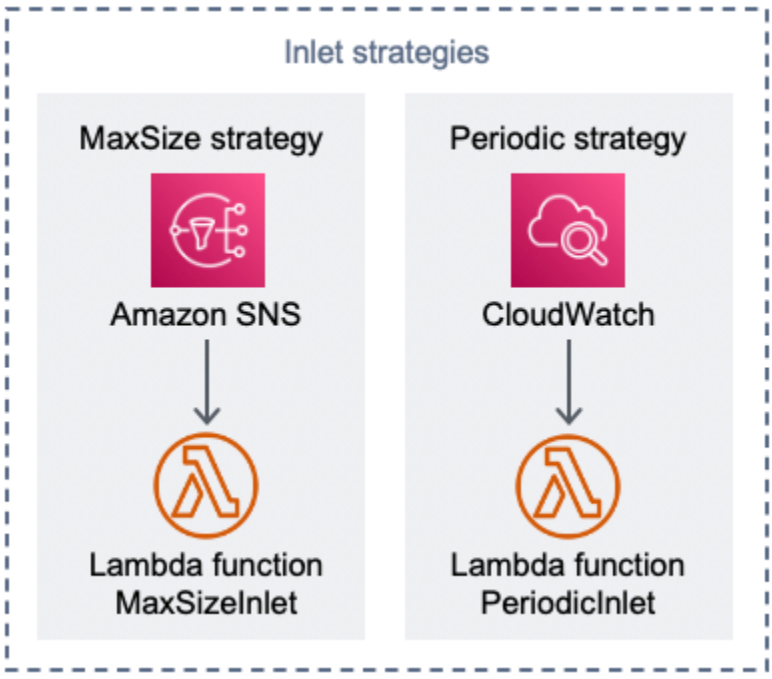
Virtual Waiting Room on AWS の OpenID アダプターのコンポーネント

1. CloudFront ディストリビューションは、S3 バケットのコンテンツをユーザーに提供します。
2. S3 バケットは、サンプル待合室のページをホストします。
3. Amazon API Gateway の API は、OIDC ID プロバイダーの Lambda オーソライザー機能をサポートする既存のウェブホスティングソフトウェアで使用できる OIDC 互換 API のセットを提供します。
4. `APIHandler` Lambda 関数は、すべての API Gateway リソースパスへのリクエストを処理します。同じモジュール内のさまざまな Python 関数が、各 API のパスにマッピングされています。例えば、API Gateway の `/authorize` リソースパスを指定すると、Lambda 関数内の `authorize()` が呼び出されます。
5. OIDC 設定は Secrets Manager に保存されます。

インレットストラテジーのサンプル

インレットストラテジーは、ターゲットサイトでより多くのユーザーに対応するために、ソリューションのサービングカウンターをいつ前に進めるべきかを決定します。待合室のインレットストラテジーに関する概念的な情報については、「設計上の考慮事項」を参照してください。

このソリューションには、サンプルとして 2 つのインレットストラテジー (MaxSize、Periodic) が用意されています。



Virtual Waiting Room on AWS のインレットストラテジーのコンポーネント

インレットストラテジーの MaxSize オプション:

1. クライアントは、`MaxSizeInlet` Lambda 関数を呼び出す Amazon SNS 通知を発行して、メッセージペイロードに基づいてサービングカウンターに増分を加算します。
2. `MaxSizeInlet` Lambda 関数は、サービングカウンターの増分量を決定するためのメッセージの受信を想定しています。

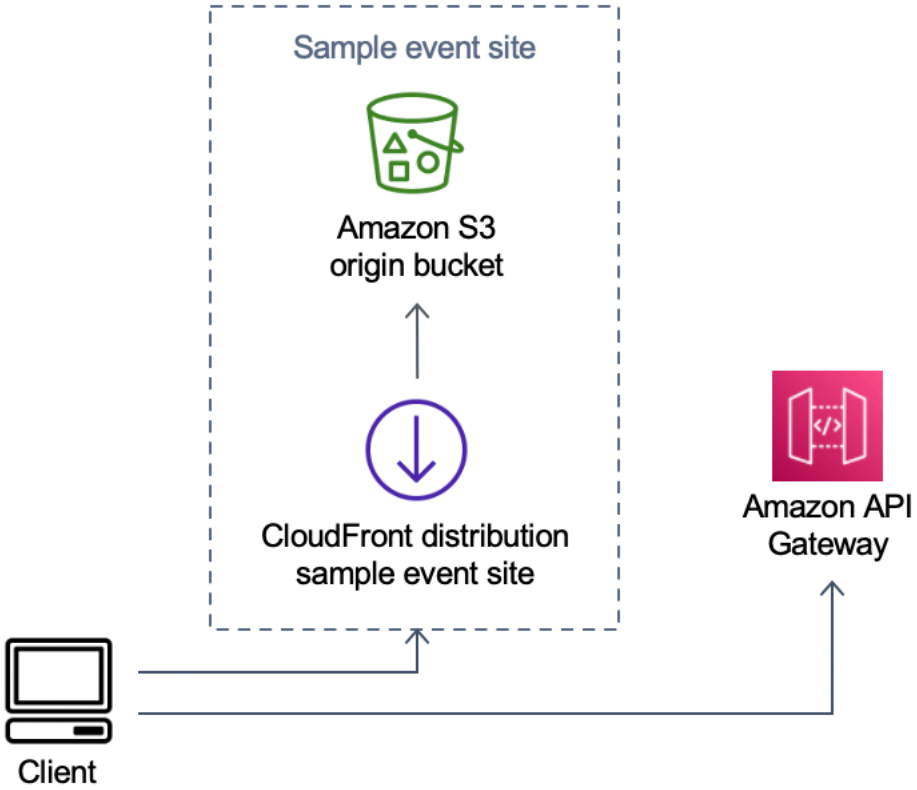
インレットストラテジーの Periodic オプション:

3. CloudWatch ルールは 1 分ごとに Lambda 関数を呼び出して、サービングカウンターを一定量増やします。
4. `PeriodicInlet` Lambda 関数は、指定された開始時刻から終了時刻までの間に時刻がある場合、指定された増分値をサービングカウンターに加算します。オプションで、CloudWatch アラームをチェックして、アラームが `OK` の状態であれば増分の加算を行い、それ以外の場合はスキップします。

サンプル待合室

サンプル待合室は、最小限の構成でエンドツーエンドの待合室ソリューションを実演するために、カスタムオーソライザーに加えて、パブリック API およびプライベート API と統合されています。S3 バケットに保存されているメインのウェブページは、CloudFront のオリジンとして使用されます。ユーザーは次の手順を実行するように指示されます。

1. サイトに入るための待合室で列に並びます。
2. クライアントの列内での位置情報 (順番) を取得します。
3. 待合室の処理待ち順序を取得します。
4. 処理待ちの順序がクライアントの位置情報と同じかそれ以上になったら、トークンセットを取得します。
5. トークンを使用して、Lambda オーソライザーで保護された API を呼び出します。



Virtual Waiting Room on AWS のサンプルイベントサイトのコンポーネント

1. S3 バケットは、待合室とコントロールパネルのサンプルコンテンツをホストします。
2. CloudFront ディストリビューションは、S3 バケットのコンテンツをユーザーに配信します。
3. `/search` や `/checkout` などのショッピングサイトのようなリソースパスを備えた API Gateway のサンプルをデプロイします。この API はスタックによってインストールされ、トークンのオーソライザーを使用して設定されます。これは、待合室で API を保護する簡単な方法の例として意図されています。有効なトークンを提示するリクエストが Lambda に転送されますが、それ以外の場合はエラーが返されます。この API には、アタッチされた Lambda 関数からのレスポンス以外の機能はありません。

セキュリティ

AWS インフラストラクチャでシステムを構築すると、お客様と AWS の間でセキュリティ上の責任が分担されます。この責任共有モデルにより、AWS がホストオペレーティングシステムと仮想化レイヤーからサービスが運用されている施設の物理的なセキュリティに至るまでのコンポーネントを運用、管理、および制御するため、お客様の運用上の負担を軽減するのに役立ちます。AWS セキュリティの詳細については、「AWS クラウドセキュリティ」を参照してください。

Elasticache (Redis OSS) には、プライベート VPC 内のネットワークインターフェイスが割り当てられます。Elasticache (Redis OSS) とやり取りする Lambda 関数にも、VPC 内のネットワークインターフェイスが割り当てられます。他のすべてのリソースは、共有される AWS のネットワークスペースでネットワークに接続できます。他の AWS のサービスとやり取りする VPC インターフェイスを持つ Lambda 関数は、VPC エンドポイントを使用してこれらのサービスに接続します。

JSON Web Token の作成と検証に使用されるパブリックキーとプライベートキーは、デプロイ時に生成され、Secrets Manager に保存されます。Elasticache (Redis OSS) への接続に使用するパスワードも、デプロイ時に生成され、Secrets Manager に保存されます。プライベートキーと Elasticache (Redis OSS) のパスワードには、ソリューションのいずれの API からもアクセスできません。

パブリック API には、必ず CloudFront 経由でアクセスする必要があります。このソリューションでは、API Gateway の API キーが生成されます。これは、CloudFront の `x-api-key` カスタムヘッダーの値として使用されます。CloudFront がオリジンリクエストを行うときに、このヘッダーが含められます。詳細については、「Amazon CloudFront 開発者ガイド」の「オリジンリクエストにカスタムヘッダーを追加する」を参照してください。

プライベート API は、呼び出し時に AWS IAM 認証をリクエストするように設定されています。このソリューションでは、プライベート API を呼び出すための適切なアクセス許可を持つ、`ProtectedAPIGroup` という IAM ユーザーグループが作成されます。このグループに追加された IAM ユーザーには、プライベート API を呼び出す権限が付与されます。

ロールで使用される IAM ポリシーと、ソリューションによって作成されたさまざまなリソースにアタッチされるアクセス許可では、必要なタスクの実行に際して必要なアクセス許可のみが付与されます。

S3 バケット、SQS キュー、ソリューションによって生成された SNS トピックなどのリソースにおける保管時および転送中の暗号化については、可能な箇所はすべて有効化されます。

モニタリング

コア API のスタックには、ソリューションの使用中にモニタリングをして問題を検出できるように、CloudWatch アラームがいくつか含まれています。このスタックは、Lambda 関数のエラーとスロットル条件に対するアラームを作成します。1 分間にエラーまたはスロットル条件が発生した場合は、このアラームの状態を `OK` から `ALARM` に変更します。

また、このスタックは、各 API Gateway のデプロイに関する 4XX と 5XX のステータスコードのアラームも作成します。1 分間以内に API から 4XX または 5XX のステータスコードが返された場合、アラームの状態は `OK` から `ALARM` に変わります。

エラーやスロットルが発生せずに 1 分が経過すると、これらのアラームは `OK` の状態に戻ります。

IAM ロール

AWS Identity and Access Management (IAM) ロールを使用すると、AWS クラウドのサービスとユーザーに対してアクセスポリシーとアクセス許可をきめ細かく割り当てることができます。このソリューションでは、リージョンのリソースを作成するためのアクセス権をソリューションの AWS Lambda 関数に付与する IAM ロールが作成されます。

Amazon CloudFront

`virtual-waiting-room-on-aws.template` CloudFormation テンプレートでは、待合室のコアパブリック API とプライベート API を作成すると共に、パブリック API 用に CloudFront ディストリビューションもデプロイされます。CloudFront ではパブリック API からのレスポンスがキャッシュされるため、作業を実行する API Gateway および Lambda 関数の負荷が軽減されます。

このソリューションには、オプションのサンプル待合室のテンプレートも含まれています。このテンプレートでは、Amazon Simple Storage Service (Amazon S3) バケットでホストされる単純なウェブアプリケーションをデプロイできます。レイテンシーを軽減してセキュリティを向上させるために、Amazon CloudFront ディストリビューションはオリジンアクセスアイデンティティ (このソリューションのウェブサイトのバケットコンテンツにパブリックアクセスを提供するための CloudFront ユーザー) を使用してデプロイされます。詳細については、「Amazon CloudFront デベロッパーガイド」の「オリジンアクセス ID を使用して Amazon S3 コンテンツへのアクセスを制限する」を参照してください。

セキュリティグループ

このソリューションで作成される VPC のセキュリティグループは、Elasticache (Redis OSS) へのネットワークトラフィックを制御および分離するように設計されています。Elasticache (Redis OSS) と通信する必要がある Lambda は、Elasticache (Redis OSS) と同じセキュリティグループに配置されます。デプロイが完了し起動したら、セキュリティグループを確認し、必要に応じてアクセスをさらに制限することをお勧めします。

設計上の考慮事項

デプロイオプション

初めてインストールする場合や、何をインストールすればよいかわからない場合は、`virtual-waiting-room-on-aws-getting-started.template` でネストされた CloudFormation テンプレートをデプロイしてください。このテンプレートは、コア、オーソライザー、サンプル待合室のテンプレートをインストールします。これにより、単純なフローを持つ待合室を最小限の構成で構築できます。

サポートされるプロトコル

Virtual Waiting Room on AWS ソリューションは、次のものと統合できます。

- JSON Web Token の検証ライブラリとツール
- 既存の API Gateway のデプロイ
- REST API のクライアント
- OpenID のクライアントとプロバイダー

待合室のインレット (滞留) ストラテジー

インレットストラテジーとは、待合室からウェブサイトにクライアントを移動させるために必要なロジックとデータをカプセル化したものです。インレットストラテジーは、Lambda 関数、コンテナ、Amazon EC2 インスタンス、またはその他のコンピューティングリソースとして実装できます。待合室のパブリック API とプライベート API を呼び出すことができれば、クラウドリソースである必要はありません。インレットストラテジーでは、待合室、ウェブサイト、またはその他の外部インジケータに関するイベントを受け取ります。これらは、より多くのクライアントがトークンを受け取り、サイトに入るタイミングを決定するために役立ちます。インレットストラテジーには、複数のアプローチがあります。どちらを採用するかは、利用可能なリソースと、保護されるウェブサイトの設計上の制約によって異なります。

インレットストラテジーが使用する主なアクションは、さらにサイトに入ることができるクライアントの数を示す相対値を指定して、`increment_serving_num` Amazon API Gateway のプライベート API を呼び出すことです。このセクションでは、インレットストラテジーの 2 つのサンプルについて

説明します。これらは、そのまま使用することも、カスタマイズすることもでき、まったく異なるアプローチを採用することもできます。

MaxSize

MaxSize ストラテジーを使用する場合、Lambda の `MaxSizeInlet` 関数には、ウェブサイトを同時に使用できるクライアント数の最大値が設定されます。これは固定値です。クライアントは、`MaxSizeInlet` Lambda 関数を呼び出す Amazon SNS 通知を発行して、メッセージペイロードに基づいてサービングカウンターに増分を加算します。SNS メッセージのソースはさまざまな場所から取得できます。これには、ウェブサイトのコードや、サイトの使用レベルを監視するモニタリングツールなどが含まれます。

`MaxSizeInlet` Lambda 関数では、次を含むメッセージの受信を想定しています。

- `exited` ：完了したトランザクションの数
- 完了とマークするリクエスト ID のリスト
- 中止とマークされるリクエスト ID のリスト

このデータは、サービングカウンターに増分として加算する値を決定するために使用されます。現在の同時接続クライアント数によっては、カウンターに加算できる追加のキャパシティがない場合もあります。

Periodic

Periodic ストラテジーを使用する場合は、CloudWatch ルールにより、サービングカウンターに一定量の増分を加算するための `PeriodicInlet` Lambda 関数が 1 分ごとに呼び出されます。Periodic インレットは、イベントの開始時刻、終了時刻、増分量でパラメータ化されます。このストラテジーでは、オプションで CloudWatch アラームもチェックして、アラームが `OK` の状態であれば増分の加算を実行し、それ以外の場合はスキップします。サイトインテグレーターは、使用率メトリクスをアラームに接続し、そのアラームを使用して Periodic インレットを一時停止することもできます。このストラテジーでは、現在時刻が開始時刻から終了時刻までの範囲内にあり、オプションで指定されたアラームが `OK` の状態である間のみ、処理待ちの順序を変更します。

ソリューションのカスタマイズと拡張

組織のサイト管理者は、待合室で使用する統合方法を決定する必要があります。2 つのオプションがあります。

1. API および API Gateway のオーソライザーを直接使用する基本的な統合。
2. ID プロバイダーによる OpenID 統合。

上記の統合に加えて、ドメイン名のリダイレクトの設定が必要になる場合があります。また、カスタマイズした待合室のサイトページもデプロイする必要があります。

Virtual Waiting Room on AWS ソリューションは、EventBridge を利用したイベント通知 (単一方向) と、REST API を利用した通信 (双方向) という 2 つのメカニズムで拡張できるように設計されています。

クォータ

Virtual Waiting Room on AWS の主なスケール制限は、インストールされた AWS リージョンの Lambda のスロットル制限です。デフォルトの Lambda 同時実行クォータを使用して AWS アカウントにインストールした場合、Virtual Waiting Room on AWS ソリューションでは、キュー内の位置をリクエストするクライアントを 1 秒あたり最大 500 件まで処理できます。1 秒あたり 500 クライアントのレートは、すべての Lambda 関数の同時クォータ制限を独占的に利用できるソリューションに基づいています。アカウントのリージョンが、Lambda 関数を呼び出す他のソリューションと共有されている場合、Virtual Waiting Room on AWS ソリューションは、少なくとも 1,000 件以上の同時呼び出しができる必要があります。CloudWatch メトリクスを使用すると、アカウント内での Lambda の同時呼び出し数を時系列でグラフ化し、判断を下すことができます。Service Quotas コンソールを使用すると、上限の引き上げをリクエストできます。Lambda のスロットル制限を引き上げると、追加の呼び出しが実際に発生した場合にのみ、毎月のアカウントの利用料金が変わります。

1 秒あたり 500 クライアントを追加するごとに、スロットル制限を 1,000 ずつ引き上げます。

1 秒あたりの受信ユーザー数 (予想)	推奨される同時実行数のクォータ
0 ~ 500	1,000 (デフォルト)
501 ~ 1,000	2,000
1,001 ~ 1,500	3,000

Lambda には、3,000 件の同時呼び出しという固定されたバースト制限があります。詳細については、「Lambda 関数のスケーリングについて」を参照してください。クライアントコードは、一時的なスロットル状況を示すエラーコードが返される場合、いくつかの API コールを想定し、再試行す

る必要があります。サンプル待合室のクライアントには、大容量のイベント、および高いバーストが発生するイベントで使用されるクライアントの設計方法の例として利用できるコードが含まれています。

このソリューションでは、カスタムの設定手順により、Lambda の予約済み同時実行数およびプロビジョニングされた同時実行数にも対応できます。詳細については、「Lambda の予約済み同時実行数の管理」を参照してください。

待合室に入り、トークンを受け取り、トランザクションへと続行できるユーザーの上限数は、Elasticache (Redis OSS) カウンターの上限によって制限されます。これらのカウンターは、待合室の処理待ち順序とソリューションの状態サマリーの追跡に使用されます。Elasticache (Redis OSS) で使用するカウンターの上限は 9,223,372,036,854,775,807 です。待合室のユーザーに発行された各トークンのコピーを保存するためには、DynamoDB テーブルが使用されます。DynamoDB では、テーブルのサイズに実質的な制限はありません。

リージョンデプロイ

このソリューションで使用されるサービスは、すべての AWS リージョンでサポートされています。AWS のサービスのリージョンごとの最新情報については、「AWS リージョン別のサービスのリスト」を参照してください。

AWS CloudFormation テンプレート

デプロイを自動化するために、このソリューションでは次の AWS CloudFormation テンプレートが使用されており、デプロイ前にダウンロード可能です。

初めてインストールする場合や、何をインストールすればよいかわからない場合は、`virtual-waiting-room-on-aws-getting-started.template` AWS CloudFormation テンプレートをデプロイすることで、コア、オーソライザー、およびサンプル待合室のコードテンプレートがインストールされます。これにより、シンプルなフローで動作する待合室をテストできます。

View template virtual-waiting-room-on-aws-api-gateway-cw-logs-role.template: このテンプレートは、アカウントレベルで CloudWatch ログ記録のアクセス許可を行うために API Gateway にデフォルトロールの ARN を追加する場合に使用します。アカウントでこのテンプレートのデプロイが必要かどうかの詳細については、「前提条件」を参照してください。

View template virtual-waiting-room-on-aws-getting-started.template: このネストされたテンプレートは、コア、オーソライザー、およびサンプル待合室のスタックをインストールする場合に使用します。

View template virtual-waiting-room-on-aws.template: このコアテンプレートは、待合室のイベントを作成するためのパブリックおよびプライベートの REST API とクラウドサービスをインストールする場合に使用します。このテンプレートは、待合室の REST API、Elasticache (Redis OSS)、DynamoDB テーブルを必要とするアカウントおおびリージョンにインストールします。

View template virtual-waiting-room-on-aws-authorizers.template: このテンプレートは、待合室で発行されたトークンを検証することとエンドユーザーの API を保護することを目的とした Lambda オーソライザーをインストールするために使用します。コアスタックが必要です。このスタックをデプロイするためのパラメータとして、コアスタックからの一部の出力が必要になります。これはオプションのテンプレートです。

View template

virtual-waiting-room-on-aws-openid.template: このテンプレートは、オーソライザーインターフェイスと待合室を統合するために、OpenID ID プロバイダーをインストールするために使用します。コアスタックが必要です。このスタックをデプロイするには、コアスタックからの一部の出力が必要になります。これはオプションのテンプレートです。

View template

virtual-waiting-room-on-aws-sample-inlet-strategy.template: このテンプレートは、ターゲットサイトと待合室の間で使用することを目的としたインレットストラテジーのサンプルをインストールするために使用します。インレットストラテジーは、ターゲットサイトに入ることのできるユーザー数を増やすタイミングを決定するロジックの抽象化に役立ちます。コアスタックが必要です。このスタックをデプロイするには、コアスタックからの出力が必要です。これはオプションのテンプレートです。

View template

virtual-waiting-room-on-aws-sample.template: このテンプレートは、待合室およびターゲットサイトに必要な最小ウェブ設定のサンプルと API Gateway 設定をインストールするために使用します。コアスタックとオーソライザースタックが必要です。このスタックをデプロイするためのパラメータとして、コアスタックとオーソライザースタックからの出力が必要になります。これはオプションのテンプレートです。

自動化されたデプロイ

ソリューションを開始する前に、このガイドに記載されているコスト、アーキテクチャ、ネットワークセキュリティ、その他の考慮事項を確認してください。このセクションのステップバイステップの手順に従って、ソリューションを設定してアカウントにデプロイします。

デプロイ時間: 約 30 分 (getting-started スタックのみ)

前提条件

- Administrator Access に相当する AWS アカウントコンソールのアクセス許可。
- API Gateway から CloudWatch ログをアクティブ化します。
  - API Gateway コンソールにサインインし、スタックをインストールするリージョンを選択します。

    このリージョンに既存の API が定義されている場合:

    1. 任意の API を選択します。
    2. 左のナビゲーションから、[設定] を選択します。
    3. CloudWatch ログロールの ARN フィールドの値を確認します。
    - ARN がない場合は、`virtual-waiting-room-on-aws-api-gateway-cw-logs-role.template` をインストールします。
    - ARN がある場合は、「getting-started スタックの起動」から始めます。

      このリージョンに既存の API が定義されていない場合は、`virtual-waiting-room-on-aws-api-gateway-cw-logs-role.template` をインストールします。
- 保護するターゲットサイトのアーキテクチャと実装の詳細に関する知識。

デプロイの概要

次の手順を使用して、このソリューションを AWS にデプロイします。詳細な手順については、各ステップのリンクをクリックしてください。

Step 1. getting-started スタックを起動する

- AWS CloudFormation テンプレートを AWS アカウントに起動します。
- テンプレートパラメータを確認し、必要に応じてデフォルト値を入力または調整します。

Step 2. (オプション) 待合室をテストする

- IAM で保護された API を呼び出すための AWS キーを生成します。
- サンプル待合室のコントロールパネルを開きます。
- サンプル待合室をテストします。

ステップ 1. getting-started スタックを起動する

この自動化された AWS CloudFormation テンプレートは、コア、オーソライザー、サンプル待合室テンプレートをデプロイします。これにより、動作中の待機室を表示してテストできます。スタックを起動する前に、前提条件を読んで理解しておく必要があります。

> **Note**
>
> このソリューションの実行中に使用した AWS サービスのコストは、お客様の負担となります。詳細は、このガイドの「コスト」セクションに移動して、このソリューションで使用する各 AWS のサービスのウェブ料金ページを参照してください。

1. AWS マネジメントコンソールにサインインし、ボタンを選択すると、 `virtual-waiting-room-on-aws-getting-started.template` AWS CloudFormation テンプレートが起動します。

   Launch solution

   または、テンプレートをダウンロードして、独自にカスタマイズすることもできます。
2. テンプレートはデフォルトで米国東部 (バージニア北部) リージョンで起動します。別の AWS リージョンでソリューションを起動するには、コンソールのナビゲーションバーのリージョンセレクターを使用します。
3. [スタックの作成] ページで、正しいテンプレート URL が [Amazon S3 URL] テキストボックスに表示されていることを確認し、[次へ] を選択します。
4. [スタックの詳細を指定] ページで、ソリューションのスタックに名前を割り当てます。名前の文字数制限に関する詳細は、「AWS Identity and Access Management ユーザーガイド」の「IAM および AWS STS クォータ」を参照してください。
5. [パラメータ] で、このソリューションのテンプレートパラメータを確認し、必要に応じて変更します。このソリューションは以下のデフォルト値を使用します。

パラメータ	デフォルト	説明
Event ID	`Sample`	待合室のこのインスタンスに固有の ID (GUID 形式推奨)。
Validity Period	`3600`	トークンの有効期間 (秒単位)。
Enable Events Generation	`false`	`true` に設定すると、待合室に関連するメトリクスが 1 分ごとにイベントバスに書き込まれます。
Elasticache (Redis OSS) Port	`1785`	Elasticache (Redis OSS) サーバーへの接続に使用するポート番号。Elasticache (Redis OSS) のデフォルトポートである 6379 は、使用しないでください。
EnableQueuePositionExpiry	`true`	`false` に設定すると、キュー位置の有効期限は適用されません。
QueuePositionExpiryPeriod	`900`	キュー位置がトークンを生成できなくなるまでの秒単位の時間間隔です。
IncrSvcOnQueuePositionExpiry	`false`	`true` に設定すると、トークンを正常に生成できなかった有効期限切れのキュー位置に基づいて、サービングカウンターを自動的に進めます。

6. [Next] を選択します。
7. [スタックオプションの設定] ページで、[次へ] を選択します。

8. [確認] ページで、設定を確認して確定します。テンプレートで AWS Identity and Access Management (IAM) リソースを作成することを確認するチェックボックスをオンにします。
9. [スタックの作成] を選択してスタックをデプロイします。

   AWS CloudFormation コンソールの [ステータス] 列でスタックのステータスを表示できます。約 30 分で CREATE_COMPLETE ステータスが表示されます。

ステップ 2. (オプション) 待合室をテストする

getting-started スタックをデプロイした場合は、次の手順で待合室の機能をテストできます。テストを完了するには、コアスタックで、IAM で保護された API を呼び出すためのアクセス許可を持つ AWS キーが必要になります。

IAM で保護された API を呼び出すための AWS キーを生成する

1. `aws-virtual-waiting-room-getting-started.template` CloudFormation テンプレートがデプロイされた AWS アカウントで、IAM ユーザーを作成または使用します。
2. プログラムによるアクセス権を IAM ユーザーに付与します。IAM ユーザー用の新しいアクセスキーのセットを作成する場合は、表示されたときにキーファイルをダウンロードします。待合室をテストするには、IAM ユーザーのアクセスキー ID とシークレットアクセスキーが必要です。
3. テンプレートによって作成された ProtectedAPIGroup IAM ユーザーグループに IAM ユーザーを追加します。

サンプル待合室のコントロールパネルを開く

1. AWS CloudFormation コンソールにサインインし、ソリューションの getting-started スタックを選択します。
2. [出力] タブを選択します。
3. [キー] 列で [ControlPanelURL] を見つけ、対応する値を選択します。
4. 新しいタブまたはブラウザウィンドウでコントロールパネルを開きます。
5. コントロールパネルで [設定] セクションを展開します。
6. 「IAM で保護された API を呼び出すための AWS キーを生成する」で取得したアクセスキー ID とシークレットアクセスキーを入力します。エンドポイントとイベント ID は URL パラメータから入力されます。
7. [使用] を選択します。ボタンは認証情報の入力後にアクティブになります。

サンプル待合室をテストする

1. AWS CloudFormation コンソールで、ソリューションの getting-started スタックを選択します。
2. [出力] タブを選択します。
3. [キー] 列で [WaitingRoomURL] を見つけ、対応する値を選択します。
4. 待合室を開き、[予約] を選択して待合室に入ります。
5. コントロールパネルが表示されているブラウザタブに戻ります。
6. [サービングカウンターの増分] で、[変更] を選択します。これにより、100 人のユーザーが待合室からターゲットサイトに移動できます。
7. 待合室に戻り、[今すぐチェックアウト] を選択します。ターゲットサイトにリダイレクトされます。
8. [今すぐ購入] を選択して、ターゲットサイトでのトランザクションを完了します。

個別スタックのデプロイ

コアスタックは、待合室の主要な機能を取得するために必須となる唯一のスタックです。その他のスタックはすべてオプションです。待合室で発行されるトークンの検証や、既に使用している API を保護する方法がない場合は、オーソライザースタックを起動します。オーソライザーのインターフェイスと待合室の統合用に OpenID ID プロバイダーが必要である場合は、OpenID スタックを起動します。インレットストラテジースタックのサンプルには、保護対象のサイトに入ることのできるユーザー数を増やす方法とタイミングに関する例が含まれています。

1. コアスタックを起動する

デプロイ時間: 約 20 分

この自動化された AWS CloudFormation テンプレートを使用すると、Virtual Waiting Room on AWS を AWS クラウドにデプロイできます。スタックを起動する前に前提条件を完了しておく必要があります。

> **Note**
>
> このソリューションの実行中に使用した AWS サービスのコストは、お客様の負担となります。詳細は、このガイドの「コスト」セクションに移動して、このソリューションで使用する各 AWS のサービスのウェブ料金ページを参照してください。

1. AWS マネジメントコンソール にサインインし、ボタンを選択すると、`aws-virtual-waiting-room-on-aws.template` AWS CloudFormation テンプレートが起動します。

   Launch solution

   または、テンプレートをダウンロードして、独自にカスタマイズすることもできます。
2. テンプレートはデフォルトで米国東部 (バージニア北部) リージョンで起動します。別の AWS リージョンでソリューションを起動するには、コンソールのナビゲーションバーのリージョンセレクターを使用します。
3. [スタックの作成] ページで、正しいテンプレート URL が [Amazon S3 URL] テキストボックスに表示されていることを確認し、[次へ] を選択します。

4. [スタックの詳細を指定] ページで、ソリューションのスタックに名前を割り当てます。名前の文字数制限に関する詳細は、「AWS Identity and Access Management ユーザーガイド」の「IAM および AWS STS クォータ」を参照してください。
5. [パラメータ] で、このソリューションのテンプレートパラメータを確認し、必要に応じて変更します。このソリューションは以下のデフォルト値を使用します。

パラメータ	デフォルト	説明
Event ID	`Sample`	待合室のこのインスタンスに固有の ID (GUID 形式推奨)。
Validity Period	`3600`	トークンの有効期間 (秒単位)。
Enable Events Generation	`false`	`true` に設定すると、待合室に関連するメトリクスが 1 分ごとにイベントバスに書き込まれます。
Elasticache (Redis OSS) Port	`1785`	Elasticache (Redis OSS) サーバーへの接続に使用するポート番号。Elasticache (Redis OSS) のデフォルトポートである 6379 は、使用しないでください。
EnableQueuePositionExpiry	`true`	`false` に設定すると、キュー位置の有効期限は適用されません。
QueuePositionExpiryPeriod	`900`	キュー位置がトークンを生成できなくなるまでの秒単位の時間間隔です。
IncrSvcOnQueuePositionExpiry	`false`	`true` に設定すると、トークンを正常に生成できなかった有効期限切れのキュー位置に

パラメータ	デフォルト	説明
		基づいて、サービングカウンターを自動的に進めます。

6. [Next] を選択します。
7. [スタックオプションの設定] ページで、[次へ] を選択します。
8. [確認] ページで、設定を確認して確定します。テンプレートで AWS Identity and Access Management (IAM) リソースを作成することを確認するチェックボックスをオンにします。
9. [スタックの作成] を選択してスタックをデプロイします。

   スタックのステータスは、AWS CloudFormation コンソールの [ステータス] 列で表示できます。約 20 分で CREATE_COMPLETE のステータスが表示されます。

2. (オプション) オーソライザースタックを起動する

デプロイ時間: 約 5 分

1. AWS マネジメントコンソール にサインインし、ボタンを選択すると、 `aws-virtual-waiting-room-on-aws-authorizers.template` AWS CloudFormation テンプレートが起動します。

   Launch solution

   または、テンプレートをダウンロードして、独自にカスタマイズすることもできます。
2. テンプレートはデフォルトで米国東部 (バージニア北部) リージョンで起動します。別の AWS リージョンでソリューションを起動するには、コンソールのナビゲーションバーのリージョンセレクターを使用します。
3. [スタックの作成] ページで、正しいテンプレート URL が [Amazon S3 URL] テキストボックスに表示されていることを確認し、[次へ] を選択します。
4. [スタックの詳細を指定] ページで、ソリューションのスタックに名前を割り当てます。名前の文字数制限に関する詳細は、「AWS Identity and Access Management ユーザーガイド」の「IAM および AWS STS クォータ」を参照してください。
5. [パラメータ] で、このソリューションのテンプレートパラメータを確認し、必要に応じて変更します。このソリューションは以下のデフォルト値を使用します。

パラメータ	デフォルト	説明
Public API Endpoint	*<####>*	仮想待合室 API のパブリックエンドポイント。
Waiting Room Event ID	`Sample`	待合室のイベント ID。
Issuer URI	*<####>*	パブリックキーとトークンの発行者 URI。

6. [Next] を選択します。
7. [スタックオプションの設定] ページで、[次へ] を選択します。
8. [確認] ページで、設定を確認して確定します。テンプレートで AWS Identity and Access Management (IAM) リソースを作成することを確認するチェックボックスをオンにします。
9. [スタックの作成] を選択してスタックをデプロイします。

   スタックのステータスは、AWS CloudFormation コンソールの [ステータス] 列で表示できます。約 5 分で CREATE_COMPLETE ステータスが表示されます。

3. (オプション) OpenID スタックを起動する

デプロイ時間: 約 5 分

1. AWS マネジメントコンソール にサインインし、ボタンを選択すると、 `aws-virtual-waiting-room-on-aws-openid.template` AWS CloudFormation テンプレートが起動します。

   Launch solution

   または、テンプレートをダウンロードして、独自にカスタマイズすることもできます。
2. テンプレートはデフォルトで米国東部 (バージニア北部) リージョンで起動します。別の AWS リージョンでソリューションを起動するには、コンソールのナビゲーションバーのリージョンセレクターを使用します。
3. [スタックの作成] ページで、正しいテンプレート URL が [Amazon S3 URL] テキストボックスに表示されていることを確認し、[次へ] を選択します。

4. [スタックの詳細を指定] ページで、ソリューションのスタックに名前を割り当てます。名前の文字数制限に関する詳細は、「AWS Identity and Access Management ユーザーガイド」の「IAM および AWS STS クォータ」を参照してください。
5. [パラメータ] で、このソリューションのテンプレートパラメータを確認し、必要に応じて変更します。このソリューションは以下のデフォルト値を使用します。

パラメータ	デフォルト	説明
Public API Endpoint	<####>	仮想待合室 API のパブリックエンドポイントの URL。
Private API Endpoint	<####>	仮想待合室 API のプライベートエンドポイントの URL。
API Region	<####>	パブリックおよびプライベートの待合室 API の AWS リージョン名。
Event ID	Sample	待合室のイベント ID。

6. [Next] を選択します。
7. [スタックオプションの設定] ページで、[次へ] を選択します。
8. [確認] ページで、設定を確認して確定します。テンプレートで AWS Identity and Access Management (IAM) リソースを作成することを確認するチェックボックスをオンにします。
9. [スタックの作成] を選択してスタックをデプロイします。

   スタックのステータスは、AWS CloudFormation コンソールの [ステータス] 列で表示できます。約 5 分で CREATE_COMPLETE ステータスが表示されます。

4. (オプション) サンプルのインレットストラテジースタックを起動する

デプロイ時間: 約 2 分

1. AWS マネジメントコンソール にサインインし、ボタンを選択すると、`aws-virtual-waiting-room-sample-inlet-strategy.template` AWS CloudFormation テンプレートが起動します。

Launch solution ま

たは、テンプレートをダウンロードして、独自にカスタマイズすることもできます。

2. テンプレートはデフォルトで米国東部 (バージニア北部) リージョンで起動します。別の AWS リージョンでソリューションを起動するには、コンソールのナビゲーションバーのリージョンセレクターを使用します。
3. [スタックの作成] ページで、正しいテンプレート URL が [Amazon S3 URL] テキストボックスに表示されていることを確認し、[次へ] を選択します。
4. [スタックの詳細を指定] ページで、ソリューションのスタックに名前を割り当てます。名前の文字数制限に関する詳細は、「AWS Identity and Access Management ユーザーガイド」の「IAM および AWS STS クォータ」を参照してください。
5. [パラメータ] で、このソリューションのテンプレートパラメータを確認し、必要に応じて変更します。このソリューションは以下のデフォルト値を使用します。

パラメータ	デフォルト	説明
Event ID	`Sample`	待合室のイベント ID。
Private Core API Endpoint	`<####>`	仮想待合室 API のプライベートエンドポイントの URL。
Core API Region	`<####>`	コア API がインストールされている AWS リージョン。
Inlet Strategy	`Periodic`	デプロイするインレットストラテジー。`Periodic` では、サービング数に対して 1 分ごとに増分が加算されます。`MaxSize` では、特定の時間にダウンストリームのターゲットサイトで処理できるトランザクションの最大数に基づいて、サービング数に増分が加算されます。

パラメータ	デフォルト	説明
Increment By	*<####>*	サービングカウンターに 1 分ごとに加算する増分値。Periodic インレットストラテジーを選択した場合は必須です。
Start Time	*<####>*	サービング数への増分の加算を開始する時刻のタイムスタンプ (エポック秒)。Periodic インレットストラテジーを選択した場合は必須です。
End Time	*<####>*	サービング数への増分の加算を終了する時刻のタイムスタンプ (エポック秒)。0 のままにすると、サービング数への増分の加算は無限に行われます。Periodic インレットストラテジーを選択した場合は必須です。
CloudWatch Alarm Name	*<####>*	Periodic インレットストラテジーに関連付ける AmazonCloudWatch アラーム名 (オプション)。この値が指定され、アラーム状態になると、サービング数への増分の加算は行われません。Periodic インレットストラテジーにのみ適用されます。

パラメータ	デフォルト	説明
Max Size	<####>	ダウンストリームのターゲットサイトが一度に処理できるトランザクションの最大数 (MaxSize ストラテジーの場合)。

6. [Next] を選択します。
7. [スタックオプションの設定] ページで、[次へ] を選択します。
8. [確認] ページで、設定を確認して確定します。テンプレートで AWS Identity and Access Management (IAM) リソースを作成することを確認するチェックボックスをオンにします。
9. [スタックの作成] を選択してスタックをデプロイします。

   スタックのステータスは、AWS CloudFormation コンソールの [ステータス] 列で表示できます。約 2 分で CREATE_COMPLETE ステータスが表示されます。

5. (オプション) サンプルの待合室スタックを起動する

デプロイ時間: 約 5 分

1. AWS マネジメントコンソール にサインインし、ボタンを選択すると、`aws-virtual-waiting-room-sample.template` AWS CloudFormation テンプレートが起動します。

   Launch solution

   または、テンプレートをダウンロードして、独自にカスタマイズすることもできます。
2. テンプレートはデフォルトで米国東部 (バージニア北部) リージョンで起動します。別の AWS リージョンでソリューションを起動するには、コンソールのナビゲーションバーのリージョンセレクターを使用します。
3. [スタックの作成] ページで、正しいテンプレート URL が [Amazon S3 URL] テキストボックスに表示されていることを確認し、[次へ] を選択します。
4. [スタックの詳細を指定] ページで、ソリューションのスタックに名前を割り当てます。名前の文字数制限に関する詳細は、「AWS Identity and Access Management ユーザーガイド」の「IAM および AWS STS クォータ」を参照してください。

5. [パラメータ] で、このソリューションのテンプレートパラメータを確認し、必要に応じて変更します。このソリューションは以下のデフォルト値を使用します。

パラメータ	デフォルト	説明
API Gateway Region	`<####>`	API Gateway の AWS リージョン名。
Authorizer ARN	`<####>`	API Gateway の Lambda オーソライザーの ARN。
Event ID	`Sample`	待合室の Event ID。
Private API Endpoint	`<####>`	仮想待合室 API のプライベートエンドポイントの URL。
Public API Endpoint	`<####>`	仮想待合室 API のパブリックエンドポイントの URL。

6. [Next] を選択します。
7. [スタックオプションの設定] ページで、[次へ] を選択します。
8. [確認] ページで、設定を確認して確定します。テンプレートで AWS Identity and Access Management (IAM) リソースを作成することを確認するチェックボックスをオンにします。
9. [スタックの作成] を選択してスタックをデプロイします。

   スタックのステータスは、AWS CloudFormation コンソールの [ステータス] 列で表示できます。約 5 分で CREATE_COMPLETE ステータスが表示されます。

以前のバージョンからのスタックの更新

スタックを削除し、新しいバージョンの新しいスタックを作成することをお勧めします。現在、CloudFormation スタックの更新を使用した新しいバージョンへの移行はサポートされていません。「ソリューションをアンインストールする」、「getting-started スタックを起動する」を参照してください。

> **Note**
>
> 進行中のイベントをサポートするためにソリューションを積極的に使用していない場合は、新しいバージョンに移行することをお勧めします。

パフォーマンスデータ

Virtual Waiting Room on AWS の負荷テストには、Locust というツールを使用しました。シミュレート対象のイベントサイズは、10,000 ~ 100,000 クライアントです。負荷テストの環境は、次の設定で構成されていました。

- AWS クラウドのデプロイ用にカスタマイズされた Locust 2.x
- 4 つの AWS リージョン (us-west-1、us-west-2、us-east-1、us-east-2)
- 1 つのリージョンにつき 10 の c5.4xlarge Amazon EC2 ホスト (合計 40)
- 1 つのホストにつき 32 の Locust プロセス
- シミュレート対象のユーザーは 1,280 のプロセスに均等に分散

各ユーザーのプロセスで実施したエンドツーエンドの API テストの手順:

1. assign_queue_num を呼び出し、リクエスト ID を受け取る。
2. リクエスト ID で、ユーザーのキュー位置が返されるまで queue_num のループを実行する (短時間)。
3. 戻り値がユーザーのキュー位置以上になるまで serving_num のループを実行する (長時間)。
4. waiting_room_size を低頻度で呼び出して、待機中のユーザー数を取得する。
5. generate_token を呼び出して、ターゲットサイトで使用する JWT を受け取る。

結果

実質的には、待合室で処理できるクライアントの数に上限はありません。

ユーザーが待合室に入るレートは、デプロイされているリージョンにおける Lambda 関数の同時実行クォータに影響します。

ロードテストでは、CloudFront で使用されたキャッシュポリシーにより、デフォルトの API Gateway リクエスト制限である 1 秒あたり 10,000 リクエストを超えることはできませんでした。

get_queue_num Lambda 関数の呼び出しレートは、待合室に入るユーザーのレートと比較して、ほぼ 1 対 1 になります。この Lambda 関数は、同時実行数制限またはバースト制限により、入るユーザーのレートが高くなるとスロットリングされることがあります。get_queue_num Lambda 関数の呼び出し数が多いためにスロットリングが発生すると、副作用として他の Lambda 関数に影響する

可能性があります。このタイプの一時的なスケーリングエラーが生じた際においても、クライアントソフトウェアが再試行またはバックオフロジックで適切に対応できる場合では、システム全体の動作が継続されます。

コアスタックによって設定された CloudFront ディストリビューションでは、デフォルトのクォータ設定において、各ユーザーが少なくとも毎秒 `serving_num` API をポーリングする前提で、250,000 ユーザーを収容する待合室を処理できます。

トラブルシューティング

このセクションでは、このソリューションのトラブルシューティングについて説明します。

これらの手順で問題が解決しない場合は、「AWS サポートへのお問い合わせ」に、このソリューションに関する AWS サポートのケースを開く方法が記載されています。

API からの 4xx レスポンスステータス

- イベント ID またはリクエスト ID、あるいはその両方が正しくない場合に発生する可能性があります。これは、関連する Lambda 関数の CloudWatch Logs で発生します。
- プライベート API は IAM で認証済みであり、クライアントでは、プライベート API を呼び出すアクセス許可を持つ AWS キーが必要になります。これは API Gateway の CloudWatch Logs で発生します。

API からの 5xx レスポンスステータス

- スロットリングされた Lambda または API Gateway からのレスポンスです。*<LambdaFunctionName>*ThrottlesAlarm CloudWatch アラームを確認してください。
- バックエンドの設定ミスです。詳細については、*<LambdaFunctionName>*ErrorsAlarm CloudWatch アラームおよび CloudWatch Logs を確認してください。

5XXErrorPublic/PrivateApiAlarm

- API が 60 秒以内に呼び出し元に 5XX ステータスを返した場合、このアラーム状態は ALARM になります。
- 60 秒間 5xx ステータスが返されない場合、このアラームは OK に戻ります。
- このアラームは、API Gateway にエラーを返す Lambda 関数または Lambda ランタイムによって開始できます。

4XXErrorPublic/PrivateApiAlarm

- API が 60 秒以内に呼び出し元に 4XX ステータスを返した場合、このアラーム状態は ALARM になります。
- 60 秒間 4XX ステータスが返されない場合、このアラームは OK に戻ります。
- このアラームは、正しくない API URL によって発生する可能性があります。

*<LambdaFunctionName>*ThrottlesAlarm

- 指定の Lambda で 60 秒以内に同時実行数制限に到達した場合、このアラーム状態は ALARM になります。
- 60 秒間スロットリングが発生しなかった場合、このアラームは OK に戻ります。
- アカウントのリージョンにおいて、同時実行数の上限を引き上げる必要が生じる場合があります。
- Lambda のバースト制限が発生する場合もあります。これに対応するには、クライアントに再試行ロジックが必要になります。

*<LambdaFunctionName>*ErrorsAlarm

- 指定の Lambda で 60 秒以内にランタイム実行エラーが発生した場合、このアラーム状態は ALARM になります。
- 60 秒間スロットリングが発生しなかった場合、このアラームは OK に戻ります。
- このエラーは、バックエンドの設定ミスが原因で発生する可能性があります。
- このエラーは、Lambda のコードのバグが原因で発生する可能性があります。

AWS Support に問い合わせる

AWS デベロッパーサポート、AWS ビジネスサポート、または AWS エンタープライズサポートをご利用の場合は、サポートセンターを利用して、このソリューションに関する エキスパートのサポートを受けることができます。次のセクションで、その方法を説明します。

ケースの作成

1. サポートセンターにサインインします。
2. [ケースを作成] を選択します。

どのようなサポートをご希望ですか?

1. [技術] を選択します。
2. サービスで、[ソリューション] を選択します。
3. [カテゴリ] で、[その他のソリューション] を選択します。
4. 重要度で、ユースケースに最も適したオプションを選択します。

5. サービス、カテゴリ、重要度を入力すると、インターフェースに一般的なトラブルシューティングの質問へのリンクが表示されます。これらのリンクを使用しても問題を解決できない場合は、[次のステップ: 追加情報] を選択します。

追加情報

1. 件名に、質問または問題を要約したテキストを入力します。
2. 説明に、問題の詳細を入力します。
3. [ファイルを添付] を選択します。
4. AWS Support がリクエストを処理するために必要な情報を添付します。

ケースの迅速な解決にご協力ください

1. 必要な情報を記入します。
2. [次のステップ: 今すぐ解決またはお問い合わせ] を選択します。

今すぐ解決またはお問い合わせ

1. 今すぐ解決の解決策を確認します。
2. これらの解決策で問題を解決できない場合は、[お問い合わせ] を選択し、必要な情報を入力して [送信] を選択します。

その他のリソース

AWS サービス	
• AWS CloudFormation	• Amazon DynamoDB
• Amazon Simple Storage Service	• Amazon API Gateway
• AWS Lambda	• AWS Secrets Manager
• Amazon CloudFront	• Amazon Simple Queue Service
• Amazon EventBridge	• Amazon CloudWatch
• Elasticache (Redis OSS)	• Amazon Comprehend
• Amazon Virtual Private Cloud	• AWS Identity and Access Management

ソリューションをアンインストールする

Virtual Waiting Room on AWS ソリューションは、AWS マネジメントコンソールから、または AWS Command Line Interface を使用してアンインストールできます。このソリューションによって作成されたさまざまなリソースによってログの保存に使用された S3 バケットは、手動で削除する必要があります。AWS ソリューション実装では、これらの S3 バケットが自動的に削除されないため、ソリューションを削除した後も、ログイベントを確認することができます。

ソリューションによって作成された `ProtectedAPIGroup` IAM ユーザーグループに IAM ユーザーを手動で追加した場合は、ソリューションをアンインストールする前に IAM ユーザーグループから IAM ユーザーを削除します。この手順に従わないと、IAM ユーザーグループおよび関連付けられた IAM ポリシーを削除できません。

デプロイしたスタックごとに、次の手順に従ってください。

AWS Management Console の使用

1. AWS CloudFormation コンソール にサインインします。
2. [スタック] ページで、このソリューションのインストールスタックを選択します。
3. [削除] を選択します。

AWS Command Line Interface の使用

AWS Command Line Interface (AWS CLI) が環境で使用可能かどうかを判断します。インストール手順については、「AWS CLI ユーザーガイド」の「AWS Command Line Interface とは」を参照してください。AWS CLI が使用可能なことを確認後、以下のコマンドを実行します。

```
$ aws cloudformation delete-stack --stack-name <installation-stack-name>
```

Amazon S3 バケットの削除

このソリューションでは、AWS CloudFormation スタックを削除して、誤ってデータを損失しないようにするために、このソリューションで作成された Amazon S3 バケット (オプトインリージョンへのデプロイ用) を保持するように設定されています。ソリューションをアンインストールした後に、データを保持する必要がない場合は、S3 バケットを手動で削除できます。Amazon S3 バケットを削除するには、次の手順に従います。

1. Amazon S3 コンソール にサインインします。
2. 左側のナビゲーションペインで、[バケット] を選択します。
3. *<stack-name>* S3 バケットを探します。
4. S3 バケットを選択し、続いて [削除] を選択します。

AWS CLI を使用して S3 バケットを削除するには、以下のコマンドを実行します。

```
$ aws s3 rb s3://<bucket-name> --force
```

ソースコード

GitHub リポジトリにアクセスして、このソリューションのソースファイルをダウンロードし、カスタマイズを他のユーザーと共有できます。

寄稿者

- Jim Thario
- Thyag Ramachandran
- Joan Morgan
- Justin Pirtle
- Allen Moheimani
- Garvit Singh
- Bassem Wanis

リビジョン

日付	変更
2021 年 11 月	初回リリース
2022 年 9 月	バージョン 1.1: 期限切れのキュー位置に基づく、サービングカウンターの自動増分加算。Elasticache (Redis OSS) の使用状況の一部を DynamoDB に再配置。キュー位置の残りの有効期限を取得するためのパブリック API エンドポイント。詳細については、GitHub リポジトリ内の CHANGELOG.md ファイルを参照してください。
2023 年 4 月	バージョン 1.1.1: すべての新しい S3 バケットの S3 オブジェクト所有権の新しいデフォルト設定 (ACL 無効) による影響を軽減。詳細については、GitHub リポジトリ内の CHANGELOG.md ファイルを参照してください。
2023 年 11 月	バージョン 1.1.2: セキュリティの脆弱性を解決するためにパッケージバージョンを更新。詳細については、GitHub リポジトリ内の CHANGELOG.md ファイルを参照してください。
2024 年 3 月	バージョン 1.1.3: 次の 3 つの問題に対応: 待合室のサイズ内に期限切れのキュー位置が残る、リセット後も queue_num API が古い結果を返す、OpenID アダプターの `/userInfo` API で断続的な障害が発生する。詳細については、GitHub リポジトリ内の CHANGELOG.md ファイルを参照してください。
2024 年 4 月	バージョン 1.1.4: セキュリティの脆弱性を解決するためにパッケージバージョンを更

日付	変更
	新。詳細については、GitHub リポジトリ内の CHANGELOG.md ファイルを参照してください。
2024 年 6 月	バージョン 1.1.5: セキュリティの脆弱性を解決するためにパッケージバージョンを更新。詳細については、GitHub リポジトリ内の CHANGELOG.md ファイルを参照してください。
2024 年 8 月	バージョン 1.1.6: セキュリティの脆弱性を解決するためにパッケージバージョンを更新。詳細については、GitHub リポジトリ内の CHANGELOG.md ファイルを参照してください。
2024 年 8 月	バージョン 1.1.7: セキュリティの脆弱性を解決するためにパッケージバージョンを更新。詳細については、GitHub リポジトリ内の CHANGELOG.md ファイルを参照してください。
2024 年 9 月	バージョン 1.1.8: セキュリティの脆弱性を解決するためにパッケージバージョンを更新。詳細については、GitHub リポジトリ内の CHANGELOG.md ファイルを参照してください。
2024 年 11 月	バージョン 1.1.9: セキュリティの脆弱性を解決するためにパッケージバージョンを更新。詳細については、GitHub リポジトリ内の CHANGELOG.md ファイルを参照してください。

日付	変更
2024 年 11 月	バージョン 1.1.10: セキュリティの脆弱性を解決するためにパッケージバージョンを更新。詳細については、GitHub リポジトリ内の CHANGELOG.md ファイルを参照してください。

注意

お客様は、本書に記載されている情報を独自に評価する責任を負うものとします。本書は、(a) 情報提供のみを目的とし、(b) AWS の現行製品と慣行について説明しており、これらは予告なしに変更されることがあり、(c) AWS およびその関連会社、サプライヤー、またはライセンサーからの契約上の義務や保証をもたらすものではありません。AWS の製品やサービスは、明示または黙示を問わず、一切の保証、表明、条件なしに「現状のまま」提供されます。お客様に対する AWS の責任は、AWS 契約により規定されます。本書は、AWS とお客様の間で行われるいかなる契約の一部でもなく、そのような契約の内容を変更するものでもありません。

Virtual Waiting Room on AWS は、Apache ライセンスバージョン 2.0 の条件に基づいてライセンス提供されます。