AWS Ground Station 상담원 사용 설명서

AWS Ground Station

AWS Ground Station: AWS Ground Station 상담원 사용 설명서

Table of Contents

개요 1
AWS Ground Station 에이전트란 무엇입니까? 1
에이전트의 특징 AWS Ground Station 2
에이전트 요구 사항 3
VPC다이어그램 4
지원되는 운영 체제 5
AWS Ground Station 에이전트를 통한 데이터 전송 6
다중 데이터 흐름, 단일 수신기 6
다중 데이터 흐름, 다중 수신기 7
Amazon EC2 인스턴스 선택 및 CPU 계획 9
지원되는 Amazon EC2 인스턴스 유형 9
CPU코어 플래닝 10
아키텍처 정보 수집 11
CPU대입 예제 12
부록: `lscpu -p` c5.24xlarge의 출력 (전체) 13
에이전트 설치 16
AWS CloudFormation 템플릿 사용 16
1단계: 리소스 생성 AWS 16
2단계: 에이전트 상태 확인 16
수동 설치: EC2 16
1단계: AWS 리소스 생성 16
2단계: EC2 인스턴스 생성 17
3단계: 에이전트 다운로드 및 설치 17
4단계: 에이전트 구성 18
5단계: 성능 조정 적용 18
6단계: 에이전트 관리 19
에이전트 관리 20
AWS Ground Station 에이전트 구성 20
AWS Ground Station 에이전트 시작 20
AWS Ground Station 에이전트 스톱 21
AWS Ground Station 에이전트 업그레이드 21
AWS Ground Station 에이전트 다운그레이드 22
AWS Ground Station 에이전트 제거 23
AWS Ground Station 에이전트 상태 23

AWS Ground Station 상담원 RPM 정보 24
에이전트 구성 25
에이전트 구성 파일 25
예 25
필드 분류 25
EC2인스턴스 성능 조정 29
하드웨어 인터럽트 및 수신 대기열 조정 (영향 및 네트워크) CPU 29
Tune Rx 인터럽트 병합 - 네트워크에 영향을 미칩니다. 30
Tune Rx 링 버퍼 - 네트워크에 영향을 미칩니다. 31
CPUC-State 조정 - 영향 CPU 31
인그레스 포트 예약 - 네트워크에 영향을 미칩니다. 31
재부팅 32
부록: 인터럽트/튜닝을 위한 권장 파라미터 RPS 32
DiGif 접촉 받기 위한 준비 34
모범 사례 35
아마존 EC2 모범 사례 35
Linux 스케줄러 35
AWS Ground Station 관리형 프리픽스 목록 35
단일 접점 제한 35
AWS Ground Station 에이전트와 함께 서비스 및 프로세스 실행 35
예를 들어, c5.24xlarge 인스턴스를 사용해 36
어피니타이징 서비스 (시스템드) 36
어피니타이징 프로세스 (스크립트) 37
문제 해결 39
에이전트 시작 실패 39
문제 해결 39
AWS Ground Station 에이전트 로그 40
연락처가 없습니다. 40
지원 받기 41
상담원 릴리스 노트 42
최신 에이전트 버전 42
버전 1.0.3555.0 42
더 이상 사용되지 않는 에이전트 버전 42
버전 1.0.2942.0 42
버전 1.0.2716.0 43
버전 1.0.2677.0 43

RPM설치 검증 45
최신 에이전트 버전 42
버전 1.0.3555.0 42
다음을 확인하십시오. RPM 45
문서 기록 47
..... xlviii

개요

AWS Ground Station 에이전트란 무엇입니까?

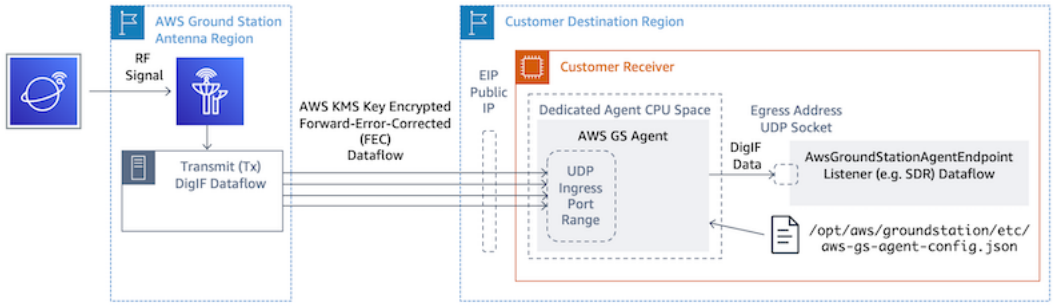
로 제공되는 AWS Ground Station 에이전트를 사용하면 Ground Station 연결 중에 동기식 광대역 디지털 중간 주파수 (DiGif) 데이터 흐름을 수신 (다운링크) 할 수 있습니다. RPM AWS 데이터 전달을 위한 두 가지 옵션을 선택할 수 있습니다.

1. EC2인스턴스로의 데이터 전송 - 소유한 EC2 인스턴스로의 데이터 전송. AWS Ground Station 에이전트를 관리합니다. 이 옵션은 실시간에 가까운 데이터 처리가 필요한 경우에 가장 적합할 수 있습니다. 데이터 전송에 대한 자세한 내용은 Amazon Elastic Compute 클라우드로 EC2 데이터 전송 가이드를 참조하십시오.
2. S3 버킷으로 데이터 전송 - Ground Station 관리 서비스를 통해 소유한 AWS S3 버킷으로 데이터를 전송합니다. S3 데이터 전송에 대한 자세한 내용은 시작하기 AWS Ground Station 안내서를 참조하십시오.

두 가지 데이터 전송 모드 모두 AWS 리소스 세트를 생성해야 합니다. 신뢰성, 정확성 및 지원 가능성을 CloudFormation 보장하려면 를 사용하여 AWS 리소스를 생성하는 것이 좋습니다. 각 연락처는 EC2 또는 S3에만 데이터를 전송할 수 있으며 두 연락처 모두에 동시에 데이터를 전송할 수는 없습니다.

> **Note**
>
> S3 데이터 전송은 Ground Station의 관리형 서비스이므로 이 가이드에서는 EC2 인스턴스로의 데이터 전송에 중점을 둡니다.

다음 다이어그램은 소프트웨어 정의 라디오 () 또는 유사한 리스너를 사용하여 AWS Ground Station 안테나 영역에서 EC2 인스턴스로의 DiGif 데이터 흐름을 보여줍니다. SDR



에이전트의 특징 AWS Ground Station

AWS Ground Station 에이전트는 디지털 중간 주파수 (DigiF) 다운링크 데이터를 수신하고 다음을 가능하게 하는 복호화된 데이터를 송신합니다.

- 40에서 MHz MHz 400까지의 대역폭의 DiGif 다운링크 기능을 제공합니다.
- 네트워크 상의 모든 퍼블릭 IP (AWSElastic IP) 로 고속, 저지터 DigiF 데이터 전송 AWS
- 순방향 오류 수정 () 을 사용하여 데이터를 안정적으로 전달합니다. FEC
- 암호화를 위한 고객 관리 AWS KMS 키를 사용하여 데이터를 안전하게 전송합니다.

에이전트 요구 사항

> **Note**
>
> 이 AWS Ground Station 에이전트 가이드에서는 시작 안내서를 사용하여 Ground Station에 AWS Ground Station 온보딩했다고 가정합니다.

AWS Ground Station Agent Receiver EC2 인스턴스에는 DigiF 데이터를 엔드포인트에 안정적이고 안전하게 전달하기 위한 종속 AWS 리소스 세트가 필요합니다.

1. 수신기를 시작하는 VPC 데 사용할 A입니다. EC2
2. 데이터 암호화/복호화를 위한 AWS KMS 키.
3. 세션 관리자용으로 구성된 SSH 키 또는 EC2 인스턴스 프로필 SSM
4. 네트워크/보안 그룹 규칙은 다음을 허용합니다.
    1. UDP데이터 흐름 엔드포인트 그룹에 지정된 포트를 통해 들어오는 AWS Ground Station 트래픽. 에이전트는 데이터를 수신 데이터 흐름 엔드포인트로 전달하는 데 사용되는 다양한 연속 포트를 예약합니다.
    2. SSH인스턴스 액세스 (참고: AWS 세션 관리자를 사용하여 EC2 인스턴스에 액세스할 수도 있음).
    3. 에이전트 관리를 위해 공개적으로 액세스할 수 있는 S3 버킷에 대한 읽기 액세스.
    4. SSL포트 443을 통한 트래픽으로 에이전트가 AWS Ground Station 서비스와 통신할 수 있습니다.
    5. AWS Ground Station 관리형 접두사 목록의 트래픽 `com.amazonaws.global.groundstation`

또한 퍼블릭 서브넷을 포함하는 VPC 구성이 필요합니다. 서브넷 구성의 배경은 VPC사용 설명서를 참조하십시오.

호환되는 구성:

1. 퍼블릭 서브넷의 EC2 인스턴스와 연결된 엘라스틱 IP.
2. EC2인스턴스에 연결된 퍼블릭 서브넷 (퍼블릭 서브넷과 동일한 가용 영역에 있는 모든 서브넷) 의 엘라스틱 IP. ENI

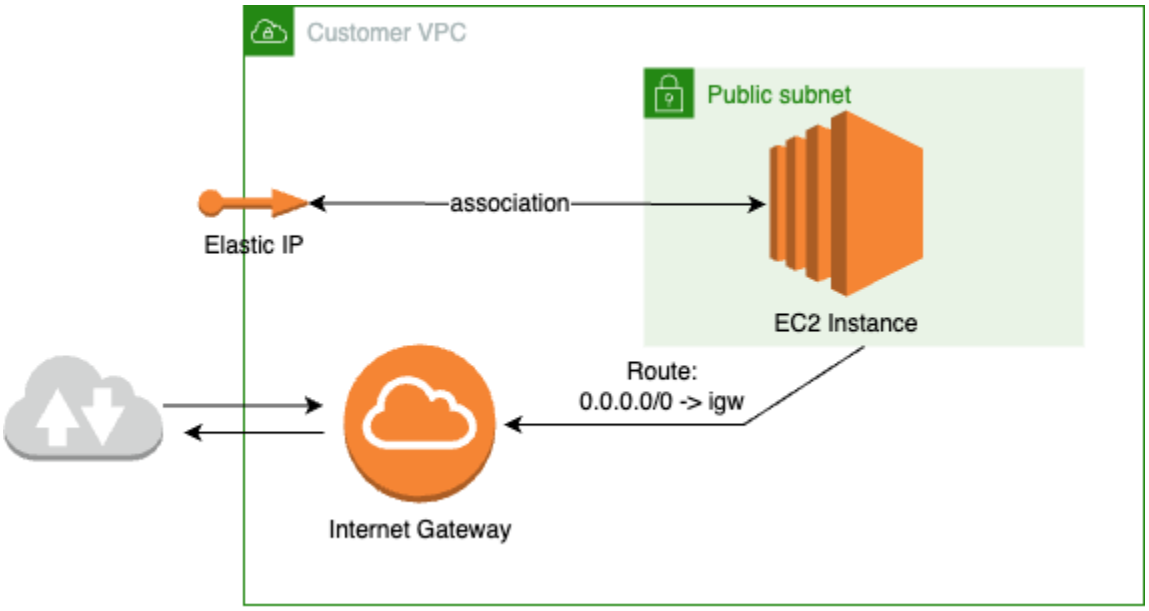
EC2인스턴스와 동일한 보안 그룹을 사용하거나 최소한 다음으로 구성된 최소 규칙 세트를 갖춘 보안 그룹을 지정할 수 있습니다.

- UDP데이터 흐름 엔드포인트 그룹에 지정된 포트를 통해 들어오는 AWS Ground Station 트래픽.

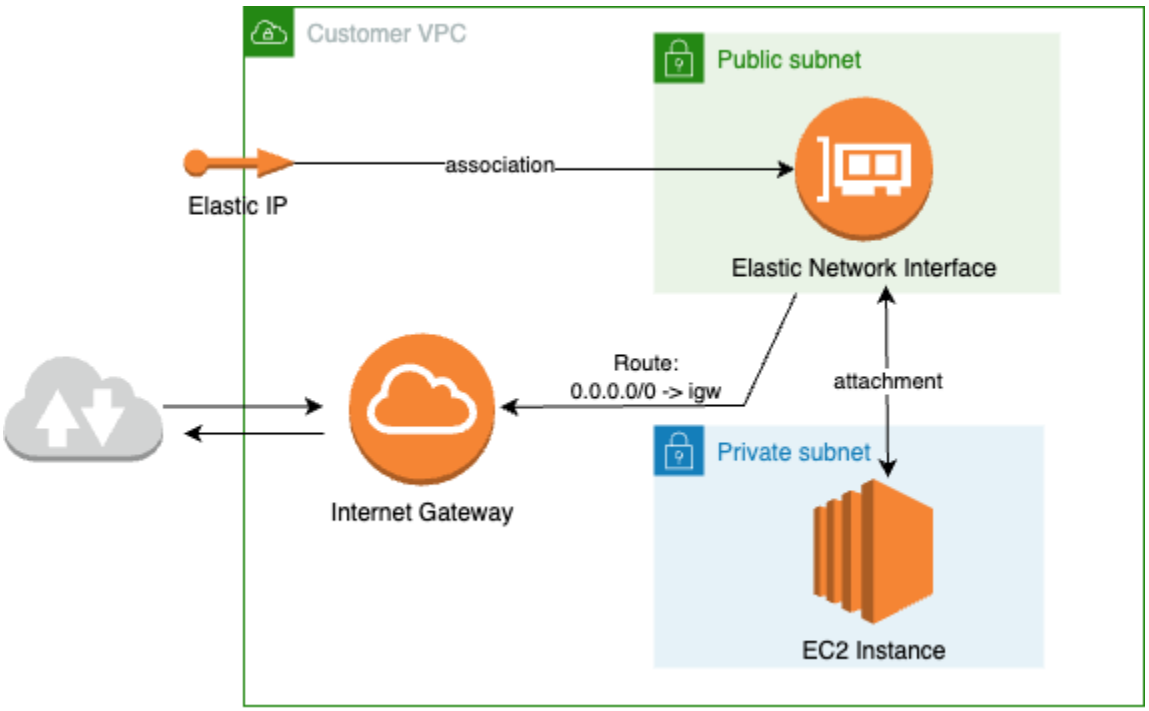
이러한 리소스가 미리 구성된 AWS CloudFormation EC2 데이터 전송 템플릿의 예는 공용 방송 위성 활용 AWS Ground Station 에이전트 (광대역) 를 참조하십시오.

VPC다이어그램

다이어그램: 퍼블릭 서브넷의 EC2 인스턴스와 연결된 엘라스틱 IP



다이어그램: 프라이빗 ENI 서브넷의 EC2 인스턴스에 연결된 퍼블릭 서브넷의 엘라스틱 IP



지원되는 운영 체제

Amazon Linux 2(커널 5.10+)

지원되는 인스턴스 유형은 다음과 같습니다. Amazon EC2 인스턴스 선택 및 CPU 계획

AWS Ground Station 에이전트를 통한 데이터 전송

아래 다이어그램은 광대역 디지털 중간 주파수 (DiGif) 접점 AWS Ground Station 중에 데이터가 어떻게 흐르는지에 대한 개요를 제공합니다.

AWS Ground Station 상담원은 연락처의 데이터플레인 구성 요소를 오케스트레이션합니다. 연락처를 예약하기 전에 에이전트를 올바르게 구성 및 시작하고 등록 (에이전트 시작 시 자동으로 등록) 해야 합니다. AWS Ground Station또한 데이터 수신 소프트웨어 (예: 소프트웨어 정의 라디오) 가 실행 중이고에서 데이터를 수신하도록 구성되어 있어야 합니다. AwsGroundStationAgentEndpointegressAddress

AWS Ground Station 에이전트는 소프트웨어 정의 라디오 (SDR) egressAddress 가 수신 중인 대상 엔드포인트로 전송하기 전에 백그라운드에서 작업을 AWS Ground Station 수신하고 전송 중에 적용된 AWS KMS 암호화를 실행 취소합니다. AWS Ground Station 에이전트와 기본 구성 요소는 구성 파일에 설정된 CPU 경계를 준수하여 인스턴스에서 실행되는 다른 애플리케이션의 성능에 영향을 미치지 않도록 합니다.

연락처와 관련된 Receiver 인스턴스에서 AWS Ground Station 에이전트를 실행해야 합니다. 단일 Receiver 인스턴스에서 모든 데이터 흐름을 수신하려는 경우 아래와 같이 단일 AWS Ground Station 에이전트로 여러 데이터 흐름을 조정할 수 있습니다.

다중 데이터 흐름, 단일 수신기

예제 시나리오:

동일한 수신기 인스턴스에서 DigiF 데이터 플로우에 따라 두 개의 안테나 다운링크를 수신하려고 합니다. EC2 두 개의 다운링크는 200과 100이 됩니다. MHz MHz

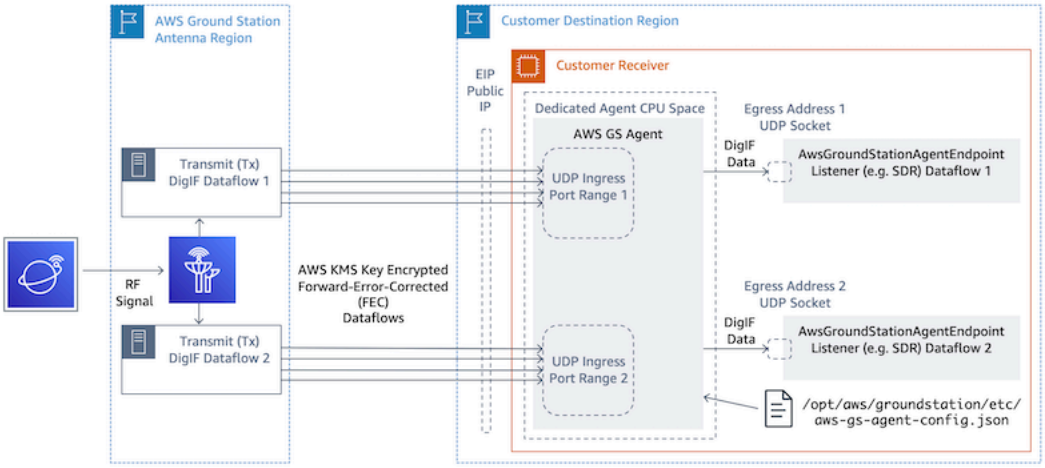
AwsGroundStationAgentEndpoints:

각 데이터 흐름에 하나씩, 총 두 개의 `AwsGroundStationAgentEndpoint` 리소스가 있습니다. 두 엔드포인트는 동일한 퍼블릭 IP 주소(`ingressAddress.socketAddress.name`)를 갖게 됩니다. 데이터 플로우가 동일한 인스턴스에서 수신되므로 `portRange` 인그레스가 겹치지 않아야 합니다. EC2 두 `egressAddress.socketAddress.port` 모두 고유해야 합니다.

CPU계획:

- 인스턴스에서 단일 AWS Ground Station 에이전트를 실행하기 위한 코어 1개 (2vCPU)
- DigiF 데이터플로우 1을 수신하기 위한 코어 6개 (12vCPU) (표에서 MHz 200개 조회) CPU코어 플래닝

- DigiF 데이터플로우 2를 수신하기 위한 코어 4개 (8vCPU) (표에서 MHz 100개 조회) CPU코어 플래닝
- 총 전용 상담원 CPU 공간 = 동일한 소켓에 11코어 (22vCPU)



다중 데이터 흐름, 다중 수신기

예제 시나리오

DigiF 데이터가 서로 다른 수신기 인스턴스에서 흐르기 때문에 두 개의 안테나 다운링크를 수신하려고 합니다. EC2 두 다운링크 모두 400개가 됩니다. MHz

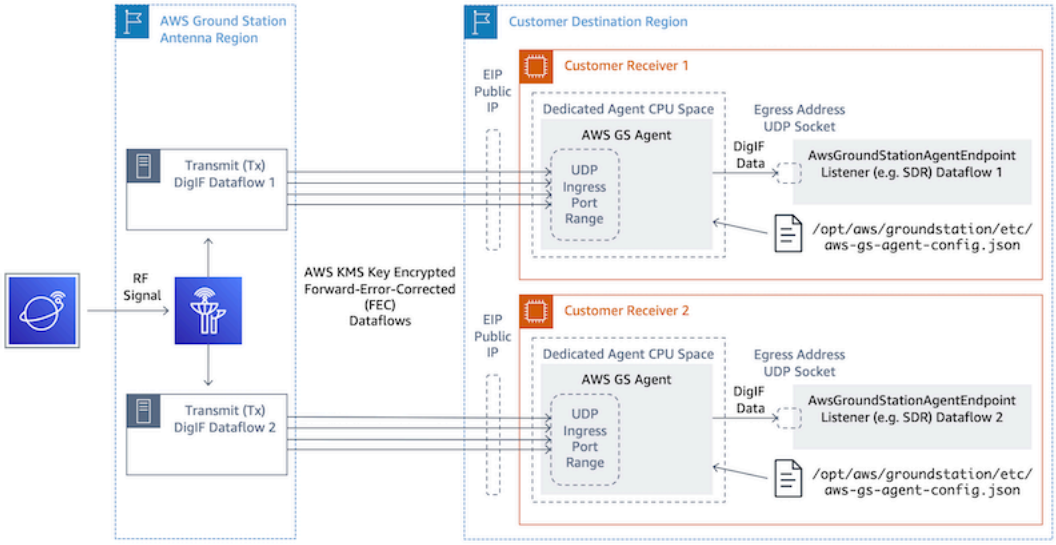
AwsGroundStationAgentEndpoints:

각 데이터 흐름에 하나씩, 총 두 개의 `AwsGroundStationAgentEndpoint` 리소스가 있습니다. 엔드포인트는 다른 퍼블릭 IP 주소(`ingressAddress.socketAddress.name`)를 갖게 됩니다. 데이터 흐름이 별도의 인프라에서 수신되고 서로 충돌하지 않으므로 `ingressAddress` 또는 `egressAddress`에 대한 포트 값에는 제한이 없습니다.

CPU계획:

- 수신기 인스턴스 1
  - 인스턴스에서 단일 AWS Ground Station 에이전트를 실행하기 위한 코어 1개 (2vCPU)
  - 9코어 (18vCPU) 로 DigiF 데이터플로우 1을 수신할 수 있습니다 (표에서 MHz 400회 조회). CPU 코어 플래닝
  - 총 전용 상담원 CPU 공간 = 동일한 소켓의 코어 10개 (20vCPU)

- 수신기 인스턴스 2
  - 인스턴스에서 단일 AWS Ground Station 에이전트를 실행하기 위한 코어 1개 (2vCPU)
  - 9코어 (18vCPU) 로 DigiF 데이터플로우 2를 수신할 수 있습니다 (표에서 MHz 400회 조회). CPU 코어 플래닝
  - 총 전용 상담원 CPU 공간 = 동일한 소켓의 코어 10개 (20vCPU)



Amazon EC2 인스턴스 선택 및 CPU 계획

지원되는 Amazon EC2 인스턴스 유형

컴퓨팅 집약적인 데이터 전송 워크플로로 인해 AWS Ground Station 에이전트를 작동하려면 전용 CPU 코어가 필요합니다. 다음 인스턴스 유형을 지원합니다. 사용 사례에 가장 적합한 인스턴스 유형을 CPU코어 플래닝 결정하려면 참조하세요.

인스턴스 유형	기본값 vCPUs	기본 CPU 코어
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
m4.16xlarge	64	32
m5.12xlarge	48	24
m5.24xlarge	96	48
m6i.32xlarge	128	64
p3dn.24xlarge	96	48
p4d.24xlarge	96	48

인스턴스 유형	기본값 vCPUs	기본 CPU 코어
r5.24xlarge	96	48
r5.metal	96	48
r5n.24xlarge	96	48
r5n.metal	96	48
r6i.32xlarge	128	64

CPU코어 플래닝

AWS Ground Station 에이전트에는 각 데이터 흐름에 대해 L3 캐시를 공유하는 전용 프로세서 코어가 필요합니다. 에이전트는 하이퍼 스레드 (HT) CPU 쌍을 활용하도록 설계되었으며 사용하기 위해서는 HT 쌍을 예약해야 합니다. 하이퍼스레드 쌍은 단일 코어에 포함된 가상 CPUs (vCPU) 쌍입니다. 다음 표에서는 데이터 흐름 데이터 속도를 단일 데이터 흐름에 대해 에이전트용으로 예약된 필수 코어 수에 매핑한 것입니다. 이 표는 Cascade Lake 이상을 가정하며 지원되는 모든 인스턴스 유형에 유효합니다. CPUs 대역폭이 표의 항목 사이에 있는 경우 다음으로 높은 항목을 선택하십시오.

에이전트는 관리 및 조정을 위해 추가 예약 코어가 필요하므로 필요한 총 코어는 각 데이터 흐름에 필요한 코어 (아래 차트 참조) 와 추가 코어 1개 (2) 의 합계입니다. vCPUs

AntennaDownlink MHz대역폭 ()	예상 VITA -49.2 디지프 데이터 속도 (MB/s)	코어 수 (HT 페어) CPU	합계 v CPU
50	1000	3	6
100	2000	4	8
150	3000	5	10
200	4000	6	12
250	5000	6	12
300	6000	7	14

AntennaDownlink MHz대역폭 ()	예상 VITA -49.2 디지프 데이터 속도 (MB/s)	코어 수 (HT 페어) CPU	합계 v CPU
350	7000	8	16
400	8000	9	18

아키텍처 정보 수집

`lscpu`시스템 아키텍처에 대한 정보를 제공합니다. 기본 출력에는 어느 NUMA 노드 vCPUs ("CPU“로 표시됨) 가 어떤 노드에 속하는지 (그리고 각 NUMA 노드는 L3 캐시를 공유함) 가 표시됩니다. 아래에서는 에이전트를 구성하는 데 필요한 정보를 수집하기 위해 `c5.24xlarge` 인스턴스를 검사합니다 AWS Ground Station . 여기에는 수vCPUs, 코어 수, v와 CPU 노드 간 연결과 같은 유용한 정보가 포함됩니다.

```
> 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 에이전트 전용 코어에는 할당된 각 코어의 두 코어가 모두 vCPUs 포함되어야 합니다. 데이터 흐름의 모든 코어는 동일한 노드에 있어야 합니다. NUMA `lscpu`명령 `-p` 옵션은 에이전트를 구성하는 데 필요한 코어 투 CPU 연결을 제공합니다. 관련 필드에는 CPU (이를 v라고 CPU 함), Core 및 L3 (해당 코어가 공유하는 L3 캐시를 나타냄) 이 있습니다. 대부분의 인텔 프로세서에서 NUMA 노드는 L3 캐시와 동일하다는 점에 유의하십시오.

a에 대한 `lscpu -p` 출력의 다음 하위 집합 `c5.24xlarge` (명확성을 위해 축약 및 형식 지정) 을 고려해 보십시오.

```
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
```

출력에서 코어 0에는 vCPUs 0과 16이 포함되고, 코어 1에는 1과 17이 포함되며, 코어 2에는 2와 18이 포함되어 vCPUs 있음을 알 수 있습니다. vCPUs 즉, 하이퍼 스레드 쌍은 0과 16, 1과 17, 2와 18입니다.

CPU대입 예제

예를 들어 350의 이중 극성 광대역 다운링크용 `c5.24xlarge` 인스턴스를 사용하겠습니다. MHz 의 표를 보면 350 MHz 다운링크에는 단일 데이터 흐름에 8코어 (16vCPUs) 가 필요하다는 CPU코어 플래닝 것을 알 수 있습니다. 즉, 두 개의 데이터 흐름을 사용하는 이 이중 극성 설정에는 총 16개의 코어 (32개) 와 에이전트용 코어 1개 (2개vCPUs) 가 필요합니다. vCPUs

`lscpu`출력에는 및 가 포함되어 있다는 것을 알고 있습니다. `c5.24xlarge` `NUMA node0 CPU(s): 0-23,48-71` `NUMA node1 CPU(s): 24-47,72-95` NUMA노드 0에는 필요 이상의 용량이 있으므로 0-23 및 48-71의 코어에서만 할당합니다.

먼저 L3 캐시 또는 노드를 공유하는 각 데이터 흐름에 대해 8개의 코어를 선택합니다. NUMA 그런 다음 출력에서 해당하는 항목 vCPUs (“CPU" 로 표시됨) 을 검색합니다. `lscpu -p` 부록: `lscpu -p` c5.24xlarge의 출력 (전체) 코어 선택 프로세스의 예는 다음과 같을 수 있습니다.

- OS용으로 코어 0-1을 예약하십시오.
- 플로우 1:2-9 및 50-57에 매핑되는 코어 2-9를 vCPUs 선택합니다.
- 플로우 2:10-17 및 58-65에 매핑되는 코어 10-17을 선택합니다. vCPUs
- 에이전트 코어: 18과 66에 매핑되는 코어 18을 선택합니다. vCPUs

그 결과 vCPUs 2-18과 50-66이 되므로 에이전트를 제공할 목록은 다음과 같습니다. `[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]` 에 설명된 CPUs 대로 이러한 프로세스에서 자체 프로세스가 실행되고 있지 않은지 확인해야 합니다. AWS Ground Station 에이전트와 함께 서비스 및 프로세스 실행

이 예제에서 선택한 특정 코어는 다소 임의적이라는 점에 유의하십시오. 각 데이터 흐름에 대해 모두 L3 캐시를 공유해야 한다는 요구 사항을 모두 충족하면 다른 코어 집합도 작동할 수 있습니다.

부록: `lscpu -p` c5.24xlarge의 출력 (전체)

```
> 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
```

에이전트 설치

AWS Ground Station 에이전트는 다음과 같은 방법으로 설치할 수 있습니다.

1. AWS CloudFormation 템플릿 (권장).
2. Amazon에 수동 설치EC2.

AWS CloudFormation 템플릿 사용

EC2데이터 전송 AWS CloudFormation 템플릿은 데이터를 EC2 인스턴스에 전달하는 데 필요한 AWS 리소스를 생성합니다. 이 AWS CloudFormation 템플릿은 AMI AWS Ground Station 에이전트가 사전 설치된 AWS Ground Station 관리 템플릿을 사용합니다. 그러면 생성된 EC2 인스턴스의 부팅 스크립트가 에이전트 구성 파일을 채우고 필요한 성능 조정 () EC2인스턴스 성능 조정 을 적용합니다.

1단계: 리소스 생성 AWS

AWSGround Station Agent (광대역) 를 활용하는 공공 방송 위성 템플릿을 사용하여 AWS 리소스 스택을 생성하십시오.

2단계: 에이전트 상태 확인

기본적으로 에이전트는 구성되고 활성 (시작) 됩니다. 에이전트 상태를 확인하려면 EC2 인스턴스 (SSH또는 SSM 세션 관리자) 에 연결하고 을 참조하십시오AWS Ground Station 에이전트 상태.

수동 설치: EC2

Ground Station에서는 CloudFormation 템플릿을 사용하여 AWS 리소스를 프로비저닝할 것을 권장하지만, 표준 템플릿으로는 충분하지 않은 사용 사례가 있을 수 있습니다. 이러한 경우에는 필요에 맞게 템플릿을 사용자 지정하는 것이 좋습니다. 그래도 요구 사항을 충족하지 못하는 경우 AWS 리소스를 수동으로 생성하고 에이전트를 설치할 수 있습니다.

1단계: AWS 리소스 생성

연락에 필요한 AWS 리소스를 수동으로 설정하는 방법에 대한 지침은 예제 미션 프로필 구성을 참조하십시오.

AwsGroundStationAgentEndpoint리소스는 AWS Ground Station 에이전트를 통해 DigiF 데이터 흐름을 수신하기 위한 엔드포인트를 정의하며, 성공적인 연락을 취하기 위해 매우 중요합니다. API설명

서는 API참조에 있지만 이 섹션에서는 에이전트와 관련된 개념에 대해 간략하게 설명합니다. AWS Ground Station

AWS Ground Station 엔드포인트는 에이전트가 안테나로부터 AWS KMS 암호화된 UDP 트래픽을 수신하는 곳입니다. `ingressAddress` `socketAddressname`는 연결된 EC2 EIP 인스턴스의 퍼블릭 IP입니다. `portRange`는 다른 용도로 예약된 범위 내의 연속 포트는 300개 이상이어야 합니다. 자세한 내용은 인그레스 포트 예약 - 네트워크에 영향을 미칩니다. 섹션을 참조하세요. 이러한 포트는 Receiver VPC 인스턴스가 실행 중인 보안 그룹의 UDP 인그레스 트래픽을 허용하도록 구성해야 합니다.

엔드포인트는 에이전트가 DigiF 데이터 흐름을 사용자에게 전달하는 곳입니다. `egressAddress` 이 위치의 UDP 소켓을 통해 데이터를 수신하는 애플리케이션 (예:SDR) 이 있어야 합니다.

2단계: EC2 인스턴스 생성

다음 모듈을 지원합니다.AMIs

1. AWS Ground Station AMI- `groundstation-al2-gs-agent-ami-*` 여기서 AMI *는 빌드 날짜 - 에이전트가 설치된 상태로 제공됩니다 (권장).
2. `amzn2-ami-kernel-5.10-hvm-x86_64-gp2`.

3단계: 에이전트 다운로드 및 설치

> **Note**
>
> 이전 단계에서 AWS Ground Station 에이전트를 AMI선택하지 않은 경우 이 섹션의 단계를 완료해야 합니다.

에이전트 다운로드

AWS Ground Station 에이전트는 지역별 S3 버킷에서 사용할 수 있으며, AWS 명령줄 (CLI) 을 사용하여 지원 EC2 인스턴스에 다운로드할 수 있습니다. `s3://groundstation-wb-digif-software-${AWS::Region}/aws-groundstation-agent/latest/amazon_linux_2_x86_64/aws-groundstation-agent.rpm` 여기서 $ {AWS: :Region} 은 지원되는 AWSGround Station 콘솔 및 데이터 전송 지역 중 하나를 나타냅니다.

예: us-east-2 AWS 지역에서 로컬로 /tmp 폴더로 최신 rpm 버전을 다운로드합니다.

```
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
```

특정 버전의 AWS Ground Station 에이전트를 다운로드해야 하는 경우 S3 버킷의 버전별 폴더에서 다운로드할 수 있습니다.

예: us-east-2 AWS 지역에서 로컬로 /tmp 폴더로 rpm 버전 1.0.2716.0을 다운로드합니다.

```
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**
>
> 다운로드한 제품의 판매처를 RPM 확인하려면 의 지침을 따르십시오. AWS Ground StationRPM설치 검증

에이전트 설치

```
sudo yum install ${MY_RPM_FILE_PATH}

Example: Assumes agent is in the "/tmp" directory
sudo yum install /tmp/aws-groundstation-agent.rpm
```

4단계: 에이전트 구성

에이전트를 설치한 후에는 에이전트 구성 파일을 업데이트해야 합니다. 에이전트 구성을 참조하세요.

5단계: 성능 조정 적용

AWS Ground Station 에이전트 AMI: 이전 단계에서 AWS Ground Station AMI 에이전트를 선택한 경우 다음 성능 조정을 적용하십시오.

- 하드웨어 인터럽트 및 수신 대기열 조정 (영향 및 네트워크) CPU

- 인그레스 포트 예약 - 네트워크에 영향을 미칩니다.
- 재부팅

기타 AMIs: 이전 단계에서 다른 AMI 것을 선택한 경우 아래에 나열된 모든 튜닝을 EC2인스턴스 성능 조정 적용하고 인스턴스를 재부팅하십시오.

6단계: 에이전트 관리

에이전트 상태를 시작, 중지 및 확인하려면 에이전트 관리을 참조하세요.

에이전트 관리

AWS Ground Station 에이전트는 내장된 Linux 명령 도구를 사용하여 에이전트를 구성, 시작, 중지, 업그레이드, 다운그레이드 및 제거할 수 있는 다음과 같은 기능을 제공합니다.

주제

- AWS Ground Station 에이전트 구성
- AWS Ground Station 에이전트 시작
- AWS Ground Station 에이전트 스톱
- AWS Ground Station 에이전트 업그레이드
- AWS Ground Station 에이전트 다운그레이드
- AWS Ground Station 에이전트 제거
- AWS Ground Station 에이전트 상태
- AWS Ground Station 상담원 RPM 정보

AWS Ground Station 에이전트 구성

`/opt/aws/groundstation/etc` aws-gs-agent-config.json이라는 단일 파일이 포함되어야 하는 위치로 이동합니다. 에이전트 구성 파일 부분 참조

AWS Ground Station 에이전트 시작

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

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

에이전트가 활성 상태임을 보여주는 출력을 생성해야 합니다.

```
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 에이전트 스톱

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

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

에이전트가 비활성(중지됨) 상태임을 나타내는 출력을 생성해야 합니다.

```
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 에이전트 업그레이드

1. 에이전트의 최신 버전을 다운로드합니다. 에이전트 다운로드을 참조하세요.
2. 에이전트를 중지합니다.

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

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

3. 에이전트 업데이트

```
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 에이전트 다운그레이드

1. 필요한 에이전트 버전을 다운로드하세요. 에이전트 다운로드을 참조하세요.
2. 에이전트 다운로드

```
# 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 에이전트 제거

에이전트를 제거하면 /opt/aws/groundstation/etc/ .json의 이름이 aws-gs-agent-config /opt/aws/groundstation/etc/ .json.rpmsave로 바뀝니다. aws-gs-agent-config 동일한 인스턴스에 에이전트를 다시 설치하면 .json의 기본값이 기록되므로 리소스에 해당하는 올바른 값으로 업데이트해야 합니다. aws-gs-agent-config AWS 에이전트 구성 파일을 참조하세요.

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

AWS Ground Station 에이전트 상태

에이전트 상태는 활성(에이전트 실행 중) 또는 비활성(에이전트 중지됨)입니다.

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

예제 출력에는 에이전트가 설치되어 있고, 비활성 상태(중지됨) 및 활성화됨(부팅 시 서비스 시작)이 표시됩니다.

```
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 상담원 RPM 정보

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

출력값은 다음과 같습니다.

> **Note**
>
> “버전”은 에이전트가 게시한 최신 버전에 따라 다를 수 있습니다.

```
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
```

에이전트 구성

에이전트를 설치한 후에는 /opt/aws/groundstation/etc/aws-gs-agent-config.json에서 에이전트 구성 파일을 업데이트해야 합니다.

에이전트 구성 파일

예

```
{
  "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
}
```

필드 분류

기능

기능은 Dataflow Endpoint Group Amazon Resource Names으로 지정됩니다.

필수: True

형식: 문자열 배열

- 값: 기능 ARNs → 문자열

예시:

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

디바이스

이 필드에는 현재 EC2 “장치”를 열거하는 데 필요한 추가 필드가 있습니다.

필수: True

형식: 개체

구성원:

- privateIps
- publicIps
- agentCpuCores
- networkAdapters

privateIps

이 필드는 현재 사용되지 않지만 향후 사용 사례에 포함됩니다. 값이 포함되지 않은 경우 기본값은 [“127.0.0.1”] 입니다.

필수: 거짓

형식: 문자열 배열

- 값: IP 주소 → 문자열

예제:

```
"privateIps": [
    "127.0.0.1"
],
```

publicIps

데이터 흐름 엔드포인트 그룹당 엘라스틱 IP (EIP)

필수: True

형식: 문자열 배열

- 값: IP 주소 → 문자열

예제:

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

agentCPUCores

이는 프로세스를 위해 예약된 가상 코어를 지정합니다. aws-gs-agent 이 값을 적절하게 CPU코어 플래닝 설정하기 위한 요구 사항은 를 참조하세요.

필수: True

형식: Int 배열

- 값: 코어 넘버 → int

예제:

```
"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

이는 데이터를 수신할 이더넷 어댑터 또는 연결된 인터페이스에 해당합니다. ENIs

필수: False

형식: 문자열 배열

- 값: 이더넷 어댑터 이름(`ifconfig`를 실행하여 찾을 수 있음)

예제:

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

EC2인스턴스 성능 조정

> **Note**
>
> CloudFormation 템플릿을 사용하여 AWS 리소스를 프로비저닝한 경우 이러한 조정이 자동으로 적용됩니다. EC2인스턴스를 AMI 사용하거나 수동으로 생성한 경우 가장 안정적인 성능을 얻으려면 이러한 성능 조정을 적용해야 합니다.

조정을 적용한 후에는 반드시 인스턴스를 재부팅해야 합니다.

주제

- 하드웨어 인터럽트 및 수신 대기열 조정 (영향 및 네트워크) CPU
- Tune Rx 인터럽트 병합 - 네트워크에 영향을 미칩니다.
- Tune Rx 링 버퍼 - 네트워크에 영향을 미칩니다.
- CPUC-State 조정 - 영향 CPU
- 인그레스 포트 예약 - 네트워크에 영향을 미칩니다.
- 재부팅

하드웨어 인터럽트 및 수신 대기열 조정 (영향 및 네트워크) CPU

이 섹션에서는 systemd SMPIRQs, 수신 패킷 조정 (RPS) 및 수신 흐름 조정 () 의 CPU 핵심 사용을 구성합니다. RFS 사용 중인 인스턴스 유형에 따른 권장 설정 세트는 부록: 인터럽트/튜닝을 위한 권장 파라미터 RPS을 참조하세요.

1. 시스템 프로세스를 에이전트 코어에서 멀리 떨어진 곳에 고정하십시오. CPU
2. 하드웨어 인터럽트 요청을 에이전트 코어에서 멀리 라우팅합니다. CPU
3. 단일 네트워크 인터페이스 카드의 하드웨어 대기열이 네트워크 트래픽의 병목 현상이 되지 않도록 RPS 구성하십시오.
4. CPU캐시 적중률을 높여 네트워크 지연 시간을 RFS 줄이도록 구성하십시오.

에서 제공하는 `set_irq_affinity.sh` 스크립트는 위의 모든 RPM 사항을 자동으로 구성합니다. crontab에 추가하여 부팅할 때마다 적용되도록 하십시오.

```
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
```

- `interrupt_core_list`커널과 OS용으로 예약된 코어로 교체하십시오. 일반적으로 첫 번째와 두 번째 코어는 하이퍼 스레드 코어 쌍과 함께 사용됩니다. 위에서 선택한 코어와 겹치지 않아야 합니다. (예: 하이퍼 스레드 인스턴스의 경우 '0,1,48,49', 96-1 인스턴스의 경우 '0,1,48,49') CPU
- `rps_core_mask`들어오는 패킷을 처리해야 하는 패킷을 지정하는 16진수 비트 마스크이며, 각 숫자는 4를 나타냅니다. CPUs CPUs 또한 오른쪽부터 시작하여 8자마다 쉼표로 구분해야 합니다. 모두 CPUs 허용하고 캐싱이 밸런싱을 처리하도록 하는 것이 좋습니다.
  - 각 인스턴스 유형에 대한 권장 파라미터 목록을 보려면 부록: 인터럽트/튜닝을 위한 권장 파라미터 RPS을 참조하세요.
- CPU96-인스턴스의 예:

```
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
```

Tune Rx 인터럽트 병합 - 네트워크에 영향을 미칩니다.

인터럽트 통합은 너무 많은 인터럽트로 인해 호스트 시스템이 넘쳐나는 것을 방지하고 네트워크 처리량을 높이는 데 도움이 됩니다. 이 구성에서는 패킷이 수집되고 128마이크로초마다 단일 인터럽트가 생성됩니다. crontab에 추가하여 부팅할 때마다 적용되도록 하십시오.

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

- 데이터를 수신하도록 구성된 네트워크 인터페이스(이더넷 어댑터)로 `interface`를 교체하세요. `eth0`일반적으로 이는 인스턴스에 할당된 기본 네트워크 인터페이스입니다. EC2

Tune Rx 링 버퍼 - 네트워크에 영향을 미칩니다.

연결 버퍼가 폭주하는 동안 패킷 드롭이나 오버런을 방지하려면 Rx 링 버퍼의 링 항목 수를 늘리세요. crontab에 추가하여 부팅할 때마다 올바르게 설정되도록 하십시오.

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

- 데이터를 수신하도록 구성된 네트워크 인터페이스(이더넷 어댑터)로 `interface`를 교체하세요. `eth0`일반적으로 이는 인스턴스에 할당된 기본 네트워크 인터페이스이기 때문입니다. EC2
- c6i.32xlarge 인스턴스를 설정하는 경우 링 버퍼를 16384 대신 8192로 설정하도록 명령을 수정해야 합니다.

CPUC-State 조정 - 영향 CPU

연결 시작 중에 패킷 손실을 초래할 수 있는 유휴 CPU 상태를 방지하려면 C 상태를 설정하십시오. 인스턴스 재부팅이 필요합니다.

```
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
```

인그레스 포트 예약 - 네트워크에 영향을 미칩니다.

커널 사용과 충돌하지 않도록 `AwsGroundStationAgentEndpoint`의 수신 주소 포트 범위 내의 모든 포트를 예약하세요. 포트 사용 충돌로 인해 접촉 및 데이터 전송 실패가 발생할 수 있습니다.

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

- 예: `echo "net.ipv4.ip_local_reserved_ports=42000-43500" >> /etc/sysctl.conf`.

재부팅

모든 조정이 성공적으로 적용된 후 조정이 적용되도록 하려면 인스턴스를 재부팅합니다.

```
sudo reboot
```

부록: 인터럽트/튜닝을 위한 권장 파라미터 RPS

이 섹션에서는 튜닝 섹션 하드웨어 인터럽트 및 수신 큐 - 영향 및 네트워크에 사용할 권장 파라미터 값을 결정합니다. CPU

Family	인스턴스 유형	${interrupt_core_list}	${rps_core_mask}
c6i	• c6i.32xlarge	• 0,1,64,65	• ffffff, ffffff, ffffff, ffffff
C5	• c5.24xlarge • c5.18xlarge • c5.12xlarge	• 0,1,48,49 • 0,1,36,37 • 0,1,24,25	• ffffff, ffffff, ffffff • ff, ffffff, ffffff • ffff,ffffffff
C5n	• c5n.metal • c5n.18xlarge	• 0,1,36,37 • 0,1,36,37	• ff, ffffff, ffffff • ff, ffffff, ffffff
m5	• m5.24xlarge • m5.12xlarge	• 0,1,48,49 • 0,1,24,25	• ffffff, ffffff, ffffff • ffff,ffffffff

Family	인스턴스 유형	${interrupt_core_list}	${rps_core_mask}
r5	• r5.metal • r5.24xlarge	• 0,1,48,49 • 0,1,48,49	• ffffff, ffffff, ffffff • ffffff, ffffff, ffffff
R5n	• r5n.metal • r5n.24xlarge	• 0,1,48,49 • 0,1,48,49	• ffffff, ffffff, ffffff • ffffff, ffffff, ffffff
g4dn	• g4dn.metal • g4dn.16xlarge • g4dn.12xlarge	• 0,1,48,49 • 0,1,32,33 • 0,1,24,25	• ffffff, ffffff, ffffff • ffffff, ffffff • ffff,ffffffff
P4d	• p4d.24xlarge	• 0,1,48,49	• ffffff, ffffff, ffffff
p3dn	• p3dn.24xlarge	• 0,1,48,49	• ffffff, ffffff, ffffff

DiGif 접촉 받기 위한 준비

1. 원하는 데이터 흐름에 대한 CPU 핵심 계획을 검토하고 에이전트가 사용할 수 있는 코어 목록을 제공하십시오. CPU코어 플래닝을 참조하세요.
2. AWS Ground Station 에이전트 구성 파일을 검토하십시오. AWS Ground Station 에이전트 구성을 참조하세요.
3. 필요한 성능 조정이 적용되었는지 확인합니다. EC2인스턴스 성능 조정을 참조하세요.
4. 언급된 모든 모범 사례를 따르고 있는지 확인하십시오. 모범 사례을 참조하세요.
5. 다음을 통해 예약된 연락 시작 시간 이전에 AWS Ground Station 상담원이 시작되었는지 확인하십시오.

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

6. 다음을 통해 예정된 연락 시작 시간 전에 AWS Ground Station 상담원이 정상 상태인지 확인합니다.

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

   귀하의 i가 `agentStatus` 맞는지, `awsGroundStationAgentEndpoint` 있는지가 ACTIVE 맞는지 확인하십시오HEALTHY. `auditResults`

모범 사례

아마존 EC2 모범 사례

현재 EC2 모범 사례를 따르고 충분한 데이터 스토리지 가용성을 보장하십시오.

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

Linux 스케줄러

Linux 스케줄러는 해당 프로세스가 특정 코어에 고정되어 있지 않은 경우 UDP 소켓의 패킷을 재정렬할 수 있습니다. 데이터를 보내거나 받는 모든 스레드는 UDP 데이터 전송 기간 동안 자신을 특정 코어에 고정해야 합니다.

AWS Ground Station 관리형 프리픽스 목록

안테나와의 통신을 허용하는 네트워크 규칙을 지정할 때는 `com.amazonaws.global.groundstation` AWS -managed 접두사 목록을 활용하는 것이 좋습니다. 관리형 접두사 목록에 대한 자세한 내용은 AWS 관리형 접두사 목록 사용을 AWS 참조하십시오.

단일 접점 제한

AWSGround Station Agent는 접점당 여러 스트림을 지원하지만 한 번에 하나의 접점만 지원합니다. 일정 문제를 방지하려면 여러 데이터 흐름 엔드포인트 그룹에서 인스턴스를 공유하지 마세요. 단일 에이전트 구성이 여러 개의 다른 DFEG ARNs 구성과 연결된 경우 등록에 실패합니다.

AWS Ground Station 에이전트와 함께 서비스 및 프로세스 실행

에이전트와 동일한 EC2 인스턴스에서 서비스와 프로세스를 시작할 때는 AWS Ground Station AWS Ground Station 에이전트와 Linux 커널에서 vCPUs 사용하지 않도록 바인딩하는 것이 중요합니다. 이렇게 하면 병목 현상이 발생하고 연결 중에 데이터가 손실될 수도 있기 때문입니다. 특정 항목에 대한 이러한 바인딩 개념을 vCPUs 어피니티라고 합니다.

피해야 할 코어:

- `agentCpuCores`에서 에이전트 구성 파일

- 하드웨어 인터럽트 및 수신 대기열 조정 (영향 및 네트워크) CPU의 `interrupt_core_list`
  - 기본값은 다음에서 찾을 수 있습니다. 부록: 인터럽트/튜닝을 위한 권장 파라미터 RPS

예를 들어, **c5.24xlarge** 인스턴스를 사용해

다음과 같이 지정한 경우

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

그리고 실행했습니다.

```
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
```

그런 다음 다음 코어는 피하십시오.

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

어피니타이징 서비스 (시스템드)

새로 출시된 서비스는 앞서 언급한 서비스와 자동으로 조화를 이룹니다. `interrupt_core_list` 출시된 서비스의 사용 사례에 추가 코어가 필요하거나 덜 혼잡한 코어가 필요한 경우에는 이 섹션을 따르세요.

다음 명령을 사용하여 서비스가 현재 어떤 어피니티로 구성되어 있는지 확인하세요.

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

다음과 같은 빈 값이 표시되면 위 `CPUAffinity=` 명령의 기본 코어를 사용할 가능성이 높다는 뜻입니다. `...bin/set_irq_affinity.sh <using the cores here> ...`

특정 어피니티를 오버라이드하고 설정하려면 다음을 실행하여 서비스 파일의 위치를 찾으세요.

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

vinano, 등을 사용하여 파일을 열고 수정한 다음 다음과 같은 CPUAffinity=<core list> [Service] 섹션에 입력합니다.

```
[Unit]
...

[Service]
...
CPUAffinity=2,3

[Install]
...
```

파일을 저장하고 서비스를 다시 시작하여 다음을 사용하여 어피니티를 적용합니다.

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

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

자세한 내용은 Red Hat Enterprise Linux 8 - 커널 관리, 모니터링 및 업데이트 - 27장을 참조하십시오. systemd를 사용하여 CPU 어피니티 및 NUMA 정책 구성.

어피니타이징 프로세스 (스크립트)

새로 시작한 스크립트와 프로세스는 수동으로 초기화하는 것이 좋습니다. 기본 Linux 동작에서는 시스템의 모든 코어를 사용할 수 있기 때문입니다.

실행 중인 프로세스 (예: python, bash 스크립트 등) 의 핵심 충돌을 방지하려면 다음을 사용하여 프로세스를 시작하십시오.

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

프로세스가 이미 실행 중인 경우 pidoftop, 또는 ps 같은 명령을 사용하여 특정 프로세스의 프로세스 ID (PID) 를 찾으십시오. 를 PID 사용하면 다음과 같은 현재 친화도를 확인할 수 있습니다.

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

다음과 같이 수정할 수 있습니다.

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

작업 세트에 대한 자세한 내용은 taskset - Linux 매뉴얼 페이지를 참조하십시오.

문제 해결

에이전트 시작 실패

여러 가지 이유로 AWS Ground Station 에이전트가 시작되지 않을 수 있지만 가장 일반적인 시나리오는 잘못 구성된 에이전트 구성 파일일 수 있습니다. 에이전트를 시작한 후(AWS Ground Station 에이전트 시작 참조) 다음과 같은 상태가 표시될 수 있습니다.

```
#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)
```

문제 해결

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

다음과 같은 결과가 출력될 수 있습니다.

```
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” 이후에 에이전트를 시작하지 못하면 에이전트 구성에 문제가 있는 것입니다. 에이전트 구성을 확인하려면 에이전트 구성 파일을 참조하세요.

AWS Ground Station 에이전트 로그

AWS Ground Station 에이전트는 연락처 실행, 오류 및 상태에 대한 정보를 에이전트를 실행하는 인스턴스의 로그 파일에 기록합니다. 인스턴스에 수동으로 연결하여 로그 파일을 볼 수 있습니다.

다음 위치에서 인스턴스에 대한 로그를 볼 수 있습니다.

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

연락처가 없습니다.

연락처를 예약하려면 건강한 AWS Ground Station 상담원이 필요합니다. 다음을 통해 문의하여 AWS Ground Station 상담원이 시작되었고 정상인지 확인하세요. AWS Ground Station API get-dataflow-endpoint-group

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

귀하의 이름이 맞고 `agentStatus` 있는 것인지 `awsGroundStationAgentEndpoint` ACTIVE 아닌지 확인하십시오. `auditResults` HEALTHY

지원 받기

AWS지원을 통해 Ground Station 팀에 문의하세요.

1. 영향을 받는 모든 접촉에 대해 `contact_id`를 제공하세요. 이 정보가 없으면 AWS Ground Station 팀에서 특정 연락처를 조사할 수 없습니다.
2. 이미 수행한 모든 문제 해결 단계에 대한 세부 정보를 제공하세요.
3. 명령을 실행하는 동안 발견된 오류 메시지를 문제 해결 지침에 제공하세요.

상담원 릴리스 노트

최신 에이전트 버전

버전 1.0.3555.0

출시일: 2024년 3월 27일

지원 종료일: 2024년 8월 31일

RPM체크섬:

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

변경 사항:

- 태스킹 시작 시 선택한 실행 파일 버전에 대한 에이전트 메트릭을 추가합니다.
- 다른 버전을 사용할 수 있는 경우 특정 실행 버전을 사용하지 않도록 구성 파일 지원을 추가하세요.
- 네트워크 및 라우팅 진단을 추가합니다.
- 추가 보안 기능.
- 일부 지표 보고 오류가 로그 파일 대신 stdout/journal에 기록되는 문제를 수정했습니다.
- 네트워크에 연결할 수 없는 소켓 오류를 정상적으로 처리합니다.
- 소스 에이전트와 대상 에이전트 간의 패킷 손실과 지연 시간을 측정합니다.
- aws-gs-datapipe 버전 2.0은 새 프로토콜 기능을 지원하고 연락처를 새 프로토콜로 투명하게 업그레이드할 수 있는 기능을 지원합니다.

더 이상 사용되지 않는 에이전트 버전

버전 1.0.2942.0

출시일: 2023년 6월 26일

지원 종료일: 2024년 5월 31일

RPM체크섬:

- SHA256: 7d94b642577504308a58bab28f938507f2591d4e1b2c7ea170b77bea97b5a9b6
- MD5: 661ff2b8f11aba5d657a6586b56e0d8f

변경 사항:

- 에이전트가 디스크에서 RPM 업데이트되고 변경 내용을 적용하려면 에이전트를 다시 시작해야 하는 경우에 대한 오류 로그가 추가되었습니다.
- 에이전트 사용 설명서 조정 단계를 따르고 올바르게 적용할 수 있도록 네트워크 조정 검증을 추가했습니다.
- 에이전트 로그에서 로그 보관에 대한 잘못된 경고를 발생시킨 버그를 수정합니다.
- 패킷 손실 감지가 개선되었습니다.
- 에이전트가 이미 실행 중인 RPM 경우 설치 또는 업그레이드를 방지하도록 에이전트 설치가 업데이트되었습니다.

버전 1.0.2716.0

출시일: 2023년 3월 15일

지원 종료일: 2024년 5월 31일

RPM체크섬:

- SHA256: cb05b6a77dfcd5c66d81c0072ac550affbcefefc372cc5562ee52fb220844929
- MD5: 65266490c4013b433ec39ee50008116c

변경 사항:

- 에이전트가 태스킹 중에 실패를 경험하는 경우 로그 업로드를 활성화하세요.
- 제공된 네트워크 조정 스크립트에서 Linux 호환성 버그를 수정합니다.

버전 1.0.2677.0

출시일: 2023년 2월 15일

지원 종료일: 2024년 5월 31일

RPM체크섬:

- SHA256: 77cfe94acb00af7ca637264b17c9b21bd7afdc85b99dffdd627aec9e99397489
- MD5: b8533be7644bb4d12ab84de21341adac

변경 사항:

- 정식 버전으로 제공되는 첫 번째 에이전트 릴리스.

RPM설치 검증

최신 RPM 버전, 에서 RPM 검증한 MD5 해시, sha256sum을 사용한 SHA256 해시가 아래에 나와 있습니다. 이 값을 조합하면 그라운드 스테이션 에이전트에 사용되는 RPM 버전을 검증하는 데 사용할 수 있습니다.

최신 에이전트 버전

버전 1.0.3555.0

출시일: 2024년 3월 27일

지원 종료일: 2024년 8월 31일

RPM체크섬:

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

변경 사항:

- 태스킹 시작 시 선택한 실행 파일 버전에 대한 에이전트 메트릭을 추가합니다.
- 다른 버전을 사용할 수 있는 경우 특정 실행 버전을 사용하지 않도록 구성 파일 지원을 추가하세요.
- 네트워크 및 라우팅 진단을 추가합니다.
- 추가 보안 기능.
- 일부 지표 보고 오류가 로그 파일 대신 stdout/journal에 기록되는 문제를 수정했습니다.
- 네트워크에 연결할 수 없는 소켓 오류를 정상적으로 처리합니다.
- 소스 에이전트와 대상 에이전트 간의 패킷 손실과 지연 시간을 측정합니다.
- aws-gs-datapipe 버전 2.0은 새 프로토콜 기능을 지원하고 연락처를 새 프로토콜로 투명하게 업그레이드할 수 있는 기능을 지원합니다.

다음을 확인하십시오. RPM

이 RPM 설치를 확인하는 데 필요한 도구는 다음과 같습니다.

- sha256sum
- RPM

두 도구 모두 Amazon Linux 2에서 기본적으로 제공됩니다. 이러한 도구는 사용 중인 버전이 올바른지 확인하는 데 RPM 도움이 됩니다. 먼저 S3 RPM 버킷에서 최신 버전을 다운로드합니다 (다운로드 에이전트 다운로드 지침은 참조RPM). 이 파일이 다운로드되면 몇 가지 확인해야 할 사항이 있습니다.

- 파일의 sha256sum을 계산합니다. RPM 사용 중인 컴퓨팅 인스턴스의 명령줄에서 다음 작업을 수행합니다.

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

  이 값을 가져와 위 표와 비교하세요. 이는 다운로드된 RPM 파일이 AWS Ground Station에서 고객에게 판매한 유효한 파일임을 나타냅니다. 해시가 일치하지 않는 경우 를 설치하지 말고 컴퓨팅 인스턴스에서 삭제하십시오. RPM
- 파일의 MD5 해시도 확인하여 파일이 손상되지 RPM 않았는지 확인하십시오. 이 작업을 수행하려면 다음 RPM 명령을 실행하여 명령줄 도구를 사용하십시오.

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

  여기에 나열된 MD5 해시가 위 표에 있는 버전의 MD5 해시와 동일한지 확인하십시오. AWSDocs에 나열된 이 표에 대해 이 두 해시가 모두 검증되면 고객은 다운로드하여 설치한 해시가 안전하고 손상되지 RPM 않은 버전인지 확인할 수 있습니다. RPM

AWS Ground Station 에이전트 사용 설명서의 문서 기록

다음 표에는 AWS Ground Station 에이전트 사용 설명서의 각 릴리스에서 변경된 주요 내용이 설명되어 있습니다.

변경 사항	설명	날짜
설명서 업데이트	에이전트 요구 사항에 서브넷과 Amazon EC2 인스턴스를 동일한 가용 영역에 유지하는 것에 대한 설명이 추가되었습니다.	2024년 7월 18일
설명서 업데이트	AWS Ground Station 에이전트를 자체 사용 설명서로 나눕니다. 이전 변경 사항은 AWSGround Station 사용 설명서의 문서 기록을 참조하십시오.	2024년 7월 18일

기계 번역으로 제공되는 번역입니다. 제공된 번역과 원본 영어의 내용이 상충하는 경우에는 영어 버전이 우선합니다.