Astra Data Store를 모니터링합니다

Astra Data Store

NetApp
June 01, 2022

목차

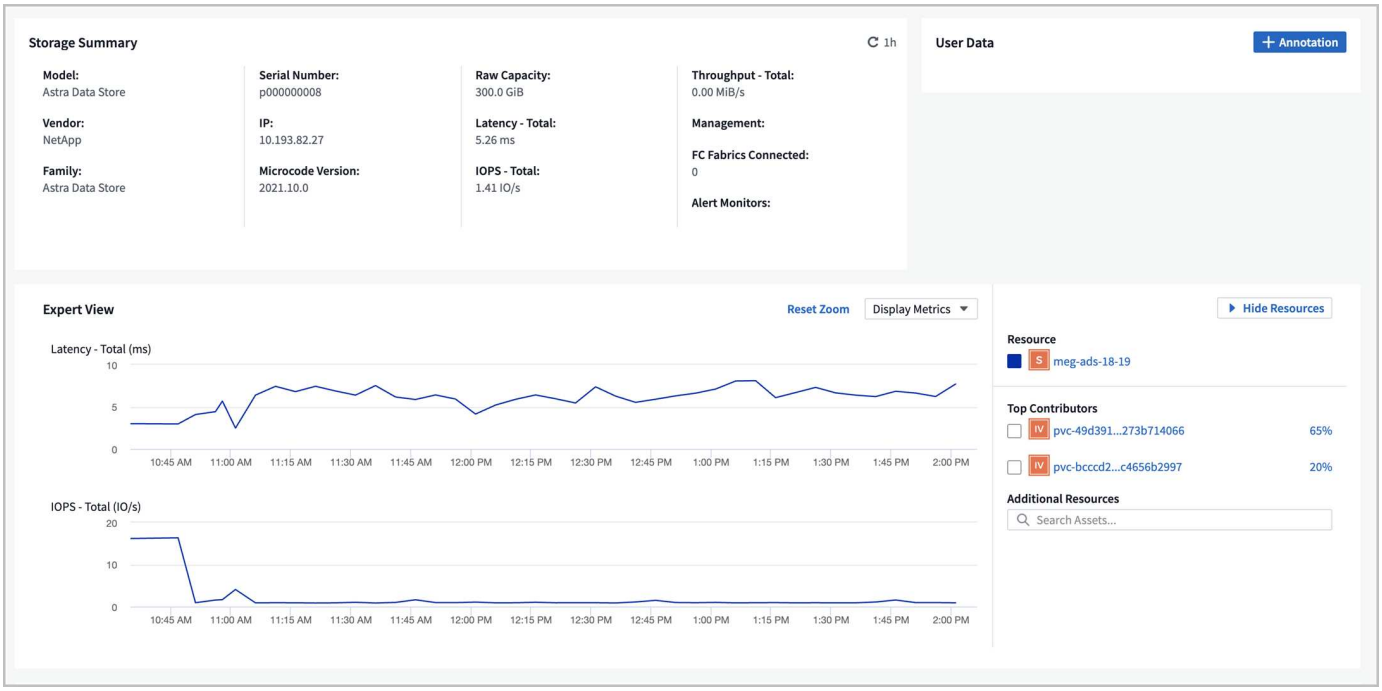
Astra Data Store를 모니터링합니다

Cloud Insights를 통해 메트릭을 모니터링합니다

Cloud Insights를 사용하여 Astra 데이터 저장소 메트릭을 모니터링할 수 있습니다.

다음은 Cloud Insights에 표시되는 몇 가지 Astra 데이터 저장소 메트릭입니다

.



을 사용하여 Astra Data Store에서 생성된 메트릭 목록을 표시할 수도 있습니다 [Open Metrics API help].

다음 작업을 완료할 수 있습니다.

- [Complete Cloud Insights connection prerequisite tasks]
- [Acquisition Unit storage]
- [Download and run the installation script]
- [Edit the Cloud Insights connection]
- [Disconnect from Cloud Insights]

Cloud Insights 연결 필수 작업을 완료합니다

Astra 데이터 저장소를 Cloud Insights와 연결하기 전에 다음 작업을 완료해야 합니다.

- "Astra Data Store Monitoring Operator를 설치합니다" 이것은 Astra Data Store 설치 지침의 일부입니다.
- "kubbeck-astrads 바이너리를 설치합니다" 이것은 Astra Data Store 설치 지침의 일부입니다.
- "Cloud Insights 계정을 만듭니다".
- 'awk, curl, grep', 'jq' 명령을 사용할 수 있는지 확인합니다

다음 정보를 수집합니다.

- 획득 장치, 데이터 수집, 데이터 수집 및 로그 수집 등의 범주에 대한 읽기/쓰기 권한이 있는 * Cloud Insights API 액세스 토큰 *. 이 작업은 읽기/쓰기 작업, 획득 장치 설정 및 데이터 수집 프로세스 설정에 사용됩니다.
- * Kubernetes API 서버 IP 주소 및 포트 *. Astra Data Store 클러스터를 모니터링하는 데 사용됩니다.
- * Kubernetes API 토큰 *. Kubernetes API를 호출하는 데 사용됩니다.
- * 영구 볼륨 구성 *. 영구 볼륨의 프로비저닝 방법에 대한 정보입니다.

획득 장치 저장

획득 장치는 설치 파일, 구성 데이터 및 로그를 저장하기 위해 세 개의 영구 볼륨이 필요합니다. Monitoring Operator는 기본 스토리지 클래스를 사용하여 영구 볼륨 클레임을 생성합니다. 설치 프로그램 스크립트를 실행할 때 '-s' 옵션을 사용하여 다른 스토리지 클래스 이름을 지정할 수 있습니다.

Kubernetes 클러스터에 스토리지 공급자(예: NetApp Trident)가 없으면 설치 관리자 스크립트를 실행할 때 '-r' 옵션을 사용하여 로컬 파일 시스템 경로를 제공할 수 있습니다. '-r' 옵션이 설정되면 설치 프로그램 스크립트는 제공된 디렉토리 내에 세 개의 영구 볼륨을 생성합니다. 이 디렉토리에는 최소 150GB의 여유 공간이 필요합니다.

설치 스크립트를 다운로드하고 실행합니다

Cloud Insights는 모니터링 오퍼레이터를 통해 Astra 데이터 저장소 모니터링을 활성화하는 Bash 스크립트를 제공합니다. 설치 스크립트는 Astra Data Store Collector와 Fluent Bit Agent를 사용하여 획득 장치를 설치합니다.

Cloud Insights 테넌트 도메인 이름과 선택한 Cloud Insights API 액세스 토큰은 다운로드될 때 설치 프로그램 스크립트에 포함됩니다.

그런 다음 다음과 같이 메트릭이 전송됩니다.

- Cloud Insights 획득 장치는 Cloud Insights 데이터 레이크에 메트릭을 전송합니다.
- Fluent bit는 로그 수집 서비스에 로그를 보냅니다.

설치 프로그램 스크립트 도움말을 표시합니다

설치 프로그램 스크립트에 대한 전체 도움말 텍스트가 아래에 나와 있습니다.

설치 프로그램 스크립트 도움말 텍스트 표시:

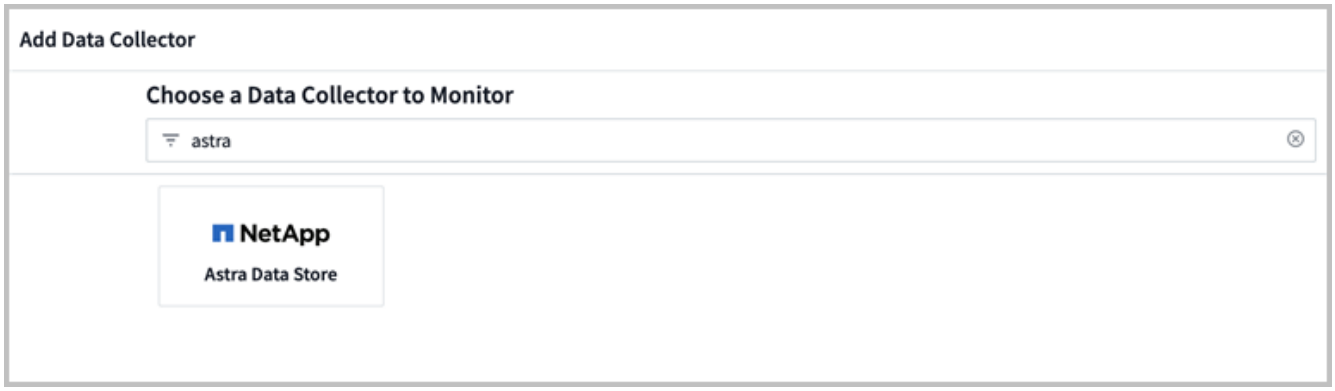
```
./cloudinsights-ads-monitoring.sh -h
```

응답:

```
USAGE: cloudinsights-ads-monitoring.sh [OPTIONS]
Configure monitoring of Astra Data Store by Cloud Insights.
OPTIONS:
  -h                      Display this help message.
  -d ci_domain_name       Cloud Insights tenant domain name.
  -i kubernetes_ip        Kubernetes API server IP address.
  -k ci_api_key           Cloud Insights API Access Token.
  -n namespace            Namespace for monitoring components. (default:
netapp-monitoring)
  -p kubernetes_port      Kubernetes API server port. (default: 6443)
  -r root_pv_dir          Create 3 Persistent Volumes in this directory
for the Acquisition Unit.
                          Only specify this option if there is no Storage
Provisioner installed and the PVs do not already exist.
  -s storage_class        Storage Class name for provisioning Acquisition
Unit PVs. If not specified, the default storage class will be used.
  -t kubernetes_token     Kubernetes API server token.
```

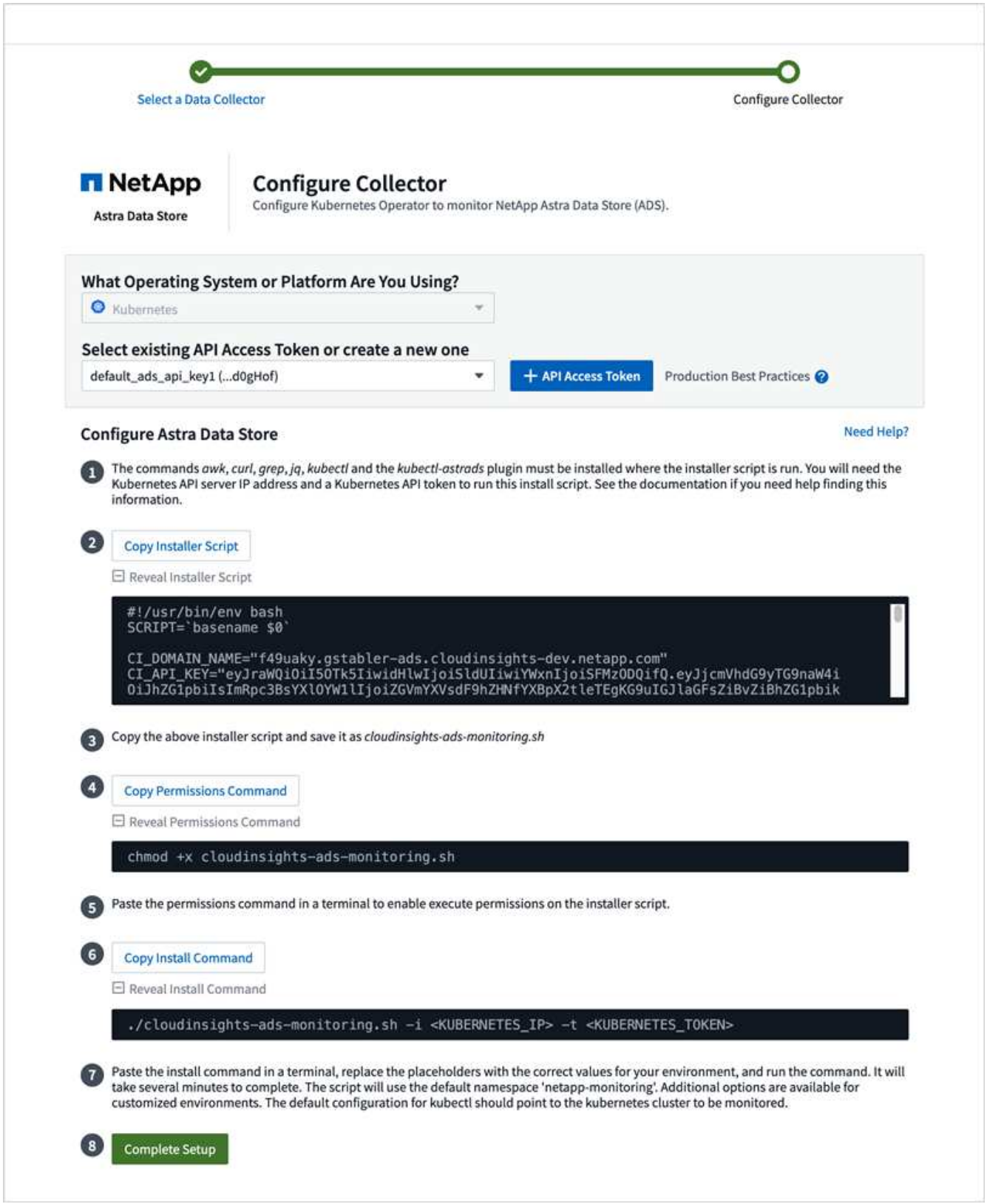
설치 스크립트를 실행합니다

1. Cloud Insights 계정이 없는 경우 계정을 만듭니다.
2. Cloud Insights에 로그인합니다.
3. Cloud Insights 메뉴에서 * Admin * > * Data Collector * 를 클릭합니다.
4. 새 수집기를 추가하려면 * + Data Collector * 를 클릭합니다.



5. Astra Data Store * 타일을 클릭합니다.
6. 올바른 Cloud Insights API 액세스 토큰을 선택하거나 새 토큰을 생성하십시오.
7. 지침에 따라 설치 프로그램 스크립트를 다운로드하고, 권한을 업데이트하고, 스크립트를 실행합니다.

   이 스크립트에는 Cloud Insights 테넌트 URL 및 선택한 Cloud Insights API 액세스 토큰이 포함되어 있습니다.



8. 스크립트가 완료되면 * 설치 완료 * 를 클릭합니다.

   설치 스크립트가 완료되면 데이터 소스 목록에 Astra Data Store Collector가 나타납니다.

오류로 인해 스크립트가 종료되면 나중에 오류가 해결되면 스크립트를 다시 실행할 수 있습니다. 이 스크립트는 환경에서 기본 설정을 사용하지 않는 경우 모니터링 운영자 네임스페이스 및 Kubernetes API 서버 포트와 같은 추가 매개 변수를 지원합니다. 사용법과 도움말 텍스트를 보려면 의 ``h' 옵션을 사용하십시오./cloudinsights-ads-monitoring.sh -h.

설치 스크립트는 구성이 성공할 때 다음과 같은 출력을 생성합니다.

```
Configuring Cloud Insights monitoring for Astra Data Store . . .
Configuring monitoring namespace
...
Configuring output sink and Fluent Bit plugins
Configuring Acquisition Unit
...
Acquisition Unit has been installed successfully.
Configuring Astra Data Store data collector
Astra Data Store collector data '<CLUSTER_NAME>' created
Configuration done!
```

상담원 **CR**의 예

다음은 설치 프로그램 스크립트를 실행한 후 Monitoring-NetApp 에이전트 CR이 어떻게 보일지에 대한 예입니다.

```
 spec:
  au:
    isEnabled: true
    storageClassName: auto-sc
  cluster-name: meg-ads-21-22-29-30
  docker-repo: docker.repo.eng.netapp.com/global/astra
  fluent-bit:
  - name: ads-tail
    outputs:
    - sink: ADS_STDOUT
    substitutions:
    - key: TAG
      value: firetapems
    - key: LOG_FILE
      values:
      - /var/log/firetap/*/ems/ems
      - /var/log/firetap/ems/*/ems/ems
    - key: ADS_CLUSTER_NAME
      value: meg-ads-21-22-28-29-30
  - name: agent
  - name: ads-tail-ci
    outputs:
    - sink: CI
    substitutions:
    - key: TAG
      value: netapp.ads
    - key: LOG_FILE
      values:
      - /var/log/firetap/*/ems/ems
      - /var/log/firetap/ems/*/ems/ems
    - key: ADS_CLUSTER_NAME
      value: meg-ads-21-22-28-29-30
  output-sink:
  - api-key: abcd
    domain-name: bzl9ngz.gst-adsdemo.ci-dev.netapp.com
    name: CI
  serviceAccount: sa-netapp-monitoring
status:
  au-pod-status: UP
  au-uuid: eddeccc6-3aa3-4dd2-a98c-220085fae6a9
```

Cloud Insights 연결을 편집합니다

나중에 Kubernetes API 토큰 또는 Cloud Insights API 액세스 토큰을 편집할 수 있습니다.

- Kubernetes API 토큰을 업데이트하려면 Cloud Insights UI에서 Astra Data Store Collector를 편집해야 합니다.
- 원격 측정 및 로그에 사용되는 Cloud Insights API 액세스 토큰을 업데이트하려면 kubctl 명령을 사용하여 모니터링 오퍼레이터 CR을 편집해야 합니다.

Kubernetes API 토큰을 업데이트합니다

1. Cloud Insights에 로그인합니다.
2. Admin * > * Data Collector * 를 선택하여 Data Collector 페이지에 액세스합니다.
3. Astra Data Store 클러스터의 항목을 찾습니다.
4. 페이지 오른쪽에 있는 메뉴를 클릭하고 * 편집 * 을 선택합니다.
5. Kubernetes API 토큰 필드를 새 값으로 업데이트합니다.
6. Collector 저장 * 을 선택합니다.

Cloud Insights API 액세스 토큰을 업데이트합니다

1. Cloud Insights에 로그인합니다.
2. 관리자 * > * API 액세스 * 를 선택하고 * + API 액세스 토큰 * 을 클릭하여 새 Cloud Insights API 액세스 토큰을 만듭니다.
3. 상담원 CR 편집:

```
kubectl --namespace netapp-monitoring edit agent agent-monitoring-netapp
```

4. 출력 싱크 섹션을 찾아 이름이 'CI'인 항목을 찾습니다.
5. 'api-key'라는 레이블의 경우 현재 값을 새 Cloud Insights API 액세스 토큰으로 바꿉니다.

   섹션은 다음과 같이 표시됩니다.

```
output-sink:
 - api-key: <api key value>
   domain-name: <tenant url>
   name: CI
```

6. 편집기 창을 저장하고 종료합니다.

모니터링 운영자는 새로운 Cloud Insights API 액세스 토큰을 사용하도록 Fluent 비트를 업데이트합니다.

Cloud Insights와의 연결을 해제합니다

Cloud Insights와의 연결을 끊으려면 먼저 Cloud Insights UI에서 Astra 데이터 저장소 수집기를 삭제해야 합니다. 이 작업이 완료되면 모니터링 작동자가 획득 장치, Telegraf(구성된 경우) 및 Fluent 비트 구성을 제거할 수 있습니다.

Astra Data Store Collector를 제거합니다

1. Cloud Insights에 로그인합니다.
2. Admin * > * Data Collector * 를 선택하여 Data Collector 페이지에 액세스합니다.
3. Astra Data Store 클러스터의 항목을 찾습니다.
4. 화면 오른쪽의 메뉴를 선택하고 * Delete * 를 선택합니다.
5. 확인 페이지에서 * 삭제 * 를 클릭합니다.

획득 장치**, Telegraf(**전신**)(**구성된 경우**)** 및 **Fluent bit**를 제거합니다

1. 상담원 CR 편집:

```
kubectl --namespace netapp-monitoring edit agent agent-monitoring-netapp
```

2. au 섹션을 찾아 IsEnabled를 false로 설정합니다
3. '유창한 비트' 섹션을 찾아 ads-tail-CI'라는 플러그인을 제거합니다. 플러그인이 더 이상 없으면 "fluent-bit" 섹션을 제거할 수 있습니다.
4. 텔레그라프가 구성된 경우 텔레그라프 섹션을 찾아 광고 공개 메트릭이라는 플러그인을 제거합니다. 플러그인이 더 이상 없으면 Telegraf 섹션을 제거할 수 있습니다.
5. 출력 싱크 섹션을 찾아 'CI'라는 싱크를 제거합니다.
6. 편집기 창을 저장하고 종료합니다.

   모니터링 오퍼레이터는 Telegraf(Telegraf)(구성된 경우) 및 Fluent 비트 구성을 업데이트하고 획득 장치 포드를 삭제합니다.

7. Storage Provisioner 대신 Acquisition Unit PVS에 로컬 디렉토리를 사용한 경우 PVS를 삭제합니다.

```
kubectl delete pv au-lib au-log au-pv
```

   그런 다음 획득 장치가 실행 중인 노드에서 실제 디렉토리를 삭제합니다.

8. 획득 장치 포드가 삭제된 후 Cloud Insights에서 획득 장치를 삭제할 수 있습니다.
   a. Cloud Insights 메뉴에서 * Admin * > * Data Collector * 를 선택합니다.
   b. Acquisition Units(획득 단위) * 탭을 클릭합니다.
   c. 획득 장치 포드 옆에 있는 메뉴를 클릭합니다.
   d. 삭제 * 를 선택합니다.

모니터링 오퍼레이터는 Telegraf(전신)(구성된 경우) 및 Fluent bit 구성을 업데이트하고 획득 장치를 제거합니다.

메트릭 **API** 도움말을 엽니다

다음은 Astra Data Store에서 메트릭을 수집하는 데 사용할 수 있는 API 목록입니다.

- "도움말" 줄에 메트릭이 설명되어 있습니다.
- "유형" 선은 메트릭이 게이지 또는 카운터인지 여부를 나타냅니다.

```
# HELP astrads_cluster_capacity_logical_percent Percentage cluster logical
capacity that is used (0-100)
# TYPE astrads_cluster_capacity_logical_percent gauge
# HELP astrads_cluster_capacity_max_logical Max Logical capacity of the
cluster in bytes
# TYPE astrads_cluster_capacity_max_logical gauge
# HELP astrads_cluster_capacity_max_physical The sum of the space in the
cluster in bytes for storing data after provisioning efficiencies, data
reduction algorithms and replication schemes are applied
# TYPE astrads_cluster_capacity_max_physical gauge
# HELP astrads_cluster_capacity_ops The IO operations capacity of the
cluster
# TYPE astrads_cluster_capacity_ops gauge
# HELP astrads_cluster_capacity_physical_percent The percentage of cluster
physical capacity that is used (0-100)
# TYPE astrads_cluster_capacity_physical_percent gauge
# HELP astrads_cluster_capacity_used_logical The sum of the bytes of data
in all volumes in the cluster before provisioning efficiencies, data
reduction algorithms and replication schemes are applied
# TYPE astrads_cluster_capacity_used_logical gauge
# HELP astrads_cluster_capacity_used_physical Used Physical capacity of a
cluster in bytes
# TYPE astrads_cluster_capacity_used_physical gauge
# HELP astrads_cluster_other_latency The sum of the accumulated latency in
seconds for other IO operations of all the volumes in a cluster. Divide by
astrads_cluster_other_ops to get the average latency per other operation
# TYPE astrads_cluster_other_latency counter
# HELP astrads_cluster_other_ops The sum of the other IO operations of all
the volumes in a cluster
# TYPE astrads_cluster_other_ops counter
# HELP astrads_cluster_read_latency The sum of the accumulated latency in
seconds of read IO operations of all the volumes in a cluster. Divide by
astrads_cluster_read_ops to get the average latency per read operation
# TYPE astrads_cluster_read_latency counter
# HELP astrads_cluster_read_ops The sum of the read IO operations of all
the volumes in a cluster
# TYPE astrads_cluster_read_ops counter
# HELP astrads_cluster_read_throughput The sum of the read throughput of
all the volumes in a cluster in bytes
# TYPE astrads_cluster_read_throughput counter
# HELP astrads_cluster_storage_efficiency Efficacy of data reduction
technologies. (logical used / physical used)
```

```
# TYPE astrads_cluster_storage_efficiency gauge
# HELP astrads_cluster_total_latency The sum of the accumulated latency in
seconds of all IO operations of all the volumes in a cluster. Divide by
astrads_cluster_total_ops to get average latency per operation
# TYPE astrads_cluster_total_latency counter
# HELP astrads_cluster_total_ops The sum of the IO operations of all the
volumes in a cluster
# TYPE astrads_cluster_total_ops counter
# HELP astrads_cluster_total_throughput The sum of the read and write
throughput of all the volumes in a cluster in bytes
# TYPE astrads_cluster_total_throughput counter
# HELP astrads_cluster_utilization_factor The ratio of the current cluster
IO operations based on recent IO sizes to the cluster iops capacity. (0.0
- 1.0)
# TYPE astrads_cluster_utilization_factor gauge
# HELP astrads_cluster_volume_used The sum of used capacity of all the
volumes in a cluster in bytes
# TYPE astrads_cluster_volume_used gauge
# HELP astrads_cluster_write_latency The sum of the accumulated latency in
seconds of write IO operations of all the volumes in a cluster. Divide by
astrads_cluster_write_ops to get the average latency per write operation
# TYPE astrads_cluster_write_latency counter
# HELP astrads_cluster_write_ops The sum of the write IO operations of all
the volumes in a cluster
# TYPE astrads_cluster_write_ops counter
# HELP astrads_cluster_write_throughput The sum of the write throughput of
all the volumes in a cluster in bytes
# TYPE astrads_cluster_write_throughput counter
# HELP astrads_disk_base_seconds Base for busy, pending and queued.
Seconds since collection began
# TYPE astrads_disk_base_seconds counter
# HELP astrads_disk_busy Seconds the disk was busy. 100 *
(astrads_disk_busy / astrads_disk_base_seconds) = percent busy (0-100)
# TYPE astrads_disk_busy counter
# HELP astrads_disk_capacity Raw Capacity of a disk in bytes
# TYPE astrads_disk_capacity gauge
# HELP astrads_disk_io_pending Summation of the count of pending io
operations for a disk times time. Divide by astrads_disk_base_seconds to
get the average pending operation count
# TYPE astrads_disk_io_pending counter
# HELP astrads_disk_io_queued Summation of the count of queued io
operations for a disk times time. Divide by astrads_disk_base_seconds to
get the average queued operations count
# TYPE astrads_disk_io_queued counter
# HELP astrads_disk_read_latency Total accumulated latency in seconds for
disk reads. Divide by astrads_disk_read_ops to get the average latency per
```

```
read operation
# TYPE astrads_disk_read_latency counter
# HELP astrads_disk_read_ops Total number of read operations for a disk
# TYPE astrads_disk_read_ops counter
# HELP astrads_disk_read_throughput Total bytes read from a disk
# TYPE astrads_disk_read_throughput counter
# HELP astrads_disk_write_latency Total accumulated latency in seconds for
disk writes. Divide by astrads_disk_write_ops to get the average latency
per write operation
# TYPE astrads_disk_write_latency counter
# HELP astrads_disk_write_ops Total number of write operations for a disk
# TYPE astrads_disk_write_ops counter
# HELP astrads_disk_write_throughput Total bytes written to a disk
# TYPE astrads_disk_write_throughput counter
# HELP astrads_value_scrape_duration Duration to scrape values
# TYPE astrads_value_scrape_duration gauge
# HELP astrads_volume_capacity_available The minimum of the available
capacity of a volume and the available capacity of the cluster in bytes
# TYPE astrads_volume_capacity_available gauge
# HELP astrads_volume_capacity_available_logical Logical available
capacity of a volume in bytes
# TYPE astrads_volume_capacity_available_logical gauge
# HELP astrads_volume_capacity_percent Percentage of volume capacity
available (0-100). (capacity available / provisioned) * 100
# TYPE astrads_volume_capacity_percent gauge
# HELP astrads_volume_capacity_provisioned Provisioned capacity of a
volume in bytes after setting aside the snapshot reserve. (size - snapshot
reserve = provisioned)
# TYPE astrads_volume_capacity_provisioned gauge
# HELP astrads_volume_capacity_size Total capacity of a volume in bytes
# TYPE astrads_volume_capacity_size gauge
# HELP astrads_volume_capacity_snapshot_reserve_percent Snapshot reserve
percentage of a volume (0-100)
# TYPE astrads_volume_capacity_snapshot_reserve_percent gauge
# HELP astrads_volume_capacity_snapshot_used The amount of volume snapshot
data that is not in the active file system in bytes
# TYPE astrads_volume_capacity_snapshot_used gauge
# HELP astrads_volume_capacity_used Used capacity of a volume in bytes.
This is bytes in the active filesystem unless snapshots are consuming more
than the snapshot reserve. (bytes in the active file system + MAX(0,
snapshot_used-(snapshot_reserve_percent/100*size))
# TYPE astrads_volume_capacity_used gauge
# HELP astrads_volume_other_latency Total accumulated latency in seconds
for operations on a volume that are neither read or write. Divide by
astrads_volume_other_ops to get the average latency per other operation
# TYPE astrads_volume_other_latency counter
```

```
# HELP astrads_volume_other_ops Total number of operations for a volume
that are neither read or write
# TYPE astrads_volume_other_ops counter
# HELP astrads_volume_read_latency Total accumulated read latency in
seconds for a volume. Divide by astrads_volume_read_ops to get the average
latency per read operation
# TYPE astrads_volume_read_latency counter
# HELP astrads_volume_read_ops Total number of read operations for a
volume
# TYPE astrads_volume_read_ops counter
# HELP astrads_volume_read_throughput Total read throughput for a volume
in bytes
# TYPE astrads_volume_read_throughput counter
# HELP astrads_volume_total_latency Total accumulated latency in seconds
for all operations on a volume. Divide by astrads_volume_total_ops to get
the average latency per operation
# TYPE astrads_volume_total_latency counter
# HELP astrads_volume_total_ops Total number of operations for a volume
# TYPE astrads_volume_total_ops counter
# HELP astrads_volume_total_throughput Total thoughput for a volume in
bytes
# TYPE astrads_volume_total_throughput counter
# HELP astrads_volume_write_latency Total accumulated write latency in
seconds for volume. Divide by astrads_volume_write_ops to get the average
latency per write operation
# TYPE astrads_volume_write_latency counter
# HELP astrads_volume_write_ops Total number of write operations for a
volume
# TYPE astrads_volume_write_ops counter
# HELP astrads_volume_write_throughput Total write thoughput for a volume
in bytes
# TYPE astrads_volume_write_throughput counter
```

Prometheus 및 Grafana를 사용하여 메트릭을 모니터링합니다

Prometheus 및 Grafana를 사용하여 Astra Data Store 메트릭을 모니터링할 수 있습니다. Astra Data Store Kubernetes 클러스터 메트릭 엔드포인트에서 메트릭을 수집하도록 Prometheus를 구성할 수 있으며 Grafana를 사용하여 메트릭 데이터를 시각화할 수 있습니다.

무엇을 '필요로 할거야

- Astra Data Store 클러스터 또는 Astra Data Store 클러스터와 통신할 수 있는 다른 클러스터에 Prometheus 및 Grafana 패키지를 다운로드하여 설치했는지 확인하십시오. 공식 설명서의 지침에 따라 각 도구를 설치합니다.
  - "Prometheus를 설치합니다"
  - "Grafana를 설치합니다"
- Prometheus 및 Grafana는 Astra Data Store Kubernetes 클러스터와 통신할 수 있어야 합니다. Prometheus 및

Grafana가 Astra Data Store 클러스터에 설치되어 있지 않은 경우, Astra Data Store 클러스터에서 실행 중인 메트릭 서비스와 통신할 수 있어야 합니다.

Prometheus를 구성합니다

Astra Data Store는 Kubernetes 클러스터의 TCP 포트 9341에 메트릭 서비스를 제공합니다. 이 서비스에서 메트릭을 수집하려면 Prometheus를 구성해야 합니다.

단계

1. Prometheus 설치를 위해 'Prometheus.yml' 설정 파일을 편집합니다.
2. Astra Data Store 서비스 이름과 해당 포트를 가리키는 서비스 목표를 추가합니다. 예를 들면 다음과 같습니다.

```
scrape_configs:
static_configs:
- targets: ['astrads-metrics-service.astrads-system:9341']
```

3. Prometheus 서비스를 시작합니다.

Grafana 구성

Grafana를 구성하여 Prometheus에서 수집한 메트릭을 표시할 수 있습니다.

단계

1. Grafana 설치를 위해 'datasources.yAML' 구성 파일을 편집합니다.
2. Prometheus를 데이터 소스로 추가합니다. 예를 들면 다음과 같습니다.

```
apiVersion: 1

datasources:
  - name: astradatastore-prometheus
    type: prometheus
    access: proxy
    url: http://localhost:9090
    jsonData:
      manageAlerts: false
```

3. Grafana 서비스를 시작합니다.
4. 에 대한 Grafana 설명서의 지침을 따릅니다 "시작하십시오".

Grafana 대시보드 템플릿을 가져옵니다

Astra Data Store를 설치하기 위해 다운로드한 번들 파일에는 Grafana 내에서 가져올 수 있는 Grafana 대시보드 템플릿 파일이 포함되어 있습니다. 이러한 대시보드 템플릿을 사용하면 Astra Data Store에서 사용 가능한 메트릭의 유형과 이를 보는 방법을 확인할 수 있습니다.

단계

1. Astra Data Store의 .tar.gz bundle을 엽니다.
2. 'manifests' 디렉터리를 엽니다.
3. graana_cluster.json과 graana_volume.json 파일을 추출한다.
4. Grafana 웹 UI 사용, "에서 대시보드 템플릿 파일을 Grafana로 가져옵니다".

이벤트 로그를 구성하고 모니터링합니다

EMS(이벤트 관리 시스템) 로그를 모니터링하려면 다음과 같은 높은 수준의 작업을 수행해야 합니다.

- [Configure monitoring in the Astra Data Store cluster custom resource (CR)]
- [Set up Cloud Insights]
- [Stream event logs to Elastic].

Astra Data Store 클러스터 사용자 지정 리소스(CR)에서 모니터링 구성

Astra Data Store 클러스터 CR에 모니터링 옵션이 구성되어 있지 않은 경우 "Astrads" 확장을 사용하여 설정할 수 있습니다.

입력:

```
kubectl astrads monitoring setup -n <NAMESPACE OF AGENT INSTALLED> -r
<DOCKER REPO TO FIND FLUENT/TELEGRAF ETC IMAGES>
```

여기서,

- 설치된 에이전트의 네임스페이스: 모니터링 에이전트의 기본 이름인 모니터링 에이전트의 네임스페이스를 입력합니다. 모니터링 오퍼레이터의 경우 NetApp CR입니다.
- '-r'은 Fluent 또는 Telegraf 이미지가 있는 Docker 레지스트리를 설정하는 데 선택 사항입니다. 기본적으로 경로는 변경할 수 있는 dddocker.repo.eng.netapp.com/global/astra` 로 설정됩니다.

Cloud Insights를 설정합니다

로그를 보려면 Cloud Insights 설정은 선택 사항이지만 Cloud Insights를 사용하여 데이터를 보는 것이 좋습니다. 을 참조하십시오 "NetApp Cloud Insights 설정 방법" Astra Data Store와 함께 사용

이벤트 로그를 Elastic로 스트리밍합니다

EMS 이벤트 및 기타 POD 로그를 Elastic 등의 타사 엔드포인트로 스트리밍하려면 "Astrads" 확장을 사용합니다.

입력:

```
kubectl astrads monitoring --host <ELASTIC HOST NAME> --port <ELASTIC HOST
PORT> es
```

탄성 호스트 이름은 IP 주소일 수 있습니다.