ユーザーガイド

AWS フォールトインジェクションサービス

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

AWS フォールトインジェクションサービス: ユーザーガイド

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

Amazon の商標とトレードドレスは、Amazon 以外の製品またはサービスとの関連において、顧客に混乱を招いたり、Amazon の名誉または信用を毀損するような方法で使用することはできません。Amazon が所有しない他の商標はすべてそれぞれの所有者に帰属します。所有者は必ずしも Amazon との提携や関連があるわけではありません。また、Amazon の支援を受けているとはかぎりません。

Table of Contents

AWS Fault Injection Service とは

AWS Fault Injection Service (AWS FIS) は、 AWS ワークロードでフォールトインジェクション実験を実行できるようにするマネージドサービスです。フォールトインジェクションは、カオス工学の原則に基づいています。これらの実験では、アプリケーションの応答を観察できるように、破壊的なイベントを作成することで、アプリケーションに負荷をあたえます。その後、この情報を使用して、アプリケーションのパフォーマンスと復元力を向上させ、期待どおりに動作させることができます。

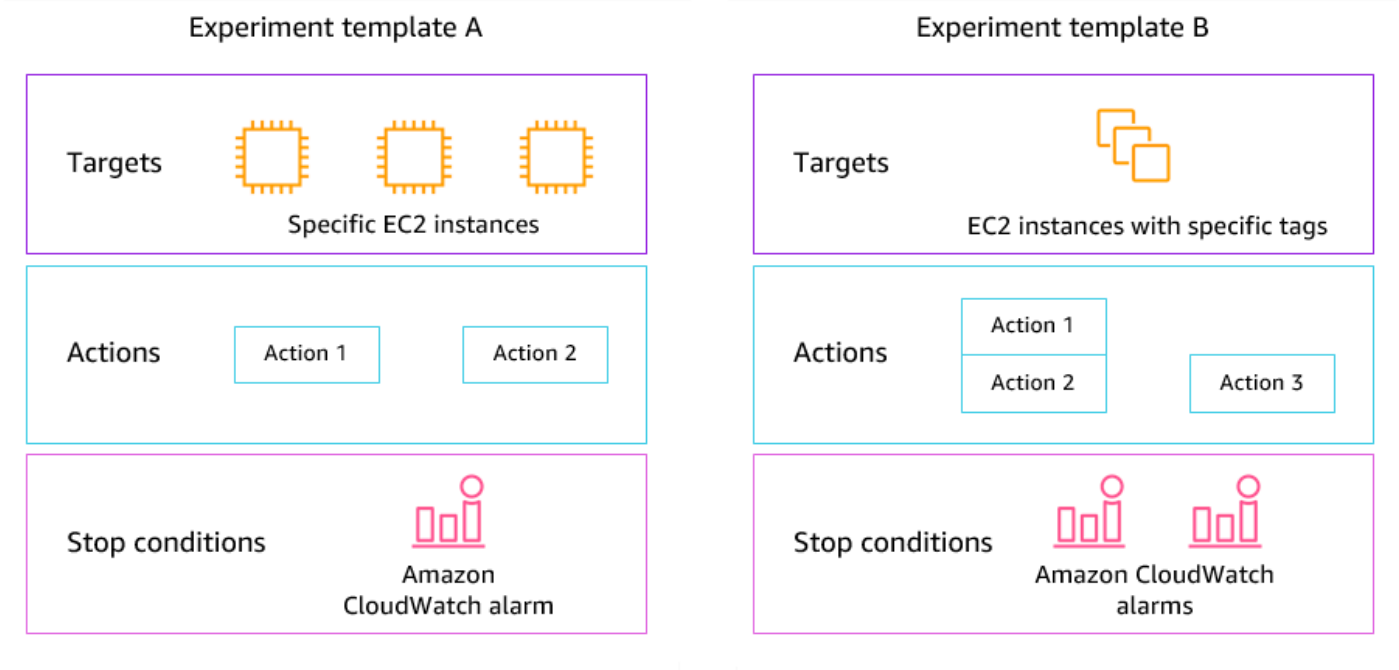
AWS FIS を使用するには、実際に見つけるのが難しいアプリケーションの問題を発見するために必要な条件を作成するのに役立つ実験をセットアップして実行します。 AWS FIS は、中断を生成するテンプレートと、特定の条件が満たされた場合に自動的にロールバックや実験の停止など、本番環境で実験を実行するために必要なコントロールとガードレールを提供します。

> **Important**
>
> AWS FIS は、システム内の実際の AWS リソースに対して実際のアクションを実行します。したがって、 AWS FIS を使用して本番環境で実験を実行する前に、計画フェーズを完了し、実稼働前の環境で実験を実行することを強くお勧めします。

実験の計画の詳細については、「信頼性のテスト」と「AWS FIS 実験の計画」を参照してください。 AWS FIS の詳細については、AWS 「Fault Injection Service」を参照してください。

AWS FIS の概念

AWS FIS を使用するには、 AWS リソースで実験を実行して、障害条件下でアプリケーションまたはシステムがどのように動作するかの理論をテストします。実験を実行するには、まず実験テンプレートを作成します。実験テンプレートは、実験の青写真です。実験テンプレートには、実験のアクション、ターゲット、および停止条件が含まれています。作成した実験テンプレートは、実験の実行に使用できます。実験の実行中に、その進行状況を追跡し、そのステータスを表示できます。実験は、実験内のすべてのアクションが実行された時点で完了します。



アクション

アクションは、実験中に AWS FIS が AWS リソースに対して実行するアクティビティです。 AWS FIS は、 AWS リソースのタイプに基づいて事前設定された一連のアクションを提供します。各アクションは、実験中、または実験を停止するまで、指定された期間実行されます。アクションは、順番に、または同時に (並行して) 実行できます。

ターゲット

ターゲットは、実験中に AWS FIS がアクションを実行する 1 つ以上の AWS リソースです。特定のリソースを選択することも、タグや状態などの特定の基準に基づいてリソースのグループを選択することもできます。

停止条件

AWS FIS は、 AWS ワークロードで実験を安全に実行するために必要なコントロールとガードレールを提供します。停止条件は、Amazon CloudWatch アラームとして定義したしきい値に達した場合に実験を停止するメカニズムです。実験の実行中に停止条件がトリガーされると、 AWS FIS は実験を停止します。

サポート対象 AWS のサービス

AWS FIS は、 のサービス全体で AWS 特定のタイプのターゲットに対して事前設定されたアクションを提供します。 AWS FIS は、次の のターゲットリソースのアクションをサポートします AWS のサービス。

- Amazon CloudWatch
- Amazon DynamoDB
- Amazon EBS
- Amazon EC2
- Amazon ECS
- Amazon EKS
- Amazon ElastiCache
- Amazon RDS
- Amazon S3
- AWS Systems Manager
- Amazon VPC

シングルアカウント実験の場合、ターゲットリソースは実験 AWS アカウント と同じ にある必要があります。 AWS FIS マルチアカウント実験を使用して、別の AWS アカウント アカウントのリソースをターゲットとする AWS FIS 実験を実行できます。

詳細については、「のアクション AWS FIS」を参照してください。

AWS FIS へのアクセス

AWS FIS は、次のいずれかの方法で操作できます。

- AWS Management Console — AWS FIS へのアクセスに使用できるウェブインターフェイスを提供します。詳細については、「AWS Management Consoleの操作」を参照してください。
- AWS Command Line Interface （ AWS CLI ） — AWS FIS を含む幅広い AWS のサービス用のコマンドを提供し、Windows、macOS、Linux でサポートされています。詳細については、「AWS Command Line Interface」を参照してください。 AWS FIS のコマンドの詳細については、「 コマンドAWS CLI リファレンス」の「fis」を参照してください。

- AWS CloudFormation — AWS リソースを記述するテンプレートを作成します。テンプレートを使用すると、これらのリソースを単一のユニットとして提供および管理できます。詳細については、AWS Fault Injection Service のリソースタイプのリファレンスを参照してください。
- AWS SDKs — 言語固有の APIs を提供し、署名の計算、リクエストの再試行処理、エラー処理など、接続に関する多くの詳細を処理します。詳細については、AWS SDK を参照してください。
- HTTPS API — HTTPS リクエストを使用して呼び出す低レベル API アクションを提供します。詳細については、「AWS Fault Injection Service API リファレンス」を参照してください。

AWS FIS の料金

実験のターゲットアカウントの数に基づき、アクションの実行開始から終了まで 1 分ごとに課金されます。詳細については、「AWS FIS の料金」を参照してください。

AWS FIS 実験の計画

フォールトインジェクションは、サーバー停止や API スロットリングなどの破壊的なイベントを作成することで、テスト環境や本番環境でアプリケーションに負荷をあたえるプロセスです。システムの応答を観察することで、改善を実装できます。システム上で実験を実行すると、システムに依存している顧客に影響を与える前に、システム上の弱点を制御された方法で特定するのに役立ちます。そうすれば、問題をプロアクティブに解決して、予測不可能な結果を防ぐことができます。

AWS FIS を使用したフォールトインジェクション実験を開始する前に、次の原則とガイドラインの内容をよく理解しておいてください。

> **Important**
>
> AWS FIS は、システム内の実際の AWS リソースで実際のアクションを実行します。したがって、AWS FIS で実験を実行する前に、まず、プリプロダクション環境またはテスト環境で計画段階とテストを必ず完了しておいてください。

内容

基本原則とガイドライン

AWS FIS で実験を始める前に、以下の手順を済ませておいてください。

1. 実験のターゲット展開を特定する — まず、ターゲットデプロイを特定します。これが最初の実験の場合は、プリプロダクションまたはテスト環境で開始することをお勧めします。
2. アプリケーションアーキテクチャを確認する: 各コンポーネントのすべてのアプリケーションコンポーネント、依存関係、およびリカバリ手順を特定していることを確認する必要があります。まず、アプリケーションアーキテクチャを見直します。アプリケーションによっては、「AWS Well-Architected フレームワーク」を参照してください。
3. 定常状態の動作を定義する - レイテンシー、CPU 負荷、1 分あたりの失敗したサインイン、再試行回数、ページ読み込み速度など、重要な技術的およびビジネス指標の観点から、システムの定常状態の動作を定義します。

4. 仮説を形成する — 実験中にシステムの動作がどのように変化すると予想されるかについての仮説を作成します。仮説の定義は次の形式に従います。

   ##################を実行した場合、######################は#を超えないものとします。

   例えば、認証サービスの仮説を次のように設定します。ネットワーク遅延が 10% 増加すると、サインイン失敗が 1% 未満増加する。実験の完了後、アプリケーションの復元力がビジネスおよび技術的な期待に沿っているかどうかを評価します。

AWS FIS を使用するときは、次のガイドラインに従うことをお勧めします。

- 常にテスト環境で AWS FIS を使用した実験を始めてください。決して本番環境では開始しないでください。フォールトインジェクション実験を進めていくと、テスト環境以外の制御環境でも実験できるようになります。
- アプリケーションの復元力に対するチームの自信を構築するには、以下を実行するなど、小規模で簡単な実験から始めましょう。ターゲットに対する aws:ec2:stop-instances アクション。
- フォールトインジェクションは、実際の問題を引き起こす可能性があります。慎重に進み、顧客が影響を受けないように、最初のフォールトインジェクションがテストインスタンス上にあることを確認してください。
- テスト、テスト、テストを繰り返します。フォールトインジェクションは、十分に計画された実験で制御された環境で実装されることを意図しています。これにより、乱流条件に耐えるアプリケーションやツールの能力に自信を持たせることができます。
- 始める前に、優れた監視およびアラートプログラムを用意することを強くお勧めします。それがなければ、持続可能なフォールトインジェクションの実践に不可欠な実験の影響を理解したり測定したりすることはできません。

実験計画ガイドライン

AWS FIS を使用すれば、AWS リソースで実験を実行して、障害条件下でアプリケーションやシステムがどのように動作するかの理論をテストすることができます。

次に示したのは、AWS FIS 実験の計画に推奨されるガイドラインです。

- 停止履歴の確認 - システムの以前の停止とイベントを確認します。これは、システムの全体的な健全性と回復力を把握するのに役立ちます。システムで実験を実行する前に、システムの既知の問題と弱点に対処する必要があります。

- 最も大きな影響を持つサービスを特定する - サービスを確認し、エンドユーザーまたは顧客に障害が発生した場合や正常に機能しない場合に、最も大きな影響を与えるサービスを特定します。
- ターゲットシステムを特定する — ターゲットシステムは、実験を実行するシステムです。AWS FIS を初めて使用する場合、または、これまでにフォールトインジェクション実験を実行したことがない場合は、プリプロダクションまたはテストシステムで実験を実行することをお勧めします。
- チームに相談する — 彼らが心配しているものを聞いてください。仮説を立てて、彼らの懸念を証明または反証することができます。また、チームに心配していないことを聞くこともできます。この質問は、2つのよくある誤謬を明らかにすることができます。サンクコスト誤謬と確認バイアスの誤謬です。チームの回答に基づいて仮説を形成すると、システムの状態の現実に関する詳細情報を提供できます。
- アプリケーションアーキテクチャを確認する - システムまたはアプリケーションのレビューを実施し、すべてのコンポーネントのすべてのアプリケーションコンポーネント、依存関係、およびリカバリ手順を特定していることを確認します。

  「AWSWell-Architected フレームワーク」を確認することをお勧めします。このフレームワークは、アプリケーションとワークロードのために、安全で、高パフォーマンス、耐障害性、および効率的なインフラストラクチャを構築するのに役立ちます。詳細については、「AWS Well-Architected」を参照してください。
- 該当するメトリクスを特定する — Amazon CloudWatch メトリクスを使用して、 AWSリソースに対する実験の影響をモニタリングできます。これらのメトリクスを使用して、アプリケーションが最適に実行されているときのベースラインまたは「定常状態」を判断できます。その後、実験中または実験後にこれらのメトリクスを監視して、影響を判断できます。詳細については、「Amazon CloudWatch による AWS FIS 使用状況メトリクスのモニタリング」を参照してください。
- システムの許容可能なパフォーマンスしきい値を定義する — システムの許容可能な定常状態を表すメトリクスを特定します。このメトリクスを使用して、実験の停止条件を表す 1 つ以上の CloudWatch アラームを作成します。アラームがトリガーされると、実験は自動的に停止します。詳細については、「AWS FIS の停止条件」を参照してください。

AWS Fault Injection Service のチュートリアル

以下のチュートリアルでは、 AWS Fault Injection Service (AWS FIS) を使用して実験を作成および実行する方法を示します。

チュートリアル

- チュートリアル : AWS FIS によるインスタンスの停止と開始をテスト
- チュートリアル: AWS FIS を使用して、インスタンス上で CPU ストレスを実行する
- チュートリアル: AWS FIS でスポットインスタンスの中断をテストする
- チュートリアル: 接続イベントをシミュレートする
- チュートリアル: 定期的な実験をスケジュールする

チュートリアル : AWS FIS によるインスタンスの停止と開始をテスト

アプリケーションがインスタンスの停止と開始を処理する方法を、AWS Fault Injection Service (AWS FIS) でテストします。このチュートリアルを使用して、 1 つのインスタンスを停止し、次に 2 番目のインスタンスを停止する AWS FIS `aws:ec2:stop-instances` アクションを使用する実験テンプレートを作成します。

前提条件

このチュートリアルを完了するには、以下を確実におこなってください。

- アカウントで 2 つのテスト EC2 インスタンスを起動します。インスタンスを起動したら、両方のインスタンスの ID を記録します。
- AWS FIS サービスが、ユーザーに代わって `aws:ec2:stop-instances` アクションを実行するための IAM ロールを作成します。詳細については、「AWS FIS 実験用の IAM ロール」を参照してください。
- AWS FIS にアクセスできることを確認します。詳細については、「AWS FIS ポリシー例」を参照してください。

ステップ 1: 実験テンプレートを作成する

AWS FIS コンソールで実験テンプレートを作成します。テンプレートで、それぞれ 3 分間連続して実行する 2 つのアクションを指定します。最初のアクションで、AWS FIS がランダムに選択したテストインスタンスの 1 つが停止します。2 番目のアクションでは、両方のテストインスタンスが停止します。

実験テンプレートを作成するには

1. https://console.aws.amazon.com/fis/ で AWS FIS コンソールを開きます。
2. ナビゲーションペインで、[実験テンプレート] を選択します。
3. [実験テンプレートの作成] を選択します。
4. [説明と名前] に、テンプレートの説明と名前を入力します。
5. [操作] で、以下の作業を行います。
   a. [アクションの追加] を選択します。
   b. アクションに名前を入力します。例えば、**stopOneInstance** と入力します。
   c. [アクションの種類] で、aws:ec2:stop-instances を選択します。
   d. [ターゲット] には、AWS FIS で作成されたターゲットをそのまま使用してください。
   e. [アクションパラメータ] [時間後にインスタンスを開始] に 3 分 (PT3M) を指定します。
   f. [保存] を選択します。
6. [ターゲット] で、以下の作業を行います。
   a. AWS FIS が前のステップで自動的に作成したターゲットの [編集] を選択します。
   b. デフォルト名を、よりわかりやすい名前に置き換えます。例えば、**oneRandomInstance**と入力します。
   c. [リソースタイプ] がaws:ec2:instance になっていることを確認します。
   d. [ターゲットメソッド] で、リソース ID をクリックし、2 つのテストインスタンスの ID を選択します。
   e. 選択モードで、[カウント] を選択します。[リソース数] に、「**1**」と入力します。
   f. [保存] を選択します。
7. [ターゲットの追加] を選択して、以下を実行します。
   a. ターゲットの名前を入力します。例えば、 **bothInstances**と入力します。
   b. [リソースタイプ] で、[aws:ec2:instance] を選択します。

c. [ターゲットメソッド] で、リソース ID をクリックし、2 つのテストインスタンスの ID を選択します。

d. [選択モード] で、すべてを選択します。

e. [保存] を選択します。

8. [アクション] セクションで、[アクションの追加] を選択します。以下の操作を実行します。

   a. [Name (名前)] に、アクションの名前を入力します。例えば、**stopBothInstances** と入力します。

   b. [アクションの種類] で、aws:ec2:stop-instances を選択します。

   c. 開始後で、最初に追加したアクションを選択します (**stopOneInstance**)。

   d. ターゲットで、追加した 2 番目のターゲットを選択します (**bothInstances**)。

   e. [アクションパラメータ]、[指定時間後にインスタンスを開始] で 3 分 (PT3M) を指定します。

   f. [保存] を選択します。

9. [サービスアクセス] では、[既存の IAM ロールを使用する] を選択し、このチュートリアルの前提条件の説明に従って作成した IAM ロールを選択します。ロールが表示されない場合は、必要な信頼関係があることを確認してください。詳細については、「the section called “実験ロール”」を参照してください。

10. (オプション)[タグ] では、[タグの追加] を選択して、タグのキーと値を指定します。追加するタグは、テンプレートを使用して実行される実験ではなく、実験テンプレートに適用されます。

11. [実験テンプレートの作成] を選択します。確認を求められたら、「**create**」と入力して、[実験テンプレートを作成する] を選択します。

(オプション) 実験テンプレート JSON を表示するには

[エクスポート] タブを選択します。次に、前述のコンソールプロシージャで作成された JSON の例を示します。

```
{
    "description": "Test instance stop and start",
    "targets": {
        "bothInstances": {
            "resourceType": "aws:ec2:instance",
            "resourceArns": [
                "arn:aws:ec2:region:123456789012:instance/instance_id_1",
                "arn:aws:ec2:region:123456789012:instance/instance_id_2"
```

```
            ],
            "selectionMode": "ALL"
        },
        "oneRandomInstance": {
            "resourceType": "aws:ec2:instance",
            "resourceArns": [
                "arn:aws:ec2:region:123456789012:instance/instance_id_1",
                "arn:aws:ec2:region:123456789012:instance/instance_id_2"
            ],
            "selectionMode": "COUNT(1)"
        }
    },
    "actions": {
        "stopBothInstances": {
            "actionId": "aws:ec2:stop-instances",
            "parameters": {
                "startInstancesAfterDuration": "PT3M"
            },
            "targets": {
                "Instances": "bothInstances"
            },
            "startAfter": [
                "stopOneInstance"
            ]
        },
        "stopOneInstance": {
            "actionId": "aws:ec2:stop-instances",
            "parameters": {
                "startInstancesAfterDuration": "PT3M"
            },
            "targets": {
                "Instances": "oneRandomInstance"
            }
        }
    },
    "stopConditions": [
        {
            "source": "none"
        }
    ],
    "roleArn": "arn:aws:iam::123456789012:role/AllowFISEC2Actions",
    "tags": {}
}
```

ステップ 2: 実験を開始する

実験テンプレートの作成が完了したら、それを使用して実験を開始できます。

実験を開始するには

1. 作成した実験テンプレートの詳細ページに移動する必要があります。または、[実験テンプレート] の ID を選択して、詳細ページを開きます。
2. [実験を開始する] を選択します。
3. (オプション) 実験にタグを追加するには、[新しいタグを追加] を選択し、タグキーとタグ値を入力します。
4. [実験を開始する] を選択します。確認を求められたら、**start** を入力して、[実験を開始する] を選択します。

ステップ 3: 実験の進行状況を追跡する

実験の完了、停止、または失敗するまで、実行中の実験の進行状況を追跡できます。

実験の進行状況を追跡するには

1. 始めたばかりの実験の詳細ページにいるはずです。または、[実験] の ID を選択して、詳細ページを開きます。
2. 実験の状態を表示するには、[詳細] ペインの [状態] を確認してください。詳細については、「実験状態」を参照してください。
3. 実験の状態が [実行中] のときは、次のステップに進みます。

ステップ 4: 実験結果の確認

インスタンスが実験によって停止され、期待どおりに開始されたことを確認できます。

実験の結果を検証するには

1. 新しいブラウザタブまたはウィンドウで、Amazon EC2 コンソール https://console.aws.amazon.com/ec2/ を開きます。これにより、AWS FIS コンソールで実験の進行状況を追跡し続けながら、Amazon EC2 コンソールで実験の結果を表示することができます。
2. ナビゲーションペインで、[Instances] (インスタンス) を選択します。

3. 最初のアクションの状態が保留中から実行中(AWS FIS コンソール) に変わるとき、いずれかのターゲットインスタンスの状態が実行中から停止(Amazon EC2 コンソール) に変わります。
4. 3 分後、最初のアクションの状態が完了とすると、第 2 アクションの状態は実行中となり、もう一方のターゲットインスタンスの状態は停止に変わります。
5. 3 分後、2 回目のアクションの状態が完了の場合、ターゲットインスタンスの状態は実行中となり、実験の状態は完了に変わります。

ステップ 5：クリーンアップ

実験で作成したテスト EC2 インスタンスが不要になった場合は、終了できます。

インスタンスを終了するには

1. Amazon EC2 コンソール (https://console.aws.amazon.com/ec2/) を開きます。
2. ナビゲーションペインで、[Instances] (インスタンス) を選択します。
3. 両方のテストインスタンスを選択し、[インスタンスの状態]、[インスタンスの終了] の順に選択します。
4. 確認を求めるメッセージが表示されたら、[終了] を選択します。

実験テンプレートが不要になった場合は、削除できます。

AWS FIS コンソールを使用して実験テンプレートを削除するには

1. https://console.aws.amazon.com/fis/ で AWS FIS コンソールを開きます。
2. ナビゲーションペインで、[実験テンプレート] を選択します。
3. 実験テンプレートを選択し、[アクション]、[実験テンプレートの削除] を選択します。
4. 確認を求められたら、**`delete`** と入力し、[実験テンプレートの削除] を選択します。

チュートリアル: AWS FIS を使用して、インスタンス上で CPU ストレスを実行する

アプリケーションが CPU ストレスをどのように処理するかを、AWS (AWS FIS) Fault Injection Service でテストします。このチュートリアルでは、AWS FIS を使用し、インスタンス上で CPU ストレスを実行する、事前設定済みの SSM ドキュメントを実行する実験テンプレートを作成します。

このチュートリアルでは、インスタンスの CPU 使用率が事前設定済みのしきい値を超えた場合に、停止条件を使用して実験を停止します。

詳細については、「the section called “事前設定された AWS FIS SSM ドキュメント”」を参照してください。

前提条件

AWS FIS を使用して CPU ストレスを実行する前に、以下の前提条件を満たしてください。

IAM ロールの作成

ロールを作成し、ユーザーに代わって AWS FIS が `aws:ssm:send-command` アクションを実行するためのポリシーをアタッチします。詳細については、「AWS FIS 実験用の IAM ロール」を参照してください。

AWS FIS へのアクセスを検証する

AWS FIS にアクセスできることを確認します。詳細については、「AWS FIS ポリシー例」を参照してください。

テスト EC2 インスタンスを準備する

- 事前設定済みの SSM ドキュメントの要求に応じて、Amazon Linux 2 または Ubuntu を使用して EC2 インスタンスを起動します。
- インスタンスは SSM によって管理されている必要があります。インスタンスが SSM によって管理されていることを確認するには、Fleet Manager コンソールを開きます。インスタンスが SSM によって管理されていない場合は、SSM エージェントがインストールされていること、およびインスタンスに AmazonSSMManagedInstanceCore ポリシーを持つ IAM ロールがアタッチされていることを確認します。インストールされている SSM Agentを確認するには、インスタンスに接続し、以下のコマンドを実行します。

  Amazon Linux 2

  ```
  yum info amazon-ssm-agent
  ```

  Ubuntu

  ```
  apt list amazon-ssm-agent
  ```

- インスタンスの詳細モニタリングを有効にします。追加料金で、1 分間のデータを取得できます。インスタンスを選択し、[Actions] (アクション)、[Monitor and troubleshoot] (モニタリングとトラブルシューティング)、[Manage detailed monitoring] (詳細モニタリングの管理) の順に選択します。

ステップ 1: 停止条件の CloudWatch アラームを作成する

CPU 使用率が指定したしきい値を超えた場合に実験を停止できるように CloudWatch アラームを設定します。次の手順では、ターゲットインスタンスの CPU 使用率を 50% に設定します。詳細については、「停止条件」を参照してください。

CPU 使用率がしきい値を超えたことを示すアラームを作成するには

1. Amazon EC2 コンソール (https://console.aws.amazon.com/ec2/) を開きます。
2. ナビゲーションペインで、[Instances] (インスタンス) を選択します。
3. ターゲットインスタンスを選択し、アクション、監視とトラブルシューティング、 CloudWatch アラームの管理を選択します。
4. [Alarm notification] (アラーム通知) で、トグルを使用して Amazon SNS 通知をオフにします。
5. アラームのしきい値を使用する場合は、以下の設定を使用します。
   - サンプルのグループ化: 最大
   - サンプリングするデータのタイプ: CPU 使用率
   - 割合 (%): **50**
   - 間隔: **1 Minute**
6. アラームの設定が終わったら、[Create] を選択します。

ステップ 2: 実験テンプレートを作成する

AWS FIS コンソールで実験テンプレートを作成する。テンプレートでは、実行するアクション aws:ssm:send-command/AWSFIS-Run-CPU-Stress を指定します。

実験テンプレートを作成するには

1. https://console.aws.amazon.com/fis/ で AWS FIS コンソールを開きます。
2. ナビゲーションペインで、[実験テンプレート] を選択します。
3. [実験テンプレートの作成] を選択します。
4. [説明と名前] に、テンプレートの説明と名前を入力します。

5. [操作] で、以下の作業を行います。

   a. [アクションの追加] を選択します。
   b. アクションに名前を入力します。例えば、**runCpuStress** と入力します。
   c. アクションタイプ で、aws:ssm:send-command/AWSFIS-Run-CPU-Stress を選択します。これにより、SSM ドキュメントの ARN がドキュメント ARN に自動的に追加されます。
   d. ターゲット については、AWS FIS が作成したターゲットをそのまま使用してください。
   e. [アクションパラメーター]、[ドキュメントパラメータ] には、次のように入力します。

   ```
   {"DurationSeconds":"120"}
   ```

   f. [アクションパラメータ]、[継続時間]で、5 分 (PT5M) を指定します。
   g. [保存] を選択します。
6. [ターゲット] で、以下の作業を行います。

   a. AWS FIS が前のステップで自動的に作成したターゲットの [編集] を選択します。
   b. デフォルト名を、よりわかりやすい名前に置き換えます。例えば、 **testInstance**と入力します。
   c. [リソースタイプ] でaws:ec2:instanceになっていることを確認します。
   d. [ターゲットメソッド] で、リソース ID を選択し、テストインスタンスの ID を選択します。
   e. 選択モードで、すべてを選択します。
   f. [保存] を選択します。
7. [サービスアクセス] では、[既存の IAM ロールを使用する] を選択し、このチュートリアルの前提条件の説明に従って作成した IAM ロールを選択します。ロールが表示されない場合は、必要な信頼関係があることを確認してください。詳細については、「the section called “実験ロール”」を参照してください。
8. 停止条件 で、ステップ 1 で作成した CloudWatch アラームを選択します。
9. (オプション) タグで、[タグの追加] を選択して、タグのキーと値を指定します。追加するタグは、テンプレートを使用して実行される実験ではなく、実験テンプレートに適用されます。
10. [実験テンプレートの作成] を選択します。

(オプション) 実験テンプレート JSON を表示するには

[エクスポート] タブを選択します。次に、前述のコンソールプロシージャで作成された JSON の例を示します。

```
{
    "description": "Test CPU stress predefined SSM document",
    "targets": {
        "testInstance": {
            "resourceType": "aws:ec2:instance",
            "resourceArns": [
                "arn:aws:ec2:region:123456789012:instance/instance_id"
            ],
            "selectionMode": "ALL"
        }
    },
    "actions": {
        "runCpuStress": {
            "actionId": "aws:ssm:send-command",
            "parameters": {
                "documentArn": "arn:aws:ssm:region::document/AWSFIS-Run-CPU-Stress",
                "documentParameters": "{\"DurationSeconds\":\"120\"}",
                "duration": "PT5M"
            },
            "targets": {
                "Instances": "testInstance"
            }
        }
    },
    "stopConditions": [
        {
            "source": "aws:cloudwatch:alarm",
            "value": "arn:aws:cloudwatch:region:123456789012:alarm:awsec2-instance_id-
GreaterThanOrEqualToThreshold-CPUUtilization"
        }
    ],
    "roleArn": "arn:aws:iam::123456789012:role/AllowFISSSMActions",
    "tags": {}
}
```

ステップ 3: 実験を開始する

実験テンプレートの作成が完了したら、それを使用して実験を開始できます。

実験を開始するには

1. 作成した実験テンプレートの詳細ページに移動する必要があります。または、[実験テンプレート] の ID を選択して、詳細ページを開きます。
2. [実験を開始する] を選択します。
3. (オプション) 実験にタグを追加するには、[新しいタグを追加] を選択し、タグキーとタグ値を入力します。
4. [実験を開始する] を選択します。確認を求められたら、**start** をクリックします。[実験を開始する] を選択します。

ステップ 4: 実験の進行状況を追跡する

実験が完了、停止、または失敗するまで、実行中の実験の進行状況を追跡できます。

実験の進行状況を追跡するには

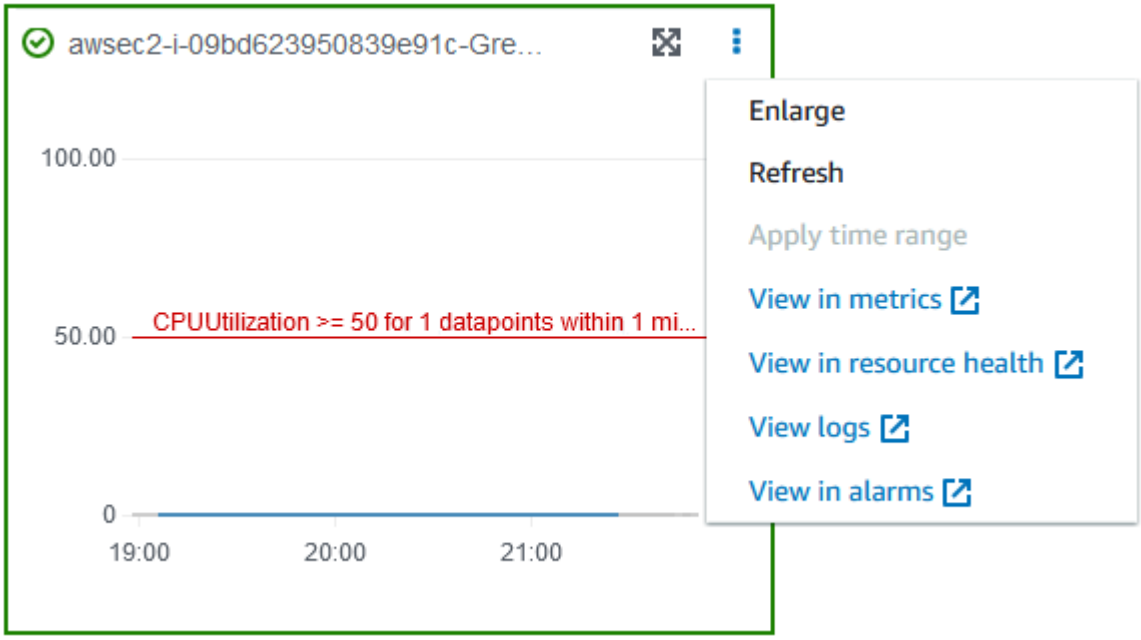
1. 始めたばかりの実験の詳細ページにいるはずです。または、[Experiments] (実験) の ID を選択して、詳細ページを開きます。
2. 実験の状態を表示するには、状態のの詳細ペインを確認してください。詳細については、「実験状態」を参照してください。
3. 実験状態が実行中のときは、次のステップに進みます。

ステップ 5: 実験結果の検証

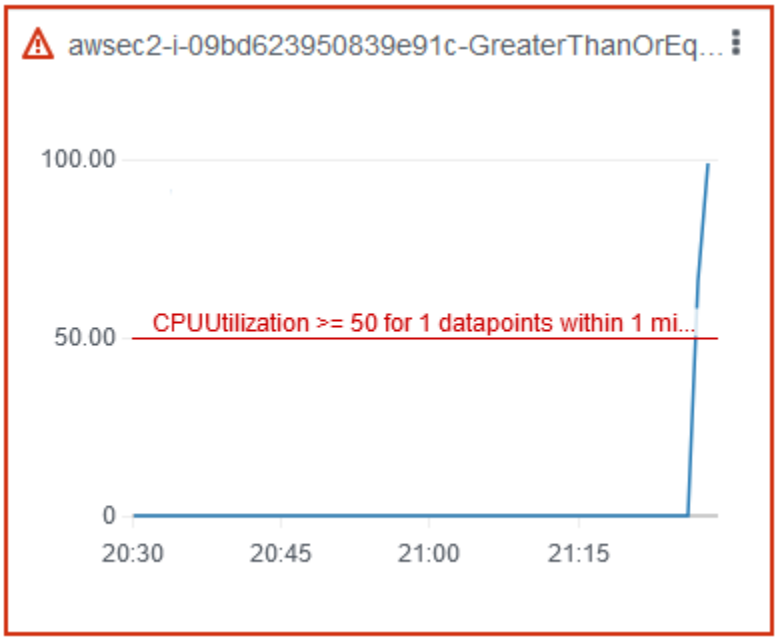
実験の実行中に、インスタンスの CPU 使用率を監視できます。CPU 使用率がしきい値に達すると、アラームがトリガーされ、停止条件によって実験が停止されます。

実験の結果を検証するには

1. [停止条件] タブを選択します。緑の境界線と緑のチェックマークアイコンは、アラームの初期状態が OK であることを示しています。赤い線は、アラームのしきい値を示します。より詳細なグラフが必要な場合は、ウイジェットメニューから[拡大する] を選択します。



2. CPU 使用率がしきい値を超えると、停止条件タブの赤い境界線と赤い感嘆符のアイコンが、アラームの状態が ALARM に変わったことを示します。詳細ペインでは、実験の状態は停止です。状態を選択すると、表示されるメッセージは「実験を停止条件で停止しました」です。



3. CPU 使用率がしきい値を下回ると、緑の境界線と緑のチェックマークアイコンは、アラームの状態が OK に変わったことを示します。

4. (オプション) ウィジェットメニューから [アラームの表示] を選択します。これにより、CloudWatch コンソールでアラームの詳細ページが開き、アラームの詳細を確認したり、アラーム設定を編集したりできます。

ステップ 6: クリーンアップする

この実験で作成したテスト EC2 インスタンスが不要になった場合は、終了できます。

インスタンスを終了するには

1. Amazon EC2 コンソール (https://console.aws.amazon.com/ec2/) を開きます。
2. ナビゲーションペインで、[Instances] (インスタンス) を選択します。
3. テストインスタンスを選択し、[インスタンスの状態]、[インスタンスの終了] の順に選択します。
4. 確認を求めるメッセージが表示されたら、[終了] を選択します。

実験テンプレートが不要になった場合は、削除できます。

AWS FIS コンソールを使用して実験テンプレートを削除するには

1. https://console.aws.amazon.com/fis/ で AWS FIS コンソールを開きます。
2. ナビゲーションペインで、[実験テンプレート] を選択します。
3. 実験テンプレートを選択し、[アクション]、[実験テンプレートの削除] を選択します。
4. 確認を求められたら、**delete** と入力し、[実験テンプレートの削除] を選択します。

チュートリアル: AWS FIS でスポットインスタンスの中断をテストする

スポットインスタンスは、利用可能な予備の EC2 キャパシティーを使用します。オンデマンド料金と比較して最大 90% の割引が適用されます。しかし、Amazon EC2 は、必要なときにスポットインスタンスを中断できます。スポットインスタンスを使用する場合には、中断に備えておく必要があります。詳細については、[Amazon EC2 ユーザーガイド] の[スポットインスタンス中断] を参照してください。

スポットインスタンスの中断をアプリケーションが処理する方法を AWS Fault Injection Service (AWS FIS) でテストできます。このチュートリアルでは、 スポットインスタンスの 1 つを AWS FIS

aws:ec2:send-spot-instance-interruptions アクションで中断する実験テンプレートを作成します。

また、Amazon EC2 コンソールで実験を開始するには、『Amazon EC2 ユーザーガイド』の「スポットインスタンスの中断を開始する」を参照してください。

前提条件

AWS FIS でスポットインスタンスを中断する前に、以下の前提条件が満たされていることを確認してください。

1. IAM ロールの作成

ロールを作成し、AWS FIS にユーザーに代わって aws:ec2:send-spot-instance-interruptions アクションを実行できるポリシーをアタッチします。詳細については、「AWS FIS 実験用の IAM ロール」を参照してください。

2. AWS FIS へのアクセスを検証する

AWS FIS にアクセスできることを確認します。詳細については、「AWS FIS ポリシー例」を参照してください。

3. (オプション) スポットインスタンスリクエストを作成する

この実験に新しいスポットインスタンスを使用する場合は、run-instances コマンドでスポットインスタンスをリクエストします。デフォルトでは、スポットインスタンスは中断されると終了します。stop に割り込み動作を設定した場合は、タイプを persistent に設定する必要もあります。休止状態プロセスがすぐに始まるので、このチュートリアルでは、中断動作を hibernate に設定しないでください。

```
aws ec2 run-instances \
    --image-id ami-0ab193018fEXAMPLE \
    --instance-type "t2.micro" \
    --count 1 \
    --subnet-id subnet-1234567890abcdef0 \
    --security-group-ids sg-111222333444aaab \
    --instance-market-options file://spot-options.json \
    --query Instances[*].InstanceId
```

次は、spot-options.json ファイルの例です。

```
{
```

```
    "MarketType": "spot",
    "SpotOptions": {
        "SpotInstanceType": "persistent",
        "InstanceInterruptionBehavior": "stop"
    }
}
```

サンプルコマンドの `--query` オプションを使用すると、コマンドはスポットインスタンスのインスタンス ID のみが返ります。以下は出力例です。

```
[
    "i-0abcdef1234567890"
]
```

4. AWS FIS がターゲットのスポットインスタンスを識別できるようにタグを追加します。

タグ Name=interruptMe をターゲットのスポットインスタンスに追加するには、create-tags コマンドを使用します。

```
aws ec2 create-tags \
    --resources i-0abcdef1234567890 \
    --tags Key=Name,Value=interruptMe
```

ステップ 1: 実験テンプレートを作成する

AWS FIS コンソールで実験テンプレートを作成する。テンプレートで、実行するアクションを指定します。アクションは、指定されたタグでスポットインスタンスを中断します。タグが付いたスポットインスタンスが複数ある場合、AWS FIS によってそのうちの 1 つがランダムに選択されます。

実験テンプレートを作成するには

1. https://console.aws.amazon.com/fis/ で AWS FIS コンソールを開きます。
2. ナビゲーションペインで、[実験テンプレート] を選択します。
3. [実験テンプレートの作成] を選択します。
4. [説明と名前] に、テンプレートの説明と名前を入力します。
5. [操作] で、以下の作業を行います。
    a. [アクションの追加] を選択します。
    b. アクションに名前を入力します。例えば、**`interruptSpotInstance`** と入力します。

c. アクションタイプ で、aws:ec2:send-spot-instance-interruptions を選択します。

d. ターゲット については、AWS FIS が作成したターゲットをそのまま使用してください。

e. [アクションパラメーター] の [中断までの時間] に 2 分 (PT2M) を指定します。

f. [保存] を選択します。

6. [ターゲット] で、以下の作業を行います。

   a. AWS FIS が前のステップで自動的に作成したターゲットの [編集] を選択します。

   b. デフォルト名を、よりわかりやすい名前に置き換えます。例えば、**oneSpotInstance** と入力します。

   c. [リソースタイプ] が aws:ec2:spot-instance になっていることを確認します。

   d. [ターゲットメソッド] に、リソースタグ、フィルター、パラメータを選択します。

   e. [リソースタグ] で、新しいタグを追加を選択し、タグのキーとタグの値を入力します。このチュートリアルの前提条件の説明に従って、スポットインスタンスに追加したタグを使用して割り込みます。

   f. [リソースフィルター] で [新しいフィルターを追加] を選択し、パス **State.Name** と値 **running** を入力します。

   g. [選択モード] で、[カウント] を選択します。[リソース数] に、「**1**」と入力します。

   h. [保存] を選択します。

7. [サービスアクセス] では、[既存の IAM ロールを使用する] を選択し、このチュートリアルの前提条件の説明に従って作成した IAM ロールを選択します。ロールが表示されない場合は、必要な信頼関係があることを確認してください。詳細については、「the section called “実験ロール”」を参照してください。

8. (オプション)[タグ] では、[タグの追加] を選択して、タグのキーと値を指定します。追加するタグは、テンプレートを使用して実行される実験ではなく、実験テンプレートに適用されます。

9. [実験テンプレートの作成] を選択します。確認を求められたら、「**create**」と入力して、[実験テンプレートを作成する] を選択します。

(オプション) 実験テンプレート JSON を表示するには

[エクスポート] タブを選択します。次に、前述のコンソールプロシージャで作成された JSON の例を示します。

```
{
    "description": "Test Spot Instance interruptions",
```

```
    "targets": {
        "oneSpotInstance": {
            "resourceType": "aws:ec2:spot-instance",
            "resourceTags": {
                "Name": "interruptMe"
            },
            "filters": [
                {
                    "path": "State.Name",
                    "values": [
                        "running"
                    ]
                }
            ],
            "selectionMode": "COUNT(1)"
        }
    },
    "actions": {
        "interruptSpotInstance": {
            "actionId": "aws:ec2:send-spot-instance-interruptions",
            "parameters": {
                "durationBeforeInterruption": "PT2M"
            },
            "targets": {
                "SpotInstances": "oneSpotInstance"
            }
        }
    },
    "stopConditions": [
        {
            "source": "none"
        }
    ],
    "roleArn": "arn:aws:iam::123456789012:role/AllowFISSpotInterruptionActions",
    "tags": {
        "Name": "my-template"
    }
}
```

ステップ 2: 実験を開始する

実験テンプレートの作成が完了したら、それを使用して実験を開始できます。

実験を開始するには

1. 作成した実験テンプレートの詳細ページに移動する必要があります。または、[実験テンプレート] の ID を選択して、詳細ページを開きます。
2. [実験を開始する] を選択します。
3. (オプション) 実験にタグを追加するには、[新しいタグを追加] を選択し、タグキーとタグ値を入力します。
4. [実験を開始する] を選択します。確認を求められたら、**start** を入力して、[実験を開始する] を選択します。

ステップ 3: 実験の進行状況を追跡する

実験の完了、停止、または失敗するまで、実行中の実験の進行状況を追跡できます。

実験の進行状況を追跡するには

1. 始めたばかりの実験の詳細ページにいるはずです。または、[実験] の ID を選択して、詳細ページを開きます。
2. 実験の状態を表示するには、[詳細] ペインの [状態] を確認してください。詳細については、「実験状態」を参照してください。
3. 実験の状態が [実行中] のときは、次のステップに進みます。

ステップ 4: 実験結果の検証

この実験のアクションが完了すると、以下の状況が発生します。

- ターゲットのスポットインスタンスはインスタンスの再調整に関する推奨事項を受け取ります。
- スポットインスタンスの中断通知は、Amazon EC2 がインスタンスを停止または終了する 2 分前に発行されます。
- 2 分後、スポットインスタンスは終了または停止します。
- AWS FIS で停止されたスポットインスタンスは、ユーザーにより再起動されるまで停止状態を維持します。

インスタンスが実験によって中断されたことを検証するには

1. Amazon EC2 コンソール (https://console.aws.amazon.com/ec2/) を開きます。

2. ナビゲーションペインで、[スポットリクエスト] を開いてから、別のブラウザタブまたはウィンドウで[インスタンス] を開きます。
3. スポットリクエストで、スポットインスタンスリクエストを選択します。初期ステータスは、`fulfilled` です。実験が完了すると、ステータスは次のように変わります。
   - `terminate` - ステータスが `instance-terminated-by-experiment` に変わります。
   - `stop` - ステータスが `marked-for-stop-by-experiment` そしてその後 `instance-stopped-by-experiment` に変わります。
4. インスタンスで、スポットインスタンスを選択します。初期ステータスは、`Running` です。スポットインスタンスの中断通知を受け取った後 2 分後、ステータスは次のように変化します。
   - `stop` - ステータスが `Stopping` そしてその後 `Stopped` に変わります。
   - `terminate` - ステータスが `Shutting-down` そしてその後 `Terminated` に変わります。

ステップ 5：クリーンアップ

この実験のテストスポットインスタンスを中断動作 `stop` で作成した場合で、そのスポットインスタンスが必要でなくなったら、スポットインスタンスリクエストをキャンセルし、スポットインスタンスを終了することができます。

リクエストをキャンセルし、AWS CLI を使用してインスタンスを終了するには

1. cancel-spot-instance-requests コマンドを使用して、スポットインスタンスリクエストをキャンセルします。

```
aws ec2 cancel-spot-instance-requests --spot-instance-request-ids sir-ksie869j
```

2. インスタンスを終了するには、terminate-instances コマンドを使用します。

```
aws ec2 terminate-instances --instance-ids i-0abcdef1234567890
```

実験テンプレートが不要になった場合には、それを削除することができます。

AWS FIS コンソールを使用して実験テンプレートを削除するには

1. https://console.aws.amazon.com/fis/ で AWS FIS コンソールを開きます。
2. ナビゲーションペインで、[実験テンプレート] を選択します。
3. 実験テンプレートを選択し、[アクション]、[実験テンプレートの削除] を選択します。

4. 確認を求められたら、**delete** と入力し、[実験テンプレートの削除] を選択します。

チュートリアル: 接続イベントをシミュレートする

AWS Fault Injection Service (AWS FIS) を使用して、さまざまな接続イベントをシミュレートできます。 AWS FIS は、次のいずれかの方法でネットワーク接続をブロックすることで接続イベントをシミュレートします。

- `all` - サブネットに出入りするすべてのトラフィックを拒否します。このオプションでは、サブネット内のネットワークインターフェースに出入りするトラフィックを含め、サブネット内トラフィックが許可されることに注意してください。
- `availability-zone` - 他のアベイラビリティーゾーン内のサブネットとの間の VPC 内トラフィックを拒否します。
- dynamodb - 現在のリージョンの DynamoDB のリージョナルエンドポイントとの間のトラフィックを拒否します。
- `prefix-list` - 指定されたプレフィックスリストとの間で送受信されるトラフィックを拒否します。
- `s3` - 現在のリージョンの Amazon S3 のリージョンエンドポイントとの間のトラフィックを拒否します。
- `vpc` - VPC に出入りするトラフィックを拒否します。

このチュートリアルを使用して、 AWS FIS `aws:network:disrupt-connectivity`アクションを使用してターゲットサブネットで Amazon S3 との接続が失われる実験テンプレートを作成します。

トピック

前提条件

このチュートリアルを開始する前に、 で適切なアクセス許可を持つロールと AWS アカウント、テスト用の Amazon EC2 インスタンスが必要です。

のアクセス許可を持つロール AWS アカウント

ロールを作成し、ユーザーに代わって AWS FIS が`aws:network:disrupt-connectivity`アクションを実行できるようにするポリシーをアタッチします。

IAM ロールには次のポリシーが必要です。

- AWSFaultInjectionSimulatorNetworkAccess – ネットワークインフラストラクチャに関連する AWS FIS アクションを実行するために必要な Amazon EC2 ネットワークおよびその他の のサービスで AWS FIS サービスアクセス許可を付与します。

> **Note**
>
> わかりやすくするために、このチュートリアルでは AWS マネージドポリシーを使用します。実稼働環境では、代わりに、ユースケースに必要な最小限の許可のみを与えることをお勧めします。
> IAM ロールの作成方法の詳細については、「IAM ユーザーガイド」の AWS 「FIS 実験用の IAM ロール (AWS CLI)」または「IAM ロールの作成 (コンソール)」を参照してください。

テスト Amazon EC2 インスタンス

テスト Amazon EC2 インスタンスを起動し、接続します。次のチュートリアルを使用して、Amazon EC2 インスタンスを起動して接続できます。チュートリアル: Amazon EC2 ユーザーガイドの「Amazon EC2 Linux インスタンスの使用を開始する」。 Amazon EC2

ステップ 1: AWS FIS 実験テンプレートを作成する

AWS FIS を使用して実験テンプレートを作成します AWS Management Console。 AWS FIS テンプレートは、アクション、ターゲット、停止条件、実験ロールで構成されます。テンプレートの仕組みの詳細については、「AWS FIS 用実験テンプレート」を参照してください。

作業を開始する前に、次の項目が揃っていることを確認してください。

- 正確な権限を持つ IAM ロール。
- Amazon EC2 インスタンス。
- Amazon EC2 インスタンスのサブネット ID。

実験テンプレートを作成するには

1. https://console.aws.amazon.com/fis/ で AWS FIS コンソールを開きます。
2. ナビゲーションペインで、[実験テンプレート] を選択します。
3. [Create experiment template (実験テンプレートの作成)] を選択します。
4. `Amazon S3 Network Disrupt Connectivity` などテンプレートの説明を入力します。
5. [アクション] で、[アクションの追加] を選択します。
   a. [名前] には、「`disruptConnectivity`」と入力します。
   b. [アクションタイプ] には、aws:network:disrupt-connectivity を選択します。
   c. [アクションパラメータ] で、[期間] を `2 minutes` に設定します。
   d. [スコープ] で 「s3」 を選択します。
   e. 上部の [保存] を選択します。
6. [ターゲット] に、自動的に作成されたターゲットが表示されます。[編集] を選択します。
   a. [リソースタイプ] が `aws:ec2:subnet` であることを確認してください。
   b. [ターゲットメソッド] で [リソース ID] を選択し、前提条件のステップで Amazon EC2 インスタンスを作成するときに使用したサブネットを選択します。
   c. [選択モード] が 「すべて」 であることを確認してください。
   d. [Save (保存)] を選択します。
7. [サービスアクセス] で、このチュートリアルの 「前提条件」の説明に従って作成した IAM ロールを選択します。ロールが表示されない場合は、必要な信頼関係があることを確認してください。詳細については、「the section called “実験ロール”」を参照してください。
8. (オプション) 停止条件 で CloudWatch、条件が発生した場合に実験を停止するアラームを選択できます。詳細については、 「AWS FIS の停止条件」を参照してください。
9. (オプション) ログ で、Amazon S3 バケットを選択するか、実験 CloudWatch のために にログを送信できます。
10. [実験テンプレートを作成する] を選択します。確認を求められたら、「`create`」と入力してください。[実験テンプレートの]作成 を選択します。

ステップ 2: Amazon S3 エンドポイントに ping を実行する

Amazon EC2 インスタンスが Amazon S3 エンドポイントに到達できることを確認してください。

1. 「前提条件」のステップで作成した Amazon EC2 インスタンスに接続します。

   トラブルシューティングについては、「Amazon EC2 ユーザーガイド」の「インスタンスへの接続のトラブルシューティング」を参照してください。 Amazon EC2

2. インスタンス AWS リージョン が配置されている を確認します。これを行うには、Amazon EC2 コンソールに接続し、次のコマンドを実行します。

   ```
   hostname
   ```

   例えば、`us-west-2` で Amazon EC2 インスタンスを起動した場合、次の出力が表示されます。

   ```
   [ec2-user@ip-172.16.0.0 ~]$ hostname
   ip-172.16.0.0.us-west-2.compute.internal
   ```

3. で Amazon S3 エンドポイントに Ping を実行します AWS リージョン。*AWS #####* は、実際のリージョンに置き換えます。

   ```
   ping -c 1 s3.AWS #####.amazonaws.com
   ```

   出力では、次の例に示すように、パケット損失が 0% で ping が成功したことがわかります。

   ```
   PING s3.us-west-2.amazonaws.com (x.x.x.x) 56(84) bytes of data.
   64 bytes from s3-us-west-2.amazonaws.com (x.x.x.x: icmp_seq=1 ttl=249 time=1.30 ms

   --- s3.us-west-2.amazonaws.com ping statistics ---
   1 packets transmitted, 1 received, 0% packet loss, time 0ms
   rtt min/avg/max/mdev = 1.306/1.306/1.306/0.000 ms
   ```

ステップ 3: AWS FIS 実験を開始する

作成した実験テンプレートで、実験を開始します。

1. https://console.aws.amazon.com/fis/ で AWS FIS コンソールを開きます。

2. 左のナビゲーションペインで、[実験テンプレート] を選択します。
3. 作成した実験テンプレートの ID を選択して、その詳細ページを開きます。
4. [実験を開始する] を選択します。
5. (オプション) 確認ページで、実験のタグを追加します。
6. 確認ページで [実験を開始する] を選択します。

ステップ 4: AWS FIS 実験の進行状況を追跡する

実験の完了、停止、または失敗するまで、実行中の実験の進行状況を追跡できます。

1. 始めたばかりの実験の詳細ページが表示されているはずです。または、[実験] を選択して、実験の ID を選択して詳細ページを開きます。
2. 実験の状態を表示するには、[詳細] ペインの [状態] を確認してください。詳細については、「実験状態」を参照してください。
3. 実験の状態が[実行中] のときは、次のステップに進みます。

ステップ 5: Amazon S3 ネットワークの中断を確認する

実験の進行状況は、Amazon S3 エンドポイントに ping を送信することで検証できます。

- Amazon EC2 インスタンスから、 AWS リージョンの Amazon S3 エンドポイントに対して ping を実行します。*AWS #####* は、実際のリージョンに置き換えます。

  ```
  ping -c 1 s3.AWS #####.amazonaws.com
  ```

  出力では、次の例に示すように、パケット損失が 100% の ping に失敗した ping が表示されます。

  ```
  ping -c 1 s3.us-west-2.amazonaws.com
  PING s3.us-west-2.amazonaws.com (x.x.x.x) 56(84) bytes of data.

  --- s3.us-west-2.amazonaws.com ping statistics ---
  1 packets transmitted, 0 received, 100% packet loss, time 0ms
  ```

ステップ 5：クリーンアップ

実験で作成した Amazon EC2 インスタンスや AWS FIS テンプレートが不要になった場合は、削除できます。

Amazon EC2 インスタンスを削除するには

1. Amazon EC2 コンソール (https://console.aws.amazon.com/ec2/) を開きます。
2. ナビゲーションペインで、[インスタンス] を選択します。
3. インスタンスを選択し、[インスタンスの状態]、[インスタンスの終了] の順に選択します。
4. 確認を求めるメッセージが表示されたら、[終了] を選択します。

AWS FIS コンソールを使用して実験テンプレートを削除するには

1. https://console.aws.amazon.com/fis/ で AWS FIS コンソールを開きます。
2. ナビゲーションペインで、[Experiment templates (実験テンプレート)] を選択します。
3. 実験テンプレートを選択し、[アクション]、[実験テンプレートの削除] を選択します。
4. 確認を求められたら、`delete` と入力し、[実験テンプレートの削除] を選択してください。

チュートリアル: 定期的な実験をスケジュールする

AWS Fault Injection Service (AWS FIS) では、AWS ワークロードに対してフォールトインジェクション実験を行うことができます。これらの実験は、指定したターゲットに対して実行する 1 つ以上のアクションがあるテンプレート上で実行します。また、Amazon EventBridge を使用すると、実験を 1 回限りのタスクまたは定期的なタスクとしてスケジュールできます。

このチュートリアルを使用して、5 分ごとに AWS FIS 実験テンプレートを実行する EventBridge スケジュールを作成します。

タスク

- 前提条件
- ステップ 1: IAM ポリシーとロールを作成する
- ステップ 2: Amazon EventBridge スケジューラーを作成する
- ステップ 3: 実験を検証
- ステップ 4: クリーンアップする

前提条件

このチュートリアルを開始する前に、スケジュールに従って実行する AWS FIS 実験テンプレートを用意しておく必要があります。動作する実験テンプレートがすでにある場合は、テンプレート ID と AWS リージョン をメモしておいてください。それ以外の場合は、the section called “インスタンスの停止と開始をテスト” の手順に従ってテンプレートを作成してから、このチュートリアルに戻ってください。

ステップ 1: IAM ポリシーとロールを作成する

IAM ポリシーとロールを作成するには

1. IAM コンソール (https://console.aws.amazon.com/iam/) を開きます。
2. 左側のナビゲーションペインから、[ロール] を選択し、[ロールの作成] を選択します。
3. [カスタムトラストポリシー] を選択し、次のスニペットを挿入すると、Amazon EventBridge スケジューラ–がユーザーに代わってロールを引き受けます。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                  "Service": "scheduler.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

   [Next (次へ)] をクリックします。
4. [アクセス許可の追加] で [バケットポリシーの追加] を選択します。
5. [JSON] を選択し、以下のポリシーを挿入します。*your-experiment-template-id* 値を、前提条件ステップの実験のテンプレート ID に置き換えます。

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

```
        {
            "Effect": "Allow",
            "Action": "fis:StartExperiment",
            "Resource": [
                "arn:aws:fis:*:*:experiment-template/your-experiment-template-id",
                "arn:aws:fis:*:*:experiment/*"
            ]
        }
    ]
}
```

特定のタグ値を持つ AWS FIS 実験のみを実行するようにスケジューラーを制限できます。例えば、次のポリシーではすべての AWS FIS 実験テンプレートに対する `StartExperiment` アクセス許可が与えられますが、`Purpose=Schedule` タグ付きの実験のみがスケジューラーによって実行されるように制限されます。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "fis:StartExperiment",
            "Resource": "arn:aws:fis:*:*:experiment/*"
        },
        {
            "Effect": "Allow",
            "Action": "fis:StartExperiment",
            "Resource": "arn:aws:fis:*:*:experiment-template/*",
            "Condition": {
                "StringEquals": {
                    "aws:ResourceTag/Purpose": "Schedule"
                }
            }
        }
    ]
}
```

[次へ : タグ] を選択します。

6. [次へ: レビュー] を選択します。
7. [ポリシーの確認] で、ポリシー `FIS_RecurringExperiment` に名前を付け、[ポリシーの作成] を選択します。

8. [アクセス許可の追加] で、新しい `FIS_RecurringExperiment` ポリシーをロールに追加し、[次へ] を選択します。
9. [名前、確認、および作成] で、ロール `FIS_RecurringExperiment_role` に名前を付け、[ロールを作成] を選択します。

ステップ 2: Amazon EventBridge スケジューラーを作成する

Amazon EventBridge スケジューラーを作成するには

1. https://console.aws.amazon.com/events/ で Amazon EventBridge 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 [スケジュール]を選択します。
3. AWS FIS 実験テンプレートと同じ AWS リージョン にいることを確認します。
4. [スケジュールの作成] を選択し、以下を入力します。
   - [スケジュール名] に、`FIS_recurring_experiment_tutorial` を挿入します。
   - [スケジュールパターン] で、[定期的なスケジュール] を選択します。
   - [スケジュールタイプ] で [レートベースのスケジュール] を選択します。
   - [レート表現] で [5 分] を選択します。
   - [フレキシブルタイムウィンドウ] で [オフ] を選択します。
   - (オプション) [タイムフレーム] でタイムゾーンを選択します。
   - [次へ] をクリックします。
5. [ターゲットを選択] で [すべての API] を選択し、「AWSFIS」を検索します。
6. AWS FIS を選択し、 を選択しますStartExperiment。
7. [入力] に、次の JSON ペイロードを挿入します。*`your-experiment-template-id`* 値を実験のテンプレート ID に置き換えます。`ClientToken` はスケジューラーの一意の識別子です。このチュートリアルでは、Amazon EventBridge スケジューラ–で許可されているコンテキストキーワードを使用しています。詳細については、「Amazon ユーザーガイド」の「コンテキスト属性の追加」を参照してください。 EventBridge

```
{
    "ClientToken": "<aws.scheduler.execution-id>",
    "ExperimentTemplateId": "your-experiment-template-id"
}
```

[次へ] をクリックします。

8. (オプション) [設定] では、再試行ポリシー、デッドレターキュー (DLQ)、暗号化の設定を設定できます。または、デフォルト値のままにします。
9. [許可] で、[既存のロールを使用] を選択してから、`FIS_RecurringExperiment_role`を検索します。
10. [次へ] をクリックします。
11. [スケジュールの確認と作成] で、スケジューラーの詳細を確認し、[スケジュールの作成] を選択します。

ステップ 3: 実験を検証

AWS FIS 実験が予定通りに実行されたことを確認するには

1. https://console.aws.amazon.com/fis/ で AWS FIS コンソール を開きます。
2. 左のナビゲーションペインで [実験] を選択します。
3. スケジュールを作成して 5 分後に、実験の実行が表示されます。

ステップ 4: クリーンアップする

Amazon EventBridge スケジューラーを無効にするには

1. https://console.aws.amazon.com/events/ で Amazon EventBridge 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 [スケジュール] を選択します。
3. 新しく作成したスケジューラーを選択し、[無効] を選択します。

のアクション AWS FIS

アクションは、 AWS Fault Injection Service () を使用してターゲットで実行するフォールトインジェクションアクティビティですAWS FIS。 AWS FIS は、 AWS サービス全体で特定のタイプのターゲットに対して事前設定されたアクションを提供します。実験テンプレートにアクションを追加し、実験の実行に使用します。

内容

アクション識別子

各 AWS FIS アクションには、次の形式の識別子があります。

```
aws:service-name:action-type
```

例えば、以下のアクションは、ターゲットの Amazon EC2 インスタンスを停止します。

```
aws:ec2:stop-instances
```

アクションの完全なリストについては、「AWS FIS アクションリファレンス」を参照してください。を使用してリストを取得するには AWS CLI、「」を参照してくださいアクションを一覧表示する。

アクションパラメータ

一部の AWS FIS アクションには、アクションに固有の追加のパラメータがあります。これらのパラメータは、アクションの実行 AWS FIS 時に に情報を渡すために使用されます。

AWS FIS は、 `aws:ssm:send-command`アクションを使用してカスタム障害タイプをサポートします。このアクションは、SSM エージェントと SSM コマンドドキュメントを使用して、ターゲットインスタンスに障害条件を作成します。`aws:ssm:send-command` アクションには、SSM ドキュメントの Amazon リソースネーム (ARN) を値として取る `documentArn` パラメータが含まれています。実験テンプレートにアクションを追加するときに、パラメータの値を指定します。

`aws:ssm:send-command` アクションのパラメータを指定する方法の詳細については、「aws:ssm:send-command アクションを使用します」を参照してください。

可能な場合は、アクションパラメータ内のロールバック設定を入力できます (ポストアクションともよばれます) 。ポストアクションは、ターゲットをアクションが実行される前の状態に戻します。ポストアクションは、アクション期間に指定された時間後に実行されます。すべてのアクションがポストアクションをサポートできるわけではありません。例えば、アクションによって Amazon EC2 インスタンスが終了した場合、インスタンス終了後にこのインスタンスを回復することはできません。

アクションターゲット

アクションは、指定したターゲットリソースで実行されます。ターゲットを定義したら、アクションを定義するときにその名前を指定できます。

```
"targets": {
    "resource_type": "resource_name"
}
```

AWS FIS アクションは、アクションターゲットに対して次のリソースタイプをサポートします。

- Auto Scaling groups – Amazon EC2 Auto Scaling グループ
- Buckets - Amazon S3 バケット
- Cluster - Amazon EKS クラスター
- Clusters - Amazon ECS クラスターまたは Amazon Aurora DB クラスター
- DBInstances - Amazon RDS DB インスタンス
- Encrypted global tables – Amazon DynamoDB、カスタマーマネージドキーで暗号化されたグローバルテーブル
- グローバルテーブル – Amazon DynamoDB ; グローバルテーブル
- Instances - Amazon EC2 インスタンス

- Nodegroups – Amazon EKS のノードグループ
- Pods - Amazon EKS の Kubernetes ポッド
- ReplicationGroups — ElastiCache レプリケーショングループを再配布する
- Roles – IAM ロール
- SpotInstances – Amazon EC2 スポットインスタンス
- Subnets - VPC サブネット
- Tasks - Amazon ECS タスク
- TransitGateways – トランジットゲートウェイ
- Volumes - Amazon EBS ボリューム

例については、「the section called “アクションの例”」を参照してください。

AWS FIS アクションリファレンス

このリファレンスでは、アクションパラメータや必要な IAM アクセス許可に関する情報など AWS FIS、 の一般的なアクションについて説明します。 AWS FIS コンソールまたは AWS Command Line Interface () の list-actions コマンドを使用して、サポートされている AWS FIS アクションを一覧表示することもできますAWS CLI。

詳細については、「のアクション AWS FIS」および「AWS Fault Injection Service と IAM の連携方法」を参照してください。

アクション

- フォールトインジェクションアクション
- Wait アクション
- Amazon CloudWatch アクション
- Amazon DynamoDB のアクション
- Amazon EBS アクション
- Amazon EC2 アクション
- Amazon ECS アクション
- Amazon EKS アクション
- Amazon ElastiCache アクション

- ネットワークアクション
- Amazon RDS アクション
- Amazon S3 のアクション
- Systems Manager アクション

フォールトインジェクションアクション

AWS FIS では、次の障害挿入アクションがサポートされています。

アクション

- aws:fis:inject-api-internal-error
- aws:fis:inject-api-throttle-error
- aws:fis:inject-api-unavailable-error

aws:fis:inject-api-internal-error

ターゲットの IAM ロールからのリクエストに内部エラーを挿入します。

リソースタイプ

- aws:iam:role

パラメータ

- duration - 所要時間。1 分から 12 時間です。 AWS FIS API では、値は ISO 8601 形式の文字列です。例えば、PT1M は 1 分を表します。 AWS FIS コンソールで、秒数、分数、または時間数を入力します。
- service – ターゲット AWS API 名前空間。サポート対象の値は `ec2` です。
- percentage - 障害の注入対象になるコールの割合 (1 ~ 100)。
- operations - 障害の注入対象になる操作 (カンマ区切り)。`ec2` 名前空間の API アクションのリストについては、「Amazon EC2 API リファレンス」のアクションを参照してください。

アクセス許可

- `fis:InjectApiInternalError`

aws:fis:inject-api-throttle-error

ターゲットの IAM ロールからのリクエストにスロットリングエラーを挿入します。

リソースタイプ

- aws:iam:role

パラメータ

- duration - 所要時間。1 分から 12 時間です。 AWS FIS API では、値は ISO 8601 形式の文字列です。例えば、PT1M は 1 分を表します。 AWS FIS コンソールで、秒数、分数、または時間数を入力します。
- service – ターゲット AWS API 名前空間。サポート対象の値は `ec2` です。
- percentage - 障害の注入対象になるコールの割合 (1 ~ 100)。
- operations - 障害の注入対象になる操作 (カンマ区切り)。`ec2` 名前空間の API アクションのリストについては、「Amazon EC2 API リファレンス」のアクションを参照してください。

アクセス許可

- `fis:InjectApiThrottleError`

aws:fis:inject-api-unavailable-error

ターゲットの IAM ロールからのリクエストに Unavailable エラーを挿入します。

リソースタイプ

- aws:iam:role

パラメータ

- duration - 所要時間。1 分から 12 時間です。 AWS FIS API では、値は ISO 8601 形式の文字列です。例えば、PT1M は 1 分を表します。 AWS FIS コンソールで、秒数、分数、または時間数を入力します。
- service – ターゲット AWS API 名前空間。サポート対象の値は `ec2` です。
- percentage - 障害の注入対象になるコールの割合 (1 ~ 100)。

- operations - 障害の注入対象になる操作 (カンマ区切り)。`ec2` 名前空間の API アクションのリストについては、「Amazon EC2 API リファレンス」のアクションを参照してください。

アクセス許可

- `fis:InjectApiUnavailableError`

Wait アクション

AWS FIS は、次の待機アクションをサポートします。

aws:fis:wait

AWS FIS 待機アクションを実行します。

パラメータ

- duration - 所要時間。1 分から 12 時間です。 AWS FIS API では、値は ISO 8601 形式の文字列です。例えば、PT1M は 1 分を表します。 AWS FIS コンソールで、秒数、分数、または時間数を入力します。

アクセス許可

- なし

Amazon CloudWatch アクション

AWS FIS は、次の Amazon CloudWatch アクションをサポートします。

aws:cloudwatch:assert-alarm-state

指定したアラームが指定したアラーム状態のいずれかになっていることを確認します。

リソースタイプ

- なし

パラメータ

- alarmArns - アラームの ARN (カンマ区切り)。最大 5 つのアラームを指定できます。
- alarmStates - アラーム状態 (カンマ区切り)。指定できるアラーム状態は、`OK`、`ALARM`、および `INSUFFICIENT_DATA` です。

アクセス許可

- `cloudwatch:DescribeAlarms`

Amazon DynamoDB のアクション

AWS FIS は、次の Amazon DynamoDB アクションをサポートします。

aws:dynamodb:global-table-pause-replication

Amazon DynamoDB グローバルテーブルのレプリケーションを任意のレプリカテーブルに一時停止します。テーブルは、アクションが開始した後、最大 5 分間はレプリケートが続行されることがあります。

次のステートメントは、ターゲット DynamoDB グローバルテーブルのポリシーに動的に追加されます。

```
{
   "Statement":[
      {
         "Sid": "DoNotModifyFisDynamoDbPauseReplicationEXPxxxxxxxxxxxxxx"
         "Effect":"Deny",
         "Principal":{
            "AWS":"arn:aws:iam::123456789012:role/aws-service-role/
replication.dynamodb.amazonaws.com/AWSServiceRoleForDynamoDBReplication"
         },
         "Action":[
            "dynamodb:GetItem",
            "dynamodb:PutItem",
            "dynamodb:UpdateItem",
            "dynamodb:DeleteItem",
            "dynamodb:DescribeTable",
            "dynamodb:UpdateTable",
            "dynamodb:Scan",
```

```
            "dynamodb:DescribeTimeToLive",
            "dynamodb:UpdateTimeToLive"
         ],
         "Resource":"arn:aws:dynamodb:us-east-1:123456789012:table/ExampleGlobalTable",
         "Condition": {
            "DateLessThan": {
            "aws:CurrentTime": "2024-04-10T09:51:41.511Z"
         }
       }
      }
   ]
}
```

次のステートメントは、ターゲット DynamoDB グローバルテーブルのストリームのポリシーに動的に追加されます。

```
{
   "Statement":[
      {
         "Sid": "DoNotModifyFisDynamoDbPauseReplicationEXPxxxxxxxxxxxxxx"
         "Effect":"Deny",
         "Principal":{
            "AWS":"arn:aws:iam::123456789012:role/aws-service-role/
replication.dynamodb.amazonaws.com/AWSServiceRoleForDynamoDBReplication"
         },
         "Action":[
            "dynamodb:GetRecords",
            "dynamodb:DescribeStream",
            "dynamodb:GetShardIterator"
         ],
         "Resource":"arn:aws:dynamodb:us-east-1:123456789012:table/ExampleGlobalTable/
stream/2023-08-31T09:50:24.025",
         "Condition": {
            "DateLessThan": {
            "aws:CurrentTime": "2024-04-10T09:51:41.511Z"
         }
      }
   ]
}
```

ターゲットテーブルまたはストリームにアタッチされたリソースポリシーがない場合、実験中はリソースポリシーが作成され、実験が終了すると自動的に削除されます。それ以外の場合、障害ステー

トメントは既存のポリシーに挿入され、既存のポリシーステートメントに追加の変更はありません。その後、障害ステートメントは実験の最後にポリシーから削除されます。

リソースタイプ

- aws:dynamodb:global-table

パラメータ

- duration – AWS FIS API では、値は ISO 8601 形式の文字列です。例えば、PT1M は 1 分を表します。 AWS FIS コンソールで、秒数、分数、または時間数を入力します。

アクセス許可

- `dynamodb:PutResourcePolicy`
- `dynamodb:DeleteResourcePolicy`
- `dynamodb:GetResourcePolicy`
- `dynamodb:DescribeTable`
- `tag:GetResources`

Amazon EBS アクション

AWS FIS は、次の Amazon EBS アクションをサポートします。

aws:ebs:pause-volume-io

ターゲット EBS ボリュームの I/O オペレーションを一時停止します。ターゲットボリュームは同じアベイラビリティーゾーンになければならず、Nitro System に基づいて構築されたインスタンスにアタッチしておく必要があります。ボリュームは、Outpost 上のインスタンスにはアタッチできません。

Amazon EC2 コンソールで実験を開始する方法については、『Amazon EC2 ユーザーガイド』の「Amazon EBS での障害テスト」を参照してください。

リソースタイプ

- aws:ec2:ebs-volume

パラメータ

- duration - 所要時間。1 秒から 12 時間です。 AWS FIS API では、値は ISO 8601 形式の文字列です。たとえば、PT1M は 1 分、PT5S は 5 秒、PT6H は 6 時間を表します。 AWS FIS コンソールで、秒数、分数、または時間数を入力します。PT5S のように時間が短かければ、I/O は指定された時間だけ一時停止するはずですが、実験の初期化には時間がかかるため、実験が完了するまでの時間が長くなることがあります。

アクセス許可

- `ec2:DescribeVolumes`
- `ec2:PauseVolumeIO`
- `tag:GetResources`

Amazon EC2 アクション

AWS FIS は、次の Amazon EC2 アクションをサポートします。

アクション

- aws:ec2:api-insufficient-instance-capacity-error
- aws:ec2:asg-insufficient-instance-capacity-error
- aws:ec2:reboot-instances
- aws:ec2:send-spot-instance-interruptions
- aws:ec2:stop-instances
- aws:ec2:terminate-instances

AWS FIS は、 AWS Systems Manager SSM エージェントを介した障害挿入アクションもサポートしています。システムマネージャーは EC2 インスタンスで実行するアクションを定義する SSM ドキュメントを使用します。独自のドキュメントを使用したカスタム障害の注入や、事前設定済みの SSM ドキュメントの使用が可能です。詳細については、「the section called “SSM ドキュメントを使用する”」を参照してください。

aws:ec2:api-insufficient-instance-capacity-error

ターゲットの IAM ロールからのリクエストで `InsufficientInstanceCapacity` エラーを挿入します。サポートされているオペレーションは RunInstances、、 CreateCapacityReservation、

StartInstances、 CreateFleet コールです。複数のアベイラビリティーゾーンでのキャパシティタスクを含むリクエストはサポートされていません。このアクションでは、リソースタグ、フィルタ、またはパラメータを使用したターゲットの定義はサポートされていません。

リソースタイプ

- aws:iam:role

パラメータ

- duration – AWS FIS API では、値は ISO 8601 形式の文字列です。例えば、PT1M は 1 分を表します。 AWS FIS コンソールで、秒数、分数、または時間数を入力します。
- availabilityzoneIdentifiers - アベイラビリティーゾーンのカンマ区切りリスト。ゾーン ID (`"use1-az1, use1-az2"` など) とゾーン名 (`"us-east-1a"` など) をサポートします。
- percentage - 障害の注入対象になるコールの割合 (1 ~ 100)。

アクセス許可

- 条件キー `ec2:FisActionId` の値の `ec2:InjectApiError` を `aws:ec2:api-insufficient-instance-capacity-error` に設定し、`ec2:FisTargetArns` 条件キーをターゲットの IAM ロールに設定します。

ポリシーの例については、「例: `ec2:InjectApiError` の条件キーを使用します。」を参照してください。

aws:ec2:asg-insufficient-instance-capacity-error

ターゲットの Auto Scaling グループからのリクエストで `InsufficientInstanceCapacity` エラーを挿入します。このアクションは、起動テンプレートを使用する Auto Scaling グループのみをサポートします。インスタンス容量不足エラーに関する詳細については、「Amazon EC2 ユーザーガイド」を参照してください。

リソースタイプ

- aws:ec2:autoscaling-group

パラメータ

- duration – AWS FIS API では、値は ISO 8601 形式の文字列です。例えば、PT1M は 1 分を表します。 AWS FIS コンソールで、秒数、分数、または時間数を入力します。
- availabilityzoneIdentifiers - アベイラビリティーゾーンのカンマ区切りリスト。ゾーン ID (`"use1-az1, use1-az2"` など) とゾーン名 (`"us-east-1a"` など) をサポートします。
- percentage - オプション。障害を挿入するターゲット Auto Scaling グループの起動リクエストの割合 (1 ~ 100)。デフォルトは 100 です。

アクセス許可

- `ec2:InjectApiError`条件キー ec2:FisActionId 値を に設定`aws:ec2:asg-insufficient-instance-capacity-error`し、`ec2:FisTargetArns`条件キーをターゲット Auto Scaling グループに設定します。
- `autoscaling:DescribeAutoScalingGroups`

ポリシーの例については、「例: `ec2:InjectApiError` の条件キーを使用します。」を参照してください。

aws:ec2:reboot-instances

ターゲット EC2 インスタンスで Amazon EC2 API アクションを実行します。 RebootInstances

リソースタイプ

- aws:ec2:instance

パラメータ

- なし

アクセス許可

- `ec2:RebootInstances`
- `ec2:DescribeInstances`

AWS マネージドポリシー

- AWSFaultInjectionSimulatorEC2Access

aws:ec2:send-spot-instance-interruptions

ターゲットのスポットインスタンスを中断します。スポットインスタンスの中断通知を送信して、スポットインスタンスを中断する 2 分前にスポットインスタンスをターゲットにします。中断時間は、指定された期間BeforeInterruptionパラメータによって決まります。中断時間の 2 分後、スポットインスタンスは中断動作に応じて終了するか停止します。 AWS FIS によって停止されたスポットインスタンスは、ユーザーにより再起動されるまで停止状態を維持します。

アクションの開始直後に、ターゲットインスタンスは、EC2 インスタンスの再調整の推奨指示を受け取ります。期間BeforeInterruption を指定した場合、再調整に関する推奨事項と中断通知の間に遅延が生じる可能性があります。

詳細については、「the section called “スポットインスタンスの中断をテストする”」を参照してください。または、Amazon EC2 コンソールで実験を開始する方法については、『Amazon EC2 ユーザーガイド』の「スポットインスタンスの中断を開始する」を参照してください。

リソースタイプ

- aws:ec2:spot-instance

パラメータ

- durationBeforeInterruption - インスタンスが中断されるまでの待機時間。2 分から 15 分です。 AWS FIS API では、値は ISO 8601 形式の文字列です。例えば、PT2M は 2 分を表します。 AWS FIS コンソールで、分数を入力します。

アクセス許可

- `ec2:SendSpotInstanceInterruptions`
- `ec2:DescribeInstances`

AWS マネージドポリシー

- AWSFaultInjectionSimulatorEC2Access

aws:ec2:stop-instances

ターゲット EC2 インスタンスで Amazon EC2 API アクションを実行します。 StopInstances

リソースタイプ

- aws:ec2:instance

パラメータ

- startInstancesAfterDuration - オプション。インスタンスの起動までの待機時間。1 分から 12 時間です。 AWS FIS API では、値は ISO 8601 形式の文字列です。例えば、PT1M は 1 分を表します。 AWS FIS コンソールで、秒数、分数、または時間数を入力します。インスタンスに暗号化された EBS ボリュームがある場合は、ボリュームの暗号化に使用される KMS キーにアクセス AWS FIS 許可を付与するか、実験ロールを KMS キーポリシーに追加する必要があります。
- completeIfInstancesTerminated - オプション。true の場合、そして `startInstancesAfterDuration` も true の場合、ターゲットとなる EC2 インスタンスが FIS 外部の別のリクエストによって終了され、再起動できなくなっても、このアクションは失敗しません。例えば、Auto Scaling グループは、このアクションが完了する前に管理され、停止された EC2 インスタンスを終了することがあります。デフォルトは False です。

アクセス許可

- `ec2:StopInstances`
- `ec2:StartInstances`
- `ec2:DescribeInstances` - オプション。アクションの終了時にインスタンスの状態を検証するために、完全なIfInstances終了で必要です。
- `kms:CreateGrant` - オプション。暗号化されたボリュームを持つインスタンスを再起動するInstancesAfter開始期間に必要です。

AWS マネージドポリシー

- AWSFaultInjectionSimulatorEC2Access

aws:ec2:terminate-instances

ターゲット EC2 インスタンスで Amazon EC2 API アクションを実行します。 TerminateInstances

リソースタイプ

- aws:ec2:instance

パラメータ

- なし

アクセス許可

- `ec2:TerminateInstances`
- `ec2:DescribeInstances`

AWS マネージドポリシー

- AWSFaultInjectionSimulatorEC2Access

Amazon ECS アクション

AWS FIS は、次の Amazon ECS アクションをサポートします。

アクション

- aws:ecs:drain-container-instances
- aws:ecs:stop-task
- aws:ecs:task-cpu-stress
- aws:ecs:task-io-stress
- aws:ecs:task-kill-process
- aws:ecs:task-network-blackhole-port
- aws:ecs:task-network-latency
- aws:ecs:task-network-packet-loss

aws:ecs:drain-container-instances

Amazon ECS API アクションを実行UpdateContainerInstancesStateして、ターゲットクラスターの基盤となる Amazon EC2 インスタンスの指定された割合をドレインします。

リソースタイプ

- aws:ecs:cluster

パラメータ

- drainagePercentage - パーセンテージ（1 から 100）。
- duration - 所要時間。1 分から 12 時間です。 AWS FIS API では、値は ISO 8601 形式の文字列です。例えば、PT1M は 1 分を表します。 AWS FIS コンソールで、秒数、分数、または時間数を入力します。

アクセス許可

- `ecs:DescribeClusters`
- `ecs:UpdateContainerInstancesState`
- `ecs:ListContainerInstances`
- `tag:GetResources`

AWS マネージドポリシー

- AWSFaultInjectionSimulatorECSAccess

aws:ecs:stop-task

Amazon ECS API アクションを実行StopTaskして、ターゲットタスクを停止します。

リソースタイプ

- aws:ecs:task

パラメータ

- なし

アクセス許可

- `ecs:DescribeTasks`

- `ecs:ListTasks`
- `ecs:StopTask`
- `tag:GetResources`

AWS マネージドポリシー

- AWSFaultInjectionSimulatorECSAccess

aws:ecs:task-cpu-stress

ターゲットタスクに CPU ストレスを実行します。AWSFIS-Run-CPU-Stress SSM ドキュメントを使用します。タスクは によって管理される必要があります AWS Systems Manager。詳細については、「ECS タスクアクションを使用します」を参照してください。

リソースタイプ

- aws:ecs:task

パラメータ

- duration - ストレステストの所要時間 (ISO 8601 形式) 。
- percent - オプション。0 (無負荷) から 100 (全負荷) までの目標負荷率。デフォルトは 100 です。
- workers - オプション。使用するストレッサーの数。デフォルト値は 0 で、すべてのストレッサーを使用します。
- installDependencies - オプション。値が `True` の場合、ターゲットインスタンスに必要な依存関係がインストールされていなければ、SSM エージェントのサイドカーコンテナに Systems Manager によってインストールされます。デフォルトは `True` です。依存関係は stress-ng です。

アクセス許可

- `ssm:SendCommand`
- `ssm:ListCommands`
- `ssm:CancelCommand`

aws:ecs:task-io-stress

ターゲットタスクに I/O ストレスを実行します。AWSFIS-Run-IO-Stress SSM ドキュメントを使用します。タスクは によって管理される必要があります AWS Systems Manager。詳細については、「ECS タスクアクションを使用します」を参照してください。

リソースタイプ

- aws:ecs:task

パラメータ

- duration - ストレステストの所要時間（ISO 8601 形式）。
- percent - オプション。ストレステスト中に使用するファイルシステム上の空き領域の割合。デフォルトは 80% です。
- workers - オプション。ワーカー数 シーケンシャル、ランダム、およびメモリマップされた読み取り/書き込み操作、強制同期、およびキャッシュドロップの組み合わせを実行するワーカー。複数の子プロセスが同じファイルに対して異なる I/O 操作を実行します。デフォルト は 1 です。
- installDependencies - オプション。値が `True` の場合、ターゲットインスタンスに必要な依存関係がインストールされていなければ、SSM エージェントのサイドカーコンテナに Systems Manager によってインストールされます。デフォルトは `True` です。依存関係は stress-ng です。

アクセス許可

- `ssm:SendCommand`
- `ssm:ListCommands`
- `ssm:CancelCommand`

aws:ecs:task-kill-process

killall コマンドを使用して、タスク内の指定されたプロセスを停止します。AWSFIS-Run-Kill-Process SSM ドキュメントを使用します。タスク定義は `pidMode` を `task` に設定している必要があります。タスクは によって管理される必要があります AWS Systems Manager。詳細については、「ECS タスクアクションを使用します」を参照してください。

リソースタイプ

- aws:ecs:task

パラメータ

- processName – 停止するプロセスの名前。
- signal - オプション。コマンドと一緒に送信するシグナル。指定できる値は、`SIGTERM` (受信者が無視することを選択できる) と `SIGKILL` (無視できない) です。デフォルト: `SIGTERM`。
- installDependencies - オプション。値が `True` の場合、ターゲットインスタンスに必要な依存関係がインストールされていなければ、SSM エージェントのサイドカーコンテナに Systems Manager によってインストールされます。デフォルトは `True` です。依存関係は killall です。

アクセス許可

- `ssm:SendCommand`
- `ssm:ListCommands`
- `ssm:CancelCommand`

aws:ecs:task-network-blackhole-port

指定されたプロトコルおよびポートのインバウンドトラフィックまたはアウトバウンドトラフィックをドロップします。AWSFIS-Run-Network-Blackhole-Port SSM ドキュメントを使用します。タスク定義は `pidMode` を `task` に設定している必要があります。タスクは によって管理される必要があります AWS Systems Manager。タスク定義では `networkMode` を `bridge` に設定できません。詳細については、「ECS タスクアクションを使用します」を参照してください。

リソースタイプ

- aws:ecs:task

パラメータ

- duration - テストの所要時間 (ISO 8601 形式) 。
- port - ポート番号。
- trafficType - トラフィックの種類。指定できる値は `ingress` および `egress` です。

- protocol - オプション。プロトコル。指定できる値は `tcp` および udp です。デフォルトは `tcp` です。
- installDependencies - オプション。値が `True` の場合、ターゲットインスタンスに必要な依存関係がインストールされていなければ、SSM エージェントのサイドカーコンテナに Systems Manager によってインストールされます。デフォルトは `True` です。依存関係は、atd、dig、および iptables です。

アクセス許可

- `ssm:SendCommand`
- `ssm:ListCommands`
- `ssm:CancelCommand`

aws:ecs:task-network-latency

特定のソースへのトラフィックまたは特定のソースからのトラフィックのためのツール tc を使用してネットワークインターフェイスにレイテンシーとジッタを追加します。AWSFIS-Run-Network-Latency-Sources SSM ドキュメントを使用します。タスク定義は `pidMode` を `task` に設定している必要があります。タスクは によって管理される必要があります AWS Systems Manager。タスク定義では `networkMode` を `bridge` に設定できません。詳細については、「ECS タスクアクションを使用します」を参照してください。

リソースタイプ

- aws:ecs:task

パラメータ

- duration - テストの所要時間 (ISO 8601 形式) 。
- interface - オプション。ネットワークインターフェース。デフォルト: `eth0`。
- delayMilliseconds - オプション。遅延 (ミリ秒単位) 。デフォルトは 200 です。
- jitterMilliseconds - オプション。ジッタ (ミリ秒単位)。デフォルトは 10 です。
- sources - オプション。ソースはカンマで区切ります。指定できる値は、IPv4 アドレス、IPv4 CIDR ブロック、ドメイン名、`DYNAMODB`、および `S3` です。`DYNAMODB` または `S3` を指定した場合、これは現在のリージョンのリージョナルエンドポイントにのみ適用されます。デフォルトは 0.0.0/0 で、すべての IPv4 トラフィックに一致します。

- installDependencies - オプション。値が `True` の場合、ターゲットインスタンスに必要な依存関係がインストールされていなければ、SSM エージェントのサイドカーコンテナに Systems Manager によってインストールされます。デフォルトは `True` です。依存関係は、atd、dig、jq および tc です。

アクセス許可

- `ssm:SendCommand`
- `ssm:ListCommands`
- `ssm:CancelCommand`

aws:ecs:task-network-packet-loss

tc ツールを使用してネットワークインターフェイスへのパッケージの損失を追加します。AWSFIS-Run-Network-Packet-Loss-Sources SSM ドキュメントを使用します。タスク定義は `pidMode` を `task` に設定している必要があります。タスクは によって管理される必要があります AWS Systems Manager。タスク定義では `networkMode` を `bridge` に設定できません。詳細については、「ECS タスクアクションを使用します」を参照してください。

リソースタイプ

- aws:ecs:task

パラメータ

- duration - テストの所要時間（ISO 8601 形式）。
- interface - オプション。ネットワークインターフェース。デフォルト: `eth0`。
- lossPercent - オプション。パケット損失の割合。デフォルトは 7% です。
- sources - オプション。ソースはカンマで区切ります。指定できる値は、IPv4 アドレス、IPv4 CIDR ブロック、ドメイン名、`DYNAMODB`、および `S3` です。`DYNAMODB` または `S3` を指定した場合、これは現在のリージョンのリージョナルエンドポイントにのみ適用されます。デフォルトは 0.0.0/0 で、すべての IPv4 トラフィックに一致します。
- installDependencies - オプション。値が `True` の場合、ターゲットインスタンスに必要な依存関係がインストールされていなければ、SSM エージェントのサイドカーコンテナに Systems Manager によってインストールされます。デフォルトは `True` です。依存関係は、atd、dig、jq および tc です。

アクセス許可

- `ssm:SendCommand`
- `ssm:ListCommands`
- `ssm:CancelCommand`

Amazon EKS アクション

AWS FIS は、次の Amazon EKS アクションをサポートします。

アクション

- aws:eks:inject-kubernetes-custom-resource
- aws:eks:pod-cpu-stress
- aws:eks:pod-delete
- aws:eks:pod-io-stress
- aws:eks:pod-memory-stress
- aws:eks:pod-network-blackhole-port
- aws:eks:pod-network-latency
- aws:eks:pod-network-packet-loss
- aws:eks:terminate-nodegroup-instances

aws:eks:inject-kubernetes-custom-resource

単一のターゲットクラスターで ChaosMesh または Litmus 実験を実行します。ターゲットクラスターに ChaosMesh または Litmus をインストールする必要があります。

実験テンプレートを作成して `aws:eks:cluster` タイプのターゲットを定義する場合、このアクションは単一の Amazon リソースネーム (ARN) を対象とする必要があります。このアクションでは、リソースタグ、フィルタ、またはパラメータを使用したターゲットの定義はサポートされていません。

をインストールするときは ChaosMesh、適切なコンテナランタイムを指定する必要があります。Amazon EKS バージョン 1.23 以降、デフォルトのランタイムを Docker からcontainerd に変更しました。バージョン 1.24 以降、Docker は削除しました。

リソースタイプ

- aws:eks:cluster

パラメータ

- kubernetesApiVersion - Kubernetes カスタムリソースの API バージョン。指定できる値は `chaos-mesh.org/v1alpha1` | `litmuschaos.io/v1alpha1` です。
- kubernetesKind - Kubernetes カスタムリソースの種類。値は API バージョンに依存します。
  - `chaos-mesh.org/v1alpha1` – 指定可能な値は次のとおりです: | `AWSChaos` | `DNSChaos` | `GCPChaos` | `HTTPChaos` | `IOChaos` | `JVMChaos` | `KernelChaos` | `NetworkChaos` | `PhysicalMachineChaos` | `PodChaos` | `PodHttpChaos` | `PodIOChaos` | `PodNetworkChaos` | `Schedule` | `StressChaos` | `TimeChaos` | 。
  - `litmuschaos.io/v1alpha1` – 有効な値は `ChaosEngine` です。
- kubernetesNamespace - Kubernetes 名前空間。
- kubernetesSpec - JSON 形式の Kubernetes カスタムリソースの `spec` セクション。
- maxDuration – オートメーション実行が完了するまでの最大許容時間。1 分から 12 時間です。AWS FIS API では、値は ISO 8601 形式の文字列です。例えば、PT1M は 1 分を表します。 AWS FIS コンソールで、秒数、分数、または時間数を入力します。

アクセス許可

このアクションには AWS Identity and Access Management (IAM) アクセス許可は必要ありません。このアクションを使用するために必要な権限は、Kubernetes が RBAC 認証を使用して制御します。詳細については、Kubernetes ドキュメントの RBAC 認証の使用を参照してください。Chaos Mesh の詳細については、Chaos Mesh の公式ドキュメントを参照してください。Litmus の詳細については、『Litmus の公式ドキュメント』を参照してください。

aws:eks:pod-cpu-stress

ターゲットポッドに CPU ストレスを実行します。詳細については、「EKS ポッドアクションを使用する」を参照してください。

リソースタイプ

- aws:eks:pod

パラメータ

- duration - ストレステストの所要時間 (ISO 8601 形式) 。
- percent - オプション。0 (無負荷) から 100 (全負荷) までの目標負荷率。デフォルトは 100 です。
- workers - オプション。使用するストレッサーの数。デフォルト値は 0 で、すべてのストレッサーを使用します。
- kubernetesServiceAccount – Kubernetes サービスアカウント。必要な権限の詳細については、「the section called “Kubernetes サービスアカウントを設定する”」を参照してください。
- fisPodContainerImage - オプション。フォールトインジェクターポッドの作成に使用するコンテナイメージ。デフォルトでは、 が提供するイメージが使用されます AWS FIS。詳細については、「the section called “ポッドコンテナイメージ”」を参照してください。
- maxErrorsPercent - オプション。フォールトインジェクションが失敗するまでに障害が発生する可能性のあるターゲットの割合。デフォルトは 0 です。
- fisPodLabels - オプション。FIS によって作成された障害オーケストレーションポッドにアタッチされている Kubernetes ラベル。
- fisPodAnnotations - オプション。FIS によって作成された障害オーケストレーションポッドにアタッチされる Kubernetes アノテーション。
- fisPodSecurityPolicy - オプション。FIS とエフェメラルコンテナによって作成された障害オーケストレーションポッドに使用する Kubernetes セキュリティ標準ポリシー。指定できる値は、`privileged`、`baseline`、および ですrestricted。このアクションは、すべてのポリシーレベルと互換性があります。

アクセス許可

- `eks:DescribeCluster`
- `ec2:DescribeSubnets`
- `tag:GetResources`

AWS マネージドポリシー

- AWSFaultInjectionSimulatorEKSAccess

aws:eks:pod-delete

ターゲットポッドを削除します。詳細については、「EKS ポッドアクションを使用する」を参照してください。

リソースタイプ

- aws:eks:pod

パラメータ

- gracePeriodSeconds - オプション。ポッドが正常に終了するまでの待機時間 (秒単位)。値が 0 の場合、アクションはすぐに実行されます。値が nil の場合、Pod のデフォルトの猶予期間を使用します。
- kubernetesServiceAccount - Kubernetes サービスアカウント 必要な権限の詳細については、「the section called “Kubernetes サービスアカウントを設定する”」を参照してください。
- fisPodContainerImage - オプション。フォールトインジェクターポッドの作成に使用するコンテナイメージ。デフォルトでは、 が提供するイメージが使用されます AWS FIS。詳細については、「the section called “ポッドコンテナイメージ”」を参照してください。
- maxErrorsPercent - オプション。フォールトインジェクションが失敗するまでに障害が発生する可能性のあるターゲットの割合。デフォルトは 0 です。
- fisPodLabels - オプション。FIS によって作成された障害オーケストレーションポッドにアタッチされている Kubernetes ラベル。
- fisPodAnnotations - オプション。FIS によって作成された障害オーケストレーションポッドにアタッチされる Kubernetes アノテーション。
- fisPodSecurityPolicy - オプション。FIS とエフェメラルコンテナによって作成された障害オーケストレーションポッドに使用する Kubernetes セキュリティ標準ポリシー。指定できる値は、`privileged`、`baseline`、および ですrestricted。このアクションは、すべてのポリシーレベルと互換性があります。

アクセス許可

- `eks:DescribeCluster`
- `ec2:DescribeSubnets`
- `tag:GetResources`

AWS マネージドポリシー

- AWSFaultInjectionSimulatorEKSAccess

aws:eks:pod-io-stress

ターゲットポッドに I/O ストレスを実行します。詳細については、「EKS ポッドアクションを使用する」を参照してください。

リソースタイプ

- aws:eks:pod

パラメータ

- duration - ストレステストの所要時間（ISO 8601 形式）。
- workers - オプション。ワーカー数 シーケンシャル、ランダム、およびメモリマップされた読み取り/書き込み操作、強制同期、およびキャッシュドロップの組み合わせを実行するワーカー。複数の子プロセスが同じファイルに対して異なる I/O 操作を実行します。デフォルト は 1 です。
- percent - オプション。ストレステスト中に使用するファイルシステム上の空き領域の割合。デフォルトは 80% です。
- kubernetesServiceAccount – Kubernetes サービスアカウント 必要な権限の詳細については、「the section called “Kubernetes サービスアカウントを設定する”」を参照してください。
- fisPodContainerImage - オプション。フォールトインジェクターポッドの作成に使用するコンテナイメージ。デフォルトでは、 が提供するイメージが使用されます AWS FIS。詳細については、「the section called “ポッドコンテナイメージ”」を参照してください。
- maxErrorsPercent - オプション。フォールトインジェクションが失敗するまでに障害が発生する可能性のあるターゲットの割合。デフォルトは 0 です。
- fisPodLabels - オプション。FIS によって作成された障害オーケストレーションポッドにアタッチされている Kubernetes ラベル。
- fisPodAnnotations - オプション。FIS によって作成された障害オーケストレーションポッドにアタッチされる Kubernetes アノテーション。
- fisPodSecurityPolicy - オプション。FIS とエフェメラルコンテナによって作成された障害オーケストレーションポッドに使用する Kubernetes セキュリティ標準ポリシー。指定できる値は、`privileged`、`baseline`、および ですrestricted。このアクションは、すべてのポリシーレベルと互換性があります。

アクセス許可

- `eks:DescribeCluster`
- `ec2:DescribeSubnets`
- `tag:GetResources`

AWS マネージドポリシー

- AWSFaultInjectionSimulatorEKSAccess

aws:eks:pod-memory-stress

ターゲットポッドにメモリストレスを実行します。詳細については、「EKS ポッドアクションを使用する」を参照してください。

リソースタイプ

- aws:eks:pod

パラメータ

- duration - ストレステストの所要時間（ISO 8601 形式）。
- workers - オプション。使用するストレッサーの数。デフォルト は 1 です。
- percent - オプション。ストレステスト中に使用する仮想メモリの割合。デフォルトは 80% です。
- kubernetesServiceAccount – Kubernetes サービスアカウント 必要な権限の詳細については、「the section called “Kubernetes サービスアカウントを設定する”」を参照してください。
- fisPodContainerImage - オプション。フォールトインジェクターポッドの作成に使用するコンテナイメージ。デフォルトでは、 が提供するイメージが使用されます AWS FIS。詳細については、「the section called “ポッドコンテナイメージ”」を参照してください。
- maxErrorsPercent - オプション。フォールトインジェクションが失敗するまでに障害が発生する可能性のあるターゲットの割合。デフォルトは 0 です。
- fisPodLabels - オプション。FIS によって作成された障害オーケストレーションポッドにアタッチされている Kubernetes ラベル。
- fisPodAnnotations - オプション。FIS によって作成された障害オーケストレーションポッドにアタッチされる Kubernetes アノテーション。

- fisPodSecurityPolicy - オプション。FIS とエフェメラルコンテナによって作成された障害オーケストレーションポッドに使用する Kubernetes セキュリティ標準ポリシー。指定できる値は、`privileged`、`baseline`、および ですrestricted。このアクションは、すべてのポリシーレベルと互換性があります。

アクセス許可

- `eks:DescribeCluster`
- `ec2:DescribeSubnets`
- `tag:GetResources`

AWS マネージドポリシー

- AWSFaultInjectionSimulatorEKSAccess

aws:eks:pod-network-blackhole-port

プロトコルおよびポートのインバウンドトラフィックまたはアウトバウンドトラフィックをドロップします。Kubernetes セキュリティ標準privilegedポリシーとのみ互換性があります。詳細については、「EKS ポッドアクションを使用する」を参照してください。

リソースタイプ

- aws:eks:pod

パラメータ

- duration - テストの所要時間（ISO 8601 形式）。
- protocol - オプション。プロトコル。指定できる値は `tcp` および `udp` です。デフォルトは `tcp` です。
- trafficType - トラフィックの種類。指定できる値は `ingress` および `egress` です。
- port - ポート番号。
- kubernetesServiceAccount - Kubernetes サービスアカウント 必要な権限の詳細については、「the section called “Kubernetes サービスアカウントを設定する”」を参照してください。

- fisPodContainerImage - オプション。フォールトインジェクターポッドの作成に使用するコンテナイメージ。デフォルトでは、 が提供するイメージが使用されます AWS FIS。詳細については、「the section called “ポッドコンテナイメージ”」を参照してください。
- maxErrorsPercent - オプション。フォールトインジェクションが失敗するまでに障害が発生する可能性のあるターゲットの割合。デフォルトは 0 です。
- fisPodLabels - オプション。FIS によって作成された障害オーケストレーションポッドにアタッチされている Kubernetes ラベル。
- fisPodAnnotations - オプション。FIS によって作成された障害オーケストレーションポッドにアタッチされる Kubernetes アノテーション。

アクセス許可

- `eks:DescribeCluster`
- `ec2:DescribeSubnets`
- `tag:GetResources`

AWS マネージドポリシー

- AWSFaultInjectionSimulatorEKSAccess

aws:eks:pod-network-latency

特定のソースへのトラフィックまたは特定のソースからのトラフィックのためのツール tc を使用してネットワークインターフェイスにレイテンシーとジッタを追加します。Kubernetes セキュリティ標準`privileged`ポリシーとのみ互換性があります。詳細については、「EKS ポッドアクションを使用する」を参照してください。

リソースタイプ

- aws:eks:pod

パラメータ

- duration - テストの所要時間 (ISO 8601 形式) 。
- interface - オプション。ネットワークインターフェース。デフォルト: `eth0`。
- delayMilliseconds - オプション。遅延 (ミリ秒単位) 。デフォルトは 200 です。

- jitterMilliseconds - オプション。ジッタ (ミリ秒単位)。デフォルトは 10 です。
- sources - オプション。ソースはカンマで区切ります。指定できる値は、IPv4 アドレス、IPv4 CIDR ブロック、ドメイン名、DYNAMODB、および S3 です。DYNAMODB または S3 を指定した場合、これは現在のリージョンのリージョナルエンドポイントにのみ適用されます。デフォルトは 0.0.0/0 で、すべての IPv4 トラフィックに一致します。
- kubernetesServiceAccount - Kubernetes サービスアカウント 必要な権限の詳細については、「the section called “Kubernetes サービスアカウントを設定する”」を参照してください。
- fisPodContainerImage - オプション。フォールトインジェクターポッドの作成に使用するコンテナイメージ。デフォルトでは、 が提供するイメージが使用されます AWS FIS。詳細については、「the section called “ポッドコンテナイメージ”」を参照してください。
- maxErrorsPercent - オプション。フォールトインジェクションが失敗するまでに障害が発生する可能性のあるターゲットの割合。デフォルトは 0 です。
- fisPodLabels - オプション。FIS によって作成された障害オーケストレーションポッドにアタッチされている Kubernetes ラベル。
- fisPodAnnotations - オプション。FIS によって作成された障害オーケストレーションポッドにアタッチされる Kubernetes アノテーション。

アクセス許可

- `eks:DescribeCluster`
- `ec2:DescribeSubnets`
- `tag:GetResources`

AWS マネージドポリシー

- AWSFaultInjectionSimulatorEKSAccess

aws:eks:pod-network-packet-loss

tc ツールを使用してネットワークインターフェイスへのパッケージの損失を追加します。Kubernetes セキュリティ標準privilegedポリシーとのみ互換性があります。詳細については、「EKS ポッドアクションを使用する」を参照してください。

リソースタイプ

- aws:eks:pod

パラメータ

- duration - テストの所要時間（ISO 8601 形式）。
- interface - オプション。ネットワークインターフェース。デフォルト: `eth0`。
- lossPercent - オプション。パケット損失の割合。デフォルトは 7% です。
- sources - オプション。ソースはカンマで区切ります。指定できる値は、IPv4 アドレス、IPv4 CIDR ブロック、ドメイン名、`DYNAMODB`、および `S3` です。`DYNAMODB` または `S3` を指定した場合、これは現在のリージョンのリージョナルエンドポイントにのみ適用されます。デフォルトは 0.0.0/0 で、すべての IPv4 トラフィックに一致します。
- kubernetesServiceAccount - Kubernetes サービスアカウント 必要な権限の詳細については、「the section called “Kubernetes サービスアカウントを設定する”」を参照してください。
- fisPodContainerImage - オプション。フォールトインジェクターポッドの作成に使用するコンテナイメージ。デフォルトでは、 が提供するイメージが使用されます AWS FIS。詳細については、「the section called “ポッドコンテナイメージ”」を参照してください。
- maxErrorsPercent - オプション。フォールトインジェクションが失敗するまでに障害が発生する可能性のあるターゲットの割合。デフォルトは 0 です。
- fisPodLabels - オプション。FIS によって作成された障害オーケストレーションポッドにアタッチされている Kubernetes ラベル。
- fisPodAnnotations - オプション。FIS によって作成された障害オーケストレーションポッドにアタッチされる Kubernetes アノテーション。

アクセス許可

- `eks:DescribeCluster`
- `ec2:DescribeSubnets`
- `tag:GetResources`

AWS マネージドポリシー

- AWSFaultInjectionSimulatorEKSAccess

aws:eks:terminate-nodegroup-instances

ターゲットノードグループTerminateInstancesで Amazon EC2 API アクションを実行します。

リソースタイプ

- aws:eks:nodegroup

パラメータ

- instanceTerminationPercentage - 終了するインスタンスの割合 (1 ~ 100)。

アクセス許可

- `ec2:DescribeInstances`
- `ec2:TerminateInstances`
- `eks:DescribeNodegroup`
- `tag:GetResources`

AWS マネージドポリシー

- AWSFaultInjectionSimulatorEKSAccess

Amazon ElastiCache アクション

AWS FIS は、次の ElastiCache アクションをサポートします。

aws:elasticache:interrupt-cluster-az-power

ターゲットの Redis レプリケーショングループの指定されたアベイラビリティーゾーンのノードへの電力を中断します。プライマリノードがターゲットの場合、レプリケーションの遅延が最も小さいリードレプリカがプライマリに昇格されます。このアクションの間、指定されたアベイラビリティーゾーンのリードレプリカの置き換えはブロックされます。つまり、ターゲットのレプリケーショングループは少ない容量で動作します。

リソースタイプ

- aws:elasticache:redis-replicationgroup

パラメータ

- duration - 所要時間。1 分から 12 時間です。 AWS FIS API では、値は ISO 8601 形式の文字列です。例えば、PT1M は 1 分を表します。 AWS FIS コンソールで、秒数、分数、または時間数を入力します。

アクセス許可

- `elasticache:InterruptClusterAzPower`
- `elasticache:DescribeReplicationGroups`
- `tag:GetResources`

ネットワークアクション

AWS FIS は、次のネットワークアクションをサポートします。

アクション

- aws:network:disrupt-connectivity
- aws:network:route-table-disrupt-cross-region-connectivity
- aws:network:transit-gateway-disrupt-cross-region-connectivity

aws:network:disrupt-connectivity

ターゲットサブネットへの指定されたトラフィックを拒否します。ネットワーク ACLs。

リソースタイプ

- aws:ec2:subnet

パラメータ

- scope - 拒否するトラフィックのタイプ。スコープが でない場合`all`、ネットワーク ACLsです。指定できる値は以下のとおりです。
  - `all` - サブネットに出入りするすべてのトラフィックを拒否します。このオプションでは、サブネット内のネットワークインターフェースに出入りするトラフィックを含め、サブネット内トラフィックが許可されることに注意してください。

- `availability-zone` - 他のアベイラビリティーゾーン内のサブネットとの間の VPC 内トラフィックを拒否します。VPC でターゲットにできるサブネットの最大数は 30 です。
- dynamodb - 現在のリージョンの DynamoDB のリージョナルエンドポイントとの間のトラフィックを拒否します。
- `prefix-list` - 指定されたプレフィックスリストとの間で送受信されるトラフィックを拒否します。
- `s3` - 現在のリージョンの Amazon S3 のリージョンエンドポイントとの間のトラフィックを拒否します。
- `vpc` - VPC に出入りするトラフィックを拒否します。

- duration - 所要時間。1 分から 12 時間です。 AWS FIS API では、値は ISO 8601 形式の文字列です。例えば、PT1M は 1 分を表します。 AWS FIS コンソールで、秒数、分数、または時間数を入力します。
- prefixListIdentifier - スコープが`prefix-list` の場合、これはカスタマー管理プレフィックスリストの識別子です。名前、ID、または ARN を指定できます。このプレフィックスリストは、最大 10 エントリです。

アクセス許可

- `ec2:CreateNetworkAcl` - managedByFIS=true というタグが付いたネットワーク ACL を作成します。
- `ec2:CreateNetworkAclEntry` - ネットワーク ACL には managedByFIS=true というタグが付いている必要があります。
- `ec2:CreateTags`
- `ec2:DeleteNetworkAcl` - ネットワーク ACL には managedByFIS=true というタグが付いている必要があります。
- `ec2:DescribeManagedPrefixLists`
- `ec2:DescribeNetworkAcls`
- `ec2:DescribeSubnets`
- `ec2:DescribeVpcs`
- `ec2:GetManagedPrefixListEntries`
- `ec2:ReplaceNetworkAclAssociation`

AWS マネージドポリシー

- AWSFaultInjectionSimulatorNetworkAccess

aws:network:route-table-disrupt-cross-region-connectivity

ターゲットサブネットから発信され、指定されたリージョンを宛てのトラフィックをブロックします。分離するリージョンのすべてのルートを含むルートテーブルを作成します。FIS がこれらのルートテーブルを作成できるようにするには、 の Amazon VPC クォータ`routes per route table`を 250 に増やし、既存のルートテーブル内のルート数を足します。

リソースタイプ

- aws:ec2:subnet

パラメータ

- `region` - 分離するリージョンのコード (eu-west-1 など)。
- `duration` — アクションが継続する時間の長さ。 AWS FIS API では、値は ISO 8601 形式の文字列です。例えば、PT1M は 1 分を表します。 AWS FIS コンソールで、秒数、分数、または時間数を入力します。

アクセス許可

- `ec2:AssociateRouteTable`
- `ec2:CreateManagedPrefixList` †
- `ec2:CreateNetworkInterface` †
- `ec2:CreateRoute` †
- `ec2:CreateRouteTable` †
- `ec2:CreateTags` †
- `ec2:DeleteManagedPrefixList` †
- `ec2:DeleteNetworkInterface` †
- `ec2:DeleteRouteTable` †
- `ec2:DescribeManagedPrefixLists`
- `ec2:DescribeNetworkInterfaces`

- `ec2:DescribeRouteTables`
- `ec2:DescribeSubnets`
- `ec2:DescribeVpcPeeringConnections`
- `ec2:DescribeVpcs`
- `ec2:DisassociateRouteTable`
- `ec2:GetManagedPrefixListEntries`
- `ec2:ModifyManagedPrefixList` †
- `ec2:ModifyVpcEndpoint`
- `ec2:ReplaceRouteTableAssociation`

† タグ managedByFIS=true を使用してスコープを限定します。

AWS マネージドポリシー

- AWSFaultInjectionSimulatorNetworkAccess

aws:network:transit-gateway-disrupt-cross-region-connectivity

指定されたリージョンを宛先とするターゲットのトランジットゲートウェイピアリングアタッチメントからのトラフィックをブロックします。

リソースタイプ

- aws:ec2:transit-gateway

パラメータ

- `region` - 分離するリージョンのコード (eu-west-1 など)。
- `duration` — アクションが継続する時間の長さ。 AWS FIS API では、値は ISO 8601 形式の文字列です。例えば、PT1M は 1 分を表します。 AWS FIS コンソールで、秒数、分数、または時間数を入力します。

アクセス許可

- `ec2:AssociateTransitGatewayRouteTable`

- `ec2:DescribeTransitGatewayAttachments`
- `ec2:DescribeTransitGatewayPeeringAttachments`
- `ec2:DescribeTransitGateways`
- `ec2:DisassociateTransitGatewayRouteTable`

AWS マネージドポリシー

- AWSFaultInjectionSimulatorNetworkAccess

Amazon RDS アクション

AWS FIS は、次の Amazon RDS アクションをサポートします。

アクション

- aws:rds:failover-db-cluster
- aws:rds:reboot-db-instances

aws:rds:failover-db-cluster

ターゲットの Aurora DB クラスターに対して Amazon RDS API アクション FailoverDBCluster を実行します。

リソースタイプ

- aws:rds:cluster

パラメータ

- なし

アクセス許可

- `rds:FailoverDBCluster`
- `rds:DescribeDBClusters`
- `tag:GetResources`

AWS マネージドポリシー

- AWSFaultInjectionSimulatorRDSAccess

aws:rds:reboot-db-instances

ターゲットの DB インスタンスに対して Amazon RDS API アクション RebootDBInstance を実行します。

リソースタイプ

- aws:rds:db

パラメータ

- forceFailover - オプション。値が true の場合、インスタンスがマルチ AZ の場合は、1 つのアベイラビリティーゾーンから別のアベイラビリティーゾーンへのフェイルオーバーを強制的に実行できます。デフォルトは False です。

アクセス許可

- `rds:RebootDBInstance`
- `rds:DescribeDBInstances`
- `tag:GetResources`

AWS マネージドポリシー

- AWSFaultInjectionSimulatorRDSAccess

Amazon S3 のアクション

AWS FIS は、次の Amazon S3 アクションをサポートします。

アクション

- aws:s3:bucket-pause-replication

aws:s3:bucket-pause-replication

ターゲットのソースバケットから宛先バケットへのレプリケーションを一時停止します。送信先バケットは、異なる AWS リージョンでも、ソースバケットと同じリージョン内でも配置することができます。既存のオブジェクトは、アクションが開始した後、最大 1 時間レプリケートが継続することがあります。このアクションはタグによるターゲティングのみをサポートします。Amazon S3 レプリケーションに関する詳細については、「Amazon S3 ユーザーガイド」を参照してください。

リソースタイプ

- aws:s3:bucket

パラメータ

- duration - 所要時間。1 分から 12 時間です。 AWS FIS API では、値は ISO 8601 形式の文字列です。例えば、PT1M は 1 分を表します。 AWS FIS コンソールで、秒数、分数、または時間数を入力します。
- region - 宛先バケットが配置されている AWS リージョン。
- destinationBuckets - オプション。宛先 S3 バケットのカンマ区切りリスト。
- prefixes - オプション。レプリケーションルールフィルターからの S3 オブジェクトキープレフィックスのカンマ区切りリスト。プレフィックスに基づくフィルターを使用したターゲットバケットのレプリケーションルールは一時停止されます。

アクセス許可

- 条件キーが `S3:IsReplicationPauseRequest` の `S3:PutReplicationConfiguration` を `True` に設定する
- 条件キーが `S3:IsReplicationPauseRequest` の `S3:GetReplicationConfiguration` を `True` に設定する
- `S3:PauseReplication`
- `S3:ListAllMyBuckets`
- `tag:GetResources`

ポリシーの例については、「例: `aws:s3:bucket-pause-replication` の条件キーを使用します。」を参照してください。

Systems Manager アクション

AWS FIS は、次の Systems Manager アクションをサポートします。

アクション

- aws:ssm:send-command
- aws:ssm:start-automation-execution

aws:ssm:send-command

ターゲット EC2 インスタンスSendCommandで Systems Manager API アクションを実行します。Systems Manager ドキュメント (SSM ドキュメント) は、Systems Manager がインスタンスで実行するアクションを定義します。詳細については、「aws:ssm:send-command アクションを使用します」を参照してください。

リソースタイプ

- aws:ec2:instance

パラメータ

- documentArn - ドキュメントの Amazon リソースネーム (ARN) 。コンソールでは、事前設定された AWS FIS SSM ドキュメント のいずれかに対応する値をアクションタイプ から選択すると、このパラメータが完了します。
- documentVersion - オプション。ドキュメントのバージョン。空の場合、デフォルトバージョンが実行されます。
- documentParameters - 条件付き。ドキュメントが受け入れる必須およびオプションのパラメータ。形式は JSON オブジェクトで、キーは文字列で、値は文字列または文字列の配列です。
- duration - 所要時間。1 分から 12 時間です。 AWS FIS API では、値は ISO 8601 形式の文字列です。例えば、PT1M は 1 分を表します。 AWS FIS コンソールで、秒数、分数、または時間数を入力します。

アクセス許可

- `ssm:SendCommand`
- `ssm:ListCommands`

- `ssm:CancelCommand`

AWS マネージドポリシー

- AWSFaultInjectionSimulatorEC2Access

aws:ssm:start-automation-execution

Systems Manager API アクションStartAutomationの実行 を実行します。

リソースタイプ

- なし

パラメータ

- documentArn - オートメーションドキュメントの Amazon リソースネーム (ARN)。
- documentVersion - オプション。ドキュメントのバージョン。空の場合、デフォルトバージョンが実行されます。
- documentParameters - 条件付き。ドキュメントが受け入れる必須およびオプションのパラメータ。形式は JSON オブジェクトで、キーは文字列、値は文字列または文字列の配列です。
- maxDuration - オートメーション実行が完了するまでの最大許容時間。1 分から 12 時間です。AWS FIS API では、値は ISO 8601 形式の文字列です。例えば、PT1M は 1 分を表します。 AWS FIS コンソールで、秒数、分数、または時間数を入力します。

アクセス許可

- `ssm:GetAutomationExecution`
- `ssm:StartAutomationExecution`
- `ssm:StopAutomationExecution`
- `iam:PassRole` - オプション。自動化ドキュメントが役割を引き受ける場合は必須です。

AWS マネージドポリシー

- AWSFaultInjectionSimulatorSSMAccess

AWS FIS で Systems Manager SSM ドキュメントを使用する

AWS FIS は、 AWS Systems Manager SSM エージェントおよび AWS FIS アクション を通じてカスタム障害タイプをサポートしますaws:ssm:send-command。一般的な障害挿入アクションの作成に使用できる事前設定済みの Systems Manager SSM ドキュメント (SSM ドキュメント) は、AWSFIS- プレフィックスで始まるパブリック AWS ドキュメントとして使用できます。

SSM Agentは、 Amazon EC2 インスタンス、オンプレミスサーバー、または仮想マシン (VM) にインストールして設定することができる Amazon のソフトウェアです。これにより、これらのリソースは Systems Manager で管理できるようになります。エージェントは Systems Manager からのリクエストを処理し、リクエストに指定されたとおりにそれらを実行します。独自の SSM ドキュメントを含めて、カスタム障害を注入したり、Amazon が所有するパブリックドキュメントの 1 つを参照したりできます。

要件

SSM Agent がターゲットに対して実行することになるアクションについては、次のことを確認する必要があります。

- エージェントがターゲットにインストールされていること。SSM Agent が、Amazon マシンイメージ (AMI) にデフォルトで事前インストールされていること。または、インスタンスに SSM Agent をインストールできます。詳細については、『AWS Systems Manager ユーザーガイド』の「Linux 用 EC2 インスタンスに SSM Agent を手動でインストールする」を参照してください。
- Systems Manager にはインスタンスでアクションを実行する権限がありません。IAM インスタンスプロファイルを使用してアクセスを許可する必要があります。詳細については、『AWS Systems Manager ユーザーガイド』の「Systems Manager の IAM インスタンスプロファイルを作成する」および「IAM インスタンスプロファイルを EC2 インスタンスにアタッチする」を参照してください。

aws:ssm:send-command アクションを使用します

SSM ドキュメントは、マネージドインスタンスで Systems Manager が実行するアクションを定義します。Systems Manager には、事前設定済みのドキュメントが多数含まれていますが、独自のドキュメントを作成することもできます。独自の SSM ドキュメント作成に関する詳細については、『AWS Systems Manager ユーザーガイド』の「Systems Manager のドキュメントを作成する」を参照してください。SSM ドキュメント全般の詳細については、『AWS Systems Manager ユーザーガイド』の「AWS Systems Manager ドキュメント」を参照してください。

AWS FIS は、事前設定された SSM ドキュメントを提供します。事前設定された SSM ドキュメントは、 AWS Systems Manager コンソールのドキュメント https://console.aws.amazon.com/systems-manager/documents で表示できます。 AWS FIS コンソールで、事前設定されたドキュメントの選択から選択することもできます。詳細については、「事前設定された AWS FIS SSM ドキュメント」を参照してください。

AWS FIS 実験で SSM ドキュメントを使用するには、 aws:ssm:send-commandアクションを使用できます。このアクションは、ターゲットインスタンスで指定された SSM ドキュメントをフェッチして実行します。

実験テンプレート内で `aws:ssm:send-command` アクションを使用する場合は、以下を含むアクションの追加パラメータを指定する必要があります。

- documentArn – 必須。SSM ドキュメントの Amazon リソースネーム (ARN) 。
- documentParameters - 条件付き。SSM ドキュメントに指定されている必須およびオプションのパラメータ。形式は JSON オブジェクトで、キーは文字列、値は文字列または文字列の配列です。
- documentVersion - オプション。実行する SSM ドキュメントのバージョン。

Systems Manager コンソールまたはコマンドラインを使用して、SSM ドキュメントの情報 (ドキュメントのパラメータを含む) を表示できます。

コンソールを使用して、SSM ドキュメントについての情報を表示するには

1. https://console.aws.amazon.com/systems-manager/ で AWS Systems Manager コンソールを開きます。
2. ナビゲーションペインで、[ドキュメント] を選択します。
3. ドキュメントを選択し、[Details (詳細)] タブをクリックします。

コマンドラインを使用して、SSM ドキュメントについての情報を表示するには

SSM describe-document コマンドを使用します。

事前設定された AWS FIS SSM ドキュメント

実験テンプレートの `aws:ssm:send-command`アクションで、事前設定された AWS FIS SSM ドキュメントを使用できます。

要件

- AWS FIS が提供する事前設定済みの SSM ドキュメントは、以下のオペレーティングシステムでのみサポートされています。
  - Amazon Linux 2023、Amazon Linux 2、Amazon Linux
  - Ubuntu
  - RHEL 7、8、9
  - CentOS 7、8、9
- AWS FIS が提供する事前設定済みの SSM ドキュメントは、EC2 インスタンスでのみサポートされています。オンプレミスサーバーなど、他のタイプのマネージドノードではサポートされていません。

ECS タスクの実験でこれらの SSM ドキュメントを使用するには、対応する the section called "Amazon ECS アクション" を使用してください。たとえば、aws:ecs:task-cpu-stress アクションは AWSFIS-Run-CPU-Stress ドキュメントを使用します。

ドキュメント

- AWSFIS-Run-CPU-Stress
- AWSFIS-Run-Disk-Fill
- AWSFIS-Run-IO-Stress
- AWSFIS-Run-Kill-Process
- AWSFIS-Run-Memory-Stress
- AWSFIS-Run-Network-Blackhole-Port
- AWSFIS-Run-Network-Latency
- AWSFIS-Run-Network-Latency-Sources
- AWSFIS-Run-Network-Packet-Loss
- AWSFIS-Run-Network-Packet-Loss-Sources

AWSFIS-Run-CPU-Stress

stress-ng ツールを使用して、インスタンスに対して CPU ストレスを実行します。AWSFIS-Run-CPU-Stress SSM ドキュメントを使用します。

アクションタイプ (コンソールのみ)

aws:ssm:send-command/AWSFIS-Run-CPU-Stress

ARN

arn:aws:ssm:region::document/AWSFIS-Run-CPU-Stress

ドキュメントパラメータ

- DurationSeconds – 必須。CPU ストレステストの継続時間 (秒単位)。
- CPU - オプション。使用する CPU ストレッサーの数。デフォルト値は 0 で、すべての CPU ストレッサーを使用します。
- LoadPercent - オプション。0 (無負荷) から 100 (全負荷) までのターゲット CPU 負荷率。デフォルトは 100 です。
- InstallDependencies - オプション。値が `True` の場合、ターゲットインスタンスに必要な依存関係が、Systems Manager によってンストールされます (まだインストールされていない場合)。デフォルトは `True` です。依存関係は stress-ng です。

以下はコンソールで入力できる文字列の例です。

```
{"DurationSeconds":"60", "InstallDependencies":"True"}
```

AWSFIS-Run-Disk-Fill

インスタンスのルートボリュームにディスク容量を割り当てて、ディスクフル障害をシミュレートします。AWSFIS-Run-Disk-Fill SSM ドキュメントを使用します。

この障害を挿入する実験が手動または停止条件によって停止された場合、 AWS FIS は実行中の SSM ドキュメントをキャンセルしてロールバックを試みます。ただし、障害または障害とアプリケーションのアクティビティが原因で、ディスクが 100% フルになると、Systems Manager でキャンセル操作を完了できないことがあります。そのため、実験を中止しなければならない場合は、ディスクを 100% フルにしないでください。

アクションタイプ (コンソールのみ)

aws:ssm:send-command/AWSFIS-Run-Disk-Fill

ARN

arn:aws:ssm:region::document/AWSFIS-Run-Disk-Fill

ドキュメントパラメータ

- DurationSeconds – 必須。ディスクフィルテストの継続時間 (秒単位)。
- Percent - オプション。ディスクフィルテスト中に割り当てるディスクの割合。デフォルトは 95% です。
- InstallDependencies - オプション。値が `True` の場合、ターゲットインスタンスに必要な依存関係が、Systems Manager によってンストールされます (まだインストールされていない場合)。デフォルトは `True` です。依存関係は、atd と fallocate です。

以下はコンソールに入力できる文字列の例です。

```
{"DurationSeconds":"60", "InstallDependencies":"True"}
```

AWSFIS-Run-IO-Stress

stress-ng ツールを使用して、インスタンスに対して IO ストレスを実行します。AWSFIS-Run-IO-Stress SSM ドキュメントを使用します。

アクションタイプ (コンソールのみ)

aws:ssm:send-command/AWSFIS-Run-IO-Stress

ARN

arn:aws:ssm:region::document/AWSFIS-Run-IO-Stress

ドキュメントパラメータ

- DurationSeconds – 必須。IO ストレステストの継続時間 (秒単位)。
- Workers - オプション。シーケンシャル、ランダム、およびメモリマップされた読み取り/書き込み操作、強制同期、およびキャッシュドロップの組み合わせを実行する Workers の数。複数の子プロセスが同じファイルに対して異なる I/O 操作を実行します。デフォルト は 1 です。
- Percent - オプション。I/O ストレステスト中に使用するファイルシステム上の空き領域の割合。デフォルトは 80% です。
- InstallDependencies - オプション。値が `True` の場合、ターゲットインスタンスに必要な依存関係が、Systems Manager によってンストールされます (まだインストールされていない場合)。デフォルトは `True` です。依存関係は stress-ng です。

以下はコンソールに入力できる文字列の例です。

```
{"Workers":"1", "Percent":"80", "DurationSeconds":"60", "InstallDependencies":"True"}
```

AWSFIS-Run-Kill-Process

killall コマンドを使用して、インスタンス内の指定されたプロセスを停止します。AWSFIS-Run-Kill-Process SSM ドキュメントを使用します。

アクションタイプ (コンソールのみ)

aws:ssm:send-command/AWSFIS-Run-Kill-Process

ARN

arn:aws:ssm:region::document/AWSFIS-Run-Kill-Process

ドキュメントパラメータ

- ProcessName – 必須。停止するプロセスの名前。
- Signal - オプション。コマンドと一緒に送信するシグナル。指定できる値は、`SIGTERM` (受信者が無視することを選択できる) と `SIGKILL` (無視できない) です。デフォルト: `SIGTERM`。
- InstallDependencies - オプション。値が `True` の場合、ターゲットインスタンスに必要な依存関係が、Systems Manager によってンストールされます (まだインストールされていない場合)。デフォルトは `True` です。依存関係は killall です。

以下はコンソールに入力できる文字列の例です。

```
{"ProcessName":"myapplication", "Signal":"SIGTERM"}
```

AWSFIS-Run-Memory-Stress

stress-ng ツールを使用して、インスタンスに対してメモリストレスを実行します。AWSFIS-Run-Memory-Stress SSM ドキュメントを使用します。

アクションタイプ (コンソールのみ)

aws:ssm:send-command/AWSFIS-Run-Memory-Stress

ARN

arn:aws:ssm:region::document/AWSFIS-Run-Memory-Stress

ドキュメントパラメータ

- DurationSeconds – 必須。メモリストレステストの継続時間 (秒単位)。
- Workers - オプション。仮想メモリストレッサーの数。デフォルト は 1 です。
- Percent – 必須。メモリストレステスト中に使用する仮想メモリの割合。
- InstallDependencies - オプション。値が `True` の場合、ターゲットインスタンスに必要な依存関係が、Systems Manager によってンストールされます (まだインストールされていない場合)。デフォルトは `True` です。依存関係は stress-ng です。

以下はコンソールに入力できる文字列の例です。

```
{"Percent":"80", "DurationSeconds":"60", "InstallDependencies":"True"}
```

AWSFIS-Run-Network-Blackhole-Port

iptablesツールを使用して、プロトコルおよびポートのインバウンドトラフィックまたはアウトバウンドトラフィックをドロップします。AWSFIS-Run-Network-Blackhole-Port SSM ドキュメントを使用します。

アクションタイプ (コンソールのみ)

aws:ssm:send-command/AWSFIS-Run-Network-Blackhole-Port

ARN

arn:aws:ssm:region::document/AWSFIS-Run-Network-Blackhole-Port

ドキュメントパラメータ

- Protocol – 必須。プロトコル。指定できる値は `tcp` および udp です。
- Port – 必須。ポート番号。
- TrafficType - オプション。トラフィックの種類。指定できる値は `ingress` および `egress` です。デフォルトは `ingress` です。
- DurationSeconds – 必須。ネットワークブラックホールテストの継続時間 (秒単位)。
- InstallDependencies - オプション。値が `True` の場合、ターゲットインスタンスに必要な依存関係が、Systems Manager によってンストールされます (まだインストールされていない場合)。デフォルトは `True` です。依存関係は、atd、dig、および iptables です。

以下はコンソールに入力できる文字列の例です。

```
{"Protocol":"tcp", "Port":"8080", "TrafficType":"egress", "DurationSeconds":"60",
 "InstallDependencies":"True"}
```

AWSFIS-Run-Network-Latency

tc ツールを使用して、ネットワークインターフェイスにレイテンシーを追加します。AWSFIS-Run-Network-Latency SSM ドキュメントを使用します。

アクションタイプ (コンソールのみ)

aws:ssm:send-command/AWSFIS-Run-Network-Latency

ARN

arn:aws:ssm:region::document/AWSFIS-Run-Network-Latency

ドキュメントパラメータ

- Interface - オプション。ネットワークインターフェース。デフォルト: `eth0`。
- DelayMilliseconds - オプション。遅延 (ミリ秒単位) 。デフォルトは 200 です。
- DurationSeconds – 必須。ネットワーク遅延テストの継続時間 (秒単位)。
- InstallDependencies - オプション。値が `True` の場合、ターゲットインスタンスに必要な依存関係が、Systems Manager によってンストールされます (まだインストールされていない場合)。デフォルトは `True` です。依存関係は、atd、dig、および tc です。

以下はコンソールに入力できる文字列の例です。

```
{"DelayMilliseconds":"200", "Interface":"eth0", "DurationSeconds":"60",
 "InstallDependencies":"True"}
```

AWSFIS-Run-Network-Latency-Sources

特定のソースへのトラフィックまたは特定のソースからのトラフィックのためのツール tc を使用してネットワークインターフェイスにレイテンシーとジッタを追加します。AWSFIS-Run-Network-Latency-Sources SSM ドキュメントを使用します。

アクションタイプ (コンソールのみ)

aws:ssm:send-command/AWSFIS-Run-Network-Latency-Sources

ARN

arn:aws:ssm:region::document/AWSFIS-Run-Network-Latency-Sources

ドキュメントパラメータ

- Interface - オプション。ネットワークインターフェース。デフォルト: `eth0`。
- DelayMilliseconds - オプション。遅延 (ミリ秒単位) 。デフォルトは 200 です。
- JitterMilliseconds - オプション。ジッタ (ミリ秒単位)。デフォルトは 10 です。
- Sources – 必須。ソースはカンマで区切ります。指定できる値は、IPv4 アドレス、IPv4 CIDR ブロック、ドメイン名、`DYNAMODB`、および `S3` です。`DYNAMODB` または `S3` を指定した場合、これは現在のリージョンのリージョナルエンドポイントにのみ適用されます。
- TrafficType - オプション。トラフィックの種類。指定できる値は `ingress` および `egress` です。デフォルトは `ingress` です。
- DurationSeconds – 必須。ネットワーク遅延テストの継続時間 (秒単位)。
- InstallDependencies - オプション。値が `True` の場合、ターゲットインスタンスに必要な依存関係が、Systems Manager によってンストールされます (まだインストールされていない場合)。デフォルトは `True` です。依存関係は、atd、dig、 jq および tc です。

以下はコンソールに入力できる文字列の例です。

```
{"DelayMilliseconds":"200", "JitterMilliseconds":"15",
 "Sources":"S3,www.example.com,72.21.198.67", "Interface":"eth0",
 "TrafficType":"egress", "DurationSeconds":"60", "InstallDependencies":"True"}
```

AWSFIS-Run-Network-Packet-Loss

tc ツールを使用してネットワークインターフェイスへのパッケージの損失を追加します。AWSFIS-Run-Network-Packet-Loss SSM ドキュメントを使用します。

アクションタイプ (コンソールのみ)

aws:ssm:send-command/AWSFIS-Run-Network-Packet-Loss

ARN

arn:aws:ssm:region::document/AWSFIS-Run-Network-Packet-Loss

ドキュメントパラメータ

- Interface - オプション。ネットワークインターフェース。デフォルト: `eth0`。
- LossPercent - オプション。パケット損失の割合。デフォルトは 7% です。
- DurationSeconds – 必須。ネットワークパケット損失テストの継続時間 (秒単位)。
- InstallDependencies - オプション。値が `True` の場合、ターゲットインスタンスに必要な依存関係が、Systems Manager によってンストールされます。デフォルトは `True` です。依存関係は、atd、dig、および tc です。

以下はコンソールに入力できる文字列の例です。

```
{"LossPercent":"15", "Interface":"eth0", "DurationSeconds":"60",
 "InstallDependencies":"True"}
```

AWSFIS-Run-Network-Packet-Loss-Sources

特定のソースへのトラフィックまたは特定のソースからのトラフィックのためのツール tc を使用してネットワークインターフェイスへのパッケージの損失を追加します。AWSFIS-Run-Network-Packet-Loss-Sources SSM ドキュメントを使用します。

アクションタイプ (コンソールのみ)

aws:ssm:send-command/AWSFIS-Run-Network-Packet-Loss-Sources

ARN

arn:aws:ssm:region::document/AWSFIS-Run-Network-Packet-Loss-Sources

ドキュメントパラメータ

- Interface - オプション。ネットワークインターフェース。デフォルト: `eth0`。
- LossPercent - オプション。パケット損失の割合。デフォルトは 7% です。
- Sources – 必須。ソースはカンマで区切ります。指定できる値は、IPv4 アドレス、IPv4 CIDR ブロック、ドメイン名、`DYNAMODB`、および `S3` です。`DYNAMODB` または `S3` を指定した場合、これは現在のリージョンのリージョナルエンドポイントにのみ適用されます。
- TrafficType - オプション。トラフィックの種類。指定できる値は `ingress` および `egress` です。デフォルトは `ingress` です。
- DurationSeconds – 必須。ネットワークパケット損失テストの継続時間 (秒単位)。

- InstallDependencies - オプション。値が `True` の場合、ターゲットインスタンスに必要な依存関係が、Systems Manager によってンストールされます。デフォルトは `True` です。依存関係は、atd、dig、 jq および tc です。

以下はコンソールに入力できる文字列の例です。

```
{"LossPercent":"15", "Sources":"S3,www.example.com,72.21.198.67", "Interface":"eth0",
 "TrafficType":"egress", "DurationSeconds":"60", "InstallDependencies":"True"}
```

例

実験テンプレートの例については、「the section called “事前設定された AWS FIS SSM ドキュメントを実行する”」を参照してください。

チュートリアルの例については、「インスタンス上で CPU ストレスを実行する」を参照してください。

トラブルシューティング

次の手順を使用して、問題のトラブルシューティングを行います。

SSM ドキュメントを使用して問題のトラブルシューティングを行います

1. https://console.aws.amazon.com/systems-manager/ で AWS Systems Manager コンソールを開きます。
2. ナビゲーションペインの [Node Management (ノード管理)] で、[Run Command (コマンドを実行)] を選択します。
3. [コマンド履歴] タブでは、フィルターを使用してドキュメントの実行場所を特定します。
4. コマンドの ID を選択して、詳細ページを開きます。
5. インスタンスの ID を選択します。各ステップの出力とエラーを確認します。

AWS FIS aws:ecs:task アクションを使用する

aws:ecs:task アクションを使用して Amazon ECS タスクに障害を発生させることができます。

これらのアクションでは、SSM エージェントをサイドカーコンテナとして使用して、障害挿入を実行する SSM ドキュメントを実行し、サイドカーコンテナを介して Amazon ECS タスクを SSM マ

ネージドインスタンスとして登録します。これらのアクションを使用するには、Amazon ECS タスク定義を更新して SSM エージェントをサイドカーコンテナとして追加する必要があります。これにより、実行中のタスクを SSM マネージドインスタンスとして登録することができます。をターゲットとする AWS FIS 実験を実行すると`aws:ecs:task`、 AWS FIS は、 AWS FIS 実験テンプレートで指定したターゲット Amazon ECS タスクを、マネージドインスタンスに追加されたリソースタグを使用して `ECS_TASK_ARN`SSM マネージドインスタンスのセットにマッピングします。タグ値は SSM ドキュメントを実行する必要があり、関連する Amazon ECS タスクの ARN であるため、実験を実行するときに削除されないようにする必要があります。

アクション

- the section called “aws:ecs:task-cpu-stress”
- the section called “aws:ecs:task-io-stress”
- the section called “aws:ecs:task-kill-process”
- the section called “aws:ecs:task-network-blackhole-port”
- the section called “aws:ecs:task-network-latency”
- the section called “aws:ecs:task-network-packet-loss”

制限事項

- 以下のアクションは では機能しません AWS Fargate。
  - aws:ecs:task-kill-process
  - aws:ecs:task-network-blackhole-port
  - aws:ecs:task-network-latency
  - aws:ecs:task-network-packet-loss
- ECS Exec を有効にしている場合、これらのアクションを使用する前に無効にしておく必要があります。

要件

- AWS FIS 実験ロール に次のアクセス許可を追加します。
  - `ssm:SendCommand`
  - `ssm:ListCommands`
  - `ssm:CancelCommand`

- Amazon ECS のタスク IAM ロールに次の権限を追加します。
  - `ssm:CreateActivation`
  - `ssm:AddTagsToResource`
  - `iam:PassRole`

  マネージドインスタンスロールの ARN を `iam:PassRole` のリソースとして指定できることに注意してください。
- Amazon ECS タスク実行 IAM ロールを作成し、AmazonECSTaskExecutionRolePolicy 管理ポリシーを追加します。
- マネージドインスタンスとして登録されたタスクにアタッチされたマネージドインスタンスロールに次のアクセス権限を追加します。
  - `ssm:DeleteActivation`
  - `ssm:DeregisterManagedInstance`
- マネージドインスタンスとして登録されたタスクにアタッチされたマネージドインスタンスロールに AmazonSSMManagedInstance Core マネージドポリシーを追加します。
- `MANAGED_INSTANCE_ROLE_NAME` 環境変数をマネージドインスタンスロールの名前に設定します。
- SSM エージェントコンテナを ECS タスク定義に追加します。コマンドスクリプトは ECS タスクを管理対象インスタンスとして登録します。

```
{
    "name": "amazon-ssm-agent",
    "image": "public.ecr.aws/amazon-ssm-agent/amazon-ssm-agent:latest",
    "cpu": 0,
    "links": [],
    "portMappings": [],
    "essential": false,
    "entryPoint": [],
    "command": [
        "/bin/bash",
        "-c",
        "set -e; yum upgrade -y; yum install jq procps awscli -y; term_handler()
 { echo \"Deleting SSM activation $ACTIVATION_ID\"; if ! aws ssm delete-
activation --activation-id $ACTIVATION_ID --region $ECS_TASK_REGION; then
 echo \"SSM activation $ACTIVATION_ID failed to be deleted\" 1>&2; fi;
 MANAGED_INSTANCE_ID=$(jq -e -r .ManagedInstanceID /var/lib/amazon/ssm/registration);
 echo \"Deregistering SSM Managed Instance $MANAGED_INSTANCE_ID\"; if ! aws
 ssm deregister-managed-instance --instance-id $MANAGED_INSTANCE_ID --region
```

```
 $ECS_TASK_REGION; then echo \"SSM Managed Instance $MANAGED_INSTANCE_ID
 failed to be deregistered\" 1>&2; fi; kill -SIGTERM $SSM_AGENT_PID; }; trap
 term_handler SIGTERM SIGINT; if [[ -z $MANAGED_INSTANCE_ROLE_NAME ]]; then
 echo \"Environment variable MANAGED_INSTANCE_ROLE_NAME not set, exiting\"
 1>&2; exit 1; fi; if ! ps ax | grep amazon-ssm-agent | grep -v grep > /dev/
null; then if [[ -n $ECS_CONTAINER_METADATA_URI_V4 ]] ; then echo \"Found ECS
 Container Metadata, running activation with metadata\"; TASK_METADATA=$(curl
 \"${ECS_CONTAINER_METADATA_URI_V4}/task\"); ECS_TASK_AVAILABILITY_ZONE=$(echo
 $TASK_METADATA | jq -e -r '.AvailabilityZone'); ECS_TASK_ARN=$(echo $TASK_METADATA
 | jq -e -r '.TaskARN'); ECS_TASK_REGION=$(echo $ECS_TASK_AVAILABILITY_ZONE | sed
 's/.$//'); ECS_TASK_AVAILABILITY_ZONE_REGEX='^(af|ap|ca|cn|eu|me|sa|us|us-gov)-
(central|north|(north(east|west))|south|south(east|west)|east|west)-[0-9]{1}[a-z]
{1}$'; if ! [[ $ECS_TASK_AVAILABILITY_ZONE =~ $ECS_TASK_AVAILABILITY_ZONE_REGEX ]];
 then echo \"Error extracting Availability Zone from ECS Container Metadata,
 exiting\" 1>&2; exit 1; fi; ECS_TASK_ARN_REGEX='^arn:(aws|aws-cn|aws-us-gov):ecs:
[a-z0-9-]+:[0-9]{12}:task/[a-zA-Z0-9_-]+/[a-zA-Z0-9]+$'; if ! [[ $ECS_TASK_ARN
 =~ $ECS_TASK_ARN_REGEX ]]; then echo \"Error extracting Task ARN from ECS
 Container Metadata, exiting\" 1>&2; exit 1; fi; CREATE_ACTIVATION_OUTPUT=
$(aws ssm create-activation --iam-role $MANAGED_INSTANCE_ROLE_NAME --
tags Key=ECS_TASK_AVAILABILITY_ZONE,Value=$ECS_TASK_AVAILABILITY_ZONE
 Key=ECS_TASK_ARN,Value=$ECS_TASK_ARN Key=FAULT_INJECTION_SIDECAR,Value=true --
region $ECS_TASK_REGION); ACTIVATION_CODE=$(echo $CREATE_ACTIVATION_OUTPUT | jq
 -e -r .ActivationCode); ACTIVATION_ID=$(echo $CREATE_ACTIVATION_OUTPUT | jq -e
 -r .ActivationId); if ! amazon-ssm-agent -register -code $ACTIVATION_CODE -id
 $ACTIVATION_ID -region $ECS_TASK_REGION; then echo \"Failed to register with AWS
 Systems Manager (SSM), exiting\" 1>&2; exit 1; fi; amazon-ssm-agent & SSM_AGENT_PID=
$!; wait $SSM_AGENT_PID; else echo \"ECS Container Metadata not found, exiting\"
 1>&2; exit 1; fi; else echo \"SSM agent is already running, exiting\" 1>&2; exit 1;
 fi"
    ],
    "environment": [
        {
            "name": "MANAGED_INSTANCE_ROLE_NAME",
            "value": "SSMManagedInstanceRole"
        }
    ],
    "environmentFiles": [],
    "mountPoints": [],
    "volumesFrom": [],
    "secrets": [],
    "dnsServers": [],
    "dnsSearchDomains": [],
    "extraHosts": [],
    "dockerSecurityOptions": [],
```

```
    "dockerLabels": {},
    "ulimits": [],
    "logConfiguration": {},
    "systemControls": []
}
```

スクリプトのより読みやすいバージョンについては、the section called “スクリプトのリファレンスバージョン。” を参照してください。

- `aws:ecs:task-network-blackhole-port`、`aws:ecs:task-network-latency`、および `aws:ecs:task-network-packet-loss` アクションを使用するときは、次のいずれかのオプションを使用して ECS タスク定義の SSM Agent コンテナを更新する必要があります。
  - オプション 1 - 特定の Linux 機能を追加します。

    ```
    "linuxParameters": {
        "capabilities": {
            "add": [
                "NET_ADMIN"
            ]
        }
    },
    ```

  - オプション 2 - すべての Linux 機能を追加します。

    ```
    "privileged": true,
    ```

- `aws:ecs:task-kill-process`、`aws:ecs:task-network-blackhole-port`、`aws:ecs:task-network-latency`、および `aws:ecs:task-network-packet-loss` アクションを使用する場合、ECS タスク定義では `pidMode` を `task` に設定しておく必要があります。

スクリプトのリファレンスバージョン。

参考までに、「要件」セクションにあるスクリプトをより読みやすくしたものを次に示します。

```
#!/usr/bin/env bash

# This is the activation script used to register ECS tasks as Managed Instances in SSM
# The script retrieves information form the ECS task metadata endpoint to add three
 tags to the Managed Instance
```

```
#  - ECS_TASK_AVAILABILITY_ZONE: To allow customers to target Managed Instances / Tasks
 in a specific Availability Zone
#  - ECS_TASK_ARN: To allow customers to target Managed Instances / Tasks by using the
 Task ARN
#  - FAULT_INJECTION_SIDECAR: To make it clear that the tasks were registered as
 managed instance for fault injection purposes. Value is always 'true'.
# The script will leave the SSM Agent running in the background
# When the container running this script receives a SIGTERM or SIGINT signal, it will
 do the following cleanup:
#  - Delete SSM activation
#  - Deregister SSM managed instance

set -e # stop execution instantly as a query exits while having a non-zero

yum upgrade -y
yum install jq procps awscli -y

term_handler() {
  echo "Deleting SSM activation $ACTIVATION_ID"
  if ! aws ssm delete-activation --activation-id $ACTIVATION_ID --region
 $ECS_TASK_REGION; then
    echo "SSM activation $ACTIVATION_ID failed to be deleted" 1>&2
  fi

  MANAGED_INSTANCE_ID=$(jq -e -r .ManagedInstanceID /var/lib/amazon/ssm/registration)
  echo "Deregistering SSM Managed Instance $MANAGED_INSTANCE_ID"
  if ! aws ssm deregister-managed-instance --instance-id $MANAGED_INSTANCE_ID --region
 $ECS_TASK_REGION; then
    echo "SSM Managed Instance $MANAGED_INSTANCE_ID failed to be deregistered" 1>&2
  fi

  kill -SIGTERM $SSM_AGENT_PID
}
trap term_handler SIGTERM SIGINT

# check if the required IAM role is provided
if [[ -z $MANAGED_INSTANCE_ROLE_NAME ]] ; then
  echo "Environment variable MANAGED_INSTANCE_ROLE_NAME not set, exiting" 1>&2
  exit 1
fi

# check if the agent is already running (it will be if ECS Exec is enabled)
if ! ps ax | grep amazon-ssm-agent | grep -v grep > /dev/null; then
```

```
  # check if ECS Container Metadata is available
  if [[ -n $ECS_CONTAINER_METADATA_URI_V4 ]] ; then

    # Retrieve info from ECS task metadata endpoint
    echo "Found ECS Container Metadata, running activation with metadata"
    TASK_METADATA=$(curl "${ECS_CONTAINER_METADATA_URI_V4}/task")
    ECS_TASK_AVAILABILITY_ZONE=$(echo $TASK_METADATA | jq -e -r '.AvailabilityZone')
    ECS_TASK_ARN=$(echo $TASK_METADATA | jq -e -r '.TaskARN')
    ECS_TASK_REGION=$(echo $ECS_TASK_AVAILABILITY_ZONE | sed 's/.$//')

    # validate ECS_TASK_AVAILABILITY_ZONE
    ECS_TASK_AVAILABILITY_ZONE_REGEX='^(af|ap|ca|cn|eu|me|sa|us|us-gov)-(central|north|
(north(east|west))|south|south(east|west)|east|west)-[0-9]{1}[a-z]{1}$'
    if ! [[ $ECS_TASK_AVAILABILITY_ZONE =~ $ECS_TASK_AVAILABILITY_ZONE_REGEX ]] ; then
      echo "Error extracting Availability Zone from ECS Container Metadata, exiting"
 1>&2
      exit 1
    fi

    # validate ECS_TASK_ARN
    ECS_TASK_ARN_REGEX='^arn:(aws|aws-cn|aws-us-gov):ecs:[a-z0-9-]+:[0-9]{12}:task/[a-
zA-Z0-9_-]+/[a-zA-Z0-9]+$'
    if ! [[ $ECS_TASK_ARN =~ $ECS_TASK_ARN_REGEX ]] ; then
      echo "Error extracting Task ARN from ECS Container Metadata, exiting" 1>&2
      exit 1
    fi

    # Create activation tagging with Availability Zone and Task ARN
    CREATE_ACTIVATION_OUTPUT=$(aws ssm create-activation \
      --iam-role $MANAGED_INSTANCE_ROLE_NAME \
      --tags Key=ECS_TASK_AVAILABILITY_ZONE,Value=$ECS_TASK_AVAILABILITY_ZONE
 Key=ECS_TASK_ARN,Value=$ECS_TASK_ARN Key=FAULT_INJECTION_SIDECAR,Value=true \
      --region $ECS_TASK_REGION)

    ACTIVATION_CODE=$(echo $CREATE_ACTIVATION_OUTPUT | jq -e -r .ActivationCode)
    ACTIVATION_ID=$(echo $CREATE_ACTIVATION_OUTPUT | jq -e -r .ActivationId)

    # Register with AWS Systems Manager (SSM)
    if ! amazon-ssm-agent -register -code $ACTIVATION_CODE -id $ACTIVATION_ID -region
 $ECS_TASK_REGION; then
      echo "Failed to register with AWS Systems Manager (SSM), exiting" 1>&2
      exit 1
    fi
```

```
    # the agent needs to run in the background, otherwise the trapped signal
    # won't execute the attached function until this process finishes
    amazon-ssm-agent &
    SSM_AGENT_PID=$!

    # need to keep the script alive, otherwise the container will terminate
    wait $SSM_AGENT_PID

  else
    echo "ECS Container Metadata not found, exiting" 1>&2
    exit 1
  fi

else
  echo "SSM agent is already running, exiting" 1>&2
  exit 1
fi
```

実験テンプレートの例

以下は、the section called “aws:ecs:task-cpu-stress” アクションに対する実験テンプレートの例です。

```
{
    "description": "Run CPU stress on the target ECS tasks",
    "targets": {
        "myTasks": {
            "resourceType": "aws:ecs:task",
            "resourceArns": [
                "arn:aws:ecs:us-east-1:111122223333:task/my-
cluster/09821742c0e24250b187dfed8EXAMPLE"
            ],
            "selectionMode": "ALL"
        }
    },
    "actions": {
        "EcsTask-cpu-stress": {
            "actionId": "aws:ecs:task-cpu-stress",
            "parameters": {
                "duration": "PT1M"
            },
            "targets": {
                "Tasks": "myTasks"
```

```
            }
        }
    },
    "stopConditions": [
        {
            "source": "none",
        }
    ],
    "roleArn": "arn:aws:iam::111122223333:role/fis-experiment-role",
    "tags": {}
}
```

AWS FIS aws:eks:pod アクションを使用する

aws:eks:pod アクションを使用して、EKS クラスターで実行されている Kubernetes ポッドに障害を注入できます。

アクション

- the section called “aws:eks:pod-cpu-stress”
- the section called “aws:eks:pod-delete”
- the section called “aws:eks:pod-io-stress”
- the section called “aws:eks:pod-memory-stress”
- the section called “aws:eks:pod-network-blackhole-port”
- the section called “aws:eks:pod-network-latency”
- the section called “aws:eks:pod-network-packet-loss”

制限事項

- 以下のアクションは では機能しません AWS Fargate。
  - aws:eks:pod-network-blackhole-port
  - aws:eks:pod-network-latency
  - aws:eks:pod-network-packet-loss
- 次のアクションは `bridge` ネットワークモードをサポートしていません。
  - aws:eks:pod-network-blackhole-port

  - aws:eks:pod-network-latency
  - aws:eks:pod-network-packet-loss
- リソース ARN またはリソースタグを使用して、実験テンプレート内の aws: eks:pod タイプのターゲットを識別することはできません。必要なリソースパラメータを使用してターゲットを特定する必要があります。
- アクション `aws:eks:pod-network-latency` と `aws:eks:pod-network-packet-loss` は並列で実行たり、同じポッドをターゲットにしたりしません。指定した `maxErrors` パラメータの値に応じて、アクションは完了または失敗の状態で終了することがあります。
  - `maxErrorsPercent` が 0 (デフォルト) の場合、アクションは失敗の状態で終了します。
  - それ以外の場合は、失敗が最大 `maxErrorsPercent` の予算まで追加されます。失敗した挿入の数が指定した `maxErrors` に達しない場合、アクションは完了の状態で終了します。
  - これらの障害は、ターゲットポッドに挿入されたエフェメラルコンテナのログから特定できます。`Exit Code: 16` で失敗します。
- アクション `aws:eks:pod-network-blackhole-port` は、同じポッドをターゲットにし、同じ `trafficType` を使用する他のアクションを使用して並列で実行することはしません。異なるトラフィックタイプを使用する並列アクションはサポートされています。
- FIS は、ターゲットポッドの が に設定されている場合にのみ、障害挿入のステータスsecurityContextをモニタリングできますreadOnlyRootFilesystem: false。この設定がない場合、EKS ポッドのアクションはすべて失敗します。

要件

- をコンピュータ AWS CLI にインストールします。これが必要なのは、 AWS CLI を使用して IAM ロールを作成する場合にのみ必要です。詳細については、「AWS CLIのインストールまたは更新」を参照してください。
- コンピュータに kubectl をインストールします。これが必要なのは、EKS クラスターを操作してターゲットアプリケーションを設定するか、監視する場合に限られます。詳細については、https://kubernetes.io/docs/tasks/tools/ を参照してください。
- 現在サポートされている最小バージョンは 1.23 です。

Kubernetes サービスアカウントのサービスロールを作成します。

IAM ロールを作成して、サービスロールとして使用します。詳細については、「the section called “実験ロール”」を参照してください。

Kubernetes サービスアカウントを設定する

指定した Kubernetes 名前空間のターゲットで実験を実行するように Kubernetes サービスアカウントを設定します。次の例では、サービスアカウントは *myserviceaccount* で、名前空間は*#### #*です。default は標準の Kubernetes 名前空間の1つであることに注意してください。

Kubernetes サービスアカウントを設定する

1. `rbac.yaml` という名前のファイルを作成して以下を追加します。

```
kind: ServiceAccount
apiVersion: v1
metadata:
  namespace: default
  name: myserviceaccount

---
kind: Role
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  namespace: default
  name: role-experiments
rules:
- apiGroups: [""]
  resources: ["configmaps"]
  verbs: [ "get", "create", "patch", "delete"]
- apiGroups: [""]
  resources: ["pods"]
  verbs: ["create", "list", "get", "delete", "deletecollection"]
- apiGroups: [""]
  resources: ["pods/ephemeralcontainers"]
  verbs: ["update"]
- apiGroups: [""]
  resources: ["pods/exec"]
  verbs: ["create"]
- apiGroups: ["apps"]
  resources: ["deployments"]
  verbs: ["get"]

---
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
```

```
  name: bind-role-experiments
  namespace: default
subjects:
- kind: ServiceAccount
  name: myserviceaccount
  namespace: default
- apiGroup: rbac.authorization.k8s.io
  kind: User
  name: fis-experiment
roleRef:
  kind: Role
  name: role-experiments
  apiGroup: rbac.authorization.k8s.io
```

2. 以下のコマンドを実行します。

```
kubectl apply -f rbac.yaml
```

実験ロールを Kubernetes ユーザーにマッピングします。

以下のコマンドを使用してアイデンティティマッピングを作成します。詳細については、eksctlドキュメントの「IAM ユーザーとロールの管理」を参照してください。

```
eksctl create iamidentitymapping \
    --arn arn:aws:iam::123456789012:role/fis-experiment-role \
    --username fis-experiment \
    --cluster my-cluster
```

ポッドコンテナイメージ

AWS FIS が提供するポッドコンテナイメージは Amazon ECR でホストされます。Amazon ECR からイメージを参照する場合は、イメージに完全なイメージの URI を使用する必要があります

AWS リージョン	イメージ URI
米国東部 (オハイオ)	`051821878176.dkr.ecr.us-east-2.amazonaws.com/aws-fis-pod:0.1`

AWS リージョン	イメージ URI
米国東部 (バージニア北部)	731367659002.dkr.ecr.us-east-1.amazonaws.com/aws-fis-pod:0.1
米国西部 (北カリフォルニア)	080694859247.dkr.ecr.us-west-1.amazonaws.com/aws-fis-pod:0.1
米国西部 (オレゴン)	864386544765.dkr.ecr.us-west-2.amazonaws.com/aws-fis-pod:0.1
アフリカ (ケープタウン)	056821267933.dkr.ecr.af-south-1.amazonaws.com/aws-fis-pod:0.1
アジアパシフィック (香港)	246405402639.dkr.ecr.ap-east-1.amazonaws.com/aws-fis-pod:0.1
アジアパシフィック (ムンバイ)	524781661239.dkr.ecr.ap-south-1.amazonaws.com/aws-fis-pod:0.1
アジアパシフィック (ソウル)	526524659354.dkr.ecr.ap-northeast-2.amazonaws.com/aws-fis-pod:0.1
アジアパシフィック (シンガポール)	316401638346.dkr.ecr.ap-southeast-1.amazonaws.com/aws-fis-pod:0.1
アジアパシフィック (シドニー)	488104106298.dkr.ecr.ap-southeast-2.amazonaws.com/aws-fis-pod:0.1
アジアパシフィック (東京)	635234321696.dkr.ecr.ap-northeast-1.amazonaws.com/aws-fis-pod:0.1
カナダ (中部)	490658072207.dkr.ecr.ca-central-1.amazonaws.com/aws-fis-pod:0.1
欧州 (フランクフルト)	713827034473.dkr.ecr.eu-central-1.amazonaws.com/aws-fis-pod:0.1
欧州 (アイルランド)	205866052826.dkr.ecr.eu-west-1.amazonaws.com/aws-fis-pod:0.1

AWS リージョン	イメージ URI
欧州 (ロンドン)	327424803546.dkr.ecr.eu-west-2.amazonaws.com/aws-fis-pod:0.1
欧州 (ミラノ)	478809367036.dkr.ecr.eu-south-1.amazonaws.com/aws-fis-pod:0.1
欧州 (パリ)	154605889247.dkr.ecr.eu-west-3.amazonaws.com/aws-fis-pod:0.1
欧州 (ストックホルム)	263175118295.dkr.ecr.eu-north-1.amazonaws.com/aws-fis-pod:0.1
中東 (バーレーン)	065825543785.dkr.ecr.me-south-1.amazonaws.com/aws-fis-pod:0.1
南米 (サンパウロ)	767113787785.dkr.ecr.sa-east-1.amazonaws.com/aws-fis-pod:0.1
AWS GovCloud (米国東部)	246533647532.dkr.ecr.us-gov-east-1.amazonaws.com/aws-fis-pod:0.1
AWS GovCloud (米国西部)	246529956514.dkr.ecr.us-gov-west-1.amazonaws.com/aws-fis-pod:0.1

実験テンプレートの例

以下は、the section called “aws:eks:pod-network-latency” アクションに対する実験テンプレートの例です。

```
{
    "description": "Add latency and jitter to the network interface for the target EKS
 pods",
    "targets": {
        "myPods": {
            "resourceType": "aws:eks:pod",
            "parameters": {
                "clusterIdentifier": "mycluster",
```

```
                "namespace": "default",
                "selectorType": "labelSelector",
                "selectorValue": "mylabel=mytarget"
            },
            "selectionMode": "COUNT(3)"
        }
    },
    "actions": {
        "EksPod-latency": {
            "actionId": "aws:eks:pod-network-latency",
            "description": "Add latency",
            "parameters": {
                "kubernetesServiceAccount": "myserviceaccount",
                "duration": "PT5M",
                "delayMilliseconds": "200",
                "jitterMilliseconds": "10",
                "sources": "0.0.0.0/0"
            },
            "targets": {
                "Pods": "myPods"
            }
        }
    },
    "stopConditions": [
        {
            "source": "none",
        }
    ],
    "roleArn": "arn:aws:iam::111122223333:role/fis-experiment-role",
    "tags": {
        "Name": "EksPodNetworkLatency"
    }
}
```

を使用して AWS FIS アクションを一覧表示する AWS CLI

AWS Command Line Interface（AWS CLI) を使用して、 が AWS FIS サポートするアクションに関する情報を表示できます。

前提条件

をコンピュータ AWS CLI にインストールします。使用を開始する方法については、『AWS Command Line Interface ユーザーガイド』を参照してください。のコマンドの詳細については AWS FIS、「 コマンドリファレンス」の「fis」を参照してください。 AWS CLI

例:すべてのアクションの名前を一覧表示する

次のように list-actions コマンドを使用すると、すべてのアクションの名前を一覧表示できます。

```
aws fis list-actions --query "actions[*].[id]" --output text | sort
```

以下は出力例です。

```
aws:cloudwatch:assert-alarm-state
aws:dynamodb:global-table-pause-replication
aws:ebs:pause-volume-io
aws:ec2:api-insufficient-instance-capacity-error
aws:ec2:asg-insufficient-instance-capacity-error
aws:ec2:reboot-instances
aws:ec2:send-spot-instance-interruptions
aws:ec2:stop-instances
aws:ec2:terminate-instances
aws:ecs:drain-container-instances
aws:ecs:stop-task
aws:eks:inject-kubernetes-custom-resource
aws:eks:terminate-nodegroup-instances
aws:elasticache:interrupt-cluster-az-power
aws:fis:inject-api-internal-error
aws:fis:inject-api-throttle-error
aws:fis:inject-api-unavailable-error
aws:fis:wait
aws:network:disrupt-connectivity
aws:network:route-table-disrupt-cross-region-connectivity
aws:network:transit-gateway-disrupt-cross-region-connectivity
aws:rds:failover-db-cluster
aws:rds:reboot-db-instances
aws:s3:bucket-pause-replication
aws:ssm:send-command
aws:ssm:start-automation-execution
```

例：アクションに関する情報の表示

アクションの名前がわかったら、次のように get-action コマンドを使用してアクションに関する詳細情報を表示できます。

```
aws fis get-action --id aws:ec2:reboot-instances
```

以下は出力例です。

```
{
    "action": {
        "id": "aws:ec2:reboot-instances",
        "description": "Reboot the specified EC2 instances.",
        "targets": {
            "Instances": {
                "resourceType": "aws:ec2:instance"
            }
        },
        "tags": {}
    }
}
```

AWS FIS用の実験テンプレート

実験テンプレートには、指定したターゲットに対して実験中に実行する 1 つ以上のアクションが含まれています。また、実験が範囲外になるのを防ぐ停止条件も含まれています。作成した実験テンプレートは、実験を実行するのに使用できます。

テンプレートコンポーネント

実験テンプレートを構築するには、次のコンポーネントを使用します。

アクションセット

実行する AWS FIS アクション。アクションは、指定した設定順序で実行することも、同時に実行することもできます。詳細については、「アクションセット」を参照してください。

ターゲット

AWS 特定のアクションが実行されるリソース。詳細については、「ターゲット」を参照してください。

停止条件

CloudWatch アプリケーションのパフォーマンスが許容範囲外となる閾値を定義するアラーム。実験の実行中に停止条件がトリガーされると、 AWS FIS は実験を停止します。詳細については、「停止条件」を参照してください。

実験ロール

ユーザーに代わって実験を実行できるようにするために必要な権限を AWS FIS に付与する IAM ロール。詳細については、「実験ロール」を参照してください。

実験オプション

実験テンプレートのオプション。詳細については、「実験オプション」を参照してください。

アカウントには FIS に関連するクォータがあります。 AWS たとえば、実験テンプレートごとに実行できるアクションの数にはクォータがあります。詳細については、「クォータと制限事項」を参照してください。

テンプレート構文

実験テンプレートの構文は次のとおりです。

```
{
            "description": "string",
            "targets": {},
            "actions": {},
            "stopConditions": [],
            "roleArn": "arn:aws:iam::123456789012:role/AllowFISActions",
            "experimentOptions":{},
            "tags": {}
        }
```

例については、「テンプレートの例」を参照してください。

使用を開始する

を使用して実験テンプレートを作成するには、を参照してください。 AWS Management Console実験テンプレートの作成

を使用して実験テンプレートを作成するには AWS CLI、を参照してくださいAWS FIS 実験テンプレートの例。

AWS FIS のアクションセット

実験テンプレートを作成するには、アクションセットを構成する 1 つ以上のアクションを定義する必要があります。 AWS FIS が提供する定義済みアクションのリストについては、を参照してください。アクション

アクションは、実験中に 1 回だけ実行できます。同じ実験で同じ AWS FIS アクションを複数回実行するには、異なる名前を使用してそのアクションをテンプレートに複数回追加します。

コンテンツ

アクションの構文

以下にアクションセット用の構文を示します。

```
{
    "actions": {
        "action_name": {
            "actionId": "aws:service:action-type",
            "description": "string",
            "parameters": {
                "name": "value"
             },
            "startAfter": ["action_name", ...],
            "targets": {
                "resource_type": "target_name"
            }
        }
    }
}
```

アクションを定義する場合は、以下を指定します。

action_name

アクションの名前。

actionId

アクション識別子。

description

オプションの説明。

parameters

任意のアクションパラメータ。

startAfter

このアクションを開始する前に完了する必要があるアクション。それ以外の場合、アクションは実験の開始時に実行されます。

targets

すべてのアクションターゲット。

例については、「the section called “アクションの例”」を参照してください。

アクション期間

アクションにアクションの継続時間を指定するために使用できるパラメータが含まれている場合、デフォルトでは、このアクションは指定した期間が経過した後にのみ完了とみなされます。`emptyTargetResolutionMode` 実験オプションを `skip` に設定した場合、ターゲットが解決されなかったとき、アクションは「スキップ」ステータスですぐに完了します。たとえば、5 分の時間を指定した場合、 AWS FIS は 5 分後にアクションが完了したと見なします。その後、すべてのアクションが完了するまで、次のアクションが開始されます。

期間は、アクション条件が維持される時間、またはメトリクスが監視される時間の長さのいずれかです。例えば、指定した時間の間、レイテンシーが注入されます。インスタンスの終了など、ほぼ瞬時のアクションタイプの場合、停止条件は指定された期間監視されます。

アクションがアクションパラメータ内にポストアクションを含んでいる場合、そのアクションが完了した後にポストアクションが実行されます。ポストアクションを完了するのにかかる時間によって、指定されたアクション期間から次のアクションの開始までの間に遅延が発生することがあります（または、他のすべてのアクションが完了した場合は実験を終了します）。

アクションの例

アクションの例は次のとおりです。

例

- EC2 インスタンスの停止
- スポットインスタンスの中断
- ネットワークトラフィックの中断
- EKS ワーカーの終了

例: EC2インスタンスを停止する

次のアクションは、*targetInstances* という名前のターゲットで識別された EC2 インスタンスを停止します。2 分後にターゲットのインスタンスが再起動されます。

```
"actions": {
    "stopInstances": {
        "actionId": "aws:ec2:stop-instances",
        "parameters": {
            "startInstancesAfterDuration": "PT2M"
```

```
        },
        "targets": {
            "Instances": "targetInstances"
        }
    }
}
```

例: スポットインスタンスの中断

次のアクションは、という名前のターゲットを使用して識別されたスポットインスタンスを停止します。*targetSpotInstances*2 分間待ってから、スポットインスタンスが中断されます。

```
"actions": {
    "interruptSpotInstances": {
        "actionId": "aws:ec2:send-spot-instance-interruptions",
        "parameters": {
            "durationBeforeInterruption": "PT2M"
        },
        "targets": {
            "SpotInstances": "targetSpotInstances"
        }
    }
}
```

例: ネットワークトラフィックの中断

次のアクションは、ターゲットサブネットと他のアベイラビリティーゾーン内のサブネット間のトラフィックを拒否します。

```
"actions": {
    "disruptAZConnectivity": {
        "actionId": "aws:network:disrupt-connectivity",
        "parameters": {
            "scope": "availability-zone",
            "duration": "PT5M"
        },
        "targets": {
            "Subnets": "targetSubnets"
        }
    }
}
```

例: EKS ワーカーの終了

次のアクションは、*targetNodeGroups*指定されたターゲットを使用して識別された EKS クラスター内の EC2 インスタンスの 50% を終了します。

```
"actions": {
    "terminateWorkers": {
        "actionId": "aws:eks:terminate-nodegroup-instances",
        "parameters": {
            "instanceTerminationPercentage": "50"
        },
        "targets": {
            "Nodegroups": "targetNodeGroups"
        }
    }
}
```

AWS FIS のターゲット

ターゲットは、実験中に AWS Fault Injection Service (AWS FIS) によってアクションが実行される 1 つ以上の AWS リソースです。ターゲットは実験と同じ AWS アカウントに存在することができ、またマルチアカウント実験を使用する別のアカウントに存在することもできます。別のアカウントのリソースをターゲットにする方法の詳細については、「マルチアカウント実験」を参照してください。

ターゲットは、実験テンプレートの作成を行う場合に定義します。実験テンプレートの複数のアクションに対して同じターゲットを使用できます。

AWS FIS は、アクションセット内のアクションを開始する前に、実験の開始時にすべてのターゲットを識別します。 AWS FIS は、実験全体で選択したターゲットリソースを使用します。ターゲットが見つからない場合、実験は失敗します。

目次

- 選択モード
- ターゲットの例
- フィルターの例

ターゲット構文

次に、 ターゲットの構文を示します。

```
{
    "targets": {
        "target_name": {
            "resourceType": "resource-type",
            "resourceArns": [
                "resource-arn"
            ],
            "resourceTags": {
                "tag-key": "tag-value"
            },
            "parameters": {
                "parameter-name": "parameter-value"
            },
            "filters": [
                {
                    "path": "path-string",
                    "values": ["value-string"]
                }
            ],
            "selectionMode": "value"
        }
    }
}
```

ターゲットを定義するときは、以下の情報を指定します。

target_name

ターゲットの名前。

resourceType

リソースタイプ。

resourceArns

特定のリソースの Amazon リソースネーム (ARN)。

resourceTags

特定のリソースに適用するタグ。

parameters

特定の属性でターゲットを識別するパラメータ。

filters

リソースフィルターは、特定の属性で識別されるターゲットリソースの範囲を設定します。

selectionMode

識別されたリソースの選択モード。

例については、「the section called “ターゲットの例”」を参照してください。

リソースタイプ

各 AWS FIS アクションは、特定の AWS リソースタイプで実行されます。ターゲットを定義する場合は、厳密に 1 つのリソースタイプを指定する必要があります。アクションのターゲットを指定する場合、ターゲットはアクションでサポートされているリソースタイプである必要があります。

AWS FIS では、次のリソースタイプがサポートされています。

- aws:dynamodb:global-table – Amazon DynamoDB グローバルテーブル
- aws:ec2:autoscaling-group – An Amazon EC2 Auto Scaling グループ
- aws:ec2:ebs-volume - Amazon EBS ボリューム
- aws:ec2:instance - Amazon EC2 インスタンス
- aws:ec2:spot-instance - Amazon EC2 スポットインスタンス
- aws:ec2:subnet - Amazon VPC サブネット
- aws:ec2:transit-gateway – トランジットゲートウェイ
- aws:ecs:cluster - Amazon ECS クラスター
- aws: ecs: task - Amazon ECS タスク
- aws:eks:cluster - Amazon EKS クラスター
- aws:eks:nodegroup — Amazon EKS ノードグループ

- aws:eks:pod - Kubernetes ポッド
- aws:elasticache:redis-replicationgroup – Redis ElastiCache レプリケーショングループ
- aws:iam:role — IAM ロール
- aws:rds:cluster — Amazon Aurora DB クラスター
- aws:rds:db — Amazon RDS DB インスタンス
- aws:s3:bucket – Amazon S3 バケット

ターゲットリソースを識別する

AWS FIS コンソールでターゲットを定義する場合、ターゲットとする特定の AWS リソース (特定のリソースタイプのリソース) を選択できます。または、指定した基準に基づいてリソースのグループを AWS FIS に識別させることができます。

ターゲットリソースを識別するには、次の項目を指定できます。

- リソース IDs – 特定のリソースIDs。 AWS すべてのリソース ID は、同じタイプのリソースを表す必要があります。
- リソースタグ — 特定の AWS リソースに適用されるタグ。
- リソースフィルター — 特定の属性を持つリソースを表すパスと値。詳細については、「リソースフィルター」を参照してください。
- リソースパラメータ - 特定の基準を満たすリソースを表すパラメータ。詳細については、「リソースパラメータ」を参照してください。

考慮事項

- 同一ターゲットのリソース ID とリソースタグの両方を指定することはできません。
- 同一ターゲットのリソース ID とリソースフィルターの両方を指定することはできません。
- タグ値が空のリソースタグを指定しても、ワイルドカードを指定することにはなりません。その場合、指定したタグキーと空のタグ値を持つタグがあるリソースが一対象になります。

リソースフィルター

リソースフィルターは、特定の属性に従ってターゲットリソースを識別するクエリです。 AWS FIS は、指定した AWS リソースタイプに従って、リソースの正規説明を含む API アクションの出力にクエリを適用します。クエリに一致する属性を持つリソースは、ターゲット定義に含まれます。

各フィルターは、属性パスと指定可能な値として表されます。パスは、ピリオドで区切られた一連の要素です。リソースの記述アクションの出力における属性までのパスを記述します。リソースの Describe アクションの出力がキャメルケースでも、各要素はパスカルケースで表現する必要があります。例えば、`availablityZone` ではなく、`AvailabilityZone` を属性要素として使用する必要があります。

```
"filters": [
    {
        "path": "component.component.component",
        "values": [
            "string"
        ]
    }
],
```

次の表には、各リソースタイプの正規説明を取得するために使用できる API アクションと AWS CLI コマンドが含まれています。 AWS FIS はユーザーに代わってこれらのアクションを実行して、指定したフィルターを適用します。対応するドキュメントでは、デフォルトで結果に含まれるリソースについて説明します。例えば、DescribeInstances のドキュメントは、最近終了したインスタンスが結果に表示される可能性があると述べています。

リソースタイプ	API アクション	AWS CLI コマンド
aws:ec2:autoscaling-group	DescribeAutoScalingGroups	describe-auto-scaling-groups
aws:ec2:ebs-volume	DescribeVolumes	describe-volumes
aws:ec2:instance	DescribeInstances	describe-instances
aws:ec2:subnet	DescribeSubnets	describe-subnets
aws:ec2:transit-gateway	DescribeTransitゲートウェイ	describe-transit-gateways
aws:ecs:cluster	DescribeClusters	describe-clusters
aws:ecs:task	DescribeTasks	describe-tasks
aws:eks:cluster	DescribeClusters	describe-clusters
aws:eks:nodegroup	DescribeNodegroup	describe-nodegroup

リソースタイプ	API アクション	AWS CLI コマンド
aws:elasticache:redis-replicationgroup	DescribeReplicationグループ	describe-replication-groups
aws:iam:role	ListRoles	list-roles
aws:rds:cluster	DescribeDBClusters	describe-db-clusters
aws:rds:db	DescribeDBInstances	describe-db-instances
aws:s3:bucket	ListBuckets	list-buckets

次のロジックは、すべてのリソースフィルターに適用されます。

- フィルター内の値 — OR
- フィルター間の値 — AND

例については、「the section called “フィルターの例”」を参照してください。

リソースパラメータ

リソースパラメータは、特定の基準に従ってターゲットリソースを識別します。

以下のリソースタイプはパラメータをサポートします。

aws:ec2:ebs-volume

- `availabilityZoneIdentifier` - ターゲットボリュームがあるアベイラビリティーゾーンのコード (例えば、us-east-1a)。

aws:ec2:subnet

- `availabilityZoneIdentifier` - ターゲットサブネットがあるアベイラビリティーゾーンのコード (us-east-1a など) または AZ ID (例えば、use1-az1)。
- `vpc` - ターゲットサブネットがある VPC。アカウントごとに複数の VPC はサポートしていません。

aws:ecs:task

- `cluster` - ターゲットタスクがあるクラスター。
- `service` - ターゲットタスクがあるサービス。

aws:eks:pod

- `availabilityZoneIdentifier` - オプション。ターゲットポッドがあるアベイラビリティーゾーン。例えば `us-east-1d` です。ポッドのアベイラビリティーゾーンは、hostIP とクラスターサブネットの CIDR を比較して決定します。
- `clusterIdentifier` – 必須。ターゲットの EKS クラスターの名前または ARN。
- `namespace` – 必須。ターゲットポッドの Kubernetes 名前空間。
- `selectorType` – 必須。セレクタータイプ。指定できる値は、`labelSelector`、`deploymentName` および `podName` です。
- `selectorValue` – 必須。セレクター値。この値は、`selectorType` の値によって異なります。
- `targetContainerName` - オプション。ポッド仕様で定義されたターゲットコンテナの名前。デフォルトは、各ターゲットポッド仕様で定義されている最初のコンテナです。

aws:rds:cluster

- `writerAvailabilityZoneIdentifiers` - オプション。DB クラスターのライターのアベイラビリティーゾーン。使用できる値: アベイラビリティーゾーンの識別子のカンマ区切りリスト、`all`。

aws:rds:db

- `availabilityZoneIdentifiers` - オプション。影響を受ける DB インスタンスのアベイラビリティーゾーン。使用できる値: アベイラビリティーゾーンの識別子のカンマ区切りリスト、`all`。

aws:elasticache:redis-replicationgroup

- `availabilityZoneIdentifier` – 必須。ターゲットノードを含むアベイラビリティーゾーンのコード (us-east-1a など) または AZ ID (use1-az1 など)。

選択モード

選択モードを指定して、識別されたリソースの範囲を指定します。 AWS FIS は次の選択モードをサポートしています。

- `ALL` - すべてのターゲットに対してアクションを実行します。
- `COUNT(n)` — 識別されたターゲットからランダムに選択された、指定された数のターゲットに対してアクションを実行します。例えば、COUNT (1) は、識別されたターゲットの 1 つを選択します。

- PERCENT(n) — 識別されたターゲットからランダムに選択された、指定された割合のターゲットに対してアクションを実行します。例えば、PERCENT (25) では、識別されたターゲットの 25% が選択されます。

リソース数が奇数で、50% を指定した場合、 AWS FIS は切り下げます。例えば、5 つの Amazon EC2 インスタンスをターゲットとして追加し、スコープを 50% にすると、 AWS FIS は 2 つのインスタンスに切り下げます。1 リソース未満のパーセンテージは指定できません。例えば、4 つの Amazon EC2 インスタンスとスコープを 5% に追加した場合、 AWS FIS はインスタンスを選択できません。

同じターゲットリソースタイプを使用して複数のターゲットを定義した場合、 AWS FIS は同じリソースを複数回選択できます。

使用する選択モードにかかわらず、指定したスコープでリソースが識別されない場合、実験は失敗します。

ターゲットの例

以下はターゲットの例です。

例

- 指定した VPC 内の指定したタグのあるインスタンス
- 指定されたパラメータのタスク

例: 指定した VPC 内の指定されたタグのインスタンス

この例で想定されるターゲットは、タグ env=prod の指定した VPC 内の Amazon EC2 インスタンスです。選択モードでは、 AWS FIS がこれらのターゲットの 1 つをランダムに選択することを指定します。

```
{
    "targets": {
        "randomInstance": {
            "resourceType": "aws:ec2:instance",
            "resourceTags": {
                "env": "prod"
            },
            "filters": [
```

```
                {
                    "path": "VpcId",
                    "values": [
                        "vpc-aabbcc11223344556"
                    ]
                }
            ],
            "selectionMode": "COUNT(1)"
        }
    }
}
```

例: 指定されたパラメータのあるタスク

この例で想定されるターゲットは、指定されたクラスターとサービスがある Amazon ECS タスクです。選択モードでは、 AWS FIS がこれらのターゲットのいずれかをランダムに選択することを指定します。

```
{
    "targets": {
        "randomTask": {
            "resourceType": "aws:ecs:task",
            "parameters": {
                "cluster": "myCluster",
                "service": "myService"
            },
            "selectionMode": "COUNT(1)"
        }
    }
}
```

フィルターの例

以下はフィルターの例です。

例

- EC2インスタンス
- DB クラスター

例: EC2インスタンス

aws:ec2:instance リソースタイプをサポートするアクションのフィルターを指定すると、 AWS FIS は Amazon EC2 describe-instances コマンドを使用し、フィルターを適用してターゲットを識別します。

describe-instances コマンドは、各インスタンスが `Instances` の構造体である JSON 出力を返します。以下に示したのは、*##*でマークされたフィールドの出力の一部です。これらのフィールドで、JSON 出力の構造から属性パスを指定する例を紹介します。

```
{
    "Reservations": [
        {
            "Groups": [],
            "Instances": [
                {
                    "ImageId": "ami-00111111111111111",
                    "InstanceId": "i-00aaaaaaaaaaaaaa",
                    "InstanceType": "t2.micro",
                    "KeyName": "virginia-kp",
                    "LaunchTime": "2020-09-30T11:38:17.000Z",
                    "Monitoring": {
                        "State": "disabled"
                    },
                    "Placement": {
                        "AvailabilityZone": "us-east-1a",
                        "GroupName": "",
                        "Tenancy": "default"
                    },
                    "PrivateDnsName": "ip-10-0-1-240.ec2.internal",
                    "PrivateIpAddress": "10.0.1.240",
                    "ProductCodes": [],
                    "PublicDnsName": "ec2-203-0-113-17.compute-1.amazonaws.com",
                    "PublicIpAddress": "203.0.113.17",
                    "State": {
                        "Code": 16,
                        "Name": "running"
                    },
                    "StateTransitionReason": "",
                    "SubnetId": "subnet-aabbcc11223344556",
                    "VpcId": "vpc-00bbbbbbbbbbbbbbbb",
                    ...
                },
```

```
                ...
                {
                    ...
                }
            ],
            "OwnerId": "123456789012",
            "ReservationId": "r-aaaaaabbbbb111111"
        },
        ...
    ]
}
```

リソースフィルターを使用して特定のアベイラビリティーゾーン内のインスタンスを選択するには、AvailabilityZone の属性パスを指定し、アベイラビリティーゾーンのコードを値として指定します。例:

```
"filters": [
    {
        "path": "Placement.AvailabilityZone",
        "values": [ "us-east-1a" ]
    }
],
```

リソースフィルターを使用して特定のサブネット内のインスタンスを選択するには、SubnetId の属性パスを指定し、値としてサブネットの ID を指定します。例:

```
"filters": [
    {
        "path": "SubnetId",
        "values": [ "subnet-aabbcc11223344556" ]
    }
],
```

特定のインスタンス状態にあるインスタンスを選択するには、Name の属性パスを指定します。値として、次のいずれかの状態名を指定します。pending | running | shutting-down | terminated | stopping | stopped。例:

```
"filters": [
    {
        "path": "State.Name",
        "values": [ "running" ]
```

```
    }
],
```

例: Amazon RDS クラスター (DB クラスター)

aws:rds:cluster リソースタイプをサポートするアクションのフィルターを指定すると、 AWS FIS は Amazon RDS describe-db-clusters コマンドを実行し、フィルターを適用してターゲットを識別します。

describe-db-clusters コマンドは各 DB クラスターに対して次のような JSON 出力を返します。以下に示したのは、##でマークされたフィールドがある出力の一部です。これらのフィールドを使用して、JSON 出力の構造から属性パスを指定する例を紹介します。

```
[
    {
        "AllocatedStorage": 1,
        "AvailabilityZones": [
            "us-east-2a",
            "us-east-2b",
            "us-east-2c"
        ],
        "BackupRetentionPeriod": 7,
        "DatabaseName": "",
        "DBClusterIdentifier": "database-1",
        "DBClusterParameterGroup": "default.aurora-postgresql11",
        "DBSubnetGroup": "default-vpc-01234567abc123456",
        "Status": "available",
        "EarliestRestorableTime": "2020-11-13T15:08:32.211Z",
        "Endpoint": "database-1.cluster-example.us-east-2.rds.amazonaws.com",
        "ReaderEndpoint": "database-1.cluster-ro-example.us-east-2.rds.amazonaws.com",
        "MultiAZ": false,
        "Engine": "aurora-postgresql",
        "EngineVersion": "11.7",
        ...
    }
]
```

特定の DB エンジンを使用する DB クラスターのみを返すリソースフィルターを適用するには、次の例に示すように Engine として属性パス aurora-postgresql として値を指定します。

```
"filters": [
```

```
    {
        "path": "Engine",
        "values": [ "aurora-postgresql" ]
    }
],
```

特定のアベイラビリティーゾーン内の DB クラスターのみを返すリソースフィルターを適用するには、次の例に示すように、属性のパスと値を指定します。

```
"filters": [
    {
        "path": "AvailabilityZones",
        "values": [ "us-east-2a" ]
    }
],
```

AWS FIS の停止条件

AWS Fault Injection Service (AWS FIS) は、 AWS ワークロードで実験を安全に実行するためのコントロールとガードレールを提供します。停止条件は、Amazon CloudWatch アラームとして定義したしきい値に達した場合に実験を停止するメカニズムです。実験中に停止条件がトリガーされると、AWS FIS は実験を停止します。停止した実験を再開することはできません。

停止条件を作成するには、まずアプリケーションまたはサービスの定常状態を定義します。定常状態とは、アプリケーションがビジネスまたは技術メトリクスの観点から最適に動作すると定義される状態です。例えば、レイテンシー、CPU 負荷、または再試行回数などです。定常状態を使用して、アプリケーションまたはサービスがパフォーマンスが許容できない状態になった場合に実験を停止するために使用できる CloudWatch アラームを作成できます。詳細については、「Amazon ユーザーガイド」の「Amazon CloudWatch アラームの使用」を参照してください。 CloudWatch

アカウントには、実験テンプレートで指定できる停止条件の数に制限があります。詳細については、「AWS Fault Injection Service のクォータと制限」を参照してください。

停止条件構文

実験テンプレートを作成するときは、作成した CloudWatch アラームを指定して、1 つ以上の停止条件を指定します。

```
{
```

```
    "stopConditions": [
        {
            "source": "aws:cloudwatch:alarm",
            "value": "arn:aws:cloudwatch:region:123456789012:alarm:alarm-name"
        }
    ]
}
```

次の例は、実験テンプレートが停止条件を指定していないことを示しています。

```
{
    "stopConditions": [
        {
            "source": "none"
        }
    ]
}
```

詳細はこちら

CloudWatch アラームを作成し、実験テンプレートに停止条件を追加する方法を示すチュートリアルについては、「」を参照してください[インスタンス上で CPU ストレスを実行する](#)。

AWS FIS でサポートされているリソースタイプで使用できる CloudWatch メトリクスの詳細については、以下を参照してください。

- [を使用してインスタンスをモニタリングする CloudWatch](#)
- [Amazon ECS CloudWatch メトリクス](#)
- [を使用した Amazon RDS メトリクスのモニタリング CloudWatch](#)
- [を使用した Run Command メトリクスのモニタリング CloudWatch](#)

AWS FIS 実験用の IAM ロール

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コントロールするために役立つ AWS のサービスです。AWS FIS を使用するには、AWS FIS が、ユーザーに代わって実験を実行できるように AWS FIS に必要な権限が認められる IAM ロールを作成する必要があります。この実験ロールは、実験テンプレートの作成時に指定します。単一アカウントの実験の場合、実験ロールの IAM ポリシーは、実験テンプレートでターゲットとして指定したリソース

を変更するアクセス権限を付与する必要があります。マルチアカウントの実験の場合、実験ロールはオーケストレーターロールのアクセス許可を付与し、各ターゲットアカウントの IAM ロールを継承する必要があります。詳細については、「マルチアカウント実験のアクセス許可」を参照してください。

最小権限を付与する標準のセキュリティプラクティスに従うことをお勧めします。これを行うには、ポリシーで特定のリソース ARN またはタグを指定します。

AWS FIS をすぐに使い始めることができるように、AWS 実験ロールを作成するときに指定できるマネージドポリシーを用意しています。また、これらのポリシーをモデルとして使用して、独自のインラインポリシードキュメントを作成することもできます。

内容

前提条件

開始する前に、AWS CLI をインストールして必要な信頼ポリシーを作成します。

AWS CLI のインストール

開始する前に、AWS CLI をインストールして設定します。AWS CLI を設定するときには、AWS 認証情報の入力を求められます。この手順の例では、デフォルトのリージョンも設定済みであることを前提としています。設定していない場合は、`--region` オプションを各コマンドに追加します。詳細については、「AWS CLI のインストールまたは更新」および「AWS CLI の設定」を参照してください。

信頼関係ポリシーを作成する

実験ロールには AWS FIS サービスがこのロールを引き受けることができる信頼関係が必要です。`fis-role-trust-policy.json` という名前のテキストファイルを作成して以下の信頼関係ポリシーを追加します。

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

```
        {
            "Effect": "Allow",
            "Principal": {
                "Service": [
                  "fis.amazonaws.com"
                ]
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

「混乱した代理」問題に対して自分を守るために aws:SourceAccount および aws:SourceArn 条件キーを使用することをお勧めします。ソースアカウントは実験の所有者であり、ソース ARN は実験の ARN です。例えば、次の条件ブロックを信頼ポリシーに追加する必要があります。

```
"Condition": {
    "StringEquals": {
        "aws:SourceAccount": "account_id"
    },
    "ArnLike": {
        "aws:SourceArn": "arn:aws:fis:region:account_id:experiment/*"
    }
}
```

アクセス許可を追加し、ターゲットアカウントロールを継承します (マルチアカウント実験のみ)。

マルチアカウント実験の場合、オーケストレーターアカウントがターゲットアカウントロールを継承することを許可するアクセス許可が必要です。次の例を変更してインラインポリシードキュメントとして追加し、ターゲットアカウントロールを継承することができます。

```
{
    "Effect": "Allow",
    "Action": "sts:AssumeRole",
    "Resource":[
        "arn:aws:iam::target_account_id:role/role_name"
    ]
}
```

オプション 1: 実験ロールを作成して AWS のマネージドポリシーをアタッチする

AWS FIS から AWS のマネージドポリシーのいずれかを利用すれば、すぐに使い始めることができます。

実験ロールを作成して AWS マネージドポリシーをアタッチするには

1. 実験の AWS FIS アクション用のマネージドポリシーがあることを確認します。それ以外の場合は、代わりに独自のインラインポリシードキュメントを作成する必要があります。詳細については、「the section called “AWS マネージドポリシー”」を参照してください。
2. 次の create-role コマンドを使用してロールを作成し、前提条件で作成した信頼ポリシーを追加します。

```
aws iam create-role --role-name my-fis-role --assume-role-policy-document
 file://fis-role-trust-policy.json
```

3. 管理AWSポリシーをアタッチするには、次のattach-role-policyコマンドを使用します。

```
aws iam attach-role-policy --role-name my-fis-role --policy-arn fis-policy-arn
```

fis-policy-arn は次のいずれかです。

- arn:aws:iam::aws:policy/service-role/AWSFaultInjectionSimulatorEC2Access
- arn:aws:iam::aws:policy/service-role/AWSFaultInjectionSimulatorECSAccess
- arn:aws:iam::aws:policy/service-role/AWSFaultInjectionSimulatorEKSAccess
- arn:aws:iam::aws:policy/service-role/AWSFaultInjectionSimulatorNetworkAccess
- arn:aws:iam::aws:policy/service-role/AWSFaultInjectionSimulatorRDSAccess
- arn:aws:iam::aws:policy/service-role/AWSFaultInjectionSimulatorSSMAccess

オプション 2: 実験ロールを作成してインラインポリシードキュメントを追加する

マネージドポリシーがないアクションや、特定の実験に必要な権限のみを含める場合は、このオプションを使用してください。

実験を作成してインラインポリシードキュメントを追加するには

1. 次の create-role コマンドを使用してロールを作成し、前提条件で作成した信頼ポリシーを追加します。

```
aws iam create-role --role-name my-fis-role --assume-role-policy-document
 file://fis-role-trust-policy.json
```

2. `fis-role-permissions-policy.json` という名前のテキストファイルを作成して権限ポリシーを追加します。手順の手始めに使用できる例については、以下の内容を参照してください。

   - フォールトインジェクションアクション - 以下のポリシーから開始します。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowFISExperimentRoleFaultInjectionActions",
            "Effect": "Allow",
            "Action": [
                "fis:InjectApiInternalError",
                "fis:InjectApiThrottleError",
                "fis:InjectApiUnavailableError"
            ],
            "Resource": "arn:*:fis:*:*:experiment/*"
        }
    ]
}
```

   - Amazon EBS アクション - 以下のポリシーから開始します。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ec2:DescribeVolumes"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
```

```
            "Action": [
                "ec2:PauseVolumeIO"
            ],
            "Resource": "arn:aws:ec2:*:*:volume/*"
        }
    ]
}
```

- Amazon EC2 アクション – AWSFaultInjectionSimulatorEC2Accessポリシーから開始します。
- Amazon ECS アクション – AWSFaultInjectionSimulatorECSAccessポリシーから開始します。
- Amazon EKS アクション – AWSFaultInjectionSimulatorEKSAccessポリシーから開始します。
- ネットワークアクション – AWSFaultInjectionSimulatorNetworkAccessポリシーから開始します。
- Amazon RDS アクション – AWSFaultInjectionSimulatorRDSAccessポリシーから開始します。
- Systems Manager アクション – AWSFaultInjectionSimulatorSSMAccessポリシーから開始します。

3. 次のput-role-policyコマンドを使用して、前のステップで作成したアクセス許可ポリシーを追加します。

```
aws iam put-role-policy --role-name my-fis-role --policy-name my-fis-policy --
policy-document file://fis-role-permissions-policy.json
```

実験オプション

実験オプションとは、実験のオプション設定です。実験テンプレートで特定の実験オプションを定義できます。実験を開始すると、追加の実験オプションが設定されます。

以下は、実験テンプレートで定義する実験オプションの構文です。

```
{
        "experimentOptions": {
        "accountTargeting": "single-account | multi-account",
            "emptyTargetResolutionMode": "fail | skip"
    }
}
```

実験テンプレートを作成するときに実験オプションを指定しない場合、各オプションのデフォルトが使用されます。

以下は、実験の開始時に設定した実験オプションの構文です。

```
{
        "experimentOptions": {
            "actionsMode": "run-all | skip-all"
     }
}
```

実験の開始時に実験オプションを指定しない場合、デフォルト`run-all`が使用されます。

内容

アカウントターゲット

実験でターゲットにするリソースを持つ複数の AWS アカウントがある場合は、アカウントターゲット実験オプションを使用してマルチアカウント実験を定義できます。オーケストレーターアカウントからマルチアカウント実験を実行すると、複数のターゲットアカウントのリソースに影響します。オーケストレーターアカウントは、 AWS FIS 実験テンプレートと実験を所有しています。ターゲットアカウントは、 AWS FIS 実験の影響を受ける可能性のあるリソースを持つ個々の AWS アカウントです。詳細については、「のマルチアカウント実験 AWS FIS」を参照してください。

アカウントターゲットを使用して、ターゲットリソースの場所を示します。アカウントターゲットには 2 つの値を指定できます。

- 単一アカウント — デフォルト。実験は、 AWS FIS 実験が実行される AWS アカウントのリソースのみを対象とします。
- マルチアカウント — 実験で複数の AWS アカウントのリソースを対象にできます。

ターゲットアカウントの設定

マルチアカウントの実験を実行するには、1 つ以上のターゲットアカウントの設定を定義する必要があります。ターゲットアカウントの設定では、実験でターゲットとなるリソースを含む各アカウント

の accountId、roleArn、および description を指定します。実験テンプレートのターゲットアカウントの設定のアカウント ID は一意である必要があります。

マルチアカウントの実験テンプレートを作成するとき、実験テンプレートは読み取り専用フィールド `targetAccountConfigurationsCount`、つまり実験テンプレートのすべてのターゲットアカウント設定の数を返します。

ターゲットアカウント設定の構文は次のとおりです。

```
{
    accountId: "123456789012",
    roleArn: "arn:aws:iam::123456789012:role/AllowFISActions",
    description: "fis-ec2-test"
}
```

ターゲットアカウント設定を作成する場合、次を指定します。

accountId

ターゲットアカウントの 12 桁の AWS アカウント ID。

roleArn

ターゲットアカウントでアクションを実行する AWS FIS アクセス許可を付与する IAM ロール。

description

オプションの説明。

ターゲットアカウント設定を使用する方法の詳細については、「the section called “マルチアカウント実験を使用する”」を参照してください。

ターゲット解決モードを空にする

このモードでは、ターゲットリソースが解決されていない場合でも実験を完了させることができます。

- 失敗 - デフォルト。ターゲットに対してリソースが解決されない場合、実験は `failed` ステータスですぐに終了します。
- スキップ - ターゲットに対してリソースが解決されない場合、実験は続行され、ターゲットが解決されていないアクションはスキップされます。ARN などの一意識別子を使用してターゲットが定

義されたアクションはスキップできません。一意識別子を使用して定義されたターゲットが見つからない場合、実験は failed ステータスですぐに終了します。

アクションモード

Actions モードは、実験を開始するときに指定できるオプションのパラメータです。アクションモードを skip-allに設定して、ターゲットリソースに障害を挿入する前にターゲットプレビューを生成できます。ターゲットプレビューでは、以下を確認できます。

- 必要なリソースをターゲットにするように実験テンプレートを設定したこと。リソースはランダムに削除、更新、またはサンプリングされる可能性があるため、この実験の開始時にターゲットとする実際のリソースはプレビューとは異なる場合があります。
- ログ記録設定が正しく設定されていること。
- マルチアカウント実験では、ターゲットアカウント設定ごとに IAM ロールを正しく設定しています。

> **Note**
>
> skip-all モードでは、 AWS FIS 実験を実行し、リソースに対してアクションを実行するために必要なアクセス許可があることを検証することはできません。

アクションモードパラメータは、次の値を受け入れます。

- run-all - (デフォルト) 実験はターゲットリソースに対してアクションを実行します。
- skip-all - 実験では、ターゲットリソースに対するすべてのアクションがスキップされます。

実験を開始するときにアクションモードパラメータを設定する方法の詳細については、「」を参照してください実験テンプレートからターゲットプレビューを生成します。。

AWS FIS 実験テンプレートを使用してください。

AWS FIS コンソールまたはコマンドラインを使用して実験テンプレートを作成および管理できます。作成した実験テンプレートは、実験の実行に使用できます。

タスク

- 実験テンプレートの作成
- 実験テンプレートを表示する
- 実験テンプレートからターゲットプレビューを生成します。
- テンプレートから実験を開始する
- 実験テンプレートを更新する
- 実験テンプレートにタグ付けする
- 実験テンプレートを削除する

実験テンプレートの作成

開始する前に、以下のタスクを完了します。

- 実験の計画。
- ユーザーに代わってアクションを実行する権限を AWS FIS サービスに付与する IAM ロールを作成します。詳細については、「AWS FIS 実験用の IAM ロール」を参照してください。
- FIS AWS にアクセスできることを確認してください。詳細については、「AWS FIS ポリシー例」を参照してください。

コンソールを使用して実験テンプレートを作成するには

1. https://console.aws.amazon.com/fis/ AWS で FIS コンソールを開きます。
2. ナビゲーションペインで、[実験テンプレート] を選択します。
3. [実験テンプレートの作成] を選択します。
4. (オプション) [アカウントターゲティング] の場合、[複数アカウント] を選択してマルチアカウント実験テンプレートを設定します。
5. [アカウントターゲティング] で [確認] を選択します。
6. [説明] と [名前] に、テンプレートの説明と名前を入力します。
7. [アクション] で、テンプレートに対する一連のアクションを指定します。アクションごとに、アクションの追加をクリックし、以下を完了します。
    - [名前] に、アクションの名前を入力します。

使用できる文字は、英数字、ピリオド (.)、および下線 (_) です。名前の最初は、文字で始まっている必要があります。スペースは使用できません。それぞれのアクションの名前はテンプレート内で一意である必要があります。

- (オプション) [説明] に、アクションの簡潔な説明を入力します。最大長は 512 文字です。
- (オプション) [開始後] で、現在のアクションを開始する前に完了する必要がある別のアクションをこのテンプレートから選択します。それ以外の場合、アクションは実験の開始時に実行されます。
- [アクションタイプ] AWS で FIS アクションを選択します。
- [ターゲット] で、[ターゲット] セクションで定義したターゲットを選択します。このアクションのターゲットをまだ定義していない場合、 AWS FIS によって新しいターゲットが作成されます。
- [アクションパラメータ] には、アクションのパラメータを指定します。 AWS このセクションは、FIS アクションにパラメータがある場合にのみ表示されます。
- [保存] を選択します。

8. [ターゲット] で、アクションを実行するターゲットリソースを定義します。ターゲットには、少なくとも 1 つのリソース ID または少なくとも 1 つのリソースタグを指定する必要があります。[Edit] を選択して前のステップで AWS FIS が作成したターゲットを編集するか、[Add target] を選択します。各ターゲットで、以下の作業を行います。

   - [名前] に、ターゲットの名前を入力します。

     使用できる文字は、英数字、ピリオド (.)、および下線 (_) です。名前の最初は、文字で始まっている必要があります。スペースは使用できません。各ターゲット名はテンプレート内で一意である必要があります。
   - [リソースタイプ] で、アクションでサポートされているリソースタイプを選択します。
   - [ターゲットメソッド] には、以下のいずれかを選択します。
     - [リソース ID] を選択し、リソース ID を選択、または追加します。
     - [リソースタグ]、[フィルター]、[パラメータ] を選択し、必要なタグとフィルターを追加します。詳細については、「the section called “ターゲットリソースを識別する”」を参照してください。
   - 選択モードでは、[カウント] を選択して指定した数の識別されたターゲットに対してアクションを実行するか、[パーセント] を選択して指定した割合の識別されたターゲットに対してアクションを実行します。デフォルトでは、アクションは特定されたすべてのターゲットに対して実行されます。

- [保存] を選択します。

9. 作成したターゲットでアクションを更新するには、[アクション」 でアクションを探して [編集] を選択し、[ターゲット] を更新します。複数のアクションに同じターゲットを使用できます。
10. (マルチアカウント実験のみ) [ターゲットアカウント設定] で、各ターゲットアカウントにロールの ARN とオプションの説明を追加します。ターゲットアカウントロールの ARN を CSV ファイルでアップロードするには、[すべてのターゲットアカウントのロール ARN をアップロード] を選択し、[.CSV ファイルを選択] を選択します。
11. [サービスアクセス] で、[既存の IAM ロールを使用する] を選択し、このチュートリアルの前提条件の説明に従って、作成した IAM ロールを選択します。ロールが表示されない場合は、必要な信頼関係があることを確認してください。詳細については、「the section called “実験ロール”」を参照してください。
12. (オプション) 停止条件では、停止条件の Amazon CloudWatch アラームを選択します。詳細については、「AWS FIS の停止条件」を参照してください。
13. (オプション) [ログ] には、宛先オプションを設定します。S3 バケットにログを送信するには、[Amazon S3 バケットに送信] を選択し、バケット名とプレフィックスを入力します。ログをログに送信するには CloudWatch 、[Send to CloudWatch Logs] を選択し、ロググループを入力します。
14. (オプション) タグで、[タグの追加] を選択して、タグのキーと値を指定します。追加するタグは、テンプレートを使用して実行される実験ではなく、実験テンプレートに適用されます。
15. [実験テンプレートの作成] を選択します。確認を求められたら、「**create**」と入力して、[実験テンプレートを作成する] を選択します。

CLI を使用して実験テンプレートを作成するには

create-experiment-templateコマンドを実行します。

JSON ファイルから実験テンプレートを読み込むことができます。

`--cli-input-json` パラメータを使用します。

```
aws fis create-experiment-template --cli-input-json fileb://<path-to-json-file>
```

詳細については、「AWS Command Line Interface ユーザーガイド」の「CLI スケルトンテンプレートの生成」を参照してください。テンプレートの例については、「AWS FIS 実験テンプレートの例」を参照してください。

実験テンプレートを表示する

作成した実験テンプレートを表示できます。

コンソールを使用して実験テンプレートを表示するには

1. https://console.aws.amazon.com/fis/ にある AWS FIS コンソールを開きます。
2. ナビゲーションペインで、[実験テンプレート] を選択します。
3. 特定のテンプレートに関する情報を表示するには、[実験テンプレート ID] を選択します。
4. 詳細セクションでは、テンプレートの説明と停止条件を表示できます。
5. 実験テンプレートのアクションを表示するには、[アクション] を選択します。
6. 実験テンプレートのターゲットを表示するには、[ターゲット を選択します。
7. 実験テンプレートのタグを表示するには、[タグ] を選択します。

CLI を使用して実験テンプレートを表示するには

list-experiment-templatesコマンドを使用して実験テンプレートのリストを取得し、get-experiment-templateコマンドを使用して特定の実験テンプレートに関する情報を取得します。

実験テンプレートからターゲットプレビューを生成します。

テストを開始する前に、ターゲットプレビューを生成して、テストテンプレートが想定されるリソースを対象とするように設定されていることを確認できます。リソースはランダムに削除、更新、またはサンプリングされる可能性があるため、実際の実験を開始したときの対象となるリソースはプレビューと異なる場合があります。ターゲットプレビューを生成すると、すべてのアクションをスキップするテストが開始されます。

> **Note**
>
> ターゲットプレビューを生成しても、リソースに対してアクションを実行するために必要な権限があるかどうかを確認することはできません。

コンソールを使用してターゲットプレビューを開始するには

1. https://console.aws.amazon.com/fis/ AWS で FIS コンソールを開きます。
2. ナビゲーションペインで、[実験テンプレート] を選択します。

3. 実験テンプレートのターゲットを表示するには、[ターゲット を選択します。
4. 実験テンプレートのターゲットリソースを確認するには、「プレビューを生成」を選択します。テストを実行すると、このターゲットプレビューは最新のテストのターゲットで自動的に更新されます。

CLI を使用してターゲットプレビューを開始するには

- 次の start-experiment コマンドを実行します。斜体の値を独自の値に置き換えてください。

```
aws fis start-experiment \
    --experiment-options actionsMode=skip-all \
    --experiment-template-id EXTxxxxxxxxx
```

テンプレートから実験を開始する

実験テンプレートを作成したら、そのテンプレートを使用して実験を開始できます。

実験を開始すると、指定したテンプレートのスナップショットを作成し、そのスナップショットを使用して実験を実行します。したがって、実験の実行中に実験テンプレートが更新または削除された場合、それらの変更は実行中の実験に影響を与えません。

実験を開始すると、 AWS FIS がユーザーに代わってサービスにリンクされたロールを作成します。詳細については、「AWS Fault Injection Service のサービスにリンクされたロールを使用する」を参照してください。

実験を開始したら、いつでも停止できます。詳細については、「実験を中止する」を参照してください。

コンソールを使用して実験を開始するには

1. https://console.aws.amazon.com/fis/ で AWS FIS コンソールを開きます。
2. ナビゲーションペインで、[実験テンプレート] を選択します。
3. (オプション) プレビューを生成してターゲットを検証するには:
   - [ターゲット] を選択します。
   - [プレビューを生成] を選択します。
4. 実験テンプレートを選択し、[実験を開始する] を選択します。

5. (オプション) 実験にタグを追加するには、[新しいタグを追加] を選択し、タグキーとタグ値を入力します。
6. [実験を開始する] を選択します。確認を求められたら、**start** を入力して、[実験を開始する] を選択します。

CLI を使用して実験を開始するには

start-experiment コマンドを使用します。

実験テンプレートを更新する

既存の実験テンプレートを更新できます。実験テンプレートを更新しても、そのテンプレートを使用する実行中の実験には影響しません。

コンソールを使用して実験テンプレートを更新するには

1. https://console.aws.amazon.com/fis/ AWS で FIS コンソールを開きます。
2. ナビゲーションペインで、[実験テンプレート] を選択します。
3. 実験テンプレートを選択し、[アクション]、[実験テンプレートを更新する] を選択します。
4. 必要に応じてテンプレートの詳細を変更し、[実験テンプレートを更新する] を選択します。

CLI を使用して実験テンプレートを更新するには

update-experiment-templateコマンドを実行します。

実験テンプレートにタグ付けする

実験テンプレートを整理しやすくするために、実験テンプレートに独自のタグを適用できます。また、タグベースの IAM ポリシーを使用して、実験テンプレートへのアクセスへの制御を実装することもできます。

コンソールを使用して実験テンプレートにタグを付けるには

1. https://console.aws.amazon.com/fis/ AWS で FIS コンソールを開きます。
2. ナビゲーションペインで、[実験テンプレート] を選択します。
3. 実験テンプレートを選択し、[アクション]、[タグの管理]を選択します。
4. タグを追加するには、[新しいタグを追加] を選択し、キー名とキーの値を指定します。

タグを削除するには、タグの [削除] を選択します。

5. [保存] を選択します。

CLI を使用して実験テンプレートにタグを付けるには

tag-resource コマンドを使用します。

実験テンプレートを削除する

実験テンプレートが不要になった場合は、削除できます。実験テンプレートを削除しても、そのテンプレートを使用する実行中の実験は影響を受けません。実験は、完了または停止するまで実行され続けます。ただし、削除された実験テンプレートは、コンソールの [実験] ページで見ることはできません。

コンソールを使用して実験テンプレートを削除するには

1. https://console.aws.amazon.com/fis/ AWS で FIS コンソールを開きます。
2. ナビゲーションペインで、[実験テンプレート] を選択します。
3. 実験テンプレートを選択し、[アクション]、[実験テンプレートの削除] を選択します。
4. 確認を求められたら、**delete** と入力し、[実験テンプレートの削除] を選択します。

CLI を使用して実験テンプレートを削除するには

delete-experiment-templateコマンドを実行します。

AWS FIS 実験テンプレートの例

AWS FIS API またはコマンドラインツールを使用して実験テンプレートを作成する場合は、JavaScript Object Notation (JSON) でテンプレートを作成できます。実験テンプレートのコンポーネントの詳細については、「テンプレートコンポーネント」を参照してください。

サンプルテンプレートのいずれかを使用して実験テンプレートを作成するには、それを JSON ファイルに保存します (例えば、`my-template.json`)。そして独自の値で、`##`のプレースホルダの値を置き換えて、次の実験テンプレートの作成コマンドを実行します。

```
aws fis create-experiment-template --cli-input-json file://my-template.json
```

テンプレートの例

- フィルターに基づいて EC2 インスタンスを停止する
- 指定された数の EC2 インスタンスを停止する
- 事前設定された AWS FIS SSM ドキュメントを実行する
- 事前定義されたオートメーション Runbook を実行する
- ターゲット IAM ロールを使用して EC2 インスタンスの API アクションをスロットルします
- Kubernetes クラスタ内のポッドの CPU のストレステスト

フィルターに基づいて EC2 インスタンスを停止する

次の例では、指定された VPC 内の指定されたタグを持つ、指定されたリージョンで実行中の Amazon EC2 インスタンスをすべて停止します。2 分後に再起動します。

```
{
    "tags": {
        "Name": "StopEC2InstancesWithFilters"
    },
    "description": "Stop and restart all instances in us-east-1b with the tag env=prod
 in the specified VPC",
    "targets": {
        "myInstances": {
            "resourceType": "aws:ec2:instance",
            "resourceTags": {
                "env": "prod"
```

```
            },
            "filters": [
                {
                    "path": "Placement.AvailabilityZone",
                    "values": ["us-east-1b"]
                },
                {
                    "path": "State.Name",
                    "values": ["running"]
                },
                {
                    "path": "VpcId",
                    "values": [ "vpc-aabbcc11223344556"]
                }
            ],
            "selectionMode": "ALL"
        }
    },
    "actions": {
        "StopInstances": {
            "actionId": "aws:ec2:stop-instances",
            "description": "stop the instances",
            "parameters": {
                "startInstancesAfterDuration": "PT2M"
            },
            "targets": {
                "Instances": "myInstances"
            }
        }
    },
    "stopConditions": [
        {
            "source": "aws:cloudwatch:alarm",
            "value": "arn:aws:cloudwatch:us-east-1:111122223333:alarm:alarm-name"
        }
    ],
    "roleArn": "arn:aws:iam::111122223333:role/role-name"
}
```

指定された数の EC2 インスタンスを停止する

次の例では、指定されたタグを持つ 3 つのインスタンスを停止します。 AWS FIS は、ランダムに停止する特定のインスタンスを選択します。2 分後にこれらのインスタンスが再起動されます。

```
{
    "tags": {
        "Name": "StopEC2InstancesByCount"
    },
    "description": "Stop and restart three instances with the specified tag",
    "targets": {
        "myInstances": {
            "resourceType": "aws:ec2:instance",
            "resourceTags": {
                "env": "prod"
            },
            "selectionMode": "COUNT(3)"
        }
    },
    "actions": {
        "StopInstances": {
            "actionId": "aws:ec2:stop-instances",
            "description": "stop the instances",
            "parameters": {
                "startInstancesAfterDuration": "PT2M"
            },
            "targets": {
                "Instances": "myInstances"
            }
        }
    },
    "stopConditions": [
        {
            "source": "aws:cloudwatch:alarm",
            "value": "arn:aws:cloudwatch:us-east-1:111122223333:alarm:alarm-name"
        }
    ],
    "roleArn": "arn:aws:iam::111122223333:role/role-name"
}
```

事前設定された AWS FIS SSM ドキュメントを実行する

次の例では、事前設定された AWS FIS SSM ドキュメント -Run-CPU-Stress . AWS FIS を使用して、指定された EC2 インスタンスで 60 秒間 CPU フォールトインジェクションを実行します。FIS は実験を 2 分間モニタリングします。 AWSFIS

```
{
```

```
    "tags": {
        "Name": "CPUStress"
    },
    "description": "Run a CPU fault injection on the specified instance",
    "targets": {
        "myInstance": {
            "resourceType": "aws:ec2:instance",
            "resourceArns": ["arn:aws:ec2:us-east-1:111122223333:instance/instance-
id"],
            "selectionMode": "ALL"
        }
    },
    "actions": {
        "CPUStress": {
            "actionId": "aws:ssm:send-command",
            "description": "run cpu stress using ssm",
            "parameters": {
                "duration": "PT2M",
                "documentArn": "arn:aws:ssm:us-east-1::document/AWSFIS-Run-CPU-Stress",
                "documentParameters": "{\"DurationSeconds\": \"60\",
 \"InstallDependencies\": \"True\", \"CPU\": \"0\"}"
            },
            "targets": {
                "Instances": "myInstance"
            }
        }
    },
    "stopConditions": [
        {
            "source": "aws:cloudwatch:alarm",
            "value": "arn:aws:cloudwatch:us-east-1:111122223333:alarm:alarm-name"
        }
    ],
    "roleArn": "arn:aws:iam::111122223333:role/role-name"
}
```

事前定義されたオートメーション Runbook を実行する

次の例では、Systems Manager が提供する Runbook を使用して Amazon SNS に通知を発行します。AWS-PublishSNSNotification。指定された SNS トピックに通知を発行する権限をロールに付与する必要があります。

```
{
    "description": "Publish event through SNS",
    "stopConditions": [
        {
            "source": "none"
        }
    ],
    "targets": {
    },
    "actions": {
        "sendToSns": {
            "actionId": "aws:ssm:start-automation-execution",
            "description": "Publish message to SNS",
            "parameters": {
                "documentArn": "arn:aws:ssm:us-east-1::document/AWS-
PublishSNSNotification",
                "documentParameters": "{\"Message\": \"Hello, world\", \"TopicArn\":
 \"arn:aws:sns:us-east-1:111122223333:topic-name\"}",
                "maxDuration": "PT1M"
            },
            "targets": {
            }
        }
    },
    "roleArn": "arn:aws:iam::111122223333:role/role-name"
}
```

ターゲット IAM ロールを使用して EC2 インスタンスの API アクションをスロットルします

次の例では、ターゲット定義で指定された IAM ロール (複数可) によって行われた API コールのアクション定義で指定された API コールの 100% をスロットリングします。

> **Note**
>
> Auto Scaling グループのメンバーである EC2 インスタンスをターゲットにする場合は、aws:ec2:asg-insufficient-instance-capacity-error アクションを使用し、代わりに Auto Scaling グループでターゲットにしてください。詳細については、「

ターゲットの Auto Scaling グループからのリクエストで `InsufficientInstanceCapacity` エラーを挿入します。このアクションは、起動テンプレートを使用する Auto Scaling グループのみをサポートします。インスタンス容量不足エラーに関する詳細については、「Amazon EC2 ユーザーガイド」を参照してください。

リソースタイプ

- aws:ec2:autoscaling-group

パラメータ

- duration – AWS FIS API では、値は ISO 8601 形式の文字列です。例えば、PT1M は 1 分を表します。 AWS FIS コンソールで、秒数、分数、または時間数を入力します。

- availabilityzoneIdentifiers - アベイラビリティーゾーンのカンマ区切りリスト。ゾーン ID (`"use1-az1, use1-az2"` など) とゾーン名 (`"us-east-1a"` など) をサポートします。
- percentage - オプション。障害を挿入するターゲット Auto Scaling グループの起動リクエストの割合 (1 ~ 100)。デフォルトは 100 です。

アクセス許可

- `ec2:InjectApiError`条件キー ec2:FisActionId 値を に設定`aws:ec2:asg-insufficient-instance-capacity-error`し、`ec2:FisTargetArns`条件キーをターゲット Auto Scaling グループに設定します。

- `autoscaling:DescribeAutoScalingGroups`

ポリシーの例については、「例: `ec2:InjectApiError` の条件キーを使用します。」を参照してください。

」を参照してください。

```
{
    "tags": {
        "Name": "ThrottleEC2APIActions"
```

```
    },
    "description": "Throttle the specified EC2 API actions on the specified IAM role",
    "targets": {
        "myRole": {
            "resourceType": "aws:iam:role",
            "resourceArns": ["arn:aws:iam::111122223333:role/role-name"],
            "selectionMode": "ALL"
        }
    },
    "actions": {
        "ThrottleAPI": {
            "actionId": "aws:fis:inject-api-throttle-error",
            "description": "Throttle APIs for 5 minutes",
            "parameters": {
                "service": "ec2",
                "operations": "DescribeInstances,DescribeVolumes",
                "percentage": "100",
                "duration": "PT2M"
            },
            "targets": {
                "Roles": "myRole"
            }
        }
    },
    "stopConditions": [
        {
            "source": "aws:cloudwatch:alarm",
            "value": "arn:aws:cloudwatch:us-east-1:111122223333:alarm:alarm-name"
        }
    ],
    "roleArn": "arn:aws:iam::111122223333:role/role-name"
}
```

Kubernetes クラスタ内のポッドの CPU のストレステスト

次の例では、Chaos Mesh を使用して Amazon EKS Kubernetes クラスター内のポッドの CPU について 1 分間ストレステストを行います。

```
{
    "description": "ChaosMesh StressChaos example",
    "targets": {
        "Cluster-Target-1": {
```

```
            "resourceType": "aws:eks:cluster",
            "resourceArns": [
                "arn:aws:eks:arn:aws::111122223333:cluster/cluster-id"
            ],
            "selectionMode": "ALL"
        }
    },
    "actions": {
        "TestCPUStress": {
            "actionId": "aws:eks:inject-kubernetes-custom-resource",
            "parameters": {
                "maxDuration": "PT2M",
                "kubernetesApiVersion": "chaos-mesh.org/v1alpha1",
                "kubernetesKind": "StressChaos",
                "kubernetesNamespace": "default",
                "kubernetesSpec": "{\"selector\":{\"namespaces\":[\"default\"],
\"labelSelectors\":{\"run\":\"nginx\"}},\"mode\":\"all\",\"stressors\": {\"cpu\":
{\"workers\":1,\"load\":50}},\"duration\":\"1m\"}"
            },
            "targets": {
                "Cluster": "Cluster-Target-1"
            }
        }
    },
    "stopConditions": [{
        "source": "none"
    }],
    "roleArn": "arn:aws:iam::111122223333:role/role-name",
    "tags": {}
}
```

以下の例では、Litmus を使用して Amazon EKS Kubernetes クラスター内のポッドの CPU について 1 分間ストレステストを行います。

```
{
    "description": "Litmus CPU Hog",
    "targets": {
        "MyCluster": {
            "resourceType": "aws:eks:cluster",
            "resourceArns": [
                "arn:aws:eks:arn:aws::111122223333:cluster/cluster-id"
            ],
            "selectionMode": "ALL"
```

```
        }
    },
    "actions": {
        "MyAction": {
            "actionId": "aws:eks:inject-kubernetes-custom-resource",
            "parameters": {
                "maxDuration": "PT2M",
                "kubernetesApiVersion": "litmuschaos.io/v1alpha1",
                "kubernetesKind": "ChaosEngine",
                "kubernetesNamespace": "litmus",
                "kubernetesSpec": "{\"engineState\":\"active\",\"appinfo\":
{\"appns\":\"default\",\"applabel\":\"run=nginx\",\"appkind\":\"deployment\"},
\"chaosServiceAccount\":\"litmus-admin\",\"experiments\":[{\"name\":\"pod-cpu-hog
\",\"spec\":{\"components\":{\"env\":[{\"name\":\"TOTAL_CHAOS_DURATION\",\"value\":
\"60\"},{\"name\":\"CPU_CORES\",\"value\":\"1\"},{\"name\":\"PODS_AFFECTED_PERC\",
\"value\":\"100\"},{\"name\":\"CONTAINER_RUNTIME\",\"value\":\"docker\"},{\"name\":
\"SOCKET_PATH\",\"value\":\"/var/run/docker.sock\"}]},\"probe\":[]}}],\"annotationCheck
\":\"false\"}"
            },
            "targets": {
                "Cluster": "MyCluster"
            }
        }
    },
    "stopConditions": [{
        "source": "none"
    }],
    "roleArn": "arn:aws:iam::111122223333:role/role-name",
    "tags": {}
}
```

のマルチアカウント実験 AWS FIS

マルチアカウント実験では、リージョン内の複数の AWS アカウントにまたがるアプリケーションに対して実際の障害シナリオを設定して実行できます。オーケストレーターアカウントからマルチアカウント実験を実行すると、複数のターゲットアカウントのリソースに影響します。

マルチアカウント実験を実行すると、影響を受けるリソースのあるターゲットアカウントに AWS Health Dashboard から通知され、ターゲットアカウントのユーザーが認識できるようにします。マルチアカウント実験では、次のことができます。

- AWS FIS が提供する中央コントロールとガードレールを使用して、複数のアカウントにまたがるアプリケーションで実際の障害シナリオを実行します。
- きめ細かなアクセス許可と各ターゲットの範囲を定義するタグのある IAM ロールを使用して、マルチアカウント実験の効果を制御します。
- から各アカウントで AWS FIS 実行されたアクション AWS Management Console を AWS FIS ログを通じて一元的に表示します。
- AWS を使用して、各アカウントで AWS FIS が行う API コールをモニタリングおよび監査します CloudTrail。

このセクションは、マルチアカウント実験を開始するのに役立ちます。

トピック

マルチアカウント実験の概念

マルチアカウント実験の主な概念は次のとおりです。

オーケストレーターアカウント

オーケストレーターアカウントは、 AWS FIS コンソールで実験を設定および管理し、ログ記録を一元化するための中央アカウントとして機能します。オーケストレーターアカウントは、 AWS FIS 実験テンプレートと実験を所有しています。

ターゲットアカウント

ターゲットアカウントは、 AWS FIS マルチアカウント実験の影響を受ける可能性のあるリソースを持つ個々の AWS アカウントです。

ターゲットアカウントの設定

実験テンプレートにターゲットアカウント設定を追加して、実験の一部であるターゲットアカウントを定義します。ターゲットアカウント設定は、マルチアカウント実験に必要な実験テンプレートの要素です。アカウント ID、IAM ロール、およびオプションの説明を設定して、ターゲット AWS アカウントごとに 1 つ定義します。

マルチアカウント実験の前提条件

マルチアカウント実験で停止条件を使用するには、最初にクロスアカウントアラームを設定する必要があります。IAM ロールは、マルチアカウント実験テンプレートを作成するときに定義されます。テンプレートを作成する前に、必要な IAM ロールを作成できます。

コンテンツ

マルチアカウント実験のアクセス許可

マルチアカウント実験は、IAM ロールの連鎖を使用して、 AWS FIS にターゲットアカウントのリソースでアクションを実行するアクセス許可を付与します。マルチアカウント実験の場合、各ターゲットアカウントとオーケストレーターアカウントで IAM ロールを設定します。これらの IAM ロールには、ターゲットアカウントとオーケストレーターアカウントの間、およびオーケストレーターアカウントと AWS FISとの間の信頼関係が必要です。

ターゲットアカウントの IAM ロールには、リソースでアクションを実行するのに必要なアクセス許可が含まれており、ターゲットアカウント設定を追加することにより実験テンプレートに対して作成します。ターゲットアカウントのロールを継承し、 AWS FISとの信頼関係を確立するアクセス許可を持つオーケストレーターアカウントの IAM ロールを作成します。この IAM ロールは実験テンプレートの `roleArn` として使用されます。

ロールの連鎖に関する詳細については、「IAM ユーザーガイド」の「ロールに関する用語と概念」を参照してください。

次の例では、オーケストレーターアカウント A に、ターゲットアカウント B で `aws:ebs:pause-volume-io` を使用した実験を実行するアクセス許可を設定します。

1. アカウント B で、アクションの実行に必要なアクセス許可のある IAM ロールを作成します。各アクションに必要なアクセス許可については、「the section called “アクションリファレンス”」を参照してください。次の例は、EBS Pause Volume IO アクション the section called “aws:ebs:pause-volume-io” を実行するためにターゲットアカウントが付与するアクセス許可を示しています。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ec2:DescribeVolumes"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:PauseVolumeIO"
            ],
            "Resource": "arn:aws:ec2:region:accountIdB:volume/*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "tag:GetResources"
            ],
            "Resource": "*"
        }
    ]
}
```

2. 次に、アカウント A との信頼関係を作成する信頼ポリシーをアカウント B に追加します。ステップ 3 で作成するアカウント A の IAM ロールの名前を選択します。

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

```
        {
           "Effect": "Allow",
            "Principal": {
                "AWS": "AccountIdA"
            },
            "Action": "sts:AssumeRole",
            "Condition": {
                "StringLike":{
                    "sts:ExternalId": "arn:aws:fis:region:accountIdA:experiment/*"
                },
                "ArnEquals": {
                    "aws:PrincipalArn": "arn:aws:iam::accountIdA:role/role_name"
                }
            }
        }
    ]
}
```

3. アカウント A で IAM ロールを作成します。このロール名は、ステップ 2 の信頼ポリシーで指定したロールと一致している必要があります。複数のアカウントをターゲットにするには、オーケストレーターに各ロールを継承するアクセス許可を付与します。次の例は、アカウント A がアカウント B を継承するアクセス許可を示しています。追加のターゲットアカウントがある場合、このポリシーにロール ARN を追加します。ターゲットアカウントごとに 1 つだけロール ARN を持つことができます。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "sts:AssumeRole",
            "Resource": [
                "arn:aws:iam::accountIdB:role/role_name"
            ]
        }
    ]
}
```

4. アカウント A のこの IAM ロールは実験テンプレートの `roleArn` として使用されます。次の例は、アカウント A、オーケストレーターアカウントを引き受ける AWS FIS アクセス許可を付与する IAM ロールで必要な信頼ポリシーを示しています。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": [
                    "fis.amazonaws.com"
                ]
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

Stacksets を使用し、一度に複数の IAM ロールをプロビジョニングすることもできます。を使用するには CloudFormation StackSets、 AWS アカウントに必要な StackSet アクセス許可を設定する必要があります。詳細については、「AWS の使用 CloudFormation StackSets」を参照してください。

マルチアカウント実験の停止条件 (オプション)

停止条件は、アラームとして定義したしきい値に達した場合に実験を停止する仕組みです。マルチアカウント実験に対して停止条件を設定するには、クロスアカウントアラームを使用することができます。読み取り専用のアクセス許可を使用してオーケストレーターアカウントがアラームを利用できるようにするには、各ターゲットアカウントで共有を有効にする必要があります。共有すると、Metric Math を使用して異なるターゲットアカウントからメトリクスを組み合わせることができます。その後、このアラームを実験の停止条件として追加できます。

クロスアカウントダッシュボードの詳細については、「 でのクロスアカウント機能の有効化」を参照してください CloudWatch。

マルチアカウント実験を使用する

AWS FIS コンソールまたはコマンドラインを使用して、マルチアカウント実験テンプレートを作成および管理できます。マルチアカウント実験を作成するには、`"multi-account"` としてアカウントターゲティング実験オプションを指定し、ターゲットアカウント設定を追加します。マルチアカウント実験テンプレートを作成した後、実験の実行に使用できます。

コンテンツ

- マルチアカウント実験のベストプラクティス
- マルチアカウント実験テンプレートを作成する
- ターゲットアカウント設定を更新する
- ターゲットアカウント設定を削除する

マルチアカウント実験のベストプラクティス

マルチアカウント実験を使用するためのベストプラクティスは次のとおりです。

- マルチアカウント実験用のターゲットを設定する場合、すべてのターゲットアカウントで一貫したリソースタグを使用してターゲティングすることをお勧めします。 AWS FIS 実験では、各ターゲットアカウントで一貫したタグを持つリソースを解決します。アクションは、`emptyTargetResolutionMode` が `skip` に設定されている実験を除いて、任意のターゲットアカウントで少なくとも 1 つのターゲットリソースを解決する必要があります。アクションクォータはアカウントごとに適用されます。リソース ARN によりリソースをターゲットにする場合、アクションごとに同じ単一のアカウント制限が適用されます。
- パラメータまたはフィルターを使用して 1 つ以上のアベイラビリティーゾーンのリソースをターゲットにする場合、AZ 名ではなく AZ ID を指定する必要があります。AZ ID は、アカウント間で同じアベイラビリティーゾーンを一貫して示すための一意の識別子です。アカウントのアベイラビリティーゾーンの AZ ID を調べる方法については、「Availability Zone IDs for your AWS resources」を参照してください。

マルチアカウント実験テンプレートを作成する

を使用して実験テンプレートを作成する方法を学ぶには AWS Management Console

実験テンプレートの作成 を参照してください。

CLI を使用して実験テンプレートを作成するには

1. を開く AWS Command Line Interface
2. アカウントターゲティング実験オプションを `"multi-account"` に設定 (`my-template.json` など) して保存された JSON ファイルから実験を作成するには、*##*のプレースホルダーの値を独自の値に置き換えて、次の create-experiment-template コマンドを実行します。

```
aws fis create-experiment-template --cli-input-json file://my-template.json
```

これにより、レスポンスで実験テンプレートが返されます。応答から実験テンプレートの ID である id をコピーします。

3. create-target-account-configuration コマンドを実行して、実験テンプレートにターゲットアカウント設定を追加します。ステップ 2 の id を --experiment-template-id パラメータの値として使用し、##のプレースホルダーの値を独自の値に置き換え、次を実行します。--description パラメータはオプションです。ターゲット アカウントごとにこのステップを繰り返します。

```
aws fis create-target-account-configuration --experiment-template-id EXTxxxxxxxxx
 --account-id 111122223333 --role-arn arn:aws:iam::111122223333:role/role-name --
description "my description"
```

4. get-target-account-configuration コマンドを実行して、特定のターゲットアカウント設定の詳細を取得します。

```
aws fis get-target-account-configuration --experiment-template-id EXTxxxxxxxxx --
account-id 111122223333
```

5. ターゲットアカウント設定をすべて追加したら、list-target-account-configurations コマンドを実行して、ターゲットアカウント設定が作成されたことを確認できます。

```
aws fis list-target-account-configurations --experiment-template-id EXTxxxxxxxxx
```

get-experiment-template コマンドを実行して、ターゲットアカウント設定が追加されたことを確認することもできます。テンプレートは、実験テンプレートのすべてのターゲットアカウント設定の数を示す読み取り専用フィールド targetAccountConfigurationsCount を返します。

6. 準備ができたら、start-experiment コマンドを使用して実験テンプレートを実行できます。

```
aws fis start-experiment --experiment-template-id EXTxxxxxxxxx
```

ターゲットアカウント設定を更新する

ロールの ARN またはアカウントの説明を変更する場合、既存のターゲットアカウント設定を更新できます。ターゲットアカウント設定を更新するとき、その変更はテンプレートを使用する実行中の実験には影響しません。

を使用してターゲットアカウント設定を更新するには AWS Management Console

1. https://console.aws.amazon.com/fis/ で AWS FIS コンソールを開きます。
2. ナビゲーションペインで、[実験テンプレート] を選択します。
3. 実験テンプレートを選択し、[アクション]、[実験テンプレートを更新する] を選択します。
4. [ターゲットアカウント設定] を変更し、[実験テンプレートを更新] を選択します。

CLI を使用してターゲットアカウント設定を更新するには

*##*のプレースホルダーの値を独自の値に置き換え、update-target-account-configuration コマンドを実行します。`--role-arn` および `--description` パラメータはオプションで、含まれていない場合は更新されません。

```
aws fis update-target-account-configuration --experiment-template-id EXTxxxxxxxxx
 --account-id 111122223333 --role-arn arn:aws:iam::111122223333:role/role-name --
description "my description"
```

ターゲットアカウント設定を削除する

ターゲットアカウント設定が不要になった場合には、それを削除することができます。ターゲットアカウント設定を削除するとき、テンプレートを使用する実行中の実験には影響しません。実験は、完了または停止するまで実行され続けます。

を使用してターゲットアカウント設定を削除するには AWS Management Console

1. https://console.aws.amazon.com/fis/ で AWS FIS コンソールを開きます。
2. ナビゲーションペインで、[実験テンプレート] を選択します。
3. 実験テンプレートを選択し、[アクション]、[更新] を選択します。
4. [ターゲットアカウント設定] の下で、削除するターゲットアカウントのロール ARN の [削除] を選択します。

CLI を使用してターゲットアカウント設定を削除するには

*##*のプレースホルダーの値を独自の値に置き換え、delete-target-account-configuration コマンドを実行します。

```
aws fis update-target-account-configuration --experiment-template-id EXTxxxxxxxxx --
account-id 111122223333
```

AWS FIS シナリオライブラリ

シナリオでは、アプリケーションが実行されているコンピュートリソースの中断など、お客様がアプリケーションの耐障害性をテストするために適用できるイベントや条件を定義します。シナリオは AWS が作成し、所有権利は AWS に帰属します。一般的なアプリケーション障害に対する事前定義されたターゲットと障害アクションを行って (自動スケーリンググループの 30% のインスタンスが停止するなど)、ユーザー側の Undifferentiated Heavy Lifting (付加価値を生まない作業) を最小限に抑えます。

トピック

AWS FIS シナリオの使用

シナリオはコンソール専用のシナリオライブラリを通じて提供され、 AWS FIS 実験テンプレートで実行します。シナリオで実験を実行するには、ライブラリからシナリオを選択し、ワークロードの詳細と一致するパラメータを指定して、実験テンプレートとしてアカウントに保存します。

トピック

シナリオの表示

コンソールでシナリオを表示するには

1. で AWS FIS コンソールを開きますhttps://console.aws.amazon.com/fis/。
2. ナビゲーションペインで、[シナリオライブラリ] を選択します。
3. 特定のシナリオに関する情報を表示するには、シナリオカードを選択して分割パネルを表示します。

- ページ下部の分割パネルの [説明] タブには、シナリオの簡単な説明が表示されます。また、必要な対象リソースの概要や、シナリオで使用するリソースを準備するために実行する必要のあるアクションなど、前提条件の簡単な概要も表示されます。最後に、シナリオ内のターゲットとアクションに関する追加情報や、テストがデフォルト設定で正常に実行されるまでの予想期間も確認できます。
- ページ下部の分割パネルの [コンテンツ] タブでは、シナリオから作成される実験テンプレートの一部が入力されたバージョンをプレビューできます。
- ページ下部の分割パネルの [詳細] タブには、シナリオの実装方法の詳細な説明があります。これには、シナリオの個々の側面をどのように見積もるかについての詳細な情報が含まれている場合があります。該当する場合は、停止条件として使用する指標や、実験から学ぶための観測可能性についても記載されています。最後に、結果の実験テンプレートの拡張方法に関する推奨事項を紹介します。

シナリオの使用

コンソールを使用してシナリオを使用するには:

1. で AWS FIS コンソールを開きますhttps://console.aws.amazon.com/fis/。
2. ナビゲーションペインで、[シナリオライブラリ] を選択します。
3. 特定のシナリオに関する情報を表示するには、シナリオカードを選択して分割パネルを表示します
4. シナリオを使用するには、シナリオカードを選択し、「シナリオ付きのテンプレートを作成」を選択します。
5. [実験テンプレートの作成] ビューで、不足している項目をすべて入力します。
   a. 一部のシナリオでは、複数のアクションやターゲットで共有されているパラメータを一括編集できます。この機能は、パラメータの一括編集による変更を含め、シナリオに変更を加えると無効になります。この機能を使用するには、「一括パラメータを編集」ボタンを選択します。モーダルでパラメータを編集し、[保存] ボタンを選択します。
   b. 実験テンプレートによっては、各アクションとターゲットカードで強調表示されているアクションやターゲットパラメータがない場合があります。各カードの [編集] ボタンを選択し、不足している情報を追加して、カードの [保存] ボタンを選択します。
   c. すべてのテンプレートには、サービスアクセス実行ロールが必要です。この実験テンプレートには、既存のロールを選択するか、新しいロールを作成できます。

d. 既存の AWS CloudWatch アラームを選択して、1 つ以上のオプションの停止条件を定義することをお勧めします。AWS FIS の停止条件 の詳細を確認してください。アラームをまだ設定していない場合は、「Amazon CloudWatch アラームの使用」の指示に従い、後で実験テンプレートを更新できます。

e. Amazon ログまたは Amazon S3 バケットへのオプションの実験 CloudWatch ログを有効にすることをお勧めします。AWS FIS の実験ロギング の詳細を確認してください。適切なリソースをまだ設定していない場合は、後で実験テンプレートを更新できます。

6. [実験テンプレートの作成] で [実験テンプレートの作成] を選択します。
7. AWS FIS コンソールの実験テンプレートビューから、実験の開始 を選択します。AWS FIS の実験 の詳細を確認してください。

シナリオのエクスポート

シナリオはコンソールのみで操作します。実験テンプレートと似ていますが、シナリオは完全な実験テンプレートではないため、 AWS FISに直接インポートすることはできません。シナリオを独自のオートメーションの一部として使用したい場合は、次の 2 つの方法のいずれかを使用できます。

1. の手順に従って有効な AWS FIS 実験テンプレートを作成し、そのテンプレートシナリオの使用をエクスポートします。
2. [コンテンツ] タブから シナリオの表示 の手順とステップ 3 の手順に従い、シナリオコンテンツをコピーして保存し、不足しているパラメータを手動で追加して有効な実験テンプレートを作成します。

シナリオライブラリの AWS FIS シナリオ

シナリオライブラリに含まれるシナリオは、可能な限りタグを使用するように設計されており、各シナリオでは、シナリオ説明の「前提条件」セクションと「仕組み」セクションで必要なタグについて説明しています。これらの定義済みのタグでリソースをタグ付けすることも、一括パラメータ編集機能を使用して独自のタグを設定することもできます (「シナリオの使用」を参照)。

このリファレンスでは、AWS FIS シナリオライブラリの共通のシナリオについて説明します。AWS FIS コンソールを使用して、サポートされているシナリオを一覧表示することもできます。

詳細については、「シナリオによる作業」を参照してください。

AWS FIS は、次の Amazon EC2 のシナリオをサポートしています。これらのシナリオは、タグを使用するインスタンスをターゲットにしています。独自のタグを使用することも、シナリオに含められ

ているデフォルトのタグを使用することもできます。一部のシナリオは、SSM ドキュメントを使用します。

- EC2 ストレス: インスタンス障害 - 1 つ以上の EC2 インスタンスを停止して、インスタンス障害の影響を調べてください。

  現在のリージョンで、特定のタグを持つインスタンスをターゲットにします。このシナリオでは、これらのインスタンスを停止し、アクションの終了時 (デフォルトでは 5 分) に再起動します。

- EC2 ストレス: ディスク - ディスク使用率の増加が EC2 ベースのアプリケーションに及ぼす影響を調べます。

  このシナリオでは、特定のタグがアタッチされている現在のリージョンの EC2 インスタンスをターゲットにします。このシナリオでは、アクションの実行中にターゲット EC2 インスタンスに注入されるディスク使用量の増加をカスタマイズできます。デフォルトでは、ディスクストレスアクションごとに 5 分です。

- EC2 ストレス: CPU- - CPU の増加が EC2 ベースのアプリケーションに及ぼす影響を調べます。

  このシナリオでは、特定のタグがアタッチされている現在のリージョンの EC2 インスタンスをターゲットにします。このシナリオでは、ターゲット EC2 インスタンスに注入される CPU ストレス量の増加をアクションの間 (デフォルトでは各 CPU ストレスアクションにつき 5 分) カスタマイズできます。

- EC2 ストレス: メモリ -メモリ使用量の増加が EC2 ベースのアプリケーションに及ぼす影響を調べます。

  このシナリオでは、特定のタグがアタッチされている現在のリージョンの EC2 インスタンスをターゲットにします。このシナリオでは、ターゲット EC2 インスタンスに注入されるメモリストレス量の増加をアクションの間 (デフォルトではメモリストレスアクションごとに 5 分) に合わせてカスタマイズできます。

- EC2 ストレス: ネットワークレイテンシー -ネットワークレイテンシーの増加が EC2 ベースのアプリケーションに及ぼす影響を調べます。

  このシナリオでは、特定のタグがアタッチされている現在のリージョンの EC2 インスタンスをターゲットにします。このシナリオでは、ターゲット EC2 インスタンスに注入されるネットワークレイテンシーの増加を、アクションの継続時間 (デフォルトでは各レイテンシーアクションにつき 5 分) に合わせてカスタマイズできます。

AWS FIS は、次の Amazon EKS のシナリオをサポートしています。これらのシナリオは、Kubernetes アプリケーションラベルを使用する EKS ポッドをターゲットにしています。独自のラベルを使用することも、シナリオに含められているデフォルトのラベルを使用することもできます。FIS を使用した EKS の詳細については、「EKS ポッドアクションを使用する」を参照してください。

- EKS ストレス: ポッドの削除 -1 つ以上のポッドを削除して、EKS ポッドの障害による影響を調べてください。

  このシナリオでは、アプリケーションラベルに関連付けられている現在のリージョンのポッドをターゲットにします。このシナリオでは、一致したすべてのポッドを終了します。ポッドの再作成は kubernetes の設定によって制御されます。
- EKS ストレス: CPU - CPU の増加が EKS ベースのアプリケーションに及ぼす影響を調べてください。

  このシナリオでは、アプリケーションラベルに関連付けられている現在のリージョンのポッドをターゲットにします。このシナリオでは、アクションの実行中にターゲット EKS ポッドに注入される CPU ストレスの増加量をカスタマイズできます。デフォルトでは、CPU ストレスアクションごとに 5 分です。
- EKS ストレス: ディスク - ディスク使用率の増加が EKS ベースのアプリケーションに及ぼす影響を調べてください。

  このシナリオでは、アプリケーションラベルに関連付けられている現在のリージョンのポッドをターゲットにします。このシナリオでは、アクションの間、ターゲット EKS ポッドに注入されるディスクストレスの増加量をカスタマイズできます。デフォルトでは、CPU ストレスアクションごとに 5 分です。
- EKS ストレス: メモリ -メモリ使用量の増加が EKS ベースのアプリケーションに及ぼす影響を調べてください。

  このシナリオでは、アプリケーションラベルに関連付けられている現在のリージョンのポッドをターゲットにします。このシナリオでは、アクションの実行中にターゲットの EKS ポッドに注入されるメモリストレスの増加量をカスタマイズできます。デフォルトでは、各メモリストレスアクションにつき 5 分です。
- EKS ストレス: ネットワーク遅延 -ネットワーク遅延の増加が EKS ベースのアプリケーションに及ぼす影響を調べます。

  このシナリオでは、アプリケーションラベルに関連付けられている現在のリージョンのポッドをターゲットにします。このシナリオでは、ターゲット EKS ポッドに注入されるネットワークレイ

テンシーの増加を、アクションの継続時間 (デフォルトでは 1 つのレイテンシーアクションにつき 5 分) に合わせてカスタマイズできます。

AWS FIS は、マルチ AZ およびマルチリージョンアプリケーションの次のシナリオをサポートしています。これらのシナリオは、複数のリソースタイプをターゲットにしています。

- AZ Availability: Power Interruption - アベイラビリティーゾーン (AZ) の電力が完全に中断された場合に予想される症状を挿入します。AZ Availability: Power Interruption の詳細を確認してください。
- Cross-Region: Connectivity - 実験リージョンから宛先リージョンへのアプリケーションネットワークトラフィックをブロックし、クロスリージョンデータレプリケーションを一時停止します。Cross-Region: Connectivity の使用の詳細については、こちらを参照してください。

AZ Availability: Power Interruption

AZ Availability: Power Interruption のシナリオを使用して、アベイラビリティーゾーン (AZ) の電力が完全に中断されるという予想される症状を誘発できます。

このシナリオは、AZ に完全な停電が 1 回発生したときに、マルチ AZ アプリケーションが想定どおりに動作することをデモンストレーションするために使用できます。これには、ゾーンコンピューティング (Amazon EC2、EKS、ECS) の喪失、AZ でのコンピューティングの再スケーリングの禁止、サブネット接続の喪失、RDS フェイルオーバー、 ElastiCache フェイルオーバー、応答しない EBS ボリュームが含まれます。デフォルトで、ターゲットが見つからないアクションはスキップされます。

アクション

次のアクションにより、単一の AZ で完全な停電が発生したときに予想される症状の多くを作成できます。AZ アベイラビリティー: 停電は、単一の AZ の停電中に影響が予想されるサービスにのみ影響します。デフォルトで、このシナリオでは 30 分間の停電症状を挿入し、さらに 30 分間、復旧中に発生する可能性のある症状を挿入します。

Stop-Instances

AZ の電源が中断されている間、影響を受けた AZ の EC2 インスタンスはシャットダウンします。電力が復旧すると、インスタンスは再起動します。AZ Availability: Power Interruption には aws:ec2:stop-instances が含まれ、中断の期間に影響を受けた AZ のすべてのインスタンスを停止します。この期間の後、インスタンスは再起動されます。Amazon EKS によって管理されている EC2

インスタンスを停止すると、依存する EKS ポッドが削除されます。Amazon ECS によって管理されている EC2 インスタンスを停止すると、依存する ECS タスクが停止します。

このアクションは、影響を受けた AZ で実行されている EC2 インスタンスをターゲットにしています。デフォルトで、`StopInstances` の値を持つ `AzImpairmentPower` という名前のタグのインスタンスがターゲットになります。このタグをインスタンスに追加することも、デフォルトタグを実験テンプレートの独自のタグに置き換えることもできます。デフォルトで、有効なインスタンスが見つからない場合、このアクションはスキップされます。

Stop-ASG-Instances

AZ の電源が中断されている間、影響を受けた AZ で Auto Scaling グループにより管理されている EC2 インスタンスはシャットダウンします。電力が復旧すると、インスタンスは再起動します。AZ Availability: Power Interruption には aws:ec2:stop-instances が含まれ、自動スケーリングにより管理されているものを含め、中断の期間に影響を受けた AZ のすべてのインスタンスを停止します。この期間の後、インスタンスは再起動されます。

このアクションは、影響を受けた AZ で実行されている EC2 インスタンスをターゲットにしています。デフォルトで、`IceAsg` の値を持つ `AzImpairmentPower` という名前のタグのインスタンスがターゲットになります。このタグをインスタンスに追加することも、デフォルトタグを実験テンプレートの独自のタグに置き換えることもできます。デフォルトで、有効なインスタンスが見つからない場合、このアクションはスキップされます。

インスタンスの起動を一時停止する

AZ の電源が中断されている間、AZ で容量をプロビジョニングするための EC2 API 呼び出しは失敗します。特に、`ec2:StartInstances`、`ec2:CreateFleet`、および `ec2:RunInstances` の API が影響を受けます。AZ Availability: Power Interruption includes には、aws:ec2:api-insufficient-instance-capacity-error が含まれ、影響を受けた AZ に新しいインスタンスがプロビジョニングされるのを回避します。

このアクションは、インスタンスのプロビジョニングに使用される IAM ロールをターゲットにしています。これらは ARN を使用してターゲットにする必要があります。デフォルトで、有効な IAM ロールが見つからない場合、このアクションはスキップされます。

ASG スケーリングを一時停止する

AZ の電源が中断されている間、自動スケーリングコントロールプレーンが AZ で失われた容量を回復するために行った EC2 API 呼び出しは失敗します。特に、`ec2:StartInstances`、`ec2:CreateFleet`、および `ec2:RunInstances` の API が影響を

受けます。AZ Availability: Power Interruption には、aws:ec2:asg-insufficient-instance-capacity-error が含まれ、影響を受けた AZ に新しいインスタンスがプロビジョニングされるのを回避します。これにより、Amazon EKS と Amazon ECS が影響を受けた AZ でスケーリングされることも回避されます。

このアクションは Auto Scaling グループをターゲットにしています。デフォルトで、`IceAsg` の値を持つ `AzImpairmentPower` という名前のタグの Auto Scaling グループがターゲットになります。このタグを Auto Scaling グループに追加するか、またはデフォルトタグを実験テンプレートの独自のタグに置き換えることができます。デフォルトで、有効な Auto Scaling グループが見つからない場合、このアクションはスキップされます。

ネットワーク接続を一時停止する

AZ の電源が中断されている間、AZ のネットワークは利用できなくなります。このような状況の場合、一部の AWS サービスでは、影響を受けた AZ のプライベートエンドポイントが利用できないことを反映し、DNS を更新するのに数分かかることがあります。この間、DNS ルックアップはアクセスできない IP アドレスを返すことがあります。AZ Availability: Power Interruption には、aws:network:disrupt-connectivity が含まれ、影響を受けた AZ のすべてのサブネットのネットワーク接続すべてが 2 分間ブロックされます。これにより、ほとんどのアプリケーションでタイムアウトと DNS 更新が強制されます。2 分後にアクションが終了し、AZ が利用できない状態が継続しながら、後続のリージョン別のサービス DNS の復旧が許可されます。

このアクションはサブネットをターゲットにしています。デフォルトで、`DisruptSubnet` の値を持つ `AzImpairmentPower` という名前のタグのクラスターがターゲットになります。このタグをサブネットに追加するか、またはデフォルトタグを実験テンプレートの独自のタグに置き換えることができます。デフォルトで、有効なサブネットが見つからない場合、このアクションはスキップされます。

RDS のフェイルオーバー

AZ の電源が中断されている間、影響を受けた AZ の RDS ノードはシャットダウンします。影響を受けた AZ の単一 AZ RDS ノードは完全に利用できなくなります。マルチ AZ クラスターの場合、ライターノードは影響を受けていない AZ にフェイルオーバーし、影響を受けた AZ のリーダーノードは利用できなくなります。マルチ AZ クラスターの場合、AZ Availability: Power Interruption に aws:rds:failover-db-cluster が含まれ、ライターが影響を受けた AZ に存在する場合はフェイルオーバーします。

このアクションは RDS クラスターをターゲットにしています。デフォルトでは、`DisruptRds` の値を持つ `AzImpairmentPower` という名前のタグのクラスターがターゲットになります。このタグを

クラスターに追加するか、またはデフォルトタグを実験テンプレートの独自のタグに置き換えることができます。デフォルトで、有効なクラスターが見つからない場合、このアクションはスキップされます。

Redis ElastiCache を一時停止する

AZ の電源の中断中、AZ 内の ElastiCache ノードは使用できなくなります。 AZ Availability: Power Interruptionには、影響を受ける AZ 内の ElastiCache ノードを終了するための aws:elasticache:interrupt-cluster-az-power が含まれています。中断の期間中は、影響を受けた AZ に新しいインスタンスがプロビジョニングされないため、クラスターは少ない容量のままになります。

このアクションは ElastiCache クラスターをターゲットにします。デフォルトで、`ElasticacheImpact` の値を持つ `AzImpairmentPower` という名前のタグのクラスターがターゲットになります。このタグをクラスターに追加するか、またはデフォルトタグを実験テンプレートの独自のタグに置き換えることができます。デフォルトで、有効なクラスターが見つからない場合、このアクションはスキップされます。影響を受けた AZ にライターノードが存在するクラスターのみが有効ターゲットと見なされることに注意してください。

EBS I/O を一時停止する

AZ の電源中断の後、電力が復旧したとき、ごく一部のインスタンスで EBS ボリュームが応答しなくなることがあります。AZ Availability: Power Interruption には aws:ebs:pause-io が含まれ、1 つの EBS ボリュームは応答しない状態のままになります。

デフォルトで、インスタンスの終了後も維持するように設定されたボリュームのみがターゲットになります。このアクションは、値が `APIPauseVolume` の `AzImpairmentPower` という名前のタグがあるボリュームをターゲットにしています。このタグをボリュームに追加するか、またはデフォルトタグを実験テンプレートの独自のタグに置き換えることができます。デフォルトで、有効なボリュームが見つからない場合、このアクションはスキップされます。

制限事項

- このシナリオに停止条件は含まれていません。アプリケーションに適した停止条件を実験テンプレートに追加する必要があります。
- AWS Fargate で実行されている Amazon EKS ポッドはサポートされていません。
- AWS Fargate で実行されている Amazon ECS タスクはサポートされていません。
- 読み取り可能なスタンバイ DB インスタンスが 2 つある Amazon RDS マルチ AZ はサポートされていません。この場合、インスタンスは終了し、RDS はフェイルオーバーされ、容量は直ちに影

響を受けた AZ にプロビジョニングされます。影響を受けた AZ の読み取り可能なスタンバイは引き続き利用できます。

要件

- AWS FIS 実験ロールに必要なアクセス許可を追加します。
- リソースタグは、実験のターゲットとなるリソースに適用する必要があります。独自のタグ付け規則を使用することも、シナリオで定義したデフォルトタグを使用することもできます。

アクセス許可

次のポリシーは、AZ Availability: Power Interruption シナリオを使用して実験を実行するために必要なアクセス許可を AWS FIS に付与しています。このポリシーを実験ロールにアタッチする必要があります。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowFISExperimentLoggingActionsCloudwatch",
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogDelivery",
                "logs:PutResourcePolicy",
                "logs:DescribeResourcePolicies",
                "logs:DescribeLogGroups"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": "ec2:CreateTags",
            "Resource": "arn:aws:ec2:*:*:network-acl/*",
            "Condition": {
                "StringEquals": {
                    "ec2:CreateAction": "CreateNetworkAcl",
                    "aws:RequestTag/managedByFIS": "true"
                }
            }
        },
```

```
        {
            "Effect": "Allow",
            "Action": "ec2:CreateNetworkAcl",
            "Resource": "arn:aws:ec2:*:*:network-acl/*",
            "Condition": {
                "StringEquals": {
                    "aws:RequestTag/managedByFIS": "true"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:CreateNetworkAclEntry",
                "ec2:DeleteNetworkAcl"
            ],
            "Resource": [
                "arn:aws:ec2:*:*:network-acl/*",
                "arn:aws:ec2:*:*:vpc/*"
            ],
            "Condition": {
                "StringEquals": {
                    "ec2:ResourceTag/managedByFIS": "true"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": "ec2:CreateNetworkAcl",
            "Resource": "arn:aws:ec2:*:*:vpc/*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:DescribeVpcs",
                "ec2:DescribeManagedPrefixLists",
                "ec2:DescribeSubnets",
                "ec2:DescribeNetworkAcls"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": "ec2:ReplaceNetworkAclAssociation",
```

```
            "Resource": [
                "arn:aws:ec2:*:*:subnet/*",
                "arn:aws:ec2:*:*:network-acl/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "rds:FailoverDBCluster"
            ],
            "Resource": [
                "arn:aws:rds:*:*:cluster:*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "rds:RebootDBInstance"
            ],
            "Resource": [
                "arn:aws:rds:*:*:db:*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "elasticache:DescribeReplicationGroups",
                "elasticache:InterruptClusterAzPower"
            ],
            "Resource": [
                "arn:aws:elasticache:*:*:replicationgroup:*"
            ]
        },
        {
            "Sid": "TargetResolutionByTags",
            "Effect": "Allow",
            "Action": [
                "tag:GetResources"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
```

```
                "ec2:StartInstances",
                "ec2:StopInstances"
            ],
            "Resource": "arn:aws:ec2:*:*:instance/*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:DescribeInstances"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "kms:CreateGrant"
            ],
            "Resource": [
                "arn:aws:kms:*:*:key/*"
            ],
            "Condition": {
                "StringLike": {
                    "kms:ViaService": "ec2.*.amazonaws.com"
                },
                "Bool": {
                    "kms:GrantIsForAWSResource": "true"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:DescribeVolumes"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:PauseVolumeIO"
            ],
            "Resource": "arn:aws:ec2:*:*:volume/*"
        },
        {
```

```
            "Sid": "AllowInjectAPI",
            "Effect": "Allow",
            "Action": [
                "ec2:InjectApiError"
            ],
            "Resource": [
                "*"
            ],
            "Condition": {
                "ForAnyValue:StringEquals": {
                    "ec2:FisActionId": [
                        "aws:ec2:api-insufficient-instance-capacity-error",
                        "aws:ec2:asg-insufficient-instance-capacity-error"
                    ]
                }
            }
        },
        {
            "Sid": "DescribeAsg",
            "Effect": "Allow",
            "Action": [
                "autoscaling:DescribeAutoScalingGroups"
            ],
            "Resource": [
                "*"
            ]
        }
    ]
}
```

シナリオのコンテンツ

次のコンテンツはシナリオを定義しています。この JSON を保存し、AWS コマンドラインインターフェイス (AWS CLI) から create-experiment-template コマンドを使用して実験テンプレートを作成するのに使用できます。シナリオの最新バージョンについては、FIS コンソールのシナリオライブラリを参照してください。

```
{
    "targets": {
        "IAM-role": {
            "resourceType": "aws:iam:role",
            "resourceArns": [],
```

```
            "selectionMode": "ALL"
        },
        "EBS-Volumes": {
            "resourceType": "aws:ec2:ebs-volume",
            "resourceTags": {
                "AzImpairmentPower": "ApiPauseVolume"
            },
            "selectionMode": "COUNT(1)",
            "parameters": {
                "availabilityZoneIdentifier": "us-east-1a"
            },
            "filters": [
                {
                    "path": "Attachments.DeleteOnTermination",
                    "values": [
                        "false"
                    ]
                }
            ]
        },
        "EC2-Instances": {
            "resourceType": "aws:ec2:instance",
            "resourceTags": {
                "AzImpairmentPower": "StopInstances"
            },
            "filters": [
                {
                    "path": "State.Name",
                    "values": [
                        "running"
                    ]
                },
                {
                    "path": "Placement.AvailabilityZone",
                    "values": [
                        "us-east-1a"
                    ]
                }
            ],
            "selectionMode": "ALL"
        },
        "ASG": {
            "resourceType": "aws:ec2:autoscaling-group",
            "resourceTags": {
```

```
                "AzImpairmentPower": "IceAsg"
            },
            "selectionMode": "ALL"
        },
        "ASG-EC2-Instances": {
            "resourceType": "aws:ec2:instance",
            "resourceTags": {
                "AzImpairmentPower": "IceAsg"
            },
            "filters": [
                {
                    "path": "State.Name",
                    "values": [
                        "running"
                    ]
                },
                {
                    "path": "Placement.AvailabilityZone",
                    "values": [
                        "us-east-1a"
                    ]
                }
            ],
            "selectionMode": "ALL"
        },
        "Subnet": {
            "resourceType": "aws:ec2:subnet",
            "resourceTags": {
                "AzImpairmentPower": "DisruptSubnet"
            },
            "filters": [
                {
                    "path": "AvailabilityZone",
                    "values": [
                        "us-east-1a"
                    ]
                }
            ],
            "selectionMode": "ALL",
            "parameters": {}
        },
        "RDS-Cluster": {
            "resourceType": "aws:rds:cluster",
            "resourceTags": {
```

```
                "AzImpairmentPower": "DisruptRds"
            },
            "selectionMode": "ALL",
            "parameters": {
                "writerAvailabilityZoneIdentifiers": "us-east-1a"
            }
        },
        "ElastiCache-Cluster": {
            "resourceType": "aws:elasticache:redis-replicationgroup",
            "resourceTags": {
                "AzImpairmentPower": "DisruptElasticache"
            },
            "selectionMode": "ALL",
            "parameters": {
                "availabilityZoneIdentifier": "us-east-1a"
            }
        }
    },
    "actions": {
        "Pause-Instance-Launches": {
            "actionId": "aws:ec2:api-insufficient-instance-capacity-error",
            "parameters": {
                "availabilityZoneIdentifiers": "us-east-1a",
                "duration": "PT30M",
                "percentage": "100"
            },
            "targets": {
                "Roles": "IAM-role"
            }
        },
        "Pause-EBS-IO": {
            "actionId": "aws:ebs:pause-volume-io",
            "parameters": {
                "duration": "PT30M"
            },
            "targets": {
                "Volumes": "EBS-Volumes"
            },
            "startAfter": [
                "Stop-Instances",
                "Stop-ASG-Instances"
            ]
        },
        "Stop-Instances": {
```

```
            "actionId": "aws:ec2:stop-instances",
            "parameters": {
                "completeIfInstancesTerminated": "true",
                "startInstancesAfterDuration": "PT30M"
            },
            "targets": {
                "Instances": "EC2-Instances"
            }
        },
        "Pause-ASG-Scaling": {
            "actionId": "aws:ec2:asg-insufficient-instance-capacity-error",
            "parameters": {
                "availabilityZoneIdentifiers": "us-east-1a",
                "duration": "PT30M",
                "percentage": "100"
            },
            "targets": {
                "AutoScalingGroups": "ASG"
            }
        },
        "Stop-ASG-Instances": {
            "actionId": "aws:ec2:stop-instances",
            "parameters": {
                "completeIfInstancesTerminated": "true",
                "startInstancesAfterDuration": "PT30M"
            },
            "targets": {
                "Instances": "ASG-EC2-Instances"
            }
        },
        "Pause-network-connectivity": {
            "actionId": "aws:network:disrupt-connectivity",
            "parameters": {
                "duration": "PT2M",
                "scope": "all"
            },
            "targets": {
                "Subnets": "Subnet"
            }
        },
        "Failover-RDS": {
            "actionId": "aws:rds:failover-db-cluster",
            "parameters": {},
            "targets": {
```

```
                "Clusters": "RDS-Cluster"
            }
        },
        "Pause-ElastiCache": {
            "actionId": "aws:elasticache:interrupt-cluster-az-power",
            "parameters": {
                "duration": "PT30M"
            },
            "targets": {
                "ReplicationGroups": "ElastiCache-Cluster"
            }
        }
    },
    "stopConditions": [
        {
            "source": "aws:cloudwatch:alarm",
            "value": ""
        }
    ],
    "roleArn": "",
    "tags": {
        "Name": "AZ Impairment: Power Interruption"
    },
    "logConfiguration": {
        "logSchemaVersion": 2
    },
    "experimentOptions": {
        "accountTargeting": "single-account",
        "emptyTargetResolutionMode": "skip"
    },
    "description": "Affect multiple resource types in a single AZ, targeting by tags
 and explicit ARNs, to approximate power interruption in one AZ."
}
```

Cross-Region: Connectivity

Cross-Region: Connectivity のシナリオを使用して、実験リージョンから宛先リージョンへのアプリケーションネットワークトラフィックをブロックし、Amazon S3 と Amazon DynamoDB のクロスリージョンレプリケーションを一時停止することができます。クロスリージョン: 接続は、実験を実行するリージョン (実験リージョン) からのアウトバウンドアプリケーショントラフィックに影響します。実験リージョン (宛先リージョン) から分離するリージョンからのステートレスインバウンド

トラフィックはブロックされないことがあります。AWS マネージドサービスからのトラフィックはブロックされないことがあります。

このシナリオを使用して、宛先リージョンのリソースが実験リージョンからアクセスできない場合でも、マルチリージョンアプリケーションが想定どおりに動作することをデモンストレーションできます。これには、トランジットゲートウェイとルートテーブルをターゲットにすることにより、実験リージョンから宛先リージョンへのネットワークトラフィックをブロックすることが含まれます。また、S3 と DynamoDB のクロスリージョンレプリケーションも一時停止します。デフォルトで、ターゲットが見つからないアクションはスキップされます。

アクション

次のアクションにより、含まれている AWS のサービスのクロスリージョン接続をブロックします。アクションは並列で実行されます。デフォルトで、シナリオは 3 時間トラフィックをブロックしますが、最大 12 時間まで増やすことができます。

Transit Gateway の接続を中断する

Cross Region: Connectivity には aws:network:transit-gateway-disrupt-cross-region-connectivity が含まれ、実験リージョンの VPC からトランジットゲートウェイで接続された宛先リージョンの VPC へのクロスリージョンネットワークトラフィックをブロックします。これは、実験リージョン内の VPC エンドポイントへのアクセスには影響しませんが、実験リージョンから宛先リージョンの VPC エンドポイント宛てのトラフィックはブロックします。

このアクションは、実験リージョンと宛先リージョンを接続するトランジットゲートウェイをターゲットにしています。デフォルトで、`Allowed` の値を持つ `DisruptTransitGateway` という名前のタグのトランジットゲートウェイがターゲットになります。このタグをトランジットゲートウェイに追加するか、またはデフォルトタグを実験テンプレートの独自のタグに置き換えることができます。デフォルトで、有効なトランジットゲートウェイが見つからない場合、このアクションはスキップされます。

サブネット接続を中断する

Cross Region: Connectivity には aws:network:route-table-disrupt-cross-region-connectivity が含まれ、実験リージョンの VPC から宛先リージョンのパブリック AWS IP ブロックへのクロスリージョンネットワークトラフィックをブロックします。これらのパブリック IP ブロックには、宛先リージョンの AWS のサービスエンドポイント (S3 リージョナルエンドポイントなど) と、マネージドサービスの AWS IP ブロック (ロードバランサーや Amazon API Gateway に使用される IP アドレスなど) が含まれます。このアクションは、実験リージョンから宛先リージョンへのクロスリージョン VPC ピアリング接続を介したネットワーク接続もブロックします。これは、実験リージョンの VPC

エンドポイントへのアクセスには影響しませんが、実験リージョンから宛先リージョンの VPC エンドポイント宛てのトラフィックはブロックします。

このアクションは、実験リージョンのサブネットをターゲットにしています。デフォルトで、`Allowed` の値を持つ `DisruptSubnet` という名前のタグのサブネットがターゲットになります。このタグをサブネットに追加するか、またはデフォルトタグを実験テンプレートの独自のタグに置き換えることができます。デフォルトで、有効なサブネットが見つからない場合、このアクションはスキップされます。

S3 レプリケーションを一時停止する

Cross Region: Connectivity には aws:s3:bucket-pause-replication が含まれ、実験リージョンからターゲットバケットの宛先リージョンへの S3 レプリケーションを一時停止します。宛先リージョンから実験リージョンへのレプリケーションには影響ありません。シナリオが終了した後、バケットのレプリケーションは一時停止した時点から再開されます。レプリケーションですべてのオブジェクトの同期を保つまでにかかる時間は、実験の期間と、バケットへのオブジェクトのアップロード速度に応じて異なることに注意してください。

このアクションは、宛先リージョンの S3 バケットにクロスリージョンレプリケーション (CRR) が有効になっている実験リージョンの S3 バケットをターゲットにしています。デフォルトで、`Allowed` の値を持つ `DisruptS3` という名前のタグのバケットがターゲットになります。このタグをバケットに追加するか、またはデフォルトタグを実験テンプレートの独自のタグに置き換えることができます。デフォルトで、有効なバケットが見つからない場合、このアクションはスキップされます。

DynamoDB レプリケーションを一時停止する

Cross-Region: Connectivity には、aws:dynamodb:global-table-pause-replication が含まれており、実験リージョンと、送信先リージョンを含む他のすべてのリージョン間のレプリケーションを一時停止します。これにより、実験リージョンとのレプリケーションは回避されますが、他のリージョン間のレプリケーションには影響しません。シナリオが終了した後、テーブルのレプリケーションは一時停止した時点から再開されます。レプリケーションがすべてのデータを同期させるのにかかる時間は、実験の期間とテーブルの変更レートによって異なることに注意してください。

このアクションは、実験リージョンの DynamoDB グローバルテーブルをターゲットにします。デフォルトで、`Allowed` の値を持つ `DisruptDynamoDb` という名前のタグのテーブルがターゲットになります。このタグをテーブルに追加することも、デフォルトタグを実験テンプレートの独自のタグに置き換えることもできます。デフォルトで、有効なグローバルテーブルが見つからない場合、このアクションはスキップされます。

制限事項

- このシナリオに停止条件は含まれていません。アプリケーションに適した停止条件を実験テンプレートに追加する必要があります。

要件

- AWS FIS 実験ロールに必要なアクセス許可を追加します。
- リソースタグは、実験のターゲットとなるリソースに適用する必要があります。独自のタグ付け規則を使用することも、シナリオで定義したデフォルトタグを使用することもできます。

アクセス許可

次のポリシーは、Cross-Region: Connectivity シナリオを使用して実験を実行するために必要なアクセス許可を AWS FIS に付与しています。このポリシーを実験ロールにアタッチする必要があります。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "RouteTableDisruptConnectivity1",
            "Effect": "Allow",
            "Action": "ec2:CreateRouteTable",
            "Resource": "arn:aws:ec2:*:*:route-table/*",
            "Condition": {
                "StringEquals": {
                    "aws:RequestTag/managedByFIS": "true"
                }
            }
        },
        {
            "Sid": "RouteTableDisruptConnectivity2",
            "Effect": "Allow",
            "Action": "ec2:CreateRouteTable",
            "Resource": "arn:aws:ec2:*:*:vpc/*"
        },
        {
            "Sid": "RouteTableDisruptConnectivity21",
            "Effect": "Allow",
```

```
            "Action": "ec2:CreateTags",
            "Resource": "arn:aws:ec2:*:*:route-table/*",
            "Condition": {
                "StringEquals": {
                    "ec2:CreateAction": "CreateRouteTable",
                    "aws:RequestTag/managedByFIS": "true"
                }
            }
        },
        {
            "Sid": "RouteTableDisruptConnectivity3",
            "Effect": "Allow",
            "Action": "ec2:CreateTags",
            "Resource": "arn:aws:ec2:*:*:network-interface/*",
            "Condition": {
                "StringEquals": {
                    "ec2:CreateAction": "CreateNetworkInterface",
                    "aws:RequestTag/managedByFIS": "true"
                }
            }
        },
        {
            "Sid": "RouteTableDisruptConnectivity4",
            "Effect": "Allow",
            "Action": "ec2:CreateTags",
            "Resource": "arn:aws:ec2:*:*:prefix-list/*",
            "Condition": {
                "StringEquals": {
                    "ec2:CreateAction": "CreateManagedPrefixList",
                    "aws:RequestTag/managedByFIS": "true"
                }
            }
        },
        {
            "Sid": "RouteTableDisruptConnectivity5",
            "Effect": "Allow",
            "Action": "ec2:DeleteRouteTable",
            "Resource": [
                "arn:aws:ec2:*:*:route-table/*",
                "arn:aws:ec2:*:*:vpc/*"
            ],
            "Condition": {
                "StringEquals": {
                    "ec2:ResourceTag/managedByFIS": "true"
```

```
                }
            }
        },
        {
            "Sid": "RouteTableDisruptConnectivity6",
            "Effect": "Allow",
            "Action": "ec2:CreateRoute",
            "Resource": "arn:aws:ec2:*:*:route-table/*",
            "Condition": {
                "StringEquals": {
                    "ec2:ResourceTag/managedByFIS": "true"
                }
            }
        },
        {
            "Sid": "RouteTableDisruptConnectivity7",
            "Effect": "Allow",
            "Action": "ec2:CreateNetworkInterface",
            "Resource": "arn:aws:ec2:*:*:network-interface/*",
            "Condition": {
                "StringEquals": {
                    "aws:RequestTag/managedByFIS": "true"
                }
            }
        },
        {
            "Sid": "RouteTableDisruptConnectivity8",
            "Effect": "Allow",
            "Action": "ec2:CreateNetworkInterface",
            "Resource": [
                "arn:aws:ec2:*:*:subnet/*",
                "arn:aws:ec2:*:*:security-group/*"
            ]
        },
        {
            "Sid": "RouteTableDisruptConnectivity9",
            "Effect": "Allow",
            "Action": "ec2:DeleteNetworkInterface",
            "Resource": "arn:aws:ec2:*:*:network-interface/*",
            "Condition": {
                "StringEquals": {
                    "ec2:ResourceTag/managedByFIS": "true"
                }
            }
```

```
        },
        {
            "Sid": "RouteTableDisruptConnectivity10",
            "Effect": "Allow",
            "Action": "ec2:CreateManagedPrefixList",
            "Resource": "arn:aws:ec2:*:*:prefix-list/*",
            "Condition": {
                "StringEquals": {
                    "aws:RequestTag/managedByFIS": "true"
                }
            }
        },
        {
            "Sid": "RouteTableDisruptConnectivity11",
            "Effect": "Allow",
            "Action": "ec2:DeleteManagedPrefixList",
            "Resource": "arn:aws:ec2:*:*:prefix-list/*",
            "Condition": {
                "StringEquals": {
                    "ec2:ResourceTag/managedByFIS": "true"
                }
            }
        },
        {
            "Sid": "RouteTableDisruptConnectivity12",
            "Effect": "Allow",
            "Action": "ec2:ModifyManagedPrefixList",
            "Resource": "arn:aws:ec2:*:*:prefix-list/*",
            "Condition": {
                "StringEquals": {
                    "ec2:ResourceTag/managedByFIS": "true"
                }
            }
        },
        {
            "Sid": "RouteTableDisruptConnectivity13",
            "Effect": "Allow",
            "Action": [
                "ec2:DescribeNetworkInterfaces",
                "ec2:DescribeVpcs",
                "ec2:DescribeVpcPeeringConnections",
                "ec2:DescribeManagedPrefixLists",
                "ec2:DescribeSubnets",
                "ec2:DescribeRouteTables",
```

```
                "ec2:DescribeVpcEndpoints"
            ],
            "Resource": "*"
        },
        {
            "Sid": "RouteTableDisruptConnectivity14",
            "Effect": "Allow",
            "Action": "ec2:ReplaceRouteTableAssociation",
            "Resource": [
                "arn:aws:ec2:*:*:subnet/*",
                "arn:aws:ec2:*:*:route-table/*"
            ]
        },
        {
            "Sid": "RouteTableDisruptConnectivity15",
            "Effect": "Allow",
            "Action": "ec2:GetManagedPrefixListEntries",
            "Resource": "arn:aws:ec2:*:*:prefix-list/*"
        },
        {
            "Sid": "RouteTableDisruptConnectivity16",
            "Effect": "Allow",
            "Action": "ec2:AssociateRouteTable",
            "Resource": [
                "arn:aws:ec2:*:*:subnet/*",
                "arn:aws:ec2:*:*:route-table/*"
            ]
        },
        {
            "Sid": "RouteTableDisruptConnectivity17",
            "Effect": "Allow",
            "Action": "ec2:DisassociateRouteTable",
            "Resource": [
                "arn:aws:ec2:*:*:route-table/*"
            ],
            "Condition": {
                "StringEquals": {
                    "ec2:ResourceTag/managedByFIS": "true"
                }
            }
        },
        {
            "Sid": "RouteTableDisruptConnectivity18",
            "Effect": "Allow",
```

```
            "Action": "ec2:DisassociateRouteTable",
            "Resource": [
                "arn:aws:ec2:*:*:subnet/*"
            ]
        },
        {
            "Sid": "RouteTableDisruptConnectivity19",
            "Effect": "Allow",
            "Action": "ec2:ModifyVpcEndpoint",
            "Resource": [
                "arn:aws:ec2:*:*:route-table/*"
            ],
            "Condition": {
                "StringEquals": {
                    "ec2:ResourceTag/managedByFIS": "true"
                }
            }
        },
        {
            "Sid": "RouteTableDisruptConnectivity20",
            "Effect": "Allow",
            "Action": "ec2:ModifyVpcEndpoint",
            "Resource": [
                "arn:aws:ec2:*:*:vpc-endpoint/*"
            ]
        },
        {
            "Sid": "TransitGatewayDisruptConnectivity1",
            "Effect": "Allow",
            "Action": [
                "ec2:DisassociateTransitGatewayRouteTable",
                "ec2:AssociateTransitGatewayRouteTable"
            ],
            "Resource": [
                "arn:aws:ec2:*:*:transit-gateway-route-table/*",
                "arn:aws:ec2:*:*:transit-gateway-attachment/*"
            ]
        },
        {
            "Sid": "TransitGatewayDisruptConnectivity2",
            "Effect": "Allow",
            "Action": [
                "ec2:DescribeTransitGatewayPeeringAttachments",
                "ec2:DescribeTransitGatewayAttachments",
```

```
                "ec2:DescribeTransitGateways"
            ],
            "Resource": "*"
        },
        {
            "Sid": "S3CrossRegion1",
            "Effect": "Allow",
            "Action": [
                "s3:ListAllMyBuckets"
            ],
            "Resource": "*"
        },
        {
            "Sid": "S3CrossRegion2",
            "Effect": "Allow",
            "Action": [
                "tag:GetResources"
            ],
            "Resource": "*"
        },
        {
            "Sid": "S3CrossRegion3",
            "Effect": "Allow",
            "Action": [
                "s3:PauseReplication"
            ],
            "Resource": "arn:aws:s3:::*",
            "Condition": {
                "StringLike": {
                    "s3:DestinationRegion": "*"
                }
            }
        },
        {
            "Sid": "S3CrossRegion4",
            "Effect": "Allow",
            "Action": [
                "s3:GetReplicationConfiguration",
                "s3:PutReplicationConfiguration"
            ],
            "Resource": "arn:aws:s3:::*",
            "Condition": {
                "BoolIfExists": {
                    "s3:isReplicationPauseRequest": "true"
```

```
                }
            }
        },
        {
            "Sid": "DdbCrossRegion1",
            "Effect": "Allow",
            "Action": [
                "tag:GetResources"
            ],
            "Resource": "*"
        },
        {
            "Sid": "DdbCrossRegion2",
            "Effect": "Allow",
            "Action": [
                "dynamodb:DescribeTable",
                "dynamodb:DescribeGlobalTable"
            ],
            "Resource": [
                "arn:aws:dynamodb:*:*:table/*",
                "arn:aws:dynamodb:*:*:global-table/*"
            ]
        },
        {
            "Sid": "DdbCrossRegion3",
            "Effect": "Allow",
            "Action": [
                "kms:DescribeKey",
                "kms:GetKeyPolicy",
                "kms:PutKeyPolicy"
            ],
            "Resource": "arn:aws:kms:*:*:key/*"
        }
    ]
}
```

シナリオのコンテンツ

次のコンテンツはシナリオを定義しています。この JSON を保存し、AWS コマンドラインインターフェイス (AWS CLI) から create-experiment-template コマンドを使用して実験テンプレートを作成するのに使用できます。シナリオの最新バージョンについては、FIS コンソールのシナリオライブラリを参照してください。

```
{
        "targets": {
                "Transit-Gateway": {
                        "resourceType": "aws:ec2:transit-gateway",
                        "resourceTags": {
                                "TgwTag": "TgwValue"
                        },
                        "selectionMode": "ALL"
                },
                "Subnet": {
                        "resourceType": "aws:ec2:subnet",
                        "resourceTags": {
                                "SubnetKey": "SubnetValue"
                        },
                        "selectionMode": "ALL",
                        "parameters": {}
                },
                "S3-Bucket": {
                        "resourceType": "aws:s3:bucket",
                        "resourceTags": {
                                "S3Impact": "Allowed"
                        },
                        "selectionMode": "ALL"
                },
                "DynamoDB-Global-Table": {
                        "resourceType": "aws:dynamodb:encrypted-global-table",
                        "resourceTags": {
                                "DisruptDynamoDb": "Allowed"
                        },
                        "selectionMode": "ALL"
                }
        },
        "actions": {
                "Disrupt-Transit-Gateway-Connectivity": {
                        "actionId": "aws:network:transit-gateway-disrupt-cross-region-
connectivity",
                        "parameters": {
                                "duration": "PT3H",
                                "region": "eu-west-1"
                        },
                        "targets": {
                                "TransitGateways": "Transit-Gateway"
                        }
```

```
                },
                "Disrupt-Subnet-Connectivity": {
                        "actionId": "aws:network:route-table-disrupt-cross-region-
connectivity",
                        "parameters": {
                                "duration": "PT3H",
                                "region": "eu-west-1"
                        },
                        "targets": {
                                "Subnets": "Subnet"
                        }
                },
                "Pause-S3-Replication": {
                        "actionId": "aws:s3:bucket-pause-replication",
                        "parameters": {
                                "duration": "PT3H",
                                "region": "eu-west-1"
                        },
                        "targets": {
                                "Buckets": "S3-Bucket"
                        }
                },
                "Pause-DynamoDB-Replication": {
                        "actionId": "aws:dynamodb:encrypted-global-table-pause-
replication",
                        "parameters": {
                                "duration": "PT3H"
                        },
                        "targets": {
                                "Tables": "DynamoDB-Global-Table"
                        }
                }
        },
        "stopConditions": [
                {
                        "source": "none"
                }
        ],
        "roleArn": "",
        "logConfiguration": {
                "logSchemaVersion": 2
        },
        "tags": {
                "Name": "Cross-Region: Connectivity"
```

```
        },
        "experimentOptions": {
                "accountTargeting": "single-account",
                "emptyTargetResolutionMode": "skip"
        },
        "description": "Block application network traffic from experiment Region to
 target Region and pause cross-Region replication"
}
```

AWS FIS の実験

AWS FIS を使用すると、 AWS ワークロードでフォールトインジェクション実験を実行できます。開始するには、実験テンプレートを作成します。実験テンプレートを作成したら、実験を開始するために使用できます。

以下のいずれかが発生すると、実験が終了します。

- すべて行動テンプレートが正常に完了しました。
- ある停止条件がトリガーされます。
- エラーのため、アクションを完了できません。例えば、ターゲットが見つかりません。
- 実験は手動で停止しました。

停止または失敗した実験は再開できません。また、完了した実験を再実行することはできません。ただし、同じ実験テンプレートから新しい実験を開始することはできます。必要に応じて、新しい実験で再度指定する前に、実験テンプレートを更新できます。

タスク

- 実験を開始する
- 実験を表示します。
- 実験にタグを付けるには
- 実験を中止する
- 解決済みターゲットを一覧表示する

実験を開始する

実験は、実験テンプレートから開始します。詳細については、「テンプレートから実験を開始する」を参照してください。

Amazon EventBridgeを使用して、1 回限りのタスクまたは定期的なタスクとして実験をスケジュールできます。詳細については、「チュートリアル: 定期的な実験をスケジュールする」を参照してください。

以下の機能を使用して実験を監視することができます。

- AWS FIS コンソールで実験を表示します。詳細については、「実験を表示します。」を参照してください。
- 実験内のターゲットリソースの Amazon CloudWatch メトリクスを表示するか、 AWS FIS 使用状況メトリクスを表示します。詳細については、「CloudWatch を使用したモニタリング」を参照してください。
- 実験ロギングを有効にすると、テストの実行時にテストに関する詳細情報を取得できます。詳細については、実験ロギングを参照してください。

実験を表示します。

実行中の実験の進行状況を表示したり、完了、停止、または失敗した実験を表示できます。

テストを停止、完了、失敗したテストは 120 日後にアカウントから自動的に削除されます。

コンソールを使用して実験を表示するには

1. https://console.aws.amazon.com/fis/ で AWS FIS コンソールを開きます。
2. ナビゲーションペインで [実験] を選択します。
3. 実験の実験 ID を選択して、詳細ページを開きます。
4. 次の 1 つ以上の操作を行います。
   - [詳細]、[状態] で実験の状態を確認してください。
   - [アクション] タブを選択すると、実験アクションに関する情報が表示されます。
   - [ターゲット] タブを選択すると、実験ターゲットの情報が表示されます。
   - [タイムライン] タブを選択すると、開始時間と終了時間に基づいてアクションが視覚的に表示されます。

CLI を使用して実験を表示するには

list-experiments コマンドを使用して実験を一覧表示します。そして、実験のリストを取得し、get-experiment コマンドを使用して、特定の実験に関する情報を取得します。

実験状態

実験は、次に示す状態のいずれかになります。

- 保留中 - 実験は保留中です。

- 開始 - 実験は開始準備中です。
- 実行中 - 実験は実行中です。
- 完了 - 実験のすべてのアクションが正常に完了しました。
- 停止中 - 停止条件がトリガーされたか、実験が手動で停止しました。
- 停止 - 実験で実行中または保留中のアクションはすべて停止します。
- 失敗 - パーミッションの不足や構文が正しくないなどのエラーにより、実験が失敗しました。

アクションの状態

アクションは、次に示す状態のいずれかになります。

- 保留中 - 実験が開始されていないか、アクションが実験で後で開始されるため、アクションは保留中です。
- 開始中 - アクションが開始する準備中です。
- 実行しています - アクションが実行中です。
- 完了 - アクションは正常に完了しました。
- キャンセルしました - アクションが始まる前に実験が停止しました。
- スキップ - アクションはスキップされました。
- 停止中 - アクションは停止しています。
- 停止 - 実験で実行中または保留中のアクションはすべて停止します。
- 失敗 - 権限の不足や構文が正しくないなど、クライアントエラーが原因でアクションが失敗しました。

実験にタグを付けるには

実験を整理しやすくするために、実験にタグを適用できます。また、実験へのアクセスを制御するためにタグベースの IAM ポリシーを実装することもできます。

コンソールを使用して実験にタグを付けるには

1. https://console.aws.amazon.com/fis/ で AWS FIS コンソールを開きます。
2. ナビゲーションペインで [実験] を選択します。
3. 実験を選択し、[アクション]、[タグの管理] を選択します。
4. タグを追加するには、新しいタグを追加を選択し、キーと値を指定します。

タグを削除するには、タグの [Remove (削除)] を選択します。

5. [Save (保存)] を選択します。

CLI を使用して実験にタグを付けるには

[tag-resource] コマンドを使用します。

実験を中止する

実行中のチャネルはいつでも停止できます。実験を停止すると、あるアクションに対して完了していないポストアクションは、実験が停止する前に完了します。停止した実験を再開することはできません。

コンソールを使用して実験を中止するには

1. https://console.aws.amazon.com/fis/ で AWS FIS コンソールを開きます。
2. ナビゲーションペインで [実験] を選択します。
3. 実験を選択し、[Stop experiment (実験を中止)] を選択します。
4. 確認のダイアログボックスで [Stop experiment (実験を中止)] を選択します。

CLI を使用して実験を停止するには

stop-experiment コマンドを使用します。

解決済みターゲットを一覧表示する

ターゲットの解決が終了した後、実験で解決されたターゲットの情報を表示できます。

コンソールを使用して解決済みターゲットを表示する

1. https://console.aws.amazon.com/fis/ で AWS FIS コンソールを開きます。
2. ナビゲーションペインで [実験] を選択します。
3. テストを選択し、[レポート] を選択します。
4. 解決されたターゲット情報は [リソース] に表示されます。

CLI を使用して解決済みターゲットを表示する

list-experiment-resolved-targets コマンドを使用します。

実験スケジューラー

AWS Fault Injection Service (FIS) を使用すると、AWS ワークロードで障害注入実験を行うことができます。これらの実験は、指定されたターゲットに対して実行する 1 つ以上のアクションがあるテンプレート上で実行します。実験を 1 回限りのタスクまたは定期的なタスクとして、基本的に FIS コンソールからスケジュールできるようになりました。スケジュールしたルールに加えて、FIS では、新しいスケジューリング機能が提供されるようになりました。FIS がスケ EventBridge ジューラと統合され、ユーザーに代わってルールが作成されます。ス EventBridge ケジューラは、1 つの中央マネージドサービスからタスクを作成、実行、管理できるサーバーレススケジューラです。

> **Important**
>
> を使用した Experiment Scheduler AWS Fault Injection Service は、AWS GovCloud （米国東部) および AWS GovCloud （米国西部) では使用できません。

トピック

開始

実行ロールは、AWS Fault Injection Service がス EventBridge ケジューラとやり取りし、Event Bridge スケジューラが FIS Experiment を開始するために引き受ける IAM ロールです。このロールにアクセス許可ポリシーをアタッチして、 FIS Experiment を呼び出すためのアクセス許可をスケジューラに付与 EventBridgeします。次の手順では、実験の開始 EventBridge を許可する新しい実行ロールとポリシーを作成する方法について説明します。

AWS CLI でスケジューラーロールを作成する

これは、Event Bridge がお客様に代わって実験をスケジュールするために必要な IAM ロールです。

1. 次のロール JSON ポリシーを引き受けるためにコピーし、 `fis-execution-role.json` という名前でローカルに保存します。この信頼ポリシーにより、ス EventBridge ケジューラはユーザーに代わってロールを引き受けることができます。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "scheduler.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

2. AWS コマンドラインインターフェイス (AWS CLI) を開き、次のコマンドを実行して新しいロールを作成します。`FisSchedulerExecutionRole` をこのロールに割り当てる名前に置き換えます。

```
aws iam create-role --role-name FisSchedulerExecutionRole --assume-role-policy-
document file://fis-execution-role.json
```

成功すると、 次の出力が表示されます。

```
{
    "Role": {
        "Path": "/",
        "RoleName": "FisSchedulerExecutionRole",
        "RoleId": "AROAZL22PDN5A6WKRBQNU",
        "Arn": "arn:aws:iam::123456789012:role/FisSchedulerExecutionRole",
        "CreateDate": "2023-08-24T17:23:05+00:00",
        "AssumeRolePolicyDocument": {
            "Version": "2012-10-17",
            "Statement": [
                {
                    "Effect": "Allow",
                    "Principal": {
                        "Service": "scheduler.amazonaws.com"
                    },
```

```
                    "Action": "sts:AssumeRole"
                }
            ]
        }
    }
}
```

3. ス EventBridge ケジューラが実験を呼び出すことを許可する新しいポリシーを作成するには、次の JSON をコピーしてローカルに として保存します `fis-start-experiment-permissions.json`。次のポリシーでは、ス EventBridge ケジューラがアカウント内のすべての実験テンプレートで `fis:StartExperiment`アクションを呼び出すことを許可します。ロールを 1 つのテストテンプレートに限定したい場合は、`"arn:aws:fis:*:*:experiment-template/*"` 末尾にある `*` を実験テンプレートの ID に置き換えてください。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "fis:StartExperiment",
            "Resource": [
                "arn:aws:fis:*:*:experiment-template/*",
                "arn:aws:fis:*:*:experiment/*"
            ]
        }
    ]
}
```

4. 次のコマンドを実行して、新しい権限ポリシーを作成します。 `FisSchedulerPolicy` をこのポリシーに割り当てる名前に置き換えます。

```
aws iam create-policy --policy-name FisSchedulerPolicy --policy-document file://fis-start-experiment-permissions.json
```

成功すると、次の出力が表示されます。ポリシーの ARN を書き留めておきます。この ARN を使用して、次のステップで、このポリシーを実行ロールにアタッチします。

```
{
    "Policy": {
```

```
        "PolicyName": "FisSchedulerPolicy",
        "PolicyId": "ANPAZL22PDN5ESVUWXLBD",
        "Arn": "arn:aws:iam::123456789012:policy/FisSchedulerPolicy",
        "Path": "/",
        "DefaultVersionId": "v1",
        "AttachmentCount": 0,
        "PermissionsBoundaryUsageCount": 0,
        "IsAttachable": true,
        "CreateDate": "2023-08-24T17:34:45+00:00",
        "UpdateDate": "2023-08-24T17:34:45+00:00"
    }
}
```

5. 次のコマンドを実行して、ポリシーを実行ロールにアタッチします。`your-policy-arn` を、前のステップで作成したポリシーの ARN に置き換えます。`FisSchedulerExecutionRole` を実行ロールの名前に置き換えます。

```
aws iam attach-role-policy --policy-arn your-policy-arn --role-name
 FisSchedulerExecutionRole
```

   `attach-role-policy` オペレーションではコマンドラインにレスポンスは返りません。

6. 特定のタグ値がある AWS FIS 実験のみを実行するようにスケジューラーを制御できます。たとえば、次のポリシーでは、すべての AWS FIS 実験テンプレートの `fis:StartExperiment` 権限が得られますが、スケジューラーが実行できるのは `Purpose=Schedule` がタグ付けされた実験のみに限定されます。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "fis:StartExperiment",
            "Resource": "arn:aws:fis:*:*:experiment/*"
        },
        {
            "Effect": "Allow",
            "Action": "fis:StartExperiment",
            "Resource": "arn:aws:fis:*:*:experiment-template/*",
            "Condition": {
                "StringEquals": {
```

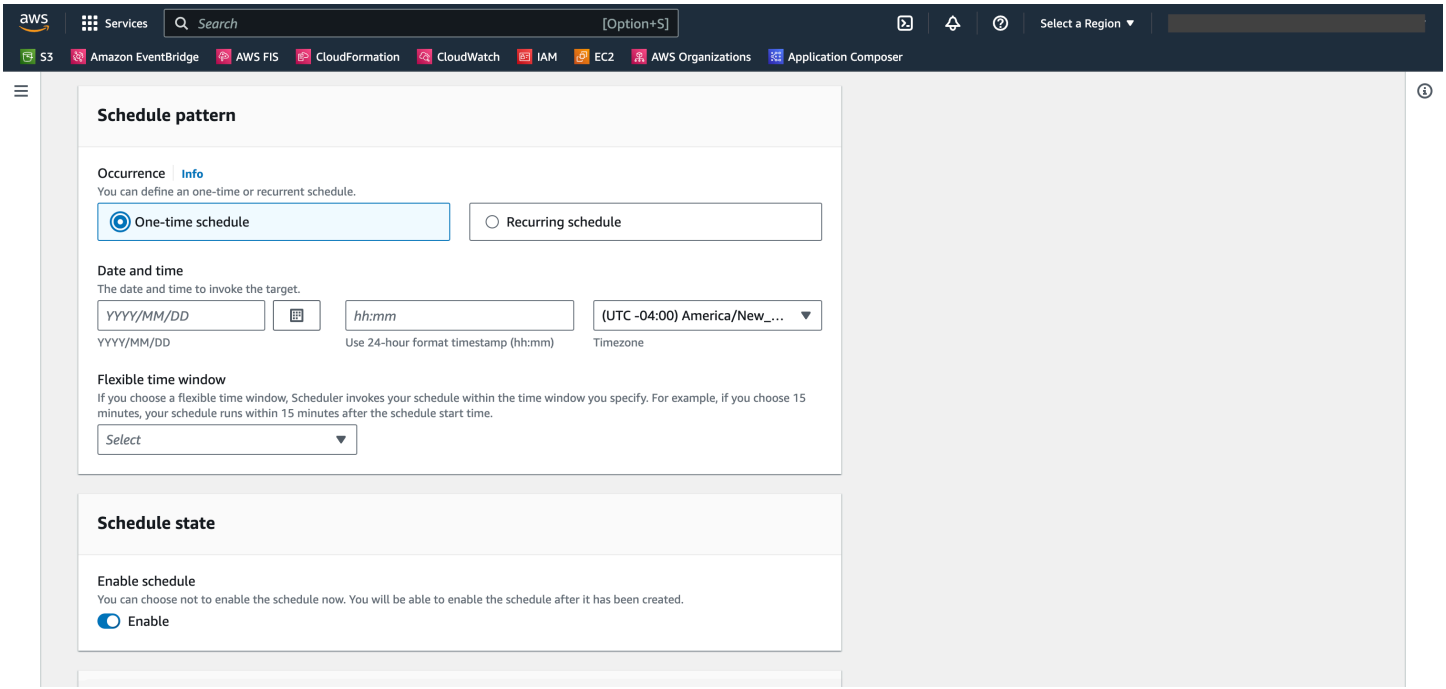
```
                    "aws:ResourceTag/Purpose": "Schedule"
                }
            }
        }
    ]
}
```

FIS 実験をスケジュールする

実験をスケジュールする前に、スケジュールを呼び出す [実験テンプレート] を1 つ以上実行する必要があります。既存の AWS リソースを使用するか、新しいリソースを作成できます。

実験テンプレートを作成したら、[アクション] をクリックして [実験をスケジュールする] を選択します。実験のスケジュールページが表示されます。スケジュール名は自動的に入力されます。

[スケジュールパターン] セクションの指示に従って、1 回限りのスケジュールまたは定期的なスケジュールを選択します。必須の入力フィールドを入力し、権限に移動します。



スケジュール状態は、デフォルトで有効になっています。注:スケジュール状態を無効にすると、スケジュールを作成しても実験はスケジュールされません。

AWS FIS Experiment Scheduler は、スEventBridge ケジューラ の上に構築されています。サポート対象の各種スケジュールタイプについては、ドキュメントを参照してください。

コンソールでスケジュールを更新するには

1. AWS FIS コンソールを開きます。
2. ナビゲーションペインで、[実験テンプレート] を選択します。
3. スケジュールを作成する [実験テンプレート] を選択します。
4. [アクション] をクリックし、ドロップダウンから [実験のスケジュール設定] を選択します。
   a. [スケジュール名] に、名前が自動入力されます。
   b. [スケジュールパターン] で [定期スケジュール] を選択します。
   c. [スケジュールタイプ] では、レートベースのスケジュールを選択できます。「スケジュールタイプ」を参照してください。
   d. [レート表現] で、実験の実行時間よりも遅いレート (5 分など) を選択します。
   e. [タイムフレーム] で、[タイムゾーン] を選択します。
   f. [開始日時] で、開始日と時間を指定します。
   g. [終了日時] で、終了日と時間を指定します。
   h. [スケジュールの状態] で、[スケジュールオプションを有効にする] を切り替えます。
   i. [許可] で、[既存のロールを使用] を選択してから、 `FisSchedulerExecutionRole` を選択します。
   j. [Next (次へ)] をクリックします。
5. [スケジュールの確認と作成] を選択し、スケジューラーの詳細を確認して [スケジュールの作成] を選択します。

実験スケジュールの更新

都合の良い日時にイベントが発生するように、実験予定を更新できます。

コンソールで実験の実行を更新するには

1. Amazon FIS コンソールを開きます。
2. ナビゲーションペインで、[実験テンプレート] を選択します。
3. スケジュールがすでに作成されているリソースタイプ:スケジュールテストテンプレートを選択します。
4. テンプレートの実験 ID をクリックします。次に、[スケジュール] タブに移動します。

5. 実験に関連する既存のスケジュールがあるかどうかを確認します。関連するスケジュールを選択し、[スケジュールを更新] ボタンをクリックします。

コンソールを使用して実験の実行を無効化または削除します。

実験をスケジュールに従って処理または実行しないようにするには、ルールを削除または無効化します。次の手順では、実験の実行を削除または無効化する方法について説明します。

ルールを削除または無効化するには

1. Amazon FIS コンソールを開きます。
2. ナビゲーションペインで、[実験テンプレート] を選択します。
3. スケジュールがすでに作成されているリソースタイプ:スケジュールテストテンプレートを選択します。
4. テンプレートの実験 ID をクリックします。次に、[スケジュール] タブに移動します。
5. 実験に関連する既存のスケジュールがあるかどうかを確認します。関連するスケジュールを選択し、[スケジュールを更新] ボタンをクリックします。
6. 次のいずれかを行います。
   a. スケジュールを削除するには、ルール [スケジュールを削除] の横にあるボタンを選択します。`delete` を入力し、[スケジュールを削除] ボタンをクリックします。
   b. スケジュールを無効にするには、ルール [スケジュールを無効化] の横にあるボタンを選択します。`disable` を入力して [スケジュールを無効にする] ボタンをクリックします。

AWS FIS のモニタリング

AWS Fault Injection Service (AWS FIS) 実験の進行状況と影響は、次のツールでモニタリングできます。

AWS FIS コンソールと AWS CLI

AWS FIS コンソールまたは AWS CLI で、実行中の実験の進行状況をモニタリングします。実験の各アクションのステータスと、各アクションの結果を表示できます。詳細については、「the section called “実験を表示します。”」を参照してください。

CloudWatch 使用状況メトリクスとアラーム

CloudWatch 使用状況メトリクスを使用して、アカウントの リソースの使用状況を可視化します。 AWSFIS 使用状況メトリクスは、AWS のサービスクォータに対応しています。使用量がサービスクォータに近づいたときに警告するアラームを設定することもできます。詳細については、「CloudWatch を使用したモニタリング」を参照してください。

実験AWSが範囲外になるタイミングを定義する CloudWatch アラームを作成することで、FIS 実験の停止条件を作成することもできます。アラームがトリガーされると、実験は停止します。詳細については、「停止条件」を参照してください。 CloudWatch アラームの作成の詳細については、「Amazon CloudWatch ユーザーガイド」の「静的しきい値に基づいて CloudWatch アラームを作成する」および「異常検出に基づいて CloudWatch アラームを作成する」を参照してください。

AWS FIS 実験ロギング

実験ロギングを有効にすると、実験の実行時に実験に関する詳細情報を取得できます。詳細については、実験ロギングを参照してください。

実験状態変更イベント

Amazon EventBridge では、システムイベントやリソースの変更に自動的に対応できます。 AWS 実験の状態が変化すると、FIS は通知を送信します。目的のイベントには、そのイベントがルールに一致した場合に自動的に実行するアクションを指定するルールを作成できます。例えば、Amazon SNS トピックへの通知の送信や、Lambda 関数の呼び出しなどです。詳細については、「を使用したモニタリング EventBridge」を参照してください。

CloudTrail ログ

AWS CloudTrail を使用して、FIS API AWS に対して行われた呼び出しに関する詳細情報をキャプチャし、Amazon S3 にログファイルとして保存します。 は、実験を実行しているリソースの

サービス APIs に対して行われた呼び出し CloudTrail もログに記録します。これらの CloudTrail ログを使用して、行われた呼び出し、呼び出し元のソース IP アドレス、呼び出し者、呼び出し日時などを確認できます。

AWS Health Dashboard の通知

AWS Health は、リソースのパフォーマンスと、AWS のサービスおよびアカウントの可用性を継続的に可視化します。実験を開始するとき、AWS FISは AWS Health Dashboard に通知を送信します。通知は、実験の期間中、マルチアカウント実験を含め、実験のターゲットとなるリソースを含むアカウントごとに表示されます。`aws:ssm:start-automation-execution` や `aws:fis:wait` など、ターゲットを含まないアクションのみを使用したマルチアカウント実験では、通知は送信されません。実験を許可するために使用されたロールに関する情報は、「影響を受けるリソース」の下に表示されます。AWS Health Dashboard の詳細については、「AWS Health ユーザーガイド」の「AWS Health Dashboard」を参照してください。

> **Note**
>
> AWS Health は、ベストエフォートベースでイベントを配信します。

Amazon CloudWatch による AWS FIS 使用状況メトリクスのモニタリング

Amazon CloudWatch を使用して、ターゲットに対する AWS FIS実験の影響をモニタリングできます。AWS FIS の使用状況もモニタリングできます。

実験状態の表示の詳細については、「実験を表示します。」を参照してください。。

AWS FIS 実験を監視する

AWS FIS 実験を計画する際は、実験のターゲットリソースタイプのベースラインまたは「定常状態」を識別するために使用できる CloudWatch メトリクスを特定します。実験を開始した後、実験テンプレートで選択したターゲットの CloudWatch メトリクスをモニタリングできます。

AWS FIS でサポートされているターゲットリソースタイプで使用できるCloudWatch メトリクスに関する詳細については、、以下を参照してください。

- CloudWatch によるインスタンスのモニタリング
- Amazon ECS CloudWatch メトリクス

- CloudWatch による Amazon RDS メトリクスのモニタリング
- CloudWatch による Run Command メトリクスのモニタリング

AWS FIS 使用状況メトリクス

CloudWatch 使用状況メトリクスを使用して、アカウントのリソースの使用状況を把握できます。これらのメトリクスを使用して、CloudWatch グラフやダッシュボードで現在のサービスの使用状況を可視化できます。

AWS FIS 使用状況メトリクスは、AWS のサービスクォータに対応しています。使用量がサービスクォータに近づいたときに警告するアラームを設定することもできます。CloudWatch アラームの使用の詳細については、「Amazon CloudWatch ユーザーガイド」を参照してください。

AWS FIS は、AWS 使用状況 名前空間に以下のメトリクスを公開します。

メトリクス	説明
`ResourceCount`	アカウントで実行されている指定されたリソースの合計数。リソースは、メトリクスに関連付けられたディメンションによって定義されます。

以下のディメンションは、AWS FIS によって発行される使用状況メトリクスを絞り込むために使用されます。

ディメンション	説明
`Service`	リソースを含む AWS のサービスの名前。AWS FIS 使用状況メトリクスの場合、このディメンションの値は `FIS` です。
`Type`	報告されるエンティティタイプ。現在、AWS FIS 使用状況メトリクスの有効な値は `Resource` のみです。
`Resource`	実行中のリソースタイプ。実験テンプレートの場合、指定できる値は `Experimen`

ディメンション	説明
	tTemplates で、アクティブな実験には ActiveExperiments を指定します。
Class	このディメンションは、将来の利用のために予約されています。

Amazon を使用した AWS FIS 実験のモニタリング EventBridge

実験の状態が変更されると、 AWS FIS は通知を送信します。これらの通知は、Amazon EventBridge（以前はイベントと呼ばれていました) CloudWatch を通じてイベントとして利用できます。 AWS FIS は、ベストエフォートベースでこれらのイベントを発行します。イベントは、ほぼリアルタイムで EventBridge に配信されます。

では EventBridge、イベントに応答してプログラムによるアクションをトリガーするルールを作成できます。例えば、SNS トピックを呼び出して E メール通知を送信するルールや、Lambda 関数を呼び出して何らかのアクションを実行するルールを設定できます。

の詳細については EventBridge、「Amazon ユーザーガイド」の EventBridge「Amazon の開始方法 EventBridge 」を参照してください。

実験状態変更イベントの構文は次のとおりです。

```
{
    "version": "0",
    "id": "12345678-1234-1234-1234-123456789012",
    "detail-type": "FIS Experiment State Change",
    "source": "aws.fis",
    "account": "123456789012",
    "time": "yyyy-mm-ddThh:mm:ssZ",
    "region": "region",
    "resources": [
        "arn:aws:fis:region:account_id:experiment/experiment-id"
    ],
    "detail": {
        "experiment-id": "EXPaBCD1efg2HIJkL3",
        "experiment-template-id": "EXTa1b2c3de5f6g7h",
        "new-state": {
            "status": "new_value",
```

```
            "reason": "reason_string"
        },
        "old-state": {
           "status": "old_value",
           "reason": "reason_string"
        }
    }
}
```

`experiment-id`

状態が変化した実験の ID。

`experiment-template-id`

実験で使用される実験テンプレートの ID。

`new_value`

実験の新しい状態。指定できる値は以下のとおりです。

- `completed`
- `failed`
- `initiating`
- `running`
- `stopped`
- `stopping`

`old_value`

実験の以前の状態。指定できる値は以下のとおりです。

- `initiating`
- `pending`
- `running`
- `stopping`

AWS FIS の実験ロギング

実験ロギングを使用すると、実験の実行時に実験に関する詳細情報を取得できます。

実験ロギングについては、各ログの宛先タイプに関連付けられたコストに基づいて経費が請求されます。詳細については、「Amazon CloudWatch 料金表」（有料利用枠、ログ 、提供されるログ の下）および「Amazon S3 料金表 」を参照してください。 Amazon S3

アクセス許可

設定した各ログ送信先にログを送信する権限を AWS FIS に与える必要があります。詳細については、Amazon CloudWatch Logs ユーザーガイドの以下を参照してください。

- ログに送信される CloudWatch ログ
- Amazon S3 に送信されたログ

ログスキーマ

以下は実験ロギングで使用されるスキーマです。スキーマの現在のバージョンは 2 です。`details` のフィールドは `log_type` の値によって異なります。`resolved_targets` のフィールドは `target_type` の値によって異なります。詳細については、「the section called “ログレコードの例”」を参照してください。

```
{
    "id": "EXP123abc456def789",
    "log_type": "experiment-start | target-resolution-start | target-resolution-detail
 | target-resolution-end | action-start | action-error | action-end | experiment-end",
    "event_timestamp": "yyyy-mm-ddThh:mm:ssZ",
    "version": "2",
    "details": {
        "account_id":"123456789012",
        "action_end_time": "yyyy-mm-ddThh:mm:ssZ",
        "action_id": "String",
        "action_name": "String",
        "action_start_time": "yyyy-mm-ddThh:mm:ssZ",
        "action_state": {
            "status": "pending | initiating | running | completed | cancelled |
 stopping | stopped | failed",
            "reason": "String"
        },
        "action_targets": "String to string map",
        "error_information": "String",
        "experiment_end_time": "yyyy-mm-ddThh:mm:ssZ",
        "experiment_state": {
```

```
            "status": "pending | initiating | running | completed | stopping | stopped
 | failed",
            "reason": "String"
        },
        "experiment_start_time": "yyyy-mm-ddThh:mm:ssZ",
        "experiment_template_id": "String",
        "page": Number,
        "parameters": "String to string map",
        "resolved_targets": [
            {
               "field": "value"
            }
        ],
        "resolved_targets_count": Number,
        "status": "failed | completed",
        "target_name": "String",
        "target_resolution_end_time": "yyyy-mm-ddThh:mm:ssZ",
        "target_resolution_start_time": "yyyy-mm-ddThh:mm:ssZ",
        "target_type": "String",
        "total_pages": Number,
        "total_resolved_targets_count": Number

    }
}
```

リリースノート

- バージョン 2 で次が導入されました。
  - `target_type` フィールドを導入し、`resolved_targets` フィールドを ARN のリストからオブジェクトのリストに変更しました。`resolved_targets` オブジェクトの有効なフィールドは、ターゲットのリソースタイプである `target_type` の値によって異なります。
  - `account_id` フィールドを追加する `action-error` および `target-resolution-detail` イベントタイプ。
- バージョン 1 は初期リリースです。

ログの宛先

AWS FIS は次の宛先へのログ配信をサポートしています。

- Amazon S3 バケット

- Amazon CloudWatch Logs ロググループ

S3 ログ配信

ログは次の場所に配信されます。

```
bucket-and-optional-prefix/AWSLogs/account-id/fis/region/experiment-
id/YYYY/MM/DD/account-id_awsfislogs_region_experiment-id_YYYYMMDDHHMMZ_hash.log
```

ログがバケットに配信されるまでに数分かかることがあります。

CloudWatch ログの配信

ログは /aws/fis/*experiment-id* という名前のログストリームに配信されます。

ログは 1 分以内にロググループに配信されます。

ログレコードの例

以下は、ランダムに選択された EC2 インスタンスで aws:ec2:reboot-instances アクションを実行する実験のログレコードの例です。

レコード

- experiment-start
- target-resolution-start
- target-resolution-detail
- target-resolution-end
- action-start
- action-end
- action-error
- experiment-end

experiment-start

以下に示したのは、`experiment-start` イベントのレコードの例です。

```
{
```

```
    "id": "EXPhjAXCGY78HV2a4A",
    "log_type": "experiment-start",
    "event_timestamp": "2023-05-31T18:50:45Z",
    "version": "2",
    "details": {
        "experiment_template_id": "EXTCDh1M8HHkhxoaQ",
        "experiment_start_time": "2023-05-31T18:50:43Z"
    }
}
```

target-resolution-start

以下に示したのは、`target-resolution-start` イベントのレコードの例です。

```
{
    "id": "EXPhjAXCGY78HV2a4A",
    "log_type": "target-resolution-start",
    "event_timestamp": "2023-05-31T18:50:45Z",
    "version": "2",
    "details": {
        "target_resolution_start_time": "2023-05-31T18:50:45Z",
        "target_name": "EC2InstancesToReboot"
    }
}
```

target-resolution-detail

以下に示したのは、`target-resolution-detail` イベントのレコードの例です。ターゲットの解決に失敗した場合、レコードには `error_information` フィールドも含まれます。

```
{
    "id": "EXPhjAXCGY78HV2a4A",
    "log_type": "target-resolution-detail",
    "event_timestamp": "2023-05-31T18:50:45Z",
    "version": "2",
    "details": {
        "target_resolution_end_time": "2023-05-31T18:50:45Z",
        "target_name": "EC2InstancesToReboot",
        "target_type": "aws:ec2:instance",
        "account_id": "123456789012",
        "resolved_targets_count": 2,
        "status": "completed"
```

```
    }
}
```

target-resolution-end

ターゲットの解決に失敗した場合、レコードには error_information フィールドも含まれます。total_pages が 1 より大きい場合、解決されたターゲットの数が 1 つのレコードのサイズ制限を超えています。残りの解決済みターゲットを含むレコードが他にも target-resolution-end レコードあります。

EC2 アクションの target-resolution-end イベントのレコード例を以下に示します。

```
{
    "id": "EXPhjAXCGY78HV2a4A",
    "log_type": "target-resolution-end",
    "event_timestamp": "2023-05-31T18:50:45Z",
    "version": "2",
    "details": {
        "target_resolution_end_time": "2023-05-31T18:50:46Z",
        "target_name": "EC2InstanceToReboot",
        "target_type": "aws:ec2:instance",
        "resolved_targets": [
            {
                "arn": "arn:aws:ec2:us-east-1:123456789012:instance/
i-0f7ee2abffc330de5"
            }
        ],
        "page": 1,
        "total_pages": 1
    }
}
```

以下は EKS アクションの target-resolution-end イベントのレコード例です。

```
{
    "id": "EXP24YfiucfyVPJpEJn",
    "log_type": "target-resolution-end",
    "event_timestamp": "2023-05-31T18:50:45Z",
    "version": "2",
    "details": {
        "target_resolution_end_time": "2023-05-31T18:50:46Z",
```

```
        "target_name": "myPods",
        "target_type": "aws:eks:pod",
        "resolved_targets": [
            {
                "pod_name": "example-696fb6498b-sxhw5",
                "namespace": "default",
                "cluster_arn": "arn:aws:eks:us-east-1:123456789012:cluster/fis-demo-
cluster",
                "target_container_name": "example"
            }
        ],
        "page": 1,
        "total_pages": 1
    }
}
```

action-start

以下に示したのは、action-start イベントのレコードの例です。実験テンプレートでアクションのパラメータが指定されている場合、レコードには parameters フィールドも含まれます。

```
{
    "id": "EXPhjAXCGY78HV2a4A",
    "log_type": "action-start",
    "event_timestamp": "2023-05-31T18:50:56Z",
    "version": "2",
    "details": {
        "action_name": "Reboot",
        "action_id": "aws:ec2:reboot-instances",
        "action_start_time": "2023-05-31T18:50:56Z",
        "action_targets": {"Instances":"EC2InstancesToReboot"}
    }
}
```

action-error

以下に示したのは、action-error イベントのレコードの例です。このイベントは、アクションが失敗したときにのみ返されます。アクションが失敗したアカウントごとに返されます。

```
{
    "id": "EXPhjAXCGY78HV2a4A",
    "log_type": "action-error",
```

```
    "event_timestamp": "2023-05-31T18:50:56Z",
    "version": "2",
    "details": {
        "action_name": "pause-io",
        "action_id": "aws:ebs:pause-volume-io",
        "account_id": "123456789012",
        "action_state": {
            "status": "failed",
            "reason":"Unable to start Pause Volume IO. Target volumes must be attached
 to an instance type based on the Nitro system. VolumeId(s): [vol-1234567890abcdef0]:"
        }
    }
}
```

action-end

以下に示したのは、`action-end` イベントのレコードの例です。

```
{
    "id": "EXPhjAXCGY78HV2a4A",
    "log_type": "action-end",
    "event_timestamp": "2023-05-31T18:50:56Z",
    "version": "2",
    "details": {
        "action_name": "Reboot",
        "action_id": "aws:ec2:reboot-instances",
        "action_end_time": "2023-05-31T18:50:56Z",
        "action_state": {
            "status": "completed",
            "reason": "Action was completed."
        }
    }
}
```

experiment-end

以下に示したのは、`experiment-end` イベントのレコードの例です。

```
{
    "id": "EXPhjAXCGY78HV2a4A",
    "log_type": "experiment-end",
    "event_timestamp": "2023-05-31T18:50:57Z",
```

```
    "version": "2",
    "details": {
        "experiment_end_time": "2023-05-31T18:50:57Z",
        "experiment_state": {
            "status": "completed",
            "reason": "Experiment completed"
        }
    }
}
```

実験ロギングを有効にする

実験ロギングは、デフォルトで無効になっています。実験用に実験ログを受け取るには、ロギングを有効にした実験テンプレートから実験を作成する必要があります。過去にロギングに使用したことのない宛先を使用する設定のテストを初めて実行するときは、この宛先へのログ配信を設定するため、実験開始までに約 15 秒の遅延が生じます。

コンソールで実験ロギングの作成を有効にするには

1. https://console.aws.amazon.com/fis/ で AWS FIS コンソールを開きます。
2. ナビゲーションペインで、[実験テンプレート] を選択します。
3. 実験テンプレートを選択し、[アクション]、[実験テンプレートを更新する] を選択します。
4. [ログ] では、送信先オプションを設定します。S3 バケットにログを送信するには、[Amazon S3 バケットに送信] を選択し、バケット名とプレフィックスを入力します。ログを CloudWatch Logs に送信するには、 CloudWatch ログに送信を選択し、ロググループを入力します。
5. [実験テンプレートの作成] を選択します。

AWS CLI を使用した実験ロギングを有効にするには

update-experiment-template コマンドを使用して、ログ設定を指定します。

実験ロギングの無効化

実験のログを受け取りたくない場合は、実験ロギングの作成を無効にできます。

コンソールを使用して実験ロギングの作成を無効にするには

1. https://console.aws.amazon.com/fis/ で AWS FIS コンソール を開きます。
2. ナビゲーションペインで、[実験テンプレート] を選択します。

3. 実験テンプレートを選択し、[アクション]、[実験テンプレートを更新する] を選択します。
4. ログ の場合、Amazon S3 バケットへの送信 と CloudWatch ログ への送信 をクリアします。
5. [実験テンプレートの更新] を選択します。

AWS CLI を使用して実験ロギングの作成を無効にするには

update-experiment-template コマンドを使用して、空のログ設定を指定します。

AWS CloudTrail での API コールのログ記録

AWS Fault Injection Service (AWS FIS) はAWS CloudTrail、FIS のユーザー、ロール、または AWS AWSのサービスによって実行されたアクションを記録するサービスである AWS と統合されています。 は、FIS のすべての API コールをイベントとして CloudTrail キャプチャします。取得される呼び出しには、AWS FIS コンソールの呼び出しと、AWS FIS API オペレーションへのコード呼び出しがあります。証跡を作成する場合は、FIS の CloudTrail イベントなど、Amazon S3 AWS バケットへのイベントの継続的な配信を有効にすることができます。証跡を設定しない場合でも、コンソールのイベント履歴 で最新の CloudTrail イベントを表示できます。 で収集された情報を使用して CloudTrail、FIS AWS に対するリクエスト、リクエスト元の IP アドレス、リクエスト者、リクエスト日時などの詳細を確認できます。

の詳細については CloudTrail、「 AWS CloudTrailユーザーガイド」を参照してください。

の使用 CloudTrail

CloudTrail アカウントを作成するAWS アカウントと、 は で有効になります。AWS FIS でアクティビティが発生すると、そのアクティビティは CloudTrail イベント履歴 の他のAWSサービスイベントとともにイベントに記録されます。最近のイベントは、AWS アカウント で表示、検索、ダウンロードできます。詳細については、「イベント履歴 での CloudTrail イベントの表示」を参照してください。

AWS FIS に関するイベントなど、AWS アカウント 内でのイベントの継続的な記録については、証跡を作成します。証跡により、 はログファイル CloudTrail を Amazon S3 バケットに配信できます。デフォルトでは、コンソールで証跡を作成すると、すべての AWS リージョンに証跡が適用されます。追跡は、AWS パーティションのすべてのリージョンからのイベントをログに記録し、指定した Amazon S3 バケットにログファイルを配信します。さらに、 CloudTrail ログで収集されたデータをより詳細に分析し、それに基づく対応を行うように他の AWSサービスを設定できます。詳細については、次を参照してください。

- AWS アカウントの証跡を作成します
- CloudTrail サポートされているサービスと統合
- の Amazon SNS 通知の設定 CloudTrail
- 複数のリージョンからの CloudTrail ログファイルの受信と複数のアカウントからの CloudTrail ログファイルの受信

すべての AWS FIS アクションは によってログに記録 CloudTrail され、「AWSFault Injection Service API リファレンス」に記載されています。ターゲットリソースで実行される実験アクションについては、リソースを所有するサービスの API リファレンスドキュメントを参照してください。例えば、Amazon EC2 インスタンスで実行されるアクションについては、「Amazon EC2 API リファレンス」を参照してください。

各イベントまたはログエントリには、誰がリクエストを生成したかという情報が含まれます。ID 情報は次の判断に役立ちます。

- リクエストが、ルートとユーザー認証情報のどちらを使用して送信されたか。
- リクエストが、ロールとフェデレーションユーザーの一時的なセキュリティ認証情報のどちらを使用して送信されたか。
- リクエストが、別の AWS サービスによって送信されたかどうか。

詳細については、「CloudTrail userIdentity 要素」を参照してください。

AWS FIS ログファイルエントリについて理解する

証跡は、指定した Amazon S3 バケットにイベントをログファイルとして配信できるようにする設定です。 CloudTrail ログファイルには、1 つ以上のログエントリが含まれます。イベントは任意の送信元からの単一のリクエストを表し、リクエストされたアクション、アクションの日時、リクエストパラメータなどに関する情報が含まれます。 CloudTrail ログファイルは、パブリック API コールの順序付けられたスタックトレースではないため、特定の順序では表示されません。

AWS FIS `StopExperiment`アクションの呼び出しの CloudTrail ログエントリの例を次に示します。

```
{
  "eventVersion": "1.08",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "AKIAIOSFODNN7EXAMPLE:jdoe",
    "arn": "arn:aws:sts::111122223333:assumed-role/example/jdoe",
```

```
    "accountId": "111122223333",
    "accessKeyId": "AKIAI44QH8DHBEXAMPLE",
    "sessionContext": {
      "sessionIssuer": {
        "type": "Role",
        "principalId": "AKIAIOSFODNN7EXAMPLE",
        "arn": "arn:aws:iam::111122223333:role/example",
        "accountId": "111122223333",
        "userName": "example"
      },
      "webIdFederationData": {},
      "attributes": {
        "creationDate": "2020-12-03T09:40:42Z",
        "mfaAuthenticated": "false"
      }
    }
  },
  "eventTime": "2020-12-03T09:44:20Z",
  "eventSource": "fis.amazonaws.com",
  "eventName": "StopExperiment",
  "awsRegion": "us-east-1",
  "sourceIPAddress": "192.51.100.25",
  "userAgent": "Boto3/1.22.9 Python/3.8.13 Linux/5.4.186-113.361.amzn2int.x86_64
 Botocore/1.25.9",
  "requestParameters": {
    "clientToken": "1234abc5-6def-789g-012h-ijklm34no56p",
    "experimentTemplateId": "ABCDE1fgHIJkLmNop",
    "tags": {}
  },
  "responseElements": {
    "experiment": {
      "actions": {
        "exampleAction1": {
          "actionId": "aws:ec2:stop-instances",
          "duration": "PT10M",
          "state": {
            "reason": "Initial state",
            "status": "pending"
          },
          "targets": {
            "Instances": "exampleTag1"
          }
        },
        "exampleAction2": {
```

```
          "actionId": "aws:ec2:stop-instances",
          "duration": "PT10M",
          "state": {
            "reason": "Initial state",
            "status": "pending"
          },
          "targets": {
            "Instances": "exampleTag2"
          }
        }
      },
      "creationTime": 1605788649.95,
      "endTime": 1606988660.846,
      "experimentTemplateId": "ABCDE1fgHIJkLmNop",
      "id": "ABCDE1fgHIJkLmNop",
      "roleArn": "arn:aws:iam::111122223333:role/AllowFISActions",
      "startTime": 1605788650.109,
      "state": {
        "reason": "Experiment stopped",
        "status": "stopping"
      },
      "stopConditions": [
        {
          "source": "aws:cloudwatch:alarm",
          "value": "arn:aws:cloudwatch:us-east-1:111122223333:alarm:example"
        }
      ],
      "tags": {},
      "targets": {
        "ExampleTag1": {
          "resourceTags": {
            "Example": "tag1"
          },
          "resourceType": "aws:ec2:instance",
          "selectionMode": "RANDOM(1)"
        },
        "ExampleTag2": {
          "resourceTags": {
            "Example": "tag2"
          },
          "resourceType": "aws:ec2:instance",
          "selectionMode": "RANDOM(1)"
        }
      }
```

```
    }
  },
  "requestID": "1abcd23e-f4gh-567j-klm8-9np01q234r56",
  "eventID": "1234a56b-c78d-9e0f-g1h2-34jk56m7n890",
  "readOnly": false,
  "eventType": "AwsApiCall",
  "managementEvent": true,
  "recipientAccountId": "111122223333",
  "eventCategory": "Management"

}
```

以下は、 FIS アクションを含む実験の一部として AWS FIS が呼び出した API `aws:ssm:send-command` AWS アクションの CloudTrail ログエントリの例です。この `userIdentity` 要素には、ロールを引き受けて取得した一時的な認証情報を使用して行われたリクエストが反映されます。引き受けたロールの名前が `userName` に表示されます。実験の ID EXP21nT17WMzA6dnUgz は、想定されるロールの ARN `principalId` に含まれており、その一部としても表示されます。

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "AROATZZZ4JPIXUEXAMPLE:EXP21nT17WMzA6dnUgz",
        "arn": "arn:aws:sts::111122223333:assumed-role/AllowActions/
EXP21nT17WMzA6dnUgz",
        "accountId": "111122223333",
        "accessKeyId": "AKIAI44QH8DHBEXAMPLE",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "AROATZZZ4JPIXUEXAMPLE",
                "arn": "arn:aws:iam::111122223333:role/AllowActions",
                "accountId": "111122223333",
                "userName": "AllowActions"
            },
            "webIdFederationData": {},
            "attributes": {
                "creationDate": "2022-05-30T13:23:19Z",
                "mfaAuthenticated": "false"
            }
        },
        "invokedBy": "fis.amazonaws.com"
    },
```

```
    "eventTime": "2022-05-30T13:23:19Z",
    "eventSource": "ssm.amazonaws.com",
    "eventName": "ListCommands",
    "awsRegion": "us-east-2",
    "sourceIPAddress": "fis.amazonaws.com",
    "userAgent": "fis.amazonaws.com",
    "requestParameters": {
        "commandId": "51dab97f-489b-41a8-a8a9-c9854955dc65"
    },
    "responseElements": null,
    "requestID": "23709ced-c19e-471a-9d95-cf1a06b50ee6",
    "eventID": "145fe5a6-e9d5-45cc-be25-b7923b950c83",
    "readOnly": true,
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333",
    "eventCategory": "Management"
}
```

AWS Fault Injection Service 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からメリットを得られます。

セキュリティは、 AWS とユーザーの間で共有される責任です。責任共有モデルでは、これをクラウドのセキュリティおよびクラウド内のセキュリティと説明しています。

- クラウドのセキュリティ — AWS 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。コンプライアンスAWS プログラム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監査者は定期的にセキュリティの有効性をテストおよび検証。 AWS Fault Injection Service 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、「コンプライアンスプログラムAWS による対象範囲内のサービスコンプライアンスプログラム」を参照してください。
- クラウドのセキュリティ — 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、データの機密性、会社の要件、適用される法律や規制など、その他の要因についても責任を負います。

このドキュメントは、 AWS FIS 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 AWS FIS を設定する方法について説明します。また、 AWS FIS リソースのモニタリングや保護に役立つ他の AWS のサービスの使用方法についても説明します。

内容

AWS Fault Injection Service でのデータ保護

責任 AWS 共有モデル、 AWS Fault Injection Service でのデータ保護に適用されます。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。お客様は、このインフラストラクチャでホストされているコンテ

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のよくある質問」を参照してください。欧州でのデータ保護の詳細については、AWS セキュリティブログに投稿された「AWS 責任共有モデルおよび GDPR」のブログ記事を参照してください。

データ保護の目的で、 認証情報を保護し AWS アカウント 、 AWS IAM Identity Center または AWS Identity and Access Management (IAM) を使用して個々のユーザーを設定することをお勧めします。この方法により、それぞれのジョブを遂行するために必要な権限のみが各ユーザーに付与されます。また、次の方法でデータを保護することもお勧めします:

- 各アカウントで多要素認証 (MFA) を使用します。
- SSL/TLS を使用して AWS リソースと通信します。TLS 1.2 は必須であり TLS 1.3 がお勧めです。
- 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。
- 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 などの高度なマネージド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 140-2 検証済みの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) 140-2」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は、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配置し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 FIS AWS CLIまたは他の AWS のサービス を使用する場合も同様です。AWS SDKs 名前に使用する自由記述のテキストフィールドやタグに入力したデータは、課金や診断ログに使用される場合があります。外部サーバーへの URL を提供する場合は、そのサーバーへのリクエストを検証するための認証情報を URL に含めないように強くお勧めします。

保管中の暗号化

AWS FIS は常に保管中のデータを暗号化します。 AWS FIS のデータは、保管時に透過的なサーバー側の暗号化を使用して暗号化されます。これは、機密データの保護における負担と複雑な作業を減らすのに役立ちます。保管時に暗号化することで、セキュリティを重視したアプリケーションを構築して、暗号化のコンプライアンスと規制の要件を満たすことができます。

転送中の暗号化

AWS FIS は、サービスと他の統合サービスの間で転送中のデータを暗号化します AWS 。 AWS FIS と統合サービス間を通過するすべてのデータは、Transport Layer Security (TLS) を使用して暗号化されます。他の統合 AWS サービスの詳細については、「」を参照してください[サポート対象 AWS のサービス](#)。

AWS Fault Injection Service の Identity and Access Management

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 AWS FIS リソースの使用を承認する (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 AWS のサービス 使用できる です。

内容

対象者

AWS Identity and Access Management (IAM) の使用方法は、 AWS FIS で行う作業によって異なります。

サービスユーザー - AWS FIS サービスを使用してジョブを実行する場合、管理者から必要な認証情報とアクセス許可が与えられます。さらに多くの AWS FIS 機能を使用して作業を行う場合は、追加のアクセス許可が必要になることがあります。アクセスの管理方法を理解しておくと、管理者に適切な許可をリクエストするうえで役立ちます。

サービス管理者 – 社内の AWS FIS リソースを担当している場合は、通常、 AWS FIS へのフルアクセスがあります。サービスユーザーがどの AWS FIS 機能やリソースにアクセスするかを決めるのは

管理者の仕事です。その後、IAM 管理者にリクエストを送信して、サービスユーザーの権限を変更する必要があります。このページの情報を点検して、IAM の基本概念を理解してください。

IAM 管理者 – IAM 管理者は、 AWS FIS へのアクセスを管理するポリシーの作成方法の詳細について確認する場合があります。

アイデンティティを使用した認証

認証とは、ID 認証情報 AWS を使用して にサインインする方法です。として、IAM ユーザーとして AWS アカウントのルートユーザー、または IAM ロールを引き受けて認証（にサインイン AWS) される必要があります。

ID ソースを介して提供された認証情報を使用して、フェデレーティッド ID AWS として にサインインできます。 AWS IAM Identity Center（IAM Identity Center) ユーザー、会社のシングルサインオン認証、Google または Facebook の認証情報は、フェデレーティッド ID の例です。フェデレーティッド ID としてサインインする場合、IAM ロールを使用して、前もって管理者により ID フェデレーションが設定されています。フェデレーション AWS を使用して にアクセスすると、間接的にロールを引き受けることになります。

ユーザーのタイプに応じて、 AWS Management Console または AWS アクセスポータルにサインインできます。へのサインインの詳細については AWS、「 ユーザーガイド」の「 へのサインイン AWS アカウント方法AWS サインイン 」を参照してください。

AWS プログラムで にアクセスする場合、 は Software Development Kit (SDK) とコマンドラインインターフェイス (CLI) AWS を提供し、認証情報を使用してリクエストに暗号で署名します。 AWS ツールを使用しない場合は、リクエストに自分で署名する必要があります。推奨される方法を使用してリクエストを自分で署名する方法の詳細については、IAM ユーザーガイドの API AWS リクエストの署名を参照してください。

使用する認証方法を問わず、追加セキュリティ情報の提供をリクエストされる場合もあります。例えば、 AWS では、多要素認証 (MFA) を使用してアカウントのセキュリティを向上させることをお勧めします。詳細については、「AWS IAM Identity Center ユーザーガイド」の「多要素認証」および「IAM ユーザーガイド」の「AWSでの多要素認証 (MFA) の使用」を参照してください。

AWS アカウント ルートユーザー

を作成するときは AWS アカウント、アカウント内のすべての AWS のサービス およびリソースへの完全なアクセス権を持つ 1 つのサインインアイデンティティから始めます。この ID は AWS アカウント ルートユーザーと呼ばれ、アカウントの作成に使用した E メールアドレスとパスワードでサインインすることでアクセスできます。日常的なタスクには、ルートユーザーを使用しないことを強く

お勧めします。ルートユーザーの認証情報は保護し、ルートユーザーでしか実行できないタスクを実行するときに使用します。ルートユーザーとしてサインインする必要があるタスクの完全なリストについては、「IAM ユーザーガイド」の「ルートユーザー認証情報が必要なタスク」を参照してください。

フェデレーティッドアイデンティティ

ベストプラクティスとして、管理者アクセスを必要とするユーザーを含む人間のユーザーに、一時的な認証情報を使用して にアクセスするための ID プロバイダーとのフェデレーションの使用を要求 AWS のサービス します。

フェデレーティッド ID は、エンタープライズユーザーディレクトリ、ウェブ ID プロバイダー、AWS Directory Service、Identity Center ディレクトリのユーザー、または ID ソースを通じて提供された認証情報 AWS のサービス を使用して にアクセスするユーザーです。フェデレーティッド ID がにアクセスすると AWS アカウント、ロールを引き受け、ロールは一時的な認証情報を提供します。

アクセスを一元管理する場合は、 AWS IAM Identity Centerを使用することをお勧めします。IAM Identity Center でユーザーとグループを作成することも、独自の ID ソース内のユーザーとグループのセットに接続して同期して、すべての AWS アカウント とアプリケーションで使用することもできます。IAM Identity Center の詳細については、「AWS IAM Identity Center ユーザーガイド」の「IAM Identity Center とは」を参照してください。

IAM ユーザーとグループ

IAM ユーザーは、単一のユーザーまたはアプリケーションに対して特定のアクセス許可 AWS アカウント を持つ 内のアイデンティティです。可能であれば、パスワードやアクセスキーなどの長期的な認証情報を保有する IAM ユーザーを作成する代わりに、一時認証情報を使用することをお勧めします。ただし、IAM ユーザーでの長期的な認証情報が必要な特定のユースケースがある場合は、アクセスキーをローテーションすることをお勧めします。詳細については、IAM ユーザーガイド の「長期的な認証情報を必要とするユースケースのためにアクセスキーを定期的にローテーションする」を参照してください。

IAM グループは、IAM ユーザーの集団を指定するアイデンティティです。グループとしてサインインすることはできません。グループを使用して、複数のユーザーに対して一度に権限を指定できます。多数のユーザーグループがある場合、グループを使用することで権限の管理が容易になります。例えば、IAMAdmins という名前のグループを設定して、そのグループに IAM リソースを管理する許可を与えることができます。

ユーザーは、ロールとは異なります。ユーザーは 1 人の人または 1 つのアプリケーションに一意に関連付けられますが、ロールはそれを必要とする任意の人が引き受けるようになっています。ユー

ザーには永続的な長期の認証情報がありますが、ロールでは一時的な認証情報が提供されます。詳細については、「IAM ユーザーガイド」の「IAM ユーザー (ロールではなく) の作成が適している場合」を参照してください。

IAM ロール

IAM ロールは、特定のアクセス許可 AWS アカウント を持つ 内のアイデンティティです。これは IAM ユーザーに似ていますが、特定のユーザーには関連付けられていません。ロール を切り替える AWS Management Console ことで、 で IAM ロールを一時的に引き受けることができます。ロールを引き受けるには、 または AWS API AWS CLI オペレーションを呼び出すか、カスタム URL を使用します。ロールを使用する方法の詳細については、「IAM ユーザーガイド」の「IAM ロールの使用」を参照してください。

IAM ロールと一時的な認証情報は、次の状況で役立ちます:

- フェデレーションユーザーアクセス – フェデレーティッド ID に許可を割り当てるには、ロールを作成してそのロールの許可を定義します。フェデレーティッド ID が認証されると、その ID はロールに関連付けられ、ロールで定義されている許可が付与されます。フェデレーションの詳細については、「IAM ユーザーガイド」の「Creating a role for a third-party Identity Provider」(サードパーティーアイデンティティプロバイダー向けロールの作成) を参照してください。IAM Identity Center を使用する場合は、許可セットを設定します。アイデンティティが認証後にアクセスできるものを制御するため、IAM Identity Center は、権限セットを IAM のロールに関連付けます。アクセス許可セットの詳細については、「AWS IAM Identity Center ユーザーガイド」の「アクセス許可セット」を参照してください。
- 一時的な IAM ユーザー権限 - IAM ユーザーまたはロールは、特定のタスクに対して複数の異なる権限を一時的に IAM ロールで引き受けることができます。
- クロスアカウントアクセス - IAM ロールを使用して、自分のアカウントのリソースにアクセスすることを、別のアカウントの人物 (信頼済みプリンシパル) に許可できます。クロスアカウントアクセス権を付与する主な方法は、ロールを使用することです。ただし、一部の では AWS のサービス、 (ロールをプロキシとして使用する代わりに) ポリシーをリソースに直接アタッチできます。クロスアカウントアクセスのロールとリソースベースのポリシーの違いについては、IAM ユーザーガイドの「IAM でのクロスアカウントリソースアクセス」を参照してください。
- クロスサービスアクセス — 一部の は、他の の機能 AWS のサービス を使用します AWS のサービス。例えば、あるサービスで呼び出しを行うと、通常そのサービスによって Amazon EC2 でアプリケーションが実行されたり、Amazon S3 にオブジェクトが保存されたりします。サービスでは、呼び出し元プリンシパルの許可、サービスロール、またはサービスリンクロールを使用してこれを行う場合があります。

- 転送アクセスセッション (FAS) – IAM ユーザーまたはロールを使用して でアクションを実行する場合 AWS、ユーザーはプリンシパルと見なされます。一部のサービスを使用する際に、アクションを実行することで、別のサービスの別のアクションがトリガーされることがあります。FAS は、 を呼び出すプリンシパルのアクセス許可を AWS のサービス、ダウンストリームサービス AWS のサービス へのリクエストリクエストリクエストと組み合わせて使用します。FAS リクエストは、サービスが他の AWS のサービス またはリソースとのやり取りを完了する必要があるリクエストを受け取った場合にのみ行われます。この場合、両方のアクションを実行するためのアクセス許可が必要です。FAS リクエストを行う際のポリシーの詳細については、「転送アクセスセッション」を参照してください。
- サービスロール - サービスがユーザーに代わってアクションを実行するために引き受ける IAM ロールです。IAM 管理者は、IAM 内からサービスロールを作成、変更、削除できます。詳細については、「IAM ユーザーガイド」の「AWS のサービスにアクセス許可を委任するロールの作成」を参照してください。
- サービスにリンクされたロール – サービスにリンクされたロールは、 にリンクされたサービスロールの一種です AWS のサービス。サービスは、ユーザーに代わってアクションを実行するロールを引き受けることができます。サービスにリンクされたロールは に表示され AWS アカウント 、サービスによって所有されます。IAM 管理者は、サービスリンクロールの許可を表示できますが、編集することはできません。

- Amazon EC2 で実行されているアプリケーション – IAM ロールを使用して、EC2 インスタンスで実行され、 AWS CLI または AWS API リクエストを行うアプリケーションの一時的な認証情報を管理できます。これは、EC2 インスタンス内でのアクセスキーの保存に推奨されます。 AWS ロールを EC2 インスタンスに割り当て、そのすべてのアプリケーションで使用できるようにするには、インスタンスにアタッチされたインスタンスプロファイルを作成します。インスタンスプロファイルにはロールが含まれ、EC2 インスタンスで実行されるプログラムは一時的な認証情報を取得できます。詳細については、「IAM ユーザーガイド」の「Amazon EC2 インスタンスで実行されるアプリケーションに IAM ロールを使用して許可を付与する」を参照してください。

IAM ロールと IAM ユーザーのどちらを使用するかについては、「IAM ユーザーガイド」の「(IAM ユーザーではなく) IAM ロールをいつ作成したら良いのか?」を参照してください。

ポリシーを使用したアクセスの管理

でアクセスを制御する AWS には、ポリシーを作成し、 AWS ID またはリソースにアタッチします。ポリシーは AWS 、アイデンティティまたはリソースに関連付けられているときにアクセス許可を定義する のオブジェクトです。 は、プリンシパル (ユーザー、ルートユーザー、またはロールセッション) がリクエストを行うときに、これらのポリシー AWS を評価します。ポリシーでの権限によ

り、リクエストが許可されるか拒否されるかが決まります。ほとんどのポリシーは JSON ドキュメント AWS として に保存されます。JSON ポリシードキュメントの構造と内容の詳細については、「IAM ユーザーガイド」の「JSON ポリシー概要」を参照してください。

管理者は AWS JSON 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んなリソースにどんな条件でアクションを実行できるかということです。

デフォルトでは、ユーザーやロールに権限はありません。IAM 管理者は、リソースで必要なアクションを実行するための権限をユーザーに付与する IAM ポリシーを作成できます。その後、管理者はロールに IAM ポリシーを追加し、ユーザーはロールを引き継ぐことができます。

IAM ポリシーは、オペレーションの実行方法を問わず、アクションの許可を定義します。例えば、`iam:GetRole` アクションを許可するポリシーがあるとします。そのポリシーを持つユーザーは、 AWS Management Console、、 AWS CLIまたは AWS API からロール情報を取得できます。

アイデンティティベースのポリシー

アイデンティティベースポリシーは、IAM ユーザー、ユーザーのグループ、ロールなど、アイデンティティにアタッチできる JSON 権限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 ポリシーの作成」を参照してください。

アイデンティティベースのポリシーは、さらにインラインポリシーまたはマネージドポリシーに分類できます。インラインポリシーは、単一のユーザー、グループ、またはロールに直接埋め込まれています。管理ポリシーは、 内の複数のユーザー、グループ、ロールにアタッチできるスタンドアロンポリシーです AWS アカウント。管理ポリシーには、 AWS 管理ポリシーとカスタマー管理ポリシーが含まれます。マネージドポリシーまたはインラインポリシーのいずれかを選択する方法については、「IAM ユーザーガイド」の「マネージドポリシーとインラインポリシーの比較」を参照してください。

リソースベースのポリシー

リソースベースのポリシーは、リソースに添付する JSON ポリシードキュメントです。リソースベースのポリシーには例として、IAM ロールの信頼ポリシー や Amazon S3 バケットポリシー があげられます。リソースベースのポリシーをサポートするサービスでは、サービス管理者はポリシーを使用して特定のリソースへのアクセスを制御できます。ポリシーがアタッチされているリソースの場合、指定されたプリンシパルがそのリソースに対して実行できるアクションと条件は、ポリシーによって定義されます。リソースベースのポリシーでは、プリンシパルを指定する必要があります。プ

リンシパルには、アカウント、ユーザー、ロール、フェデレーティッドユーザー、または を含めることができます AWS のサービス。

リソースベースのポリシーは、そのサービス内にあるインラインポリシーです。リソースベースのポリシーでは、IAM の AWS マネージドポリシーを使用できません。

アクセスコントロールリスト (ACL)

アクセスコントロールリスト (ACL) は、どのプリンシパル (アカウントメンバー、ユーザー、またはロール) がリソースにアクセスするための許可を持つかを制御します。ACL はリソースベースのポリシーに似ていますが、JSON ポリシードキュメント形式は使用しません。

Amazon S3、 AWS WAF、および Amazon VPC は、ACLs。ACL の詳細については、『Amazon Simple Storage Service デベロッパーガイド』の「アクセスコントロールリスト (ACL) の概要」を参照してください。

その他のポリシータイプ

AWS は、一般的ではない追加のポリシータイプをサポートします。これらのポリシータイプでは、より一般的なポリシータイプで付与された最大の権限を設定できます。

- アクセス許可の境界 - アクセス許可の境界は、アイデンティティベースのポリシーによって IAM エンティティ (IAM ユーザーまたはロール) に付与できる権限の上限を設定する高度な機能です。エンティティにアクセス許可の境界を設定できます。結果として得られる権限は、エンティティのアイデンティティベースポリシーとそのアクセス許可の境界の共通部分になります。`Principal` フィールドでユーザーまたはロールを指定するリソースベースのポリシーでは、アクセス許可の境界は制限されません。これらのポリシーのいずれかを明示的に拒否した場合、権限は無効になります。アクセス許可の境界の詳細については、「IAM ユーザーガイド」の「IAM エンティティのアクセス許可の境界」を参照してください。
- サービスコントロールポリシー (SCPs) – SCPs は、 の組織または組織単位 (OU) に対する最大アクセス許可を指定する JSON ポリシーです AWS Organizations。 AWS Organizations は、 AWS アカウント ビジネスが所有する複数の をグループ化して一元管理するためのサービスです。組織内のすべての機能を有効にすると、サービスコントロールポリシー (SCP) を一部またはすべてのアカウントに適用できます。SCP は、各 を含むメンバーアカウントのエンティティのアクセス許可を制限します AWS アカウントのルートユーザー。Organizations と SCP の詳細については、AWS Organizations ユーザーガイドの「SCP の仕組み」を参照してください。
- セッションポリシー - セッションポリシーは、ロールまたはフェデレーションユーザーの一時的なセッションをプログラムで作成する際にパラメータとして渡す高度なポリシーです。結果として

セッションの権限は、ユーザーまたはロールのアイデンティティベースポリシーとセッションポリシーの共通部分になります。また、リソースベースのポリシーから権限が派生する場合もあります。これらのポリシーのいずれかを明示的に拒否した場合、権限は無効になります。詳細については、「IAM ユーザーガイド」の「セッションポリシー」を参照してください。

複数のポリシータイプ

1 つのリクエストに複数のタイプのポリシーが適用されると、結果として作成される権限を理解するのがさらに難しくなります。複数のポリシータイプが関与する場合にリクエストを許可するかどうか AWS を決定する方法については、IAM ユーザーガイドの「ポリシー評価ロジック」を参照してください。

AWS Fault Injection Service と IAM の連携方法

IAM を使用して AWS FIS へのアクセスを管理する前に、 AWS FIS で使用できる IAM 機能について学びます。

AWS Fault Injection Service で使用できる IAM 機能

IAM 機能	AWS FIS サポート
アイデンティティベースのポリシー	Yes
リソースベースのポリシー	No
ポリシーアクション	Yes
ポリシーリソース	はい
ポリシー条件キー (サービス固有)	はい
ACL	No
ABAC (ポリシー内のタグ)	はい
一時的な認証情報	Yes
プリンシパル権限	Yes
サービスロール	あり

IAM 機能	AWS FIS サポート
サービスリンクロール	はい

AWS FIS 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を把握するには、「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。

AWS FIS 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ポリシーをサポートする	Yes

アイデンティティベースポリシーは、IAM ユーザー、ユーザーのグループ、ロールなど、アイデンティティにアタッチできる JSON 許可ポリシードキュメントです。これらのポリシーは、ユーザーとロールが実行できるアクション、リソース、および条件をコントロールします。アイデンティティベース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 ポリシーの作成」を参照してください。

IAM アイデンティティベースのポリシーでは、許可または拒否するアクションとリソース、およびアクションを許可または拒否する条件を指定できます。プリンシパルは、それが添付されているユーザーまたはロールに適用されるため、アイデンティティベースのポリシーでは指定できません。JSON ポリシーで使用できるすべての要素については、「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。

AWS FIS のアイデンティティベースのポリシーの例

AWS FIS 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 Fault Injection Service ポリシーの例。

AWS FIS 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	No

リソースベースのポリシーは、リソースに添付する JSON ポリシードキュメントです。リソースベースのポリシーには例として、IAM ロールの信頼ポリシー や Amazon S3 バケットポリシー があ

げられます。リソースベースのポリシーをサポートするサービスでは、サービス管理者はポリシーを使用して特定のリソースへのアクセスを制御できます。ポリシーがアタッチされているリソースの場合、指定されたプリンシパルがそのリソースに対して実行できるアクションと条件は、ポリシーによって定義されます。リソースベースのポリシーでは、プリンシパルを指定する必要があります。プリンシパルには、アカウント、ユーザー、ロール、フェデレーティッドユーザー、または を含めることができます AWS のサービス。

クロスアカウントアクセスを有効にするには、アカウント全体、または別のアカウントの IAM エンティティをリソースベースのポリシーのプリンシパルとして指定します。リソースベースのポリシーにクロスアカウントのプリンシパルを追加しても、信頼関係は半分しか確立されない点に注意してください。プリンシパルとリソースが異なる にある場合 AWS アカウント、信頼されたアカウントの IAM 管理者は、プリンシパルエンティティ (ユーザーまたはロール) にリソースへのアクセス許可も付与する必要があります。IAM 管理者は、アイデンティティベースのポリシーをエンティティにアタッチすることで権限を付与します。ただし、リソースベースのポリシーで、同じアカウントのプリンシパルへのアクセス権が付与されている場合は、アイデンティティベースのポリシーをさらに付与する必要はありません。詳細については、「IAM ユーザーガイド」の「IAM でのクロスアカウントリソースアクセス」を参照してください。

AWS FIS のポリシーアクション

ポリシーアクションに対するサポート	はい

管理者は AWS JSON 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んなリソースにどんな条件でアクションを実行できるかということです。

JSON ポリシーの`Action`要素には、ポリシー内のアクセスを許可または拒否するために使用できるアクションが記述されます。ポリシーアクションの名前は通常、関連付けられた AWS API オペレーションと同じです。一致する API オペレーションのない許可のみのアクションなど、いくつかの例外があります。また、ポリシーに複数のアクションが必要なオペレーションもあります。これらの追加アクションは、依存アクションと呼ばれます。

このアクションは、関連付けられたオペレーションを実行するための権限を付与するポリシーで使用されます。

AWS FIS アクションのリストを確認するには、「サービス認証リファレンス」のAWS 「Fault Injection Service で定義されるアクション」を参照してください。

AWS FIS 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。

```
fis
```

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": [
    "fis:action1",
    "fis:action2"
]
```

ワイルドカード (*) を使用して複数アクションを指定できます。例えば、List という単語で始まるすべてのアクションを指定するには、次のアクションを含めます。

```
"Action": "fis:List*"
```

AWS FIS のポリシーリソース

ポリシーリソースに対するサポート	はい

管理者は AWS JSON 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どのような条件でアクションを実行できるかということです。

Resource JSON ポリシー要素は、アクションが適用されるオブジェクトを指定します。ステートメントには、Resource または NotResource要素を含める必要があります。ベストプラクティスとして、Amazon リソースネーム (ARN) を使用してリソースを指定します。これは、リソースレベルの許可と呼ばれる特定のリソースタイプをサポートするアクションに対して実行できます。

オペレーションのリスト化など、リソースレベルの権限をサポートしないアクションの場合は、ステートメントがすべてのリソースに適用されることを示すために、ワイルドカード (*) を使用します。

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

一部の AWS FIS API アクションは、複数のリソースをサポートします。複数リソースを単一ステートメントで指定するには、ARN をカンマで区切ります。

```
"Resource": [
    "resource1",
```

```
    "resource2"
]
```

AWS FIS リソースタイプとその ARNs」のAWS 「Fault Injection Service で定義されるリソースタイプ」を参照してください。 どのアクションで各リソースの ARN を指定できるかについては、AWS 「Fault Injection Service で定義されるアクション」を参照してください。

AWS FIS のポリシー条件キー

サービス固有のポリシー条件キーのサポート	はい

管理者は AWS JSON 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んなリソースにどんな条件でアクションを実行できるかということです。

`Condition` 要素 (または `Condition` ブロック) を使用すると、ステートメントが有効な条件を指定できます。`Condition` 要素はオプションです。イコールや未満などの 条件演算子 を使用して条件式を作成することで、ポリシーの条件とリクエスト内の値を一致させることができます。

1 つのステートメントに複数の `Condition` 要素を指定するか、1 つの `Condition` 要素に複数のキーを指定すると、 AWS は `AND` 論理演算子を使用してそれらを評価します。1 つの条件キーに複数の値を指定すると、 は論理`OR`オペレーションを使用して条件 AWS を評価します。ステートメントの権限が付与される前にすべての条件が満たされる必要があります。

条件を指定する際にプレースホルダー変数も使用できます。例えば IAM ユーザーに、IAM ユーザー名がタグ付けされている場合のみリソースにアクセスできる権限を付与することができます。詳細については、「IAM ユーザーガイド」の「IAM ポリシーの要素: 変数およびタグ」を参照してください。

AWS は、グローバル条件キーとサービス固有の条件キーをサポートします。すべての AWS グローバル条件キーを確認するには、「IAM ユーザーガイド」のAWS 「 グローバル条件コンテキストキー」を参照してください。

AWS FIS 条件キーのリストを確認するには、「サービス認証リファレンス」の「 AWS Fault Injection Service の条件キー」を参照してください。条件キーを使用できるアクションとリソースについては、AWS 「Fault Injection Service で定義されるアクション」を参照してください。

AWS FIS 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 Fault Injection Service ポリシーの例。

AWS FIS ACLs

ACL のサポート	No

アクセスコントロールリスト (ACL) は、どのプリンシパル (アカウントメンバー、ユーザー、またはロール) がリソースにアクセスするための許可を持つかをコントロールします。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。

AWS FIS での ABAC

ABAC のサポート (ポリシー内のタグ)	はい

属性ベースのアクセス制御 (ABAC) は、属性に基づいてアクセス許可を定義するアクセス許可戦略です。では AWS、これらの属性はタグ と呼ばれます。タグは、IAM エンティティ (ユーザーまたはロール) および多くの AWS リソースにアタッチできます。エンティティとリソースのタグ付けは、ABAC の最初の手順です。その後、プリンシパルのタグがアクセスしようとしているリソースのタグと一致した場合にオペレーションを許可するように ABAC ポリシーをします。

ABAC は、急成長する環境やポリシー管理が煩雑になる状況で役立ちます。

タグに基づいてアクセスを管理するには、`aws:ResourceTag/key-name`、`aws:RequestTag/key-name`、または `aws:TagKeys` の条件キーを使用して、ポリシーの条件要素でタグ情報を提供します。

サービスがすべてのリソースタイプに対して 3 つの条件キーすべてをサポートする場合、そのサービスの値ははいです。サービスが一部のリソースタイプに対してのみ 3 つの条件キーのすべてをサポートする場合、値は「部分的」になります。

ABAC の詳細については、IAM ユーザーガイド の「ABAC とは?」を参照してください。ABAC をセットアップするステップを説明するチュートリアルについては、「IAM ユーザーガイド」の「属性に基づくアクセスコントロール (ABAC) を使用する」を参照してください。

リソースのタグに基づいてリソースへのアクセスを制限するため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、「例 : タグを使用してリソースの使用量を制御する」を参照してください。

AWS FIS 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	はい

一部の AWS のサービス は、一時的な認証情報を使用してサインインすると機能しません。一時的な認証情報 AWS のサービス を使用する などの詳細については、IAM ユーザーガイドのAWS のサービス 「IAM と連携する 」を参照してください。

ユーザー名とパスワード以外の AWS Management Console 方法で にサインインする場合、一時的な認証情報を使用します。例えば、会社の Single Sign-On (SSO) リンク AWS を使用して にアクセスすると、そのプロセスによって一時的な認証情報が自動的に作成されます。また、ユーザーとしてコンソールにサインインしてからロールを切り替える場合も、一時的な認証情報が自動的に作成されます。ロールの切り替えに関する詳細については、「IAM ユーザーガイド」の「ロールへの切り替え (コンソール)」を参照してください。

一時的な認証情報は、 AWS CLI または AWS API を使用して手動で作成できます。その後、これらの一時的な認証情報を使用して . AWS recommends にアクセスできます AWS。この際、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することをお勧めします。詳細については、「IAM の一時的セキュリティ認証情報」を参照してください。

AWS FIS のクロスサービスプリンシパル許可

フォワードアクセスセッション (FAS) をサポート	はい

IAM ユーザーまたはロールを使用して でアクションを実行すると AWS、プリンシパルと見なされます。一部のサービスを使用する際に、アクションを実行することで、別のサービスの別のアクションがトリガーされることがあります。FAS は、 を呼び出すプリンシパルのアクセス許可を AWS のサービス、ダウンストリームサービス AWS のサービス へのリクエストのリクエストと組み合わせて使用します。FAS リクエストは、サービスが他の AWS のサービス またはリソースとのやり取りを完了する必要があるリクエストを受け取った場合にのみ行われます。この場合、両方のアクションを実行するためのアクセス許可が必要です。FAS リクエストを行う際のポリシーの詳細については、「転送アクセスセッション」を参照してください。

AWS FIS のサービスロール

サービスロールに対するサポート	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 のサービスにアクセス許可を委任するロールの作成」を参照してください。

AWS FIS 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	はい

サービスにリンクされたロールは、 にリンクされたサービスロールの一種です AWS のサービス。サービスは、ユーザーに代わってアクションを実行するロールを引き受けることができます。サービスにリンクされたロールは に表示され AWS アカウント 、サービスによって所有されます。IAM 管理者は、サービスリンクロールの許可を表示できますが、編集することはできません。

AWS FIS 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、「」を参照してください AWS Fault Injection Service のサービスにリンクされたロールを使用する。

AWS Fault Injection Service ポリシーの例

デフォルトでは、ユーザーとロールには AWS FIS 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。また、、 AWS Command Line Interface （ AWS CLI ） AWS Management Console、または AWS API を使用してタスクを実行することはできません。IAM 管理者は、リソースで必要なアクションを実行するための権限をユーザーに付与する IAM ポリシーを作成できます。その後、管理者はロールに IAM ポリシーを追加し、ユーザーはロールを引き受けることができます。

これらサンプルの JSON ポリシードキュメントを使用して、IAM アイデンティティベースのポリシーを作成する方法については、「IAM ユーザーガイド」の「IAM ポリシーの作成」を参照してください。

各リソースタイプの ARN の形式など、 AWS FIS で定義されるアクションとリソースタイプの詳細については、「サービス認証リファレンス」のAWS 「Fault Injection Service 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。 ARNs

内容

ポリシーのベストプラクティス

ID ベースのポリシーは、ユーザーのアカウントで誰かが AWS FIS リソースを作成、アクセス、または削除できるどうかを決定します。これらのアクションを実行すると、 AWS アカウントに料金が発生する可能性があります。アイデンティティベースポリシーを作成したり編集したりする際には、以下のガイドラインと推奨事項に従ってください:

- AWS 管理ポリシーを開始し、最小特権のアクセス許可に移行する – ユーザーとワークロードにアクセス許可を付与するには、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して、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「AWS ジョブ機能の管理ポリシー」を参照してください。
- 最小特権を適用する – IAM ポリシーで許可を設定する場合は、タスクの実行に必要な許可のみを付与します。これを行うには、特定の条件下で特定のリソースに対して実行できるアクションを定義します。これは、最小特権アクセス許可とも呼ばれています。IAM を使用して許可を適用する方法の詳細については、「IAM ユーザーガイド」の「IAM でのポリシーとアクセス許可」を参照してください。
- IAM ポリシーで条件を使用してアクセスをさらに制限する - ポリシーに条件を追加して、アクションやリソースへのアクセスを制限できます。例えば、ポリシー条件を記述して、すべてのリクエストを SSL を使用して送信するように指定できます。条件を使用して、 などの特定の を通じてサービスアクションが使用される場合に AWS のサービス、サービスアクションへのアクセスを許

可することもできます AWS CloudFormation。詳細については、「IAM ユーザーガイド」の [IAM JSON policy elements: Condition] (IAM JSON ポリシー要素: 条件) を参照してください。

- IAM Access Analyzer を使用して IAM ポリシーを検証し、安全で機能的な権限を確保する - IAM Access Analyzer は、新規および既存のポリシーを検証して、ポリシーが IAM ポリシー言語 (JSON) および IAM のベストプラクティスに準拠するようにします。IAM アクセスアナライザーは 100 を超えるポリシーチェックと実用的な推奨事項を提供し、安全で機能的なポリシーの作成をサポートします。詳細については、「IAM ユーザーガイド」の「IAM Access Analyzer ポリシーの検証」を参照してください。
- 多要素認証 (MFA) を要求する – で IAM ユーザーまたはルートユーザーを必要とするシナリオがある場合は AWS アカウント、セキュリティを強化するために MFA を有効にします。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 保護 API アクセスの設定」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。

例: AWS FIS コンソールの使用

AWS Fault Injection Service 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 AWS FIS リソースの詳細を一覧表示および表示できます AWS アカウント。最小限必要な許可よりも制限が厳しいアイデンティティベースのポリシーを作成すると、そのポリシーを持つエンティティ (ユーザーまたはロール) に対してコンソールが意図したとおりに機能しません。

AWS CLI または AWS API のみを呼び出すユーザーには、最小限のコンソールアクセス許可を付与する必要はありません。代わりに、実行しようとしている API オペレーションに一致するアクションのみへのアクセスが許可されます。

次のポリシー例では、 AWS FIS コンソールを使用してすべての AWS FIS リソースを一覧表示および表示するアクセス許可を付与しますが、作成、更新、削除することはできません。また、実験テンプレートで指定できるすべての AWS FIS アクションで使用可能なリソースを表示するアクセス許可も付与します。

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

```
            "Effect": "Allow",
            "Action": [
                "fis:List*",
                "fis:Get*"
            ],
            "Resource": "*"
        },
        {
            "Sid": "AdditionalReadOnlyActions",
            "Effect": "Allow",
            "Action": [
                "ssm:Describe*",
                "ssm:Get*",
                "ssm:List*",
                "ec2:DescribeInstances",
                "rds:DescribeDBClusters",
                "ecs:DescribeClusters",
                "ecs:ListContainerInstances",
                "eks:DescribeNodegroup",
                "cloudwatch:DescribeAlarms",
                "iam:ListRoles"
            ],
            "Resource": "*"
        },
        {
            "Sid": "PermissionsToCreateServiceLinkedRole",
            "Effect": "Allow",
            "Action": "iam:CreateServiceLinkedRole",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "iam:AWSServiceName": "fis.amazonaws.com"
                }
            }
        }
    ]
}
```

例: 使用可能な AWS FIS アクションを一覧表示する

次のポリシーは、使用可能な AWS FIS アクションを一覧表示するアクセス許可を付与します。

```
{
```

```
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "fis:ListActions"
            ],
            "Resource": "arn:aws:fis:*:*:action/*"
        }
    ]
}
```

例 : 特定のアクションの実験テンプレートを作成する

次のポリシーでは、アクション `aws:ec2:stop-instances` の実験テンプレートを作成する権限を付与します。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "PolicyExample",
      "Effect": "Allow",
      "Action": [
          "fis:CreateExperimentTemplate"
      ],
      "Resource": [
        "arn:aws:fis:*:*:action/aws:ec2:stop-instances",
        "arn:aws:fis:*:*:experiment-template/*"
      ]
    },
    {
      "Sid": "PolicyPassRoleExample",
      "Effect": "Allow",
      "Action": [
          "iam:PassRole"
      ],
      "Resource": [
        "arn:aws:iam::account-id:role/role-name"
      ]
    }
  ]
}
```

例：実験を開始する

次のポリシーは、指定した IAM ロールと実験テンプレートを使用して実験を開始するアクセス権限を付与します。また、 AWS FIS がユーザーに代わってサービスにリンクされたロールを作成することもできます。詳細については、「AWS Fault Injection Service のサービスにリンクされたロールを使用する」を参照してください。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "PolicyExample",
      "Effect": "Allow",
      "Action": [
          "fis:StartExperiment"
      ],
      "Resource": [
        "arn:aws:fis:*:*:experiment-template/experiment-template-id",
        "arn:aws:fis:*:*:experiment/*"
      ]
    },
    {
        "Sid": "PolicyExampleforServiceLinkedRole",
        "Effect": "Allow",
        "Action": "iam:CreateServiceLinkedRole",
        "Resource": "*",
        "Condition": {
            "StringEquals": {
                "iam:AWSServiceName": "fis.amazonaws.com"
            }
        }
    }
  ]
}
```

例：タグを使用してリソースの使用量を制御する

次のポリシーは、Purpose=Test タグを持つ実験テンプレートから実験を実行するアクセス権限を付与します。実験テンプレートを作成または変更するアクセス権限は付与されず、指定したタグを持たないテンプレートを使用して実験を実行することはできません。

```
{
```

```
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "fis:StartExperiment",
            "Resource": "arn:aws:fis:*:*:experiment-template/*",
            "Condition": {
                "StringEquals": {
                    "aws:ResourceTag/Purpose": "Test"
                }
            }
        }
    ]
}
```

例 : 特定のタグを持つ実験テンプレートを削除する

次のポリシーでは、Purpose=Test タグ付きの実験テンプレートを削除する権限を付与します。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "fis:DeleteExperimentTemplate"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:ResourceTag/Purpose": "Test"
                }
            }
        }
    ]
}
```

例: ユーザーにそれぞれのアクセス権限の表示を許可する

この例では、ユーザーアイデンティティにアタッチされたインラインおよびマネージドポリシーの表示を IAM ユーザーに許可するポリシーの作成方法を示します。このポリシーには、コンソールで、

または AWS CLI または AWS API を使用してプログラムでこのアクションを実行するアクセス許可が含まれています。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ViewOwnUserInfo",
            "Effect": "Allow",
            "Action": [
                "iam:GetUserPolicy",
                "iam:ListGroupsForUser",
                "iam:ListAttachedUserPolicies",
                "iam:ListUserPolicies",
                "iam:GetUser"
            ],
            "Resource": ["arn:aws:iam::*:user/${aws:username}"]
        },
        {
            "Sid": "NavigateInConsole",
            "Effect": "Allow",
            "Action": [
                "iam:GetGroupPolicy",
                "iam:GetPolicyVersion",
                "iam:GetPolicy",
                "iam:ListAttachedGroupPolicies",
                "iam:ListGroupPolicies",
                "iam:ListPolicyVersions",
                "iam:ListPolicies",
                "iam:ListUsers"
            ],
            "Resource": "*"
        }
    ]
}
```

例: **ec2:InjectApiError** の条件キーを使用します。

次の例のポリシーでは、`ec2:FisTargetArns` 条件キーを使用してターゲットリソースの範囲を制限します。このポリシーは、 AWS FIS アクション `aws:ec2:api-insufficient-instance-capacity-error`および を許可します`aws:ec2:asg-insufficient-instance-capacity-error`。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "ec2:InjectApiError",
            "Resource": "*",
            "Condition": {
                "ForAllValues:StringEquals": {
                    "ec2:FisActionId": [
                        "aws:ec2:api-insufficient-instance-capacity-error",
                    ],
                    "ec2:FisTargetArns": [
                        "arn:aws:iam:*:*:role:role-name"
                    ]
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": "ec2:InjectApiError",
            "Resource": "*",
            "Condition": {
                "ForAllValues:StringEquals": {
                    "ec2:FisActionId": [
                        "aws:ec2:asg-insufficient-instance-capacity-error"
                    ],
                    "ec2:FisTargetArns": [

 "arn:aws:autoscaling:*:*:autoScalingGroup:uuid:autoScalingGroupName/asg-name"
                    ]
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": "autoscaling:DescribeAutoScalingGroups",
            "Resource": "*"
        }
    ]
}
```

例: **aws:s3:bucket-pause-replication** の条件キーを使用します。

次のポリシー例では、 S3:IsReplicationPauseRequest条件キーを使用して、 AWS FIS アクション のコンテキストで AWS FIS によって使用されるGetReplicationConfiguration場合にのみ、 PutReplicationConfigurationおよび を許可しますaws:s3:bucket-pause-replication。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "S3:PauseReplication"
            ],
            "Resource": "arn:aws:s3:::mybucket",
            "Condition": {
                "StringEquals": {
                    "s3:DestinationRegion": "region"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "S3:PutReplicationConfiguration",
                "S3:GetReplicationConfiguration"
            ],
            "Resource": "arn:aws:s3:::mybucket",
            "Condition": {
                "BoolIfExists": {
                    "s3:IsReplicationPauseRequest": "true"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "S3:ListBucket"
            ],
            "Resource": "arn:aws:s3:::*"
        },
        {
```

```
            "Effect": "Allow",
            "Action": [
                "tag:GetResources"
            ],
            "Resource": "*"
        }
    ]
    }
```

AWS Fault Injection Service のサービスにリンクされたロールを使用する

AWS Fault Injection Service 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、 AWS FIS 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、 AWS FIS による事前定義済みのロールであり、ユーザーに代わってサービスから AWS の他のサービスを呼び出すために必要なすべての権限を備えています。

サービスにリンクされたロールを使用すると、実験のモニタリングとリソースの選択を管理するために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、 AWS FIS の設定が簡単になります。AWS FIS 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、特に定義されている場合を除き、 AWS FIS のみがそのロールを引き受けることができます。定義したアクセス許可には、信頼ポリシーと許可ポリシーが含まれます。こ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。

サービスにリンクされたロールに加えて、実験テンプレートでターゲットとして指定したリソースを変更するアクセス権限を付与する IAM ロールも指定する必要があります。詳細については、「AWS FIS 実験用の IAM ロール」を参照してください。

サービスリンクロールは、関連する リソースを削除した後でしか削除できません。これにより、リソースにアクセスするためのアクセス許可を誤って削除することがないため、 AWS FIS リソースが保護されます。

AWS FIS のサービスにリンクされたロールのアクセス許可

AWS FIS は、 という名前のサービスにリンクされたロールAWSServiceRoleForFISを使用して、実験のモニタリングとリソース選択を管理できるようにします。

AWSServiceRoleForFIS サービスにリンクされたロールは、次のサービスを信頼してロールを引き受けます。

- `fis.amazonaws.com`

AWSServiceRoleForFIS サービスにリンクされたロールは、 マネージドポリシー AmazonFISServiceRole ポリシー を使用します。このポリシーにより、 AWS FIS は実験のモニタリングとリソース選択を管理できます。詳細については、「 マネージドポリシーリファレンス」の AmazonFISServiceRole ポリシー」を参照してください。 AWS

サービスリンクロールの作成、編集、削除を IAM エンティティ (ユーザー、グループ、ロールなど) に許可するには、アクセス許可を設定する必要があります。AWSServiceRoleForFIS サービスにリンクされたロールを正常に作成するには、 で AWS FIS を使用する IAM アイデンティティに必要なアクセス許可が必要です。必要な許可を付与するには、次のポリシーを IAM アイデンティティにアタッチします。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "iam:CreateServiceLinkedRole",
            "Resource": "*",
            "Condition": {
                "StringLike": {
                    "iam:AWSServiceName": "fis.amazonaws.com"
                }
            }
        }
    ]
}
```

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権限」を参照してください。

AWS FIS のサービスにリンクされたロールを作成する

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。 AWS Management Console、、 AWS CLIまたは AWS API で AWS FIS 実験を開始すると、 AWS FIS 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。

このサービスリンクロールを削除した後で再度作成する必要が生じた場合は、同じ方法でアカウントにロールを再作成できます。 AWS FIS 実験を開始すると、 AWS FIS によってサービスにリンクされたロールが再度作成されます。

AWS FIS のサービスにリンクされたロールを編集する

AWS FIS では、AWSServiceRoleForFIS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した後は、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 を使用した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。

AWS FIS のサービスにリンクされたロールを削除する

サービスリンクロールが必要な機能またはサービスが不要になった場合には、そのロールを削除することをお勧めします。そうすることで、積極的にモニタリングまたは保守されていない未使用のエンティティを排除できます。ただし、手動で削除する前に、サービスリンクロールのリソースをクリーンアップする必要があります。

> **Note**
>
> リソースをクリーンアップしようとしたときに AWS FIS サービスがロールを使用している場合、クリーンアップが失敗する可能性があります。失敗した場合は、数分待ってから操作を再試行してください。

で使用される AWS FIS リソースをクリーンアップするには AWSServiceRoleForFIS

どの実験も現在実行されていないことを確認してください。必要に応じて、実験を中止してください。詳細については、「実験を中止する」を参照してください。

サービスにリンクされたロールを IAM で手動削除するには

IAM コンソール、、または AWS API を使用して AWS CLI、AWSServiceRoleForFISサービスにリンクされたロールを削除します。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 FIS サービスにリンクされたロールでサポートされているリージョン

AWS FIS は、サービスが利用可能なすべてのリージョンでサービスにリンクされたロールの使用をサポートしています。詳細については、「AWS Fault Injection Service エンドポイントとクォータ」を参照してください。

AWSAWS Fault Injection Service の マネージドポリシー

AWS 管理ポリシーは、 によって作成および管理されるスタンドアロンポリシーです AWS。 AWS 管理ポリシーは、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ように設計されているため、ユーザー、グループ、ロールにアクセス許可の割り当てを開始できます。

AWS 管理ポリシーは、すべての AWS お客様が使用できるため、特定のユースケースに対して最小特権のアクセス許可を付与しない場合があることに注意してください。ユースケース別にカスタマーマネージドポリシーを定義して、マネージドポリシーを絞り込むことをお勧めします。

AWS 管理ポリシーで定義されているアクセス許可は変更できません。が AWS 管理ポリシーで定義されたアクセス許可 AWS を更新すると、ポリシーがアタッチされているすべてのプリンシパル ID (ユーザー、グループ、ロール) が更新されます。 は、新しい AWS のサービス が起動されたとき、または既存のサービスで新しい API AWS オペレーションが使用可能になったときに、 AWS 管理ポリシーを更新する可能性が最も高くなります。

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 マネージドポリシー: AmazonFISServiceRole ポリシー

このポリシーは、 という名前のサービスにリンクされたロールにアタッチAWSServiceRoleForFISされ、 AWS FIS が実験のモニタリングとリソース選択を管理できるようにします。詳細については、「AWS Fault Injection Service のサービスにリンクされたロールを使用する」を参照してください。

AWS マネージドポリシー： AWSFaultInjectionSimulatorEC2Access

実験ロールでこのポリシーを使用して、Amazon EC2 の AWS FIS アクションを使用する実験を実行するための FIS アクセス許可を付与します。 AWS Amazon EC2 詳細については、「the section called “実験ロール”」を参照してください。

このポリシーのアクセス許可を確認するには、「 マネージドポリシーリファレンスAWSFaultInjectionSimulatorEC2Access」の「」を参照してください。 AWS

AWS マネージドポリシー： AWSFaultInjectionSimulatorECSAccess

実験ロールでこのポリシーを使用して、Amazon ECS の AWS FIS アクションを使用する実験を実行するためのアクセス許可を FIS に付与します。 AWS詳細については、「the section called “実験ロール”」を参照してください。

このポリシーのアクセス許可を確認するには、「 マネージドポリシーリファレンスAWSFaultInjectionSimulatorECSAccess」の「」を参照してください。 AWS

AWS マネージドポリシー： AWSFaultInjectionSimulatorEKSAccess

実験ロールでこのポリシーを使用して、Amazon EKS の AWS FIS アクションを使用する実験を実行するためのアクセス許可を FIS に付与します。 AWS詳細については、「the section called “実験ロール”」を参照してください。

このポリシーのアクセス許可を確認するには、「 マネージドポリシーリファレンスAWSFaultInjectionSimulatorEKSAccess」の「」を参照してください。 AWS

AWS マネージドポリシー： AWSFaultInjectionSimulatorNetworkAccess

このポリシーを実験ロールで使用して、 AWS FIS AWS ネットワークアクション を使用する実験を実行するアクセス許可を FIS に付与します。詳細については、「the section called “実験ロール”」を参照してください。

このポリシーのアクセス許可を確認するには、「 マネージドポリシーリファレンスAWSFaultInjectionSimulatorNetworkAccess」の「」を参照してください。 AWS

AWS マネージドポリシー： AWSFaultInjectionSimulatorRDSAccess

実験ロールでこのポリシーを使用して、Amazon RDS の AWS FIS アクションを使用する実験を実行するためのアクセス許可を FIS に付与します。 AWS詳細については、「the section called “実験ロール”」を参照してください。

このポリシーのアクセス許可を確認するには、「 マネージドポリシーリファレンスAWSFaultInjectionSimulatorRDSAccess」の「」を参照してください。 AWS

AWS マネージドポリシー： AWSFaultInjectionSimulatorSSMAccess

実験ロールでこのポリシーを使用して、Systems Manager の AWS FIS アクションを使用する実験を実行するためのアクセス許可を FIS に付与します。 AWS詳細については、「the section called “実験ロール”」を参照してください。

このポリシーのアクセス許可を確認するには、「 マネージドポリシーリファレンスAWSFaultInjectionSimulatorSSMAccess」の「」を参照してください。 AWS

AWSAWS マネージドポリシーの FIS 更新

AWS FIS の AWS マネージドポリシーの更新に関する詳細を、このサービスがこれらの変更の追跡を開始した以降の分について表示します。

変更	説明	日付
AWSFaultInjectionSimulatorECSAccess - 既存ポリシーへの更新	AWS FIS が ECS ターゲットを解決できるようにするアクセス許可を追加しました。	2024 年 1 月 25 日
AWSFaultInjectionSimulatorNetworkAccess – 既存ポリシーへの更新	AWS FIS が aws:network:route-table-disrupt-cross-region-connectivityおよび aws:network:transit-gateway-disrupt-cross-region-connectivityアクションを使用して実験を実行できるようにするアクセス許可を追加しました。	2024 年 1 月 25 日
AWSFaultInjectionSimulatorEC2Access – 既存ポリシーへの更新	AWS FIS が EC2 インスタンスを解決できるようにするアクセス許可を追加しました。	2023 年 11 月 13 日
AWSFaultInjectionSimulatorEKSAccess – 既存ポリシーへの更新	AWS FIS が EKS ターゲットを解決できるようにするアクセス許可を追加しました。	2023 年 11 月 13 日
AWSFaultInjectionSimulatorRDSAccess – 既存ポリシーへの更新	AWS FIS が RDS ターゲットを解決できるようにするアクセス許可を追加しました。	2023 年 11 月 13 日
AWSFaultInjectionSimulatorEC2Access – 既存ポリシーへの更新	AWS FIS が EC2 インスタンスで SSM ドキュメントを実行し、EC2 インスタンスを終了できるようにするアクセス許可を追加しました。	2023 年 6 月 2 日
AWSFaultInjectionSimulatorSSMAccess – 既存ポリシーへの更新	AWS FIS が EC2 インスタンスで SSM ドキュメントを実行できるようにするアクセス許可を追加しました。	2023 年 6 月 2 日
AWSFaultInjectionSimulatorECSAccess – 既存ポリシーへの更新	AWS FIS が新しいaws:ecs:taskアクションを使用して実験を実行できる	2023 年 6 月 1 日

変更	説明	日付
	ようにするアクセス許可を追加しました。	
AWSFaultInjectionSimulatorEKSAccess – 既存ポリシーへの更新	AWS FIS が新しいaws:eks:podアクションを使用して実験を実行できるようにするアクセス許可を追加しました。	2023 年 6 月 1 日
AWSFaultInjectionSimulatorEC2Access - 新しいポリシー	AWS FIS が Amazon EC2 の AWS FIS アクションを使用する実験を実行できるようにするポリシーを追加しました。	2022 年 10 月 26 日
AWSFaultInjectionSimulatorECSAccess - 新しいポリシー	AWS FIS が Amazon ECS の AWS FIS アクションを使用する実験を実行できるようにするポリシーを追加しました。	2022 年 10 月 26 日
AWSFaultInjectionSimulatorEKSAccess - 新しいポリシー	AWS FIS が Amazon EKS の AWS FIS アクションを使用する実験を実行できるようにするポリシーを追加しました。	2022 年 10 月 26 日
AWSFaultInjectionSimulatorNetworkAccess - 新しいポリシー	AWS FIS が AWS FIS ネットワークアクションを使用する実験を実行できるようにするポリシーを追加しました。	2022 年 10 月 26 日
AWSFaultInjectionSimulatorRDSAccess - 新しいポリシー	AWS FIS が Amazon RDS の AWS FIS アクションを使用する実験を実行できるようにするポリシーを追加しました。	2022 年 10 月 26 日
AWSFaultInjectionSimulatorSSMAccess - 新しいポリシー	AWS FIS が Systems Manager の AWS FIS アクションを使用する実験を実行できるようにするポリシーを追加しました。	2022 年 10 月 26 日

変更	説明	日付
AmazonFISServiceRole ポリシー — 既存のポリシーの更新	AWS FIS がサブネットを記述できるようにするアクセス許可を追加しました。	2022 年 10 月 26 日
AmazonFISServiceRole ポリシー — 既存のポリシーの更新	AWS FIS が EKS クラスターを記述できるようにするアクセス許可を追加しました。	2022 年 7 月 7 日
AmazonFISServiceRole ポリシー — 既存のポリシーの更新	AWS FIS がクラスター内のタスクを一覧表示および記述できるようにするアクセス許可を追加しました。	2022 年 2 月 7 日
AmazonFISServiceRole ポリシー — 既存のポリシーの更新	`events:DescribeRule` アクションの `events:ManagedBy` 条件を削除しました。	2022 年 1 月 6 日
AmazonFISServiceRole ポリシー — 既存のポリシーの更新	停止条件で使用される CloudWatch アラームの履歴を AWS FIS が取得できるようにするアクセス許可を追加しました。	2021 年 6 月 30 日
AWS FIS が変更の追跡を開始しました	AWS FIS が AWS マネージドポリシーの変更の追跡を開始	2021 年 3 月 1 日

AWS Fault Injection Service 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 Fault Injection Service は、 AWS グローバルネットワークセキュリティで保護されています。 AWS セキュリティサービスと がインフラストラクチャ AWS を保護する方法については、AWS 「 クラウドセキュリティ」を参照してください。インフラストラクチャセキュリティのベストプラクティスを使用して AWS 環境を設計するには、「セキュリティの柱 AWS Well-Architected Framework」の「Infrastructure Protection」を参照してください。

が AWS 公開した API コールを使用して、ネットワーク経由で AWS FIS にアクセスします。クライアントは以下をサポートする必要があります:

- Transport Layer Security (TLS)。TLS 1.2 は必須で TLS 1.3 がお勧めです。

- DHE (楕円ディフィー・ヘルマン鍵共有) や ECDHE (楕円曲線ディフィー・ヘルマン鍵共有) などの完全前方秘匿性 (PFS) による暗号スイート。これらのモードは、Java 7 以降など、ほとんどの最新システムでサポートされています。

また、リクエストには、アクセスキー ID と、IAM プリンシパルに関連付けられているシークレットアクセスキーを使用して署名する必要があります。または、AWS Security Token Service (AWS STS) を使用して、一時的なセキュリティ認証情報を生成し、リクエストに署名することもできます。

インターフェイス VPC エンドポイント (AWS PrivateLink) を使用して AWS FIS にアクセスする

インターフェイス VPC エンドポイント を作成することで、VPC と AWS Fault Injection Service 間のプライベート接続を確立できます。VPC エンドポイントは、インターネットゲートウェイAWS PrivateLink、NAT デバイス、VPN 接続、 AWS Direct Connect 接続のいずれも必要とせずに AWS FIS APIs にプライベートにアクセスできるテクノロジーである を利用しています。VPC 内のインスタンスは、パブリック IP アドレスがなくても AWS FIS APIs。

各インターフェイスエンドポイントは、サブネット内の 1 つ以上の Elastic Network Interface によって表されます。

詳細については、「 AWS PrivateLink ガイド」の「 AWS のサービス によるアクセス AWS PrivateLink」を参照してください。

AWS FIS VPC エンドポイントに関する考慮事項

AWS FIS のインターフェイス VPC エンドポイントを設定する前に、「 AWS PrivateLink ガイド」の「インターフェイス VPC エンドポイント AWS のサービス を使用して にアクセスする」を参照してください。

AWS FIS は、VPC からのすべての API アクションの呼び出しをサポートしています。

AWS FIS 用のインターフェイス VPC エンドポイントを作成する

AWS FIS サービスの VPC エンドポイントは、Amazon VPC コンソールまたは AWS Command Line Interface () を使用して作成できますAWS CLI。詳細については、『AWS PrivateLink ガイド』の「Create a VPC endpoint (VPC エンドポイントを作成)を参照してください。

次のサービス名を使用して AWS FIS の VPC エンドポイントを作成します: `com.amazonaws.region.fis`。

エンドポイントのプライベート DNS を有効にすると、 など、リージョンのデフォルトの DNS 名を使用して AWS FIS に API リクエストを実行できます`fis.us-east-1.amazonaws.com`。

AWS FIS 用の VPC エンドポイントポリシーを作成する

AWS FIS へのアクセスを制御するエンドポイントポリシーを VPC エンドポイントにアタッチできます。このポリシーでは、以下の情報を指定します。

- アクションを実行できるプリンシパル。
- 実行可能なアクション。
- このアクションを実行できるリソース。

詳細については、『AWS PrivateLink ガイド』の「Control access to VPC endpoints using endpoint policies (エンドポイントポリシーを使用して VPC エンドポイントへのアクセスをコントロールする)」を参照してください。

例: 特定の AWS FIS アクションの VPC エンドポイントポリシー

次の VPC エンドポイントポリシーは、すべてのリソースに対するリストされた AWS FIS アクションへのアクセスをすべてのプリンシパルに付与します。

```
{
   "Statement":[
      {
         "Effect":"Allow",
         "Action":[
            "fis:ListExperimentTemplates",
            "fis:StartExperiment",
            "fis:StopExperiment",
            "fis:GetExperiment"
         ],
         "Resource":"*",
         "Principal":"*"
      }
   ]
}
```

例: 特定の からのアクセスを拒否する VPC エンドポイントポリシー AWS アカウント

次の VPC エンドポイントポリシーは、すべてのアクションとリソースへの指定された AWS アカウント アクセスを拒否しますが、他のすべてのアクションとリソース AWS アカウント へのアクセスを許可します。

```
{
   "Statement":[
      {
         "Effect": "Allow",
         "Action": "*",
         "Resource": "*",
         "Principal": "*"
      },
      {
         "Effect":"Deny",
         "Action": "*",
         "Resource": "*",
         "Principal": {
           "AWS": [ "123456789012" ]
         }
      }
   ]
}
```

AWS FIS リソースのタグ付け

タグは、ユーザーまたはAWSがAWSリソースに割り当てるメタデータラベルです。各タグは、キーと値から構成されます。ユーザーが割り当てるタグは、ユーザーがキーと値を定義します。例えば、1 つのリソースのキーを `purpose` と定義し、値を `test` と定義します。

タグは、以下のことに役立ちます。

- AWS リソースの特定と整理。多くの AWS のサービスではタグ付けがサポートされるため、さまざまなサービスからリソースに同じタグを割り当てて、リソースの関連を示すことができます。
- AWS リソースへのアクセスを制御します。詳細については、 IAM ユーザーガイドの「タグを使用したアクセス制御」を参照してください。

タグ指定の制限

AWS FIS リソースのタグには、以下のような基本制限があります。

- リソースに割り当てることができるタグの最大数 :50
- キーの最大長: 128 文字 (ユニコード)
- 値の最大長: 256 文字 (ユニコード)
- キーと値に有効な文字は次の通りです: a ~ z、A ~ Z、0 ~ 9、スペース、特殊文字 (_ . : / = + - @)
- キーと値では大文字と小文字が区別されます
- `aws:` はキーのプレフィックスとして使用できません。AWS 用に予約済みです。

タグを操作する

以下の AWS Fault Injection Service (AWS FIS) のリソースがタグ付けをサポートしています。

- アクション
- 実験
- 実験テンプレート

コンソールで、実験用のタグと実験テンプレートを操作できます。詳細については、次を参照してください。

- 実験にタグを付けるには
- 実験テンプレートにタグ付けする

アクション、実験、実験テンプレートのタグは以下の AWS CLI コマンドで操作できます。

- tag-resource - リソースにタグを追加します。
- untag-resource - リソースからタグを削除します。
- list-tags-for-resource – 特定のリソースのタグを一覧表示します。

AWS Fault Injection Service のクォータと制限

AWS アカウント には、サービスごとに、以前は制限と呼ばれていたデフォルトのクォータがあります AWS 。特に明記されていない限り、クォータは地域固有です。一部のクォータについては引き上げをリクエストできますが、一部のクォータについてはリクエストできません。

AWS FIS のクォータを表示するには、Service Quotas コンソール を開きます。ナビゲーションペインで、AWS のサービスを選択し、[AWS Fault Injection Service] を選択します。

クォータの引き上げをリクエストするには、「Service Quotas ユーザーガイド」の「クォータ引き上げリクエスト」を参照してください。

には、 AWS FIS に関連する次のクォータ AWS アカウント があります。

名前	デフォルト	引き上げ可能	説明
アクション期間 (時間)	サポートされている各リージョン: 12	いいえ	このアカウントで現在のリージョンにおいて 1 つのアクションの実行に使用できる最大時間。
実験テンプレートあたりのアクション	サポートされている各リージョン: 20	いいえ	このアカウントで現在のリージョンにおいて作成できる実験テンプレート内のアクションの最大数。
アクティブな実験	サポートされている各リージョン : 5	いいえ	このアカウントで現在のリージョンにおいて同時に実行できるアクティブな実験の最大数。

名前	デフォルト	引き上げ可能	説明
完了した実験データの保持日数	サポートされている各リージョン: 120	いいえ	現在のリージョンで、このアカウントで完了した実験に関するデータを AWS FIS が保持できる最大日数。
実験期間 (時間)	サポートされている各リージョン: 12	いいえ	このアカウントで現在のリージョンにおいて 1 つの実験の実行に使用できる最大時間。
[実験テンプレート]	サポートされている各リージョン: 500	いいえ	このアカウントで現在のリージョンにおいて作成できる実験テンプレートの最大数。
aws:network:route-table-disrupt-cross-region-connectivity のマネージドプレフィックスリストの最大数	サポートされている各リージョン: 15	いいえ	アクションごとに aws:network:route-table-disrupt-cross-region-connectivity will allow マネージドプレフィックスリストの最大数。
aws:network:route-table-disrupt-cross-region-connectivity のルートテーブルの最大数	サポートされている各リージョン: 10	いいえ	アクションごとに aws:network:route-table-disrupt-cross-region-connectivity will が許可するルートテーブルの最大数。

名前	デフォルト	引き上げ可能	説明
aws:network:route-table-disrupt-cross-region-connectivity のルートの最大数	サポートされている各リージョン: 200	いいえ	アクションごとに aws:network:route-table-disrupt-cross-region-connectivity will が許可するルートの最大数。
実験あたりの並列アクション	サポートされている各リージョン: 10	いいえ	このアカウントで現在のリージョンにおいて並行して実行できる実験のアクションの最大数。
実験テンプレートあたりの停止条件	サポートされている各リージョン：5	いいえ	このアカウントで現在のリージョンにおいて実験テンプレートに追加できる停止条件の最大数。
aws:ec2:asg-insufficient-instance-capacity-error のターゲットの Auto Scaling グループ	サポートされている各リージョン：5	はい	実験ごとに、タグを使用してターゲットを特定するときに aws:ec2:asg-insufficient-instance-capacity-error can がターゲットにできる Auto Scaling グループの最大数。

名前	デフォルト	引き上げ可能	説明
aws:s3:bucket-pause-replication のターゲットのバケット	サポートされている各リージョン: 20	はい	実験ごとに、タグを使用してターゲットを特定するときに aws:s3:bucket-pause-replication can がターゲットにできる S3 バケットの最大数。
aws:ecs:drain-container-instances のターゲットクラスター	サポートされている各リージョン：5	はい	実験ごとに、タグを使用してターゲットを特定するときに aws:ecs:drain-container-instances can がターゲットにできるクラスターの最大数。
aws:rds:failover-db-cluster のターゲットクラスター	サポートされている各リージョン：5	はい	実験ごとに、タグを使用してターゲットを特定するときに aws:rds:failover-db-cluster can がターゲットにできるクラスターの最大数。
aws:rds:reboot-db-instances のターゲット DB インスタンス	サポートされている各リージョン：5	はい	実験ごとに、タグを使用してターゲットを特定するときに aws:rds:reboot-db-instances can がターゲットにできる DBInstances の最大数。

名前	デフォルト	引き上げ可能	説明
aws:ec2:reboot-instances のターゲットインスタンス	サポートされている各リージョン：5	はい	実験ごとに、タグを使用してターゲットを特定するときに、aws:ec2:reboot-instances がターゲットにできるインスタンスの最大数。
aws:ec2:stop-instances のターゲットインスタンス	サポートされている各リージョン：5	はい	実験ごとに、タグを使用してターゲットを特定するときに、aws:ec2:stop-instances がターゲットにできるインスタンスの最大数。
aws:ec2:terminate-instances のターゲットインスタンス	サポートされている各リージョン：5	はい	実験ごとに、タグを使用してターゲットを特定するときに、aws:ec2:terminate-instances がターゲットにできるインスタンスの最大数。
aws:ssm:send-command のターゲットインスタンス	サポートされている各リージョン：5	はい	実験ごとに、タグを使用してターゲットを特定するときに、aws:ssm:send-command がターゲットにできるインスタンスの最大数。

名前	デフォルト	引き上げ可能	説明
aws:eks:terminate-nodegroup-instances のターゲット Nodegroups	サポートされている各リージョン：5	はい	実験ごとに、タグを使用してターゲットを特定するときに aws:eks:terminate-nodegroup-instances can がターゲットにできる Nodegroup の最大数。
aws:eks:pod-cpu-stress のターゲットポッド	サポートされている各リージョン: 50	はい	実験ごとに、パラメータを使用してターゲットを特定するときに aws:eks:pod-cpu-stress can がターゲットにできるポッドの最大数。
aws:eks:pod-delete のターゲットポッド	サポートされている各リージョン: 50	はい	実験ごとに、パラメータを使用してターゲットを特定するときに、aws:eks:pod-delete がターゲットにできるポッドの最大数。
aws:eks:pod-io-stress のターゲットポッド	サポートされている各リージョン: 50	はい	実験ごとに、パラメータを使用してターゲットを特定するときに aws:eks:pod-io-stress can がターゲットにできるポッドの最大数。

名前	デフォルト	引き上げ可能	説明
aws:eks:pod-memory-stress のターゲットポッド	サポートされている各リージョン: 50	はい	実験ごとに、パラメータを使用してターゲットを特定するときに aws:eks:pod-memory-stress can がターゲットにできるポッドの最大数。
aws:eks:pod-network-blackhole-port のターゲットポッド	サポートされている各リージョン: 50	はい	実験ごとに、パラメータを使用してターゲットを特定するときに aws:eks:pod-network-blackhole-port can がターゲットにできるポッドの最大数。
aws:eks:pod-network-latency のターゲットポッド	サポートされている各リージョン: 50	はい	実験ごとに、パラメータを使用してターゲットを特定するときに aws:eks:pod-network-latency can がターゲットにできるポッドの最大数。

名前	デフォルト	引き上げ可能	説明
aws:eks:pod-network-packet-loss のターゲットポッド	サポートされている各リージョン: 50	はい	実験ごとに、パラメータを使用してターゲットを特定するときに aws:eks:pod-network-packet-loss can がターゲットにできるポッドの最大数。
aws:elasticache:interrupt-cluster-az-power ReplicationGroups のターゲット	サポートされている各リージョン：5	はい	実験ごとに、タグ/パラメータを使用してターゲットを特定するときに aws:elasticache:interrupt-cluster-az-power can がターゲット ReplicationGroups にできる の最大数。
aws:ec2:send-spot-instance-interruptions SpotInstances のターゲット	サポートされている各リージョン：5	はい	実験ごとに、タグを使用してターゲットを特定するときに aws:ec2:send-spot-instance-interruptions can がターゲット SpotInstances にできる の最大数。

名前	デフォルト	引き上げ可能	説明
aws:network:disrupt-connectivity のターゲットサブネット	サポートされている各リージョン：5	はい	実験ごとに、タグを使用してターゲットを特定するときに、aws:network:disrupt-connectivity がターゲットにできるサブネットの最大数。5 を超えるクォータは、パラメータ scope:all にのみ適用されます。別のスコープタイプでより高いクォータが必要な場合は、https://console.aws.amazon.com/support/home#/. のカスタマーサポートにお問い合わせください。
aws:network:route-table-disrupt-cross-region-connectivity のターゲットのサブネット	サポートされている各リージョン: 6	はい	実験ごとに、タグを使用してターゲットを特定するときに aws:network:route-table-disrupt-cross-region-connectivity can がターゲットにできるサブネットの最大数。
aws:ecs:stop-task のターゲットタスク	サポートされている各リージョン：5	はい	実験ごとに、タグを使用してターゲットを特定するときに、aws:ecs:stop-task がターゲットにできるタスクの最大数。

名前	デフォルト	引き上げ可能	説明
aws:ecs:task-cpu-stress のターゲットタスク	サポートされている各リージョン：5	はい	実験ごとに、タグ/パラメータを使用してターゲットを特定するときに aws:ecs:task-cpu-stress can がターゲットにできるタスクの最大数。
aws:ecs:task-io-stress のターゲットタスク	サポートされている各リージョン：5	はい	実験ごとに、タグ/パラメータを使用してターゲットを特定するときに aws:ecs:task-io-stress can がターゲットにできるタスクの最大数。
aws:ecs:task-kill-process のターゲットタスク	サポートされている各リージョン：5	はい	実験ごとに、タグ/パラメータを使用してターゲットを特定するときに aws:ecs:task-kill-process can がターゲットにできるタスクの最大数。
aws:ecs:task-network-blackhole-port のターゲットタスク	サポートされている各リージョン：5	はい	実験ごとに、タグ/パラメータを使用してターゲットを特定するときに aws:ecs:task-network-blackhole-port can がターゲットにできるタスクの最大数。

名前	デフォルト	引き上げ可能	説明
aws:ecs:task-network-latency のターゲットタスク	サポートされている各リージョン：5	はい	実験ごとに、タグ/パラメータを使用してターゲットを特定するときに aws:ecs:task-network-latency can がターゲットにできるタスクの最大数。
aws:ecs:task-network-packet-loss のターゲットタスク	サポートされている各リージョン：5	はい	実験ごとに、タグ/パラメータを使用してターゲットを特定するときに aws:ecs:task-network-packet-loss can がターゲットにできるタスクの最大数。
aws:network:transit-gateway-disrupt-cross-region-connectivity TransitGateways のターゲット	サポートされている各リージョン：5	はい	実験ごとに、タグを使用してターゲットを特定するときに aws:network:transit-gateway-disrupt-cross-region-connectivity can がターゲットにできる Transit Gateway の最大数。
実験テンプレートごとのターゲットアカウント設定	サポートされている各リージョン: 10	はい	このアカウントで現在のリージョンにおいて作成できる実験テンプレートのターゲットアカウント設定の最大数。

名前	デフォルト	引き上げ可能	説明
aws:dynamodb:global-table-pause-replication action のターゲットテーブル	サポートされている各リージョン：5	はい	実験ごとに aws:dynamodb:global-table-pause-replication can がターゲットとするグローバルテーブルの最大数。

AWS FIS の使用には、次の追加制限が適用されます。

名前	制限
`aws:elasticache:interrupt-cluster-az-power` アクションのターゲット	1 日あたり、1 つのリージョンのアカウントあたり 10 個の`aws:elasticache:redis-replicationgroup` クラスターに障害が発生することに制限されています。AWS サポートセンターコンソール でサポートケースを作成することで、引き上げをリクエストできます。

ドキュメント履歴

次の表は、AWS Fault Injection Service ユーザーガイドのドキュメントに関する重要な更新事項についてまとめたものです。

変更	説明	日付
新しいアクション	`aws:dynamodb:global-table-pause-replication` アクションを使用して、ターゲットグローバルテーブルとそのレプリカテーブル間のデータレプリケーションを一時停止できるようになりました。`aws:dynamodb:encrypted-global-table-pause-replication` アクションはサポートされなくなります。	2024 年 4 月 24 日
新しいアクションモードの実験オプション	実験を実行する前にターゲットプレビューを生成する`skip-all`には、アクションモードを に設定できます。	2024 年 3 月 13 日
AWS マネージドポリシーの更新	AWS FIS が既存の管理ポリシーを更新しました。	2024 年 1 月 25 日
新しいシナリオとアクション	AWS FIS シナリオ Cross-Region:Connectivity と AZ Availability: Power Interruption を使用できるようになりました。	2023 年 11 月 30 日
新しいアクション	aws:ec2:asg-insufficient-instance-capacity-error アクショ	2023 年 11 月 30 日

	ンが使用できるようになりました。	
新しいアクション	aws:ec2:api-insufficient-instance-capacity-error アクションが使用できるようになりました。	2023 年 11 月 30 日
新しいアクション	aws:network:route-table-disrupt-cross-region-connectivity アクションが使用できるようになりました。	2023 年 11 月 30 日
新しいアクション	aws:network:transit-gateway-disrupt-cross-region-connectivity アクションが使用できるようになりました。	2023 年 11 月 30 日
新しいアクション	aws:dynamodb:encrypted-global-table-pause-replication アクションが使用できるようになりました。	2023 年 11 月 30 日
新しいアクション	aws:s3:bucket-pause-replication アクションが使用できるようになりました。	2023 年 11 月 30 日
新しいアクション	aws:elasticache:interrupt-cluster-az-power アクションが使用できるようになりました。	2023 年 11 月 30 日
新しい実験オプション	アカウントターゲット設定と空のターゲット解決に AWS FIS 実験オプションを使用できるようになりました。	2023 年 11 月 27 日

AWS FIS の名前変更	サービス名を AWS Fault Injection Service に更新しました。	2023 年 11 月 15 日
AWS マネージドポリシーの更新	AWS FIS が既存の管理ポリシーを更新しました。	2023 年 11 月 13 日
新しいシナリオライブラリ	AWS FIS シナリオライブラリ機能を使用できるようになりました。	2023 年 11 月 7 日
新しい実験スケジューラー	AWS FIS 実験スケジューラ機能を使用できるようになりました。	2023 年 11 月 7 日
AWS マネージドポリシーの更新	AWS FIS が既存の管理ポリシーを更新しました。	2023 年 6 月 2 日
新しいアクション	新しい aws:ecs:task アクションと aws:eks:pod アクションを使用できます。	2023 年 6 月 1 日
AWS マネージドポリシーの更新	AWS FIS が既存の管理ポリシーを更新しました。	2023 年 6 月 1 日
新しい事前設定済みの SSM ドキュメント	事前設定済みの SSM ドキュメント AWSFIS-Run-Disk-Fill を使用できます。	2023 年 4 月 28 日
新しいアクション	aws:ebs:pause-volume-io アクションを使用すると、アタッチされているターゲットのボリュームとインスタンス間の I/O を一時停止できます。	2023 年 1 月 27 日

新しいアクション	aws:network:disrupt-connectivity アクションを使用すると、ターゲットサブネットへの特定の種類のトラフィックを拒否できます。	2022 年 10 月 26 日
新しいアクション	aws:eks:inject-kubernetes-custom-resource アクションを使用して、単一のターゲットクラスターで ChaosMesh または Litmus 実験を実行できます。	2022 年 7 月 7 日
実験ロギング	実験アクティビティログを CloudWatch Logs または S3 バケットに送信するように実験テンプレートを設定できます。	2022 年 2 月 28 日
新しいジョブ通知	実験の状態が変更されると、AWS FIS は通知を発行します。これらの通知は、Amazon を通じてイベントとして利用可能になります EventBridge。	2022 年 2 月 24 日
新しいアクション	aws:ecs:stop-task アクションを使用して、指定したタスクを停止できます。	2022 年 2 月 9 日
新しいアクション	aws:cloudwatch:assert-alarm-stateアクションを使用して、指定したアラームが指定されたアラーム状態のいずれかにあることを確認します。	2021 年 11 月 5 日

新しい事前設定済みの SSM ドキュメント	事前設定済みの SSM ドキュメントを使用できます。 AWSFIS-Run-IO-Stress、 AWSFIS-Run-Network-Blackhold-Port、 AWSFIS-Run-Network-Latency-Sources、 AWSFIS-Run-Network-Packet-Loss、 AWSFIS-Run-Network-Packet-Loss-Sources。	2021 年 11 月 4 日
新しいアクション	スポットインスタンスの中断通知をターゲットのスポットインスタンスに送信し、ターゲットのスポットインスタンスを中断する aws:ec2:send-spot-instance-interruptions アクションを使用できます。	2021 年 10 月 20 日
新しいアクション	オートメーション Runbook の実行を開始する aws:ssm:start-automation-execution アクションを使用できます。	2021 年 9 月 17 日
初回リリース	AWS Fault Injection Service ユーザーガイドの初回リリース。	2021 年 3 月 15 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。