AWS Ground Station エージェントユーザーガイド

AWS Ground Station

AWS Ground Station: AWS Ground Station エージェントユーザーガイド

Table of Contents

概要

AWS Ground Station エージェントとは

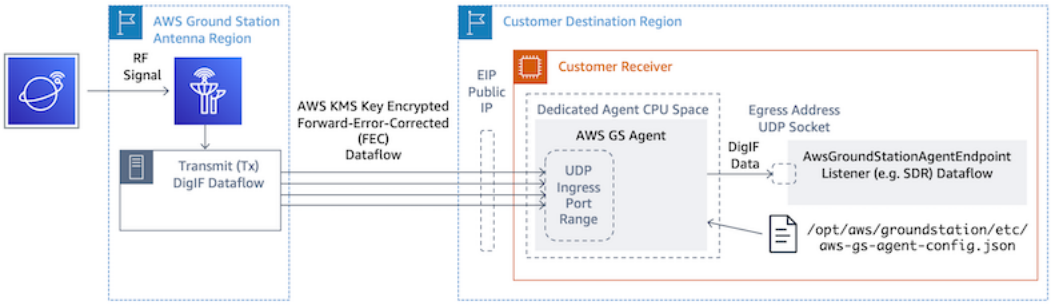
として利用可能な AWS Ground Station エージェントを使用するとRPM、AWSGround Station の問い合わせ中に同期広帯域デジタル中周波数 (DigIF) データフローを受信 (ダウンリンク) できます。データ配信には、次の 2 つのオプションを選択できます。

1. EC2 インスタンスへのデータ配信 - 所有しているEC2インスタンスへのデータ配信。 AWS Ground Station エージェントを管理します。このオプションは、ほぼリアルタイムのデータ処理が必要な場合に最適です。データ配信の詳細については、「Amazon Elastic Compute Cloud へのEC2データ配信」ガイドを参照してください。
2. S3 バケットへのデータ配信 - Ground Station AWS マネージドサービスを介して所有する S3 バケットへのデータ配信。S3 データ配信の詳細については、「入門 AWS Ground Stationガイド」を参照してください。

どちらのデータ配信モードでも、一連のAWSリソースを作成する必要があります。 CloudFormation を使用してAWSリソースを作成することは、信頼性、精度、サポート性を確保するために強くお勧めします。各問い合わせは、 EC2または S3 にのみデータを配信できますが、両方に同時に配信することはできません。

> **Note**
>
> S3 データ配信は Ground Station マネージドサービスであるため、このガイドではEC2インスタンス (複数可) へのデータ配信に焦点を当てています。

次の図は、Software-Defined Radio (SDR) または同様のリスナーを使用した AWS Ground Station アンテナリージョンからEC2インスタンスへの DigIF データフローを示しています。



AWS Ground Station エージェントの特徴

AWS Ground Station エージェントは、デジタル中間周波数 (DigIF) ダウンリンクデータを受信し、復号されたデータを出力して、以下を有効にします。

- DigIF ダウンリンク機能は 40 から 400 MHzの帯域幅MHzです。
- AWS ネットワーク上のパブリック IP (AWS Elastic IP) への高速で低ジッターの DigIF データ配信。
- Forward Error Correction () を使用した信頼性の高いデータ配信FEC。
- 暗号化にカスタマーマネージド AWS KMS キーを使用してデータ配信を保護します。

エージェントの要件

> **Note**
>
> この AWS Ground Station エージェントガイドは、AWS Ground Station 入門ガイドを使用して Ground Station にオンボーディングしたことを前提としています。

AWS Ground Station エージェントレシーバーEC2インスタンスには、DigIF データをエンドポイントに確実かつ安全に配信するための一連の依存AWSリソースが必要です。

1. EC2 レシーバーを起動する VPC 。
2. データ暗号化/復号のAWSKMSキー。
3. SSM Session Manager 用に設定されたSSHキーまたはEC2インスタンスプロファイル。
4. 以下のことを許可するネットワーク/セキュリティグループのルール:
    1. UDP データフローエンドポイントグループで指定されたポート AWS Ground Station 上のからのトラフィック。エージェントは、入力データフローエンドポイントにデータを配信するために使用される一連の連続したポートを予約します。
    2. SSH インスタンスへのアクセス (注: AWS Session Manager を使用してEC2インスタンスにアクセスすることもできます) 。
    3. エージェント管理用の、パブリックにアクセス可能な S3 バケットへの読み取りアクセス。
    4. SSL ポート 443 のトラフィックにより、エージェントは AWS Ground Station サービスと通信できます。
    5. AWS Ground Station マネージドプレフィックスリスト からのトラフィック`com.amazonaws.global.groundstation`。

さらに、パブリックサブネットを含むVPC設定が必要です。サブネット設定の背景については、VPC ユーザーガイドを参照してください。

互換性のある設定:

1. パブリックサブネット内のEC2インスタンスに関連付けられた Elastic IP。
2. インスタンスにアタッチされたパブリックサブネットENI内の に関連付けられた Elastic IP EC2 (パブリックサブネットと同じアベイラビリティーゾーン内の任意のサブネット内) 。

EC2 インスタンスと同じセキュリティグループを使用するか、少なくとも以下の最小ルールセットを持つセキュリティグループを指定できます。

- UDP データフローエンドポイントグループで指定されたポート AWS Ground Station 上の からのトラフィック。

これらのリソースが事前設定されたデータ配信テンプレートの例 AWS CloudFormation EC2については、AWS Ground Station 「エージェント (広帯域) を利用するパブリックブロードキャスト衛星」を参照してください。

VPC 図

図: パブリックサブネット内のEC2インスタンスに関連付けられた Elastic IP

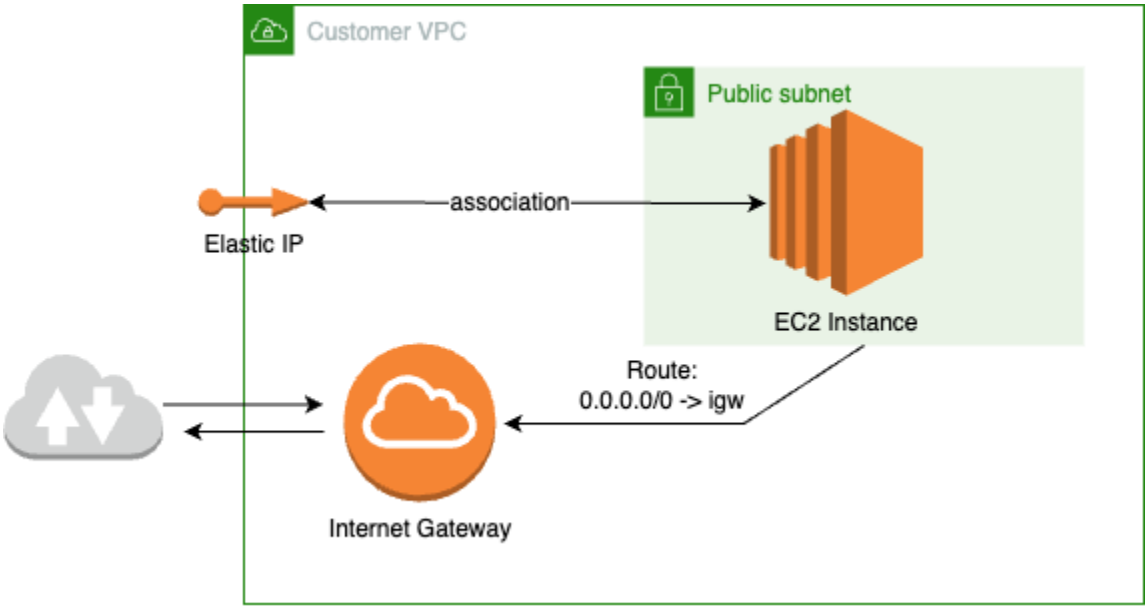
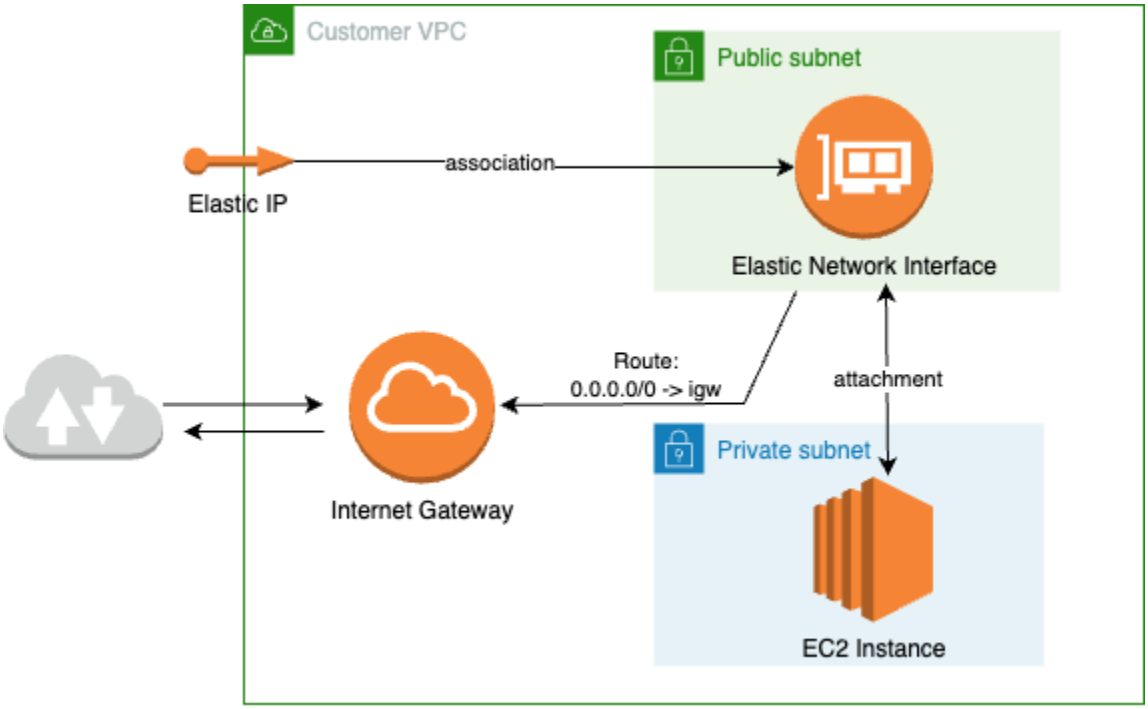


図: パブリックサブネットENI内の に関連付けられた Elastic IP。プライベートサブネット内のEC2インスタンスにアタッチされます。



サポートされるオペレーティングシステム

Amazon Linux 2 (5.10+ カーネル)

サポートされているインスタンスタイプは、「」に記載されています。 Amazon EC2インスタンスを選択し、アーキテクチャのCPUコアを予約する

AWS Ground Station エージェント経由でデータを受信する

以下の図は、広帯域デジタル中間周波数 (DigIF) コンタクト AWS Ground Station 中にデータがどのように流れるかの概要を示しています。

AWS Ground Station エージェントは、問い合わせのデータプレーンコンポーネントのオーケストレーションを処理します。コンタクトをスケジュールする前に、エージェントが正しく設定され、起動され、 に登録されている必要があります (エージェントの起動時に自動的に登録されます) AWS Ground Station。さらに、データ受信ソフトウェア (ソフトウェア定義ラジオなど) AwsGroundStationAgentEndpoint が実行され、 でデータを受信するように設定されている必要がありますegressAddress。

舞台裏では、 AWS Ground Station エージェントは Software Defined Radio (SDR) egressAddress がリッスンしている送信先エンドポイントに転送する前に、転送中に適用された AWS KMS 暗号化からタスクを受け取り、元 AWS Ground Station に戻します。 AWS Ground Station エージェントとその基盤となるコンポーネントは、設定ファイルで設定されたCPU境界を尊重し、インスタンスで実行されている他のアプリケーションのパフォーマンスに影響を与えないようにします。

エージェントは、受信者インスタンスで AWS Ground Station 実行され、問い合わせに関与している必要があります。1 つのレシーバーインスタンスですべてのデータフローを受信する場合は、1 つの AWS Ground Station エージェントで、以下に示すように複数のデータフローをオーケストレーションできます。

複数のデータフロー、単一のレシーバー

シナリオの例:

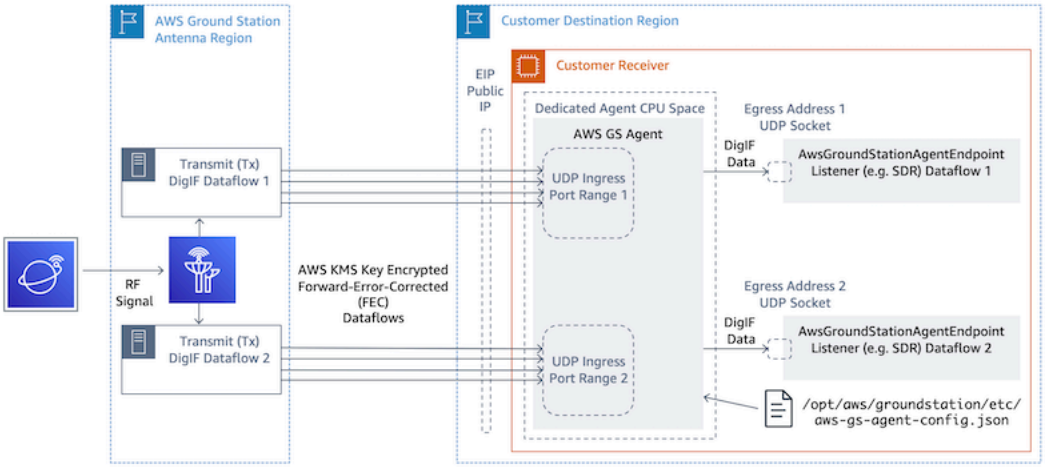
同じEC2レシーバインスタンスで、DigIF データフローとして 2 つのアンテナダウンリンクを受信したいと考えています。2 つのダウンリンクは 200MHz と 100 になりますMHz。

AwsGroundStationAgentEndpoints:

各データフローに 1 つずつ、合計 2 つの `AwsGroundStationAgentEndpoint` リソースがあります。両方のエンドポイントには同じパブリック IP アドレス (`ingressAddress.socketAddress.name`) が割り当てられます。データフローは同じEC2インスタンスで受信されているため、進入`portRange`の は重複しないでください。両方の `egressAddress.socketAddress.port`は一意である必要があります。

CPU 計画：

- インスタンスで単一の AWS Ground Station エージェントを実行するための 1 コア (2 v CPU) 。
- DigIF Dataflow 1 (CPU コアプランニングテーブルで 200MHz ルックアップCPU) を受信する 6 コア (12 v)。
- DigIF Dataflow 2 (CPU コアプランニングテーブルで 100MHz ルックアップCPU) を受信する 4 コア (8 v)。
- 専有エージェントCPUスペースの合計 = 同じソケットで 11 コア (22 v CPU) 。



複数のデータフロー、複数のレシーバー

シナリオの例:

異なるEC2レシーバインスタンスで、DigIF データフローとして 2 つのアンテナダウンリンクを受信したいと考えています。どちらのダウンリンクも 400 になりますMHz。

AwsGroundStationAgentEndpoints:

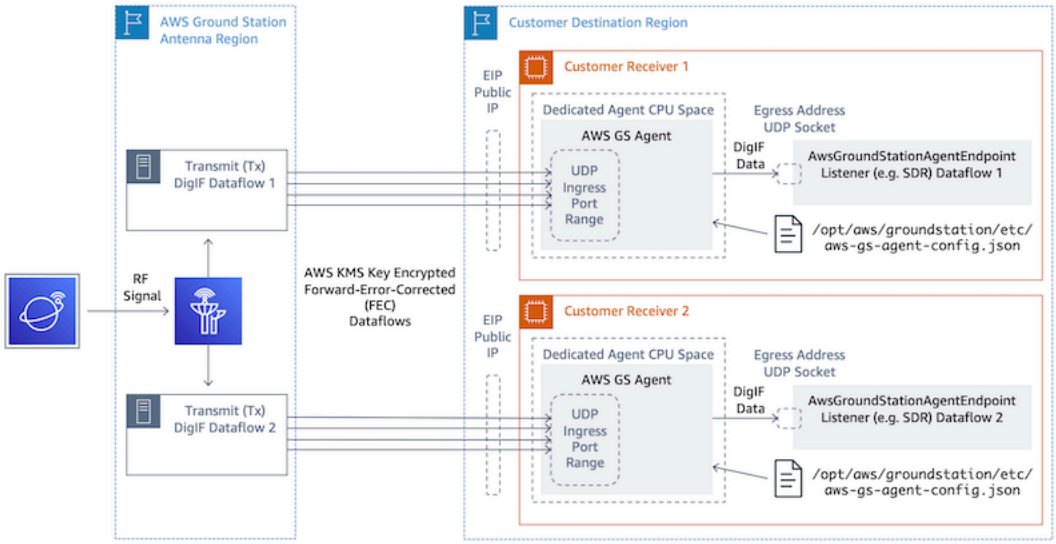
各データフローに 1 つずつ、合計 2 つの `AwsGroundStationAgentEndpoint` リソースがあります。エンドポイントには異なるパブリック IP アドレス (`ingressAddress.socketAddress.name`) が割り当てられます。データフローは別のインフラストラクチャで受信され、互いに競合しないため、`ingressAddress` または `egressAddress` のいずれのポート値にも制限はありません。

CPU 計画 :

- レシーバーインスタンス 1

- インスタンスで単一の AWS Ground Station エージェントを実行するための 1 コア (2 v CPU) 。
- DigIF Dataflow 1 (CPU コアプランニングテーブルで 400MHz ルックアップCPU) を受信する 9 コア (18 v)。
- 専有エージェントCPUスペースの合計 = 同じソケットに 10 コア (20 v CPU) 。

- レシーバーインスタンス 2
  - インスタンスで単一の AWS Ground Station エージェントを実行するための 1 コア (2 v CPU) 。
  - DigIF Dataflow 2 (CPU コアプランニングテーブルで 400MHz ルックアップCPU) を受信する 9 コア (18 v)。
  - 専有エージェントCPUスペースの合計 = 同じソケットに 10 コア (20 v CPU) 。



Amazon EC2インスタンスを選択し、アーキテクチャのCPUコアを予約する

サポートされている Amazon EC2インスタンスタイプ

AWS Ground Station エージェントは、コンピューティング負荷の高いデータ配信ワークフローのために、専用CPUコアを運用する必要があります。次のインスタンスタイプがサポートされています。どのインスタンスタイプがお客様のユースケースに最も適しているかを判断するには、「CPUコアプランニング」を参照してください。

インスタンスタイプ	デフォルト vCPUs	デフォルトのCPUコア
c5.12xlarge	48	24
c5.18xlarge	72	36
c5.24xlarge	96	48
c5n.18xlarge	72	36
c5n.metal	72	36
c6i.32xlarge	128	64
g4dn.12xlarge	48	24
g4dn.16xlarge	64	32
g4dn.metal	96	48
m5.12xlarge	48	24
m5.24xlarge	96	48
m6i.32xlarge	128	64
p3dn.24xlarge	96	48
p4d.24xlarge	96	48

インスタンスタイプ	デフォルト vCPUs	デフォルトのCPUコア
r5.24xlarge	96	48
r5.metal	96	48
r5n.24xlarge	96	48
r5n.metal	96	48
r6i.32xlarge	128	64

CPU コアプランニング

AWS Ground Station エージェントには、データフローごとに L3 キャッシュを共有する専用プロセッサコアが必要です。エージェントは、ハイパースレッド (HT) CPUペアを活用するように設計されており、その使用のために HT ペアを予約する必要があります。ハイパースレッドペアは、単一のコア内に含まれる仮想 CPUs (v CPU) のペアです。次の表は、データフローデータレートと、1 つのデータフロー用にエージェント用に予約された必要なコア数とのマッピングを示しています。このテーブルは Cascade Lake 以降を前提CPUsとしており、サポートされているすべてのインスタンスタイプで有効です。帯域幅がテーブル内のエントリの間にある場合は、次に高いものを選択します。

エージェントは管理と調整のために追加の予約済みコアを必要とするため、必要なコアの合計は、各データフローに必要なコアの合計 (次のチャートから) と 1 つの追加コア (2 vCPUs) の合計になります。

AntennaDownlink 帯域幅 (MHz)	予想される VITA-49.2 DigIF データレート (Mb/秒)	コア数 (HT CPUペア)	合計 vCPU
50	1,000	3	6
100	2000	4	8
150	3000	5	10
200	4000	6	12
250	5000	6	12

AntennaDownlink 帯域幅 (MHz)	予想される VITA-49.2 DigIF データレート (Mb/秒)	コア数 (HT CPUペア)	合計 vCPU
300	6000	7	14
350	7000	8	16
400	8000	9	18

アーキテクチャ情報の収集

`lscpu` は、システムのアーキテクチャに関する情報を提供します。基本的な出力は、どのノードvCPUs（および各NUMAノードが L3 キャッシュNUMAを共有する) に属しているか (CPU「」とラベル付け) を示します。次に、 AWS Ground Station エージェントを設定するために必要な情報を収集するために、`c5.24xlarge`インスタンスを調べます。これには、 の数vCPUs、コア、 vCPU-to-node関連付けなどの有用な情報が含まれます。

```
> lscpu
Architecture: x86_64
CPU op-mode(s): 32-bit, 64-bit
Byte Order: Little Endian
CPU(s): 96
On-line CPU(s) list: 0-95
Thread(s) per core: 2             <------
Core(s) per socket: 24
Socket(s): 2
NUMA node(s): 2
Vendor ID: GenuineIntel
CPU family: 6
Model: 85
Model name: Intel(R) Xeon(R) Platinum 8275CL CPU @ 3.00GHz
Stepping: 7
CPU MHz: 3601.704
BogoMIPS: 6000.01
Hypervisor vendor: KVM
Virtualization type: full
L1d cache: 32K
L1i cache: 32K
L2 cache: 1024K
```

```
L3 cache: 36608K
NUMA node0 CPU(s): 0-23,48-71     <------
NUMA node1 CPU(s): 24-47,72-95    <------
```

AWS Ground Station エージェント専用のコアには、割り当てられたコア vCPUs ごとに両方を含める必要があります。データフローのすべてのコアは、同じNUMAノードに存在する必要があります。`lscpu` コマンド`-p`のオプションにより、エージェントの設定に必要なCPU関連付けへのコアが提供されます。関連するフィールドは、 CPU (v と呼ばれるCPU) 、 Core、L3 (そのコアによって共有される L3 キャッシュを示す) です。ほとんどの Intel プロセッサーでは、NUMAノードは L3 キャッシュと等しいことに注意してください。

出力の次のサブセットを `c5.24xlarge` (わかりやすくするために省略およびフォーマット) `lscpu -p`にすることを検討してください。

```
CPU,Core,Socket,Node,,L1d,L1i,L2,L3
0   0    0      0     0   0   0  0
1   1    0      0     1   1   1  0
2   2    0      0     2   2   2  0
3   3    0      0     3   3   3  0
...
16  0    0      0     0   0   0  0
17  1    0      0     1   1   1  0
18  2    0      0     2   2   2  0
19  3    0      0     3   3   3  0
```

出力から、コア 0 には vCPUs 0 と 16、コア 1 には vCPUs 1 と 17、コア 2 には vCPUs 2 と 18 が含まれていることがわかります。つまり、ハイパースレッドペアは 0 と 16、1 と 17、2 と 18 です。

CPU 割り当ての例

例として、350 のデュアル極性ワイドバンドダウンリンクに`c5.24xlarge`インスタンスを使用しますMHz。の表からCPU コアプランニング、350 MHzダウンリンクには 1 つのデータフローに 8 コア (16 vCPUs) が必要であることがわかります。つまり、2 つのデータフローを使用するこのデュアル極性設定では、エージェントに合計 16 コア (32 vCPUs) と 1 コア (2 vCPUs) が必要です。

のlscpu出力には NUMA node0 CPU(s): 0-23,48-71と c5.24xlargeが含まれていることがわかっていますNUMA node1 CPU(s): 24-47,72-95。NUMA node0 には必要以上のものがあるため、コアからのみ割り当てます: 0 ~ 23 および 48 ~ 71。

まず、L3 キャッシュまたはNUMAノードを共有するデータフローごとに 8 つのコアを選択します。次に、 のlscpu -p出力で対応する vCPUs（CPU「」というラベル) を検索します付録: c5.24xlarge のlscpu -p出力 (フル)。コア選択プロセスの例は次のようになります。

- OS 用にコア 0 ~ 1 を予約します。
- フロー 1: 2 ~ 9 および 50 ~ 57 にマッピングされるコア vCPUs 2 ~ 9 を選択します。
- フロー 2: 10 ~ 17 および 58 ~ 65 にマッピングされるコア vCPUs 10 ~ 17 を選択します。
- エージェントコア: 18 と 66 にマッピングされるコア vCPUs 18 を選択します。

これにより vCPUs 2 ~ 18 と 50 ~ 66 になるため、エージェントを指定するリストは です[2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66]。で説明されているCPUsように、独自のプロセスがこれらのプロセスで実行されていないことを確認する必要があります AWS Ground Station エージェントと一緒にサービスとプロセスを実行する。

この例で選択した特定のコアは、ある程度任意であることに注意してください。他のコアセットは、データフローごとに L3 キャッシュを共有するすべての要件を満たしている限り機能します。

付録: c5.24xlarge のlscpu -p出力 (フル)

```
> lscpu -p
# The following is the parsable format, which can be fed to other
# programs. Each different item in every column has an unique ID
# starting from zero.
# CPU,Core,Socket,Node,,L1d,L1i,L2,L3
0,0,0,0,,0,0,0,0
1,1,0,0,,1,1,1,0
2,2,0,0,,2,2,2,0
3,3,0,0,,3,3,3,0
4,4,0,0,,4,4,4,0
5,5,0,0,,5,5,5,0
6,6,0,0,,6,6,6,0
7,7,0,0,,7,7,7,0
8,8,0,0,,8,8,8,0
```

```
9,9,0,0,,9,9,9,0
10,10,0,0,,10,10,10,0
11,11,0,0,,11,11,11,0
12,12,0,0,,12,12,12,0
13,13,0,0,,13,13,13,0
14,14,0,0,,14,14,14,0
15,15,0,0,,15,15,15,0
16,16,0,0,,16,16,16,0
17,17,0,0,,17,17,17,0
18,18,0,0,,18,18,18,0
19,19,0,0,,19,19,19,0
20,20,0,0,,20,20,20,0
21,21,0,0,,21,21,21,0
22,22,0,0,,22,22,22,0
23,23,0,0,,23,23,23,0
24,24,1,1,,24,24,24,1
25,25,1,1,,25,25,25,1
26,26,1,1,,26,26,26,1
27,27,1,1,,27,27,27,1
28,28,1,1,,28,28,28,1
29,29,1,1,,29,29,29,1
30,30,1,1,,30,30,30,1
31,31,1,1,,31,31,31,1
32,32,1,1,,32,32,32,1
33,33,1,1,,33,33,33,1
34,34,1,1,,34,34,34,1
35,35,1,1,,35,35,35,1
36,36,1,1,,36,36,36,1
37,37,1,1,,37,37,37,1
38,38,1,1,,38,38,38,1
39,39,1,1,,39,39,39,1
40,40,1,1,,40,40,40,1
41,41,1,1,,41,41,41,1
42,42,1,1,,42,42,42,1
43,43,1,1,,43,43,43,1
44,44,1,1,,44,44,44,1
45,45,1,1,,45,45,45,1
46,46,1,1,,46,46,46,1
47,47,1,1,,47,47,47,1
48,0,0,0,,0,0,0,0
49,1,0,0,,1,1,1,0
50,2,0,0,,2,2,2,0
51,3,0,0,,3,3,3,0
52,4,0,0,,4,4,4,0
```

```
53,5,0,0,,5,5,5,0
54,6,0,0,,6,6,6,0
55,7,0,0,,7,7,7,0
56,8,0,0,,8,8,8,0
57,9,0,0,,9,9,9,0
58,10,0,0,,10,10,10,0
59,11,0,0,,11,11,11,0
60,12,0,0,,12,12,12,0
61,13,0,0,,13,13,13,0
62,14,0,0,,14,14,14,0
63,15,0,0,,15,15,15,0
64,16,0,0,,16,16,16,0
65,17,0,0,,17,17,17,0
66,18,0,0,,18,18,18,0
67,19,0,0,,19,19,19,0
68,20,0,0,,20,20,20,0
69,21,0,0,,21,21,21,0
70,22,0,0,,22,22,22,0
71,23,0,0,,23,23,23,0
72,24,1,1,,24,24,24,1
73,25,1,1,,25,25,25,1
74,26,1,1,,26,26,26,1
75,27,1,1,,27,27,27,1
76,28,1,1,,28,28,28,1
77,29,1,1,,29,29,29,1
78,30,1,1,,30,30,30,1
79,31,1,1,,31,31,31,1
80,32,1,1,,32,32,32,1
81,33,1,1,,33,33,33,1
82,34,1,1,,34,34,34,1
83,35,1,1,,35,35,35,1
84,36,1,1,,36,36,36,1
85,37,1,1,,37,37,37,1
86,38,1,1,,38,38,38,1
87,39,1,1,,39,39,39,1
88,40,1,1,,40,40,40,1
89,41,1,1,,41,41,41,1
90,42,1,1,,42,42,42,1
91,43,1,1,,43,43,43,1
92,44,1,1,,44,44,44,1
93,45,1,1,,45,45,45,1
94,46,1,1,,46,46,46,1
95,47,1,1,,47,47,47,1
```

エージェントのインストール

AWS Ground Station エージェントは、次の方法でインストールできます。

1. AWS CloudFormation テンプレート (推奨) 。
2. Amazon への手動インストールEC2。

AWS CloudFormation テンプレートを使用する

EC2 データ配信 AWS CloudFormation テンプレートは、データをEC2インスタンスに配信するために必要なAWSリソースを作成します。この AWS CloudFormation テンプレートは、 AWS Ground Station エージェントがプリインストールされている AWS Ground Station マネージドAMI型 を使用します。次に、作成されたEC2インスタンスのブートスクリプトがエージェント設定ファイルに入力され、必要なパフォーマンス調整 () が適用されますインスタンスのパフォーマンスEC2を調整する。

ステップ 1: AWS リソースを作成する

AWS Ground Station Agent (ワイドバンド) を使用して、 テンプレートパブリックブロードキャスト衛星を使用してAWSリソーススタックを作成します。

ステップ 2: エージェントのステータスを確認する

デフォルトでは、エージェントは設定され、アクティブ (開始) になります。エージェントのステータスを確認するには、EC2インスタンス (SSH または SSM Session Manager) に接続し、「」を参照してくださいAWS Ground Station エージェントのステータス。

に手動でインストールする EC2

Ground Station では CloudFormation 、テンプレートを使用してAWSリソースをプロビジョニングすることを推奨していますが、標準テンプレートでは不十分なユースケースがある場合があります。このような場合は、ニーズに合わせてテンプレートをカスタマイズすることをお勧めします。それでも要件が満たされない場合は、AWSリソースを手動で作成し、エージェントをインストールできます。

ステップ 1: AWSリソースを作成する

問い合わせに必要なAWSリソースを手動で設定する手順については、「ミッションプロファイル設定の例」を参照してください。

AwsGroundStationAgentEndpoint リソースは、 AWS Ground Station エージェント経由で DigIF データフローを受信するためのエンドポイントを定義し、正常な問い合わせを行う上で不可欠です。APIドキュメントはAPIリファレンス にありますが、このセクションでは AWS Ground Station エージェントに関連する概念について簡単に説明します。

エンドポイント`ingressAddress`は、 AWS Ground Station エージェントがアンテナから AWS KMS 暗号化されたUDPトラフィックを受信する場所です。`socketAddress` name は、EC2インスタンスのパブリック IP です (添付の からEIP) 。`portRange` は、他の使用時間から予約されている範囲内に 300 個以上の連続したポートである必要があります。手順については、「進入ポートの予約 - ネットワークに影響します」を参照してください。これらのポートは、レシーバーインスタンスが実行されVPCている のセキュリティグループへのUDP進入トラフィックを許可するように設定する必要があります。

エンドポイント`egressAddress`は、エージェントが DigIF データフローをユーザーに渡す場所です。この場所にあるUDPソケット経由でデータを受信するアプリケーション (例: SDR) が必要です。

ステップ 2: EC2インスタンスを作成する

次の AMIs がサポートされています。

1. AWS Ground Station AMI - `groundstation-al2-gs-agent-ami-*` ここで、`*` は が構築AMIされた日付 - エージェントがインストールされています (推奨) 。
2. `amzn2-ami-kernel-5.10-hvm-x86_64-gp2`.

ステップ 2: エージェントをダウンロードしてインストールする

> **Note**
>
> 前のステップで AWS Ground Station エージェントを選択しなかった場合は、このセクションAMIのステップを完了する必要があります。

エージェントをダウンロードする

AWS Ground Station エージェントはリージョン固有の S3 バケットから利用でき、${AWS::Region} `s3://groundstation-wb-digif-software-${AWS::Region}/aws-groundstation-agent/latest/amazon_linux_2_x86_64/aws-groundstation-agent.rpm`がサポートされている AWS Ground Station コンソールとデータ配信リージョンの 1 つを参照するAWSコマンドライン (CLI) を使用してサポートEC2インスタンスにダウンロードできます。

例: AWSリージョン us-east-2 から /tmp フォルダに最新の rpm バージョンをローカルにダウンロードします。

```
aws s3 --region us-east-2 cp s3://groundstation-wb-digif-software-us-east-2/aws-groundstation-agent/latest/amazon_linux_2_x86_64/aws-groundstation-agent.rpm /tmp
```

特定のバージョンの AWS Ground Station エージェントをダウンロードする必要がある場合は、S3 バケット内のバージョン固有のフォルダからダウンロードできます。

例: rpm のバージョン 1.0.2716.0 をAWSリージョン us-east-2 からローカルに /tmp フォルダにダウンロードします。

```
aws s3 --region us-east-2 cp s3://groundstation-wb-digif-software-us-east-2/aws-groundstation-agent/1.0.2716.0/amazon_linux_2_x86_64/aws-groundstation-agent.rpm /tmp
```

> **Note**
>
> ダウンロードRPMした が によって提供されたことを確認する場合は AWS Ground Station、「」の手順に従いますRPM インストールの検証。

エージェントをインストールする

```
sudo yum install ${MY_RPM_FILE_PATH}

Example: Assumes agent is in the "/tmp" directory
```

```
sudo yum install /tmp/aws-groundstation-agent.rpm
```

ステップ 4: エージェントを設定する

エージェントをインストールしたら、エージェント設定ファイルを更新する必要があります。「エージェントを設定する」を参照してください。

ステップ 5: パフォーマンスチューニングを適用する

AWS Ground Station エージェント AMI: 前のステップAMIで AWS Ground Station エージェントを選択した場合は、次のパフォーマンス調整を適用します。

- ハードウェアの中断と受信キューの調整 - 影響CPUとネットワーク
- 進入ポートの予約 - ネットワークに影響します
- 再起動

その他AMIs: 前のステップAMIで他の を選択した場合は、 および にリストされているすべてのチューニングを適用インスタンスのパフォーマンスEC2を調整するし、インスタンスを再起動します。

ステップ 6: エージェントを管理する

エージェントの起動、停止、およびステータスの確認については、「エージェントを管理する」を参照してください。

エージェントを管理する

AWS Ground Station エージェントには、組み込み Linux コマンドツールを使用してエージェントを設定、開始、停止、アップグレード、ダウングレード、アンインストールするための以下の機能があります。

トピック

AWS Ground Station エージェント設定

に移動します。これには`/opt/aws/groundstation/etc`、 aws-gs-agent-config.json という名前の単一のファイルが含まれている必要があります。「エージェント設定ファイル」を参照してください。

AWS Ground Station エージェント起動

```
#start
sudo systemctl start aws-groundstation-agent

#check status
systemctl status aws-groundstation-agent
```

エージェントがアクティブであることを示す出力を生成する必要があります。

```
aws-groundstation-agent.service - aws-groundstation-agent
Loaded: loaded (/usr/lib/systemd/system/aws-groundstation-agent.service; enabled;
 vendor preset: disabled)
Active: active (running) since Tue 2023-03-14 00:39:08 UTC; 1 day 13h ago
Docs: https://aws.amazon.com/ground-station/
Main PID: 8811 (aws-gs-agent)
CGroup: /system.slice/aws-groundstation-agent.service
##8811 /opt/aws/groundstation/bin/aws-gs-agent production
```

AWS Ground Station エージェント停止

```
#stop
sudo systemctl stop aws-groundstation-agent

#check status
systemctl status aws-groundstation-agent
```

エージェントが非アクティブ (停止中) であることを示す出力を生成する必要があります。

```
aws-groundstation-agent.service - aws-groundstation-agent
Loaded: loaded (/usr/lib/systemd/system/aws-groundstation-agent.service; enabled;
 vendor preset: disabled)
Active: inactive (dead) since Thu 2023-03-09 15:35:08 UTC; 6min ago
Docs: https://aws.amazon.com/ground-station/
Process: 84182 ExecStart=/opt/aws/groundstation/bin/launch-aws-gs-agent (code=exited,
 status=0/SUCCESS)
Main PID: 84182 (code=exited, status=0/SUCCESS)
```

AWS Ground Station エージェントのアップグレード

1. エージェントの最新バージョンをダウンロードします。「エージェントをダウンロードする」を参照してください。
2. エージェントを停止します。

```
#stop
sudo systemctl stop aws-groundstation-agent

#confirm inactive (stopped) state
systemctl status aws-groundstation-agent
```

3. エージェントを更新します。

```
sudo yum update ${MY_RPM_FILE_PATH}

# check the new version has been installed correctly by comparing the agent version
 with the starting agent version
yum info aws-groundstation-agent

# reload the systemd configuration
sudo systemctl daemon-reload

# restart the agent
sudo systemctl restart aws-groundstation-agent

# check agent status
systemctl status aws-groundstation-agent
```

AWS Ground Station エージェントダウングレード

1. 必要なエージェントバージョンをダウンロードします。「エージェントをダウンロードする」を参照してください。
2. エージェントをダウングレードします。

```
# get the starting agent version
yum info aws-groundstation-agent

# stop the agent service
sudo systemctl stop aws-groundstation-agent
```

```
# downgrade the rpm
sudo yum downgrade ${MY_RPM_FILE_PATH}

# check the new version has been installed correctly by comparing the agent version
 with the starting agent version
yum info aws-groundstation-agent

# reload the systemd configuration
sudo systemctl daemon-reload

# restart the agent
sudo systemctl restart aws-groundstation-agent

# check agent status
systemctl status aws-groundstation-agent
```

AWS Ground Station エージェントアンインストール

エージェントをアンインストールすると、rename /opt/aws/groundstation/etc/aws-gs-agent-config.json to /opt/aws/groundstation/etc/aws-gs-agent-config.json.rpmsave になります。エージェントを同じインスタンスに再度インストールすると、 aws-gs-agent-config.json のデフォルト値が書き込まれ、AWSリソースに対応する正しい値で更新する必要があります。「エージェント設定ファイル」を参照してください。

```
sudo yum remove aws-groundstation-agent
```

AWS Ground Station エージェントのステータス

エージェントのステータスは、アクティブ (エージェントが実行中) か、非アクティブ (エージェントが停止中) のいずれかです。

```
systemctl status aws-groundstation-agent
```

出力例には、エージェントがインストール済み、非アクティブ (停止中)、有効 (起動時にサービスを開始) のステータスが表示されます。

```
aws-groundstation-agent.service - aws-groundstation-agent
Loaded: loaded (/usr/lib/systemd/system/aws-groundstation-agent.service; enabled;
 vendor preset: disabled)
Active: inactive (dead) since Thu 2023-03-09 15:35:08 UTC; 6min ago
Docs: https://aws.amazon.com/ground-station/
Process: 84182 ExecStart=/opt/aws/groundstation/bin/launch-aws-gs-agent (code=exited,
 status=0/SUCCESS)
Main PID: 84182 (code=exited, status=0/SUCCESS)
```

AWS Ground Station エージェントRPM情報

```
yum info aws-groundstation-agent
```

出力は次のとおりです。

> **Note**
>
> 「バージョン」は、エージェントが公開している最新のバージョンによって異なる場合があります。

```
Loaded plugins: extras_suggestions, langpacks, priorities, update-motd
Installed Packages
Name        : aws-groundstation-agent
Arch        : x86_64
Version     : 1.0.2677.0
Release     : 1
Size        : 51 M
Repo        : installed
Summary     : Client software for AWS Ground Station
URL         : https://aws.amazon.com/ground-station/
License     : Proprietary
```

```
Description : This package provides client applications for use with AWS Ground Station
```

エージェントを設定する

エージェントをインストールしたら、/opt/aws/groundstation/etc/aws-gs-agent-config.json でエージェント設定ファイルを更新する必要があります。

エージェント設定ファイル

例

```
{
  "capabilities": [
    "arn:aws:groundstation:eu-central-1:123456789012:dataflow-endpoint-group/
bb6c19ea-1517-47d3-99fa-3760f078f100"
  ],
  "device": {
  "privateIps": [
    "127.0.0.1"
  ],
  "publicIps": [
    "1.2.3.4"
  ],
  "agentCpuCores":
 [ 24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,39,40,41,42,43,44,72,73,74,75,76,77,78,79,80,81
}
```

フィールドの内訳

機能

機能は、データフローエンドポイントグループの Amazon リソースネームとして指定されます。

必須: True

形式: 文字列配列

- 値: 機能 ARNs → 文字列

例:

```
"capabilities": [
    "arn:aws:groundstation:${AWS::Region}:${AWS::AccountId}:dataflow-endpoint-group/
${DataflowEndpointGroupId}"
]
```

デバイス

このフィールドには、現在のEC2「デバイス」を列挙するために必要な追加のフィールドが含まれます。

必須: True

フォーマット: オブジェクト

メンバー:

- privateIps
- publicIps
- agentCpuCores
- networkAdapters

privateIps

このフィールドは現在使用されていませんが、今後のユースケースに備えて含まれています。値が含まれていない場合、デフォルトで [“127.0.0.1”] になります。

必須: False

形式: 文字列配列

- 値: IP アドレス → 文字列

例:

```
"privateIps": [
    "127.0.0.1"
```

```
],
```

publicIps

データフローエンドポイントグループあたりの Elastic IP (EIP) 。

必須: True

形式: 文字列配列

- 値: IP アドレス → 文字列

例:

```
"publicIps": [
    "9.8.7.6"
],
```

agentCPUCores

これにより、プロセス用に aws-gs-agent予約される仮想コアを指定します。この値を適切に設定するための要件については、「CPU コアプランニング」を参照してください。

必須: True

形式: 整数配列

- 値: コア数 → 整数

例:

```
"agentCpuCores": [

 24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,39,40,41,42,43,44,72,73,74,75,76,77,78,79,80,81,8
]
```

networkAdapters

これは、データを受信するイーサネットアダプター、または にアタッチENIsされたインターフェイスに対応します。

必須: False

形式: 文字列配列

- 値: イーサネットアダプタの名前 (`ifconfig` を実行すると検索できます)

例:

```
"networkAdapters": [
    "eth0"
]
```

インスタンスのパフォーマンスEC2を調整する

> **Note**
>
> CloudFormation テンプレートを使用してAWSリソースをプロビジョニングした場合、これらの調整は自動的に適用されます。を使用した場合、AMIまたはEC2インスタンスを手動で作成した場合は、これらのパフォーマンス調整を適用して、最も信頼性の高いパフォーマンスを実現する必要があります。

チューニングを適用した後は、必ずインスタンスを再起動してください。

トピック

- ハードウェアの中断と受信キューの調整 - 影響CPUとネットワーク
- Rx 割り込みの合体を調整する - ネットワークに影響します
- Rx リングバッファの調整 - ネットワークに影響
- C-State CPU の調整 - 影響 CPU
- 進入ポートの予約 - ネットワークに影響します
- 再起動

ハードウェアの中断と受信キューの調整 - 影響CPUとネットワーク

このセクションでは、systemd、、Receive Packet Steering (RPS) SMPIRQs、Receive Flow Steering () のCPUコア使用量を設定しますRFS。使用しているインスタンスタイプに基づく一連の推奨設定については、「付録: 割り込み/RPSチューニングの推奨パラメータ」を参照してください。

1. システム化されたプロセスをエージェントCPUコアから遠ざけて固定します。
2. ハードウェア割り込みリクエストをエージェントCPUコアから遠ざけてルーティングします。
3. 単一のネットワークインターフェイスカードのハードウェアキューがネットワークトラフィックのボトルネックにならないRPSように を設定します。
4. CPU キャッシュヒット率を高め、ネットワークレイテンシーを減らすRFSように を設定します。

が提供するset_irq_affinity.shスクリプトは、上記のすべてRPMを設定します。を crontab に追加すると、起動ごとに適用されます。

```
echo "@reboot sudo /opt/aws/groundstation/bin/set_irq_affinity.sh
 '${interrupt_core_list}' '${rps_core_mask}' >> /var/log/user-data.log 2>&1" >>/var/
spool/cron/root
```

- をカーネルと OS 用に予約されたコアinterrupt_core_listに置き換えます。通常、1 番目と 2 番目は、ハイパースレッドコアペアとともに使用します。これと、上記で選択したコアとが重複しないようにしてください。（例: ハイパースレッドの 96CPU インスタンスの場合は「0,1,48,49」）。
- rps_core_mask は、受信パケットを処理するCPUs必要がある 16 進ビットマスクで、各桁は 4 を表しますCPUs。また、右から 8 文字ごとにカンマで区切る必要があります。すべてを許可CPUsし、キャッシュでバランシングを処理することをお勧めします。
  - 各インスタンスタイプに推奨されるパラメータのリストについては、「付録: 割り込み/RPS チューニングの推奨パラメータ」を参照してください。
- 96CPU インスタンスの例：

```
echo "@reboot sudo /opt/aws/groundstation/bin/set_irq_affinity.sh '0,1,48,49'
 'ffffffff,ffffffff,ffffffff' >> /var/log/user-data.log 2>&1" >>/var/spool/cron/root
```

Rx 割り込みの合体を調整する - ネットワークに影響します

割り込み合体により、ホストシステムに大量の割り込みが発生するのを防ぎ、ネットワークのスループットを向上させることができます。この構成では、パケットが収集され、128 マイクロ秒ごとに 1 つの割り込みが発生します。を crontab に追加すると、起動ごとに適用されます。

```
echo "@reboot sudo ethtool -C ${interface} rx-usecs 128 tx-usecs 128 >>/var/log/user-
data.log 2>&1" >>/var/spool/cron/root
```

- `interface` を、データを受信するように設定されたネットワークインターフェイス (イーサネットアダプタ) に置き換えます。通常、これはEC2インスタンスに割り当てられたデフォルトのネットワークインターフェイス`eth0`です。

Rx リングバッファの調整 - ネットワークに影響

Rx リングバッファのリングエントリ数を増やして、バースト接続中のパケットドロップやオーバーランを防止します。を crontab に追加すると、起動ごとに正しく設定されます。

```
echo "@reboot sudo ethtool -G ${interface} rx 16384 >>/var/log/user-data.log 2>&1" >>/
var/spool/cron/root
```

- `interface` を、データを受信するように設定されたネットワークインターフェイス (イーサネットアダプタ) に置き換えます。通常、これはEC2インスタンスに割り当てられたデフォルトのネットワークインターフェイス`eth0`です。
- c6i.32xlarge インスタンスを設定する場合、リングバッファを、16384 の代わりに 8192 に設定するようにコマンドを変更する必要があります。

C-State CPU の調整 - 影響 CPU

CPU C 状態を設定して、問い合わせの開始中にパケットが失われる可能性のあるアイドルを防止します。インスタンスの再起動が必要です。

```
echo "GRUB_CMDLINE_LINUX_DEFAULT=\"console=tty0 console=ttyS0,115200n8
 net.ifnames=0 biosdevname=0 nvme_core.io_timeout=4294967295 intel_idle.max_cstate=1
 processor.max_cstate=1 max_cstate=1\"" >/etc/default/grub
echo "GRUB_TIMEOUT=0" >>/etc/default/grub
grub2-mkconfig -o /boot/grub2/grub.cfg
```

進入ポートの予約 - ネットワークに影響します

カーネルの使用状況と競合しないように、AwsGroundStationAgentEndpoint の入力アドレスのポート範囲内のすべてのポートを予約します。ポートの使用が競合すると、コンタクトやデータ配信の失敗につながります。

```
echo "net.ipv4.ip_local_reserved_ports=${port_range_min}-${port_range_max}" >> /etc/
sysctl.conf
```

- 例えば、echo "net.ipv4.ip_local_reserved_ports=42000-43500" >> /etc/sysctl.conf などです。

再起動

すべてのチューニングが正常に適用されたら、インスタンスを再起動してチューニングを有効にします。

```
sudo reboot
```

付録: 割り込み/RPSチューニングの推奨パラメータ

このセクションでは、チューニングセクション Tune Hardware Interrupts and Receive Queues - Impacts CPU and Network で使用する推奨パラメータ値を決定します。

ファミリー	インスタンスタイプ	${interrupt_core_list}	${rps_core_mask}
C6i	• c6i.32xlarge	• 0、1、64、	• ffffffff, ffffffff, ffffffff, ffffffff

ファミリー	インスタンスタイプ	${interrupt_core_list}	${rps_core_mask}
c5	• c5.24xlarge • c5.18xlarge • c5.12xlarge	• 0、1、48、 • 0、1、36、 • 0、1、24、	• ffffffff, ffffffff, ffffffff • ff、ffffffff、ffffffff • ffff,ffffffff
c5n	• c5n.metal • c5n.18xlarge	• 0、1、36、 • 0、1、36、	• ff、ffffffff、ffffffff • ff、ffffffff、ffffffff
m5	• m5.24xlarge • m5.12xlarge	• 0、1、48、 • 0、1、24、	• ffffffff, ffffffff, ffffffff • ffff,ffffffff
r5	• r5.metal • r5.24xlarge	• 0、1、48、 • 0、1、48、	• ffffffff, ffffffff, ffffffff • ffffffff, ffffffff, ffffffff
r5n	• r5n.metal • r5n.24xlarge	• 0、1、48、 • 0、1、48、	• ffffffff, ffffffff, ffffffff • ffffffff, ffffffff, ffffffff

ファミリー	インスタンスタイプ	${interrupt_core_list}	${rps_core_mask}
g4dn	• g4dn.metal • g4dn.16xlarge • g4dn.12xlarge	• 0、1、48、 • 0、1、32、 • 0、1、24、	• ffffffff, ffffffff, ffffffff • ffffffff, ffffffff • ffff,ffffffff
p4d	• p4d.24xlarge	• 0、1、48、	• ffffffff, ffffffff, ffffffff
p3dn	• p3dn.24xlarge	• 0、1、48、	• ffffffff, ffffffff, ffffffff

DigIF へのコンタクトの実行を準備をする

1. Core CPU Planning で目的のデータフローを確認し、エージェントが使用できるコアのリストを提供します。「CPU コアプランニング」を参照してください。
2. AWS Ground Station エージェント設定ファイルを確認します。「AWS Ground Station エージェント設定」を参照してください。
3. 必要なパフォーマンスチューニングが適用されていることを確認します。「インスタンスのパフォーマンスEC2を調整する」を参照してください。
4. 呼び出されたすべてのベストプラクティスに従っていることを確認します。「ベストプラクティス」を参照してください。
5. スケジュールされた問い合わせ開始時刻より前に AWS Ground Station 、エージェントが開始されていることを確認します。

```
systemctl status aws-groundstation-agent
```

6. 次の方法で、スケジュールされた問い合わせ開始時刻より前に AWS Ground Station エージェントが正常であることを確認します。

```
aws groundstation get-dataflow-endpoint-group --dataflow-endpoint-group-id
 ${DATAFLOW-ENDPOINT-GROUP-ID} --region ${REGION}
```

`agentStatus` の `awsGroundStationAgentEndpoint`が ACTIVEで、 `auditResults`が であることを確認しますHEALTHY。

ベストプラクティス

Amazon EC2 のベストプラクティス

現在のEC2ベストプラクティスに従い、十分なデータストレージの可用性を確保します。

https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ec2-best-practices.html

Linux スケジューラ

対応するプロセスが特定のコアに固定されていない場合、Linux スケジューラはUDPソケット上のパケットの順序を変更できます。UDP データを送受信するスレッドは、データ転送中に特定のコアに固定する必要があります。

AWS Ground Station マネージドプレフィックスリスト

アンテナからの通信を許可するネットワークルールを指定するときは、`com.amazonaws.global.groundstation` AWSマネージドプレフィックスリストを使用することをお勧めします。AWS マネージドプレフィックスリストの詳細については、「マネージドプレフィックスリストの使用」を参照してください。 AWS

単一のコンタクトの制限

AWS Ground Station Agent は、コンタクトごとに複数のストリームをサポートしますが、一度に 1 つのコンタクトのみをサポートします。スケジュールの問題を防ぐため、複数のデータフローエンドポイントグループ間でインスタンスを共有しないでください。単一のエージェント設定が複数の異なる DFEG に関連付けられている場合ARNs、登録は失敗します。

AWS Ground Station エージェントと一緒にサービスとプロセスを実行する

AWS Ground Station エージェントと同じEC2インスタンスでサービスやプロセスを起動する場合、コンタクト中にボトルネックやデータ損失が発生する可能性があるため、 AWS Ground Station エージェントや Linux カーネルが vCPUs 使用しないようにバインドすることが重要です。この特定の へのバインドの概念 vCPUs は、アフィニティと呼ばれます。

避けるべきコア：

- agentCpuCores from エージェント設定ファイル
- interrupt_core_list (ハードウェアの中断と受信キューの調整 - 影響CPUとネットワーク から)。
  - デフォルト値は から入手できます。 付録: 割り込み/RPSチューニングの推奨パラメータ

c5.24xlarge インスタンスの使用例

を指定した場合

"agentCpuCores": [24,25,26,27,72,73,74,75]"

と 実行

echo "@reboot sudo /opt/aws/groundstation/bin/set_irq_affinity.sh '0,1,48,49' 'ffffffff,ffffffff,ffffffff' >> /var/log/user-data.log 2>&1" >>/var/spool/cron/root

次に、次のコアを避けます。

0,1,24,25,26,27,48,49,72,73,74,75

サービスの最適化 (システム)

新しく起動されたサービスは、interrupt_core_list前述の に自動的にアフィニティされます。起動したサービスのユースケースで追加のコアが必要な場合、または混雑の少ないコアが必要な場合は、このセクションに従ってください。

コマンドを使用して、サービスが現在設定されているアフィニティを確認します。

```
systemctl show --property CPUAffinity <service name>
```

のような空の値が表示される場合はCPUAffinity=、上記のコマンドのデフォルトコアを使用する可能性が高いことを意味します。 ...bin/set_irq_affinity.sh <using the cores here> ...

特定のアフィニティを上書きして設定するには、 を実行してサービスファイルの場所を見つけます。

```
systemctl show -p FragmentPath <service name>
```

ファイル (vi、 などを使用) を開いて変更しnano、 CPUAffinity=<core list>を次のような[Service]セクションに配置します。

```
[Unit]
...

[Service]
...
CPUAffinity=2,3

[Install]
...
```

ファイルを保存し、サービスを再起動してアフィニティを適用します。

```
systemctl daemon-reload
systemctl restart <service name>

# Additionally confirm by re-running
systemctl show --property CPUAffinity <service name>
```

詳細については、Red Hat Enterprise Linux 8 - カーネルの管理、モニタリング、更新 - 第 27 章を参照してください。systemd を使用してCPUアフィニティとNUMAポリシーを設定する。

プロセスの最適化 (スクリプト)

新しい起動したスクリプトとプロセスでは、デフォルトの Linux 動作によりマシン上の任意のコアを使用できるため、手動でアフィニティすることが強く推奨されます。

実行中のプロセス (Python、bash スクリプトなど) のコア競合を回避するには、以下を使用してプロセスを起動します。

```
taskset -c <core list> <command>
# Example: taskset -c 8 ./bashScript.sh
```

プロセスがすでに実行されている場合は、pidof、top、 などのコマンドを使用して、特定のプロセスのプロセス ID (PID) psを見つけます。PID では、次の との現在のアフィニティを確認できます。

```
taskset -p <pid>
```

および は、次の方法で変更することができます。

```
taskset -p <core mask> <pid>
# Example: taskset -p c 32392 (which sets it to cores 0xc -> 0b1100 -> cores 2,3)
```

タスクセットの詳細については、「タスクセット - Linux man ページ」を参照してください。

トラブルシューティング

エージェントの起動の失敗

AWS Ground Station エージェントは、いくつかの理由で起動に失敗する可能性がありますが、最も一般的なシナリオは、エージェント設定ファイルの設定ミスである可能性があります。エージェントを起動すると (「AWS Ground Station エージェント起動」を参照)、次のようなステータスが表示される場合があります。

```
#agent is automatically retrying a restart
aws-groundstation-agent.service - aws-groundstation-agent
Loaded: loaded (/usr/lib/systemd/system/aws-groundstation-agent.service; enabled;
 vendor preset: disabled)
Active: activating (auto-restart) (Result: exit-code) since Fri 2023-03-10 01:48:14
 UTC; 23s ago
Docs: https://aws.amazon.com/ground-station/
Process: 43038 ExecStart=/opt/aws/groundstation/bin/launch-aws-gs-agent (code=exited,
 status=101)
Main PID: 43038 (code=exited, status=101)

#agent has failed to start
aws-groundstation-agent.service - aws-groundstation-agent
Loaded: loaded (/usr/lib/systemd/system/aws-groundstation-agent.service; enabled;
 vendor preset: disabled)
Active: failed (Result: start-limit) since Fri 2023-03-10 01:50:15 UTC; 13s ago
Docs: https://aws.amazon.com/ground-station/
Process: 43095 ExecStart=/opt/aws/groundstation/bin/launch-aws-gs-agent (code=exited,
 status=101)
Main PID: 43095 (code=exited, status=101)
```

トラブルシューティング

```
sudo journalctl -u aws-groundstation-agent | grep -i -B 3 -A 3 'Loading Config' | tail
 -6
```

これによる出力は、次のようになります。

```
launch-aws-gs-agent[43095]: Running with options Production(ProductionOptions
 { endpoint: None, region: None })
launch-aws-gs-agent[43095]: Loading Config
launch-aws-gs-agent[43095]: System has 96 logical cores
systemd[1]: aws-groundstation-agent.service: main process exited, code=exited,
 status=101/n/a
systemd[1]: Unit aws-groundstation-agent.service entered failed state.
```

「Loading Config」の後にエージェントを起動できない場合、これは、エージェント設定に問題があることを示しています。エージェント設定を検証するには、「エージェント設定ファイル」を参照してください。

AWS Ground Station エージェントログ

AWS Ground Station エージェントは、コンタクトの実行、エラー、ヘルスステータスに関する情報を、エージェントを実行しているインスタンスのログファイルに書き込みます。インスタンスに手動で接続することで、ログファイルを表示できます。

エージェントログは以下の場所で確認できます。

```
/var/log/aws/groundstation
```

利用可能な問い合わせはありません

問い合わせをスケジュールするには、正常な AWS Ground Station エージェントが必要です。を介して に AWS Ground Station APIクエリを実行して、 AWS Ground Station エージェントが起動し、正常であることを確認してください get-dataflow-endpoint-group。

```
aws groundstation get-dataflow-endpoint-group --dataflow-endpoint-group-id ${DATAFLOW-
ENDPOINT-GROUP-ID} --region ${REGION}
```

`agentStatus` の `awsGroundStationAgentEndpoint`が ACTIVEで、 `auditResults`が であることを確認しますHEALTHY。

サポート情報

AWS サポートを通じて Ground Station チームに連絡してください。

1. 影響を受けているコンタクトがあれば、`contact_id` を伝えてください。 AWS Ground Station チームは、この情報なしで特定の問い合わせを調査することはできません。
2. 既に実施したすべてのトラブルシューティング手順に関する詳細を提供してください。
3. トラブルシューティングガイダンスに記載されているコマンドの実行中に表示されたエラーメッセージがあれば、提供してください。

エージェントリリースノート

最新バージョンのエージェント

バージョン 1.0.3555.0

リリース日: 03/27/2024

サポート終了日: 08/31/2024

RPM チェックサム :

- SHA256: `108f3aceb00e5af549839cd766c56149397e448a6e1e1429c89a9eebb6bc0fc1`
- MD5: `65b72fa507fb0af32651adbb18d2e30f`

変更:

- タスクの起動中に、選択した実行可能バージョンのエージェントメトリクスを追加します。
- 他のバージョンが利用可能な場合、特定の実行可能バージョンを回避するための設定ファイルサポートを追加します。
- ネットワーク診断とルーティング診断を追加します。
- その他のセキュリティ機能。
- 一部のメトリクスレポートエラーがログファイルではなく stdout/journal に書き込まれた問題を修正しました。
- ネットワークに到達できないソケットエラーを丁寧に処理します。
- 送信元エージェントと送信先エージェント間のパケット損失とレイテンシーを測定します。
- バージョン 2.0 をリリース aws-gs-datapipeして、新しいプロトコル機能と、連絡先を新しいプロトコルに透過的にアップグレードする機能をサポートします。

非推奨のエージェントバージョン

バージョン 1.0.2942.0

リリース日: 06/26/2023

サポート終了日: 05/31/2024

RPM チェックサム :

- SHA256: `7d94b642577504308a58bab28f938507f2591d4e1b2c7ea170b77bea97b5a9b6`
- MD5: `661ff2b8f11aba5d657a6586b56e0d8f`

変更:

- ディスクでエージェントがRPM更新され、変更を有効にするためにエージェントの再起動が必要な場合のエラーログを追加しました。
- エージェントユーザーガイドのチューニングステップに従って正しく適用されるように、ネットワークチューニングの検証を追加しました。
- ログアーカイブに関する エージェントログの誤った警告の原因となったバグを修正しました。
- パケット損失検出が改善されました。
- エージェントのインストールを更新し、エージェントが既に実行RPMされている場合、 のインストールまたはアップグレードを防止しました。

バージョン 1.0.2716.0

リリース日: 03/15/2023

サポート終了日: 05/31/2024

RPM チェックサム :

- SHA256: `cb05b6a77dfcd5c66d81c0072ac550affbcefefc372cc5562ee52fb220844929`
- MD5: `65266490c4013b433ec39ee50008116c`

変更:

- タスク中にエージェントが失敗した場合、ログのアップロードを有効にします。
- 提供されたネットワーク調整スクリプトの Linux 互換性のバグを修正しました。

バージョン 1.0.2677.0

リリース日: 02/15/2023

サポート終了日: 05/31/2024

RPM チェックサム：

- SHA256: `77cfe94acb00af7ca637264b17c9b21bd7afdc85b99dffdd627aec9e99397489`
- MD5: `b8533be7644bb4d12ab84de21341adac`

変更:

- 最初に一般公開されたエージェントリリース。

RPM インストールの検証

RPM 最新バージョン、 から検証されたMD5ハッシュRPM、および sha256sum を使用したSHA256ハッシュを以下に示します。これらの値を組み合わせて、地上局エージェントに使用されるRPMバージョンを検証できます。

最新バージョンのエージェント

バージョン 1.0.3555.0

リリース日: 03/27/2024

サポート終了日: 08/31/2024

RPM チェックサム :

- SHA256: `108f3aceb00e5af549839cd766c56149397e448a6e1e1429c89a9eebb6bc0fc1`
- MD5: `65b72fa507fb0af32651adbb18d2e30f`

変更:

- タスクの起動中に、選択した実行可能バージョンのエージェントメトリクスを追加します。
- 他のバージョンが利用可能な場合、特定の実行可能バージョンを回避するための設定ファイルサポートを追加します。
- ネットワーク診断とルーティング診断を追加します。
- その他のセキュリティ機能。
- 一部のメトリクスレポートエラーがログファイルではなく stdout/journal に書き込まれた問題を修正しました。
- ネットワークに到達できないソケットエラーを丁寧に処理します。
- 送信元エージェントと送信先エージェント間のパケット損失とレイテンシーを測定します。
- バージョン 2.0 をリリース aws-gs-datapipeして、新しいプロトコル機能と、連絡先を新しいプロトコルに透過的にアップグレードする機能をサポートします。

を検証する RPM

このRPMインストールを検証するために必要なツールは次のとおりです。

- sha256sum
- rpm

Amazon Linux 2 では、どちらのツールもデフォルトで提供されています。これらのツールは、RPM使用している が正しいバージョンであることを確認するのに役立ちます。まず、S3 バケットRPMから最新の をダウンロードします (のダウンロード手順エージェントをダウンロードするについては、「」を参照してくださいRPM)。このファイルがダウンロードされたら、いくつか確認すべき点があります。

- RPM ファイルの sha256sum を計算します。使用しているコンピューティングインスタンスのコマンドラインから以下のアクションを実行します。

  ```
  sha256sum aws-groundstation-agent.rpm
  ```

  この値を取得し、上の表と比較します。これは、ダウンロードされたRPMファイルが、AWSGround Station が顧客に提供したファイルとして有効なことを示しています。ハッシュが一致しない場合は、 をインストールせずRPM、コンピューティングインスタンスから削除します。
- ファイルのMD5ハッシュもチェックして、 RPMが侵害されていないことを確認します。これを行うには、次のRPMコマンドを実行してコマンドラインツールを使用します。

  ```
  rpm -Kv ./aws-groundstation-agent.rpm
  ```

  ここに記載されているMD5ハッシュが、上記の表にあるバージョンのMD5ハッシュと同じであることを確認します。これらのハッシュの両方が Docs AWS にリストされているこのテーブルに対して検証されると、ダウンロードおよびインストールRPMされた が の安全で妥協のないバージョンであることがお客様に保証されますRPM。

AWS Ground Station エージェントユーザーガイドのドキュメント履歴

次の表は、 AWS Ground Station エージェントユーザーガイドの各リリースにおける重要な変更点を示しています。

変更	説明	日付
ドキュメントの更新	古いインスタンスファミリーのサポートを削除しました: m4。	2024 年 9 月 30 日
ドキュメントの更新	エージェント要件 で、サブネットと Amazon EC2インスタンスを同じアベイラビリティーゾーンに保持することに関するコメントを追加しました。	2024 年 7 月 18 日
ドキュメントの更新	AWS Ground Station エージェントを独自のユーザーガイドに分割します。以前の変更については、AWS「Ground Station ユーザーガイド」のドキュメント履歴を参照してください。	2024 年 7 月 18 日

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