Configuration and Deployment

Dell EMC ECS: Backing Up Elasticsearch Snapshot Data

Abstract

This document describes how to back up and restore Elasticsearch® data to Dell EMC™ ECS storage.

July 2019

H17847

Revisions

Date	Description
July 2019	Initial release

Acknowledgements

This paper was produced by the Unstructured Technical Marketing Engineering and Solution Architects team.

Author: Rich Paulson

Table of contents

Executive summary

The explosive growth of unstructured data and cloud-native applications has created demand for scalable cloud storage infrastructure in the modern datacenter. Dell EMC™ ECS is the third-generation object store platform from Dell EMC. ECS is designed from the ground up to deliver modern cloud storage API, distributed data protection, and active/active availability spanning multiple data centers.

Elasticsearch® is a distributed, RESTful search and analytics engine capable of storing your data, and includes a smart solution to back up single indices or entire clusters to a remote shared filesystem, S3 or HDFS.

Objectives

This document illustrates how configure Elasticsearch to store data in ECS using the Elasticsearch backup and restore API.

Audience

This document is intended for administrators who manage Elasticsearch deployments. This guide assumes a high level of technical knowledge for the devices and technologies described.

.

1 Solution overview

This section provides an overview of the Dell EMC ECS integration with Elasticsearch including the key technologies used.

1.1 Solution architecture

The below figure illustrates the architectural workflow used in this guide.

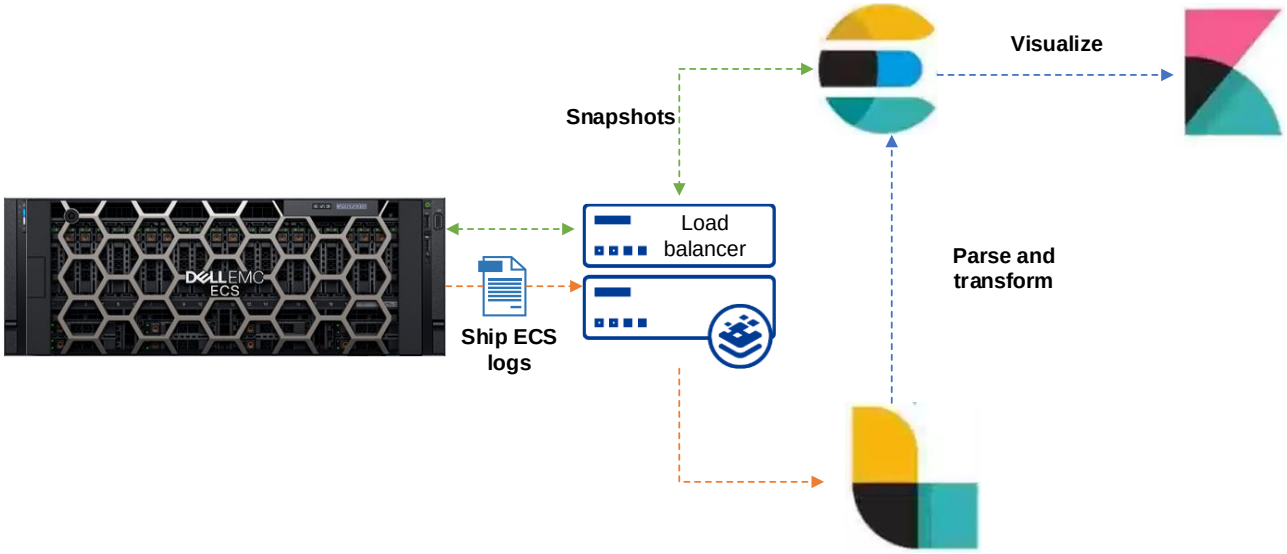

Figure 1 Architecture Workflow

1.2 Key components

The following components and versions were used for the examples in this guide.

Table 1 Dell EMC components

Component	Description
ECS appliance	EX300 (5 Nodes)
ECS version	3.3.0

Table 2 Elasticsearch components

Component	Description
Elasticsearch	7.2.0
Kibana	7.2.0
Logstash	7.2.0

2 Solution implementation

This section describes the high-level steps required to configure Elasticsearch to store snapshot data to ECS. That this guide assumes Elasticsearch and Kibana (optional) have been installed and are functioning.

That the ELK stack was installed and configured to ingest ECS access logs using the document Insights into ECS Data Utilization Using Open Source Tools.

2.1 Implementation workflow

The below figure outlines the steps to implement this solution.

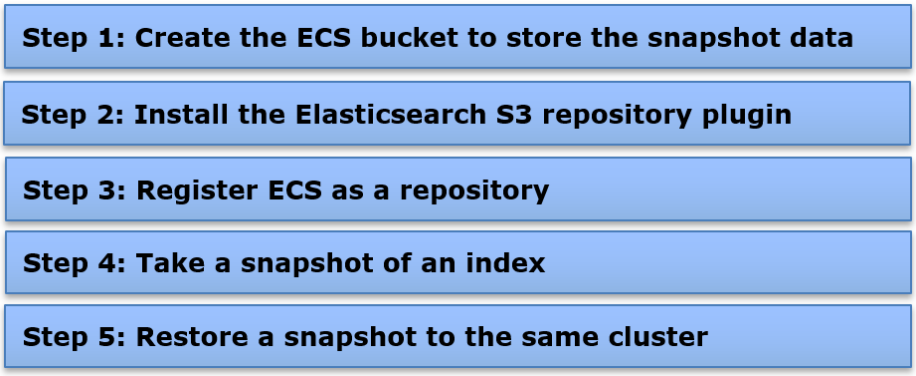

Figure 2 Implementation steps

2.2 Installation and configuration steps

The following steps are meant to be a guide for configuring ECS as an Elasticsearch S3 repository.

2.2.1 Step 1: Create a bucket in ECS to store the snapshot data

The first step is to create a bucket to store the snapshots. This can be done using the ECS Web Portal, ECS API or an S3 client such as S3 Browser. Reference the ECS Administration Guide for detailed information on creating buckets.

2.2.2 Step 2: Install the S3 plugin

The S3 repository plugin adds support to use ECS as the repository for Snapshots and Restores
Reference the Elasticsearch S3 Repository Plugin documentation for instructions on installing the plugin.

To verify that the plugin has been installed invoked the below API call:

```
[root@elk ~]# curl -X GET "localhost:9200/_cat/plugins?v&s=component&h=name,component,version,description"
name   component     version description
node-1 repository-s3 7.2.0   The S3 repository plugin adds S3 repositories
```

Figure 3 Elasticsearch S3 plugin

2.2.3 Step 3: Register ECS as a repository

Elasticsearch provides REST APIs to register an S3 repository. The below example registers ECS as a repository using the _snapshot API.

Note: By default, S3 repositories use a client named default, but this can be modified using the repository setting client. For example: client": "my_ecs_client".

Create the S3 repository

```
curl -X PUT "localhost:9200/_snapshot/ecs" -H 'Content-Type: application/json' -
d'
{
  "type": "s3",
  "settings": {
    "bucket": "<ECS Bucket>",
    "base_path": "snapshots",
    "endpoint": "<ECS Endpoint>",
    "access_key": "<ECS Object User>",
    "secret_key": "<ECS S3 Secret Key>",
    "protocol": "http"
  }
}
'
```

The following examines above settings in more detail:

- **bucket**: This the name of the bucket that was created in Step 1 to store snapshot data.
- **base_path**: [optional] This specifies the path within the bucket to store data.
- **endpoint**: A load balancer is required with ECS. This specifies the endpoint of the load balancer in front of the ECS nodes.
- **access_key**: Specifies the ECS object user to use for authentication.
- **secret_key**: The secret s3 key of the ECS object user.
- **protocol**: Specifies whether HTTP or HTTPS is used.

Note: Large files can be broken down into chunks if needed. The default setting is 1gb which should not need to be adjusted for ECS.

Additional settings can be specified for the repository and are specified in the Elasticsearch Repository settings documentation.

We can verify that our repository was successfully created by using the 'POST /_snapshot/ecs/_verify' call.

```
[root@elk ~]# curl -XPOST 'http://localhost:9200/_snapshot/ecs/_verify?format=json&pretty'
{
  "nodes" : {
    "y8OXw2nzTCum4_cVREGoQA" : {
      "name" : "node-1"
    }
  }
}
[root@elk ~]#
```

Figure 4 S3 Repository verification

2.2.4 Step 4: Take a snapshot (backup) of an Index

The next step is to take a snapshot of the logstash index which contains the ECS data access log events. Our index is named logstash-2019.07.01-000001.

Index snapshot

```
curl -X PUT "localhost:9200/_snapshot/ecs/snapshot_1" -H 'Content-Type:
application/json' -d'
{
  "indices": "logstash-2019.07.01-000001",
  "ignore_unavailable": true,
  "include_global_state": false
}
'
```

The status of the snapshot can be verified using the below API.

```
[root@elk bin]# curl http://localhost:9200/_cat/snapshots/ecs?v
id         status start_epoch start_time end_epoch  end_time duration indices successful_shards failed_shards total_shards
snapshot_1 SUCCESS 1562028385  00:46:25   1562028396 00:46:36    10.9s       1                 1             0            1
```

We can verify that snapshot data exists in the ECS bucket using an S3 client such as s3cmd, s3curl or S3 Browser. The below show the properties of our bucket storing the snapshot data.

URL: http://elasticsearch.s3.richp.local/snapshots/

Property	Value
Folder name	snapshots/
Total objects	54
Total files	54
Total folders	0
Total size	2.35 MB (2462469 bytes)

Figure 5 Snapshot data

2.2.5 Step 5: Restore a snapshot to the same cluster

A snapshot can be restored using the following command:

```
POST /_snapshot/ecs/snapshot_x/_restore
```

This example restores one of our snapshots to the same cluster. Since an existing index can be only restored if it is closed and has the same number of shards as the index in the snapshot we'll restore our snapshot to a new index.

Note: Reference the Elasticsearch Snapshot and Restore documentation for detailed information on restoring indices.

List the snapshots in the ECS repository using the below API.

```
[root@elk ~]# curl -X GET "localhost:9200/_cat/snapshots/ecs?v&s=id"
id         status start_epoch start_time end_epoch  end_time duration indices successful_shards failed_shards total_shards
snapshot_1 SUCCESS 1562028385  00:46:25   1562028396 00:46:36    10.9s       1                 1             0            1
snapshot_2 SUCCESS 1562087932  17:18:52   1562087936 17:18:56     4.1s       1                 1             0            1
snapshot_3 SUCCESS 1562088142  17:22:22   1562088144 17:22:24     1.9s       1                 1             0            1
[root@elk ~]#
```

The API call used to restore snapshot_1 to a new index is as follows:

```
curl -X POST "localhost:9200/_snapshot/ecs/snapshot_1/_restore" -H 'Content-
Type: application/json' -d'{
  "indices": "logstash-2019.07.01-000001",
  "ignore_unavailable": true,
  "include_global_state": true,
  "include_aliases": false,
  "rename_pattern": "logstash(.+)",
  "rename_replacement": "restored_index_$1"
}
```

Verify that the snapshot was restored to a new index:

```
[root@elk ~]# curl -X GET "localhost:9200/_cat/indices?v"
health status index                           uuid                   pri rep docs.count docs.deleted store.size pri.store.size
yellow open   restored_index_-2019.07.01-000001 _t40gKNkRViU9sJ7RdRkSA   1   1      13999            0      2.3mb          2.3mb
green  open   .monitoring-logstash-7-2019.07.02 p3VoCMy6Qxu_dpr7Fj-P4Q   1   0      93420            0      5.4mb          5.4mb
green  open   .monitoring-logstash-7-2019.07.01 RjFvyahITuGGmj4IwOTtgA   1   0       5541            0    687.7kb        687.7kb
green  open   .monitoring-kibana-7-2019.07.02   qO39bI4mSaqUmxvFP2-h4A   1   0       6227            0      1.3mb          1.3mb
green  open   .monitoring-es-7-2019.07.02       hXG49G0pTf-CzpaEwe9zvg   1   0      80998        62240     36.9mb         36.9mb
green  open   .kibana_task_manager              ZK8_Ac3STgCtK-3sBaIuyg   1   0          2            0     29.6kb         29.6kb
yellow open   logstash-2019.07.01-000001        cP57O-giQOamYi0_9bRVHg   1   1     176514            0     31.6mb         31.6mb
green  open   .kibana_1                         RwHCm6HUTmOqMvUwJj1CKA   1   0          5            1     68.3kb         68.3kb
green  open   .monitoring-es-7-2019.07.01       dCOvW8Y0Q1WS47GItIIPOQ   1   0       4769          735      2.8mb          2.8mb
green  open   .monitoring-kibana-7-2019.07.01   YpegrV-NQiuhZsaNDS0ZDw   1   0        475            0      299kb          299kb
[root@elk ~]#
```

Note: Our index health is yellow because the current cluster only consists of a single node, so the replicas remain unassigned simply because no other node is available to contain them.

3 Configuration and tuning

The following configuration and tuning changes are recommended.

Table 3 Configuration and tuning recommendations

Description	Detail
S3 repository chunk size	Big files can be broken down into chunks during snapshotting if needed. The chunk size can be specified in bytes or by using size value notation, i.e. 1gb, 10mb, 5kb. Defaults to 1gb. Dell EMC recommends using the default setting of 1gb.
S3 repository buffer size	The Minimum threshold below which the chunk is uploaded using a single request. Beyond this threshold, the S3 repository will use Multipart upload to split the chunk into several parts, each of buffer_size length, and to upload each part in its own request. Dell EMC recommends using the minimum default setting between 100mb and 5% of the heap size.

A Technical support and resources

Dell.com/support is focused on meeting customer needs with proven services and support.

Storage technical documents and videos provide expertise that helps to ensure customer success on Dell EMC storage platforms.

A.1 Related resources

ECS product documentation

- Dell EMC ECS product documentation
  - https://community.emc.com/docs/DOC-73931
- Dell EMC ECS Architecture and Overview
  - http://www.emc.com/collateral/white-papers/h14071-ecs-architectural-guide-wp.pdf
- Dell EMC ECS Networking and Best Practices
  - http://www.emc.com/collateral/white-paper/h15718-ecs-networking-bp-wp.pdf
- Dell EMC ECS Best Practices
  - https://www.emc.com/collateral/white-papers/h16016-ecs-best-practices-guide-wp.pdf
- Insights into ECS Data Utilization using Open Source Tools
  - https://www.emc.com/collateral/white-papers/h1596-insights-into-ecs-data-utilization-use-ost.pdf

Elasticsearch product documentation

- Introduction to Elasticsearch
  - https://www.elastic.co/guide/en/elasticsearch/reference/current/elasticsearch-intro.html
- Elasticsearch S3 plugin
  - https://www.elastic.co/guide/en/elasticsearch/plugins/current/repository-s3.html
- Elasticsearch S3 Repository Settings
  - https://www.elastic.co/guide/en/elasticsearch/plugins/current/repository-s3-repository.html