aws

ユーザーガイド

Microsoft Windows の Amazon Kinesis エージェント

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

Microsoft Windows の Amazon Kinesis エージェント: ユーザーガイド

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

Amazon の商標およびトレードドレスは、Amazon のものではない製品またはサービスと関連付けてはならず、また、お客様に混乱を招くような形や Amazon の信用を傷つけたり失わせたりする形で使用することはできません。Amazon が所有しない商標はすべてそれぞれの所有者に所属します。所有者は必ずしも Amazon と提携していたり、関連しているわけではありません。また、Amazon 後援を受けているとはかぎりません。

Table of Contents

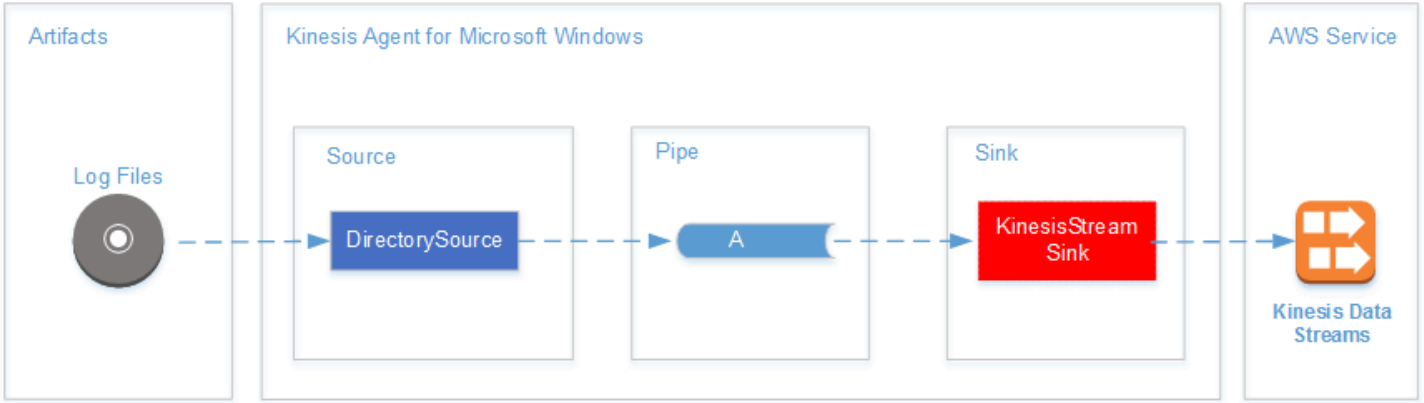
Amazon Kinesis Agent とは何ですか?

Amazon Kinesis Agent for Microsoft Windows (Windows 用 Kinesis Agent) は、設定可能で拡張可能なエージェントです。Windows デスクトップコンピュータおよびサーバーのフリート上で、オンプレミスまたは AWS クラウドで動作します。Kinesis Agent for Windows は、ログ、イベント、およびメトリクスを効率的かつ確実に収集、解析、変換、ストリーム配信します。Kinesis Data Streams,Kinesis Data Firehose,Amazon CloudWatch, および[CloudWatch Logs]。

これらのサービスから、以下を含むさまざまなその他の AWS サービスを使用して、データを保存、分析、および可視化できます。

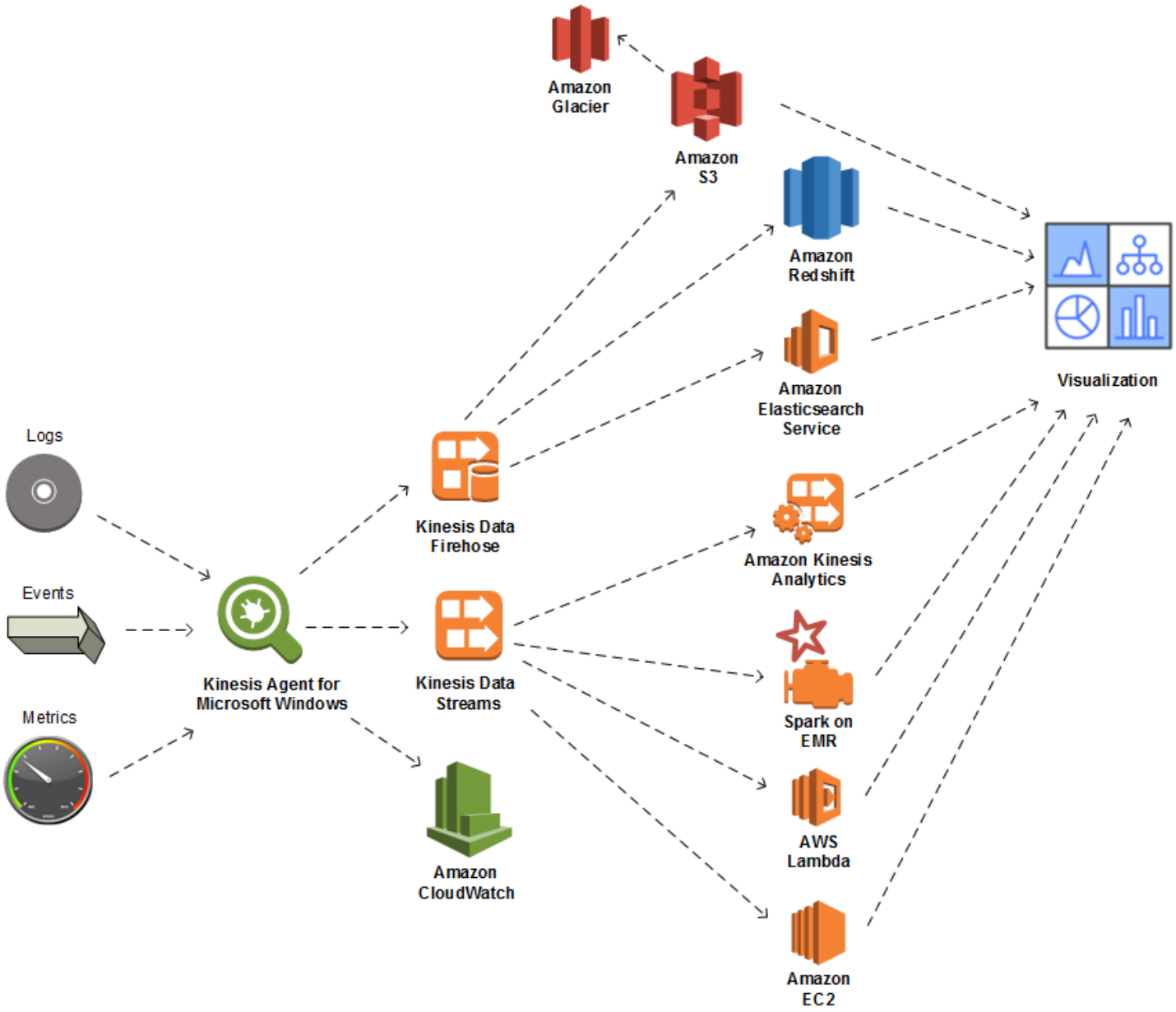
- Amazon Simple Storage Service (Amazon S3)
- Amazon Redshift
- Amazon Elasticsearch Service (Amazon ES)
- Kinesis Data Analytics
- Amazon QuickSight
- Amazon Athena
- Kibana

次の図は、Kinesis Data Streamsにログファイルをストリーミングする Kinesis Agent for Windows のシンプルな構成を示しています。



ソース、パイプ、およびシンクの詳細については、「Microsoft Windows 向けの Amazon Kinesis エージェントの概念」を参照してください。

次の図は、ストリーム処理フレームワークを使用してカスタムのリアルタイムデータパイプラインを構築する方法の一部を示しています。これらのフレームワークには、Kinesis Data Analytics、Amazon EMR 上の Apache Spark、および AWS Lambda が含まれます。



トピック

- AWS について
- Windows 用 Kinesis Agent で何ができるか
- Benefits
- Windows 用 Kinesis エージェントの使用を開始する

AWS について

Amazon Web Services (AWS) は、アプリケーションの開発時に利用できるデジタルインフラストラクチャサービスの集合体です。サービスには、演算能力、ストレージ、データベース、分析、およびアプリケーション同期 (メッセージングとキューイング) があります。AWS は従量制サービスモデルを採用しています。料金が発生するのは、自分またはアプリケーションが使用するサービスの分のみです。また、そのサービスをプロトタイプと実験に利用しやすくするために、AWS では無料利用枠も用意されています。この枠では、サービスを利用しても一定のレベル以下であれば無料です。AWS コストおよび AWS の詳細無料利用枠の使用方法については、リソースセンターのご利用開始にあたって。AWS アカウントを作成するには、AWS のホームページを開き、サインアップします。

Windows 用 Kinesis Agent で何ができるか

Windows 用 Kinesis Agent には、以下の特徴と機能があります。

ログ、イベント、およびメトリクスデータの収集

Kinesis Agent for Windows は、サーバーおよびデスクトップのフリートから 1 つ以上の AWS サービスに、ログ、イベント、およびメトリクスを収集、解析、変換、およびストリーミングします。サービスが受信したペイロードは、元のソースとは異なる形式の場合があります。たとえば、ログがサーバー上の特定のテキスト形式 (syslog 形式) などで保存される場合があります。Windows Kinesis Agent は、そのテキストを収集および解析し、AWS にストリーミングする前などに、任意で JSON 形式に変換できます。これにより、JSON を消費する一部の AWS のサービスによる単純な処理が容易になります。Kinesis Data Streams にストリーミングされるデータは、Kinesis Data Analytics によって継続的に処理され、追加のメトリクスと集約されたメトリクスを生成し、ライブダッシュボードを強化できます。データパイプラインのダウンストリームでデータが使用される方法に応じて、さまざまな AWS サービス (Amazon S3 など) を使用してデータを保存できます。

AWS サービスとの統合

Kinesis Agent for Windows では、ログファイル、イベント、およびメトリクスを複数の異なる AWS サービスに送信するように設定できます。

- Kinesis Data Firehose— ストリーミングされたデータを Amazon S3、Amazon Redshift、Amazon ES、またはSplunkさらなる分析のために。
- Kinesis Data Streams— Kinesis データ分析または Apache Spark でホストされているカスタムアプリケーションを使用して、ストリームデータを処理します。Amazon EMR。または、Amazon EC2インスタンスまたは、で実行中のカスタムのサーバーレス関数AWS Lambda。
- CloudWatch：ストリーミングされたメトリクスをグラフで表示できます。ダッシュボードに結合することもできます。次に、プリセットのしきい値を違反するメトリクス値によってトリガーされる CloudWatch アラームを設定します。
- [CloudWatch Logs]— ストリーミングされたログとイベントを保存し、AWS マネジメントコンソールで表示および検索するか、データパイプラインのさらに下流側で処理できます。

すばやいインストールと設定

Windows 用 Kinesis Agent は、ほんの数ステップでインストールおよび設定できます。詳細については、「Windows 用 Kinesis エージェントのインストール」および「Microsoft Windows 用の Amazon Kinesis エージェントの設定」を参照してください。単純な宣言型の設定ファイルによって以下を指定します。

- 収集対象のログ、イベント、およびメトリクスのソースと形式。
- 収集されたデータに適用する変換。追加データを含めることができ、既存のデータは変換およびフィルタリングできます。
- 最終データがストリーミングされた宛先、およびストリーミングペイロードのバッファリング、シャーディング、および形式。

Windows Kinesis Agent には、以下の一般的な Microsoft エンタープライズサービスによって生成されるログファイル向けに、組み込みパーサーがあります。

- Microsoft Exchange

- SharePoint
- Active Directory ドメインコントローラー
- DHCP サーバー

継続的な管理が不要

Windows 用 Kinesis Agent は、データを一切失うことなく、さまざまな状況に自動的に適応します。その機能には、ログのローテーション、再起動後のリカバリー、および一時的なネットワークまたはサーバーの中断が含まれます。Windows 用 Kinesis Agent は、新しいバージョンに自動的に更新するように設定できます。以上のいずれの状況でも、オペレーターの介入は必要ありません。

オープンアーキテクチャを使用した拡張

サーバーまたはデスクトップシステムの監視に宣言型の機能と組み込みプラグインが不十分な場合、プラグインを作成して Kinesis Agent for Windows を拡張できます。新しいプラグインによって、ログ、イベント、およびメトリクスの新しいソースと宛先が有効になります。Windows 用 Kinesis Agent のソースコードは、https://github.com/awslabs/kinesis-agent-windows。

Benefits

Windows Kinesis Agent は、データパイプラインのログ、イベント、およびメトリクスについて、初期データの収集、変換、およびストリーミングを実行します。このようなデータパイプライン構築すると、以下のような多くのメリットがあります。

分析と可視化

Windows 用 Kinesis Agent と Kinesis Data Firehose の統合およびその変換機能によって、さまざまな分析および可視化サービスとの統合が容易になります。

- Amazon QuickSight— さまざまなソースから取り込むことができるクラウドベースの BI サービス。Windows 用 Kinesis Agent は、データを変換し、KKinesis Data Firehose e 経由で Amazon S3 および Amazon Redshift にストリーミングできます。このプロセスにより、Amazon QuickSight 可視化を使用して、データから深い洞察を得られるようになります。
- Athena— データの SQL ベースのクエリを可能にする対話型クエリサービス。Windows 用 Kinesis エージェントは、Kinesis データFirehose 経由で Amazon S3 にデータを変換およびストリーミングできます。Athena は、そのデータに対してインタラクティブに SQL クエリを実行して、ログとイベントを迅速に確認および解析できます。
- Kibana— オープンソースのデータ可視化ツール。Windows 用 Kinesis エージェントは、Kinesis データ Firehose を介してデータを変換し、Amazon ES にストリーミングできます。次に、Kibana を使用してそのデータを調べることができます。ヒストグラム、折れ線グラフ、円グラフ、ヒートマップ、および地理空間グラフィックスを含むさまざまな可視化を作成して、開くことができます。

Security

Kinesis Agent for Windows を含むログおよびイベントデータ分析パイプラインでは、組織のセキュリティ違反を検出し、警告します。これは、攻撃をブロックし、阻止するのに役立ちます。

アプリケーションのパフォーマンス

Windows Kinesis Agent は、アプリケーションまたはサービスのパフォーマンスに関するログ、イベント、およびメトリクスデータを収集できます。次に、完全なデータパイプラインがこのデータを分析します。この分析によって、明らかではなかった可能性がある欠陥を検出して、報告することで、アプリケーションとサービスのパフォーマンスと信頼性を向上させるのに役立ちます。たとえば、サービス API コールの実行時間の大きな変化を検出できます。この機能は、デプロイと相関がある場合、自分が所有するサービスに関する新しいパフォーマンスの問題を見つけ、解決するのに役立ちます。

サービスオペレーション

データパイプラインでは、運用上の潜在的な問題を予測し、サービス停止を回避する方法についての洞察を得るために収集されたデータを解析できます。たとえば、ログ、イベント、およびメトリクスを解析して、現在および予測される容量の利用率を特定できます。これによって、サービスのユーザーが影響を受ける前に追加容量をオンラインで提供できます。サービス停止が発生した場合、データを解析して、停止期間中のお客様への影響を特定します。

Auditing

データパイプラインでは、Kinesis Agent for Windows が収集および変換するログ、イベント、およびメトリクスを処理できます。次に、さまざまな AWS サービスを使用して、この処理済みデータを監査できます。たとえば、Kinesis Data Firehose は、Amazon S3 にデータを保存する Kinesis Agent for Windows からデータストリームを受信できます。次に、Athena を使用してインタラクティブな SQL クエリを実行して、このデータを監査できます。

Archiving

多くの場合、最も重要な運用データは、最近収集されたデータです。ただし、数年にわたってアプリケーションおよびサービスに関して収集されたデータの解析も、長期的な計画作成などに役立ちます。大量のデータを保持することは、コストがかかることがあります。Windows 用 Kinesis Agent は、Kinesis データFirehose を介して Amazon S3 にデータを収集、変換、保存できます。したがって、Amazon S3 Glacierを使用して、古いデータをアーカイブする費用を削減できます。

Alerting

Windows 用 Kinesis エージェントは CloudWatch にメトリックスをストリーミングします。次に、CloudWatch アラームを作成して、経由で通知を送信できます。Amazon 簡易通知サービス (Amazon SNS)メトリックが一貫して特定のしきい値を超えている場合 これにより、アプリケーションとサービスに関する運用上の問題について気付きやすくなります。

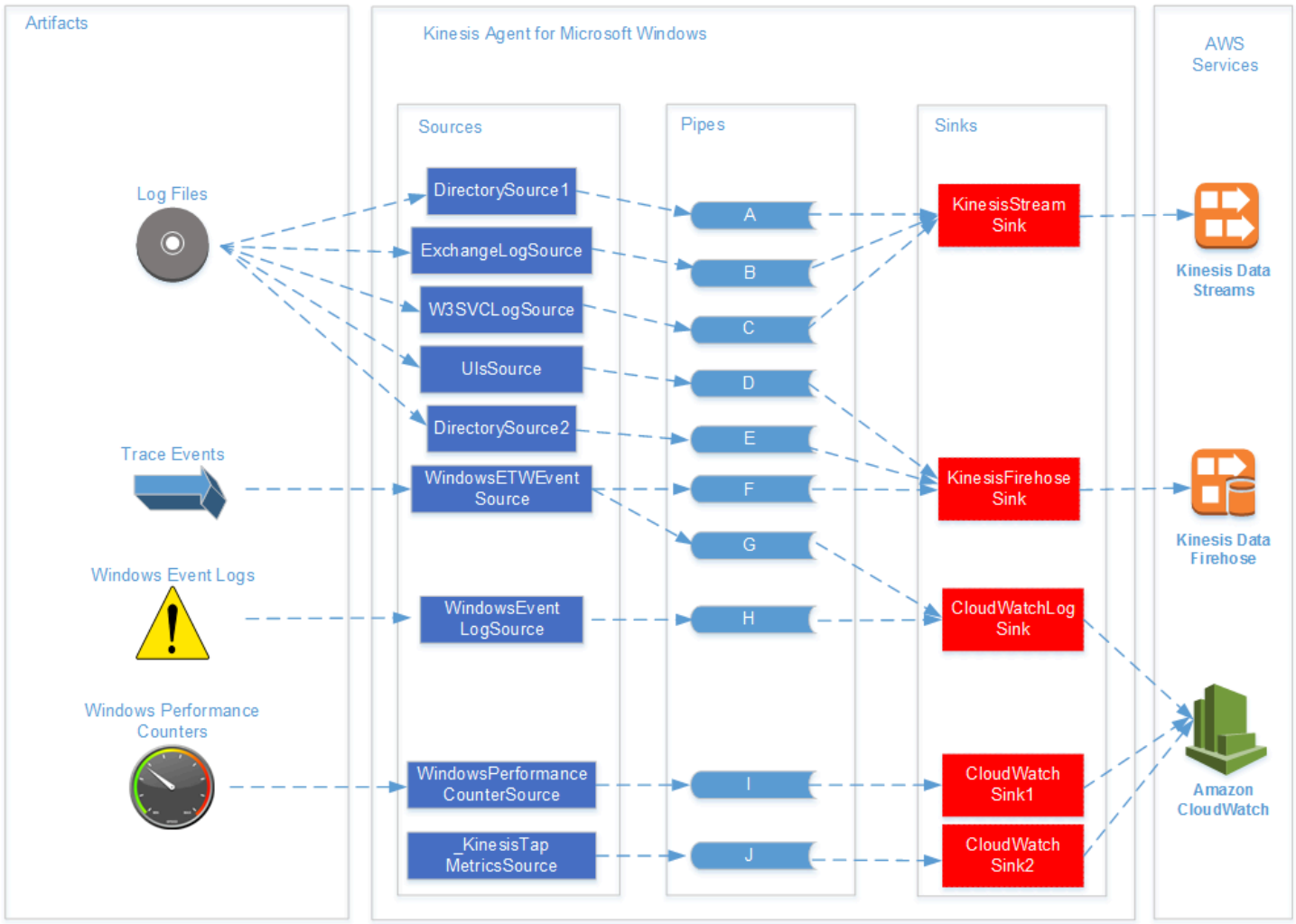
Windows 用 Kinesis エージェントの使用を開始する

Windows 用 Kinesis Agent の詳細については、次のセクションを使用して開始することをお勧めします。

- Microsoft Windows 向けの Amazon Kinesis エージェントの概念
- Microsoft Windows 用 Amazon Kinesis エージェントの使用開始

Microsoft Windows 向けの Amazon Kinesis エージェントの概念

Amazon Kinesis Agent for Microsoft Windows (Kinesis Agent for Windows) の主要な概念を理解することで、デスクトップおよびサーバーのフリートでデータを収集して、処理するためにデータパイプラインの残りの部分にストリーミングすることが容易になります。



データパイプラインの図は、以下のコンポーネントとプロセスを示しています。

1 つ以上の Kinesis Agent for Windows によって収集されるログファイル、イベント、およびメトリクスなどのアーティファクトがサーバーとデスクトップに含まれるsources。データは、オプションで変換できます。たとえば、フラットファイルのテキスト形式からオブジェクトへと変換できます。

データ (オブジェクトまたはテキスト形式) は、1 つ以上の Kinesis Agent for Windows に取り込まれます。パイプ。パイプは、1 つのソースと 1 つの Windows を接続します。sink。パイプでは、不要なデータをオプションで除外できます。

シンクでは、オブジェクトにパースされたデータを JSON または XML にオプションで変換できます。シンクは、Kinesis データストリーム、KinesKinesis Data Firehose e、または Amazon CloudWatch などの特定の ASW サービスにデータを送信します。

複数のパイプを使用することで、1 つのソースが複数のシンクに同じデータを送信できます (例として、図のパイプ F および G を参照)。複数のパイプを使用することで、異なるソースが 1 つのシンクにデータをストリーミングできます (例として、図のパイプ A、B、および C を参照)。複数のパイプを使用することで、複数のシンクから複数のソースにデータをストリーミングすることもできます。ソース、シンク、およびパイプにはタイプがあり、同じタイプのソース、シンク、またはパイプが複数存在できます。

ソース、シンク、およびパイプを宣言する設定ファイルの例については、「Windows 用 Kinesis エージェントの設定例」を参照してください。

トピック

データパイプライン

Aデータパイプラインは、アプリケーションとサービスのアラームの収集、処理、視覚化、および場合によっては生成に使用されます。Kinesis Agent for Windows は、開始時にデータパイプラインに適合します。ここでは、ログ、イベント、およびメトリックがデスクトップコンピューターまたはサーバーから収集されます。Kinesis Agent for Windows は、収集したデータをストリーミングして、データパイプラインの残りを形成するさまざまな ASAWS サービスにストリーミングします。データパイプラインには、特定のサービスの状態をリアルタイムで可視化するなどの目的があり、エンジニアがより効率的にサービスを運用するのに役立ちます。サービス状態データパイプラインは、以下のいずれかを実行します。

- これらの問題がサービスの顧客の体験に影響を与える前にエンジニアに警告します。

- リソースの使用状況の傾向を示すことで、エンジニアがサービスのコストを効率的に管理するのを支援します。これらの傾向を把握することで、エンジニアは適切なリソースレベルを調整したり、さらに自動スケーリングシナリオを実装できます。
- サービスの顧客によってレポートされる問題の根本原因を洞察します。これによって、これらの問題の解決時間が短縮され、サポート費用を削減できます。

Windows 用 Kinesis Agent を使用してデータパイプラインを構築するための詳しい手順については、チュートリアル: Windows 用 Kinesis エージェントを使用して JSON ログファイルを Amazon S3 にストリーミング。

Sources

Windows 向けKinesis エージェントsourceログ、イベント、またはメトリクスを収集します。ソースは、ソースのタイプに基づいて、そのデータの特定のプロデューサーから特定の種類のデータを収集します。たとえば、`DirectorySource` タイプは、ファイルシステムの特定のディレクトリからログファイルを収集します。データがまだ構造化されていない場合 (一部の種類のログファイルについて)、ソースは、一部の構造化された形式にテキスト表現をパースするのに役立つ場合があります。各ソースは、特定のソース宣言Windows 用 Kinesis エージェントappsettings.json設定ファイルを作成します。ソース宣言によって、特定のデータ収集要件に基づいてソースを調整するようにソースを設定するために必要な詳細情報が提供されます。設定可能な詳細の種類は、ソースタイプによって異なります。たとえば、`DirectorySource` ソースタイプの場合、ログファイルがあるディレクトリを指定する必要があります。

ソースタイプおよびソース宣言の詳細については、「ソース宣言」を参照してください。

Sinks

Windows 向けKinesis エージェントsinkは、Kinesis Agent for Windows ソースによってデータを収集させ、データパイプラインの残りを形成する複数の可能な ASAWS サービスのいずれかにそのデータをストリーミングします。各シンクは、特定のシンク宣言Windows 用 Kinesis エージェントappsettings.json設定ファイルを作成します。シンク宣言によって、特定のデータストリーミング要件に基づいてシンクを調整するようにシンクを設定するために必要な詳細情報が提供されます。設定可能な詳細の種類は、シンクタイプによって異なります。たとえば、あるシンクタイプでは、シンク宣言によって、そのタイプに提供されたデータの特定のシリアル化 `Format` を指定できます。シンク宣言でこのオプションが指定されている場合、シンクに関連付けられている AWS サービスにデータをストリーミングする前に、収集されたデータのシリアル化が実行されます。

シンクタイプとシンク宣言の詳細については、「シンク宣言」を参照してください。

Pipes

Windows 向けKinesis エージェントパイプWindows 用 Kinesis エージェントのソースの出力と Windows 用シンクの入力を接続します。オプションで、データがパイプを経由して移動する際にデータを変換できます。各パイプは、Windows 用の Kinesis エージェントの特定のパイプ宣言に対応します。`appsettings.json`設定ファイルを作成します。パイプ宣言では、パイプのソースおよびシンクなど、シンクを設定する際に必要な詳細情報が提供されます。

パイプタイプとパイプ宣言の詳細については、「パイプ宣言」を参照してください。

Microsoft Windows 用 Amazon Kinesis エージェントの使用開始

Amazon Kinesis Agent for Microsoft Windows (Kinesis Agent for Windows) を使用すると、Windows フリートから AWS サービスにログ、イベント、およびメトリクスを収集、解析、変換、およびストリーミング配信できます。以下の情報には、Kinesis Agent for Windows をインストールおよび設定するための前提条件と詳細な手順が含まれています。

トピック

Prerequisites

Windows 用 Kinesis Agent をインストールする前に、次の前提条件が揃っていることを確認してください。

- Windows 用 Kinesis エージェントの概念に精通しています。詳細については、「Microsoft Windows 向けの Amazon Kinesis エージェントの概念」を参照してください。
- データパイプラインに関するさまざまな AWS サービスを使用するための AWS アカウント。AWS アカウントの作成と設定の詳細については、「AWS アカウントのセットアップ」を参照してください。
- Windows 用 Kinesis エージェントを実行する各デスクトップまたはサーバーに Microsoft .NET Framework 4.6 以降がインストールされていること。詳細については、Microsoft .NETドキュメントの「Install the .NET Framework for developers」を参照してください。

  デスクトップまたはサーバーにインストールされている .NET Framework が最新バージョンかどうかを特定するには、次の PowerShell スクリプトを使用します。

```
[System.Version](
(Get-ChildItem 'HKLM:\SOFTWARE\Microsoft\NET Framework Setup\NDP' -recurse `
| Get-ItemProperty -Name Version -ErrorAction SilentlyContinue `
```

```
    | Where-Object { ($_.PSChildName -match 'Full') } `
    | Select-Object Version | Sort-Object -Property Version -Descending)[0]).Version
```

- Windows 用 Kinesis Agent からデータを送信するストリーム (Amazon Kinesis Data Streams を使用している場合) 。ストリームを作成するには、Kinesis Data Streams コンソールとすると、AWS CLI, またはAWS Tools for Windows PowerShell。詳細については、「」を参照してください。データストリームの作成および更新()Amazon Kinesis Data Streams 開発者ガイド。
- Windows 用 Kinesis エージェント (Amazon Kinesis データ Firehose を使用している場合) からデータを送信する Firehose 配信ストリーム。配信ストリームを作成するには、Kinesis Data Firehose コンソールとすると、AWS CLI, またはAWS Tools for Windows PowerShell。詳細については、「」を参照してください。Amazon Kinesis Data Firehose 配信ストリームの作成()Amazon Kinesis Data Firehose 開発者ガイド。

AWS アカウントのセットアップ

AWS アカウントをお持ちでない場合は、以下の手順に従ってアカウントを作成してください。

サインアップして AWS アカウントを作成するには

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

   サインアップ手順の一環として、通話呼び出しを受け取り、電話のキーパッドを用いて確認コードを入力することが求められます。

自分用の管理者ユーザーを作成し、そのユーザーを管理者グループに追加するには (コンソール)

1. [] にサインインします。IAM コンソールを選択して、アカウントの所有者としてルートユーザーをクリックし、AWS アカウントの E メールアドレスを入力します。次のページでパスワードを入力します。

   > **Note**
   >
   > 使用に関するベストプラクティスに従うことを強くお勧めします**Administrator**ルートユーザーの認証情報を追跡し、安全な場所に保管しておくIAM ユーザー。ルートユー

ザーとしてのサインインは、いくつかのアカウントとサービスの管理タスクの実行にのみ使用してください。

2. ナビゲーションペインで [Users]、[Add user] の順に選択します。
3. [ユーザー名] に「**Administrator**」と入力します。
4. [AWS Management Console access (AWS マネジメントコンソールへのアクセス)] の横にあるチェックボックスをオンにします。[Custom password (カスタムパスワード)] を選択し、その後テキストボックスに新しいパスワードを入力します。
5. (オプション) デフォルトでは、AWS は新しいユーザーの初回のサインイン時に新しいパスワードの作成を要求します。必要に応じて [User must create a new password at next sign-in (ユーザーは次回のサインイン時に新しいパスワードを作成する必要がある)] のチェックボックスをオフにして、新しいユーザーがサインインしてからパスワードをリセットできるようにできます。
6. 選択次へ: アクセス許可。
7. [Set permissions (アクセス許可の設定)] で、[Add user to group (ユーザーをグループに追加)] を選択します。
8. [Create group] を選択します。
9. [グループの作成] ダイアログボックスで、[グループ名] に「**Administrators**」と入力します。
10. 選択フィルタポリシー[]、[] の順に選択します。AWS マネージド-ジョブ機能をクリックして、テーブルの内容をフィルタリングします。
11. ポリシーリストで、[AdministratorAccess] のチェックボックスをオンにします。次に、[Create group] を選択します。

> **Note**
>
> `AdministratorAccess` アクセス許可を使用して AWS の請求およびコスト管理コンソールにアクセスするには、IAM ユーザーと IAM ロールの請求情報へのアクセスを有効にする必要があります。これを行うには、請求コンソールへのアクセスの委任に関するチュートリアルのステップ 1 の手順に従ってください。

12. グループのリストに戻り、新しいグループのチェックボックスをオンにします。必要に応じて [Refresh] を選択し、リスト内のグループを表示します。
13. 選択次へ: タグ。

14. (オプション) タグをキー - 値のペアとしてアタッチして、メタデータをユーザーに追加します。IAM でのタグの使用の詳細については、「」を参照してください。IAM エンティティのタグ付け()IAM ユーザーガイド。
15. 選択次へ: 確認新しいユーザーに追加するグループメンバーシップのリストを表示します。続行する準備ができたら、[Create user] を選択します。

このプロセスを繰り返して新しいグループとユーザーを作成して、AWS アカウントのリソースへのアクセス許可をユーザーに付与できます。ポリシーを使用して特定の AWS のリソースへのユーザーのアクセス許可を制限する方法については、「AWS リソースのアクセス管理」と「IAM アイデンティティベースのポリシーの例」を参照してください。

AWS にサインアップして、管理者アカウントを作成するには

1. AWS アカウントをお持ちでない場合は、https://aws.amazon.com/。[Create an AWS Account] を選択し、オンラインの手順に従います。

   サインアップ手順の一環として、通話呼び出しを受け取り、電話のキーパッドを用いて PIN を入力することが求められます。
2. AWS マネジメントコンソールにサインインして、IAM コンソール (https://console.aws.amazon.com/iam/) を開きます。
3. ナビゲーションペインで、[グループ]、[新しいグループの作成] の順に選択します。
4. [Group Name] にグループの名前 (例: **Administrators**) を入力し、[Next Step] を選択します。
5. ポリシーのリストで、AdministratorAccess ポリシーの横にあるチェックボックスを選択します。[フィルタ] メニューと [検索] ボックスを使用して、ポリシーのリストをフィルタリングできます。
6. [Next Step] を選択します。[Create Group] を選択します。新しいグループが [Group Name] の下に表示されます。
7. ナビゲーションペインで [ユーザー] を選択し、続いて [Create New Users (新しいユーザーの作成)] を選択します。
8. [1] ボックスにユーザー名を入力して、[Generate an access key for each user] の横にあるチェックボックスをオフにし、[Create] を選択します。
9. ユーザーのリストで、先ほど作成したユーザーの名前 (チェックボックスではない) を選択します。[Search] ボックスを使用してユーザー名を検索できます。
10. [Groups] タブを選択し、[Add User to Groups] を選択します。
11. 管理者グループの横にあるチェックボックスをオンにして、[Add to Groups] を選択します。

12. [Security Credentials] タブを選択します。[Sign-In Credentials] で、[Manage Password] を選択します。
13. [Assign a custom password] を選択し、[Password] ボックスと [Confirm Password] ボックスにパスワードを入力して、[Apply] を選択します。

Windows 用 Kinesis エージェントのインストール

Windows に Kinesis エージェントをインストールする方法は 3 つあります。

- MSI (Windows インストーラパッケージ) を使用してインストールします。
- からのインストールAWS Systems Managerサーバーおよびデスクトップを管理するための一連のサーバーです。
- PowerShell スクリプトを実行します。

> **Note**
>
> 次の手順では、KinesisTap および AWSKinesisTap という用語が使用される場合があります。これらの単語は Kinesis Agent for Windows と同じ意味を表しますが、これらの単語を使用して手順を実行するとき、それを指定する必要があります。

MSI を使用して Windows 用 Kinesis エージェントをインストールする

Windows MSI 用 Kinesis エージェントの最新の Windows MSI パッケージは、GitHub 上のキネシス-エージェント-ウィンドウのリポジトリ。MSIをダウンロードしたら、Windowsを使用してMSIを起動し、インストーラの指示に従います。インストール後、Windowsアプリケーションと同じようにアンインストールできます。

または、[] を使用することもできます。msiexecコマンドを実行して、次の例に示すように、Windows のコマンドプロンプトから Windows のコマンドプロンプトからインストールし、ログ記録を有効にし、アンインストールします。置換*AWSKinesisTap.1.1.216.4.msi* `with the appropriate version of Kinesis Agent for Windows for your application.`

Windows 用 Kinesis エージェントをサイレントインストールするには

```
msiexec /i AWSKinesisTap.1.1.216.4.msi /q
```

トラブルシューティング用のインストールメッセージを*logfile.log*:

```
msiexec /i AWSKinesisTap.1.1.216.4.msi /q /L*V logfile.log
```

コマンドプロンプトを使用して Kinesis エージェント for Windows をアンインストールするには

```
msiexec.exe /x {ADAB3982-68AA-4B45-AE09-7B9C03F3EBD3} /q
```

AWS Systems Manager を使用して Windows 用 Kinesis エージェントをインストールする

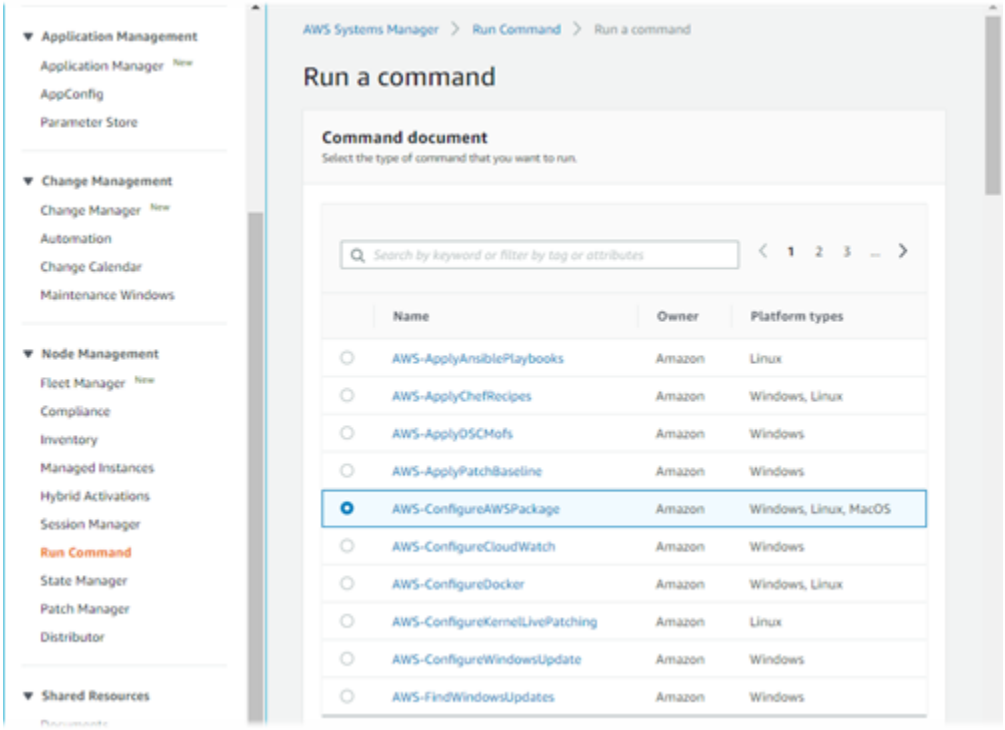
Systems Manager の実行コマンドを使用して Windows 用 Kinesis エージェントをインストールするには、以下の手順を実行します。Run Command の詳細については、」AWS Systems Manager の実行コマンド()AWS Systems Manager ユーザーガイド。Systems Manager Run コマンドを使用することに加えて、Systems Manager を使用することもできます。メンテナンスウィンドウおよびステートマネージャーを使用して、Windows 用 Kinesis エージェントの展開を時間の経過とともに自動化します。

> **Note**
>
> Windows 用 Kinesis Agent for Windows 用のSystems Manager のインストールは、以下の AWS リージョンで使用できます。AWS Systems Manager以下は例外です。
>
> - cn-north-1
> - cn-northwest-1
> - すべての AWS GovCloud リージョン

Systems Manager を使用して Windows 用 Kinesis エージェントをインストールするには

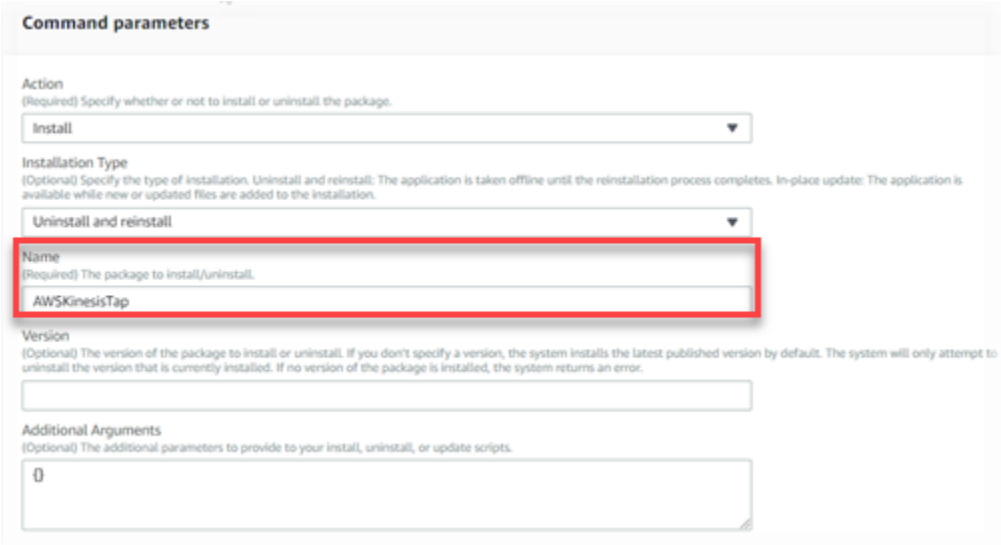
1. Kinesis Agent for Windows をインストールするインスタンスにバージョン 2.2.58.0 以降の SSM エージェントがインストールされていることを確認します。詳細については、「」を参照してください。Windows インスタンスへの SSM エージェントのインストールおよび設定()AWS Systems Manager ユーザーガイド。
2. AWS Systems Manager コンソール () を開きます。https://console.aws.amazon.com/systems-manager/。

3. ナビゲーションペインで、[] の下のノード管理] で、Run Command[]、[] の順に選択します。Run Command。
4. からコマンドのドキュメントリストで、AWS-ConfigureAWSPackagedocument.



5. []コマンドのパラメーター, 用名前にと入力します。AWSkineSistap。その他の設定はデフォルトのままにします。

> **Note**
>
> 離れるVersionAWSkinesistap パッケージの最新バージョンを指定する場合は、空白のままにします。オプションで、インストールする特定のバージョンを入力できます。



6. []ターゲットに、コマンドを実行するインスタンスを指定します。インスタンスに関連付けられたタグに基づいてインスタンスを指定するか、手動でインスタンスを選択するか、インスタンスを含むリソースグループを指定できます。
7. その他すべての設定はデフォルトのままにして、[] メニューからRun。

PowerShell を使用して Windows 用 Kinesis エージェントをインストールする

テキストエディタを使用して、次のコマンドをファイルにコピーし、PowerShell スクリプトとして保存します。私たちはInstallKinesisAgent.ps1以下に例を示します。

```
Param(
    [ValidateSet("prod", "beta", "test")]
    [string] $environment = 'prod',
    [string] $version,
    [string] $baseurl
)

# Self-elevate the script if required.
if (-Not ([Security.Principal.WindowsPrincipal]
 [Security.Principal.WindowsIdentity]::GetCurrent()).IsInRole([Security.Principal.WindowsBuiltI
 'Administrator')) {
    if ([int](Get-CimInstance -Class Win32_OperatingSystem | Select-Object -
ExpandProperty BuildNumber) -ge 6000) {
```

```
        $CommandLine = '-File "' + $MyInvocation.MyCommand.Path + '" ' +
 $MyInvocation.UnboundArguments
        Start-Process -FilePath PowerShell.exe -Verb Runas -ArgumentList $CommandLine
        Exit
    }
}

# Allows input to change base url. Useful for testing.
if ($baseurl) {
    if (!$baseUrl.EndsWith("/")) {
        throw "Invalid baseurl param value. Must end with a trailing forward slash
 ('/')"
    }

    $kinesistapBaseUrl = $baseurl
} else {
    $kinesistapBaseUrl = "https://s3-us-west-2.amazonaws.com/kinesis-agent-windows/
downloads/"
}

Write-Host "Using $kinesistapBaseUrl as base url"

$webClient = New-Object System.Net.WebClient

try {
    $packageJson = $webClient.DownloadString($kinesistapBaseUrl + 'packages.json' + '?
_t=' + [System.DateTime]::Now.Ticks) | ConvertFrom-Json
} catch {
    throw "Downloading package list failed."
}



if ($version) {
    $kinesistapPackage = $packageJson.packages | Where-Object { $_.packageName -eq
 "AWSKinesisTap.$version.nupkg" }

    if ($null -eq $kinesistapPackage) {
        throw "No package found matching input version $version"
    }
} else {
    $packageJson = $packageJson.packages | Where-Object { $_.packageName -match
 ".nupkg" }
    $kinesistapPackage = $packageJson[0]
}
```

```
$packageName = $kinesistapPackage.packageName
$checksum = $kinesistapPackage.checksum

#Create %TEMP%/kinesistap if not exists
$kinesistapTempDir = Join-Path $env:TEMP 'kinesistap'
if (![System.IO.Directory]::Exists($kinesistapTempDir)) {[void]
[System.IO.Directory]::CreateDirectory($kinesistapTempDir)}

#Download KinesisTap.x.x.x.x.nupkg package
$kinesistapNupkgPath = Join-Path $kinesistapTempDir $packageName
$webClient.DownloadFile($kinesistapBaseUrl + $packageName, $kinesistapNupkgPath)
$kinesistapUnzipPath = $kinesistapNupkgPath.Replace('.nupkg', '')

# Calculates hash of downloaded file. Downlevel compatible using .Net hashing on PS < 4
if ($PSVersionTable.PSVersion.Major -ge 4) {
    $calculatedHash = Get-FileHash $kinesistapNupkgPath -Algorithm SHA256
    $hashAsString = $calculatedHash.Hash.ToLower()
} else {
    $sha256 = New-Object System.Security.Cryptography.SHA256CryptoServiceProvider
    $calculatedHash =
 [System.BitConverter]::ToString($sha256.ComputeHash([System.IO.File]::ReadAllBytes($kinesistap
    $hashAsString = $calculatedHash.Replace("-", "").ToLower()
}

if ($checksum -eq $hashAsString) {
 Write-Host 'Local file hash matches checksum.' -ForegroundColor Green
} else {
 throw ("Get-FileHash does not match! Package may be corrupted.")
}

#Delete Unzip path if not empty
if ([System.IO.Directory]::Exists($kinesistapUnzipPath)) {Remove-Item –Path
 $kinesistapUnzipPath -Recurse -Force}

#Unzip KinesisTap.x.x.x.x.nupkg package
$null =
 [System.Reflection.Assembly]::LoadWithPartialName('System.IO.Compression.FileSystem')
[System.IO.Compression.ZipFile]::ExtractToDirectory($kinesistapNupkgPath,
 $kinesistapUnzipPath)

#Execute chocolaeyInstall.ps1 in the package and wait for completion.
$installScript = Join-Path $kinesistapUnzipPath '\tools\chocolateyInstall.ps1'
& $installScript
```

```
# Verify service installed.
$serviceName = 'AWSKinesisTap'
$service = Get-Service -Name $serviceName -ErrorAction Ignore
if ($null -eq $service) {
    throw ("Service not installed correctly.")
} else {
    Write-Host "Kinesis Tap Installed." -ForegroundColor Green
    Write-Host "After configuring run the following to start the service: Start-Service
 -Name $serviceName." -ForegroundColor Green
}
```

昇格されたコマンドプロンプトウィンドウを開きます。ファイルがダウンロードされたディレクトリで、次のコマンドを使用してスクリプトを実行します。

```
PowerShell.exe -File ".\InstallKinesisAgent.ps1"
```

Windows 用 Kinesis エージェントの特定のバージョンをインストールするには、`-version`オプション:

```
PowerShell.exe -File ".\InstallKinesisAgent.ps1" -version "version"
```

置換`#####`を Windows 用の有効な Kinesis エージェントのバージョン番号で設定します。バージョンについては、「」を参照してください。GitHub 上のキネシス-エージェント-ウィンドウのリポジトリ。

PowerShell スクリプトをリモートで実行できる多くのデプロイツールがあります。これらのツールを使用すると、サーバーまたはデスクトップのフリートへの Kinesis Agent for Windows のインストールを自動化できます。

Windows 用 Kinesis エージェントの設定と起動

Windows 用 Kinesis Agent をインストールしたら、エージェントを設定および開始する必要があります。その後は、オペレーションの介入は不要です。

Windows 用 Kinesis エージェントを設定して起動するには

1. Windows 用 Kinesis エージェントの設定ファイルを作成してデプロイします。このファイルによって、他のグローバル設定項目とともに、ソース、シンク、およびパイプが設定されます。

Windows 用 Kinesis エージェントの設定の詳細については、」Microsoft Windows 用の Amazon Kinesis エージェントの設定。

カスタマイズおよびインストール可能な詳細な設定ファイルの例については、「Windows 用 Kinesis エージェントの設定例」を参照してください。

2. 昇格された PowerShell コマンドプロンプトウィンドウを開き、次の PowerShell コマンドを使用して Windows 用 Kinesis エージェントを起動します。

```
Start-Service -Name AWSKinesisTap
```

Microsoft Windows 用の Amazon Kinesis エージェントの設定

Microsoft Windows 用 Amazon Kinesis エージェントを開始する前に、設定ファイルを作成してそれをデプロイする必要があります。設定ファイルは、Windows サーバーおよびデスクトップコンピュータのデータを収集、変換、およびさまざまな AWS のサービスにストリーミングするために必要な情報を提供します。設定ファイルは、オプションの変換と共に、ソース、シンク、およびソースをシンクに接続するパイプのセットを定義します。

Windows 用 Kinesis エージェントの構成ファイルの名前は`appsettings.json`。このファイルを`%PROGRAMFILES%\Amazon\AWSKinesisTap` にデプロイします。

トピック

基本的な設定構造

Microsoft Windows 用 Amazon Kinesis エージェントの設定ファイルの基本構造は、次のテンプレートを持つ JSON ドキュメントです。

```
{
     "Sources": [ ],
     "Sinks": [ ],
     "Pipes": [ ]
}
```

- `Sources` の値は、1 つまたは複数の ソース宣言 です。
- `Sinks` の値は、1 つまたは複数の シンク宣言 です。

- `Pipes` の値は、1 つまたは複数の パイプ宣言 です。

ソース、パイプ、およびシンクの概念の詳細については、「」を参照してください。Microsoft Windows 向けの Amazon Kinesis エージェントの概念。

次の例では、完全な`appsettings.json`設定ファイルを作成します。これは、Windows アプリケーションログイベントを Kinesis データFirehose にストリーミングするように Windows を設定するように Kinesis エージェントを設定します。

```
{
  "Sources": [
    {
      "LogName": "Application",
      "Id": "ApplicationLog",
      "SourceType": "WindowsEventLogSource"
    }
  ],
  "Sinks": [
    {
      "StreamName": "ApplicationLogFirehoseStream",
      "Region": "us-west-2",
      "Id": "MyKinesisFirehoseSink",
      "SinkType": "KinesisFirehose"
    }
  ],
  "Pipes": [
    {
      "Id": "ApplicationLogTotestKinesisFirehoseSink",
      "SourceRef": "ApplicationLog",
      "SinkRef": "MyKinesisFirehoseSink"
    }
  ]
}
```

それぞれの種類の宣言については、以下のセクションを参照してください。

- ソース宣言
- シンク宣言
- パイプ宣言

大文字と小文字の区別の設定

JSON 形式のファイルは通常大文字と小文字が区別されるため、Windows 設定ファイル内のすべてのキーと値も大文字と小文字が区別されると想定する必要があります。appsettings.json 設定ファイル内の一部のキーと値は大文字と小文字が区別されません。たとえば:

- シンクの Format キーと値のペアの値。詳細については、「シンク宣言」を参照してください。
- ソースの SourceType キーと値のペア、シンクの SinkType キーと値のペア、および Type のキーと値のペアの値 パイプとプラグイン。
- DirectorySource ソースの RecordParser キーと値のペアの値。詳細については、「DirectorySource 設定」を参照してください。
- ソースの InitialPosition キーと値のペアの値。詳細については、「ブックマーク設定」を参照してください。
- 変数置換のプレフィックス。詳細については、「シンク変数の置換の設定」を参照してください。

ソース宣言

Microsoft Windows 向け Amazon Kinesis エージェントでは、ソース宣言ログデータ、イベント、メトリクスのデータを収集する場所と対象を記述します。オプションで、そうしたデータを変換できるようにデータ解析のための情報も指定します。以下のセクションでは、Windows 用 Kinesis エージェントで使用可能な組み込みソースタイプの設定について説明します。Windows 用 Kinesis Agent は拡張可能であるため、カスタムのソースタイプを追加できます。通常、各ソースタイプの設定オブジェクトには、そのソースタイプに適切な特定のキーと値のペアが必要です。

すべてのソース宣言には、少なくとも次のキーと値のペアを含める必要があります。

`Id`

設定ファイル内で特定のソースオブジェクトを識別する一意の文字列。

`SourceType`

このソースオブジェクトのソースタイプの名前。ソースタイプは、このソースオブジェクトによって収集されるログ、イベント、またはメトリクスのデータのオリジンを指定します。また、宣言可能なソースの他の側面を制御します。

さまざまな種類のソース宣言を使用する完全な設定ファイルの例については、さまざまなソースから Kinesis Data Streams へのストリーミング を参照してください。

トピック

DirectorySource 設定

Overview

DirectorySource ソースタイプは、指定したディレクトリに保存されているファイルからログを収集します。ログファイルはさまざまな形式であるため、DirectorySource 宣言を使用してログファイルのデータの形式を指定できます。その後、ログの内容を JSON または XML などの標準形式に変換してからさまざまな AWS のサービスにストリーミングすることができます。

次に、DirectorySource 宣言の例を示します。

```
{
    "Id": "myLog",
    "SourceType": "DirectorySource",
    "Directory": "C:\\Program Data\\MyCompany\\MyService\\logs",
    "FileNameFilter": "*.log",
    "IncludeSubdirectories": true,
    "IncludeDirectoryFilter": "cpu\\cpu-1;cpu\\cpu-2;load;memory",
    "RecordParser": "Timestamp",
    "TimestampFormat": "yyyy-MM-dd HH:mm:ss.ffff",
    "Pattern": "\\d{4}-\\d{2}-\\d(2}",
    "ExtractionPattern": "",
    "TimeZoneKind": "UTC",
    "SkipLines": 0,
```

```
    "Encoding": "utf-16",
    "ExtractionRegexOptions": "Multiline"
}
```

すべての DirectorySource 宣言で、次のキーと値のペアを指定することができます。

SourceType

リテラル文字列 "DirectorySource" である必要があります (必須)。

Directory

ログファイルを含むディレクトリのパス (必須)。

FileNameFilter

オプションで、ログデータが収集されるディレクトリにある一連のファイルを、ワイルドカードによるファイル命名パターンに基づいて制限します。複数のログファイル名パターンがある場合、この機能により、単一のDirectorySource次の例に示すように、。

```
FileNameFilter: "*.log|*.txt"
```

システム管理者は、ログファイルをアーカイブする前に圧縮することがあります。指定すると"*.*"がFileNameFilterでは、既知の圧縮ファイルが除外されるようになりました。この機能により、.zip,.gz, および.bz2ファイルが誤ってストリーミングされないようにします。このキーと値のペアが指定されていない場合、デフォルトでは、ディレクトリにあるすべてのファイルからデータが収集されます。

IncludeSubdirectories

オペレーティングシステムによって制限された任意の深さにサブディレクトリを監視するように指定します。この機能は、複数の Web サイトを持つ Web サーバーを監視する場合に便利です。また、 を使用することもできますIncludeDirectoryFilter属性を使用して、フィルタで指定された特定のサブディレクトリのみを監視します。

RecordParser

指定したディレクトリで見つかったログファイルを DirectorySource ソースタイプでどのように解析するかを指定します。このキーと値のペアは必須です。有効な値は次のとおりです。

- SingleLine-ログファイルの各行は 1 つのログレコードです。
- SingleLineJson-ログファイルの各行は JSON 形式の 1 つのログレコードです。このパーサーは、オブジェクトデコレーションを使用して追加のキーと値のペアを JSON に追加す

る場合に役立ちます。詳細については、「シンクデコレーションの設定」を参照してください。SingleLineJson レコードパーサーを使用する例については、チュートリアル: Windows 用 Kinesis エージェントを使用して JSON ログファイルを Amazon S3 にストリーミング を参照してください。

- Timestamp-1 つ以上の行に 1 つのログレコードを含めることができます。ログレコードはタイムスタンプで始まります。このオプションでは、TimestampFormat キーと値のペアを指定する必要があります。
- Regex-各レコードは特定の正規表現に一致するテキストで始まります。このオプションでは、Pattern キーと値のペアを指定する必要があります。
- SysLog— ログファイルがsyslog標準形式。ログファイルはその仕様に基づいてレコードに解析されます。
- Delimited— よりシンプルなバージョンの Regex レコードパーサーで、ログレコードのデータ項目は一貫した区切り記号で区切られます。このオプションは、Regex パーサーよりも使いやすく高速で実行されます。このオプションが使用可能なときは使用することを推奨します。このオプションを使用する場合、Delimiter キーと値のペアを指定する必要があります。

TimestampField

JSON フィールドに、レコードのタイムスタンプが含まれることを指定します。これは、SingleLineJson RecordParser でのみ使用されます。このキーと値のペアはオプションです。指定しない場合、Kinesis Agent for Windows はレコードが読み取られた時刻をタイムスタンプとして使用します。このキーと値のペアを指定する利点の 1 つは、Kinesis Agent for Windows によって生成されたレイテンシー統計の精度が向上することです。

TimestampFormat

レコードに関連付けられた日付と時刻を解析する方法を指定します。この値は、文字列 epoch または .NET 日時形式文字列です。値が epoch の場合、時間が UNIX エポック時間に基づいて解析されます。UNIX エポック時間の詳細については、「UNIX 時間」を参照してください。.NET 日時形式文字列の詳細については、Microsoft .NET ドキュメントの「カスタム日時書式指定文字列」を参照してください。このキーと値のペアは、Timestamp レコードパーサーを指定する場合、または SingleLineJson レコードパーサーを TimestampField キーと値のペアと共に指定する場合にのみ必要です。

Pattern

複数行の可能性があるレコードの最初の行に一致する必要がある正規表現を指定します。このキーと値のペアは、Regex レコードパーサーにのみ必要です。

`ExtractionPattern`

名前付きグループを使用する正規表現を指定します。レコードはこの正規表現を使用して解析され、名前付きグループは解析されたレコードのフィールドを形成します。これらのフィールドは JSON や XML オブジェクトまたはドキュメントを構築する基礎として使用されます。その後、これらのオブジェクトまたはドキュメントはシンクによってさまざまな AWS のサービスにストリーミングされます。このキーと値のペアはオプションです。このペアは`Regex`レコードパーサーとタイムスタンプパーサーです。

`Timestamp` グループ名は、どのフィールドに各ログファイルの各レコードの日時が含まれるかを `Regex` パーサーに示すため、特別に処理されます。

`Delimiter`

各ログレコードの各項目を区切る文字または文字列を指定します。このキーと値のペアは `Delimited` レコードパーサーと共に使用する必要があります (およびこのパーサーでのみ使用できます)。2 文字の表記 `\t` を使用してタブ文字を表します。

`HeaderPattern`

レコードのヘッダーのセットが含まれるログファイルの行に一致させるための正規表現を指定します。ログファイルにヘッダー情報が含まれていない場合は、`Headers` キーと値のペアを使用して暗黙的なヘッダーを指定します。`HeaderPattern` キーと値のペアはオプションであり、`Delimited` レコードパーサーでのみ有効です。

> **Note**
>
> 列に対して空 (0 長) のヘッダーエントリがあると、その列のデータは `DirectorySource` によって解析された出力の最終出力からフィルタリングされます。

`Headers`

指定した区切り記号を使用して解析されたデータの列の名前を指定します。このキーと値のペアはオプションであり、`Delimited` レコードパーサーでのみ有効です。

> **Note**
>
> 列に対して空 (0 長) のヘッダーエントリがあると、その列のデータは `DirectorySource` によって解析された出力の最終出力からフィルタリングされます。

RecordPattern

レコードデータが含まれるログファイルの行を識別する正規表現を指定します。HeaderPattern によって識別されたオプションのヘッダー行を除き、指定した RecordPattern に一致しない行はレコード処理中に無視されます。このキーと値のペアはオプションであり、Delimited レコードパーサーでのみ有効です。これを指定しない場合は、デフォルトで、オプションの CommentPattern にもオプションの HeaderPattern にも一致しないすべての行は、解析可能なレコードデータを含む行であると見なされます。

CommentPattern

ログファイルのデータを解析する前に除外する必要のある、ログファイルの行を識別する正規表現を指定します。このキーと値のペアはオプションであり、Delimited レコードパーサーでのみ有効です。これを指定しない場合は、デフォルトで、オプションの HeaderPattern と一致しないすべての行は、解析可能なレコードデータを含む行であると見なされます。ただし、RecordPattern が指定されている場合を除きます。

TimeZoneKind

ログファイルのタイムスタンプをローカルタイムゾーンであると見なすか UTC タイムゾーンであると見なすかを指定します。これはオプションであり、デフォルトは UTC です。このキーと値のペアの有効な値は Local または UTC のみです。TimeZoneKind が指定されていない場合または値が UTC である場合、タイムスタンプを変更することはできません。タイムスタンプは UTC に変換され、TimeZoneKind値はLocalで、タイムスタンプを受信するシンクが CloudWatch Logs である場合、または解析されたレコードが他のシンクに送信される場合、または処理されたレコードです。メッセージに埋め込まれた日付と時刻は変換されません。

SkipLines

指定した場合、レコードの解析が発生する前に各ログファイルの先頭で無視する行数を制御します。これはオプションであり、デフォルト値は 0 です。

エンコード

デフォルトでは、Kinesis Agent for Windows は、バイトマークからエンコーディングを自動的に検出できます。ただし、一部の古いユニコード形式では、自動エンコードが正しく動作しないことがあります。次の例では、Microsoft SQL Server ログのストリーミングに必要なエンコーディングを指定します。

```
"Encoding": "utf-16"
```

エンコード名の一覧については、エンコードの一覧Microsoft .NET のドキュメントを参照してください。

抽出正規表現オプション

次を使用できます。ExtractionRegexOptions正規表現を単純化します。このキーと値のペアはオプションです。デフォルト: "None"。

次の例では、"."式を含む任意の文字に一致します\r\n。

```
"ExtractionRegexOptions" = "Multiline"
```

ExtractionRegexOptions で使用できるフィールドの一覧については、正規表現オプションの列挙型Microsoft .NET のドキュメントを参照してください。

Regex レコードパーサー

Regex レコードパーサーを TimestampFormat、Pattern、ExtractionPattern キーと値のペアと共に使用して非構造化テキストログを解析できます。たとえば、次のようなログファイルがあるとします。

```
[FATAL][2017/05/03 21:31:00.534][0x00003ca8][0000059c][][ActivationSubSystem]
[GetActivationForSystemID][0] 'ActivationException.File: EQCASLicensingSubSystem.cpp'
[FATAL][2017/05/03 21:31:00.535][0x00003ca8][0000059c][][ActivationSubSystem]
[GetActivationForSystemID][0] 'ActivationException.Line: 3999'
```

Pattern キーと値のペアに次の正規表現を指定すると、ログファイルを個々のログレコードに分割できます。

```
^\[\w+\]\[(?<TimeStamp>\d{4}/\d{2}/\d{2} \d{2}:\d{2}:\d{2}\.\d{3})\]
```

この正規表現は、次のシーケンスに一致します。

1. 評価される文字列の最初。

2. 角括弧で囲まれた 1 つ以上の単語の文字。
3. 角括弧で囲まれたタイムスタンプ。タイムスタンプは、次のシーケンスに一致します。
   a. 4 桁の年
   b. スラッシュ
   c. 2 桁の月
   d. スラッシュ
   e. 2 桁の日
   f. 空白文字
   g. 2 桁の時間
   h. コロン
   i. 2 桁の分
   j. コロン
   k. 2 桁の秒
   l. ピリオド
   m. 3 桁のミリ秒

TimestampFormat キーと値のペアに次の形式を指定すると、テキストタイムスタンプを日付と時刻に変換できます。

```
yyyy/MM/dd HH:mm:ss.fff
```

ExtractionPattern キーと値のペアによってログレコードのフィールドを抽出するために、次の正規表現を使用できます。

```
^\[(?<Severity>\w+)\]\[(?<TimeStamp>\d{4}/\d{2}/\d{2} \d{2}:\d{2}:\d{2}\.
\d{3})\]\[[^]]*\]\[[^]]*\]\[[^]]*\]\[(?<SubSystem>\w+)\]\[(?<Module>\w+)\]\[[^]]*\] '(?
<Message>.*)'$
```

この正規表現は、順に以下のグループに一致します。

1. Severity— 角括弧で囲まれた 1 つ以上の単語の文字。
2. TimeStamp-前のタイムスタンプの説明を参照してください。
3. 角括弧で囲まれた、3 つの名前のない一連のゼロ個以上の文字はスキップされます。
4. SubSystem— 角括弧で囲まれた 1 つ以上の単語の文字。

5. Module— 角括弧で囲まれた 1 つ以上の単語の文字。

6. 角括弧で囲まれた、1 つの名前のない一連のゼロ個以上の文字はスキップされます。

7. 1 つの名前のないスペースはスキップされます。

8. Message— 一重引用符で囲まれた 0 個以上の文字。

以下のソース宣言は、これらの正規表現と日時形式を組み合わせて、このようなログファイルを解析するための Kinesis Agent for Windows への完全な手順を Kinesis Agent に提供するものです。

```
{
    "Id": "PrintLog",
    "SourceType": "DirectorySource",
    "Directory": "C:\\temp\\PrintLogTest",
    "FileNameFilter": "*.log",
    "RecordParser": "Regex",
    "TimestampFormat": "yyyy/MM/dd HH:mm:ss.fff",
    "Pattern": "^\\[\\w+\\]\\[(?<TimeStamp>\\d{4}/\\d{2}/\\d{2} \\d{2}:\\d{2}:\\d{2}\\.
\\d{3})\\]",
    "ExtractionPattern": "^\\[(?<Severity>\\w+)\\]\\[(?<TimeStamp>\\d{4}/\\d{2}/\\d{2}
 \\d{2}:\\d{2}:\\d{2}\\.\\d{3})\\]\\[[^]]*\\]\\[[^]]*\\]\\[[^]]*\\]\\[(?<SubSystem>\\w
+)\\]\\[(?<Module>\\w+)\\]\\[[^]]*\\] '(?<Message>.*)'$",
    "TimeZoneKind": "UTC"
}
```

> **Note**
>
> JSON 形式ファイルでのバックスラッシュは、追加のバックスラッシュを使用してエスケープする必要があります。

正規表現の詳細については、Microsoft .NET ドキュメントの「正規表現言語 - クイックリファレンス」を参照してください。

Delimited レコードパーサー

Delimited レコードパーサーを使用すると、半構造化されたログファイルおよびデータファイルを解析することができます。これらのファイルには、データ各行のデータ列それぞれを区切る一貫した文字シーケンスが存在します。たとえば、CSV ファイルではデータの各列を区切るためにカンマを使用し、TSV ファイルではタブを使用します。

ネットワークポリシーサーバーによって生成された Microsoft NPS データベース形式ログファイルを解析するとします。そのようなファイルを以下に示します。

```
"NPS-
MASTER","IAS",03/22/2018,23:07:55,1,"user1","Domain1\user1",,,,,,,,0,"192.168.86.137","Nate
 - Test 1",,,,,,,1,,0,"311 1 192.168.0.213 03/15/2018 08:14:29
 1",,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,"Use Windows authentication for all users",1,,,,
"NPS-
MASTER","IAS",03/22/2018,23:07:55,3,,"Domain1\user1",,,,,,,,0,"192.168.86.137","Nate
 - Test 1",,,,,,,1,,16,"311 1 192.168.0.213 03/15/2018 08:14:29
 1",,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,"Use Windows authentication for all users",1,,,,
```

以下の appsettings.json 設定ファイルの例には DirectorySource 宣言が含まれています。この宣言では Delimited レコードパーサーを使用してこのテキストをオブジェクト表現に解析します。その後、JSON 形式のデータを Kinesis Data Firehose にストリーミングします。

```
{
    "Sources": [
        {
            "Id": "NPS",
            "SourceType": "DirectorySource",
            "Directory": "C:\\temp\\NPS",
            "FileNameFilter": "*.log",
            "RecordParser": "Delimited",
            "Delimiter": ",",
            "Headers": "ComputerName,ServiceName,Record-Date,Record-Time,Packet-
Type,User-Name,Fully-Qualified-Distinguished-Name,Called-Station-ID,Calling-Station-
ID,Callback-Number,Framed-IP-Address,NAS-Identifier,NAS-IP-Address,NAS-Port,Client-
Vendor,Client-IP-Address,Client-Friendly-Name,Event-Timestamp,Port-Limit,NAS-Port-
Type,Connect-Info,Framed-Protocol,Service-Type,Authentication-Type,Policy-Name,Reason-
Code,Class,Session-Timeout,Idle-Timeout,Termination-Action,EAP-Friendly-Name,Acct-
Status-Type,Acct-Delay-Time,Acct-Input-Octets,Acct-Output-Octets,Acct-Session-Id,Acct-
Authentic,Acct-Session-Time,Acct-Input-Packets,Acct-Output-Packets,Acct-Terminate-
Cause,Acct-Multi-Ssn-ID,Acct-Link-Count,Acct-Interim-Interval,Tunnel-Type,Tunnel-
Medium-Type,Tunnel-Client-Endpt,Tunnel-Server-Endpt,Acct-Tunnel-Conn,Tunnel-Pvt-
Group-ID,Tunnel-Assignment-ID,Tunnel-Preference,MS-Acct-Auth-Type,MS-Acct-EAP-Type,MS-
RAS-Version,MS-RAS-Vendor,MS-CHAP-Error,MS-CHAP-Domain,MS-MPPE-Encryption-Types,MS-
MPPE-Encryption-Policy,Proxy-Policy-Name,Provider-Type,Provider-Name,Remote-Server-
Address,MS-RAS-Client-Name,MS-RAS-Client-Version",
            "TimestampField": "{Record-Date} {Record-Time}",
            "TimestampFormat": "MM/dd/yyyy HH:mm:ss"
        }
```

```
    ],
    "Sinks": [
        {
            "Id": "npslogtest",
            "SinkType": "KinesisFirehose",
            "Region": "us-west-2",
            "StreamName": "npslogtest",
            "Format": "json"
        }
    ],
    "Pipes": [
        {
            "Id": "W3SVCLog1ToKinesisStream",
            "SourceRef": "NPS",
            "SinkRef": "npslogtest"
        }
    ]
}
```

Kinesis Data Firehose にストリーミングされた JSON 形式のデータは以下のようになります。

```
{
    "ComputerName": "NPS-MASTER",
    "ServiceName": "IAS",
    "Record-Date": "03/22/2018",
    "Record-Time": "23:07:55",
    "Packet-Type": "1",
    "User-Name": "user1",
    "Fully-Qualified-Distinguished-Name": "Domain1\\user1",
    "Called-Station-ID": "",
    "Calling-Station-ID": "",
    "Callback-Number": "",
    "Framed-IP-Address": "",
    "NAS-Identifier": "",
    "NAS-IP-Address": "",
    "NAS-Port": "",
    "Client-Vendor": "0",
    "Client-IP-Address": "192.168.86.137",
    "Client-Friendly-Name": "Nate - Test 1",
    "Event-Timestamp": "",
    "Port-Limit": "",
    "NAS-Port-Type": "",
    "Connect-Info": "",
```

```
    "Framed-Protocol": "",
    "Service-Type": "",
    "Authentication-Type": "1",
    "Policy-Name": "",
    "Reason-Code": "0",
    "Class": "311 1 192.168.0.213 03/15/2018 08:14:29 1",
    "Session-Timeout": "",
    "Idle-Timeout": "",
    "Termination-Action": "",
    "EAP-Friendly-Name": "",
    "Acct-Status-Type": "",
    "Acct-Delay-Time": "",
    "Acct-Input-Octets": "",
    "Acct-Output-Octets": "",
    "Acct-Session-Id": "",
    "Acct-Authentic": "",
    "Acct-Session-Time": "",
    "Acct-Input-Packets": "",
    "Acct-Output-Packets": "",
    "Acct-Terminate-Cause": "",
    "Acct-Multi-Ssn-ID": "",
    "Acct-Link-Count": "",
    "Acct-Interim-Interval": "",
    "Tunnel-Type": "",
    "Tunnel-Medium-Type": "",
    "Tunnel-Client-Endpt": "",
    "Tunnel-Server-Endpt": "",
    "Acct-Tunnel-Conn": "",
    "Tunnel-Pvt-Group-ID": "",
    "Tunnel-Assignment-ID": "",
    "Tunnel-Preference": "",
    "MS-Acct-Auth-Type": "",
    "MS-Acct-EAP-Type": "",
    "MS-RAS-Version": "",
    "MS-RAS-Vendor": "",
    "MS-CHAP-Error": "",
    "MS-CHAP-Domain": "",
    "MS-MPPE-Encryption-Types": "",
    "MS-MPPE-Encryption-Policy": "",
    "Proxy-Policy-Name": "Use Windows authentication for all users",
    "Provider-Type": "1",
    "Provider-Name": "",
    "Remote-Server-Address": "",
    "MS-RAS-Client-Name": "",
```

```
    "MS-RAS-Client-Version": ""
}
```

SysLog レコードパーサー

SysLog レコードパーサーの場合、解析後のソースからの出力には次の情報が含まれています。

属性	タイプ	説明
SysLogTimeStamp	文字列	syslog 形式のログファイルからの元の日付と時刻。
Hostname	文字列	syslog 形式のログファイルが存在するコンピュータの名前。
Program	文字列	ログファイルを生成したアプリケーションまたはサービスの名前。
Message	文字列	アプリケーションまたはサービスによって生成されたログメッセージ。
TimeStamp	文字列	解析された日時 (ISO 8601 形式)。

JSON に変換された SysLog データの例を次に示します。

```
{
    "SysLogTimeStamp": "Jun 18 01:34:56",
    "Hostname": "myhost1.example.mydomain.com",
    "Program": "mymailservice:",
    "Message": "Info: ICID 123456789 close",
    "TimeStamp": "2017-06-18T01:34.56.000"
}
```

Summary

以下に、DirectorySource ソースで使用可能なキーと値のペア、およびそれらのキーと値のペアに関連する RecordParser の概要を示します。

キー名	RecordParser	コメント
SourceType	すべての場合に必須	値 Directory Source が必要
Directory	すべての場合に必須	
FileNameFilter	すべての場合にオプション	
RecordParser	すべての場合に必須	
TimestampField	SingleLineJson ではオプション	
TimestampFormat	Timestamp では必須、TimestampField が指定されている場合の SingleLineJson では必須	
Pattern	Regex では必須	
ExtractionPattern	Regex ではオプション	シンクが json または xml 形式を指定している場合、Regex では必須
Delimiter	Delimited では必須	
HeaderPattern	Delimited ではオプション	
Headers	Delimited ではオプション	
RecordPattern	Delimited ではオプション	

キー名	RecordParser	コメント
CommentPattern	Delimited ではオプション	
TimeZoneKind	タイムスタンプフィールドが指定されるとき、Regex、Timestamp、SysLog、および SingleLineJson ではオプション	
SkipLines	すべての場合にオプション	

ExchangeLogSource 設定

ExchangeLogSource タイプは Microsoft Exchange からログを収集するために使用します。Exchange では、ログが複数の異なる種類のログ形式で作成されます。このソースタイプはそれらのすべてを解析します。その解析に DirectorySource タイプと Regex レコードパーサーを一緒に使用することはできますが、ExchangeLogSource を使用する方がはるかに簡単です。これは、そうしたログファイル形式用に正規表現を設計して提供する必要がないためです。次に、ExchangeLogSource 宣言の例を示します。

```
{
   "Id": "MyExchangeLog",
   "SourceType": "ExchangeLogSource",
   "Directory": "C:\\temp\\ExchangeLogTest",
   "FileNameFilter": "*.log"
}
```

すべての Exchange 宣言で、次のキーと値のペアを指定することができます。

SourceType

リテラル文字列 "ExchangeLogSource" である必要があります (必須)。

Directory

ログファイルを含むディレクトリのパス (必須)。

`FileNameFilter`

オプションで、ログデータが収集されるディレクトリにある一連のファイルを、ワイルドカードによるファイル命名パターンに基づいて制限します。このキーと値のペアが指定されていない場合、デフォルトではディレクトリにあるすべてのファイルからログデータが収集されます。

`TimestampField`

レコードの日付と時刻が含まれている列の名前。フィールド名が `date-time` または `DateTime` である場合、このキーと値のペアはオプションであり指定する必要はありません。それ以外の場合は必須です。

W3SVCLogSource 設定

`W3SVCLogSource` タイプは、Windows 用インターネットインフォメーションサービス (IIS) からログを収集するために使用します。

次に、`W3SVCLogSource` 宣言の例を示します。

```
{
   "Id": "MyW3SVCLog",
   "SourceType": "W3SVCLogSource",
   "Directory": "C:\\inetpub\\logs\\LogFiles\\W3SVC1",
   "FileNameFilter": "*.log"
}
```

すべての `W3SVCLogSource` 宣言で、次のキーと値のペアを指定することができます。

`SourceType`

リテラル文字列 `"W3SVCLogSource"` である必要があります (必須)。

`Directory`

ログファイルを含むディレクトリのパス (必須)。

`FileNameFilter`

オプションで、ログデータが収集されるディレクトリにある一連のファイルを、ワイルドカードによるファイル命名パターンに基づいて制限します。このキーと値のペアが指定されていない場合、デフォルトではディレクトリにあるすべてのファイルからログデータが収集されます。

UlsSource 設定

UlsSource タイプは Microsoft SharePoint からログを収集するために使用します。次に、UlsSource 宣言の例を示します。

```
{
    "Id": "UlsSource",
    "SourceType": "UlsSource",
    "Directory": "C:\\temp\\uls",
    "FileNameFilter": "*.log"
}
```

すべての UlsSource 宣言で、次のキーと値のペアを指定することができます。

SourceType

リテラル文字列 "UlsSource" である必要があります (必須)。

Directory

ログファイルを含むディレクトリのパス (必須)。

FileNameFilter

オプションで、ログデータが収集されるディレクトリにある一連のファイルを、ワイルドカードによるファイル命名パターンに基づいて制限します。このキーと値のペアが指定されていない場合、デフォルトではディレクトリにあるすべてのファイルからログデータが収集されます。

WindowsEventLogSource 設定

WindowsEventLogSource タイプは Windows イベントログサービスからイベントを収集するために使用します。次に、WindowsEventLogSource 宣言の例を示します。

```
{
    "Id": "mySecurityLog",
    "SourceType": "WindowsEventLogSource",
    "LogName": "Security"
}
```

すべての WindowsEventLogSource 宣言で、次のキーと値のペアを指定することができます。

`SourceType`

リテラル文字列 "`WindowsEventLogSource`" である必要があります (必須)。

`LogName`

指定されたログからイベントが収集されます。一般的な値として `Application`、`Security`、`System` がありますが、有効なすべての Windows イベントログ名を指定できます。このキーと値のペアは必須です。

`Query`

オプションで `WindowsEventLogSource` から出力されるイベントを制限します。このキーと値のペアが指定されていない場合、デフォルトでは、すべてのイベントが出力されます。この値の構文については、Windows のドキュメント Event Queries and Event XML を参照してください。ログレベルの定義については、Windows のドキュメント Event Types を参照してください。

`IncludeEventData`

オプション。このキーと値のペアの値が "`true`" であるとき、指定された Windows イベントログからイベントに関連付けられたプロバイダー固有のイベントデータを収集してストリーミングできるようにします。正常にシリアル化できるイベントデータのみが含められます。このキーと値のペアはオプションです。指定しない場合、プロバイダー固有のイベントデータは収集されません。

> **Note**
>
> イベントデータを含めることによって、このソースからストリーミングされるデータの量が大幅に増える可能性があります。イベントの最大可能サイズは、イベントデータを含めて 262,143 バイトです。

解析後の `WindowsEventLogSource` からの出力には次の情報が含まれています。

属性	タイプ	説明
`EventId`	Int	イベントタイプの識別子。
`Description`	文字列	イベントの詳細を説明するテキスト。
`LevelDisplayName`	文字列	イベントのカテゴリ (Error、Warning、Information、Success

属性	タイプ	説明
		Audit、Failure Audit のいずれか 1 つ)。
LogName	文字列	イベントが記録された場所 (一般的な値は Application 、Security、System ですが、多数の値が指定可能です)。
MachineName	文字列	イベントを記録したコンピュータ。
ProviderName	文字列	イベントを記録したアプリケーションまたはサービス。
TimeCreated	文字列	イベントが発生した時刻 (ISO 8601 形式)。
Index	Int	ログにおいてエントリが配置されている場所。
UserName	文字列	エントリを作成したユーザー (確認済みの場合)。
Keywords	文字列	イベントのタイプ。標準の値には、AuditFailure (障害が発生したセキュリティ監査イベント)、AuditSuccess (成功したセキュリティ監査イベント)、Classic (RaiseEvent 関数によって生じたイベント)、Correlation Hint (転送イベント)、SQM (サービス品質メカニズムのイベント)、WDI Context (Windows 診断インフラストラクチャのコンテキストイベント)、および WDI Diag (Windows 診断インフラストラクチャの診断イベント) が含まれます。

属性	タイプ	説明
EventData	オブジェクトのリスト	オプション。ログイベントに関するプロバイダー固有の追加データ。これは、IncludeEventData キーと値のペアの値が "true" である場合にのみ含まれます。

次に、JSON に変換されたイベントの例を示します。

```
{[
    "EventId": 7036,
    "Description": "The Amazon SSM Agent service entered the stopped state.",
    "LevelDisplayName": "Informational",
    "LogName": "System",
    "MachineName": "mymachine.mycompany.com",
    "ProviderName": "Service Control Manager",
    "TimeCreated": "2017-10-04T16:42:53.8921205Z",
    "Index": 462335,
    "UserName": null,
    "Keywords": "Classic",
    "EventData": [
    "Amazon SSM Agent",
    "stopped",
    "rPctBAMZFhYubF8zVLcrBd3bTTcNzHvY5Jc2Br0aMrxxx=="
]}
```

WindowsEventLogPollingSource 設定

WindowsEventLogPollingSourceは、ポーリングベースのメカニズムを使用して、設定されたパラメータに一致するすべての新しいイベントをイベントログから収集します。ポーリング間隔は、前回のポーリング中に収集されたイベントの数に応じて、100 ミリ秒から 5000 ミリ秒の間で動的に更新されます。次に、WindowsEventLogPollingSource 宣言の例を示します。

```
{
    "Id": "MySecurityLog",
    "SourceType": "WindowsEventLogPollingSource",
    "LogName": "Security",
    "IncludeEventData": "true",
    "Query": "",
```

```
    "CustomFilters": "ExcludeOwnSecurityEvents"
}
```

すべての WindowsEventLogPollingSource 宣言で、次のキーと値のペアを指定することができます。

SourceType

リテラル文字列 "WindowsEventLogPollingSource" である必要があります (必須)。

LogName

ログを指定します。有効なオプションは次のとおりです。Application,Security,System、またはその他の有効なログ。

IncludeEventData

省略可能。メトリックtrueの場合、JSON および XML としてストリーミングされるときに追加の EventData が含まれることを指定します。デフォルトは false です。

Query

省略可能。Windows イベントログでは、XPath 式を使用したイベントのクエリがサポートされます。Query。詳細については、「」を参照してください。イベントクエリとイベント XMLマイクロソフトのドキュメントを参照してください。

CustomFilters

省略可能。セミコロン (;). 次のフィルタを指定できます。

ExcludeOwnSecurityEvents

Windows 用 Kinesis エージェントによって生成されたセキュリティイベントを除外します。

WindowsETWEventSource 設定

WindowsETWEventSource タイプは、Windows イベントトレーシング (ETW) という機能を使用してアプリケーションおよびサービスのイベントトレースを収集するために使用します。詳細については、Windows のドキュメント Event Tracing を参照してください。

次に、WindowsETWEventSource 宣言の例を示します。

```
{
    "Id": "ClrETWEventSource",
    "SourceType": "WindowsETWEventSource",
```

```
    "ProviderName": "Microsoft-Windows-DotNETRuntime",
    "TraceLevel": "Verbose",
    "MatchAnyKeyword": 32768
}
```

すべての WindowsETWEventSource 宣言で、次のキーと値のペアを指定することができます。

SourceType

リテラル文字列 "WindowsETWEventSource" である必要があります (必須)。

ProviderName

トレースイベントを収集するために使用するイベントプロバイダーを指定します。これは、インストールされたプロバイダーの有効な ETW プロバイダー名にする必要があります。インストールされているプロバイダーを確認するには、Windows コマンドプロンプトウィンドウで次のコマンドを実行します。

```
logman query providers
```

TraceLevel

収集する必要があるトレースイベントのカテゴリを指定します。使用できる値は、Critical、Error、Warning、Informational、および Verbose です。選択されている ETW プロバイダーによって、正確な意味は異なります。

MatchAnyKeyword

この値は 64 ビットの数値で、各ビットは個別のキーワードを表します。各キーワードは、収集されるイベントのカテゴリを示します。サポートされるキーワードとその値、およびそれらと TraceLevel の関係については、そのプロバイダーのドキュメントを参照してください。たとえば、CLR ETW プロバイダーについては Microsoft .NET Framework ドキュメントの「CLR ETW キーワードおよびレベル」を参照してください。

前の例で、32768 (0x00008000) は、スローされる例外に関する情報を収集するようプロバイダーに指示する、CLR ETW プロバイダーの ExceptionKeyword を表します。16 進定数は JSON ではネイティブにサポートされませんが、それらを文字列に配置することによって MatchAnyKeyword に指定できます。いくつかの定数をカンマで区切って指定することもできます。たとえば、次のように使用して ExceptionKeyword と SecurityKeyword (0x00000400) の両方を指定します。

```
{
```

```
    "Id": "MyClrETWEventSource",
    "SourceType": "WindowsETWEventSource",
    "ProviderName": "Microsoft-Windows-DotNETRuntime",
    "TraceLevel": "Verbose",
    "MatchAnyKeyword": "0x00008000, 0x00000400"
}
```

指定したすべてのキーワードがプロバイダーに対して有効であることを保証するため、複数のキーワード値は OR を使用して結合されてそのプロバイダーに渡されます。

WindowsETWEventSource からの出力には、イベントごとに次の情報が含まれています。

属性	タイプ	説明
EventName	文字列	発生したイベントの種類。
ProviderName	文字列	イベントを検出したプロバイダー。
FormattedMessage	文字列	テキストによるイベントの概要。
ProcessID	Int	イベントを報告したプロセス。
ExecutingThreadID	Int	イベントを報告したプロセス内のスレッド。
MachineName	文字列	イベントを報告しているデスクトップまたはサーバーの名前。
Payload	ハッシュテーブル	文字列キーと、値として任意の種類のオブジェクトを持つテーブル。キーはペイロード項目名であり、値はペイロード項目の値です。ペイロードはプロバイダーに依存します。

次に、JSON に変換されたイベントの例を示します。

```
{
     "EventName": "Exception/Start",
     "ProviderName": "Microsoft-Windows-DotNETRuntime",
     "FormattedMessage": "ExceptionType=System.Exception;\r
\nExceptionMessage=Intentionally unhandled exception.;\r\nExceptionEIP=0x2ab0499;\r
\nExceptionHRESULT=-2,146,233,088;\r\nExceptionFlags=CLSCompliant;\r\nClrInstanceID=9
 ",
     "ProcessID": 3328,
     "ExecutingThreadID": 6172,
     "MachineName": "MyHost.MyCompany.com",
     "Payload":
      {
        "ExceptionType": "System.Exception",
        "ExceptionMessage": "Intentionally unhandled exception.",
        "ExceptionEIP": 44762265,
        "ExceptionHRESULT": -2146233088,
        "ExceptionFlags": 16,
        "ClrInstanceID": 9
      }
}
```

WindowsPerformanceCounterSource 設定

WindowsPerformanceCounterSource タイプは、Windows からパフォーマンスカウンターメトリクスを収集します。次に、WindowsPerformanceCounterSource 宣言の例を示します。

```
{
 "Id": "MyPerformanceCounter",
 "SourceType": "WindowsPerformanceCounterSource",
 "Categories": [{
   "Category": "Server",
   "Counters": ["Files Open", "Logon Total", "Logon/sec", "Pool Nonpaged Bytes"]
  },
  {
   "Category": "System",
   "Counters": ["Processes", "Processor Queue Length", "System Up Time"]
  },
  {
   "Category": "LogicalDisk",
   "Instances": "*",
   "Counters": [
    "% Free Space", "Avg. Disk Queue Length",
```

```
    {
     "Counter": "Disk Reads/sec",
     "Unit": "Count/Second"
    },
    "Disk Writes/sec"
   ]
  },
  {
   "Category": "Network Adapter",
   "Instances": "^Local Area Connection\* \d$",
   "Counters": ["Bytes Received/sec", "Bytes Sent/sec"]
  }
 ]
}
```

すべての WindowsPerformanceCounterSource 宣言で、次のキーと値のペアを指定することができます。

SourceType

リテラル文字列 "WindowsPerformanceCounterSource" である必要があります (必須)。

Categories

Windows から収集するための一連のパフォーマンスカウンターメトリクスグループを指定します。各メトリクスグループには、次のキーと値のペアが含まれます。

Category

収集されるメトリクスのカウンターセットを指定します (必須)。

Instances

オブジェクトごとに固有の一連のパフォーマンスカウンターがあるときに、目的のオブジェクトのセットを指定します。たとえば、カテゴリが LogicalDisk であるとき、ディスクドライブごとに一連のパフォーマンスカウンターが存在します。このキーと値のペアはオプションです。複数のインスタンスに一致させるには、ワイルドカード * と ? を使用できます。すべてのインスタンス間で値を集計するには、_Total を指定します。

また、InstanceRegexを含む正規表現を受け入れ、*ワイルドカード文字を、インスタンス名の一部として使用します。

`Counters`

指定されたカテゴリについて収集するメトリクスを指定します。このキーと値のペアは必須です。複数のカウンターに一致させるには、ワイルドカード * と ? を使用できます。`Counters` を指定するには、名前のみを使用するか、名前と単位を使用することができます。カウンターの単位が指定されていない場合、Kinesis Agent for Windows は名前から単位を推測しようとします。それらの推測が正しくない場合には、単位を明示的に指定できます。必要に応じて `Counter` 名を変更できます。より複雑な表現のカウンターは、オブジェクトに次のキーと値のペアを加えた表現になります。

`Counter`

カウンターの名前。このキーと値のペアは必須です。

`Rename`

シンクに提示するカウンターの名前。このキーと値のペアはオプションです。

`Unit`

カウンターに関連付けられている値の意味。有効なユニット名の詳細なリストについては、MetricDatum()Amazon CloudWatch API リファレンス。

複雑なカウンターの指定の例を次に示します。

```
{
   "Counter": "Disk Reads/sec,
   "Rename": "Disk Reads per second",
   "Unit": "Count/Second"
}
```

`WindowsPerformanceCounterSource`Amazon CloudWatch シンクを指定するパイプでのみ使用できます。Windows 用 Kinesis Agent for Windows 組み込みメトリクスも CloudWatch にストリーミングされる場合は、別個のシンクを使用します。サービス起動後に Kinesis Agent for Windows ログを調べて、単位が`WindowsPerformanceCounterSource`宣言. カテゴリ、インスタンス、およびカウンターの有効な名前を確認するには、PowerShell を使用します。

カウンターセットに関連するカウンターを含め、すべてのカテゴリに関する情報を見るには、PowerShell ウィンドウで次のコマンドを実行します。

```
Get-Counter -ListSet * | Sort-Object
```

カウンターセットの各カウンターで使用できるインスタンスを確認するには、PowerShell ウィンドウで次の例のようなコマンドを実行します。

```
Get-Counter -Counter "\Process(*)\% Processor Time"
```

Counter パラメータの値は、前の Get-Counter -ListSet コマンド呼び出しによってリストされた PathsWithInstances メンバーのいずれかのパスになります。

Windows 組み込みメトリクスソース Kinesis エージェント

通常のメトリックスソースに加えて、WindowsPerformanceCounterSourceタイプ (WindowsPerformanceCounterSource 設定)、CloudWatch シンクタイプは Windows 自体に関する Kinesis Agent に関するメトリクスを収集する特別なソースからメトリクスを受け取ることができます。Windows の Kinesis エージェントメトリクスは、KinesisTapWindows パフォーマンスカウンターのカテゴリ。

-MetricsFilterCloudWatch シンク宣言のキーと値のペアは、組み込みの Kinesis エージェントの Windows メトリクスソースから CloudWatch にストリーミングされるメトリクスを指定します。この値は、1 つのフィルター式、またはセミコロンで区切られた複数のフィルター式を含む文字列です。次に例を示します。

"MetricsFilter": "FilterExpression1;FilterExpression2"

1 つ以上のフィルター式に一致するメトリクスが CloudWatch にストリーミングされます。

単一インスタンスメトリクスは性質上、グローバルであり、特定のソースやシンクに関連付けられていません。複数インスタンスメトリクスは、ソース宣言またはシンク宣言の Id に基づく側面があります。各ソースタイプまたはシンクタイプは、さまざまな一連のメトリクスを持つことができます。

Windows 用 Kinesis エージェントの組み込みのメトリクス名の一覧については、Windows メトリックス用 Kinesis エージェントのリスト。

単一インスタンスメトリクスの場合、フィルター式はメトリクスの名前です。次に例を示します。

```
"MetricsFilter": "SourcesFailedToStart;SinksFailedToStart"
```

複数インスタンスメトリクスの場合、フィルター式は、メトリクスの名前、ピリオド (.)、そのメトリクスを生成したソース宣言またはシンク宣言の Id になります。たとえば、Id が MyFirehose であるシンク宣言があるとします。

```
"MetricsFilter": "KinesisFirehoseRecordsFailedNonrecoverable.MyFirehose"
```

単一と複数のインスタンスメトリクスを区別するように設計されている特別なワイルドカードパターンを使用できます。

- アスタリスク (*) は、ピリオド (.) を除くゼロ個以上の文字に一致します。
- 疑問符 (?) は、ピリオドを除く 1 文字に一致します。
- その他の文字は、その文字自体にのみ一致します。
- _Total は、ディメンション間で一致するすべての複数インスタンス値を集計する特別なトークンです。

次の例は、すべての単一インスタンスメトリクスに一致します。

```
"MetricsFilter": "*"
```

アスタリスクはピリオド文字に一致しないため、単一インスタンスメトリクスのみが含められます。

次の例は、すべての複数インスタンスメトリクスに一致します。

```
"MetricsFilter": "*.*"
```

次の例は、すべてのメトリクス (単一と複数) に一致します。

```
"MetricsFilter": "*;*.*"
```

次の例は、すべてのソースおよびシンク間で複数インスタンスメトリクスをすべて集計します。

```
"MetricsFilter": "*._Total"
```

次の例は、すべての Kinesis Data Firehose シンクの Kinesis データ Firehose メトリクスを集計します。

```
"MetricsFilter": "*Firehose*._Total"
```

次の例は、すべての単一および複数インスタンスエラーメトリクスに一致します。

```
"MetricsFilter": "*Failed*;*Error*.*;*Failed*.*"
```

次の例は、すべてのソースおよびシンク間で集計されたすべての回復不能エラーメトリクスに一致します。

```
"MetricsFilter": "*Nonrecoverable*._Total"
```

Kinesis Agent for Windows 組み込みメトリクスソースを使用するパイプの指定方法については、Windows メトリックパイプ用の Kinesis エージェントの設定。

Windows メトリックス用 Kinesis エージェントのリスト

以下は、Windows 用 Kinesis Agent で使用可能な単一インスタンスメトリクスと複数インスタンスメトリクスの一覧です。

単一インスタンスメトリクス

次の単一インスタンスメトリクスを使用できます。

`KinesisTapBuildNumber`

Windows 用 Kinesis エージェントのバージョン番号。

`PipesConnected`

正常にソースをシンクに接続したパイプの数。

`PipesFailedToConnect`

正常にソースをシンクに接続しなかったパイプの数。

`SinkFactoriesFailedToLoad`

Windows 用 Kinesis エージェントに正常にロードされなかったシンクタイプの数。

SinkFactoriesLoaded

Windows 用 Kinesis エージェントに正常にロードされたシンクタイプの数。

SinksFailedToStart

通常はシンク宣言が正しくないことが原因で、正常に開始されなかったシンクの数。

SinksStarted

正常に開始されたシンクの数。

SourcesFailedToStart

通常はソース宣言が正しくないことが原因で、正常に開始されなかったソースの数。

SourcesStarted

正常に開始されたソースの数。

SourceFactoriesFailedToLoad

Windows 用 Kinesis エージェントに正常にロードされなかったソースタイプの数。

SourceFactoriesLoaded

Windows 用 Kinesis エージェントに正常にロードされたソースタイプの数。

複数インスタンスメトリクス

次の複数インスタンスメトリクスを使用できます。

DirectorySource メトリクス

DirectorySourceBytesRead

この DirectorySource について、この間隔で読み取られたバイト数。

DirectorySourceBytesToRead

Kinesis Agent for Windows によってまだ読み取られていない、読み取り可能な既知のバイト数。

DirectorySourceFilesToProcess

Windows 用 Kinesis Agent for Windows によって検証されていない、既知の検証対象ファイル数。

`DirectorySourceRecordsRead`

この `DirectorySource` について、この間隔で読み取られたレコード数。

WindowsEventLogSource メトリクス

`EventLogSourceEventsError`

正常に読み取られなかった Windows イベントログイベント数。

`EventLogSourceEventsRead`

正常に読み取られた Windows イベントログイベント数。

KinesisFirehose シンクメトリクス

`KinesisFirehoseBytesAccepted`

この間隔で受け入れられたバイト数。

`KinesisFirehoseClientLatency`

レコードの生成からレコードの Kinesis Data Firehose サービスへのストリーミングまでの経過時間。

`KinesisFirehoseLatency`

Kinesis Data Firehose サービスに対するレコードのストリーミングの開始から終了までの経過時間。

`KinesisFirehoseNonrecoverableServiceErrors`

再試行したにもかかわらず、エラーなしでレコードを Kinesis Data Firehose サービスに送信できなかった回数。

`KinesisFirehoseRecordsAttempted`

Kinesis Data Firehose サービスへのストリーミングを試みたレコード数。

`KinesisFirehoseRecordsFailedNonrecoverable`

再試行したにもかかわらず、Kinesis Data Firehose サービスにストリーミングできなかったレコード数。

`KinesisFirehoseRecordsFailedRecoverable`

再試行した場合にのみ、Kinesis Data Firehose サービスにストリーミングが正常に行われたレコード数。

`KinesisFirehoseRecordsSuccess`

再試行せずに、Kinesis Data Firehose サービスへのストリーミングが正常に行われたレコード数。

`KinesisFirehoseRecoverableServiceErrors`

再試行した場合にのみ、Kinesis Data Firehose サービスにレコードを正常に送信できた回数。

KinesisStream メトリクス

`KinesisStreamBytesAccepted`

この間隔で受け入れられたバイト数。

`KinesisStreamClientLatency`

レコードの生成からレコードの Kinesis Data Streams サービスへのストリーミングまでの経過時間。

`KinesisStreamLatency`

Kinesis Data Streams サービスに対するレコードのストリーミングの開始から終了までの経過時間。

`KinesisStreamNonrecoverableServiceErrors`

再試行したにもかかわらず、エラーなしでレコードを Kinesis Data Streams サービスに送信できなかった回数。

`KinesisStreamRecordsAttempted`

Kinesis Data Streams サービスへのストリーミングを試みたレコード数。

`KinesisStreamRecordsFailedNonrecoverable`

再試行したにもかかわらず、Kinesis Data Streams サービスへのストリーミングが正常に行われなかったレコード数。

`KinesisStreamRecordsFailedRecoverable`

再試行した場合にのみ、Kinesis Data Streams サービスにストリーミングが正常に行われたレコード数。

`KinesisStreamRecordsSuccess`

再試行せずに、Kinesis Data Streams サービスへのストリーミングが正常に行われたレコード数。

`KinesisStreamRecoverableServiceErrors`

再試行した場合にのみ、Kinesis Data Streams サービスにレコードを正常に送信できた回数。

CloudWatchLog メトリクス

`CloudWatchLogBytesAccepted`

この間隔で受け入れられたバイト数。

`CloudWatchLogClientLatency`

レコード生成からレコードの CloudWatch Logs サービスへのストリーミングまでの経過時間。

`CloudWatchLogLatency`

CloudWatch Logs サービスに対するレコードストリーミングの開始から終了までの経過時間。

`CloudWatchLogNonrecoverableServiceErrors`

再試行したにもかかわらず、エラーなしでレコードを CloudWatch Logs サービスに送信できなかった回数。

`CloudWatchLogRecordsAttempted`

CloudWatch Logs サービスへのストリーミングを試みたレコード数。

`CloudWatchLogRecordsFailedNonrecoverable`

再試行したにもかかわらず、CloudWatch Logs サービスへのストリーミングが正常に行われなかったレコード数。

`CloudWatchLogRecordsFailedRecoverable`

再試行した場合にのみ、CloudWatch Logs サービスへのストリーミングが正常に行われたレコード数。

`CloudWatchLogRecordsSuccess`

再試行せずに、CloudWatch Logs サービスへのストリーミングが正常に行われたレコード数。

`CloudWatchLogRecoverableServiceErrors`

再試行した場合にのみ、CloudWatch Logs サービスにレコードを正常に送信できた回数。

CloudWatch Metrics

`CloudWatchLatency`

CloudWatch サービスに対するメトリクスのストリーミングの開始から終了までの平均経過時間。

`CloudWatchNonrecoverableServiceErrors`

再試行したにもかかわらず、メトリクスがエラーなしで CloudWatch サービスに送信できなかった回数。

`CloudWatchRecoverableServiceErrors`

再試行した場合にのみ、メトリクスがエラーなしで CloudWatch サービスに送信された回数。

`CloudWatchServiceSuccess`

再試行を必要とせずに、メトリクスがエラーなしで CloudWatch サービスに送信された回数。

ブックマーク設定

デフォルトでは、Kinesis Agent for Windows は、エージェントの開始後に作成されたログレコードをシンクに送信します。以前のログレコードを送信すると便利な場合もあります。たとえば、自動更新中に Kinesis Agent for Windows が停止した期間に作成されたログレコードなどです。ブックマーク機能は、シンクに送信されたレコードを追跡します。Windows 用 Kinesis エージェントがブックマークモードで起動すると、Kinesis Agent for Windows の停止後に作成されたすべてのログレコードを、続いて作成されたログレコードとともに送信します。この動作を制御するため、ファイルベースのソース宣言に次のキーと値のペアをオプションで含めることができます。

`InitialPosition`

ブックマークの初期状態を指定します。指定できる値は次のとおりです。

`EOS`

ストリームの終了 (EOS) を指定します。エージェントの実行中に作成されたログレコードのみがシンクに送信されます。

`0`

使用可能なすべてのログレコードとイベントが最初に送信されます。その後でブックマークが作成されて、Kinesis Agent for Windows が実行されているかどうかに関係なく、ブックマークの作成後に作成された新しいログレコードとイベントがすべて最終的に送信されるようになります。

`Bookmark`

ブックマークは、最新のログレコードまたはイベントの直後の状態に初期化されます。その後でブックマークが作成されて、Kinesis Agent for Windows が実行されているかどうかに関係なく、ブックマークの作成後に作成された新しいログレコードとイベントがすべて最終的に送信されるようになります。

ブックマークはデフォルトで有効になっています。ファイルは`%ProgramData%\Amazon\KinesisTap`ディレクトリに移動します。

`Timestamp`

`InitialPositionTimestamp` 値 (定義は下記) の後に作成されたログレコードとイベントが送信されます。その後でブックマークが作成されて、Kinesis Agent for Windows が実行されているかどうかに関係なく、ブックマークの作成後に作成された新しいログレコードとイベントがすべて最終的に送信されるようになります。

`InitialPositionTimestamp`

必要な最も古いログレコードまたはイベントのタイムスタンプを指定します。このキーと値のペアは、`InitialPosition` に値 `Timestamp` が含まれている場合にのみ指定します。

`BookmarkOnBufferFlush`

この設定は、ブックマーク可能なソースに追加できます。に設定した場合`true`に設定すると、ブックマークの更新は、シンクがイベントを AWS に正常に送信したときにのみ行われるようになります。1 つのソースにサブスクライブできるシンクは 1 つだけです。ログを複数の宛先に配布する場合は、データの損失に関する潜在的な問題を回避するために、ソースを複製してください。

Kinesis Agent for Windows が長時間停止しているときは、ブックマークされたログレコードやイベントが存在しなくなることがあるため、それらのブックマークの削除が必要になる可能性があります。指定の source id のブックマークファイルは `%PROGRAMDATA%\Amazon\AWSKinesisTap\source id.bm` にあります。

名前が変更されたり切り捨てられたファイルにあるブックマークは機能しません。ETW イベントおよびパフォーマンスカウンターは、その性質上、ブックマークできません。

シンク宣言

シンク宣言は、ログ、イベント、メトリクスがさまざまな AWS サービスに送信される場所と形式を指定します。以下のセクションでは、Microsoft Windows の Amazon Kinesis Agent で使用可能な組み込みシンクタイプの設定について説明します。Windows 用 Kinesis Agent は拡張可能であるため、カスタムのシンクタイプを追加できます。通常、各シンクタイプには、そのシンクタイプに関連する、設定宣言の一意のキーと値のペアが必要です。

すべてのシンク宣言には、次のキーと値のペアを含めることができます。

`Id`

設定ファイル内で特定のシンクを識別する一意の文字列 (必須)。

`SinkType`

このシンクのシンクタイプの名前 (必須)。シンクタイプは、このシンクによってストリーミングされるログ、イベント、またはメトリクスのデータの送信先を指定します。

`AccessKey`

シンクタイプに関連付けられている AWS サービスへのアクセスを許可するときに使用する AWS アクセスキーを指定します。このキーと値のペアはオプションです。詳細については、「シンクセキュリティの設定」を参照してください。

`SecretKey`

シンクタイプに関連付けられている AWS サービスへのアクセスを許可するときに使用する AWS シークレットキーを指定します。このキーと値のペアはオプションです。詳細については、「シンクセキュリティの設定」を参照してください。

`Region`

ストリーミングの送信先リソースが含まれる AWS リージョンを指定します。このキーと値のペアはオプションです。

`ProfileName`

認証に使用する AWS プロファイルを指定します。このキーと値のペアはオプションですが、指定した場合は、その値のペアが指定したアクセスキーとシークレットキーよりも優先されます。詳細については、「シンクセキュリティの設定」を参照してください。

`RoleARN`

シンクタイプに関連付けられている AWS サービスにアクセスするときに使用する IAM ロールを指定します。このオプションは、EC2 インスタンスで実行されている Kinesis Agent for Windows が EC2 インスタンスで実行されている場合に役立ちますが、インスタンスプロファイルによって参照されるロールよりも別のロールの方が適切です。たとえば、クロスアカウントロールを使用して、EC2 インスタンスと同じ AWS アカウントにないリソースを対象にすることができます。このキーと値のペアはオプションです。

`Format`

ストリーミングの前にログおよびイベントデータに適用されるシリアル化の種類を指定します。有効な値は、`json` および `xml` です。このオプションは、データパイプラインのダウンストリーム分析で特別の形式のデータが必要であるか、特別の形式のデータの使用が好ましい場合に役立ちます。このキーと値のペアはオプションです。指定しない場合は、ソースの通常のテキストがシンクからシンクタイプに関連付けられている AWS サービスにストリーミングされます。

`TextDecoration`

`Format` を指定しない場合、`TextDecoration` はログレコードまたはイベントレコードのストリーミング時に含まれる追加のテキストを指定します。詳細については、「シンクデコレーションの設定」を参照してください。このキーと値のペアはオプションです。

`ObjectDecoration`

`Format` を指定した場合、`ObjectDecoration` はシリアル化およびストリーミングの前にログレコードまたはイベントレコードに含まれる追加のデータを指定します。詳細については、「シンクデコレーションの設定」を参照してください。このキーと値のペアはオプションです。

`BufferInterval`

シンクタイプに関連付けられている AWS サービスの API コールを最小限に抑えるために、Kinesis Agent for Windows では、ストリーミングの前にログ、イベント、またはメトリクスの複数のレコードをバッファします。これにより、API コールあたりのサービス料金を節約することができます。`BufferInterval` は AWS サービスにストリーミングする前にレコードをバッファする必要がある最大時間 (秒) を指定します。このキーと値のペアはオプションです。指定する場合、値を表す文字列を使用します。

`BufferSize`

シンクタイプに関連付けられている AWS サービスの API コールを最小限に抑えるために、Kinesis Agent for Windows では、ストリーミングの前にログ、イベント、またはメトリクスの複数のレコードをバッファします。これにより、API コールあたりのサービス料金を節約することができます。`BufferSize` は AWS サービスにストリーミングする前にバッファするレコードの最大数を指定します。このキーと値のペアはオプションです。指定する場合、値を表す文字列を使用します。

`MaxAttempts`

ストリーミングが何度も失敗する場合に Kinesis Agent for Windows が AWS サービスにログ、イベント、およびメトリクスの一連のレコードをストリーミングする最大試行回数を指定します。このキーと値のペアはオプションです。指定する場合、値を表す文字列を使用します。デフォルト値は `3` です。

さまざまな種類のシンクを使用する完全な設定ファイルの例については、Windows アプリケーションイベントログからシンクへのストリーミングを参照してください。

トピック

- KinesisStream シンクの設定
- KinesisFirehose シンクの設定
- CloudWatch シンクの設定
- CloudWatchLogs シンクの設定
- ローカルFileSystemシンクの設定
- シンクセキュリティの設定
- の設定ProfileRefreshingAWSCredentialProviderAWS 認証情報の更新
- シンクデコレーションの設定
- シンク変数の置換の設定
- シンクのキューイングの設定
- シンクのプロキシの設定
- より多くのシンク属性での変数の解決の設定
- AWS シンクで RoleARN プロパティを使用する場合の AWS STS リージョンエンドポイントの設定

- AWS シンク用の VPC エンドポイントの設定
- プロキシの代替手段の設定

KinesisStream シンクの設定

-KinesisStreamシンクタイプは、Kinesis Data Streams サービスにレコードおよびイベントをログします。通常、Kinesis Data Streams にストリーミングされるデータは、さまざまな AWS サービスを使用して実行する 1 つ以上のカスタムアプリケーションによって処理されます。データは、Kinesis Data Streams を使用して設定された名前付きストリームにストリーミングされます。詳細については、「」を参照してください。Amazon Kinesis Data Streams 開発者ガイド。

次に、Kinesis Data Streams のシンク宣言の例を示します。

```
{
    "Id": "TestKinesisStreamSink",
    "SinkType": "KinesisStream",
    "StreamName": "MyTestStream",
    "Region": "us-west-2"
}
```

KinesisStream のすべてのシンク宣言には、次の追加のキーと値のペアを含めることができます。

SinkType

このペアを指定する必要があります。値はリテラル文字列 KinesisStream にする必要があります。

StreamName

ストリーミングされたデータを受信する Kinesis データストリームの名前を指定します。KinesisStreamシンクタイプ (必須)。データをストリーミングする前に、または Kinesis Data Streams API を使用したアプリケーションを介してストリームを設定します。

RecordsPerSecond

1 秒あたりに Kinesis Data Streams にストリーミングされるレコードの最大数を指定します。このキーと値のペアはオプションです。指定する場合、値を表す整数を使用します。デフォルト値は 1000 レコードです。

BytesPerSecond

1 秒あたりに Kinesis Data Streams にストリーミングされる最大バイト数を指定します。このキーと値のペアはオプションです。指定する場合、値を表す整数を使用します。デフォルト値は 1 MB です。

このシンクタイプのデフォルトの BufferInterval は 1 秒で、デフォルトの BufferSize は 500 レコードです。

KinesisFirehose シンクの設定

-KinesisFirehoseシンクタイプは、ログレコードおよびイベントを Kinesis Data Firehose サービスにストリーミングします。Kinesis Data Firehose は、ストリーミングされたデータをストレージ用にその他のサービスにストリーミングします。通常、保存されたデータは、データパイプラインのその後のステージで分析されます。データは、Kinesis Data Firehose を使用して設定された名前付き配信ストリームにストリーミングされます。詳細については、「」を参照してください。Amazon Kinesis Data Firehose 開発者ガイド。

次に、Kinesis Data Firehose シンク宣言の例を示します。

```
{
   "Id": "TestKinesisFirehoseSink",
   "SinkType": "KinesisFirehose",
   "StreamName": "MyTestFirehoseDeliveryStream",
   "Region": "us-east-1",
   "CombineRecords": "true"
}
```

KinesisFirehose のすべてのシンク宣言には、次の追加のキーと値のペアを含めることができます。

SinkType

このペアを指定する必要があります。値はリテラル文字列 KinesisFirehose にする必要があります。

`StreamName`

ストリーミングされたデータを受信する Kinesis Data Firehose 配信ストリームの名前を指定します。`KinesisStream`シンクタイプ (必須)。データをストリーミングする前に、あるいは Kinesis Data Firehose AWS を使用したアプリケーションを介して配信ストリームを設定します。

`CombineRecords`

に設定した場合`true`の場合、複数の小さなレコードを、最大サイズが 5 KB の大きいレコードに結合することを指定します。このキーと値のペアはオプションです。この関数を使用して結合されたレコードは`\n`。AWS Lambda を使用して Kinesis Data Firehose レコードを変換する場合、Lambda 関数で区切り文字を考慮する必要があります。

`RecordsPerSecond`

1 秒あたりに Kinesis Data Streams にストリーミングされるレコードの最大数を指定します。このキーと値のペアはオプションです。指定する場合、値を表す整数を使用します。デフォルト値は 5000 レコードです。

`BytesPerSecond`

1 秒あたりに Kinesis Data Streams にストリーミングされる最大バイト数を指定します。このキーと値のペアはオプションです。指定する場合、値を表す整数を使用します。デフォルト値は 5 MB です。

このシンクタイプのデフォルトの `BufferInterval` は 1 秒で、デフォルトの `BufferSize` は 500 レコードです。

CloudWatch シンクの設定

-`CloudWatch`シンクタイプは、メトリクスを CloudWatch サービスにストリーミングします。AWS マネジメントコンソールでメトリクスを表示できます。詳細については、Amazon CloudWatch ユーザーガイドを参照してください。

次に、`CloudWatch` のシンク宣言の例を示します。

```
{
   "Id": "CloudWatchSink",
   "SinkType": "CloudWatch"
}
```

CloudWatch のすべてのシンク宣言には、次の追加のキーと値のペアを含めることができます。

SinkType

このペアを指定する必要があります。値はリテラル文字列 CloudWatch にする必要があります。

Interval

Windows 用 Kinesis Agent for Windows が CloudWatch サービスにメトリクスをレポートする頻度 (秒) を指定します。このキーと値のペアはオプションです。指定する場合、値を表す整数を使用します。デフォルト値は 60 秒です。高解像度の CloudWatch メトリクスが必要な場合は 1 秒を指定します。

Namespace

メトリクスデータがレポートされる CloudWatch 名前空間を指定します。CloudWatch 名前空間により、一連のメトリクスがグループ化されます。このキーと値のペアはオプションです。デフォルト値は KinesisTap です。

Dimensions

名前空間内のメトリクスセットの分離に使用する CloudWatch ディメンションを指定します。これは、たとえばデスクトップまたはサーバーごとに個別のメトリクスセットを提供する上で有益です。このキーと値のペアはオプションです。指定する場合、値は "key1=value1;key2=value2..." の形式に準拠する必要があります。デフォルト値は "ComputerName={computername};InstanceId={instance_id}" です。この値はシンク変数の置換をサポートしています。詳細については、「シンク変数の置換の設定」を参照してください。

MetricsFilter

組み込み Kinesis Agent for Windows メトリクスソースから CloudWatch にストリーミングされるメトリクスを指定します。組み込みの Kinesis Agent for Windows メトリクスソースの詳細 (このキーと値のペアの値の構文の詳細など) については、Windows 組み込みメトリクスソース Kinesis エージェント。

CloudWatchLogs シンクの設定

-CloudWatchLogsシンクタイプは、ログレコードおよびイベントを Amazon CloudWatch Logs にストリーミングします。AWS マネジメントコンソールでログを表示したり、データパイプラインの

追加のステージでログを処理したりすることができます。データは、CloudWatch Logs で設定された名前付きログストリームにストリーミングされます。ログストリームは名前付きロググループにまとめられます。詳細については、 Amazon CloudWatch ログユーザーガイドを参照してください。

CloudWatch Logs シンク宣言の例を以下に示します。

```
{
   "Id": "MyCloudWatchLogsSink",
   "SinkType": "CloudWatchLogs",
   "BufferInterval": "60",
   "BufferSize": "100",
   "Region": "us-west-2",
   "LogGroup": "MyTestLogGroup",
   "LogStream": "MyTestStream"
}
```

CloudWatchLogs のすべてのシンク宣言には、次の追加のキーと値のペアを入力する必要があります。

SinkType

リテラル文字列 CloudWatchLogs である必要があります。

LogGroup

によってストリーミングされたログレコードおよびイベントレコードを受信するログストリームを含む CloudWatch Logs ロググループの名前を指定します。CloudWatchLogsシンクタイプ。指定されたロググループが存在しない場合は、Kinesis Agent for Windows によってその作成が試行されます。

LogStream

によってログレコードおよびイベントレコードストリームを受信する CloudWatch Logs ログストリームの名前を指定します。CloudWatchLogsシンクタイプ。この値はシンク変数の置換をサポートしています。詳細については、「シンク変数の置換の設定」を参照してください。指定されたログストリームが存在しない場合は、Kinesis Agent for Windows によってその作成が試行されます。

このシンクタイプのデフォルトの BufferInterval は 1 秒で、デフォルトの BufferSize は 500 レコードです。最大バッファサイズは 10,000 レコードです。

ローカルFileSystemシンクの設定

シンクのタイプFileSystemは、ログおよびイベントレコードを AWS サービスにストリーミングするのではなく、ローカルファイルシステム上のファイルに保存します。FileSystemシンクは、テストや診断に役立ちます。たとえば、このシンクタイプを使用して、レコードを AWS に送信する前に調べることができます。

とFileSystemシンクでは、設定パラメーターを使用して、バッチ処理、スロットリング、再試行をシミュレートして、実際の AWS シンクの動作を模倣することもできます。

接続されているすべてのソースからのすべてのレコードFileSystemシンクは、FilePath。もしFilePathが指定されなかった場合、レコードは*SinkId*.txt()%TEMP%ディレクトリーです。通常はC:\Users*UserName*\AppData\Local\Tempここで、とします。*SinkId*シンクの一意の識別子であり、*UserName*は、アクティブなユーザーの Windows ユーザー名です。

このシンクタイプは、テキストの装飾属性をサポートします。詳細については、「シンクデコレーションの設定」を参照してください。

例FileSystemシンクタイプの設定を以下の例で示します。

```
{
    "Id": "LocalFileSink",
    "SinkType": "FileSystem",
    "FilePath": "C:\\ProgramData\\Amazon\\local_sink.txt",
    "Format": "json",
    "TextDecoration": "",
    "ObjectDecoration": ""
}
```

-FileSystem設定は、以下のキーと値のペアで構成されます。

SinkType

リテラル文字列 FileSystem である必要があります。

FilePath

レコードを保存するパスとファイルを指定します。このキーと値のペアはオプションです。指定されなかった場合、デフォルト値は*TempPath**SinkId*.txtここで、とします。*TempPath*に格納されているフォルダです。%TEMP%変数と変数*SinkId*シンクの一意の識別子です。

`Format`

イベントの形式を指定しますjsonまたはxml。このキー値のペアはオプションで、大文字と小文字は区別されません。省略すると、イベントはプレーンテキストでファイルに書き込まれます。

`TextDecoration`

プレーンテキストで記述されたイベントにのみ適用されます。このキーと値のペアはオプションです。

`ObjectDecoration`

次のイベントにのみ適用されます。Formatは、 に設定されます。json。このキーと値のペアはオプションです。

高度な使用法 — 記録スロットリングと障害シミュレーション

FileSystemは、レコードスロットリングをシミュレートすることで、AWS シンクの動作を模倣できます。次のキーと値のペアを使用して、レコードスロットルおよび障害シミュレーション属性を指定できます。

宛先ファイルのロックを取得し、そのファイルへの書き込みを防止することで、FileSystemシンクを使用して、ネットワークに障害が発生したときの AWS シンクの動作をシミュレートし、調べます。

以下の例は、以下を示していますFileSystem構成をシミュレーション属性で設定します。

```
{
    "Id": "LocalFileSink",
    "SinkType": "FileSystem",
    "FilePath": "C:\\ProgramData\\Amazon\\local_sink.txt",
    "TextDecoration": "",
    "RequestsPerSecond": "100",
    "BufferSize": "10",
    "MaxBatchSize": "1024"
}
```

`RequestsPerSecond`

オプションで、文字列型として指定します。省略した場合、デフォルト値は"5"。レコード数ではなく、シンクが処理する要求 (ファイルへの書き込み) のレートを制御します。Kinesis Agent

for Windows は AWS エンドポイントに対してバッチリクエストを行うため、リクエストに複数のレコードが含まれる場合があります。

`BufferSize`

オプションで、文字列型として指定します。ファイルに保存する前にシンクがバッチ処理するイベントレコードの最大数を指定します。

`MaxBatchSize`

オプションで、文字列型として指定します。ファイルに保存する前にシンクがバッチ処理するイベントレコードデータの最大量をバイト単位で指定します。

最大レコードレート制限は、`BufferSize`リクエストあたりの最大レコード数を決定し、`RequestsPerSecond`。次の式を使用して、1 秒あたりの記録レート制限を計算できます。

`レコードレート=BufferSize*RequestsPerSecond`

上記の例の設定値を指定すると、最大レコードレートは 1000 レコード/秒です。

シンクセキュリティの設定

認証の設定

Kinesis Agent for Windows が AWS サービスにログ、イベント、およびメトリクスをストリーミングする場合、アクセスの認証が必要です。Windows 用 Kinesis Agent に認証を提供するには、いくつかの方法があります。認証を提供する方法は、Kinesis Agent for Windows が実行されている状況と特定の組織の特定のセキュリティ要件によって異なります。

- Kinesis Agent for Windows が Amazon EC2 ホストで実行されている場合、認証を提供する安全で簡単な方法は、必要な AWS サービスで必要なオペレーションへの十分なアクセス権限がある IAM ロールとそのロールを参照する EC2 インスタンスプロファイルを作成することです。インスタンスプロファイルの作成については、「インスタンスプロファイルの使用」を参照してください。IAM ロールにアタッチするポリシーについては、「」を参照してください。認可の設定。

  インスタンスプロファイルを作成した後、Kinesis Agent for Windows を使用するすべての EC2 インスタンスにそれを関連付けることができます。関連付けられたインスタンスプロファイルがすでにインスタンスにある場合は、そのインスタンスプロファイルに関連付けられているロールに適切なポリシーをアタッチすることができます。
- Kinesis Agent for Windows が 1 つのアカウントの EC2 ホストで実行されていても、シンクのターゲットであるリソースが別のアカウントに存在する場合、クロスアカウントアクセスで IAM ロー

ルを作成することができます。詳細については、「」を参照してください。チュートリアル: IAM ロールを使用した AWS アカウント間のアクセスの委任。クロスアカウントロールを作成した後、そのクロスアカウントロールの Amazon リソースネーム (ARN) をRoleARNシンク宣言でキーと値のペア。その後、そのシンクのシンクタイプに関連付けられている AWS リソースにアクセスしたときに Kinesis Agent for Windows によって指定したクロスアカウントロールの引き受けが試行されます。

- Windows の Kinesis Agent が Amazon EC2 の外部で実行されている場合 (たとえば、オンプレミス)、以下のいくつかのオプションがあります。
  - オンプレミスサーバーまたはデスクトップマシンを Amazon EC2 Systems Manager マネージドインスタンスとして登録できる場合は、以下のプロセスを使用して認証を設定します。
    1. 「ハイブリッド環境での AWS Systems Manager の設定」で説明されているプロセスを使用して、サービスロールを作成してマネージドインスタンスのアクティベーションを作成し、SSM エージェントをインストールします。
    2. 適切なポリシーをサービスロールにアタッチして、Kinesis Agent for Windows が設定されたシンクからデータをストリーミングするために必要なリソースにアクセスできるようにします。IAM ロールにアタッチするポリシーについては、「」を参照してください。認可の設定。
    3. で説明されているプロセスを使用します。の設定ProfileRefreshingAWSCredentialProviderAWS 認証情報の更新をクリックして AWS 認証情報を更新します。

    認証情報は SSM および AWS によって管理されるため、この方法が EC2 以外のインスタンスに推奨されます。
  - デフォルトシステムアカウントではなく特定のユーザーで Kinesis Agent for Windows の AWSKinesisTAP サービスを実行することが許容できる場合は、以下のプロセスを使用します。
    1. AWS サービスを使用する AWS アカウントで IAM ユーザーを作成します。作成プロセス時にこのユーザーのアクセスキーとシークレットキーをキャプチャします。この情報は、このプロセスの後のステップで必要です。
    2. 必要なサービスの必要なオペレーションへのアクセスを認証する IAM ユーザーにポリシーをアタッチします。IAM ユーザーにアタッチするポリシーについては、「」を参照してください。認可の設定。
    3. 各デスクトップまたはサーバーで AWSKinesisTap サービスを変更して、デフォルトシステムアカウントではなく特定のユーザーでそのサービスが実行されるようにします。
    4. 前のステップで記録したアクセスキーとシークレットキーを使用して SDK ストアでプロファイルを作成します。詳細については、「AWS 認証情報の設定」を参照してください。

5. %PROGRAMFILES%\Amazon\AWSKinesisTap ディレクトリの AWSKinesisTap.exe.config ファイルを更新して、前のステップで作成したプロファイルの名前を指定します。詳細については、「AWS 認証情報の設定」を参照してください。

   認証情報が特定のホストおよび特定のユーザーに対して暗号化されるため、この方法がマネージドインスタンスにすることができない EC2 以外のホストに推奨されます。

- デフォルトシステムアカウントで Kinesis Agent for Windows の AWSKinesisTAP サービスを実行する必要がある場合、共有認証情報ファイルを使用する必要があります。これは、SDK ストアを有効にするための Windows ユーザープロファイルがシステムアカウントにないためです。共有認証情報ファイルが暗号化されないため、この方法は推奨されません。共有構成ファイルの使用方法については、「」を参照してください。AWS 認証情報の設定()AWS SDK for .NET。この方法を使用する場合、NTFS 暗号化と、共有設定ファイルに制限されたファイルアクセスを使用することをお勧めします。キーは管理プラットフォームによって更新される必要があり、キーの更新が発生したときに共有設定ファイルを更新する必要があります。

シンク宣言でアクセスキーとシークレットを直接指定できますが、その宣言が暗号化されないため、この方法は推奨されません。

認可の設定

以下の適切なポリシーを、Kinesis Agent for Windows が AWS サービスにデータをストリーミングするために使用する IAM ユーザーまたはロールにアタッチします。

Kinesis Data Streams

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": [
                "kinesis:PutRecord",
                "kinesis:PutRecords"
            ],
            "Resource": "arn:aws:kinesis:*:*:stream/*"
        }
    ]
}
```

特定のリージョン、アカウント、またはストリーム名への認可を制限するには、ARN で該当するアスタリスクを特定の値に置き換えます。詳細については、「Amazon Kinesis Data Streams による IAM リソースに対するアクセスの制御」の「Kinesis Data Streams 用の Amazon リソースネーム (ARN)」を参照してください。

Kinesis Data Firehose

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor1",
            "Effect": "Allow",
            "Action": [
                "firehose:PutRecord",
                "firehose:PutRecordBatch"
            ],
            "Resource": "arn:aws:firehose:*:*:deliverystream/*"
        }
    ]
}
```

特定のリージョン、アカウント、または配信ストリーム名への認可を制限するには、ARN で該当するアスタリスクを特定の値に置き換えます。詳細については、「」を参照してください。Amazon Kinesis Data Firehose によるアクセスの制御()Amazon Kinesis Data Firehose 開発者ガイド。

CloudWatch

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor2",
            "Effect": "Allow",
            "Action": "cloudwatch:PutMetricData",
            "Resource": "*"
        }
    ]
```

```
}
```

詳細については、「」を参照してください。CloudWatch リソースに対するアクセス許可の管理の概要()Amazon CloudWatch Logs ユーザーガイド。

既存のロググループとログストリームを使用した CloudWatch Logs

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor3",
            "Effect": "Allow",
            "Action": [
                "logs:DescribeLogGroups",
                "logs:DescribeLogStreams",
                "logs:PutLogEvents"
            ],
            "Resource": "arn:aws:logs:*:*:log-group:*"
        },
        {
            "Sid": "VisualEditor4",
            "Effect": "Allow",
            "Action": "logs:PutLogEvents",
            "Resource": "arn:aws:logs:*:*:log-group:*:*:*"
        }
    ]
}
```

特定のリージョン、アカウント、ロググループ、またはログストリームへのアクセスを制限するには、ARN で該当するアスタリスクを適切な値に置き換えます。詳細については、「」を参照してください。CloudWatch Logs リソースに対するアクセス許可の管理の概要()Amazon CloudWatch Logs ユーザーガイド。

Kinesis Agent for Windows がロググループとログストリームを作成するための追加のアクセス許可を持つ CloudWatch Logs

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

```
            "Sid": "VisualEditor5",
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogStream",
                "logs:DescribeLogGroups",
                "logs:DescribeLogStreams",
                "logs:PutLogEvents"
            ],
            "Resource": "arn:aws:logs:*:*:log-group:*"
        },
        {
            "Sid": "VisualEditor6",
            "Effect": "Allow",
            "Action": "logs:PutLogEvents",
            "Resource": "arn:aws:logs:*:*:log-group:*:*:*"
        },
        {
            "Sid": "VisualEditor7",
            "Effect": "Allow",
            "Action": "logs:CreateLogGroup",
            "Resource": "*"
        }
    ]
}
```

特定のリージョン、アカウント、ロググループ、またはログストリームへのアクセスを制限するには、ARN で該当するアスタリスクを適切な値に置き換えます。詳細については、「」を参照してください。CloudWatch Logs リソースに対するアクセス許可の管理の概要()Amazon CloudWatch Logs ユーザーガイド。

EC2 タグ変数の展開に必要なアクセス許可

ec2tag 変数プレフィックスでの変数の展開の使用には、ec2:Describe* アクセス許可が必要です。

```
{
   "Version": "2012-10-17",
   "Statement": [{
      "Sid": "VisualEditor8",
      "Effect": "Allow",
      "Action": "ec2:Describe*",
      "Resource": "*"
```

```
    }
   ]
}
```

> **Note**
>
> 各ステートメントの Sid が 1 つのポリシー内で一意である限り、複数のステートメントをそのポリシーに結合することができます。ポリシーの作成の詳細については、「」を参照してください。IAM ポリシーの作成()IAM ユーザーガイド。

の設定ProfileRefreshingAWSCredentialProviderAWS 認証情報の更新

ハイブリッド環境に AWS Systems Manager を使用して AWS 認証情報を管理する場合、Systems Manager はc:\Windows\System32\config\systemprofile\.aws\credentials。ハイブリッド環境用のSystems Manager の詳細については、「」を参照してください。ハイブリッド環境用の AWS Systems Manager のセットアップ()AWS Systems Manager ユーザーガイド。

AWS .net SDK は新しい認証情報を自動的に取得しないため、ProfileRefreshingAWSCredentialProviderプラグインを使用して認証情報を更新します。

「」を使用できます。CredentialRef属性を使用して、すべての AWS 同期設定のCredentials定義ここで、CredentialType属性はに設定していますProfileRefreshingAWSCredentialProvider次の例に示すように、とします。

```
{
    "Sinks": [{
        "Id": "myCloudWatchLogsSink",
        "SinkType": "CloudWatchLogs",
        "CredentialRef": "ssmcred",
        "Region": "us-west-2",
        "LogGroup": "myLogGroup",
        "LogStream": "myLogStream"
    }],
    "Credentials": [{
        "Id": "ssmcred",
        "CredentialType": "ProfileRefreshingAWSCredentialProvider",
        "Profile": "default",
```

```
        "FilePath": "%USERPROFILE%//.aws//credentials",
        "RefreshingInterval": 300
    }]
}
```

認証情報の定義は、以下の属性をキーと値のペアとして構成します。

`Id`

シンク定義が使用して指定できる文字列を定義しますCredentialRefを使用して、この資格情報構成を参照します。

`CredentialType`

リテラル文字列ProfileRefreshingAWSCredentialProvider。

`Profile`

省略可能。デフォルト: default。

`FilePath`

省略可能。AWS 認証情報ファイルのパスを指定します。省略すると、デフォルトの %USERPROFILE%/.aws/credentials が使用されます。

`RefreshingInterval`

省略可能。認証情報が更新される頻度 (秒単位) 。省略すると、デフォルトの 300 が使用されます。

シンクデコレーションの設定

シンク宣言には、ソースから収集されたレコードを拡張するためにさまざまな AWS サービスにストリーミングするための追加のデータを指定するキーと値のペアをオプションで含めることができます。

`TextDecoration`

シンク宣言で Format が指定されていない場合にこのキーと値のペアを使用します。この値は、変数の置換が発生する特殊な形式の文字列です。たとえば、シンクに対して "{ComputerName}:::{timestamp:yyyy-MM-dd HH:mm:ss}:::{_record}" の TextDecoration が指定されているとします。ソースからテキスト The system has

resumed from sleep. が含まれているログレコードが送信され、そのソースがパイプを介してシンクに接続されると、テキスト MyComputer1:::2017-10-26 06:14:22:::The system has resumed from sleep. が、シンクタイプに関連付けられている AWS サービスにストリーミングされます。{_record} 変数はソースによって配信される元のテキストレコードを参照します。

ObjectDecoration

シンク宣言で Format が指定されている場合にこのキーと値のペアを使用して、レコードシリアル化の前にデータを追加します。たとえば、JSON Format を指定するシンクに対して "ComputerName={ComputerName};DT={timestamp:yyyy-MM-dd HH:mm:ss}" の ObjectDecoration が指定されているとします。その結果、シンクタイプに関連付けられている AWS サービスにストリーミングされた JSON にソースの元のデータに加え、以下のキーと値のペアが含まれます。

```
{
    ComputerName: "MyComputer2",
    DT: "2017-10-17 21:09:04"
}
```

ObjectDecoration の使用例については、「チュートリアル: Windows 用 Kinesis エージェントを使用して JSON ログファイルを Amazon S3 にストリーミング」を参照してください。

ObjectDecorationEx

より柔軟なデータ抽出と書式設定を可能にする式を指定します。ObjectDecoration。このフィールドは、シンクの形式がjson。式の構文を以下に示します。

```
"ObjectDecorationEx":
 "attribute1={expression1};attribute2={expression2};attribute3={expression3}(;...)"
```

たとえば、次のObjectDecorationEx属性:

```
"ObjectDecorationEx":
 "host={env:ComputerName};message={upper(_record)};time={format(_timestamp,
 'yyyyMMdd')}"
```

リテラルレコードを変換します。

System log message

式によって返される値を使用して、次のように JSON オブジェクトを作成します。

```
{
    "host": "EC2AMAZ-1234",
    "message": "SYSTEM LOG MESSAGE",
    "time": "20210201"
}
```

式の式の作成の詳細については、「」を参照してください。式の記述に関するヒント。ほとんどのObjectDecoration宣言は、タイムスタンプ変数を除いて、新しい構文を使用して動作するはずです。A{timestamp:yyyyMMdd}のフィールドObjectDecorationと表されます。{format(_timestamp,'yyyyMMdd')}がObjectDecorationEx。

TextDecorationEx

より柔軟なデータ抽出と書式設定を可能にする式を指定します。TextDecoration次の例に示すように、とします。

```
"TextDecorationEx": "Message '{lower(_record)}' at {format(_timestamp, 'yyyy-MM-
dd')}"
```

「」を使用できます。TextDecorationExを使用して JSON オブジェクトを作成します。次の例に示すように、開いた中括弧をエスケープするには '@{' を使用します。

```
"TextDecorationEx": "@{ \"var\": \"{upper($myvar1)}\" }"
```

シンクに接続されたソースのソースタイプが DirectorySource である場合、シンクでは次の追加の 3 つの変数を使用できます。

_FilePath

ログファイルの完全パス。

_FileName

ファイルのファイル名およびファイル名の拡張子。

_Position

レコードがログファイルのどこにあるかを表す整数。

これらの変数は、すべてのレコードを単一のストリームにストリーミングするシンクに接続された複数のファイルからログレコードを収集するソースを使用する場合に便利です。これらの変数の値をストリーミングレコードに挿入すると、データパイプラインのダウンストリーム分析でファイル別と各ファイルの場所別にレコードを並べ替えることができるようになります。

式の記述に関するヒント

式には、次のいずれかを指定できます。

- 変数の式。
- 定数式。たとえば、'hello',1,1.21,null,true,false。
- 次の例に示すように、関数を呼び出す呼び出し式。

```
regexp_extract('Info: MID 118667291 ICID 197973259 RID 0 To: <jd@acme.com>', 'To: (\\
\\S+)', 1)
```

特殊文字

特殊文字をエスケープするには、2 つのバックスラッシュが必要です。

Nesting

次の例に示すように、関数の呼び出しをネストすることができます。

```
format(date(2018, 11, 28), 'MMddyyyy')
```

Variables

変数には、ローカル、メタ、およびグローバルの 3 種類があります。

- ローカル変数で始まります$など$message。これらは、イベントオブジェクトのプロパティ、イベントがディクショナリである場合はエントリ、イベントが JSON オブジェクトの場合は属性を解決するために使用されます。ローカル変数にスペースや特殊文字が含まれている場合は、引用符で囲まれたローカル変数 ($'date created'。
- メタ変数アンダースコア (_) であり、イベントのメタデータに解決するために使用されます。すべてのイベントタイプで、次のメタ変数がサポートされます。

  _timestamp

  イベントのタイムスタンプ。

`_record`

イベントの生のテキスト表現。

ログイベントでは、次の追加のメタ変数がサポートされます。

`_filepath`

`_filename`

`_position`

`_linenumber`

- グローバル変数は、環境変数、EC2 インスタンスメタデータ、または EC2Tag に解決されます。パフォーマンスを向上させるため、プレフィクスを使用して、検索範囲を制限することをお勧めします。`{env:ComputerName}`,`{ec2:InstanceId}`, および`{ec2tag:Name}`。

組み込み関数

Windows 用 Kinesis Agent でサポートされる組み込み関数は以下のとおりです。引数のいずれかがNULLを処理するように設計されていませんNULL、NULLオブジェクトが返されます。

```
//string functions
int length(string input)
string lower(string input)
string lpad(string input, int size, string padstring)
string ltrim(string input)
string rpad(string input, int size, string padstring)
string rtrim(string input)
string substr(string input, int start)
string substr(string input, int start, int length)
string trim(string input)
string upper(string str)

//regular expression functions
string regexp_extract(string input, string pattern)
string regexp_extract(string input, string pattern, int group)

//date functions
DateTime date(int year, int month, int day)
```

```
DateTime date(int year, int month, int day, int hour, int minute, int second)
DateTime date(int year, int month, int day, int hour, int minute, int second, int
 millisecond)

//conversion functions
int? parse_int(string input)
decimal? parse_decimal(string input)
DateTime? parse_date(string input, string format)
string format(object o, string format)

//coalesce functions
object coalesce(object obj1, object obj2)
object coalesce(object obj1, object obj2, object obj3)
object coalesce(object obj1, object obj2, object obj3, object obj4)
object coalesce(object obj1, object obj2, object obj3, object obj4, object obj5)
object coalesce(object obj1, object obj2, object obj3, object obj4, object obj5, object
 obj6)
```

シンク変数の置換の設定

KinesisStream、KinesisFirehose、および CloudWatchLogs のシンク宣言では、LogStream または StreamName のキーと値のペアが必要です。これらのキー値の値には、Windows の Kinesis Agent によって自動的に解決される変数参照を含めることができます。を使用する場合CloudWatchLogsとすると、LogGroupキーと値のペアも必要です。このキーと値のペアには、Kinesis Agent for Windows によって自動的に解決される変数参照を含めることができます。変数は、テンプレート {*prefix*:*variablename*} を使用して指定します。ここで、*prefix*: はオプションです。サポートされているプレフィックスは次のとおりです。

- env— 変数参照は同じ名前の環境変数の値に解決されます。
- ec2— 変数参照は同じ名前の EC2 インスタンスメタデータに解決されます。
- ec2tag— 変数参照は同じ名前の EC2 インスタンスタグの値に解決されます。インスタンスタグにアクセスするには、ec2:Describe* アクセス許可が必要です。詳細については、「EC2 タグ変数の展開に必要なアクセス許可」を参照してください。

プレフィックスが指定されていない場合に variablename と同じ名前の環境変数があると、変数参照は環境変数の値に解決されます。それ以外の場合で、variablename が instance_id または hostname である場合、変数参照は同じ名前の EC2 メタデータの値に解決されます。それ以外の場合、変数参照は解決されません。

変数参照を使用した有効なキーと値のペアの例を次に示します。

```
"LogStream": "LogStream_{instance_id}"
"LogStream": "LogStream_{hostname}"
"LogStream": "LogStream_{ec2:local-hostname}"
"LogStream": "LogStream_{computername}"
"LogStream": "LogStream_{env:computername}"
```

CloudWatchLogs シンク宣言では、特殊な形式のタイムスタンプ変数をサポートしています。この変数により、ソースからの元のログレコードまたはイベントレコードのタイムスタンプでログストリームの名前の変更が可能になります。形式は次のとおりです。{timestamp:timeformat}次の例を参照してください。

```
"LogStream": "LogStream_{timestamp:yyyyMMdd}"
```

ログレコードまたはイベントレコードが 2017 年 6 月 5 日に生成された場合、前の例の LogStream のキーと値のペアの値は "LogStream_20170605" に解決されます。

承認されている場合、CloudWatchLogs シンクタイプは、必要に応じて生成された名前に基づいて新しいログストリームを自動的に作成できます。その他のシンクタイプでは、ストリームの名前以外の追加の設定が必要であるため、これを行うことはできません。

テキストおよびオブジェクトのデコレーションで発生する特殊な変数の置換があります。詳細については、「シンクデコレーションの設定」を参照してください。

シンクのキューイングの設定

KinesisStream、KinesisFirehose、および CloudWatchLogs のシンク宣言では、一時的な接続の問題が原因でそれらのシンクタイプに関連付けられている AWS サービスにストリーミングできなかったレコードのキューイングをオプションで有効にすることができます。接続が回復されたときにキューイングと自動ストリーミングの再試行を有効にするには、シンク宣言で次のキーと値のペアを使用します。

QueueType

使用するキューイングメカニズムの種類を指定します。サポートされている値は file だけです。これはレコードがファイルにキューイングされることを示します。Windows 用 Kinesis

Agent for Windows のキューイング機能を有効にするには、このキーと値のペアが必要です。指定しない場合、デフォルトでは、メモリのみにキューイングして、メモリ内のキューイングの制限に達するとストリーミングに失敗します。

`QueuePath`

キュー内のレコードのファイルが格納されているフォルダのパスを指定します。このキーと値のペアはオプションです。デフォルト値は `%PROGRAMDATA%\KinesisTap\Queue\SinkId` です。ここで SinkId はシンク宣言で `Id` の値として割り当てた識別子です。

`QueueMaxBatches`

ストリーミング用にレコードをキューイングするときに Kinesis Agent for Windows で消費可能な容量の合計量を制限します。容量は、このキーと値のペアにバッチあたりの最大バイト数を乗算した値に制限されます。`KinesisStream`、`KinesisFirehose`、および `CloudWatchLogs` のシンクタイプのバッチあたりの最大バイト数はそれぞれ、5 MB、4 MB、および 1 MB です。この制限に達すると、ストリーミング障害はキューに入れられず、回復不可能な障害としてレポートされます。このキーと値のペアはオプションです。デフォルト値は 10,000 個のバッチです。

シンクのプロキシの設定

AWS サービスにアクセスするすべての Kinesis Agent for Windows シンクタイプでプロキシを設定するには、以下の場所にある Kinesis Agent for Windows 設定ファイルを編集します。`%Program Files%\Amazon\KinesisTap\AWSKinesisTap.exe.config`。手順については、以下を参照してください。`proxy`のセクションに追加します。AWS SDK for .NET の設定ファイルリファレンス().NET 用 AWS SDK 開発者ガイド。

より多くのシンク属性での変数の解決の設定

以下の例では、使用するシンクの設定を示しています。`Region`の値について、環境変数`Region`属性キーと値のペア。を使用する場合`RoleARN`では、EC2 タグキーを指定します。`MyRoleARN`そのキーに関連付けられた値に評価されます。

```
"Id": "myCloudWatchLogsSink",
"SinkType": "CloudWatchLogs",
"LogGroup": "EC2Logs",
"LogStream": "logs-{instance_id}"
"Region": "{env:Region}"
"RoleARN": "{ec2tag:MyRoleARN}"
```

AWS シンクで RoleARN プロパティを使用する場合の AWS STS リージョンエンドポイントの設定

この機能は、Amazon EC2 で KinesiStap を使用し、RoleARNプロパティを使用して、送信先の AWS サービスで認証する外部 IAM ロールを引き受けることができます。

設定によるUseSTSRegionalEndpoints~trueでは、エージェントがリージョンのエンドポイントを使用するように指定できます (たとえば、https://sts.us-east-1.amazonaws.com) ではなく、グローバルエンドポイント (たとえば、https://sts.amazonaws.com). Regional STS エンドポイントを使用すると、オペレーションのラウンドトリップ待ち時間が短縮され、グローバルエンドポイントサービスの障害の影響が制限されます。

AWS シンク用の VPC エンドポイントの設定

シンク設定で VPC エンドポイントをCloudWatchLogs,CloudWatch,KinesisStreams, およびKinesisFirehoseシンクタイプ。VPC エンドポイントでは、AWS PrivateLink を使用する AWS のサービスや VPC エンドポイントサービスに VPC をプライベートに接続できます。インターネットゲートウェイ、NAT デバイス、VPN 接続、または AWS Direct Connect 接続は必要ありません。VPC のインスタンスは、サービスのリソースと通信するためにパブリック IP アドレスを必要としません。VPC と他のサービス間のトラフィックは、Amazon ネットワークを離れません。詳細については、「」を参照してください。VPC エンドポイント()Amazon VPC ユーザーガイド。

VPC エンドポイントを指定するには、ServiceURL次の例に示すように、プロパティCloudWatchLogsシンク構成。の値を設定します。ServiceURLに表示される値にVPC エンドポイントの詳細Amazon VPC コンソールを使用して、タブに追加します。

```
{
    "Id": "myCloudWatchLogsSink",
    "SinkType": "CloudWatchLogs",
    "LogGroup": "EC2Logs",
    "LogStream": "logs-{instance_id}",
    "ServiceURL":"https://vpce-ab1c234de56-ab7cdefg.logs.us-east-1.vpce.amazonaws.com"
}
```

プロキシの代替手段の設定

この機能を使用すると、.NET ではなく AWS SDK に組み込まれているプロキシサポートを使用して、シンク設定でプロキシサーバーを設定できます。以前は、プロキシを使用するようにエージェン

トを設定する唯一の方法は、.NET のネイティブ機能を使用することでした。これにより、プロキシファイルで定義されたプロキシを介してすべての HTTP/S 要求が自動的にルーティングされます。

現在プロキシ・サーバでエージェントを使用している場合は、この方法を使用するために変更する必要はありません。

「」を使用できます。ProxyHostおよびProxyPortプロパティを使用して、次の例のように代替プロキシを設定します。

```
{
    "Id": "myCloudWatchLogsSink",
    "SinkType": "CloudWatchLogs",
    "LogGroup": "EC2Logs",
    "LogStream": "logs-{instance_id}",
    "Region": "us-west-2",
    "ProxyHost": "myproxy.mydnsdomain.com",
    "ProxyPort": "8080"
}
```

パイプ宣言

を使用するパイプ宣言を使用してソースを接続します (ソース宣言を参照) をシンク (シンク宣言Microsoft Windows の Amazon Kinesis エージェントの。パイプ宣言は、JSON オブジェクトとして表されます。Kinesis Agent for Windows が起動した後、ログ、イベント、またはメトリクスは特定のパイプのソースから収集されます。その後、それらはそのパイプに関連付けられているシンクを使用してさまざまな AWS のサービスにストリーミングされます。

以下は パイプ宣言の例です。

```
{
   "Id": "MyAppLogToCloudWatchLogs",
   "SourceRef": "MyAppLog",
   "SinkRef": "MyCloudWatchLogsSink"
}
```

トピック

- パイプの設定
- Windows メトリックパイプ用の Kinesis エージェントの設定

パイプの設定

すべてのパイプ宣言には、次のキーと値のペアを含めることができます。

`Id`

パイプの名前を指定します (必須)。これは、設定ファイル内で一意である必要があります。

`Type`

ログデータがソースからシンクに転送されるときにパイプによって適用される変換のタイプ (ある場合) を指定します。`RegexFilterPipe` はサポートされる唯一の値です。この値は、ログレコードの基になるテキスト表現の正規表現フィルタリングを有効にします。フィルタリングを使用すると、関連するログレコードのみをデータパイプラインの下流に送信することで、送信とストレージのコストを削減できます。このキーと値のペアはオプションです。デフォルト値では、変換は行われません。

`FilterPattern`

シンクに転送される前にソースによって収集されたログレコードをフィルタリングするために使用される `RegexFilterPipe` パイプラインの正規表現を指定します。正規表現がレコードの元のテキスト表現と一致すると、ログレコードは `RegexFilterPipe` タイプのパイプで転送されます。たとえば、`DirectorySource` 宣言で `ExtractionPattern` キーと値のペアを使用するときに生成される構造化ログレコードは、`RegexFilterPipe` メカニズムを使用してフィルタリングできます。これは、このメカニズムが解析前の元のテキスト表現に対して機能するためです。このキーと値のペアはオプションですが、パイプが `RegexFilterPipe` タイプを指定している場合は指定する必要があります。

以下は `RegexFilterPipe` パイプ宣言の例です。

```
{
 "Id": "MyAppLog2ToFirehose",
 "Type": "RegexFilterPipe",
 "SourceRef": "MyAppLog2",
 "SinkRef": "MyFirehose",
 "FilterPattern": "^(10|11),.*",
 "IgnoreCase": false,
 "Negate": false
}
```

`SourceRef`

パイプのログ、イベント、およびメトリクスデータを収集するソースを定義するソース宣言の名前 (`Id` キーと値のペアの値) を指定します (必須)。

`SinkRef`

パイプのログ、イベント、およびメトリクスデータを受け取るシンクを定義するシンク宣言の名前 (`Id` キーと値のペアの値) を指定します (必須)。

`IgnoreCase`

省略可能。次の値を受け入れます`true`または`false`。に設定した場合`true`の場合、Regexは大文字と小文字を区別しない方法でレコードに一致します。

`Negate`

省略可能。次の値を受け入れます`true`または`false`。に設定した場合`true`の場合、パイプはレコードをしない正規表現に一致します。

`RegexFilterPipe` パイプタイプを使用した完全な設定ファイルの例については、「パイプの使用」を参照してください。

Windows メトリックパイプ用の Kinesis エージェントの設定

という名前の組み込みメトリクスソースがあります`_KinesisTapMetricsSource`で、Windows 用 Kinesis エージェントに関するメトリックスを生成します。ある場合`CloudWatch`シンク宣言を`Id`の`MyCloudWatchSink`次のパイプライン宣言は Windows が生成したメトリクスをそのシンクに転送します。

```
{
   "Id": "KinesisAgentMetricsToCloudWatch",
   "SourceRef": "_KinesisTapMetricsSource",
   "SinkRef": "MyCloudWatchSink"
}
```

Kinesis エージェント for Windows 組み込みメトリクスソースの詳細については、」Windows 組み込みメトリクスソース Kinesis エージェント。

設定ファイルが Windows パフォーマンスカウンターメトリクスもストリーミングする場合は、Kinesis Agent for Windows メトリクスと Windows パフォーマンスカウンターメトリックの両方に同じシンクを使用するのではなく、別々のパイプとシンクを使用することをお勧めします。

自動更新の設定

の使用appsettings.json設定ファイルを使用して Microsoft Windows 用 Amazon Kinesis エージェントおよび Windows 用 Kinesis エージェントの設定ファイルに自動更新を有効化します。更新動作を制御するには、設定ファイルで Sources、Sinks、および Pipes と同じレベルで Plugins のキーと値のペアを指定します。

Plugins のキーと値のペアにより、特にソース、シンク、およびパイプのカテゴリに当てはならない、使用する追加の一般的な機能が指定されます。たとえば、Windows 用の Kinesis エージェントを更新するためのプラグインやappsettings.json設定ファイルを作成します。プラグインは、JSON オブジェクトとして表され、常に Type のキーと値のペアがあります。Type により、プラグインに指定できるその他のキーと値のペアが決定されます。以下のプラグインタイプがサポートされています。

PackageUpdate

Windows 用 Kinesis エージェントがパッケージバージョン設定ファイルを定期的にチェックすることを指定します。パッケージバージョンファイルが、異なるバージョンの Kinesis Agent for Windows をインストールする必要があることを示している場合、Kinesis Agent for Windows はそのバージョンをダウンロードしてインストールします。PackageUpdate プラグインのキーと値のペアは次のとおりです。

Type

値は PackageUpdate という文字列である必要があり、必須です。

Interval

文字列として表される分単位の変更についてパッケージバージョンファイルをチェックする頻度を指定します。このキーと値のペアはオプションです。指定がなければ、デフォルト値は 60 分です。値が 1 より小さい場合、更新のチェックは行われません。

PackageVersion

パッケージバージョンの JSON ファイルの場所を指定します。ファイルはファイル共有 (file://)、ウェブサイト (http://)、または Amazon S3 (s3://). たとえば、値s3://mycompany/config/agent-package-version.jsonは、Windows 用 Kinesis エージェントがconfig/agent-package-version.jsonファイルでmycompanyAmazon S3 バケット。そのファイルの内容に基づいて更新を実行する必要があります。

> **Note**
>
> の値。PackageVersionキーと値のペアは、Amazon S3 では大文字と小文字が区別されます。

以下は、パッケージバージョンファイルの内容の例です。

```
{
    "Name": "AWSKinesisTap",
    "Version": "1.0.0.106",
    "PackageUrl": "https://s3-us-west-2.amazonaws.com/kinesis-agent-windows/
downloads/AWSKinesisTap.{Version}.nupkg"
}
```

-Versionキーと値のペアは、Kinesis Agent for Windows がまだインストールされていない場合にどのバージョンをインストールするかを指定します。PackageUrl の {Version} 変数参照は、Version キーと値のペアに指定した値を解決します。この例では、変数は文字列 1.0.0.106 に解決されます。この変数解決は、パッケージバージョンファイル内の特定の目的のバージョンが格納されている場所が 1 つになるように提供されています。複数のパッケージバージョンファイルを使用して、大規模なデプロイの前に新しいバージョンを検証するために Kinesis Agent for Windows の新しいバージョンを導入するペースを制御できます。Windows の Kinesis エージェントデプロイをロールバックするには、1 つ以上のパッケージバージョンファイルを変更して、環境で機能することがわかっている以前のバージョンの Kinesis Agent for Windows を指定します。

PackageVersion キーと値のペアの値は、異なるパッケージバージョンファイルの自動選択を容易にするための変数の置換の影響を受けます。変数の置換の詳細については、「シンク変数の置換の設定」を参照してください。

AccessKey

Amazon S3 のパッケージバージョンファイルへのアクセスを認証するときに使用するアクセスキーを指定します。このキーと値のペアはオプションです。このキーと値のペアを使用することはお勧めしません。推奨される別の認証方法については、「認証の設定」を参照してください。

SecretKey

Amazon S3 のパッケージバージョンファイルへのアクセスを認証するときに使用するシークレットキーを指定します。このキーと値のペアはオプションです。このキーと値のペアを使用

することはお勧めしません。推奨される別の認証方法については、「認証の設定」を参照してください。

`Region`

Amazon S3 からパッケージバージョンファイルにアクセスするときに使用するリージョンエンドポイントを指定します。このキーと値のペアはオプションです。

`ProfileName`

Amazon S3 のパッケージバージョンファイルへのアクセスを認証するときに使用するセキュリティプロファイルを指定します。詳細については、「認証の設定」を参照してください。このキーと値のペアはオプションです。

`RoleARN`

クロスアカウントのシナリオで Amazon S3 のパッケージバージョンファイルへのアクセスを認証するときに引き受けるロールを指定します。詳細については、「認証の設定」を参照してください。このキーと値のペアはオプションです。

`PackageUpdate` プラグインが指定されていない場合、アップデートが必要かどうかを判断するためにパッケージバージョンファイルはチェックされません。

`ConfigUpdate`

Windows 用 Kinesis エージェントが更新された`appsettings.json`設定ファイルが、ファイル共有、ウェブサイト、または Amazon S3 に保存されています。更新された構成ファイルが存在する場合は、Kinesis Agent for Windows によってダウンロードされインストールされます。`ConfigUpdate`キーと値のペアは次のとおりです。

`Type`

値は `ConfigUpdate` という文字列である必要があり、必須です。

`Interval`

文字列として表される新しい設定ファイルをチェックする頻度を分単位で指定します。このキーと値のペアはオプションです。指定されていない場合、デフォルトは 5 分です。値が 1 より小さい場合は、設定ファイルの更新は確認されません。

`Source`

更新された設定ファイルを検索する場所を指定します。ファイルはファイル共有 (`file://`)、ウェブサイト (`http://`)、または Amazon S3 (`s3://`). たとえば、値`s3://mycompany/config/appsettings.json`は、Windows 用 Kinesis エージェントが`config/appsettings.json`ファイルで`mycompany`Amazon S3 バケット。

> **Note**
>
> の値。SourceAmazon S3 では、キーと値のペアで大文字と小文字が区別されます。

Source キーと値のペアの値は、異なる設定ファイルの自動選択を容易にするための変数の置換の影響を受けます。変数の置換の詳細については、「シンク変数の置換の設定」を参照してください。

`Destination`

ローカルマシンの設定ファイルの保存場所を指定します。これは、相対パス、絶対パス、または %PROGRAMDATA% などの環境変数参照を含むパスです。パスが相対パスの場合は、Kinesis Agent for Windows がインストールされている場所に対する相対パスです。通常、値は .\appsettings.json である必要があります。このキーと値のペアは必須です。

`AccessKey`

Amazon S3 の設定ファイルへのアクセスを認証するときに使用するアクセスキーを指定します。このキーと値のペアはオプションです。このキーと値のペアを使用することはお勧めしません。推奨される別の認証方法については、「認証の設定」を参照してください。

`SecretKey`

Amazon S3 の設定ファイルへのアクセスを認証するときに使用するシークレットキーを指定します。このキーと値のペアはオプションです。このキーと値のペアを使用することはお勧めしません。推奨される別の認証方法については、「認証の設定」を参照してください。

`Region`

Amazon S3 から設定ファイルにアクセスするときに使用するリージョンエンドポイントを指定します。このキーと値のペアはオプションです。

`ProfileName`

Amazon S3 の設定ファイルへのアクセスを認証するときに使用するセキュリティプロファイルを指定します。詳細については、「認証の設定」を参照してください。このキーと値のペアはオプションです。

`RoleARN`

クロスアカウントのシナリオで Amazon S3 の設定ファイルへのアクセスを認証するときに引き受けるロールを指定します。詳細については、「認証の設定」を参照してください。このキーと値のペアはオプションです。

ConfigUpdate プラグインが指定されていない場合、設定ファイルの更新が必要かどうかを判断するために設定ファイルはチェックされません。

以下は、PackageUpdate および ConfigUpdate プラグインの使用方法を示す appsettings.json 設定ファイルの例です。この例では、パッケージバージョンファイルがmycompanyAmazon S3 バケットという名前のconfig/agent-package-version.json。このファイルは、約 2 時間ごとに変更がないかチェックされます。パッケージバージョンファイルで Kinesis Agent for Windows の異なるバージョンが指定されている場合、指定されたエージェントバージョンはパッケージバージョンファイル内の指定された場所からインストールされます。

また、appsettings.json設定ファイルがmycompanyAmazon S3 バケットという名前のconfig/appsettings.json。約 30 分ごとに、そのファイルは現在の設定ファイルと比較されます。異なる場合は、更新された構成ファイルが Amazon S3 からダウンロードされ、appsettings.json設定ファイルを作成します。

```
{
  "Sources": [
    {
      "Id": "ApplicationLogSource",
      "SourceType": "DirectorySource",
      "Directory": "C:\\LogSource\\",
      "FileNameFilter": "*.log",
      "RecordParser": "SingleLine"
    }
  ],
  "Sinks": [
    {
       "Id": "ApplicationLogKinesisFirehoseSink",
       "SinkType": "KinesisFirehose",
       "StreamName": "ApplicationLogFirehoseDeliveryStream",
       "Region": "us-east-1"
    }
    ],
  "Pipes": [
    {
      "Id": "ApplicationLogSourceToApplicationLogKinesisFirehoseSink",
      "SourceRef": "ApplicationLogSource",
      "SinkRef": "ApplicationLogKinesisFirehoseSink"
    }
  ],
  "Plugins": [
```

```
    {
      "Type": "PackageUpdate"
      "Interval": "120",
      "PackageVersion": "s3://mycompany/config/agent-package-version.json"
    },
    {
      "Type": "ConfigUpdate",
      "Interval": "30",
      "Source": "s3://mycompany/config/appsettings.json",
      "Destination": ".\appSettings.json"
    }
  ]
}
```

Windows 用 Kinesis エージェントの設定例

-appsettings.json設定ファイルは、Amazon Kinesis Agent for Microsoft Windows がログ、イベント、およびメトリクスを収集する方法を制御する JSON ドキュメントです。また、Windows 用 Kinesis Agent for Windows がそのデータを変換してさまざまな AWS サービスにストリーミングする方法も制御します。設定ファイルのソース、シンク、およびパイプの宣言の詳細については、「ソース宣言」、「シンク宣言」、および「パイプ宣言」を参照してください。

以下のセクションには、いくつかの異なるシナリオの設定ファイルの例が含まれています。

トピック

- さまざまなソースから Kinesis Data Streams へのストリーミング
- Windows アプリケーションイベントログからシンクへのストリーミング
- パイプの使用
- 複数のソースとパイプを使用する

さまざまなソースから Kinesis Data Streams へのストリーミング

以下の例appsettings.json設定ファイルは、さまざまなソースから Kinesis Data Streams、および Windows パフォーマンスカウンターから Amazon CloudWatch メトリクスへの口グとイベントのストリーミングを示しています。

DirectorySource、SysLog レコードパーサー

次のファイルは、syslog 形式のログレコードをストリーミングします。.logファイル拡張子C:\LogSource\ディレクトリに移動します。SyslogKinesisDataStreamus-east-1 リージョンの Kinesis データストリームのストリーム。エージェントがシャットダウンされ、後で再起動された場合でも、ログファイルのすべてのデータが送信されるように、ブックマークが設定されています。カスタムアプリケーションは SyslogKinesisDataStream ストリームからレコードを読み込んで処理できます。

```
{
  "Sources": [
    {
      "Id": "SyslogDirectorySource",
      "SourceType": "DirectorySource",
      "Directory": "C:\\LogSource\\",
      "FileNameFilter": "*.log",
      "RecordParser": "SysLog",
      "TimeZoneKind": "UTC",
      "InitialPosition": "Bookmark"
    }
  ],
  "Sinks": [
    {
      "Id": "KinesisStreamSink",
      "SinkType": "KinesisStream",
      "StreamName": "SyslogKinesisDataStream",
      "Region": "us-east-1"
    }
  ],
  "Pipes": [
    {
      "Id": "SyslogDS2KSSink",
      "SourceRef": "SyslogDirectorySource",
      "SinkRef": "KinesisStreamSink"
    }
  ]
}
```

DirectorySource、SingleLineJson レコードパーサー

次のファイルは、JSON 形式のログレコードをストリーミングします。.logファイル拡張子C:\LogSource\ディレクトリに移動します。JsonKinesisDataStreamus-east-1 リージョンの

Kinesis データストリームのストリーム。ストリーミングの前に、ComputerName キーと DT キーのキーと値のペアが、コンピュータ名とレコードが処理された日時の値を含む各 JSON オブジェクトに追加されます。カスタムアプリケーションは JsonKinesisDataStream ストリームからレコードを読み込んで処理できます。

```
{
  "Sources": [
    {
      "Id": "JsonLogSource",
      "SourceType": "DirectorySource",
      "RecordParser": "SingleLineJson",
      "Directory": "C:\\LogSource\\",
      "FileNameFilter": "*.log",
      "InitialPosition": 0
    }
  ],
  "Sinks": [
    {
      "Id": "KinesisStreamSink",
      "SinkType": "KinesisStream",
      "StreamName": "JsonKinesisDataStream",
      "Region": "us-east-1",
      "Format": "json",
      "ObjectDecoration": "ComputerName={ComputerName};DT={timestamp:yyyy-MM-dd
 HH:mm:ss}"
    }
  ],
  "Pipes": [
    {
      "Id": "JsonLogSourceToKinesisStreamSink",
      "SourceRef": "JsonLogSource",
      "SinkRef": "KinesisStreamSink"
    }
  ]
}
```

ExchangeLogSource

次のファイルは、Microsoft Exchange によって生成され、.log拡張子のC:\temp\ExchangeLog\ディレクトリに移動します。ExchangeKinesisDataStreamJSON 形式の us-east-1 リージョンの Kinesis データストリーム。Exchange ログは JSON 形式ではありませんが、Windows 用 Kinesis Agent はログを解析して JSON に変換できます。ストリーミングの前に、ComputerName キーと

DT キーのキーと値のペアが、コンピュータ名とレコードが処理された日時の値を含む各 JSON オブジェクトに追加されます。カスタムアプリケーションは ExchangeKinesisDataStream ストリームからレコードを読み込んで処理できます。

```
{
  "Sources": [
    {
       "Id": "ExchangeSource",
       "SourceType": "ExchangeLogSource",
       "Directory": "C:\\temp\\ExchangeLog\",
       "FileNameFilter": "*.log"
    }
  ],
  "Sinks": [
    {
      "Id": "KinesisStreamSink",
      "SinkType": "KinesisStream",
      "StreamName": "ExchangeKinesisDataStream",
      "Region": "us-east-1",
      "Format": "json",
      "ObjectDecoration": "ComputerName={ComputerName};DT={timestamp:yyyy-MM-dd
 HH:mm:ss}"
    }
  ],
  "Pipes": [
    {
      "Id": "ExchangeSourceToKinesisStreamSink",
      "SourceRef": "ExchangeSource",
      "SinkRef": "KinesisStreamSink"
    }
  ]
}
```

W3SVCLogSource

次のファイルは、Windows 用インターネットインフォメーションサービス (IIS) ログレコードをIISKinesisDataStreamus-east-1 リージョンの Kinesis データストリームのストリーム。カスタムアプリケーションは IISKinesisDataStream ストリームからレコードを読み込んで処理できます。IIS は Windows 用 のウェブサーバーです。

```
{
  "Sources": [
    {
       "Id": "IISLogSource",
       "SourceType": "W3SVCLogSource",
       "Directory": "C:\\inetpub\\logs\\LogFiles\\W3SVC1",
       "FileNameFilter": "*.log"
    }
  ],
  "Sinks": [
    {
      "Id": "KinesisStreamSink",
      "SinkType": "KinesisStream",
      "StreamName": "IISKinesisDataStream",
      "Region": "us-east-1"
    }
  ],
  "Pipes": [
    {
      "Id": "IISLogSourceToKinesisStreamSink",
      "SourceRef": "IISLogSource",
      "SinkRef": "KinesisStreamSink"
    }
  ]
}
```

クエリを含む **WindowsEventLogSource**

次のファイルは、Windows システムイベントログからログイベントをストリームします。`Critical`または`Error`(2 以下) を`SystemKinesisDataStream`JSON 形式の us-east-1 リージョンの Kinesis データストリーム。カスタムアプリケーションは `SystemKinesisDataStream` ストリームからレコードを読み込んで処理できます。

```
{
  "Sources": [
    {
         "Id": "SystemLogSource",
         "SourceType": "WindowsEventLogSource",
         "LogName": "System",
         "Query": "*[System/Level<=2]"
    }
  ],
```

```
  "Sinks": [
    {
      "Id": "KinesisStreamSink",
      "SinkType": "KinesisStream",
      "StreamName": "SystemKinesisDataStream",
      "Region": "us-east-1",
      "Format": "json"
    }
  ],
  "Pipes": [
    {
      "Id": "SLSourceToKSSink",
      "SourceRef": "SystemLogSource",
      "SinkRef": "KinesisStreamSink"
    }
  ]
}
```

WindowsETWEventSource

次のファイルは、Microsoft 共通言語ランタイム (CLR) の例外およびセキュリティイベントをClrKinesisDataStreamJSON 形式の us-east-1 リージョンの Kinesis データストリーム。カスタムアプリケーションは ClrKinesisDataStream ストリームからレコードを読み込んで処理できます。

```
{
  "Sources": [
    {
       "Id": "ClrETWEventSource",
       "SourceType": "WindowsETWEventSource",
       "ProviderName": "Microsoft-Windows-DotNETRuntime",
       "TraceLevel": "Verbose",
       "MatchAnyKeyword": "0x00008000, 0x00000400"
    }
  ],
  "Sinks": [
    {
      "Id": "KinesisStreamSink",
      "SinkType": "KinesisStream",
      "StreamName": "ClrKinesisDataStream",
      "Region": "us-east-1",
      "Format": "json"
    }
```

```
  ],
  "Pipes": [
    {
      "Id": "ETWSourceToKSSink",
      "SourceRef": "ClrETWEventSource",
      "SinkRef": "KinesisStreamSink"
    }
  ]
}
```

WindowsPerformanceCounterSource

次のファイルは、開いているファイルの合計、再起動以降の総ログイン試行回数、1 秒あたりのディスク読み取り数、および空きディスク領域の割合のパフォーマンスカウンターを us-east-1 リージョンの CloudWatch メトリクスにストリーミングします。これらのメトリクスは、CloudWatch でグラフ化したり、グラフからダッシュボードを構築したり、しきい値を超えたときに通知を送信するアラームを設定したりできます。

```
{
  "Sources": [
    {
      "Id": "PerformanceCounter",
      "SourceType": "WindowsPerformanceCounterSource",
      "Categories": [
        {
          "Category": "Server",
          "Counters": [
            "Files Open",
            "Logon Total"
          ]
        },
        {
          "Category": "LogicalDisk",
          "Instances": "*",
          "Counters": [
            "% Free Space",
            {
              "Counter": "Disk Reads/sec",
              "Unit": "Count/Second"
            }
          ]
        }
      ],
```

```
    }
  ],
  "Sinks": [
    {
      "Namespace": "MyServiceMetrics",
      "Region": "us-east-1",
      "Id": "CloudWatchSink",
      "SinkType": "CloudWatch"
    }
  ],
  "Pipes": [
    {
      "Id": "PerformanceCounterToCloudWatch",
      "SourceRef": "PerformanceCounter",
      "SinkRef": "CloudWatchSink"
    }
  ]
}
```

Windows アプリケーションイベントログからシンクへのストリーミング

以下の例`appsettings.json`設定ファイルは、Windows アプリケーションイベントログを Microsoft Windows 用 Amazon Kinesis Agent のさまざまなシンクにストリーミングする方法を示しています。`KinesisStream` および `CloudWatch` シンクタイプの使用例については、「さまざまなソースから Kinesis Data Streams へのストリーミング」を参照してください。

`KinesisFirehose`

次のファイルストリーム`Critical`または`Error`Windows アプリケーションは、イベントを`WindowsLogFirehoseDeliveryStream`Kinesis Data Firehose 配信ストリームは、us-east-1 リージョンの Kinesis Data Firehose への接続が中断されると、イベントはまずメモリにキューに入れられます。その後、必要に応じて、接続が回復するまでディスク上のファイルに入れられます。その後、イベントはキューから取り出されて送信され、その後に新しいイベントが続きます。

データパイプラインの要件に基づいて、ストリーミングデータをさまざまな種類のストレージおよび分析サービスに格納するように Kinesis Data Firehose を設定できます。

```
{
  "Sources": [
    {
         "Id": "ApplicationLogSource",
```

```
          "SourceType": "WindowsEventLogSource",
          "LogName": "Application",
          "Query": "*[System/Level<=2]"
      }
  ],
  "Sinks": [
    {
       "Id": "WindowsLogKinesisFirehoseSink",
       "SinkType": "KinesisFirehose",
       "StreamName": "WindowsLogFirehoseDeliveryStream",
       "Region": "us-east-1",
       "QueueType": "file"
    }
    ],
  "Pipes": [
    {
      "Id": "ALSource2ALKFSink",
      "SourceRef": "ApplicationLogSource",
      "SinkRef": "WindowsLogKinesisFirehoseSink"
    }
  ]
}
```

CloudWatchLogs

次のファイルストリームCriticalまたはErrorWindows アプリケーションログイベントを CloudWatch にストリーミングします。MyServiceApplicationLog-Groupロググループに移動します。各ストリームの名前は Stream- で始まります。それは、ストリームが作成された 4 桁の年、2 桁の月、および 2 桁の日で終了します (たとえば、Stream-20180501 は、2018 年 5 月 1 日に作成されたストリームです)。

```
{
  "Sources": [
    {
          "Id": "ApplicationLogSource",
          "SourceType": "WindowsEventLogSource",
          "LogName": "Application",
          "Query": "*[System/Level<=2]"
    }
  ],
  "Sinks": [
    {
```

```
      "Id": "CloudWatchLogsSink",
      "SinkType": "CloudWatchLogs",
      "LogGroup": "MyServiceApplicationLog-Group",
      "LogStream": "Stream-{timestamp:yyyyMMdd}",
      "Region": "us-east-1",
      "Format": "json"
    }
  ],
  "Pipes": [
    {
      "Id": "ALSource2CWLSink",
      "SourceRef": "ApplicationLogSource",
      "SinkRef": "CloudWatchLogsSink"
    }
  ]
}
```

パイプの使用

次の appsettings.json 設定ファイルの例は、パイプ関連の機能の使い方を示しています。

この例では、ログエントリをc:\LogSource
\をApplicationLogFirehoseDeliveryStreamKinesis Data Firehose 配信ストリーム。FilterPattern のキーと値のペアで指定された正規表現に一致する行のみが含まれます。具体的には、ログファイル内の行だけが10または11Kinesis Data Firehose にストリーミングされます。

```
{
  "Sources": [
    {
      "Id": "ApplicationLogSource",
      "SourceType": "DirectorySource",
      "Directory": "C:\\LogSource\\",
      "FileNameFilter": "*.log",
      "RecordParser": "SingleLine"
    }
  ],
  "Sinks": [
    {
       "Id": "ApplicationLogKinesisFirehoseSink",
       "SinkType": "KinesisFirehose",
       "StreamName": "ApplicationLogFirehoseDeliveryStream",
```

```
        "Region": "us-east-1"
    }
    ],
  "Pipes": [
    {
      "Id": "ALSourceToALKFSink",
      "Type": "RegexFilterPipe",
      "SourceRef": "ApplicationLogSource",
      "SinkRef": "ApplicationLogKinesisFirehoseSink",
      "FilterPattern": "^(10|11),.*"
    }
  ]
}
```

複数のソースとパイプを使用する

次の例の appsettings.json 設定ファイルは、複数のソースとパイプを使用する方法を示しています。

この例では、アプリケーション、セキュリティ、およびシステムの Windows イベントログをEventLogStream3 つのソース、3 つのパイプ、および 1 つのシンクを使用した Kinesis Data Firehose 配信ストリーム。

```
{
    "Sources": [
  {
    "Id": "ApplicationLog",
    "SourceType": "WindowsEventLogSource",
    "LogName": "Application"
  },
  {
    "Id": "SecurityLog",
    "SourceType": "WindowsEventLogSource",
    "LogName": "Security"
  },
  {
    "Id": "SystemLog",
    "SourceType": "WindowsEventLogSource",
    "LogName": "System"
  }
    ],
    "Sinks": [
```

```
  {
    "Id": "EventLogSink",
    "SinkType": "KinesisFirehose",
    "StreamName": "EventLogStream",
    "Format": "json"
  },
    ],
    "Pipes": [
  {
    "Id": "ApplicationLogToFirehose",
    "SourceRef": "ApplicationLog",
    "SinkRef": "EventLogSink"
  },
  {
    "Id": "SecurityLogToFirehose",
    "SourceRef": "SecurityLog",
    "SinkRef": "EventLogSink"
  },
  {
    "Id": "SystemLogToFirehose",
    "SourceRef": "SystemLog",
    "SinkRef": "EventLogSink"
  }
    ]
}
```

テレメトリクスの設定

より良いサポートを可能にするために、Amazon Kinesis Agent for Microsoft Windows はデフォルトで、エージェントのオペレーションに関する統計を収集し、AWS に送信します。この情報には個人を特定できる情報は含まれていません。収集したデータや AWS のサービスにストリーミングするデータは含まれません。60 分ごとにこのメトリクスデータを約 1 KB 収集します。

これらの統計の収集と送信を中止することができます。これを行うには、ソース、シンク、およびパイプと同じレベルで、次のキーと値のペアを appsettings.json 設定ファイルに追加します。

```
"Telemetrics":
    { "off": "true" }
```

たとえば、次の設定ファイルはソース、シンク、パイプを設定し、さらにテレメトリクスを無効にします。

```
{
  "Sources": [
    {
      "Id": "ApplicationLogSource",
      "SourceType": "DirectorySource",
      "Directory": "C:\\LogSource\\",
      "FileNameFilter": "*.log",
      "RecordParser": "SingleLine"
    }
  ],
  "Sinks": [
    {
       "Id": "ApplicationLogKinesisFirehoseSink",
       "SinkType": "KinesisFirehose",
       "StreamName": "ApplicationLogFirehoseDeliveryStream",
       "Region": "us-east-1"
    }
    ],
  "Pipes": [
    {
      "Id": "ApplicationLogSourceToApplicationLogKinesisFirehoseSink",
      "SourceRef": "ApplicationLogSource",
      "SinkRef": "ApplicationLogKinesisFirehoseSink"
    }
  ],
  "Telemetrics":
    {
      "off": "true"
    }
}
```

テレメトリが有効になっている場合は、次のような測定メトリクスを収集します。

`ClientId`

ソフトウェアのインストール時に自動的に割り当てられた一意の ID。

`ClientTimestamp`

テレメトリが収集された日時。

`OSDescription`

オペレーティングシステムの説明。

`DotnetFramework`

現在の dotnet フレームワークのバージョン。

`MemoryUsage`

Windows 用 Kinesis エージェントが消費するメモリの量 (MB)。

`CPUUsage`

Windows CPU 使用率の割合 (10 進数)。たとえば、0.01 は 1% です。

`InstanceId`

Windows 用 Kinesis エージェントが Amazon EC2 インスタンスで実行されている場合の Amazon EC2 インスタンス ID。

`InstanceType (string)`

Amazon EC2 インスタンスタイプ。Windows 用 Kinesis エージェントが Amazon EC2 インスタンスで実行されている場合です。

さらに、Windows メトリックス用 Kinesis エージェントのリスト にリストされているメトリクスを収集します。

チュートリアル: Windows 用 Kinesis エージェントを使用して JSON ログファイルを Amazon S3 にストリーミング

このチュートリアルでは、Microsoft Windows 用 Amazon Kinesis Agent (Windows 用 Kinesis Agent) を使用してデータパイプラインをセットアップするための詳細な手順を示します。

このチュートリアルには次のステップが含まれます。

- Windows 用の Kinesis Agent を使用して JSON 形式のログファイルをにストリーム処理するAmazon Simple Storage Service (Amazon S3)経由Amazon Kinesis Data Firehose。Windows 用 Kinesis エージェントの詳細については、Amazon Kinesis Agent とは何ですか?。
- オブジェクトのデコレーションを使用してストリーミングする前にログデータを強化します。詳細については、「シンクデコレーションの設定」を参照してください。
- を使用するAmazon Athenaを使用して、特定の種類のログレコードを検索します。

Prerequisites

AWS アカウントをまだ持っていない場合は、」の手順に従ってください。AWS アカウントのセットアップを得る。

トピック

- ステップ 1: AWS のサービスを設定する
- ステップ 2: Windows 用 Kinesis エージェントのインストール、構成、実行
- ステップ 3: Amazon S3 のログデータにクエリを実行する
- 次のステップ

ステップ 1: AWS のサービスを設定する

以下の手順に従って、Microsoft Windows 用 Amazon Kinesis Agent を使用して Amazon Simple Storage Service (Amazon S3) にログデータをストリーミングするためにお客様の環境を準備します。詳細と前提条件については、チュートリアル: Amazon S3 への JSON ログファイルのストリーミング を参照してください。

AWS マネジメントコンソールを使用して、AWS Identity and Access Management (IAM) 、Amazon S3、Kinesis Data Firehose e、および Amazon Elastic Compute

Cloud (Amazon EC2) を設定し、EC2 インスタンスから Amazon S3 へのログデータのストリーミングを準備します。

トピック

IAM ポリシーおよびロールを設定する

以下のポリシーを作成し、Kinesis Agent for Windows が特定の Kinesis Data Firehose 配信ストリームにレコードをストリーミングすることを承認します。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor1",
            "Effect": "Allow",
            "Action": [
                "firehose:PutRecord",
                "firehose:PutRecordBatch"
            ],
            "Resource": "arn:aws:firehose:region:account-id:deliverystream/log-
delivery-stream"
        }
    ]
}
```

置換*region*を Kinesis Data Firehose 配信ストリームが作成される AWS リージョンの名前に置き換えます (us-east-1など) 。*account-id* を配信ストリームが作成される AWS アカウントの 12 桁のアカウント ID に置き換えます。

ナビゲーションバーで、[] を選択します。サポートを選択し、サポートセンター。現在サインインしている 12 桁のアカウント番号 (ID) が [] に表示されます。サポートセンターナビゲーションペイン。

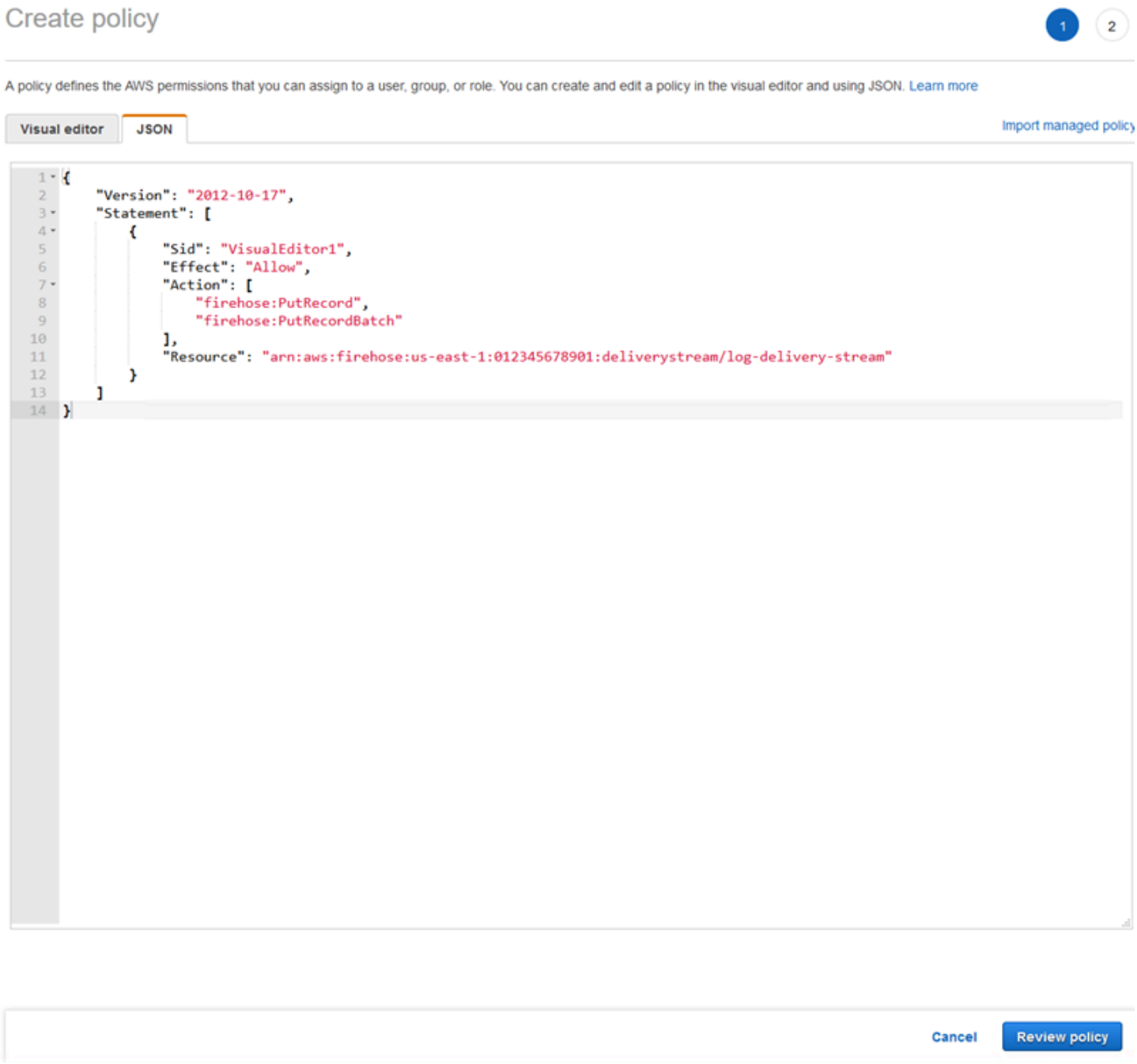
次の手順に従ってポリシーを作成します。ポリシー `log-delivery-stream-access-policy` に名前を付けます。

JSON ポリシーエディタを使用してポリシーを作成するには

1. AWS マネジメントコンソールにサインインして、IAM コンソール (https://console.aws.amazon.com/iam/) を開きます。
2. 左側にあるナビゲーションペインで、[ポリシー] を選択します。

   初めて [ポリシー] を選択する場合には、[管理ポリシーにようこそ] ページが表示されます。[Get Started] を選択します。
3. ページの上部で、[ポリシーの作成] を選択します。
4. [JSON] タブを選択します。
5. JSON ポリシードキュメントを入力します。IAM ポリシー言語の詳細については、「」を参照してください。IAM JSON ポリシーリファレンス()IAM ユーザーガイド。
6. 完了したら、[ポリシーの確認] を選択します。Policy Validator によって、構文エラーがある場合はレポートされます。

   > **Note**
   >
   > いつでも [Visual editor (ビジュアルエディタ)] タブと [JSON] タブを切り替えることができます。ただし、変更を加えたり、ポリシーの確認()Visual editor (ビジュアルエディタ)] タブで、IAM はポリシーを再構成してビジュアルエディタに合わせて最適化することがあります。詳細については、「」を参照してください。ポリシーの再構成()IAM ユーザーガイド。
7. [ポリシーの確認] ページに作成するポリシーの [名前] と [説明] (オプション) を入力します。ポリシーの [概要] を確認して、ポリシーで許可されている権限を確認します。次に [ポリシーの作成] を選択して作業を保存します。



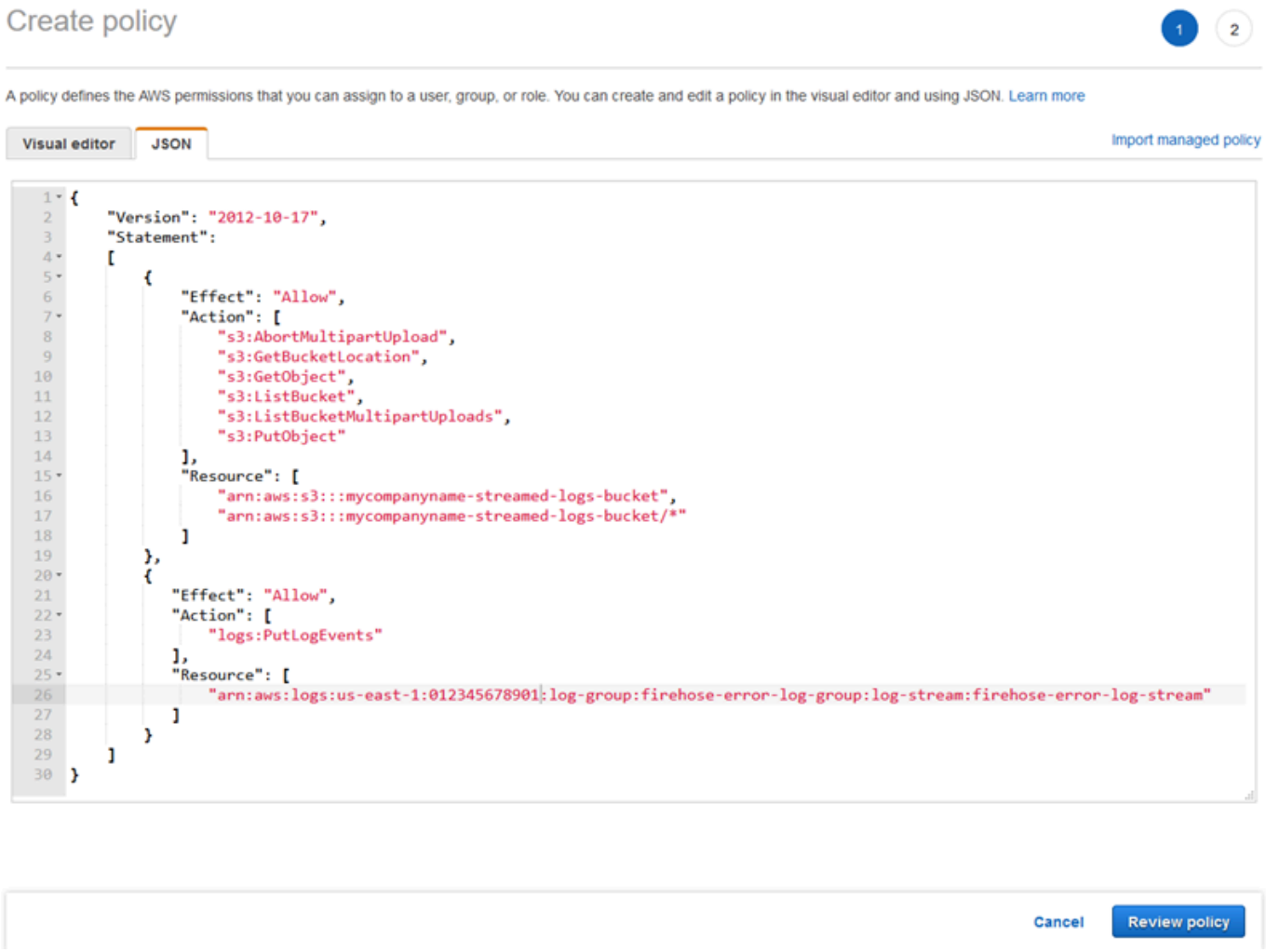
S3 バケットへのアクセス権をKinesis Data Firehose に付与するロールを作成するには

1. 前の手順を使用して、以下の JSON を使用して定義される `firehose-s3-access-policy` という名前のポリシーを作成します。

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

```
    "Statement":
    [
        {
            "Effect": "Allow",
            "Action": [
                "s3:AbortMultipartUpload",
                "s3:GetBucketLocation",
                "s3:GetObject",
                "s3:ListBucket",
                "s3:ListBucketMultipartUploads",
                "s3:PutObject"
            ],
            "Resource": [
                "arn:aws:s3:::bucket-name",
                "arn:aws:s3:::bucket-name/*"
            ]
        },
        {
           "Effect": "Allow",
           "Action": [
               "logs:PutLogEvents"
           ],
           "Resource": [
               "arn:aws:logs:region:account-id:log-group:firehose-error-log-
group:log-stream:firehose-error-log-stream"
           ]
        }
    ]
}
```

bucket-name を口グが保存される一意のバケット名に置き換えます。置換*region*CloudWatch Logs ロググループとログストリームが作成される AWS リージョンを使用します。これらは、Kinesis Data Firehose 経由でデータを Amazon S3 にストリーミングしているときにエラーが発生した場合に口グを記録するためのものです。*account-id* をロググループとログストリームが作成されるアカウントの 12 桁のアカウント ID に置き換えます。



2. IAM コンソールのナビゲーションペインで、[Roles]、[Create role] の順に選択します。
3. [AWS サービスロールタイプを選択してから、[] を選択します。Kinesisのサービス。
4. 選択Kinesis Data Firehoseユースケースに [] を選択してから、[] を選択します。次へ: アクセス許可。
5. 検索ボックスに「」と入力します。**firehose-s3-access-policy**] を選択し、そのポリシーを選択してから、[] を選択します。次へ: 確認。
6. [ロール名] ボックスに、「**firehose-s3-access-role**」と入力します。
7. [ロールの作成] を選択します。

Kinesis Agent for Windows を実行する EC2 インスタンスのインスタンスプロファイルに関連付けるロールを作成するには

1. IAM コンソールのナビゲーションペインで、[Roles]、[Create role] の順に選択します。

2. [AWS サービスロールタイプを選択し、[] を選択します。EC2。
3. 選択次へ: アクセス許可。
4. 検索ボックスに「**log-delivery-stream-access-policy**」と入力します。
5. ポリシーを選択し、[] メニューから [] を選択します。次へ: 確認。
6. [ロール名] ボックスに、「**kinesis-agent-instance-role**」と入力します。
7. [ロールの作成] を選択します。

Amazon S3 バケットを作成する

Kinesis Data Firehose がログをストリーミングする S3 バケットを作成します。

ログストレージ用 S3 バケットを作成するには

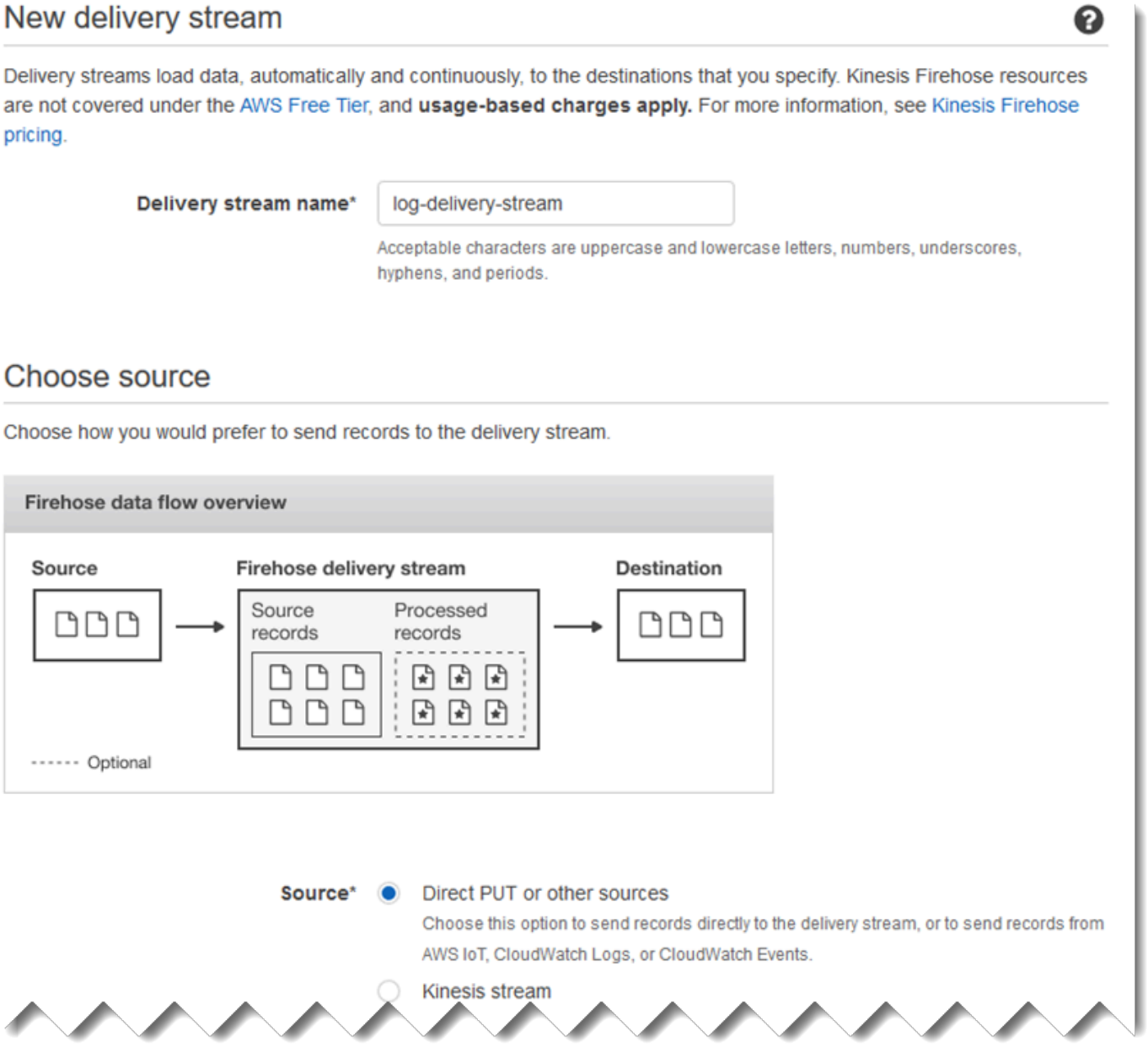
1. Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
2. [バケットを作成する] を選択します。
3. [バケット名] ボックスに、IAM ポリシーおよびロールを設定する で選択した一意の S3 バケット名を入力します。
4. バケットを作成するリージョンを選択します。通常、これは Kinesis Data Firehose 配信ストリームと Amazon EC2 インスタンスを作成するのと同じリージョンです。
5. [作成] を選択します。

Kinesis Data Firehose 配信ストリームの作成

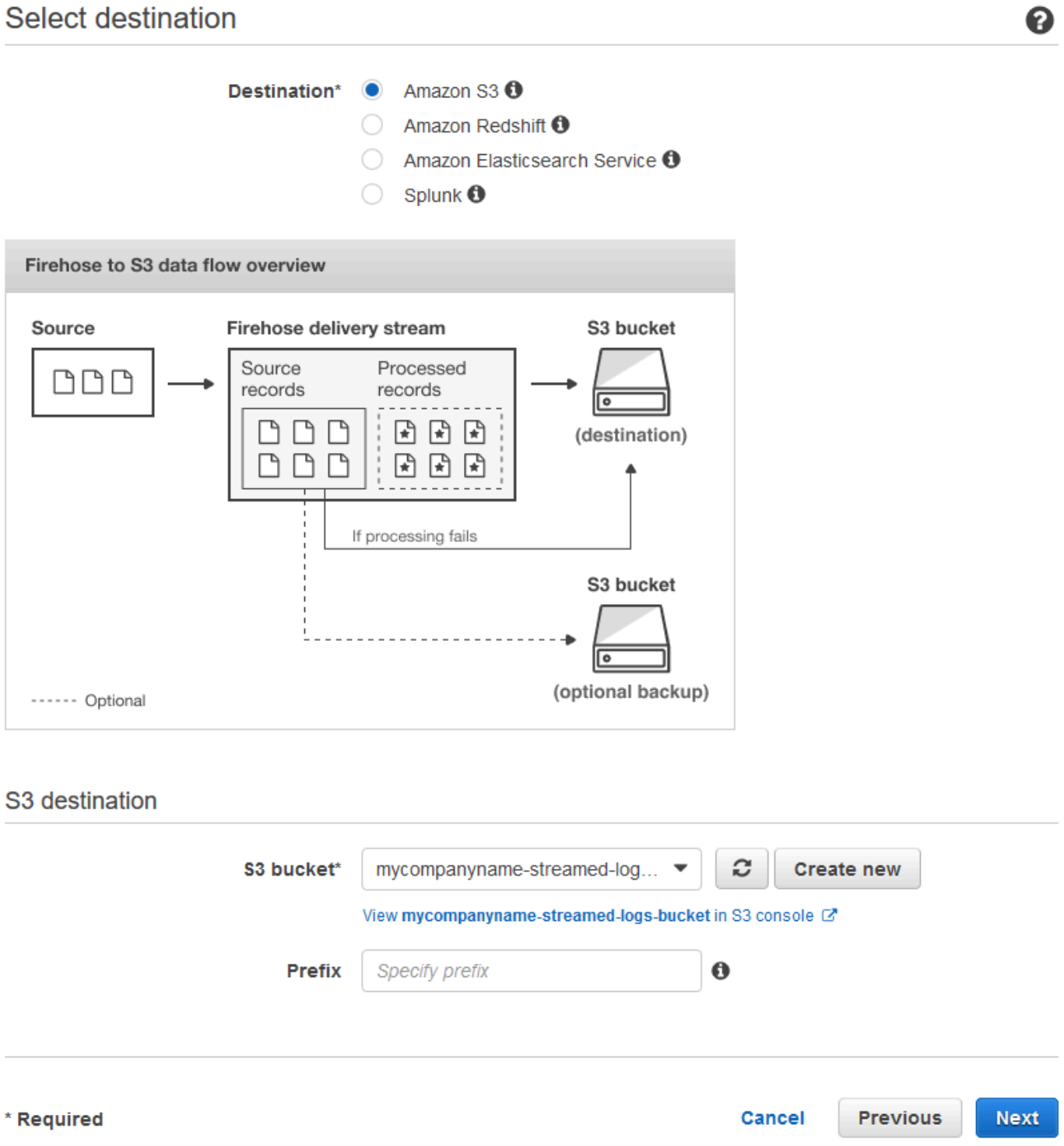
Amazon S3 にストリーム処理されたレコードを保存する Kinesis Data Firehose 配信ストリームを作成します。

Kinesis Data Firehose 配信ストリームを作成するには

1. Kinesis Data Firehose e コンソール (https://console.aws.amazon.com/firehose/。
2. [Create Delivery Stream] を選択します。
3. [Delivery stream name (配信ストリーム名)] ボックスに「**log-delivery-stream**」と入力します。
4. [ソース] で、[Direct PUT or other sources (Direct PUT またはその他のソース)] を選択します。

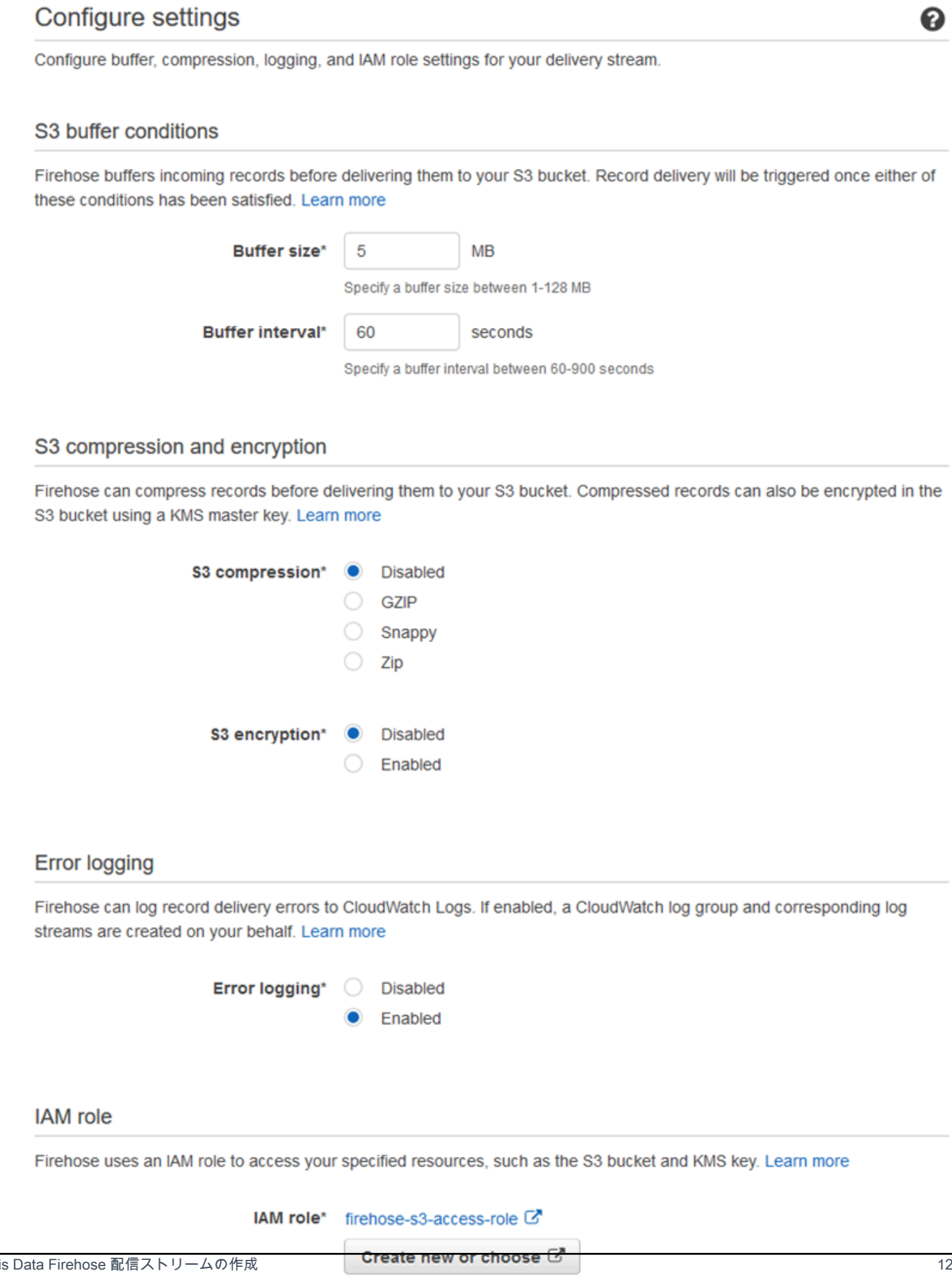


5. [Next] を選択します。
6. [次へ] をもう一度選択します。
7. 送信先として [] を選択します。Amazon S3。
8. [S3 バケット] に Amazon S3 バケットを作成する で作成したバケットの名前を選択します。



9. [Next] を選択します。
10. [Buffer interval (バッファ間隔)] ボックスに「**60**」と入力します。
11. [IAM ロール] で、[Create new or choose (新規作成または選択する)] を選択します。
12. [IAM ロール] に `firehose-s3-access-role` を選択します。

13. [許可] を選択します。


Configure settings
Configure buffer, compression, logging, and IAM role settings for your delivery stream.
S3 buffer conditions
Firehose buffers incoming records before delivering them to your S3 bucket. Record delivery will be triggered once either of these conditions has been satisfied. Learn more
Buffer size*
5
MB
Specify a buffer size between 1-128 MB
Buffer interval*
60
seconds
Specify a buffer interval between 60-900 seconds
S3 compression and encryption
Firehose can compress records before delivering them to your S3 bucket. Compressed records can also be encrypted in the S3 bucket using a KMS master key. Learn more
S3 compression*
Disabled
GZIP
Snappy
Zip
S3 encryption*
Disabled
Enabled
Error logging
Firehose can log record delivery errors to CloudWatch Logs. If enabled, a CloudWatch log group and corresponding log streams are created on your behalf. Learn more
Error logging*
Disabled
Enabled
IAM role
Firehose uses an IAM role to access your specified resources, such as the S3 bucket and KMS key. Learn more
IAM role*
firehose-s3-access-role
Create new or choose

14. [Next] を選択します。
15. [Create delivery stream (配信ストリームの作成)] を選択します。

Windows 用 Kinesis エージェントを実行する Amazon EC2 インスタンスの作成

Kinesis Data Firehose 経由でログレコードをストリーミングするために Kinesis Agent for Windows を使用する EC2 インスタンスを作成します。

EC2 インスタンスを作成するには

1. Amazon EC2 コンソール (https://console.aws.amazon.com/ec2/) を開きます。
2. 次の追加ステップを使用して、Amazon EC2 Windows インスタンスの使用開始の指示に従います。

   - インスタンスの [IAM ロール] に `kinesis-agent-instance-role` を選択します。
   - インターネットに接続しているパブリックの Virtual Private Cloud (VPC) をまだ作成していない場合は、Amazon EC2 での設定()Windows インスタンス用の Amazon EC2 ユーザーガイド。
   - ユーザーのコンピュータからのみ、またはユーザーの組織のコンピュータからのみにインスタンスへのアクセスを制限するセキュリティグループを作成または使用します。詳細については、「」を参照してください。Amazon EC2 での設定()Windows インスタンス用の Amazon EC2 ユーザーガイド。
   - 既存のキーペアを指定する場合は、そのキーペアのプライベートキーにアクセスできることを確認します。または、新規キーペアを作成し、安全な場所にプライベートキーを保存します。
   - 続行する前に、インスタンスが実行され、2 つのうち 2 つのヘルスチェックが完了するまで待機します。
   - インスタンスにはパブリック IP アドレスが必要です。割り当てられていない場合は、Elastic IP アドレス()Windows インスタンス用の Amazon EC2 ユーザーガイド。

次のステップ

ステップ 2: Windows 用 Kinesis エージェントのインストール、構成、実行

ステップ 2: Windows 用 Kinesis エージェントのインストール、構成、実行

このステップでは、AWS マネジメントコンソールを使用して、で起動したインスタンスにリモートで接続します。Windows 用 Kinesis エージェントを実行する Amazon EC2 インスタンスの作成。次に、インスタンスに Microsoft Windows 用 Amazon Kinesis Agent をインストールし、Kinesis Agent for Windows 用の設定ファイルを作成およびデプロイして、AWSkineSistapのサービス。

1. Remore Desktop Protocol (RDP) 経由でリモートでインスタンスに接続するには、ステップ 2: インスタンスへの接続()Windows インスタンス用の Amazon EC2 ユーザーガイド。
2. インスタンスで、Windows Server Manager を使用して、ユーザーと管理者の Microsoft Internet Explorer セキュリティ強化の構成を無効にします。詳細については、Microsoft TechNet ウェブサイトの How To Turn Off Internet Explorer Enhanced Security Configuration を参照してください。
3. インスタンスで、Windows 用 Kinesis エージェントをインストールして設定します。詳細については、「Windows 用 Kinesis エージェントのインストール」を参照してください。
4. インスタンスで、メモ帳を使用して Kinesis Agent for Windows 設定ファイルを作成します。ファイルを %PROGRAMFILES%\Amazon\AWSKinesisTap\appsettings.json に保存します。設定ファイルに以下のコンテンツを追加します。

```
{
  "Sources": [
    {
      "Id": "JsonLogSource",
      "SourceType": "DirectorySource",
      "RecordParser": "SingleLineJson",
      "Directory": "C:\\LogSource\\",
      "FileNameFilter": "*.log",
      "InitialPosition": 0
    }
  ],
  "Sinks": [
    {
      "Id": "FirehoseLogStream",
      "SinkType": "KinesisFirehose",
      "StreamName": "log-delivery-stream",
      "Region": "us-east-1",
      "Format": "json",
```

```
      "ObjectDecoration": "ComputerName={ComputerName};DT={timestamp:yyyy-MM-dd
 HH:mm:ss}"
    }
  ],
  "Pipes": [
    {
      "Id": "JsonLogSourceToFirehoseLogStream",
      "SourceRef": "JsonLogSource",
      "SinkRef": "FirehoseLogStream"
    }
  ]
}
```

このファイルは、Kinesis Agent for Windows が JSON 形式のログレコードを`c:\logsource\`ディレクトリ (source) を Kinesis Data Firehose 配信ストリームに`log-delivery-stream`(sink). 各ログレコードが Kinesis Data Firehose にストリーミングされる前に、コンピュータの名前とタイムスタンプが格納されている 2 つの追加のキーと値のペアによって強化されます。

5. `c:\LogSource\` ディレクトリを作成し、メモ帳を使用して、そのディレクトリに以下の内容の `test.log` ファイルを作成します。

```
{ "Message": "Copasetic message 1", "Severity": "Information" }
{ "Message": "Copasetic message 2", "Severity": "Information" }
{ "Message": "Problem message 2", "Severity": "Error" }
{ "Message": "Copasetic message 3", "Severity": "Information" }
```

6. 昇格された PowerShell セッションで、次のコマンドを使用して AWSKinesisTap サービスを開始します。

```
Start-Service -ServiceName AWSKinesisTap
```

7. ファイルエクスプローラーで、`%PROGRAMDATA%\Amazon\AWSKinesisTap\logs` ディレクトリを参照します。最新のログファイルを開きます。ログファイルは次のようになります。

```
2018-09-28 23:51:02.2472 Amazon.KinesisTap.Hosting.LogManager INFO Registered
 factory Amazon.KinesisTap.AWS.AWSEventSinkFactory.
2018-09-28 23:51:02.2784 Amazon.KinesisTap.Hosting.LogManager INFO Registered
 factory Amazon.KinesisTap.Windows.PerformanceCounterSinkFactory.
2018-09-28 23:51:02.5753 Amazon.KinesisTap.Hosting.LogManager INFO Registered
 factory Amazon.KinesisTap.Core.DirectorySourceFactory.
```

```
2018-09-28 23:51:02.5909 Amazon.KinesisTap.Hosting.LogManager INFO Registered
 factory Amazon.KinesisTap.ExchangeSource.ExchangeSourceFactory.
2018-09-28 23:51:02.5909 Amazon.KinesisTap.Hosting.LogManager INFO Registered
 factory Amazon.KinesisTap.Uls.UlsSourceFactory.
2018-09-28 23:51:02.5909 Amazon.KinesisTap.Hosting.LogManager INFO Registered
 factory Amazon.KinesisTap.Windows.WindowsSourceFactory.
2018-09-28 23:51:02.9347 Amazon.KinesisTap.Hosting.LogManager INFO Registered
 factory Amazon.KinesisTap.Core.Pipes.PipeFactory.
2018-09-28 23:51:03.5128 Amazon.KinesisTap.Hosting.LogManager INFO Registered
 factory Amazon.KinesisTap.AutoUpdate.AutoUpdateFactory.
2018-09-28 23:51:03.5440 Amazon.KinesisTap.Hosting.LogManager INFO Performance
 counter sink  started.
2018-09-28 23:51:03.7628 Amazon.KinesisTap.Hosting.LogManager INFO
 KinesisFirehoseSink id FirehoseLogStream for StreamName log-delivery-stream
 started.
2018-09-28 23:51:03.7784 Amazon.KinesisTap.Hosting.LogManager INFO Connected source
 JsonLogSource to sink FirehoseLogStream
2018-09-28 23:51:03.7940 Amazon.KinesisTap.Hosting.LogManager INFO DirectorySource
 id JsonLogSource watching directory C:\LogSource\ with filter *.log started.
```

このログファイルは、サービスが開始され、c:\LogSource\ ディレクトリからログレコードが収集されていることを示します。各行は、単一の JSON オブジェクトとして解析されます。コンピュータ名とタイムスタンプのキーと値のペアが各オブジェクトに追加されます。その後、Kinesis Data Firehose にストリーム処理されます。

8. 1 ~ 2 分で、で作成した Amazon S3 バケットに移動します。Amazon S3 バケットを作成するAWS マネジメントコンソールを使用して。コンソールで正しいリージョンを選択したことを確認してください。

   そのバケットは、現在の年のフォルダがあります。そのフォルダを開き、現在のフォルダを表示します。そのフォルダを開き、当日のフォルダを表示します。そのフォルダを開き、現在の時間 (UTC) のフォルダを表示します。そのフォルダを開き、log-delivery-stream の名前で始まる 1 つ以上の項目を表示します。

9. 最新の項目の内容を開き、ログレコードが必要な機能強化により Amazon S3 に正常に保存されていることを確認します。すべてが正しく設定されている場合、次のような内容になります。

```
{"Message":"Copasetic message 1","Severity":"Information","ComputerName":"EC2AMAZ-
ABCDEFGH","DT":"2018-09-28 23:51:04"}
{"Message":"Copasetic message 2","Severity":"Information","ComputerName":"EC2AMAZ-
ABCDEFGH","DT":"2018-09-28 23:51:04"}
{"Message":"Problem message 2","Severity":"Error","ComputerName":"EC2AMAZ-
ABCDEFGH","DT":"2018-09-28 23:51:04"}
{"Message":"Copasetic message 3","Severity":"Information","ComputerName":"EC2AMAZ-
ABCDEFGH","DT":"2018-09-28 23:51:04"}
```

10. 以下のいずれかの問題を解決するための情報については、Microsoft Windows 用 Amazon Kinesis エージェントのトラブルシューティング を参照してください。

    - Windows 用の Kinesis Agent のログファイルにエラーがあります。
    - Amazon S3 の予期されたフォルダや項目が存在しません。
    - Amazon S3 の項目の内容が正しくない。

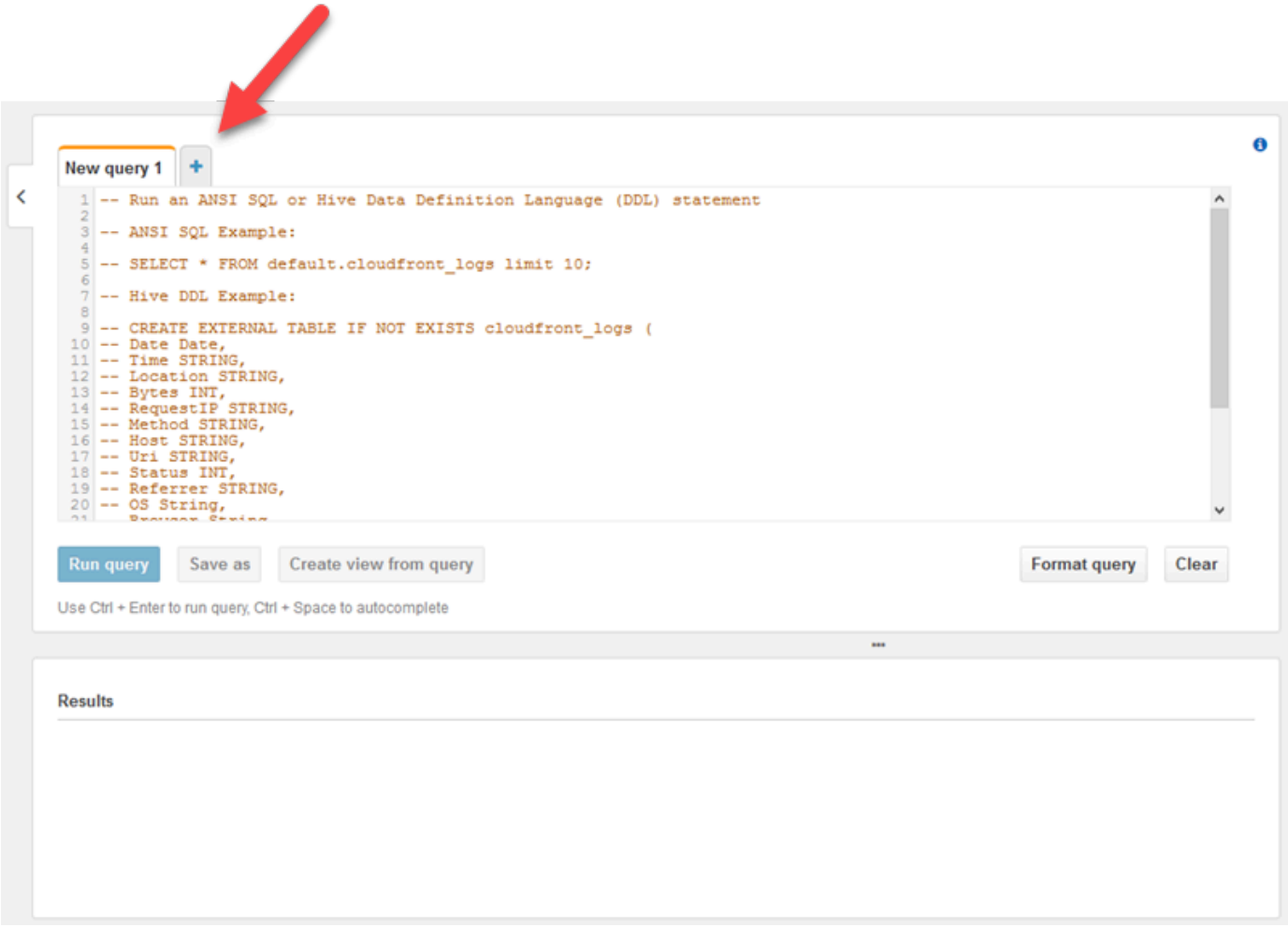
次のステップ

ステップ 3: Amazon S3 のログデータにクエリを実行する

ステップ 3: Amazon S3 のログデータにクエリを実行する

Microsoft Windows 向けこの Amazon Kinesis エージェントの最後のステップでチュートリアルでは、Amazon Athena を使用して Amazon Simple Storage Service (Amazon S3) に保存されたログデータをクエリします。

1. https://console.aws.amazon.com/athena/ で Athena コンソールを開きます。
2. プラス記号 (+) Athena、新しいクエリウィンドウを作成します。

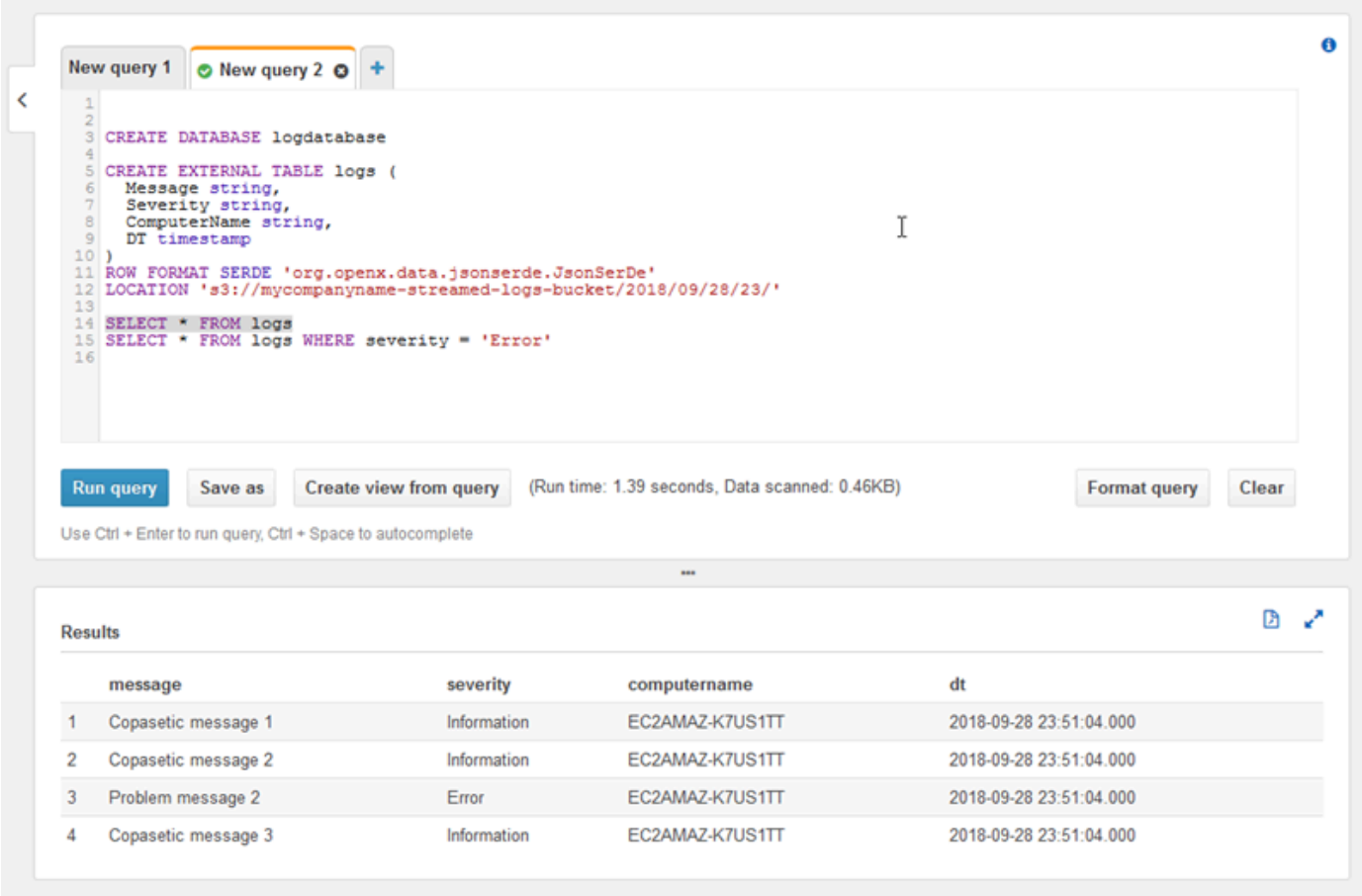


3. 次の内容をクエリウィンドウに入力します。

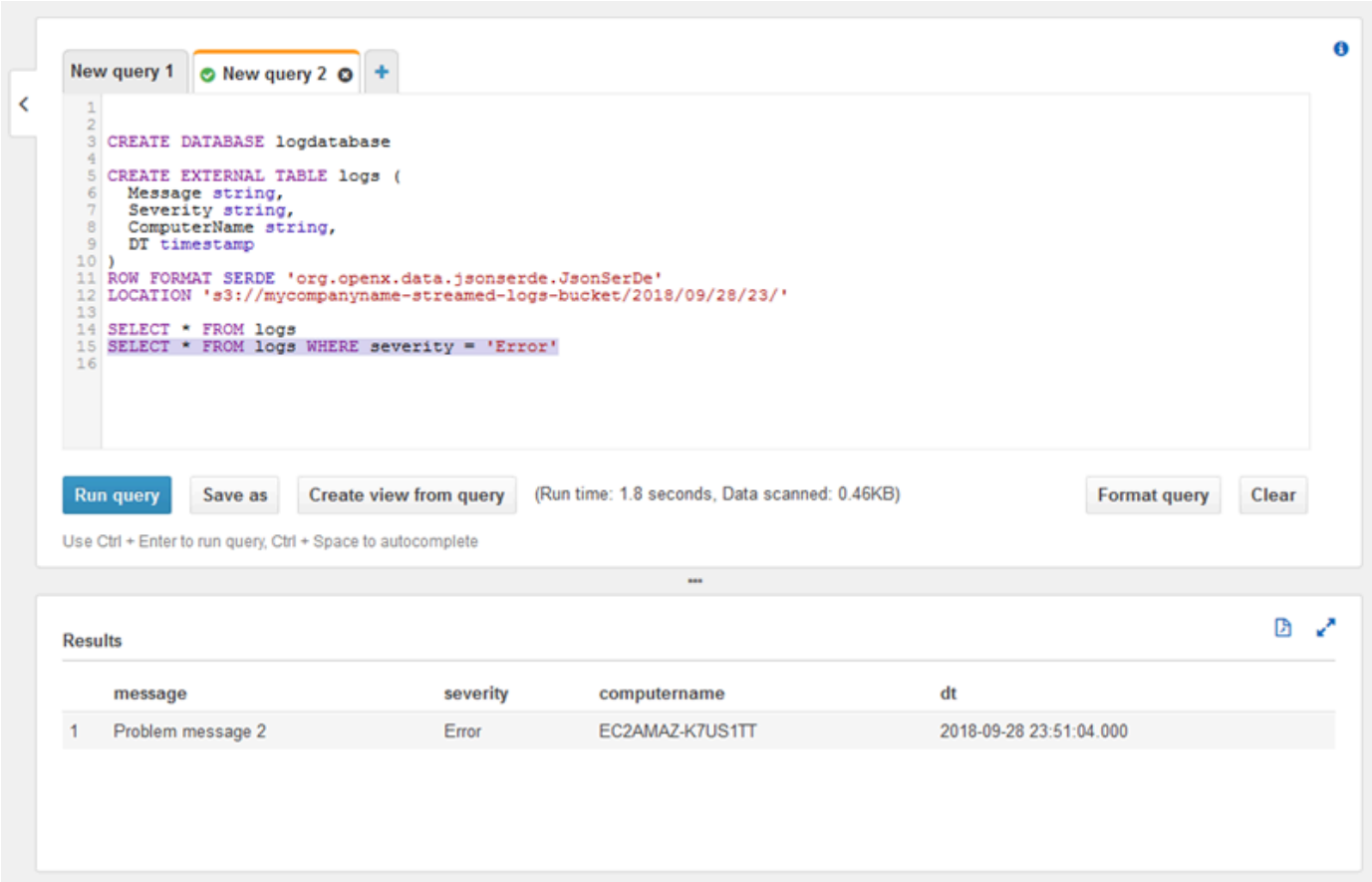
```
CREATE DATABASE logdatabase

CREATE EXTERNAL TABLE logs (
  Message string,
  Severity string,
  ComputerName string,
  DT timestamp
)
ROW FORMAT SERDE 'org.openx.data.jsonserde.JsonSerDe'
LOCATION 's3://bucket/year/month/day/hour/'

SELECT * FROM logs
SELECT * FROM logs WHERE severity = 'Error'
```

bucket を Amazon S3 バケットを作成する で作成したバケットの名前に置き換えます。置換*year*,*month*,*day*および*hour*を Amazon S3 ログファイルが作成されたときの年、月、日と時間 (UTC) で指定します。

4. CREATE DATABASE ステートメントにテキストを選択し、[クエリの実行] を選択します。これにより、Athena にログデータベースが作成されます。
5. CREATE EXTERNAL TABLE ステートメントにテキストを選択し、[クエリの実行] を選択します。これで、JSON のスキーマを Athena テーブルのスキーマにマッピングして、ログデータで S3 バケットを参照する Athena テーブルが作成されます。
6. 最初の SELECT ステートメントにテキストを選択し、[クエリの実行] を選択します。テーブルのすべての行が表示されます。



7. 2 番目の SELECT ステートメントにテキストを選択し、[クエリの実行] を選択します。Error の緊急度のログレコードを表すテーブル内の行のみが表示されます。このようなクエリでは、サイズが大きい可能性があるログレコードから興味深いログレコードが見つかります。



次のステップ

AWS マネジメントコンソールを使用して、チュートリアルで作成されたリソースをクリーンアップします。

1. EC2 インスタンスを終了します (Amazon EC2 Windows インスタンスの使用開始のステップ 3 を参照してください)。

   > **Important**
   >
   > 内にないインスタンスを起動した場合、AWS 無料利用枠に設定されている場合、インスタンスを終了するまでそのインスタンスに対して課金されます。

2. Kinesis Data Firehose 配信ストリームを削除します。

   a. Kinesis Data Firehose e コンソール (https://console.aws.amazon.com/firehose/。

   b. 作成した配信ストリームを選択します。

c. [削除] を選択します。

d. [Delete delivery stream (配信ストリームの削除)] を選択します。

3. S3 バケットを削除します。手順については、以下を参照してください。S3 バケットを削除する方法()Amazon Simple Storage Service Console。

詳細については、以下のトピックを参照してください。

- Microsoft Windows 用の Amazon Kinesis エージェントの設定
- Amazon Kinesis Data Firehose とは何ですか?
- Amazon S3 とは?
- Amazon Athena とは何ですか?

Microsoft Windows 用 Amazon Kinesis エージェントのトラブルシューティング

Microsoft Windows 用 Amazon Kinesis Agent を使用する際の問題を診断して修正するには、以下の手順を実行します。

トピック

デスクトップまたはサーバーから予想される AWS のサービスにデータがストリーミングされない

Symptoms

Windows の Kinesis Agent からデータのストリームを受け取るように設定されているさまざまな AWS サービスによってホストされるログ、イベント、およびメトリクスを調べると、Kinesis Agent for Windows によってデータがストリーミングされていません。

Causes

この問題の原因はいくつか考えられます。

- ソース、シンク、またはパイプが正しく設定されていない。
- Windows 用 Kinesis Agent の認証が正しく設定されていない。
- Windows 用 Kinesis Agent の承認が正しく設定されていない。
- `DirectorySource` 宣言で提供された正規表現に工ラーがある。
- 存在しないディレクトリが `DirectorySource` 宣言に指定されている。

- AWS サービスに無効な値が指定されているため、Windows 用 Kinesis Agent からのリクエストが拒否される。
- シンクが、指定された AWS リージョンまたは暗黙的な AWS リージョンに存在しないリソースを参照している。
- 無効なクエリが `WindowsEventLogSource` 宣言に指定されている。
- 無効な値がソースの `InitialPosition` キーと値のペアに指定されている。
- `appsettings.json` 設定ファイルが目的のファイルの JSON スキーマに準拠していない。
- AWS マネジメントコンソールで選択されたリージョンとは別のリージョンにデータがストリーミングされる。
- Windows 用 Kinesis Agent が正しくインストールされていないか、実行中でない。

Resolutions

ストリーミングされないデータに関する問題を解決するには、以下の手順を実行します。

1. Windows 用 Kinesis エージェントの口グを`%PROGRAMDATA%\Amazon\AWSKinesisTap \logs`ディレクトリに。文字列「`ERROR`」を検索します。
   a. ソースまたはシンクがロードされなかった場合は、以下を実行します。
      i. エラーメッセージを調べ、ソースまたはシンクの `Id` を見つけます。
      ii. `%PROGRAMFILES%\Amazon\AWSKinesisTap\appsettings.json` 設定ファイル内のその `Id` に対応するソースまたはシンク宣言で、見つかったエラーメッセージに関連するエラーを確認します。詳細については、「Microsoft Windows 用の Amazon Kinesis エージェントの設定」を参照してください。
      iii. エラーに関連する設定ファイルの問題を修正します。
      iv. `AWSKinesisTap` サービスを停止および開始します。次に、最新のログファイルで設定の問題が解決されたことを確認します。
   b. エラーメッセージに、パイプ用の `SourceRef` または `SinkRef` 見つからなかったことが示されている場合は、以下を実行します。
      i. パイプ `Id` を記録します。
      ii. 記録した `Id` に対応する `%PROGRAMFILES%\Amazon\AWSKinesisTap \appsettings.json` 設定ファイルでパイプ宣言を調べます。`SourceRef` と `SinkRef` のキーと値のペアの値が、参照対象のソースおよびシンク宣言の正しいスペルの `Id` であることを確認します。誤字や脱字を修正します。ソースまたはシンク宣言が設定ファイルか

ら欠落している場合は、宣言を追加します。詳細については、「Microsoft Windows 用の Amazon Kinesis エージェントの設定」を参照してください。

iii. AWSKinesisTap サービスを停止および開始します。次に、最新のログファイルで設定の問題が解決されたことを確認します。

c. エラーメッセージに、特定の IAM ユーザーまたはロールに特定のオペレーションを実行する権限が特定の IAM ユーザーまたはロールに与えられていないことが示されている場合は、以下を実行します。

i. Windows 用 Kinesis Agent で正しい IAM ユーザーまたはロールが使用されていることを確認します。そうでない場合は、シンクセキュリティの設定を実行し、正しい IAM ユーザーまたはロールが使用されるように Kinesis Agent for Windows の認証方法を調整します。

ii. 正しい IAM ユーザーまたはロールが使用されている場合は、AWS マネジメントコンソールを使用して、ユーザーまたはロールに関連付けられたポリシーを調べます。Windows の Kinesis Agent がアクセスするすべての AWS リソースのエラーメッセージに記載されているすべての権限がユーザーまたはロールに与えられていることを確認します。詳細については、「認可の設定」を参照してください。

iii. AWSKinesisTap サービスを停止および開始します。次に、最新のログファイルでセキュリティの問題が解決されたことを確認します。

d. エラーメッセージに、%PROGRAMFILES%\Amazon\AWSKinesisTap\appsettings.json 設定ファイルに含まれている正規表現を解析する際に引数エラーが発生したことが示されている場合は、以下を実行します。

i. 設定ファイルの正規表現を調べます。

ii. 正規表現の構文を確認します。正規表現を確認するために使用できるいくつかのウェブサイトがあります。または、以下のコマンドラインを使用して、DirectorySource ソース宣言の正規表現を確認します。

```
cd /D %PROGRAMFILES%\Amazon\AWSKinesisTap
ktdiag.exe /r sourceId
```

sourceId を、正規表現が正しくない DirectorySource ソース宣言の Id キーと値のペアの値に置き換えます。

iii. 設定ファイルが有効になるように正規表現に必要な修正を行います。

iv. AWSKinesisTap サービスを停止および開始します。次に、最新のログファイルで設定の問題が解決されたことを確認します。

e. エラーメッセージに、%PROGRAMFILES%\Amazon\AWSKinesisTap\appsettings.json 設定ファイルに含まれていない正規表現で、特定のシンクに関連する正規表現を解析する際に引数エラーが発生したことが示されている場合は、以下を実行します。

   i. 設定ファイルでシンク宣言を見つけます。

   ii. AWS のサービスに特に関連するキーと値のペアが、そのサービスの検証ルールに準拠する名前を使用していることを確認します。たとえば、CloudWatch Logs グループ名には正規表現を使用して指定された特定の文字のセットのみが含まれている必要があります。[\.\-_/#A-Za-z0-9]+。

   iii. シンク宣言のキーと値のペアで無効な名前を修正し、これらのリソースが AWS で適切に設定されていることを確認します。

   iv. AWSKinesisTap サービスを停止および開始します。次に、最新のログファイルで設定の問題が解決されたことを確認します。

f. エラーメッセージに、パラメータが null か、欠落しているため、ソースまたはシンクをロードできないことが示された場合は、以下を実行します。

   i. ソースまたはシンクの Id を記録します。

   ii. %PROGRAMFILES%\Amazon\AWSKinesisTap\appsettings.json 設定ファイルで、記録した Id に一致するソースまたはシンク宣言を見つけます。

   iii. 関連するシンクタイプについて、「Microsoft Windows 用の Amazon Kinesis エージェントの設定」ドキュメントのソースまたはシンクのタイプの要件と比較して、ソースまたはシンク宣言で提供されたキーと値のペアを確認します。欠落している必要なキーと値のペアをソースまたはシンク宣言に追加します。

   iv. AWSKinesisTap サービスを停止および開始します。次に、最新のログファイルで設定の問題が解決されたことを確認します。

g. エラーメッセージに、ディレクトリ名が無効であることが示されている場合は、以下を実行します。

   i. %PROGRAMFILES%\Amazon\AWSKinesisTap\appsettings.json 設定ファイルで無効なディレクトリ名を見つけます。

   ii. このディレクトリが存在し、ストリーミングする必要があるログファイルが含まれていることを確認します。

   iii. 設定ファイルで指定されたディレクトリ名の誤りや誤字を修正します。

   iv. AWSKinesisTap サービスを停止および開始します。次に、最新のログファイルで設定の問題が解決されたことを確認します。

h. エラーメッセージに、リソースが存在しないことが示されている場合は、以下を実行します。

i. %PROGRAMFILES%\Amazon\AWSKinesisTap\appsettings.json 設定ファイルで、シンク宣言に存在しないリソースのリソースリファレンスを見つけます。

ii. AWS マネジメントコンソールを使用して、シンク宣言に使用される必要がある正しい AWS リージョンでリソースを見つけます。このリソースと設定ファイルで指定されている内容を比較します。

iii. 正しいリソース名と正しいリージョンが指定されるように、設定ファイルでシンク宣言を変更します。

iv. AWSKinesisTap サービスを停止および開始します。次に、最新のログファイルで設定の問題が解決されたことを確認します。

i. エラーメッセージに、特定の WindowsEventLogSource に対してクエリが無効であることが示されている場合は、以下を実行します。

i. %PROGRAMFILES%\Amazon\AWSKinesisTap\appsettings.json 設定ファイルで、エラーメッセージに表示されているものと同じ Id の WindowsEventLogSource 宣言を見つけます。

ii. ソース宣言の Query キーと値のペアの値が「Event queries and Event XML」に準拠していることを確認します。

iii. 準拠するようにクエリを変更します。

iv. AWSKinesisTap サービスを停止および開始します。次に、最新のログファイルで設定の問題が解決されたことを確認します。

j. エラーメッセージに、無効な初期位置が存在することが示されている場合は、以下を実行します。

i. %PROGRAMFILES%\Amazon\AWSKinesisTap\appsettings.json 設定ファイルで、エラーメッセージに表示されているものと同じ Id のソース宣言を見つけます。

ii. 「ブックマーク設定」の説明に従って、許容値に準拠するようにソース宣言の InitialPosition キーと値のペアの値を変更します。

iii. AWSKinesisTap サービスを停止および開始します。次に、最新のログファイルで設定の問題が解決されたことを確認します。

2. %PROGRAMFILES%\Amazon\AWSKinesisTap\appsettings.json 設定ファイルが JSON スキーマに準拠していることを確認します。

a. コマンドプロンプトウィンドウで、以下のラインを呼び出します。

```
cd /D %PROGRAMFILES%\Amazon\AWSKinesisTap
%PROGRAMFILES%\Amazon\AWSKinesisTap\ktdiag.exe /c
```

b. %PROGRAMFILES%\Amazon\AWSKinesisTap\appsettings.json 設定ファイルで検出された問題を修正します。

c. AWSKinesisTap サービスを停止および開始します。次に、最新のログファイルで設定の問題が解決されたことを確認します。

3. より詳細なログ作成情報を取得できるように、ログ記録のレベルを変更します。

a. %PROGRAMFILES%\Amazon\AWSKinesisTap\nlog.xml 設定ファイルを以下の内容と置き換えます。

```
<?xml version="1.0" encoding="utf-8" ?>
<nlog xmlns="http://www.nlog-project.org/schemas/NLog.xsd"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://www.nlog-project.org/schemas/NLog.xsd NLog.xsd"
      autoReload="true"
      throwExceptions="false"
      internalLogLevel="Off" internalLogFile="c:\temp\nlog-internal.log" >

  <!--
  See https://github.com/nlog/nlog/wiki/Configuration-file
  for information on customizing logging rules and outputs.
   -->
  <targets>
    <!--
    add your targets here
    See https://github.com/nlog/NLog/wiki/Targets for possible targets.
    See https://github.com/nlog/NLog/wiki/Layout-Renderers for the possible layout
 renderers.
    -->

    <target name="logfile"
            xsi:type="File"
            layout="${longdate} ${logger} ${uppercase:${level}} ${message}"
            fileName="${specialfolder:folder=CommonApplicationData}/Amazon/
KinesisTap/logs/KinesisTap.log"
     maxArchiveFiles="90"
     archiveFileName="${specialfolder:folder=CommonApplicationData}/Amazon/
KinesisTap/logs/Archive-{################}.log"
     archiveNumbering="Date"
     archiveDateFormat="yyyy-MM-dd"
     archiveEvery="Day"
     />
  </targets>
```

```
  <rules>
    <logger name="*" minlevel="Debug" writeTo="logfile" />
  </rules>
</nlog>
```

b. AWSKinesisTap サービスを停止および開始します。最新のログファイルで、問題の診断および解決に役立つ可能性がある追加のメッセージがログに存在するかどうかを確認します。

4. AWS マネジメントコンソールで、正しいリージョンでリソースを参照していることを確認します。
5. Windows 用 Kinesis Agent がインストールされ、実行中であることを確認します。
   a. Windows で、[スタート] を選択して、[コントロール パネル]、[管理ツール]、[サービス] の順に移動します。
   b. AWSKinesisTap サービスを見つけます。
   c. AWSKinesis ISTAP サービスが表示されない場合は、「」の手順を使用して Windows 用 Kinesis Agent をインストールします。Microsoft Windows 用 Amazon Kinesis エージェントの使用開始。
   d. サービスが表示された場合は、サービスが実行中かどうかを特定します。サービスが実行されていない場合は、サービスのコンテキスト (右クリック) メニューを開き、[開始] を選択します。
   e. %PROGRAMDATA%\Amazon\AWSKinesisTap\logs ディレクトリの最新のログファイルを調べて、サービスが開始されたことを確認します。

Applies to

この情報は、Windows バージョン 1.0.115 以降の Kinesis Agent に適用されます。

予想されるデータがない場合がある

Symptoms

Windows 用 Kinesis Agent は、ほとんどの場合、データをストリーミングしますが、一部のデータが欠落する場合があります。

Causes

この問題の原因はいくつか考えられます。

- ブックマーク機能が使用されていない。
- AWS サービスの現在の設定に基づいた AWS サービスのデータレート制限を超過している。
- AWS サービスの API コールレートの制限を超過している。現在の`appsettings.json`設定ファイルと AWS アカウントの制限値を指定します。

Resolutions

欠落データに関する問題を解決するには、以下の手順を実行します。

1. 「ブックマーク設定」に記載されているブックマーク機能を使用することを検討してください。これにより、Windows 用 Kinesis Agent が停止して起動した場合でも、すべてのデータが最終的に送信されるようになります。
2. Windows 用 Kinesis Agent を使用して、問題を検出します。
   a. 「」の説明に従って、Windows 用 Kinesis Agent のストリーミングを有効にします。Windows メトリックパイプ用の Kinesis エージェントの設定。
   b. 1 つ以上のシンクについて非常に多くの回復不能なエラーが存在する場合は、1 秒あたりに送信されるバイト数またはレコード数を特定します。次に、データがストリーミングされるリージョンとアカウントの AWS サービスに対して設定されている制限内にその値が収まっているかどうかを特定します。
   c. 制限を超えている場合は、送信されるデータの速度または量を減少するか、制限の引き揚げをリクエストするか、シャーディングを増加します (該当する場合)。
   d. 調整を行った後、Windows の組み込みメトリクスを引き続き監視して、状況が解決したことを確認します。

Kinesis Data Streams の制限については、「」を参照してください。Kinesis Data Streams 制限()Kinesis Data Streams 開発者ガイド。Kinesis Data Firehose の制限については、」Amazon Kinesis Data Firehose 制限。

Applies to

この情報は、Windows バージョン 1.0.115 以降の Kinesis Agent に適用されます。

データが間違った形式で到着する

Symptoms

誤ったフォーマットで AWS サービスにデータが到着します。

Causes

この問題の原因はいくつか考えられます。

- appsettings.json 設定ファイルのシンク宣言の Format キーと値のペアの値が正しくない。
- DirectorySource 宣言の RecordParser キーと値のペアの値が正しくない。
- Regex レコードパーサーを使用する DirectorySource 宣言の正規表現が正しくない。

Resolutions

間違ったフォーマットに関する問題を解決するには、以下の手順を実行します。

1. %PROGRAMFILES%\Amazon\AWSKinesisTap\appsettings.json 設定ファイルでシンク宣言を確認します。
2. Format キーと値のペアの正しい値がシンク宣言ごとに指定されていることを確認します。詳細については、「シンク宣言」を参照してください。
3. DirectorySource 宣言でのソースが Format キーと値のペアの xml または json 値を指定するシンクにパイプで接続されている場合は、RecordParser キーと値のペアの以下の値のいずれかが、これらのソースによって指定されていることを確認します。
   - SingleLineJson
   - Regex
   - SysLog
   - Delimited

   その他のレコードパーサーは、テキストベースのみで、XML または JSON 形式を必要とするシンクでは正しく機能しません。
4. ログレコードが DirectorySource ソースタイプによって正しく解析されない場合は、コマンドプロンプトウィンドウで以下のラインを呼び出して、DirectorySource 宣言で指定されたタイムスタンプと正規表現のキーと値のペアを確認します。

```
cd /D %PROGRAMFILES%\Amazon\AWSKinesisTap
ktdiag.exe /r sourceID
```

sourceId を、適切に機能していないと思われる DirectorySource ソース宣言の Id キーと値のペアの値に置き換えます。ktdiag.exe によってレポートされたすべての問題を修正します。

Applies to

この情報は、Windows バージョン 1.0.115 以降の Kinesis Agent に適用されます。

パフォーマンスの問題

Symptoms

Windows Kinesis Agent をインストールして起動したら、アプリケーションおよびサービスのレイテンシーが増加します。

Causes

この問題の原因はいくつか考えられます。

- Windows 用 Kinesis Agent が実行されているマシンに、必要なデータの量をストリーミングする十分な容量がない。
- 不要なデータが 1 つ以上の AWS サービスにストリーミングされる。
- Windows 用 Kinesis Agent は、このような高いデータ転送速度用に設定されていない AWS サービスにデータをストリーミングします。
- Windows 用 Kinesis Agent が、API コールレートの制限が低すぎるアカウントで AWS サービスのオペレーションを呼び出している。

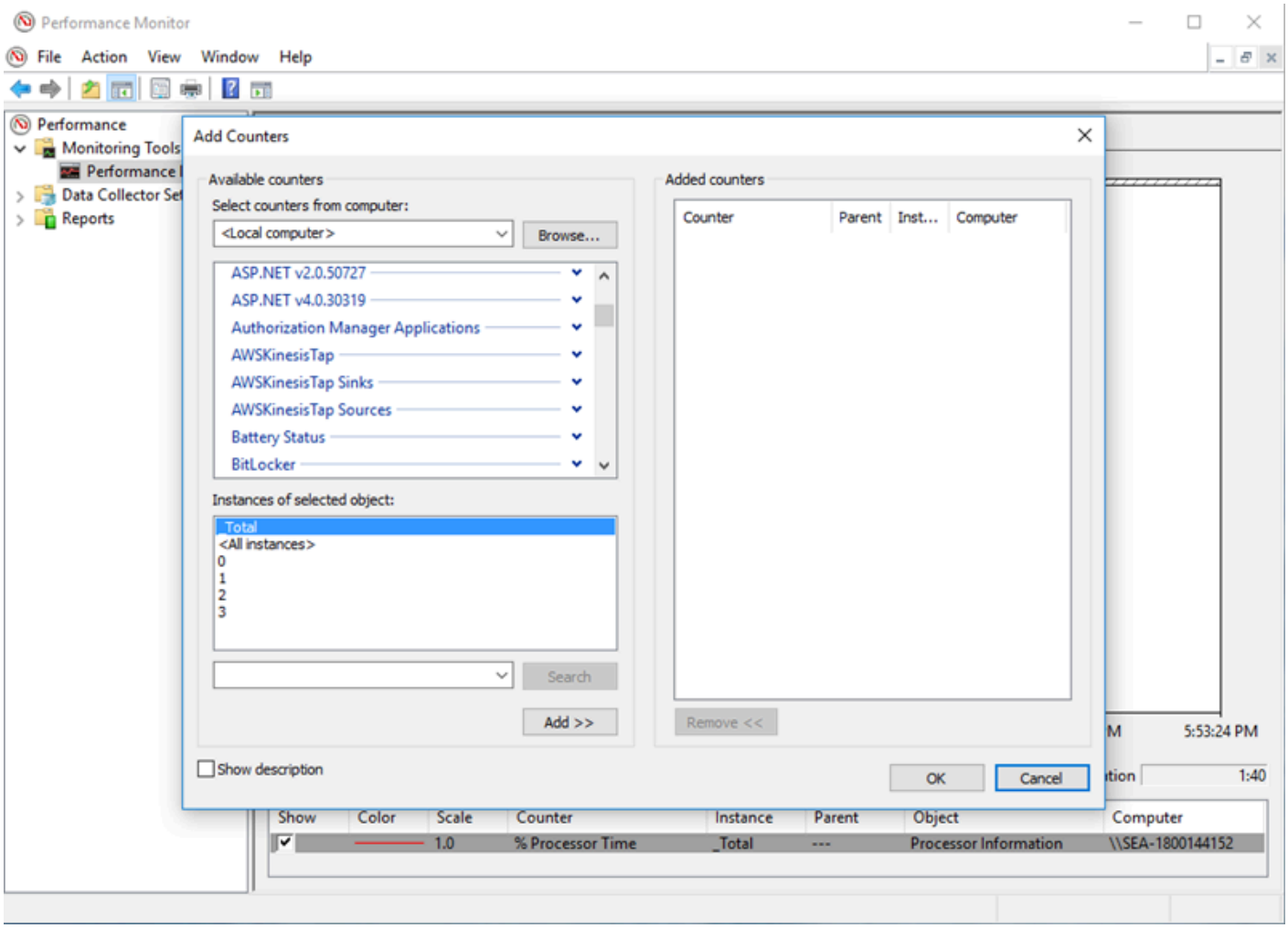
Resolutions

パフォーマンスの問題を解決するには、以下の手順を実行します。

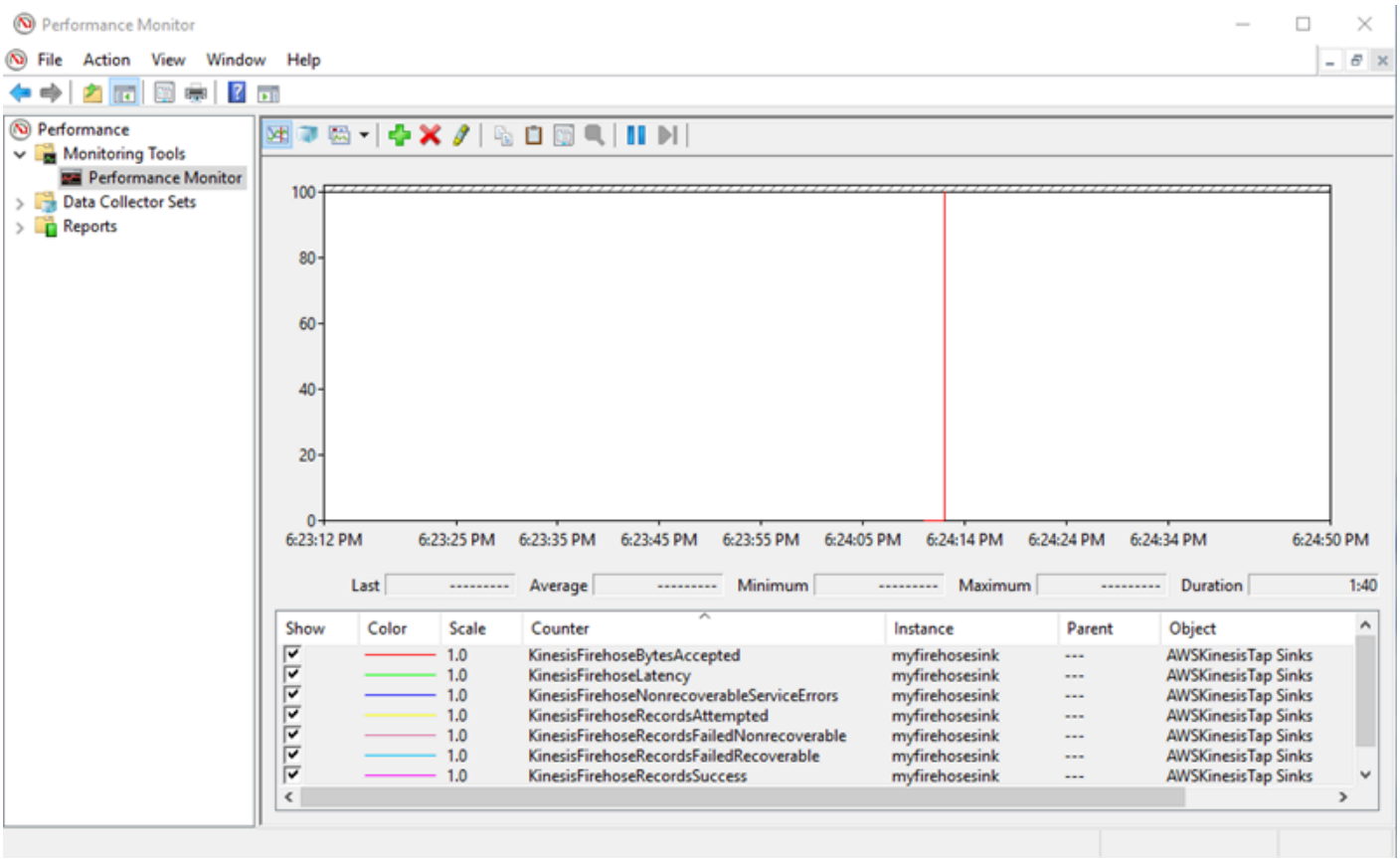
1. Windows のリソースモニターアプリケーションを使用して、メモリ、CPU、ディスク、およびネットワーク使用料を確認します。Windows の Kinesis Agent を使用して大量のデータをスト

リーミングする必要がある場合、設定によっては、これらの領域の一部でより高い容量をマシンにプロビジョニングする必要がある場合があります。

2. フィルタリングを使用して、ログデータの量を減らすことができる場合があります。
   - WindowsEventLogSource 設定 で `Query` キーと値のペアを確認します。
   - パイプの設定 でパイプラインのフィルタリングを確認します。
   - Amazon CloudWatch メトリクスのフィルタリングについては、「」を参照してください。CloudWatch シンクの設定).
3. Windows のパフォーマンスモニターアプリケーションを使用して、Windows の Kinesis エージェントメトリクスを確認するか、CloudWatch にメトリクスをストリーミングします (Windows 組み込みメトリクスソース Kinesis エージェント). Windows パフォーマンスモニターアプリケーションで、Windows のシンクおよびソース用 Kinesis Agent のカウンターを追加できます。カウンターは、[AWSKinesisTap Sinks] および [AWSKinesisTap Sources] カウンターカテゴリの下に一覧表示されます。



たとえば、Kinesis Data Firehose パフォーマンスの問題を診断するには、Kinesis Firehoseパフォーマンスカウンター。



回復可能なエラーが大量にある場合は、[%PROGRAMDATA%\Amazon\AWSKInesisTap\logsディレクトリに。KinesisStream または KinesisFirehose シンクについてスロットリングが発生する場合は、以下を実行します。

- データのストリーミングが速すぎるためにスロットリングが発生した場合は、Kinesis Data Streams のシャード数を増やすことを検討してください。詳細については、「」を参照してください。リシャーディング、拡張、並列処理()Kinesis Data Streams 開発者ガイド。
- Kinesis データストリームの API コールに制限が適用されている場合は、Kinesis Data Streams の API コールの制限を引き揚げるか、シンクのバッファサイズを増やすことを検討してください。詳細については、「」を参照してください。Kinesis Data Streams 制限()Kinesis Data Streams 開発者ガイド。
- データのストリーミングが速すぎる場合は、Kinesis Data Firehose 配信ストリームの速度制限の増加をリクエストすることを検討してください。または、API 呼び出しに制限が適用されている場合は、API 呼び出し制限の増加をリクエストするか (「Amazon Kinesis Data Firehose API コール制限」を参照)、シンクのバッファサイズを増やします。

- Kinesis データストリームストリームのシャード数を増加するか、KinesKinesis Data Firehose 配信システムの速度制限を増加したら、Windows の Kinesis エージェントを修正します。`appsettings.json`設定ファイルを使用して、シンクの 1 秒あたりのレコード数または 1 秒あたりのバイト数を増加します。これを実行しない場合、Windows 用 Kinesis Agent では制限の増加を活用できません。

Applies to

この情報は、Windows バージョン 1.0.115 以降の Kinesis Agent に適用されます。

ディスク容量の不足

Symptoms

Windows 用 Kinesis Agent が、1 つ以上のディスクドライブのディスク容量が非常に低くなっています。

Causes

この問題の原因はいくつか考えられます。

- Windows 用 Kinesis Agent のログ記録設定ファイルが正しくない。
- Windows 用 Kinesis Agent の永続的なキューが正しく設定されていない。
- その他のいくつかのアプリケーションまたはサービスによってディスク容量が消費されている。

Resolutions

ディスク容量の問題を解決するには、以下の手順を実行します。

- Windows の Kinesis Agent ログファイルが含まれているディスクのディスク容量が少ない場合は、ログファイルのディレクトリ (通常は`%PROGRAMDATA%\Amazon\AWSKinesisTap\logs`). 妥当な数のログファイルが保持されており、ログファイルが合理的なサイズであることを確認します。Windows 用 Kinesis Agent のログの場所、保持、および詳細レベルを制御するには、`%PROGRAMFILES%\Amazon\AWSKinesisTap\Nlog.xml`設定ファイルに。
- シンクのキューイング機能が有効な場合は、この機能を使用するシンク宣言を調べます。`QueuePath` キーと値のペアが参照するディスクドライブの容量が、`QueueMaxBatches` キーと値のペアを使用して指定されたバッチの最大数を含むのに十分な容量であることを確認します。

これができない場合は、指定されたディスクドライブの残りのディスク容量にデータが容易に適合するように QueueMaxBatches キーと値のペアの値を減らします。

- Windows ファイルエクスプローラーを使用して、ディスク容量を消費しているファイルを見つけ、余分なファイルを転送または削除します。大量のディスク容量を消費しているアプリケーションまたはサービスの設定を変更します。

Applies to

この情報は、Windows バージョン 1.0.115 以降の Kinesis Agent に適用されます。

トラブルシューティングツール

設定ファイルの検証に加えて、ktdiag.exeツールを使用すると、Windows の Kinesis Agent を設定および使用する際の問題を診断および解決する複数の他の機能を利用できます。ktdiag.exe ツールは %PROGRAMFILES%\Amazon\AWSKinesisTap ディレクトリにあります。

- 特定のファイルパターンを持つログファイルがディレクトリに書き込まれる場合でも、Kinesis Agent for Windows によって処理されていないと考えられる場合は、/wスイッチを使用して、これらの変更が検出されていることを確認します。たとえば、*.log ファイル名のパターンを持つログファイルが c:\foo ディレクトリに書き込まれると予想しているとします。ktdiag.exe ツールを実行する際に、ディレクトリとファイルパターンを指定して、/w スイッチを使用します。

```
cd /D %PROGRAMFILES%\Amazon\AWSKinesisTap
ktdiag /w c:\foo *.log
```

ログファイルが書き込まれている場合は、次のような出力が表示されます。

```
Type any key to exit this program...
File: c:\foo\log1.log ChangeType: Created
File: c:\foo\log1.log ChangeType: Deleted
File: c:\foo\log1.log ChangeType: Created
File: c:\foo\log1.log ChangeType: Changed
File: c:\foo\log1.log ChangeType: Changed
File: c:\foo\log1.log ChangeType: Changed
File: c:\foo\log1.log ChangeType: Changed
```

このような出力が表示されない場合は、ログの書き込み時にアプリケーションまたはサービスの問題が生じているか、Kinesis Agent for Windows の問題ではなく、セキュリティ設定の問題が存在

します。このような出力が表示されているにもかかわらず、Windows 用 Kinesis Agent がログを処理していないように思われる場合は、」デスクトップまたはサーバーから予想される AWS のサービスにデータがストリーミングされない。

- ログがごくまれにしか書き込まれない場合でも、Windows の Kinesis エージェントが正しく機能していることを確認するのに役立ちます。たとえば、以下のように、/log4net スイッチを使用して、Log4net ライブラリを使用してログを書き込むアプリケーションをシミュレートします。

```
cd /D %PROGRAMFILES%\Amazon\AWSKinesisTap
KTDiag.exe /log4net c:\foo\log2.log
```

これによって、Log4net スタイルのログファイルが c:\foo\log2.log ログファイルに書き込まれ、キーが押されるまで新しいログエントリが追加され続きます。ファイル名の後に任意で指定される追加のスイッチを使用して、複数のオプションを設定できます。

ロック: -lm、-li、または -le

ログファイルがロックされる方法を制御する以下のロッキングスイッチのいずれかを指定できます。

-lm

最小範囲のロックがログファイルに使用され、ログファイルへの最大限のアクセスが可能になります。

-li

同じプロセス内のスレッドのみが同時にログにアクセスできます。

-le

一度に 1 つのスレッドのみがログにアクセスできます。これがデフォルト値です。

-tn:*milliseconds*

ログエントリの書き込み間の時間 (*milliseconds* の数字) を指定します。デフォルト値は 1000 ミリ秒 (1 秒) です。

-sm:*bytes*

各ログエントリの時間 (*bytes* の数字) を指定します。デフォルト値は 1000 バイトです。

-bk:*number*

一度に書き込みできるログエントリの数 (*number*) を指定します。デフォルトは 1 です。

- Windows イベントログに書き込むアプリケーションをシミュレートすると役に立つ場合があります。以下のように、/e スイッチを使用して、Windows イベントログにログエントリを書き込みます。

```
cd /D %PROGRAMFILES%\Amazon\AWSKinesisTap
KTDiag.exe /e Application
```

これによって、キーが押されるまで、ログエントリが Windows アプリケーションイベントログに書き込まれます。ログの名前の後に以下の追加オプションを任意で指定できます。

`-tn:`*`milliseconds`*

ログエントリの書き込み間の時間 (*`milliseconds`* の数字) を指定します。デフォルト値は 1000 ミリ秒 (1 秒) です。

`-sm:`*`bytes`*

各ログエントリの時間 (*`bytes`* の数字) を指定します。デフォルト値は 1000 バイトです。

`-bk:`*`number`*

一度に書き込みできるログエントリの数 (*`number`*) を指定します。デフォルトは 1 です。

Windows プラグイン用 Kinesis エージェントの作成

ほとんどの場合、Microsoft Windows 用 Amazon Kinesis エージェントプラグインを作成する必要はありません。Windows 用 Kinesis Agent は高度に設定可能で、`DirectorySource`および`KinesisStream`です。これはほとんどのシナリオで十分です。既存のソースとシンクの詳細については、「Microsoft Windows 用の Amazon Kinesis エージェントの設定」を参照してください。

異常なシナリオでは、カスタムプラグインを使って Kinesis Agent for Windows を拡張する必要があるかもしれません。ここで説明するシナリオは以下のとおりです。

- `Regex` または `Delimited` レコードパーサーを使用して複雑な `DirectorySource` 宣言をパッケージ化すると、さまざまな種類の設定ファイルに簡単に適用できます。
- ファイルベースではない、または既存のレコードパーサーが提供する解析機能を超える、新型のソースを作成します。
- 現在サポートされていない AWS のサービス用のシンクを作成します。

トピック

- Windows プラグイン用 Kinesis エージェントの使用開始
- Windows プラグインファクトリ用 Kinesis エージェントの実装
- Windows プラグインソース用の Kinesis エージェントの実装
- Windows プラグインシンク用の Kinesis エージェントの実装

Windows プラグイン用 Kinesis エージェントの使用開始

カスタムプラグインに関して特別なことは何もありません。既存のすべてのソースとシンクは、Kinesis Agent for Windows の起動時にカスタムプラグインがロードするのと同じメカニズムを使用し、それらは、`appsettings.json`設定ファイル。

Windows 用の Kinesis エージェントが起動すると、次のシーケンスが発生します。

1. Windows 用 Kinesis エージェントは、をインストールディレクトリ (`%PROGRAMFILES%\Amazon\AWSKinesisTap`を実装するクラスの`IFactory<T>`で定義された`Amazon.KinesisTap.CoreAssembly` このインターフェイスは、`Amazon.KinesisTap.Core\Infrastructure\IFactory.cs`Kinesis エージェント for Windows のソースコードを参照してください。

2. Windows 用 Kinesis エージェントは、これらのクラスを含むアセンブリをロードし、RegisterFactoryメソッドを呼び出します。
3. Windows 用 Kinesis エージェントは、appsettings.json設定ファイル。設定ファイルの各ソースとシンクについては、SourceType および SinkType キーと値のペアが検証されます。SourceType と SinkType のキーと値のペアの値と同じ名前で登録されているファクトリがある場合、それらのファクトリで CreateInstance メソッドが呼び出されます。CreateInstance メソッドには、設定およびその他の情報が IPluginContext オブジェクトとして渡されます。CreateInstance メソッドは、プラグインの設定と初期化を担当します。

プラグインが正しく動作するためには、プラグインを作成する登録済みファクトリクラスが必要で、プラグインクラス自体を定義する必要があります。

Windows 用の Kinesis エージェントのソースコードは、https://github.com/awslabs/kinesis-agent-windows。

Windows プラグインファクトリ用 Kinesis エージェントの実装

Windows 用の Kinesis エージェントプラグインファクトリを実装するには、次のステップに従います。

Windows 用 Kinesis エージェントプラグインファクトリを作成するには

1. .NET Framework 4.6 をターゲットにした C# ライブラリプロジェクトを作成します。
2. Amazon.KinesisTap.Core アセンブリにリファレンスを追加します。このアセンブリは、%PROGRAMFILES%\Amazon\AWSKinesisTapディレクトリに移動します。
3. NuGet を使用して、Microsoft.Extensions.Configuration.Abstractions パッケージをインストールします。
4. NuGet を使用して、System.Reactive パッケージをインストールします。
5. NuGet を使用して、Microsoft.Extensions.Logging パッケージをインストールします。
6. IFactory<IEventSource> ソース用またはシンク用 IFactory<IEventSink> のいずれかを実装するファクトリクラスを作成します。RegisterFactory メソッドおよび CreateInstance メソッドを追加します。

   たとえば、次のコードは、ランダムデータを生成するソースを作成する Kinesis Agent for Windows プラグインファクトリを作成します。

```
using System;
```

```
using Amazon.KinesisTap.Core;
using Microsoft.Extensions.Configuration;

namespace MyCompany.MySources
{
    public class RandomSourceFactory : IFactory<ISource>
    {
        public void RegisterFactory(IFactoryCatalog<ISource> catalog)
        {
            catalog.RegisterFactory("randomsource", this);
        }

        public ISource CreateInstance(string entry, IPlugInContext context)
        {
            IConfiguration config = context.Configuration;

            switch (entry.ToLower())
            {
                case "randomsource":
                    string rateString = config["Rate"];
                    string maxString = config["Max"];
                    TimeSpan rate;
                    int max;

                    if (string.IsNullOrWhiteSpace(rateString))
                    {
                        rate = TimeSpan.FromSeconds(30);
                    }
                    else
                    {
                        if (!TimeSpan.TryParse(rateString, out rate))
                        {
                            throw new Exception($"Rate {rateString} is invalid for
 RandomSource.");
                        }
                    }

                    if (string.IsNullOrWhiteSpace(maxString))
                    {
                        max = 1000;
                    }
                    else
                    {
                        if (!int.TryParse(maxString, out max))
```

```
                        {
                            throw new Exception($"Max {maxString} is invalid for
 RandomSource.");
                        }
                    }

                    return new RandomSource(rate, max, context);
                default:
                    throw new ArgumentException($"Source {entry} is not
 recognized.", entry);
            }
        }
    }
}
```

switch ステートメントは、最終的にファクトリを拡張してさまざまな種類のインスタンスを作成する場合に備えて、CreateInstance メソッドで使用されます。

何もしないシンクを作成するシンクファクトリを作成するには、次のようなクラスを使用します。

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Amazon.KinesisTap.Core;
using Microsoft.Extensions.Configuration;

namespace MyCompany.MySinks
{
    public class NullSinkFactory : IFactory<IEventSink>
    {
        public void RegisterFactory(IFactoryCatalog<IEventSink> catalog)
        {
            catalog.RegisterFactory("nullsink", this);
        }

        public IEventSink CreateInstance(string entry, IPlugInContext context)
        {
            IConfiguration config = context.Configuration;

            switch (entry.ToLower())
```

```
            {
                case "nullsink":
                    return new NullSink(context);
                default:
                    throw new Exception("Unrecognized sink type {entry}.");
            }
        }
    }
}
```

Windows プラグインソース用の Kinesis エージェントの実装

Windows 用 Kinesis エージェントプラグインソースを実装するには、次のステップに従います。

Windows 用 Kinesis エージェントプラグインソースを作成するには

1. IEventSource<out T> インターフェイスを実装するクラスをそのソース用に以前に作成したプロジェクトのソースを選択します。

   たとえば、ランダムデータを生成するソースを定義するには、次のコードを使用します。

```
using System;
using System.Reactive.Subjects;
using System.Timers;
using Amazon.KinesisTap.Core;
using Microsoft.Extensions.Logging;

namespace MyCompany.MySources
{
    public class RandomSource : EventSource<RandomData>, IDisposable
    {
        private TimeSpan _rate;
        private int _max;
        private Timer _timer = null;
        private Random _random = new Random();
        private ISubject<IEnvelope<RandomData>> _recordSubject = new
 Subject<IEnvelope<RandomData>>();



        public RandomSource(TimeSpan rate, int max, IPlugInContext context) :
 base(context)
```

```
        {
            _rate = rate;
            _max = max;
        }

        public override void Start()
        {
            try
            {
                CleanupTimer();
                _timer = new Timer(_rate.TotalMilliseconds);
                _timer.Elapsed += (Object source, ElapsedEventArgs args) =>
                {
                    var data = new RandomData()
                    {
                        RandomValue = _random.Next(_max)
                    };
                    _recordSubject.OnNext(new Envelope<RandomData>(data));
                };
                _timer.AutoReset = true;
                _timer.Enabled = true;
                _logger?.LogInformation($"Random source id {this.Id} started with
rate {_rate.TotalMilliseconds}.");
            }
            catch (Exception e)
            {
                _logger?.LogError($"Exception during start of RandomSource id
{this.Id}: {e}");
            }
        }

        public override void Stop()
        {
            try
            {
                CleanupTimer();
                _logger?.LogInformation($"Random source id {this.Id} stopped.");

            }
            catch (Exception e)
            {
                _logger?.LogError($"Exception during stop of RandomSource id
{this.Id}: {e}");
            }
```

```
        }

        private void CleanupTimer()
        {
            if (_timer != null)
            {
                _timer.Enabled = false;
                _timer?.Dispose();
                _timer = null;
            }
        }

        public override IDisposable Subscribe(IObserver<IEnvelope<RandomData>>
 observer)
        {
            return this._recordSubject.Subscribe(observer);
        }

        public void Dispose()
        {
            CleanupTimer();
        }
    }
}
```

この例では、RandomSource クラスは Id プロパティを提供するので EventSource<T> クラスから継承します。この例ではブックマークをサポートしていませんが、この基本クラスはその機能を実装するのにも役立ちます。エンベロープは、シンクへのストリーミング用にメタデータを格納し、任意のデータをラップする方法を提供します。RandomData クラスは次のステップで定義され、このソースからの出力オブジェクトの種類を表します。

2. ソースからストリーミングされたデータを含むクラスを以前に定義したプロジェクトに追加します。

   たとえば、ランダムデータのコンテナは次のように定義できます。

```
namespace MyCompany.MySources
{
    public class RandomData
    {
        public int RandomValue { get; set; }
    }
```

```
}
```

3. 以前に定義したプロジェクトをコンパイルします。
4. アセンブリを Kinesis Agent for Windows のインストールディレクトリにコピーします。
5. を作成または更新します。appsettings.json設定ファイルを作成して、それを Kinesis Agent for Windows のインストールディレクトリに配置します。
6. Windows 用の Kinesis エージェントを停止し、起動します。
7. 現在の Kinesis エージェント for Windows ログファイル (通常は%PROGRAMDATA%\Amazon\AWSKinesisTap\logsディレクトリにあります) を確認して、カスタムソースプラグインに問題がないことを確認します。
8. データが目的の AWS のサービスに到着していることを確認してください。

を拡張する方法の例については、DirectorySource特定の口グ形式の解析を実装するには、Amazon.KinesisTap.Uls\UlsSourceFactory.csおよびAmazon.KinesisTap.Uls\UlsLogParser.csKinesis エージェント for Windows のソースコードを参照してください。

ブックマーク機能を提供するソースを作成する方法の例については、Amazon.KinesisTap.Windows\WindowsSourceFactory.csおよびAmazon.KinesisTap.Windows\EventLogSource.csKinesis エージェント for Windows のソースコードを参照してください。

Windows プラグインシンク用の Kinesis エージェントの実装

Windows 用 Kinesis エージェントプラグインシンクを実装するには、次のステップに従います。

Windows 用 Kinesis エージェントプラグインシンクを作成するには

1. IEventSink インターフェイスを実装するクラスを以前に定義したプロジェクトに追加します。

   たとえば、次のコードは、レコードの到着をログに記録すること以外は何もしないシンクを実装し、その後レコードは破棄されます。

```
using Amazon.KinesisTap.Core;
using Microsoft.Extensions.Logging;

namespace MyCompany.MySinks
{
```

```
    public class NullSink : EventSink
    {
        public NullSink(IPlugInContext context) : base(context)
        {
        }

        public override void OnNext(IEnvelope envelope)
        {
            _logger.LogInformation($"Null sink {Id} received
 {GetRecord(envelope)}.");
        }

        public override void Start()
        {
            _logger.LogInformation($"Null sink {Id} starting.");
        }

        public override void Stop()
        {
            _logger.LogInformation($"Null sink {Id} stopped.");
        }
    }
}
```

この例では、NullSink シンククラスはレコードを JSON や XML などのさまざまなシリアル化形式に変換する機能を提供するため、EventSink クラスから継承します。

2. 以前に定義したプロジェクトをコンパイルします。
3. アセンブリを Kinesis Agent for Windows のインストールディレクトリにコピーします。
4. を作成または更新します。appsettings.json設定ファイルを作成して、それを Kinesis Agent for Windows のインストールディレクトリに配置します。たとえば、RandomSource および NullSink カスタムプラグインを使用するには、次の appsettings.json 設定ファイルを使用できます。

```
{
  "Sources": [
  {
 "Id": "MyRandomSource",
 "SourceType": "RandomSource",
 "Rate": "00:00:10",
 "Max": 50
  }
```

```
  ],
  "Sinks": [
  {
    "Id": "MyNullSink",
 "SinkType": "NullSink",
 "Format": "json"
  }
  ],
  "Pipes": [
    {
   "Id": "MyRandomToNullPipe",
   "SourceRef": "MyRandomSource",
   "SinkRef": "MyNullSink"
 }
  ]
}
```

この設定では、10 秒ごとに 0 ～ 50 の乱数に設定された RandomValue で RandomData のインスタンスを送信する送信元が作成されます。受信した RandomData インスタンスを JSON に変換し、その JSON をログに記録してからインスタンスを破棄するシンクを作成します。サンプルの設定ファイルを使用するには、サンプルファクトリ、RandomSource ソースクラス、および NullSink シンククラスの両方を、前に定義したプロジェクトに必ず含めてください。

5. Windows 用の Kinesis エージェントを停止し、起動します。
6. 現在の Kinesis エージェント for Windows ログファイル (通常は%PROGRAMDATA%\Amazon\AWSKinesisTap\logsディレクトリにあります) を確認して、カスタムシンクプラグインに問題がないことを確認します。
7. データが目的の AWS のサービスに到着していることを確認してください。例の NullSink は AWS のサービスにストリーミングしないため、レコードが受信されたことを示すログメッセージを探すことでシンクの正しいオペレーションを確認できます。

   たとえば、以下と同様の命名規則を使用できます。

```
2018-10-18 12:36:36.3647 Amazon.KinesisTap.Hosting.LogManager INFO Registered
 factory Amazon.KinesisTap.AWS.AWSEventSinkFactory.
2018-10-18 12:36:36.4018 Amazon.KinesisTap.Hosting.LogManager INFO Registered
 factory Amazon.KinesisTap.Windows.PerformanceCounterSinkFactory.
2018-10-18 12:36:36.4018 Amazon.KinesisTap.Hosting.LogManager INFO Registered
 factory MyCompany.MySinks.NullSinkFactory.
```

```
2018-10-18 12:36:36.6926 Amazon.KinesisTap.Hosting.LogManager INFO Registered
 factory Amazon.KinesisTap.Core.DirectorySourceFactory.
2018-10-18 12:36:36.6926 Amazon.KinesisTap.Hosting.LogManager INFO Registered
 factory Amazon.KinesisTap.ExchangeSource.ExchangeSourceFactory.
2018-10-18 12:36:36.6926 Amazon.KinesisTap.Hosting.LogManager INFO Registered
 factory Amazon.KinesisTap.Uls.UlsSourceFactory.
2018-10-18 12:36:36.6926 Amazon.KinesisTap.Hosting.LogManager INFO Registered
 factory Amazon.KinesisTap.Windows.WindowsSourceFactory.
2018-10-18 12:36:36.6926 Amazon.KinesisTap.Hosting.LogManager INFO Registered
 factory MyCompany.MySources.RandomSourceFactory.
2018-10-18 12:36:36.9601 Amazon.KinesisTap.Hosting.LogManager INFO Registered
 factory Amazon.KinesisTap.Core.Pipes.PipeFactory.
2018-10-18 12:36:37.4694 Amazon.KinesisTap.Hosting.LogManager INFO Registered
 factory Amazon.KinesisTap.AutoUpdate.AutoUpdateFactory.
2018-10-18 12:36:37.4807 Amazon.KinesisTap.Hosting.LogManager INFO Performance
 counter sink  started.
2018-10-18 12:36:37.6250 Amazon.KinesisTap.Hosting.LogManager INFO Null sink
 MyNullSink starting.
2018-10-18 12:36:37.6250 Amazon.KinesisTap.Hosting.LogManager INFO Connected source
 MyRandomSource to sink MyNullSink
2018-10-18 12:36:37.6333 Amazon.KinesisTap.Hosting.LogManager INFO Random source id
 MyRandomSource started with rate 10000.
2018-10-18 12:36:47.8084 Amazon.KinesisTap.Hosting.LogManager INFO Null sink
 MyNullSink received {"RandomValue":14}.
2018-10-18 12:36:57.6339 Amazon.KinesisTap.Hosting.LogManager INFO Null sink
 MyNullSink received {"RandomValue":5}.
2018-10-18 12:37:07.6490 Amazon.KinesisTap.Hosting.LogManager INFO Null sink
 MyNullSink received {"RandomValue":9}.
2018-10-18 12:37:17.6494 Amazon.KinesisTap.Hosting.LogManager INFO Null sink
 MyNullSink received {"RandomValue":47}.
2018-10-18 12:37:27.6520 Amazon.KinesisTap.Hosting.LogManager INFO Null sink
 MyNullSink received {"RandomValue":25}.
2018-10-18 12:37:37.6676 Amazon.KinesisTap.Hosting.LogManager INFO Null sink
 MyNullSink received {"RandomValue":21}.
2018-10-18 12:37:47.6688 Amazon.KinesisTap.Hosting.LogManager INFO Null sink
 MyNullSink received {"RandomValue":29}.
2018-10-18 12:37:57.6700 Amazon.KinesisTap.Hosting.LogManager INFO Null sink
 MyNullSink received {"RandomValue":22}.
2018-10-18 12:38:07.6838 Amazon.KinesisTap.Hosting.LogManager INFO Null sink
 MyNullSink received {"RandomValue":32}.
2018-10-18 12:38:17.6848 Amazon.KinesisTap.Hosting.LogManager INFO Null sink
 MyNullSink received {"RandomValue":12}.
2018-10-18 12:38:27.6866 Amazon.KinesisTap.Hosting.LogManager INFO Null sink
 MyNullSink received {"RandomValue":46}.
```

```
2018-10-18 12:38:37.6880 Amazon.KinesisTap.Hosting.LogManager INFO Null sink
 MyNullSink received {"RandomValue":48}.
2018-10-18 12:38:47.6893 Amazon.KinesisTap.Hosting.LogManager INFO Null sink
 MyNullSink received {"RandomValue":39}.
2018-10-18 12:38:57.6906 Amazon.KinesisTap.Hosting.LogManager INFO Null sink
 MyNullSink received {"RandomValue":18}.
2018-10-18 12:39:07.6995 Amazon.KinesisTap.Hosting.LogManager INFO Null sink
 MyNullSink received {"RandomValue":6}.
2018-10-18 12:39:17.7004 Amazon.KinesisTap.Hosting.LogManager INFO Null sink
 MyNullSink received {"RandomValue":0}.
2018-10-18 12:39:27.7021 Amazon.KinesisTap.Hosting.LogManager INFO Null sink
 MyNullSink received {"RandomValue":3}.
2018-10-18 12:39:37.7023 Amazon.KinesisTap.Hosting.LogManager INFO Null sink
 MyNullSink received {"RandomValue":19}.
```

AWS のサービスにアクセスするシンクを作成している場合は、役に立つと思われる基本クラスがあります。使用するシンクの場合`AWSBufferedEventSink`基本クラスについては、`Amazon.KinesisTap.AWS\CloudWatchLogsSink.cs`Windows 用 Kinesis エージェントのソースコードで。

Amazon Kinesis Agent for Microsoft Windows ユーザーガイドのドキュメント履歴

API バージョン: 2018 年 10-15 月

以下の表は、Amazon Kinesis エージェント Microsoft Windows ユーザーガイド(このドキュメント)。

update-history-change	update-history-description	update-history-date
ドキュメントの大幅な更新日	MSI のインストール手順を追加しました。ディレクトリソースの設定を更新し、WindowsEventLogPollingSource を追加しました。シンク設定で、ローカルファイルシステムの同期設定、ProfileRefreshingAwscredentialProvider、テキスト装飾に関する情報、シンク属性の変数の解決、シンクの STS リージョンエンドポイントの設定、VPC エンドポイントの設定、代替プロキシサーバーの設定が追加されました。パイプに、構成属性が追加されました。	2021年2月23日
ドキュメントを更新日	Amazon S3 の場所の指定では、大文字と小文字が区別されていることを示すために、トピックを更新しました。	2018 年 11 月 7 日
初期リリース、バージョン 1.0.0.115	『Windows 用 Kinesis エージェントユーザーガイド』の最初のリリースです。	2018 年 11 月 5 日

AWS の用語集

For the latest AWS terminology, see the AWS glossary in the AWS General Reference.

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。