



AWS Ground Station 에이전트 사용 설명서

AWS Ground Station



AWS Ground Station: AWS Ground Station 에이전트 사용 설명서

Copyright © 2026 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon의 상표 및 트레이드 드레스는 Amazon 외 제품 또는 서비스와 함께, Amazon 브랜드 이미지를 떨어뜨리거나 고객에게 혼동을 일으킬 수 있는 방식으로 사용할 수 없습니다. Amazon이 소유하지 않은 기타 모든 상표는 Amazon과 제휴 관계이거나 관련이 있거나 후원 관계와 관계없이 해당 소유자의 자산입니다.

Table of Contents

개요	1
AWS Ground Station 에이전트란 무엇입니까?	1
AWS Ground Station 에이전트의 기능	2
에이전트 요구 사항	3
VPC 다이어그램	4
지원되는 운영 체제	5
AWS Ground Station 에이전트를 통해 데이터 수신	6
다중 데이터 흐름, 단일 수신기	6
여러 데이터 흐름, 여러 수신기	7
Amazon EC2 인스턴스를 선택하고 아키텍처에 대한 CPU 코어를 예약합니다.	9
지원되는 Amazon EC2 인스턴스 유형	9
CPU 코어 계획	10
아키텍처 정보 수집	11
CPU 할당 예제	13
부록: c5.24xlarge의 <code>lscpu -p</code> 출력(전체)	13
에이전트 설치	17
CloudFormation 템플릿 사용	17
1단계: AWS 리소스 생성	17
2단계: 에이전트 상태 확인	17
EC2에 수동으로 설치	17
1단계: AWS 리소스 생성	17
2단계: EC2 인스턴스 생성	18
3단계: 에이전트 다운로드 및 설치	18
4단계: 에이전트 구성	19
5단계: 성능 조정 적용	19
6단계: 에이전트 관리	20
에이전트 관리	21
AWS Ground Station 에이전트 구성	21
AWS Ground Station 에이전트 시작	21
AWS Ground Station 에이전트 중지	22
AWS Ground Station 에이전트 업그레이드	22
AWS Ground Station 에이전트 다운그레이드	23
AWS Ground Station 에이전트 제거	24
AWS Ground Station 에이전트 상태	24

AWS Ground Station 에이전트 RPM 정보	25
에이전트 구성	26
에이전트 구성 파일	26
예제	26
필드 분석	26
성능을 위해 EC2 인스턴스 튜닝	30
하드웨어 인터럽트 및 수신 대기열 튜닝 - CPU 및 네트워크에 영향을 미칩니다.	30
Rx 인터럽트 병합 조정 - 네트워크에 영향을 미침	31
Rx 링 버퍼 튜닝 - 네트워크에 영향을 줍니다.	32
CPU C 상태 조정 - CPU에 영향을 미침	32
수신 포트 예약 - 네트워크에 영향을 미칩니다.	32
재부팅	33
부록: 인터럽트/RPS 튜닝을 위한 권장 파라미터	33
DiGif 접촉 받기 위한 준비	35
모범 사례	36
Amazon EC2 모범 사례	36
Linux 스케줄러	36
AWS Ground Station 관리형 접두사 목록	36
단일 접점 제한	36
AWS Ground Station 에이전트와 함께 서비스 및 프로세스 실행	36
c5.24xlarge 인스턴스를 사용하는 예제로	37
서비스 선호(시스템)	37
프로세스 선호(스크립트)	38
문제 해결	40
에이전트 시작 실패	40
문제 해결	40
AWS Ground Station 에이전트 로그	41
사용 가능한 연락처 없음	41
지원 받기	42
에이전트 릴리스 정보	43
최신 에이전트 버전	43
버전 1.0.4382.0	43
더 이상 사용되지 않는 에이전트 버전	43
버전 1.0.3555.0	43
버전 1.0.2942.0	44
버전 1.0.2716.0	44

버전 1.0.2677.0	45
RPM 설치 검증	46
최신 에이전트 버전	43
버전 1.0.4382.0	43
RPM 확인	46
문서 기록	48
.....	xlix

개요

AWS Ground Station 에이전트란 무엇입니까?

RPM으로 사용할 수 있는 AWS Ground Station 에이전트를 사용하면 AWS Ground Station 고객 응대 중에 동기 광대역 디지털 중간 주파수(DigIF) 데이터 흐름을 수신(다운링크)할 수 있습니다. 데이터 전송을 위한 두 가지 옵션을 선택할 수 있습니다.

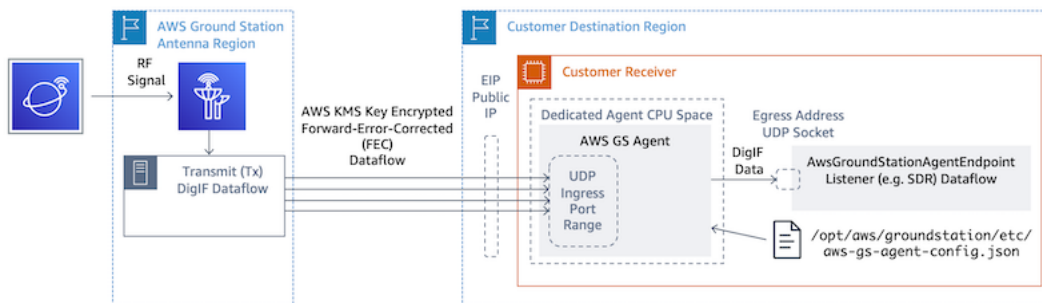
1. EC2 인스턴스로 데이터 전송 - 소유한 EC2 인스턴스로 데이터 전송. AWS Ground Station 에이전트를 관리합니다. 이 옵션은 실시간에 가까운 데이터 처리가 필요한 경우에 가장 적합할 수 있습니다. EC2 [데이터 전송에 대한 자세한 내용은 Amazon Elastic Compute Cloud로](#) 데이터 전송 가이드를 참조하세요.
2. S3 버킷으로 데이터 전송 - Ground Station 관리형 서비스를 통해 소유한 AWS S3 버킷으로 데이터 전송. S3 데이터 전송에 대한 자세한 내용은 [시작하기 AWS Ground Station](#) 가이드를 참조하세요.

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

Note

S3 데이터 전송은 Ground Station의 관리형 서비스이므로 이 안내서에서는 EC2 인스턴스로의 데이터 전송에 중점을 둡니다.

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



AWS Ground Station 에이전트의 기능

AWS Ground Station 에이전트는 디지털 중간 주파수(DigiF) 다운로드 데이터를 수신하고 복호화된 데이터를 내보내 다음을 활성화합니다.

- 40MHz ~ 400MHz 대역폭의 DigiF 다운로드 기능을 제공합니다.
- AWS 네트워크의 모든 퍼블릭 IP(AWS 탄력적 IP)로 고속, 낮은 지터 DigiF 데이터 전송.
- 순방향 오류 수정(FEC)을 사용한 안정적인 데이터 전송.
- 암호화를 위해 고객 관리형 AWS KMS 키를 사용하여 데이터를 안전하게 전송합니다.

에이전트 요구 사항

Note

이 AWS Ground Station 에이전트 안내서에서는 [AWS Ground Station 시작하기](#) 안내서를 사용하여 Ground Station에 온보딩했다고 가정합니다.

AWS Ground Station 에이전트 수신기 EC2 인스턴스는 DigIF 데이터를 엔드포인트에 안정적이고 안전하게 전송하기 위해 종속 AWS 리소스 세트가 필요합니다.

1. EC2 수신기를 시작할 VPC입니다.
2. 데이터 암호화/복호화를 위한 AWS KMS 키.
3. [SSM 세션 관리자](#)용으로 구성된 SSH 키 또는 EC2 인스턴스 프로필.
4. 네트워크/보안 그룹 규칙은 다음을 허용합니다.
 1. 데이터 흐름 엔드포인트 그룹에 지정된 포트에서 AWS Ground Station의 UDP 트래픽입니다. 에이전트는 데이터를 수신 데이터 흐름 엔드포인트로 전달하는 데 사용되는 다양한 연속 포트를 예약합니다.
 2. 인스턴스에 대한 SSH 액세스(참고: 대안으로 AWS 세션 관리자를 사용하여 EC2 인스턴스에 액세스할 수도 있음).
 3. 에이전트 관리를 위해 공개적으로 액세스할 수 있는 S3 버킷에 대한 읽기 액세스.
 4. 포트 443의 SSL 트래픽을 통해 에이전트가 AWS Ground Station 서비스와 통신할 수 있습니다.
 5. AWS Ground Station 관리형 접두사 목록의 트래픽입니다. `com.amazonaws.global.groundstation`.

또한 퍼블릭 서브넷을 포함하는 VPC 구성이 필요합니다. 서브넷 구성의 배경은 [VPC 사용 설명서](#) 섹션을 참조하십시오.

호환되는 구성:

1. 퍼블릭 서브넷에 있는 EC2 인스턴스와 연결된 탄력적 IP입니다.
2. EC2 인스턴스(퍼블릭 서브넷과 동일한 가용 영역의 모든 서브넷)에 연결된 퍼블릭 서브넷의 ENI와 연결된 탄력적 IP입니다.

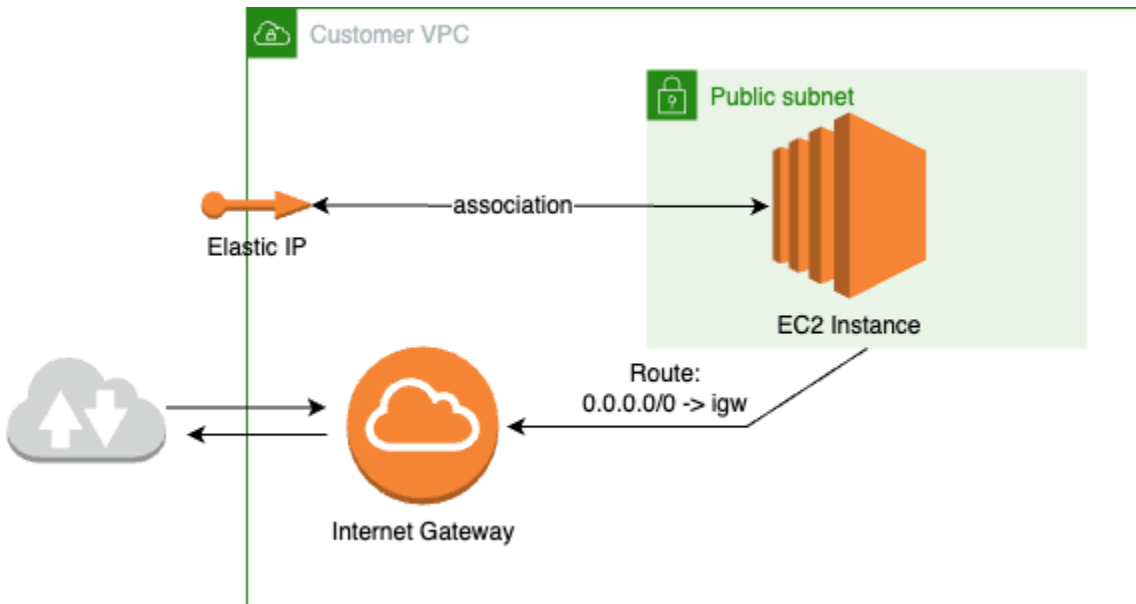
EC2 인스턴스와 동일한 보안 그룹을 사용하거나 최소한 다음으로 구성된 최소 규칙 세트를 갖춘 보안 그룹을 지정할 수 있습니다.

- 데이터 흐름 엔드포인트 그룹에 지정된 포트에서 AWS Ground Station 의 UDP 트래픽입니다.

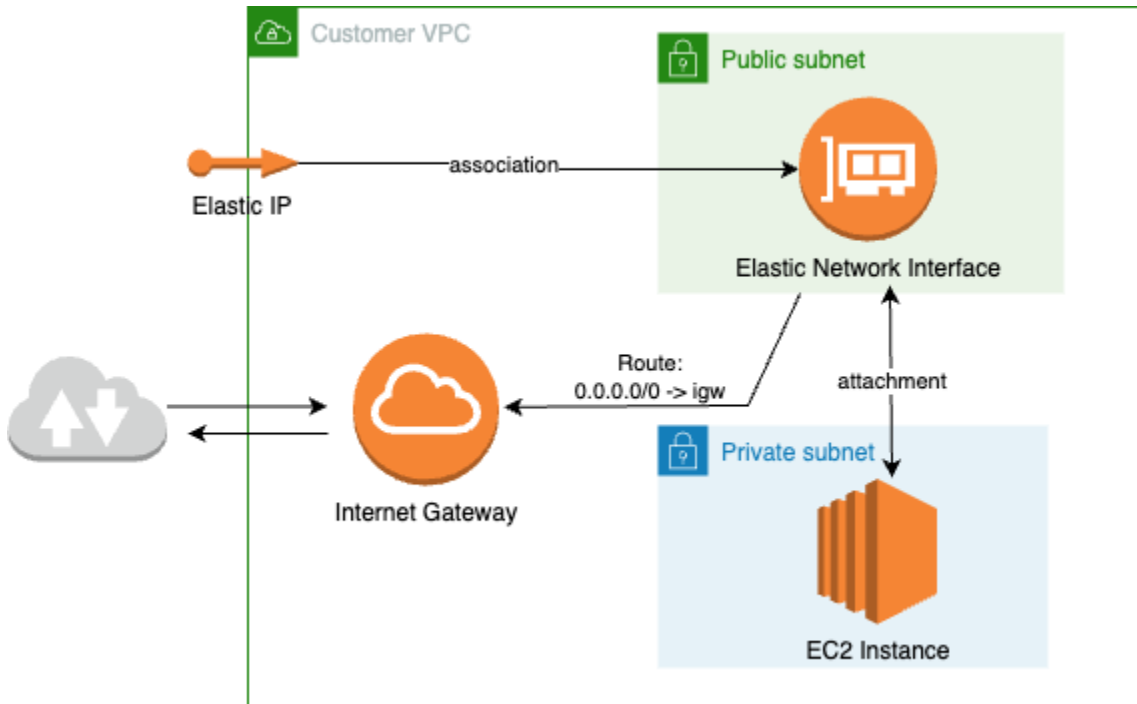
예를 들어 이러한 리소스가 미리 구성된 CloudFormation EC2 데이터 전송 템플릿은 [AWS Ground Station 에이전트\(와이드밴드\)를 활용하는 퍼블릭 브로드캐스트 위성을 참조하세요.](#)

VPC 다이어그램

다이어그램: 퍼블릭 서브넷에 있는 EC2 인스턴스와 연결된 탄력적 IP입니다



다이어그램: 퍼블릭 서브넷에 있는 ENI와 연결된 탄력적 IP로, 프라이빗 서브넷에서 EC2 인스턴스에 연결되어 있습니다



지원되는 운영 체제

Amazon Linux 2(커널 5.10+)

지원되는 인스턴스 유형은에 나열되어 있습니다. [Amazon EC2 인스턴스를 선택하고 아키텍처에 대한 CPU 코어를 예약합니다.](#)

AWS Ground Station 에이전트를 통해 데이터 수신

아래 다이어그램은 광대역 디지털 중간 주파수(DigIF) 고객 응대 AWS Ground Station 중에 데이터가 흐르는 방식에 대한 개요를 제공합니다.

AWS Ground Station 에이전트는 고객 응대의 데이터 영역 구성 요소 오케스트레이션을 처리합니다. 고객 응대를 예약하기 전에 에이전트를 올바르게 구성하고 시작하고 등록(에이전트 시작 시 등록이 자동)해야 합니다 AWS Ground Station. 또한 데이터 수신 소프트웨어(예: 소프트웨어 정의 라디오)는 [AwsGroundStationAgentEndpoint](#) egressAddress에서 데이터를 수신하도록 실행 중이고 구성되어 있어야 합니다.

백그라운드에서 AWS Ground Station 에이전트는 전송 중에 적용된 AWS KMS 암호화를 수신 AWS Ground Station 하고 실행 취소한 후 소프트웨어 정의 라디오(SDR)가 수신 대기 중인 대상 엔드포인트 egressAddress로 전달합니다. AWS Ground Station 에이전트와 기본 구성 요소는 구성 파일에 설정된 CPU 경계를 준수하여 인스턴스에서 실행되는 다른 애플리케이션의 성능에 영향을 주지 않도록 합니다.

연락처와 관련된 수신자 인스턴스에서 AWS Ground Station 에이전트를 실행해야 합니다. 단일 수신기 인스턴스에서 모든 데이터 흐름을 수신하려는 경우 단일 AWS Ground Station 에이전트는 아래와 같이 여러 데이터 흐름을 오케스트레이션할 수 있습니다.

다중 데이터 흐름, 단일 수신기

예제 시나리오:

동일한 EC2 수신기 인스턴스에서 두 개의 안테나 다운링크를 DigIF 데이터 흐름으로 수신하려고 합니다. 두 개의 다운링크는 200MHz와 100MHz입니다.

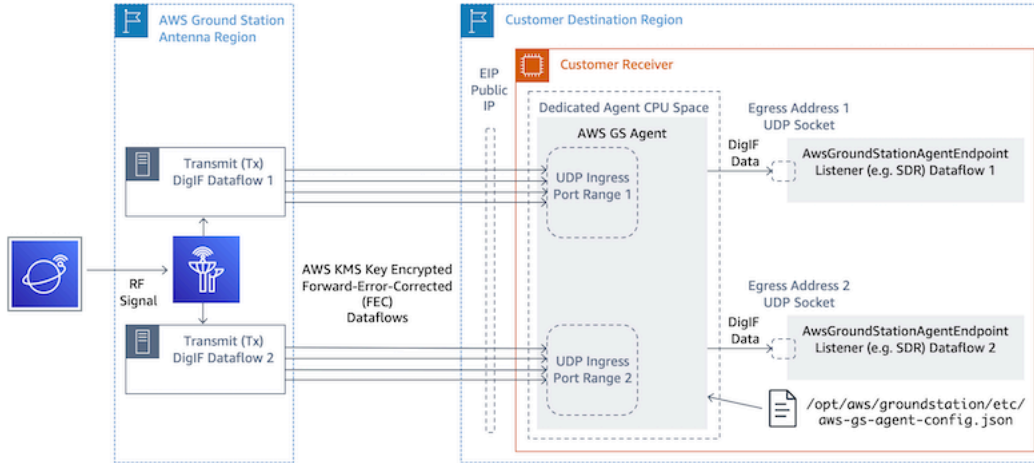
AwsGroundStationAgentEndpoints:

각 데이터 흐름에 하나씩, 총 두 개의 AwsGroundStationAgentEndpoint 리소스가 있습니다. 두 엔드포인트는 동일한 퍼블릭 IP 주소(ingressAddress.socketAddress.name)를 갖게 됩니다. 데이터 흐름이 동일한 EC2 인스턴스에서 수신되므로 수신 portRange이 겹치지 않아야 합니다. egressAddress.socketAddress.port의는 모두 고유해야 합니다.

CPU 계획:

- 인스턴스에서 단일 AWS Ground Station 에이전트를 실행하기 위한 코어 1개(vCPU 2개).

- DigIF Dataflow 1(CPU 코어 계획테이블에서 200MHz 조회)을 수신할 코어 6개(vCPU 12개).
- DigIF Dataflow 2(CPU 코어 계획테이블에서 100MHz 조회)를 수신할 코어 4개(vCPU 8개).
- 총 전용 에이전트 CPU 공간 = 동일한 소켓의 코어 11개(vCPU 22개).



여러 데이터 흐름, 여러 수신기

예제 시나리오

서로 다른 EC2 수신기 인스턴스에서 안테나 다운링크 2개를 DigIF 데이터 흐름으로 수신하려고 합니다. 두 다운링크 모두 400MHz가 됩니다.

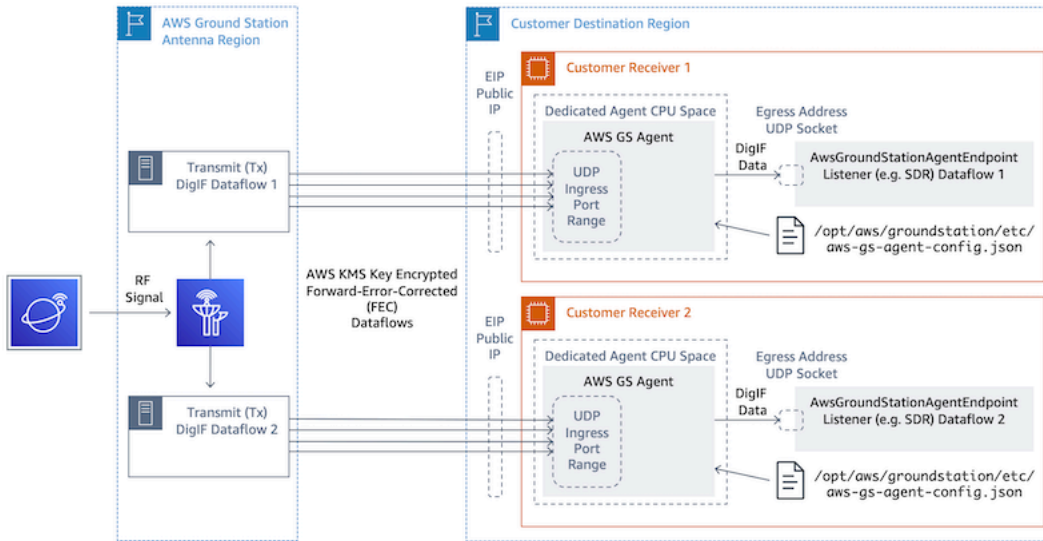
AwsGroundStationAgentEndpoints

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 egressAddress에 대한 포트 값에는 제한이 없습니다.

CPU 계획:

- 수신기 인스턴스 1
 - 인스턴스에서 단일 AWS Ground Station 에이전트를 실행하기 위한 코어 1개(vCPU 2개).
 - DigIF Dataflow 1(CPU 코어 계획테이블에서 400MHz 조회)을 수신할 코어 9개(vCPU 18개).
 - 총 전용 에이전트 CPU 공간 = 동일한 소켓의 코어 10개(vCPU 20개).
- 수신기 인스턴스 2

- 인스턴스에서 단일 AWS Ground Station 에이전트를 실행하기 위한 코어 1개(vCPU 2개).
- DigIF Dataflow 2(CPU 코어 계획 테이블에서 400MHz 조회)를 수신할 코어 9개(vCPU 18개).
- 총 전용 에이전트 CPU 공간 = 동일한 소켓의 코어 10개(vCPU 20개).



Amazon EC2 인스턴스를 선택하고 아키텍처에 대한 CPU 코어를 예약합니다.

지원되는 Amazon EC2 인스턴스 유형

AWS Ground Station 에이전트는 컴퓨팅 집약적인 데이터 전송 워크플로로 인해 전용 CPU 코어가 필요합니다. 다음 인스턴스 유형을 지원합니다. 사용 사례에 가장 적합한 인스턴스 유형을 [CPU 코어 계획](#) 결정하려면 참조하세요.

인스턴스 패밀리	인스턴스 유형	기본 vCPU	기본 CPU 코어	최대 DigIF 집계 대역폭(MHz)
C5	c5.12xlarge	48	24	180
	c5.18xlarge	72	36	380
	c5.24xlarge	96	48	380
C5n	c5n.18xlarge	72	36	400
	c5n.metal	72	36	400
c6i	c6i.24xlarge	96	48	400
	c6i.32xlarge	128	64	400
c7i	c7i.12xlarge	48	24	280
	c7i.24xlarge	96	48	400
p3dn	p3dn.24xlarge	96	48	400
g4dn	g4dn.12xlarge	48	24	400
	g4dn.16xlarge	64	32	400
	g4dn.metal	96	48	400
P4d	p4d.24xlarge	96	48	400

인스턴스 패밀리	인스턴스 유형	기본 vCPU	기본 CPU 코어	최대 DigIF 집계 대역폭(MHz)
m5	m5.8xlarge	32	16	100
	m5.12xlarge	48	24	180
	m5.24xlarge	96	48	380
m6i	m6i.32xlarge	128	64	400
r5	r5.24xlarge	96	48	380
	r5.metal	96	48	380
R5n	r5n.24xlarge	96	48	400
	r5n.metal	96	48	400
r6i	r6i.32xlarge	128	64	400

Note

최대 DigIF 집계 대역폭 열에는 각 인스턴스 유형에 결합된 모든 DigIF 데이터 흐름에 대해 지원되는 최대 집계 대역폭이 표시됩니다. 이러한 제한은 지정된 인스턴스 유형에 할당된 EC2 네트워크 용량 때문입니다. 이러한 값은 보수적인 추정치를 나타내며 DigIF 구성을 계획할 때 사용해야 합니다. 실제 대역폭은 시스템 부하 및 기타 요인에 따라 달라질 수 있습니다.

CPU 코어 계획

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 페어는 단일 코어 내에 포함된 가상 CPUs(vCPU) 페어입니다. 다음 표에서는 단일 데이터 흐름에 대해 에이전트용으로 예약된 필요한 코어 수에 데이터 흐름 데이터 속도를 매핑합니다. 이 표는 Cascade Lake 이상의 CPUs 가정하며 지원되는 모든 인스턴스 유형에 유효합니다. 대역폭이 테이블의 항목 사이에 있는 경우 다음으로 높은 항목을 선택합니다.

에이전트에는 관리 및 조정을 위한 추가 예약 코어가 필요하므로 필요한 총 코어는 각 데이터 흐름에 필요한 코어(아래 차트에서)와 단일 추가 코어(vCPUs)의 합계입니다.

안테나 다운로드 대역폭(MHz)	예상 VITA-49.2 DigIF 데이터 속도(Mb/s)	코어 개수 (HT CPU 페어)	총 vCPU
50	1000	3	6
100	2000	4	8
150	3000	5	10
200	4000	6	12
250	5000	6	12
300	6000	7	14
350	7000	8	16
400	8000	9	18

아키텍처 정보 수집

`lscpu`는 시스템의 아키텍처에 대한 정보를 제공합니다. 기본 출력은 어떤 vCPUs("CPU"로 레이블 지정됨)가 어떤 NUMA 노드(및 각 NUMA 노드가 L3 캐시를 공유함)에 속하는지 보여줍니다. 아래에서는 AWS Ground Station 에이전트를 구성하는 데 필요한 정보를 수집하기 위해 `c5.24xlarge` 인스턴스를 검사합니다. 여기에는 vCPUs 수, 코어 및 vCPU-to-node 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 코어에 대한 두 vCPUs 모두 포함되어야 합니다. 데이터 흐름의 모든 코어는 동일한 NUMA 노드에 있어야 합니다. `lscpu` 명령 `-p` 옵션은 에이전트를 구성하는 데 필요한 코어-CPU 연결을 제공합니다. 관련 필드는 CPU(vCPU라고 함), 코어 및 L3(해당 코어에서 공유되는 L3 캐시를 나타냄)입니다. 대부분의 Intel 프로세서에서 NUMA 노드는 L3 캐시와 동일합니다.

에 대한 `lscpu -p` 출력의 다음 하위 집합을 고려합니다 `c5.24xlarge`(명확성을 위해 축약되고 형식이 지정됨).

```

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 0에 vCPUs 16이 포함되어 있고, 코어 1에는 vCPUs 1 및 17이 포함되어 있으며, 코어 2에는 vCPUs 2 및 18이 포함되어 있음을 알 수 있습니다. 즉, 하이퍼스레드 페어는 0과 16, 1과 17, 2와 18입니다.

CPU 할당 예제

예를 들어, 350MHz에서 이중 극성 광대역 다운링크에 c5.24xlarge 인스턴스를 사용합니다. 이 표에서 350MHz 다운링크에는 단일 데이터 흐름에 8개의 코어(16개의 vCPU)가 필요하다는 것을 [CPU 코어 계획](#) 알 수 있습니다. vCPUs 즉, 2개의 데이터 흐름을 사용하는 이중 극성 설정에는 에이전트에 대해 총 16개의 코어(32vCPUs)와 1개의 코어(2vCPUs)가 필요합니다.

의 `lscpu` 출력에는 NUMA node0 CPU(s): 0-23, 48-71 및가 c5.24xlarge 포함됩니다. NUMA node1 CPU(s): 24-47, 72-95. NUMA node0에는 필요한 것보다 더 많은 값이 있으므로 0-23 및 48-71 코어에서만 할당합니다.

먼저 L3 캐시 또는 NUMA 노드를 공유하는 각 데이터 흐름에 대해 8개의 코어를 선택합니다. 그런 다음의 `lscpu -p` 출력에서 해당 vCPUs('CPU' 레이블 지정)를 조회합니다. [부록: c5.24xlarge의 lscpu -p 출력\(전체\)](#). 예제 코어 선택 프로세스는 다음과 같을 수 있습니다.

- OS에 대해 코어 0-1을 예약합니다.
- 흐름 1: vCPUs 2-9를 선택합니다.
- 흐름 2: vCPUs 10-17 및 58-65에 매핑되는 코어 10-17을 선택합니다.
- 에이전트 코어: vCPUs 18을 선택합니다.

이로 인해 vCPUs 2~18 및 50~66이 생성되므로 에이전트를 제공할 목록은입니다 [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 대로 자체 프로세스가 이러한 CPUs에서 실행되고 있지 않은지 확인해야 합니다. [AWS Ground Station 에이전트와 함께 서비스 및 프로세스 실행](#).

이 예제에서 선택한 특정 코어는 다소 임의적입니다. 다른 코어 세트는 각 데이터 흐름에 대한 모든 L3 캐시 공유 요구 사항을 충족하는 한 작동합니다.

부록: c5.24xlarge의 `lscpu -p` 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. CloudFormation 템플릿(권장).
2. Amazon EC2에 수동 설치

CloudFormation 템플릿 사용

EC2 데이터 전송 CloudFormation 템플릿은 EC2 인스턴스에 데이터를 전송하는 데 필요한 AWS 리소스를 생성합니다. 이 CloudFormation 템플릿은 AWS Ground Station 에이전트가 사전 설치된 AWS Ground Station 관리형 AMI를 사용합니다. 그러면 생성된 EC2 인스턴스의 부팅 스크립트가 에이전트 구성 파일을 채우고 필요한 성능 조정([성능을 위해 EC2 인스턴스 튜닝](#))을 적용합니다.

1단계: AWS 리소스 생성

AWS [AWS Ground Station](#) 리소스 스택을 생성합니다.

2단계: 에이전트 상태 확인

기본적으로 에이전트가 구성되고 활성 상태입니다(시작됨). 에이전트 상태를 확인하려면 EC2 인스턴스(SSH 또는 SSM 세션 관리자)에 연결하고 [AWS Ground Station 에이전트 상태](#)를 참조하세요.

EC2에 수동으로 설치

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 않는 경우 AWS 리소스를 수동으로 생성하여 에이전트를 설치할 수 있습니다.

1단계: AWS 리소스 생성

연락처에 필요한 AWS 리소스를 수동으로 설정하는 지침은 [미션 프로파일 구성 예제](#)를 참조하세요.

AwsGroundStationAgentEndpoint 리소스는 AWS Ground Station 에이전트를 통해 DigIF 데이터 흐름을 수신하기 위한 엔드포인트를 정의하며 성공적인 연락을 취하는 데 매우 중요합니다. API 설명서는

[API 참조](#)에 있지만 이 섹션에서는 AWS Ground Station 에이전트와 관련된 개념을 간략하게 설명합니다.

엔드포인트의 `ingressAddress`는 AWS Ground Station 에이전트가 안테나에서 AWS KMS 암호화된 UDP 트래픽을 수신하는 곳입니다. `socketAddress name`는 연결된 EIP에서 가져온 EC2 인스턴스의 퍼블릭 IP입니다. `portRange`는 다른 용도로 예약된 범위 내의 연속 포트는 300개 이상이어야 합니다. 자세한 내용은 [수신 포트 예약 - 네트워크에 영향을 미칩니다](#) 섹션을 참조하세요. 이러한 포트는 수신기 인스턴스가 실행 중인 VPC의 보안 그룹에서 UDP 수신 트래픽을 허용하도록 구성해야 합니다.

엔드포인트의 `egressAddress`는 에이전트가 DigIF 데이터 흐름을 사용자에게 전달하는 곳입니다. 이 위치에 UDP 소켓을 통해 데이터를 수신하는 애플리케이션(예: SDR)이 있어야 합니다.

2단계: EC2 인스턴스 생성

지원되는 AMI는 다음과 같습니다.

1. AWS Ground Station AMI - `groundstation-a12-gs-agent-ami-*` 여기서 *는 AMI가 빌드된 날짜이며 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 에이전트 AMI를 선택하지 않은 경우 이 섹션의 단계를 완료해야 합니다.

에이전트 다운로드

AWS Ground Station 에이전트는 리전별 S3 버킷에서 사용할 수 있으며 `${AWS::Region}s3://groundstation-wb-digif-software-${AWS::Region}/aws-groundstation-agent/latest/amazon_linux_2_x86_64/aws-groundstation-agent.rpm`이 지원되는 AWS Ground Station 콘솔 및 데이터 전송 리전 중 하나를 참조하는 AWS 명령줄(CLI)을 사용하여 지원 EC2 인스턴스에 다운로드할 수 있습니다. [AWS Ground Station](#)

예: 로컬 AWS 리전 us-east-2에서 /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: 로컬로 AWS 리전 us-east-2에서 /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 RPM이에서 판매되었는지 확인하려면의 지침을 AWS Ground Station따르세요 [RPM 설치 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 후 에이전트 구성 파일을 업데이트해야 합니다. [에이전트 구성](#)을(를) 참조하세요.

5단계: 성능 조정 적용

AWS Ground Station 에이전트 AMI: 이전 단계에서 AWS Ground Station 에이전트 AMI를 선택한 경우 다음 성능 튜닝을 적용합니다.

- [하드웨어 인터럽트 및 수신 대기열 튜닝 - CPU 및 네트워크에 영향을 미칩니다.](#)

- [수신 포트 예약 - 네트워크에 영향을 미칩니다.](#)
- [재부팅](#)

기타 AMI: 이전 단계에서 다른 AMI를 선택한 경우 [성능을 위해 EC2 인스턴스 튜닝](#)에 나열된 모든 조정을 적용하고 인스턴스를 재부팅합니다.

6단계: 에이전트 관리

에이전트 상태를 시작, 중지 및 확인하려면 [에이전트 관리](#)를 참조하세요.

에이전트 관리

AWS Ground Station 에이전트는 기본 제공 Linux 명령 도구를 사용하여 에이전트를 구성, 시작, 중지, 업그레이드, 다운그레이드 및 제거하는 다음과 같은 기능을 제공합니다.

주제

- [AWS Ground Station 에이전트 구성](#)
- [AWS Ground Station 에이전트 시작](#)
- [AWS Ground Station 에이전트 중지](#)
- [AWS Ground Station 에이전트 업그레이드](#)
- [AWS Ground Station 에이전트 다운그레이드](#)
- [AWS Ground Station 에이전트 제거](#)
- [AWS Ground Station 에이전트 상태](#)
- [AWS Ground Station 에이전트 RPM 정보](#)

AWS Ground Station 에이전트 구성

/opt/aws/groundstation/etc으로 이동하세요. 이 파일에는 aws-gs-agent-config.json이라는 단일 파일이 들어 있어야 합니다. [에이전트 구성 파일](#) 섹션을 참조하세요

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 중지

```
#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 다운로드](#)을(를) 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 다운로드](#)을(를) 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/aws-gs-agent-config.json`의 이름이 `/opt/aws/groundstation/etc/aws-gs-agent-config.json.rpmsave`로 바뀝니다. 동일한 인스턴스에 에이전트를 다시 설치하면 `aws-gs-agent-config.json`의 기본값이 작성되므로 AWS 리소스에 해당하는 올바른 값으로 업데이트해야 합니다. [에이전트 구성 파일](#)(들) 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 에이전트 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.json`에서 에이전트 구성 파일을 업데이트해야 합니다.

에이전트 구성 파일

예제

```
{
  "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 분석

기능

기능은 Dataflow Endpoint Group Amazon Resource Names으로 지정됩니다.

필수: True

형식: 문자열 배열

- 값: 기능 ARN → 문자열

예시:

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

디바이스

이 필드에는 현재 EC2 “디바이스”를 열거하는 데 필요한 추가 필드가 포함되어 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 탄력적 IP(EIP)

필수: True

형식: 문자열 배열

- 값: IP 주소 → 문자열

예제:

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

agentCPUCores

이는 aws-gs-agent 프로세스를 위해 예약된 가상 코어를 지정합니다. 이 값을 적절하게 [CPU 코어 계획](#) 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, 82  
]
```

networkAdapters

이는 데이터를 수신할 이더넷 어댑터 또는 eNIS에 연결된 인터페이스에 해당합니다.

필수: 거짓

형식: 문자열 배열

- 값: 이더넷 어댑터 이름(ifconfig를 실행하여 찾을 수 있음)

예제:

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

성능을 위해 EC2 인스턴스 튜닝

Note

CloudFormation 템플릿을 사용하여 AWS 리소스를 프로비저닝한 경우 이러한 조정이 자동으로 적용됩니다. AMI를 사용하거나 EC2 인스턴스를 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 튜닝 - CPU 및 네트워크에 영향을 미칩니다.](#)
- [Rx 인터럽트 병합 조정 - 네트워크에 영향을 미칩](#)
- [Rx 링 버퍼 튜닝 - 네트워크에 영향을 줍니다.](#)
- [CPU C 상태 조정 - CPU에 영향을 미칩](#)
- [수신 포트 예약 - 네트워크에 영향을 미칩니다.](#)
- [재부팅](#)

하드웨어 인터럽트 및 수신 대기열 튜닝 - CPU 및 네트워크에 영향을 미칩니다.

이 섹션에서는 시스템, SMP IRQ, 수신 패킷 조정(RPS) 및 수신 흐름 조정(RFS)의 CPU 코어 사용량을 구성합니다. 사용 중인 인스턴스 유형에 따른 권장 설정 세트는 [부록: 인터럽트/RPS 튜닝을 위한 권장 파라미터](#)를 참조하세요.

1. 시스템 프로세스를 에이전트 CPU 코어와 분리하여 고정하세요.
2. 하드웨어 인터럽트 요청을 에이전트 CPU 코어에서 멀리 라우팅합니다.
3. 단일 네트워크 인터페이스 카드의 하드웨어 대기열이 네트워크 트래픽의 병목 현상이 되지 않도록 RPS를 구성하세요.
4. RFS를 구성하여 CPU 캐시 적중률을 높여 네트워크 지연 시간을 줄이세요.

RPM에서 제공하는 `set_irq_affinity.sh` 스크립트는 위의 모든 사항을 자동으로 구성합니다. `crontab`에를 추가하여 부팅할 때마다 적용됩니다.

```
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
```

- `l`를 커널 및 OS용으로 예약된 코어 `interrupt_core_list`로 바꿉니다. 일반적으로 하이퍼 스레드 코어 페어와 함께 첫 번째 및 두 번째 코어입니다. 위에서 선택한 코어와 겹치지 않아야 합니다. (예: 하이퍼 스레드, 96-CPU 인스턴스의 경우 '0,1,48,49').
- `rps_core_mask`는 들어오는 패킷을 처리해야 하는 CPU를 지정하는 16진수 비트 마스크로, 각 숫자는 4개의 CPU를 나타냅니다. 또한 오른쪽부터 시작하여 8자마다 쉼표로 구분해야 합니다. 모든 CPU를 허용하고 캐싱이 밸런싱을 처리하도록 하는 것이 좋습니다.
 - 각 인스턴스 유형에 대한 권장 파라미터 목록을 보려면 [부록: 인터럽트/RPS 튜닝을 위한 권장 파라미터](#)를 참조하세요.
- 96-CPU 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
```

Rx 인터럽트 병합 조정 - 네트워크에 영향을 미침

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. `crontab`에를 추가하여 부팅할 때마다 적용됩니다.

```
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. 일반적으로 EC2 인스턴스에 할당된 기본 네트워크 인터페이스 `eth0`이기 때문입니다.

Rx 링 버퍼 튜닝 - 네트워크에 영향을 줍니다.

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

```
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. 일반적으로 EC2 인스턴스에 할당된 기본 네트워크 인터페이스 `eth0`이기 때문입니다.
- `c6i` 패밀리 인스턴스를 설정하는 경우 명령을 수정하여 링 버퍼를 8192대신 로 설정해야 합니다. `16384`.

CPU C 상태 조정 - CPU에 영향을 미칩

CPU C 상태를 설정하여 연결 시작 중에 패킷 손실을 초래할 수 있는 유힬 상태를 방지하세요. 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" >>/etc/default/grub
echo "GRUB_TIMEOUT=0" >>/etc/default/grub
grub2-mkconfig -o /boot/grub2/grub.cfg
```

수신 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
```

- 예시: `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/RPS 튜닝을 위한 권장 파라미터

이 섹션에서는 조정 섹션 하드웨어 인터럽트 및 수신 대기열 조정 - CPU 및 네트워크에 영향을 미치는 권장 매개 변수 값을 결정합니다.

패밀리	인스턴스 유형	<code>interru pt_core_list</code>	<code>{rps_cor e_mask}</code>
c7i	<ul style="list-style-type: none"> • c7i.24xlarge • c7i.12xlarge 	<ul style="list-style-type: none"> • 0,1,48,49 • 0,1,24,25 	<ul style="list-style-type: none"> • ffffff, ffffff, ffffff • ffff, ffffffff
c6i	<ul style="list-style-type: none"> • c6i.32xlarge 	<ul style="list-style-type: none"> • 0,1,64,65 	<ul style="list-style-type: none"> • ffffff, ffffff, ffffff, ffffff
C5	<ul style="list-style-type: none"> • c5.24xlarge • c5.18xlarge • c5.12xlarge 	<ul style="list-style-type: none"> • 0,1,48,49 • 0,1,36,37 • 0,1,24,25 	<ul style="list-style-type: none"> • ffffff, ffffff, ffffff • ff, ffffff, ffffff • ffff, ffffffff
C5n	<ul style="list-style-type: none"> • c5n.metal • c5n.18xlarge 	<ul style="list-style-type: none"> • 0,1,36,37 • 0,1,36,37 	<ul style="list-style-type: none"> • ff, ffffff, ffffff • ff, ffffff, ffffff

패밀리	인스턴스 유형	$\{\text{interrupt_core_list}\}$	$\{\text{rps_core_mask}\}$
m5	<ul style="list-style-type: none"> m5.24xlarge m5.12xlarge 	<ul style="list-style-type: none"> 0,1,48,49 0,1,24,25 	<ul style="list-style-type: none"> ffffff, ffffff, ffffff ffff, ffffffff
r5	<ul style="list-style-type: none"> r5.metal r5.24xlarge 	<ul style="list-style-type: none"> 0,1,48,49 0,1,48,49 	<ul style="list-style-type: none"> ffffff, ffffff, ffffff ffffff, ffffff, ffffff
R5n	<ul style="list-style-type: none"> r5n.metal r5n.24xlarge 	<ul style="list-style-type: none"> 0,1,48,49 0,1,48,49 	<ul style="list-style-type: none"> ffffff, ffffff, ffffff ffffff, ffffff, ffffff
g4dn	<ul style="list-style-type: none"> g4dn.metal g4dn.16xlarge g4dn.12xlarge 	<ul style="list-style-type: none"> 0,1,48,49 0,1,32,33 0,1,24,25 	<ul style="list-style-type: none"> ffffff, ffffff, ffffff ffffff, ffffff ffff, ffffffff
P4d	<ul style="list-style-type: none"> p4d.24xlarge 	<ul style="list-style-type: none"> 0,1,48,49 	<ul style="list-style-type: none"> ffffff, ffffff, ffffff
p3dn	<ul style="list-style-type: none"> p3dn.24xlarge 	<ul style="list-style-type: none"> 0,1,48,49 	<ul style="list-style-type: none"> ffffff, ffffff, ffffff

DiGif 접촉 받기 위한 준비

1. CPU 코어 계획을 검토하여 원하는 데이터 흐름을 확인하고 에이전트가 사용할 수 있는 코어 목록을 제공하세요. [CPU 코어 계획을\(를\)](#) 참조하세요.
2. AWS Ground Station 에이전트 구성 파일을 검토합니다. [AWS Ground Station 에이전트 구성을\(를\)](#) 참조하세요.
3. 필요한 성능 조정이 적용되었는지 확인합니다. [성능을 위해 EC2 인스턴스 튜닝을\(를\)](#) 참조하세요.
4. 호출된 모든 모범 사례를 따르고 있는지 확인합니다. [모범 사례을\(를\)](#) 참조하세요.
5. 다음을 통해 AWS Ground Station 에이전트가 예약된 고객 응대 시작 시간 이전에 시작되었는지 확인합니다.

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

6. 다음을 통해 예정된 고객 응대 시작 시간 전에 AWS Ground Station 에이전트가 정상인지 확인합니다.

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

`awsGroundStationAgentEndpoint`의 `agentStatus`이 활성 상태이고 `auditResults`가 정상 상태인지 확인하세요.

모범 사례

Amazon EC2 모범 사례

현재 EC2 모범 사례를 따르고 충분한 데이터 스토리지 가용성을 보장하세요.

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

Linux 스케줄러

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

AWS Ground Station 관리형 접두사 목록

안테나에서의 통신을 허용하는 네트워크 규칙을 지정할 때는 `com.amazonaws.global.groundstation` AWS에서 관리하는 접두사 목록을 활용하는 것이 좋습니다. AWS 관리형 접두사 목록에 대한 자세한 내용을 알아보려면 설명서의 [AWS 관리형 접두사 목록](#) 작업을 참조하세요.

단일 접점 제한

AWS Ground Station 에이전트는 접촉당 여러 스트림을 지원하지만 한 번에 하나의 접촉만 지원합니다. 일정 문제를 방지하려면 여러 데이터 흐름 엔드포인트 그룹에서 인스턴스를 공유하지 마세요. 단일 에이전트 구성이 여러 개의 서로 다른 DFEG ARN과 연결된 경우 등록에 실패합니다.

AWS Ground Station 에이전트와 함께 서비스 및 프로세스 실행

AWS Ground Station 에이전트와 동일한 EC2 인스턴스에서 서비스 및 프로세스를 시작할 때 AWS Ground Station 에이전트 및 Linux 커널에서 사용하지 않는 vCPUs에 바인딩하는 것이 중요합니다. 이로 인해 고객 응대 중에 병목 현상과 데이터 손실이 발생할 수 있기 때문입니다. 이러한 특정 vCPUs를 선호도라고 합니다.

피해야 할 코어:

- [agentCpuCores](#) 시작 에이전트 구성 파일
- [하드웨어 인터럽트 및 수신 대기열 튜닝 - CPU 및 네트워크에 영향을 미칩니다.](#)의 `interrupt_core_list`
 - 기본값에서 찾을 수 있습니다. [부록: 인터럽트/RPS 튜닝을 위한 권장 파라미터](#)

c5.24xlarge 인스턴스를 사용하는 예제로

지정한 경우

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

및 실행됨

```
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 코어를 피합니다.

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

서비스 선호(시스템)

새로 시작된 서비스는 앞서 `interrupt_core_list` 언급한를 자동으로 확인합니다. 시작된 서비스의 사용 사례에 추가 코어가 필요하거나 혼잡도가 낮은 코어가 필요한 경우 이 섹션을 따르세요.

현재 서비스가 명령으로 구성된 선호도를 확인합니다.

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

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

특정 선호도를 재정의하고 설정하려면 다음을 실행하여 서비스 파일의 위치를 찾습니다.

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

파일을 열고 수정한 다음(vi, nano 등 사용)을 다음과 같이 `CPUAffinity=<core list>` [Service] 섹션에 넣습니다.

```
[Unit]
...

[Service]
...
CPUAffinity=2,3

[Install]
...
```

파일을 저장하고 서비스를 다시 시작하여 다음과 같이 선호도를 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 참조하세요.](#) [systemd를 사용하여 CPU 선호도 및 NUMA 정책 구성.](#)

프로세스 선호(스크립트)

새로 시작된 스크립트와 프로세스를 수동으로 확인하는 것이 좋습니다. 기본 Linux 동작으로 인해 머신의 모든 코어를 사용할 수 있기 때문입니다.

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

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

프로세스가 이미 실행 중인 경우 `pidof`, `top` 또는 등의 명령을 사용하여 특정 프로세스의 프로세스 ID(PID)를 `ps` 찾습니다. PID를 사용하면 다음과 같은 현재 선호도를 확인할 수 있습니다.

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

맞는 다음을 사용하여 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 [작업 세트 - Linux 맨 페이지](#)를 참조하세요.

문제 해결

에이전트 시작 실패

여러 가지 이유로 AWS Ground Station 에이전트를 시작하지 못할 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 파일](#)을 참조하세요.

AWS Ground Station 에이전트 로그

AWS Ground Station 에이전트는 고객 응대 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
```

사용 가능한 연락처 없음

고객 응대를 예약하려면 정상 AWS Ground Station 에이전트가 필요합니다. `get-dataflow-endpoint-group`을 통해 AWS Ground Station API를 쿼리하여 AWS Ground Station 에이전트가 시작되었고 정상인지 확인하세요.

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

`awsGroundStationAgentEndpoint`의 `agentStatus`가 활성 상태이고 `auditResults`가 정상 상태인지 확인하세요.

지원 받기

AWS Support를 통해 Ground Station 팀에 문의하세요.

1. 영향을 받는 모든 접촉에 대해 `contact_id`를 제공하세요. AWS Ground Station 팀은 이 정보가 없으면 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.

에이전트 릴리스 정보

최신 에이전트 버전

버전 1.0.4382.0

릴리스 날짜: 11/18/2025

RPM 체크섬:

- SHA256: 620fd307124f1276194f2faa0104fe0549427ae18e4f5655444f8c30b919c640
- MD5: 73e06dcad44adaccbe2ab005218abfc7

변경 사항:

- 서버에서 오버로드가 표시되면 클라이언트 재시도 동작을 업데이트합니다.

더 이상 사용되지 않는 에이전트 버전

버전 1.0.3555.0

릴리스 날짜: 03/27/2024

RPM 체크섬:

- SHA256: 108f3aceb00e5af549839cd766c56149397e448a6e1e1429c89a9eebb6bc0fc1
- MD5: 65b72fa507fb0af32651adbb18d2e30f

변경 사항:

- 작업 시작 중에 선택한 실행 파일에 대한 에이전트 지표를 추가합니다.
- 다른 버전을 사용할 수 있는 경우 특정 실행 파일을 방지하기 위한 구성 파일 지원을 추가합니다.
- 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 원활하게 처리합니다.
- 소스 에이전트와 대상 에이전트 간의 패킷 손실 및 지연 시간을 측정합니다.
- 새 프로토콜 기능과 연락처를 새 프로토콜로 투명하게 업그레이드하는 기능을 지원하려면 aws-gs-datapipe 버전 2.0을 릴리스합니다.

버전 1.0.2942.0

릴리스 날짜: 06/26/2023

지원 종료일: 05/31/2024

RPM 체크섬:

- SHA256: 7d94b642577504308a58bab28f938507f2591d4e1b2c7ea170b77bea97b5a9b6
- MD5: 661ff2b8f11aba5d657a6586b56e0d8f

변경 사항:

- 에이전트 RPM이 디스크에서 업데이트되고 변경 사항을 적용하려면 에이전트를 다시 시작해야 하는 경우에 대한 오류 로그가 추가되었습니다.
- 에이전트 사용 설명서 튜닝 단계를 따르고 올바르게 적용되도록 네트워크 튜닝 검증을 추가했습니다.
- 에이전트 로그에서 로그 아카이브에 대한 잘못된 경고를 발생시킨 버그를 수정했습니다.
- 패킷 손실 감지를 개선했습니다.
- 에이전트가 이미 실행 중인 경우 RPM의 설치 또는 업그레이드를 방지하도록 에이전트 설치를 업데이트했습니다.

버전 1.0.2716.0

릴리스 날짜: 03/15/2023

지원 종료일: 05/31/2024

RPM 체크섬:

- SHA256: cb05b6a77dfcd5c66d81c0072ac550affbcefefc372cc5562ee52fb220844929
- MD5: 65266490c4013b433ec39ee50008116c

변경 사항:

- 에이전트가 작업 중에 실패할 경우 로그 업로드를 활성화합니다.
- 제공된 네트워크 튜닝 스크립트에서 Linux 호환성 버그를 수정합니다.

버전 1.0.2677.0

릴리스 날짜: 02/15/2023

지원 종료일: 05/31/2024

RPM 체크섬:

- SHA256: 77cfe94acb00af7ca637264b17c9b21bd7afdc85b99dffdd627aec9e99397489
- MD5: b8533be7644bb4d12ab84de21341adac

변경 사항:

- 최초 정식 에이전트 릴리스.

RPM 설치 검증

최신 RPM 버전, RPM에서 검증된 MD5 해시 및 SHA256sum을 사용한 SHA256 해시는 다음과 같습니다. 이 값을 조합하여 Ground Station 에이전트에 사용되는 RPM 버전을 검증할 수 있습니다.

최신 에이전트 버전

버전 1.0.4382.0

릴리스 날짜: 11/18/2025

RPM 체크섬:

- SHA256: 620fd307124f1276194f2faa0104fe0549427ae18e4f5655444f8c30b919c640
- MD5: 73e06dcad44adaccbe2ab005218abfc7

변경 사항:

- 서버에서 오버로드가 표시되면 클라이언트 재시도 동작을 업데이트합니다.

RPM 확인

이 RPM 설치를 확인하는 데 필요한 도구는 다음과 같습니다.

- [sha256sum](#)
- [RPM](#)

두 도구 모두 Amazon Linux 2에서 기본적으로 제공됩니다. 이러한 도구는 사용 중인 RPM이 올바른 버전인지 검증하는 데 도움이 됩니다. 먼저 S3 버킷에서 최신 RPM을 다운로드합니다(RPM 다운로드에 대한 지침은 [에이전트 다운로드](#) 참조). 이 파일이 다운로드되면 몇 가지 확인해야 할 사항이 있습니다.

- RPM 파일의 sha256sum을 계산하세요. 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 제공한 유효한 파일임을 보여줍니다. 해시가 일치하지 않는 경우 RPM을 설치하지 말고 컴퓨팅 인스턴스에서 삭제하세요.

- 파일의 MD5 해시도 확인하여 RPM이 손상되지 않았는지 확인하세요. 또한 AWS 명령줄 도구를 사용해 다음 CLI 명령을 실행하여 이 작업을 수행할 수 있습니다.

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

여기에 나열된 MD5 해시가 위 표에 있는 버전의 MD5 해시와 동일한지 확인하세요. AWS 문서에 나열된 이 표에 대해 두 해시가 모두 검증되면 고객은 다운로드 및 설치된 RPM이 안전하고 손상되지 않은 RPM 버전인지 확인할 수 있습니다.

AWS Ground Station 에이전트 사용 설명서의 문서 기록

다음 표에서는 AWS Ground Station 에이전트 사용 설명서의 각 릴리스에서 변경된 중요 사항에 대해 설명합니다.

변경 사항	설명	날짜
설명서 업데이트	이전 인스턴스 패밀리 m4에 대한 지원이 제거되었습니다.	2024년 9월 30일
설명서 업데이트	에이전트 요구 사항에서 서브넷과 Amazon EC2 인스턴스를 동일한 가용 영역에 유지하는 방법에 대한 설명이 추가되었습니다.	2024년 7월 18일
설명서 업데이트	AWS Ground Station 에이전트를 자체 사용 설명서로 분할합니다. 이전 변경 사항은 AWS Ground Station 사용 설명서의 문서 기록을 참조하세요 .	2024년 7월 18일

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