

HISTORICAL DATA ARCHIVE

PROJECT REPORT

submitted
by

NISHANT SINGH (M190397CA)

In partial fulfillment for the award of the Degree of

MASTER OF COMPUTER APPLICATION

**Under the guidance of
INDER GOPAL**

(CEO, IUDX UNIT, IISC Bengaluru)

SHREE. SREENU NAIK BHUKYA

(Assistant Professor, Dept. of Computer Science and Engineering, NIT Calicut)



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

NATIONAL INSTITUTE OF TECHNOLOGY CALICUT

NIT CAMPUS PO, CALICUT

KERALA, INDIA 673601

MAY 2022

ACKNOWLEDGMENT

I take this opportunity to express my sincere gratitude to all individuals, directly or indirectly, who have contributed towards the completion of this Project report.

DECLARATION

I, hereby declare that this submission is my own work and that, to the best of my knowledge and belief, it contains no material previously published or written by another person nor material which has been accepted for the award of any other degree or diploma of the university or other institute of higher learning, except where due acknowledgment has been made in the text.

Place: NIT Calicut

Date: 09/05/2022

Name: Nishant Singh

Reg. No.: M190397CA

Signature:

CERTIFICATE

This is to certify that the project report entitled **HISTORICAL DATA ARCHIVE** submitted by **Mr. NISHANT SINGH (M190397CA)** to the National Institute of Technology Calicut towards partial fulfillment of the requirements for the award of the Degree of **MASTER OF COMPUTER APPLICATIONS** is a bona fide record of the work carried out by him during his Internship at **INDIAN URBAN DATA EXCHANGE PROGRAM UNIT** during **JAN-2022 to MAY-2022** under my supervision and guidance.

Signed by External Project Guide with Name and Date.



Inder Gopal
Research Professor, IISc, CEO
IUDX Programme Unit
Place: Bengaluru
Date: 6 May 2022

Signature of Project guide with name and date

Signature of Head of Department

Nishant Singh

ABSTRACT

Historical Data Archive Project is All about Managing the Old Data, here we will store old data in the form of snapshots such that it will consume less Storage. If we see the relevance of this Problem, old data management will be in a suitable manner, and in future we can retrieve old data as per requirement. Here if we talk about approach then we used Index life cycle management for solving this problem, this data will be divided into different phases and data retrieval will be easy. To solve this problem first we used the concept of snapshots on a fixed interval for taking a snapshot of data then used Index lifecycle policy for transfer the data into different phases, finally deleted the data and restored the data. If we talk about the outcome then successfully got the snapshots of data, deleted the data index and restored the data with help of snapshots. if we talk about benefits then first benefits is if we lose data from server then we can simply use the snapshots and can recover the data easily. Second benefit is we are deleting the old data from the server so it will save the cost.

Contents

	Page Number
1. Introduction	
1.1 Problem Definition	6
1.2 Background	7
1.3 Current Status of Problem	8
1.4 Motivation	9
2. Literature Survey	
2.1 Literature Survey	10
3. Implementation	
3.1 Step 1	11
3.2 Step 2	12
3.3 Step 3	12 - 13
3.4 Step 4	13 - 15
3.5 Step 5	15 - 17
3.6 Step 6	18 - 20
3.7 Step 7	21 - 25
4. Code Approach	26 - 29
5. Results	30
6 Conclusion	31
References	32

Chapter 1

Introduction

1.1 Problem Definition

This Historical data archive project is all about managing Old data in Proper way when data becomes old after a certain period of time . There are lots of servers in the real world which generate millions of data per day. If we store this data in one place then it can generate lots of problems after some day for upcoming actions , So i want to manage this data in a proper way such that we will not face any problem for upcoming actions. Upcoming actions Aspect can be following

Aspect 1 : if the Storage drive crashes we will lose all the important data so it's important to manage this data.

Aspect 2 : Even if you lose your data there should be some certain approach so that you can restore it properly.

Sub goals : Right now I am managing data locally, in the future we can manage it on Cloud, in today's cloud is becoming idle for data management.

1.2 Background

This project is all about handling the data, which becomes old. I want a way such that

There should be 3 phases for data handling

1: if data is stored recently it should be available for the first 10 days because data is new and it should be available for everyone, if anybody wants to perform some operation on new data they should be able to do it.

2: when data becomes old it should be sent to another phase. Like there should be different phases for managing the data. For Example if data is 10 days old then i will put in phase 1, after 10 days i will transfer the data into second phase and now phase 1 will have free space for new data

3: this third stage is important because when data become 20 days i will delete that data so that storage can be free for new data in phase 2, but here we will do one operation before deleting the data such that we will take a snapshot of data so that in future we can retrieve it for further process.

4: give a demo that how you are restoring deleted data by 2 ways:

(i) : by graphical user interface

(ii) : by code also

So this is all about my project introduction and problem statement.

If we talk about the importance of this Project, this Project can save lot's of Cost in Data Management, also Multiple Process on Data will be So fast.

1.3 Current Status of Problem

Up to now for solving this problem i went through the 4 technologies :

1.3.1 logstash : it is helpful in collecting the data from the server and ingesting the collected data for the next process. (there are some sort of plugins which are available and provide the collection of data and data ingestion feature, i am using Elastic-search head plugin) .

1.3.2 Elastic-search : the data which i collected through Elastic-search head plugin, we need to store that data somewhere so for storing that data i used Elastic-search. For handling data into multiple phases there is a concept in Index lifecycle management (ILM) In Elastic-search, I went through the ILM Concepts. It provides a feature for transferring the data into different phases.

1.3.3 Kibana: this is the tool having ability to perform certain operations on data using Graphical User interface. Right now I learned how to transfer data into different phases and now explored the snapshot and deletion concept, also how to store this snapshot so that we can retrieve the data in future from this snapshot.

1.3.4 Docker : It is containerisation technology for launching the containers, with help of this i will bring elastic-search and kibana containers up. Generally docker provides an Operating system within one second and the best part is, it provides isolation of environments. If one OS is corrupted it will not affect the others.

With the Help of this technology and By Using concepts of these technologies finally I solved the Historical Data Archive Problem.

1.4 Motivation

- * This problem needs to be done because storage is a big issue in the current industry. No company is making a very large storage system because of a failure issue. If it is crashed then the whole cost becomes waste.
- * Most Important benefit is the proper way of data management . If it's a matter of social relevance then definitely it will provide ease of good.
- * In the future it is going to reduce our cost for storage because we are deleting the old data and storing the compressed version of data i.e. snapshot, as and when requirements come i will restore the data from latest snapshot which i get from the latest data.

Chapter 2

Literature Survey

2.1 Literature Survey

If we talk about the Scope of the topic then we can say that any company which is handling a big and live database can use the concept of this Project, that company can transfer data in different phases according to requirement and also if no use of old data just delete the old data after taking snapshots.

If we talk about the points, which can be discussed on the basis of this literature that can be data collection source such that what type of data you want to collect and what is the frequency of the data collection, there can be multiple use case like, we are collecting ambulance sensor system data just for clearing the traffic in emergency case. Also we can discuss the filter process on data.

If i talk about assumption in this project generally we used one Elastic search Head plugin for inserting the data such that we can test the other process if i run the all the three container Elastic search, logstash and kibana then local system can be stuck because these tools made in java which consume heavy amount of RAM and CPU.

If we talk about the summary of this Project then we can say, successfully collected all the data pushed into Elastic search and with the help of kibana we did multiple operations. And if i talk further about the area for review then this can be such that where we are storing the data right now i am storing it locally but in future we can store it on cloud.

Chapter 3

Implementation

In Implementation i will discuss my work in this Project :

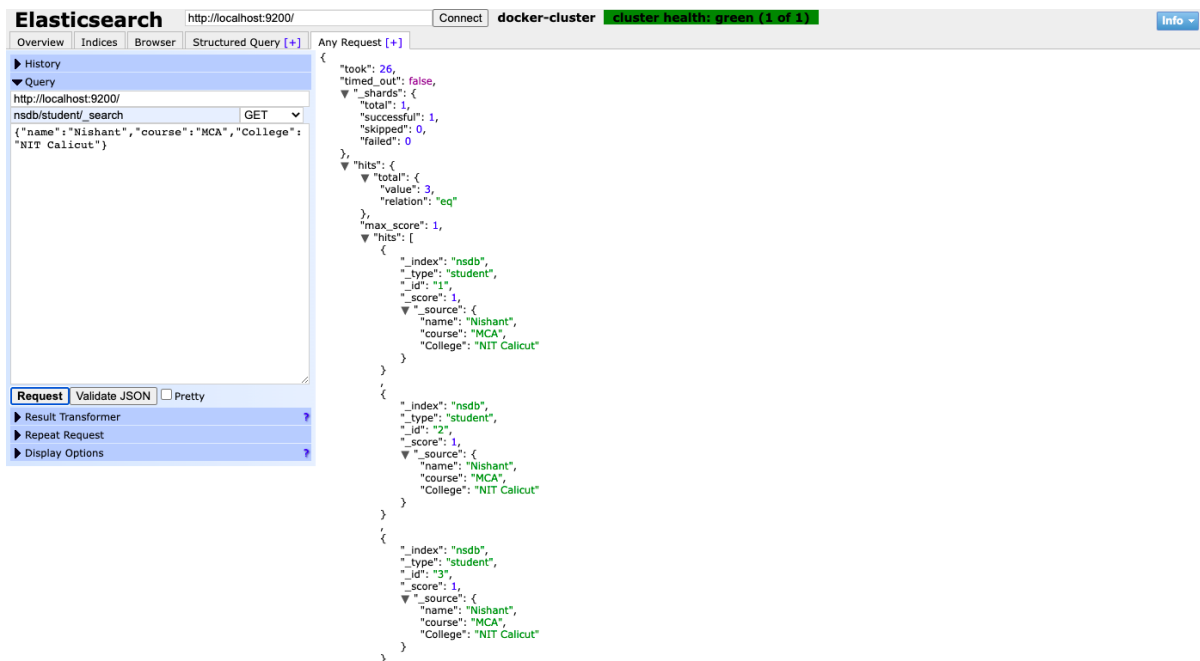
3.1 Step 1:

First I will collect data for further processing.

Launched elastic search and entered some data for testing my approach.

```
docker run -dit --name elasticsearch --net elk -p 9200:9200 -e "discovery.type=single-node"
elasticsearch:7.17.0
```

Used 1 plugin **Elastic search head** for entering data manually and checked that data is inserted correctly or not, data format should be in JSON.



3.2 Step 2:

Launched Kibana for further processing on this data

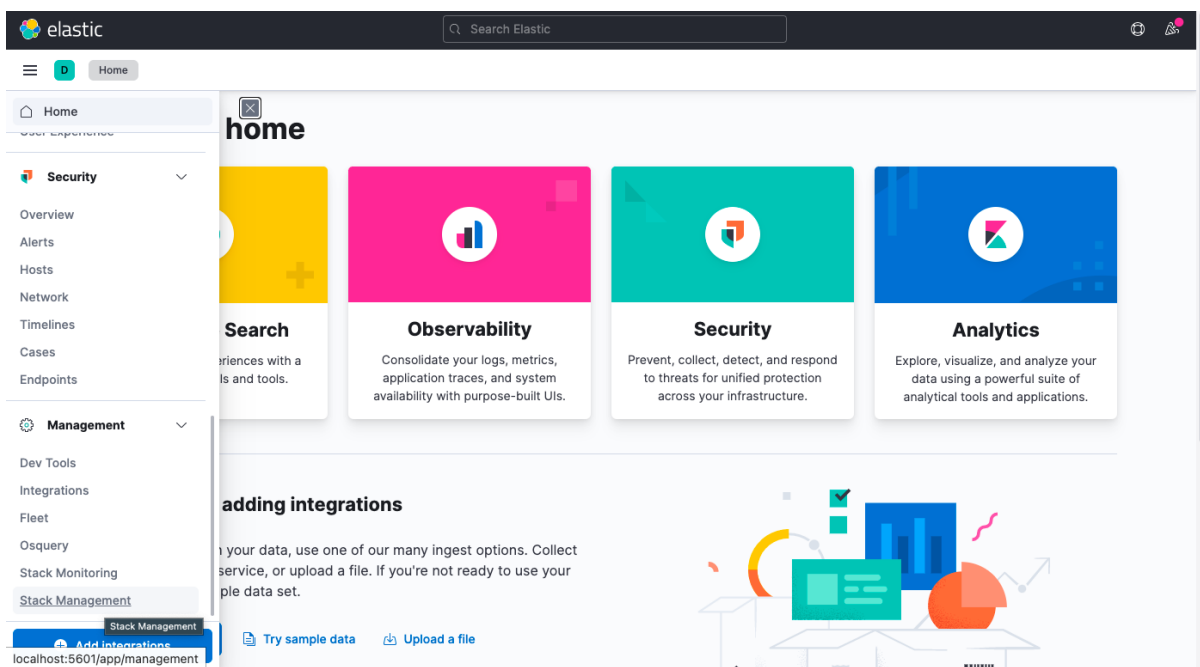
```
docker run -dit --name kibana --net elk -p 5601:5601 kibana:7.17.0
```

Opened the kibana on browser

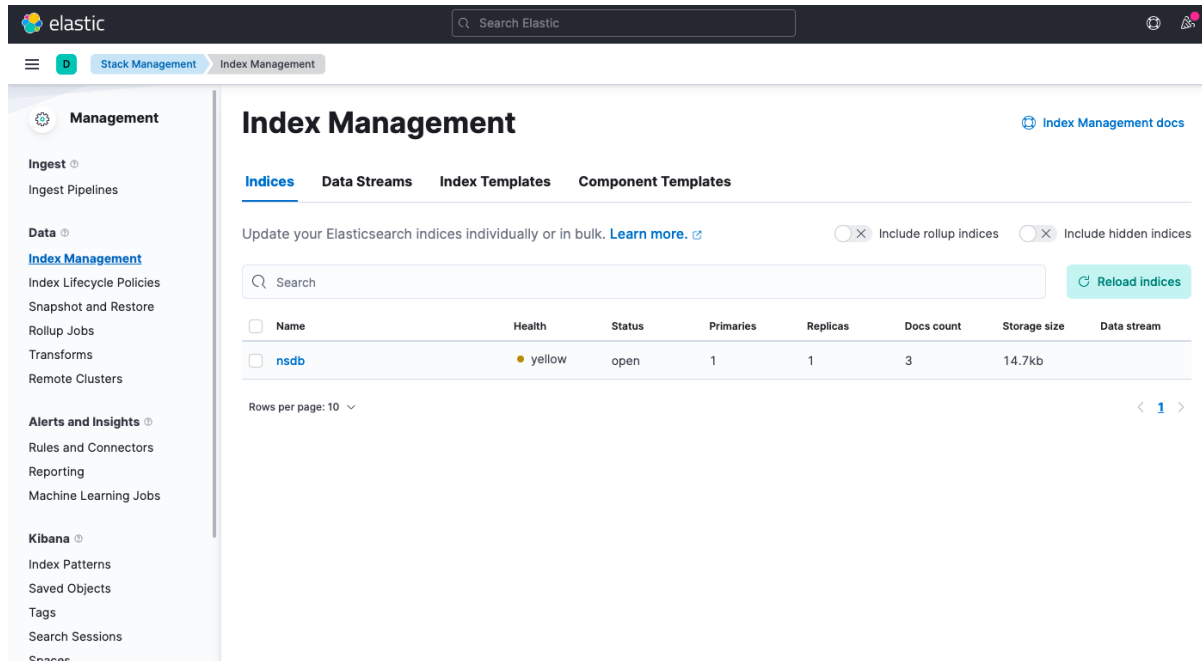
<http://localhost:5601/app/kibana>

3.3 Step 3:

Opened the stack management section



Click on index management and you will see your index come up



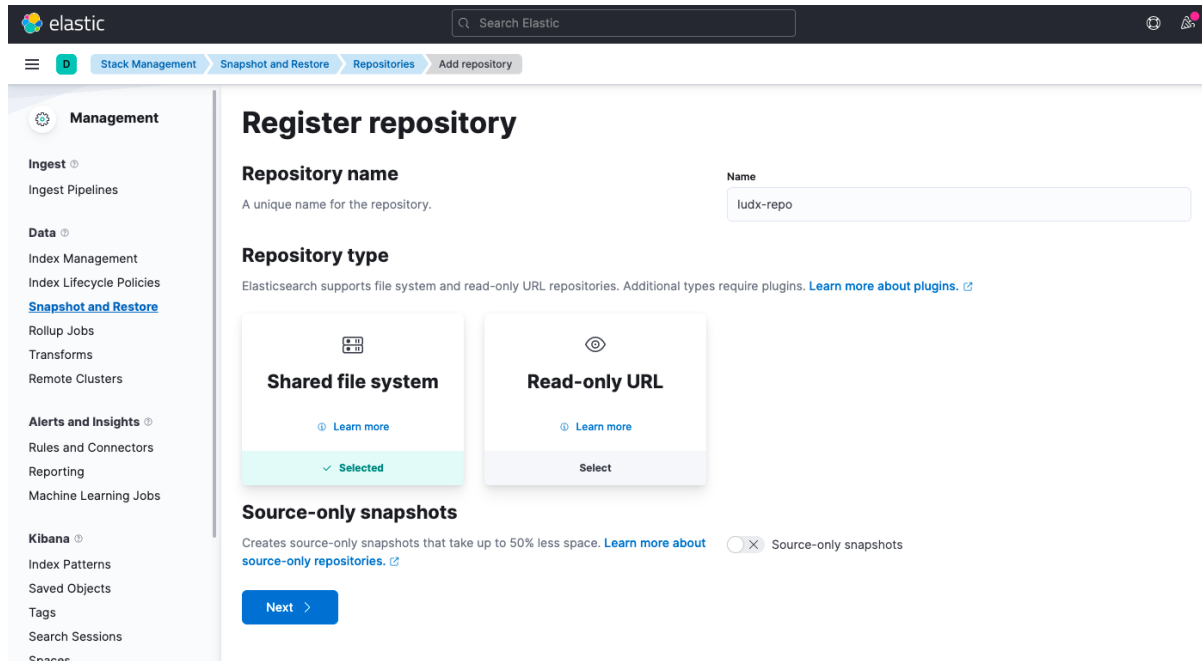
3.4 Step 4:

Now we will do process for taking snapshot, so click on snapshot and restore and register repository for taking snapshot

For registering repository there can be many options like

Shared file (local storage) OR cloud (AWS, AZURE ,GCP)

Install the required plugin according to your use case, i am going to use **Shared File System**



After this, create one directory in elastic search and give permissions for writing in this directory.

```
docker exec -it elasticsearch bash
mkdir backup_repo
chmod a+rwX backup_repo/
```

Update the elastic search configuration file and add the following line

```
Vim config/elasticsearch.yml
path.repo: /usr/share/elasticsearch/backup_repo
```

Restart elastic search so that changes can take place in the configuration file.

After this register repository with this setting

The screenshot shows the 'Register repository' settings for 'iudx-repo' in the Elasticsearch UI. The left sidebar contains navigation links under 'Management', 'Data', 'Alerts and Insights', and 'Kibana'. The main content area is titled 'Register repository' and 'iudx-repo' settings. It includes sections for 'File system location', 'Snapshot compression', 'Chunk size', 'Max snapshot bytes per second', and 'Max restore bytes per second'. The 'File system location' is set to '/usr/share/elasticsearch/backup_repo'. The 'Snapshot compression' toggle is turned off. The 'Chunk size' is set to '1g'. The 'Max snapshot bytes per second' is set to '40mb'. The 'Max restore bytes per second' is set to '10mb'.

Management

- Ingest
 - Ingest Pipelines
- Data
 - Index Management
 - Index Lifecycle Policies
 - Snapshot and Restore**
 - Rollup Jobs
 - Transforms
 - Remote Clusters
- Alerts and Insights
 - Rules and Connectors
 - Reporting
 - Machine Learning Jobs
- Kibana
 - Index Patterns
 - Saved Objects
 - Tags
 - Search Sessions
 - Spaces

Register repository

'iudx-repo' settings

[Shared file system repository docs](#)

File system location

The location must be registered in the `path.repo` setting on all master and data nodes.

Location (required)

`/usr/share/elasticsearch/backup_repo`

Snapshot compression

Compresses the index mapping and setting files for snapshots. Data files are not compressed.

☐ Compress snapshots

Chunk size

Breaks files into smaller units when taking snapshots.

Chunk size

Accepts byte size units, such as `1g`, `10mb`, `5k`, or `1024b`. Defaults to unlimited.

`1g`

Max snapshot bytes per second

Maximum rate for creating snapshots for each node.

Max snapshot bytes per second

Accepts byte size units, such as `1g`, `10mb`, `5k`, or `1024b`. Defaults to `40mb` per second.

`40mb`

Max restore bytes per second

Max restore bytes per second

`10mb`

Check connectivity

The screenshot shows the 'Snapshot and Restore' page in the Elasticsearch UI. The left sidebar contains navigation links under 'Management', 'Data', 'Alerts and Insights', and 'Kibana'. The main content area is titled 'Snapshot and Restore' and shows a table of repositories. The 'iudx-repo' repository is listed with a 'Shared' type. A modal window is open showing the details for 'iudx-repo', including its 'Repository type' (Shared file system), 'Snapshots' (Repository has no snapshots), 'Settings' (Location: /usr/share/elasticsearch/backup_repo), and 'Verification status' (Connected). The modal also shows a JSON snippet of the repository configuration and buttons for 'Close', 'Remove', and 'Edit'.

Management

- Ingest
 - Ingest Pipelines
- Data
 - Index Management
 - Index Lifecycle Policies
 - Snapshot and Restore**
 - Rollup Jobs
 - Transforms
 - Remote Clusters
- Alerts and Insights
 - Rules and Connectors
 - Reporting
 - Machine Learning Jobs
- Kibana
 - Index Patterns
 - Saved Objects
 - Tags
 - Search Sessions
 - Spaces

Snapshot and Restore

Use repositories to store and recover backups of your Elasticsearch indices and clusters.

Snapshots **Repositories** **Policies** **Restore Status**

Search...

Name	Type
iudx-repo	Shared

Rows per page: 20

iudx-repo

Repository type

Shared file system

Snapshots

Repository has no snapshots

Settings

Location

`/usr/share/elasticsearch/backup_repo`

Verification status

Connected

Details

```
{
  "nodes": {
    "2VopsIXxTreBt3dy9ToJNw": {
      "name": "986639d9fae18"
    }
  }
}
```

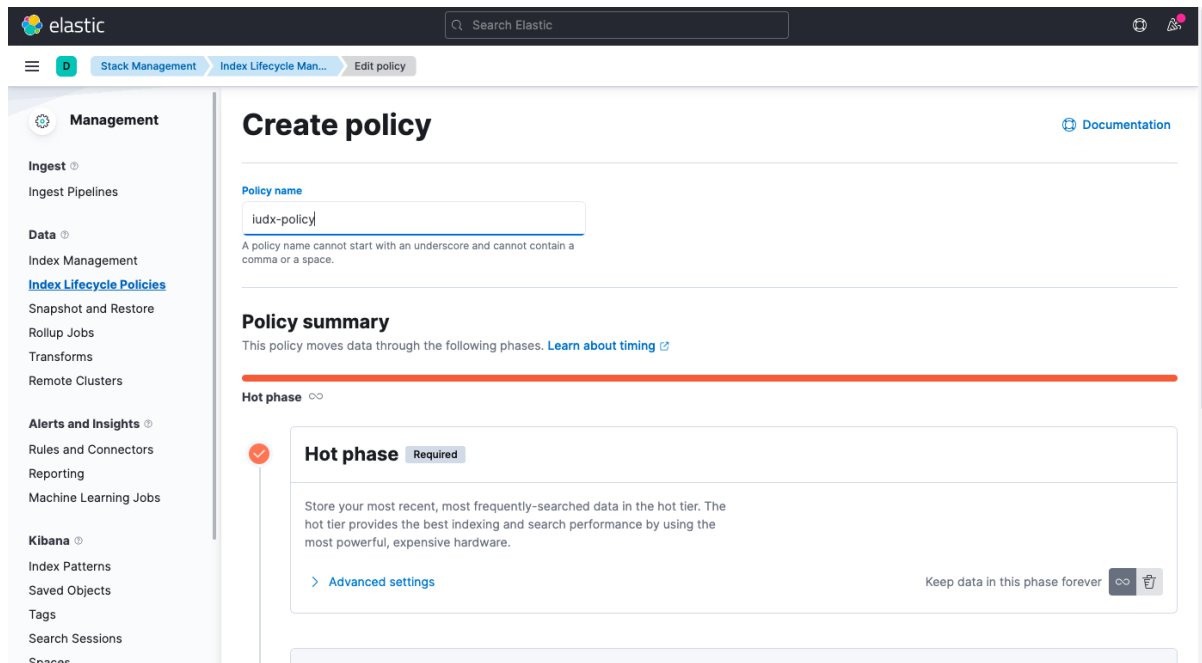
[Close](#) [Remove](#) [Edit](#)

3.5 Step 5:

Now I will create index life cycle management for data management so that searching will be faster and old data is deleted and stored somewhere in the form of a snapshot.

Here I will store it in a local backup folder, that's why I created one directory and connected it for storing the data.

Let's create ILP



elastic Search Elastic

Stack Management Index Lifecycle Man... Edit policy

Management

- Ingest
 - Ingest Pipelines
- Data
 - Index Management
 - Index Lifecycle Policies**
 - Snapshot and Restore
 - Rollup Jobs
 - Transforms
 - Remote Clusters
- Alerts and Insights
 - Rules and Connectors
 - Reporting
 - Machine Learning Jobs
- Kibana
 - Index Patterns
 - Saved Objects
 - Tags
 - Search Sessions
 - Spaces

Rollover

Start writing to a new index when the current index reaches a certain size, document count, or age. Enables you to optimize performance and manage resource usage when working with time series data.

Note: How long it takes to reach the rollover criteria in the hot phase can vary. [Learn more](#)

☐ Use recommended defaults

Roll over when an index is 30 days old or any primary shard reaches 50 gigabytes.

Enable rollover ☒

Maximum primary shard size 30 kilobytes

Maximum age 10 days

Maximum documents

Maximum index size gigabytes

Force merge

Reduce the number of segments in each index shard and clean up deleted documents. [Learn more](#)

☐ Force merge data

Shrink

Shrink the index to a new index with fewer primary shards. [Learn more](#)

☐ Shrink index

elastic Search Elastic

Stack Management Index Lifecycle Man... Edit policy

Management

- Ingest
 - Ingest Pipelines
- Data
 - Index Management
 - Index Lifecycle Policies**
 - Snapshot and Restore
 - Rollup Jobs
 - Transforms
 - Remote Clusters
- Alerts and Insights
 - Rules and Connectors
 - Reporting
 - Machine Learning Jobs
- Kibana
 - Index Patterns
 - Saved Objects
 - Tags
 - Search Sessions
 - Spaces

Warm phase ☒ Move data into phase when: 15 days old

Move data to the warm tier when you are still likely to search it, but infrequently need to update it. The warm tier is optimized for search performance over indexing performance.

[Advanced settings](#) Delete data after this phase

Cold phase ☐

Move data to the cold tier when you are searching it less often and don't need to update it. The cold tier is optimized for cost savings over search performance.

Delete phase ☒ Remove Move data into phase when: 20 days old

Delete data you no longer need.

Wait for snapshot policy

Specify a snapshot policy to be executed before the deletion of the index. This ensures that a snapshot of the deleted index is available.

Policy name (optional) iudx-snapshot

Policy created

For policy creation we need to create a snapshot such that before deleting the data we can store it. In the future from this snapshot we can restore the data by curator or Kibana GUI.

3.6 Step 6:

Now create snapshot policy such that on which interval it will take snapshot

elastic Search Elastic

Stack Management Snapshot and Restore Policies Add policy

Management

- Ingest
 - Ingest Pipelines
- Data
 - Index Management
 - Index Lifecycle Policies
 - Snapshot and Restore**
 - Rollup Jobs
 - Transforms
 - Remote Clusters
- Alerts and Insights
 - Rules and Connectors
 - Reporting
 - Machine Learning Jobs
- Kibana
 - Index Patterns
 - Saved Objects
 - Tags
 - Search Sessions
 - Snaps

Logistics [Logistics docs](#)

Policy name
A unique identifier for this policy.

Name: ludx-snapshot

Snapshot name
The name for the snapshots. A unique identifier is automatically added to each name.

Snapshot name: <nsdb-snap-{now/d}>
Supports date math expressions. [Learn more.](#)

Repository
The repository where you want to store the snapshots.

Repository: ludx-repo

Schedule
The frequency at which to take the snapshots.

Frequency: Every hour

Minute: At 20
[Create cron expression](#)

[Next](#) [Cancel](#)

elastic Search Elastic

Stack Management Snapshot and Restore Policies Add policy

Management

- Ingest
 - Ingest Pipelines
- Data
 - Index Management
 - Index Lifecycle Policies
 - Snapshot and Restore**
 - Rollup Jobs
 - Transforms
 - Remote Clusters
- Alerts and Insights
 - Rules and Connectors
 - Reporting
 - Machine Learning Jobs
- Kibana
 - Index Patterns
 - Saved Objects
 - Tags
 - Search Sessions
 - Snaps

Logistics **Snapshot settings** **Snapshot retention** **Review**

Snapshot settings [Snapshot docs](#)

Data streams and indices
To back up indices and data streams, manually select them or define index patterns to dynamically match them.

☒ All data streams and indices, including system indices

Index patterns: nsdb [Select data streams and indices](#)

Ignore unavailable indices
Ignores indices that are unavailable when taking the snapshot. Otherwise, the entire snapshot will fail.

☒ Ignore unavailable indices

Allow partial indices
Allows snapshots of indices with primary shards that are unavailable. Otherwise, the entire snapshot will fail.

☒ Allow partial indices

Include global state
Stores the global cluster state and system indices as part of the snapshot.

☒ Include global state

[Back](#) [Next](#) [Cancel](#)

elastic

Search Elastic

Stack ManagementSnapshot and RestorePoliciesAdd policy

Management

Ingest
Ingest Pipelines

Data
Index Management
Index Lifecycle Policies
Snapshot and Restore
Rollup Jobs
Transforms
Remote Clusters

Alerts and Insights
Rules and Connectors
Reporting
Machine Learning Jobs

Kibana
Index Patterns
Saved Objects
Tags
Search Sessions
Spaces

Create policy

Logistics

Snapshot settings

Snapshot retention

Review

Snapshot retention (optional)

Expiration

The time to wait before deleting snapshots.

Delete after

10

days

Snapshot docs

Snapshots to retain

The minimum and maximum number of snapshots to store for the policy.

Minimum count

5

Maximum count

10

Back

Next

Cancel

elastic

Search Elastic

Stack ManagementSnapshot and RestorePoliciesAdd policy

Management

Ingest
Ingest Pipelines

Data
Index Management
Index Lifecycle Policies
Snapshot and Restore
Rollup Jobs
Transforms
Remote Clusters

Alerts and Insights
Rules and Connectors
Reporting
Machine Learning Jobs

Kibana
Index Patterns
Saved Objects
Tags
Search Sessions
Spaces

Review policy

SummaryRequest

Logistics

Policy name
iudx-snapshot

Repository
iudx-repo

Snapshot settings

Data streams and indices
• **nsdb**

Allow partial indices
No

Snapshot retention

Delete after
10d

Min count
5

Max count
10

Snapshot name
<nsdb-snap-{now/d}>

Schedule
0 20 * * * ?

Ignore unavailable indices
No

Include global state
No

So successfully it's created here.

The screenshot shows the Elastic Stack Management console. The left sidebar contains a 'Management' menu with sections for Ingest, Data, Alerts and Insights, and Kibana. The main content area is titled 'Snapshot and Restore' and includes a sub-header 'Policies'. Below this, there's a text box showing the cron schedule for retaining snapshots: '0 30 1 * * ?'. A table lists the policies, with one policy named 'iudx-snapshot' having a schedule of '0 20 * * * ?'. The table columns include Policy, Snapshot name, Repository, Schedule, Retention, Next snapshot, and Actions.

Policy	Snapshot name	Repository	Schedule	Retention	Next snapshot	Actions
<input type="checkbox"/> iudx-snapshot	<nsdb-snap-{now/d}>	iudx-repo	0 20 * * * ?	✓	Feb 15, 2022 1:50 PM	▶ ✎ 🗑️

It will take snapshot according to define period
Here we can see taken snapshot

The screenshot shows the Elastic Stack Management console with the 'Snapshots' tab selected. The main content area displays a table of snapshots. The table columns include Snapshot, Repository, Indices, Shards, Failed shards, Date created, Duration, and Actions. One snapshot is listed with the name 'nsdb-snap-2022.02.15-bpqcstwttykn-upnz_ffgq' and a duration of 2s.

Snapshot	Repository	Indices	Shards	Failed shards	Date created	Duration	Actions
<input type="checkbox"/> nsdb-snap-2022.02.15-bpqcstwttykn-upnz_ffgq	iudx-repo	1	1	0	Feb 15, 2022 5:52 PM GMT+5:30	2s	📄 🗑️

Now I am going to restore data from the snapshot, make sure your index should be deleted . otherwise during restore the data conflict comes up.

3.7 Step 7:

We can see nsdb index is present right now by the following query

The screenshot shows the Elastic Dev Tools console with the following content:

```

1 GET _search
2 {
3   "query": {
4     "match_all": {}
5   }
6 }
7
8 GET _cat/indices
9

```

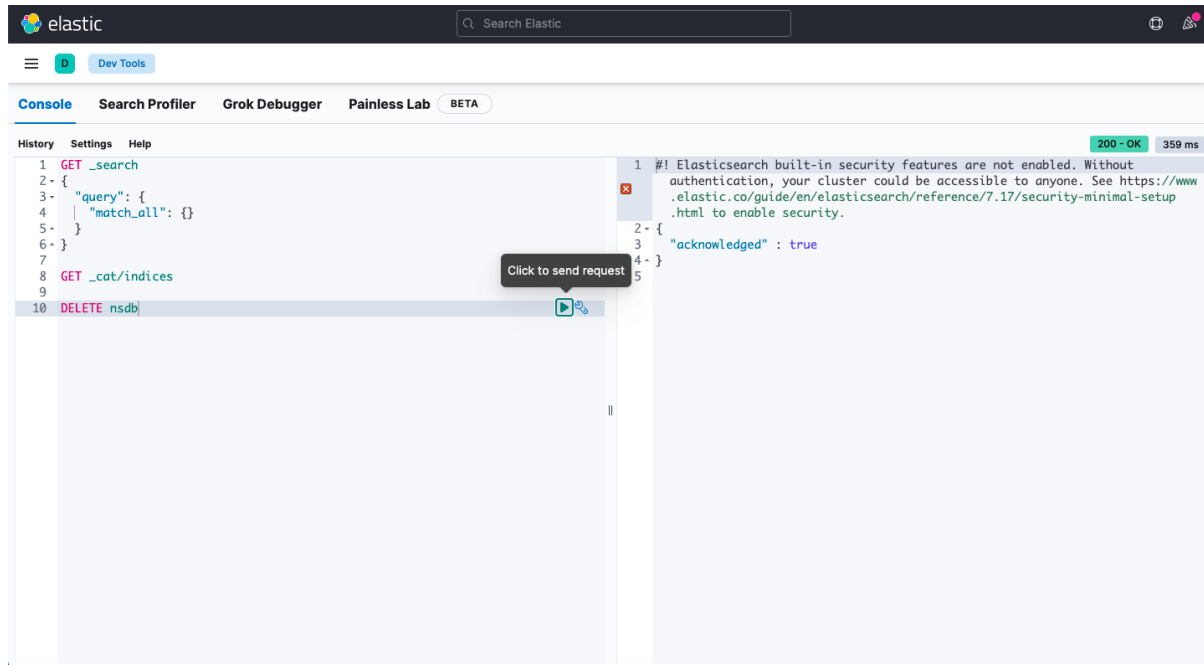
The response for the GET _cat/indices query is as follows:

```

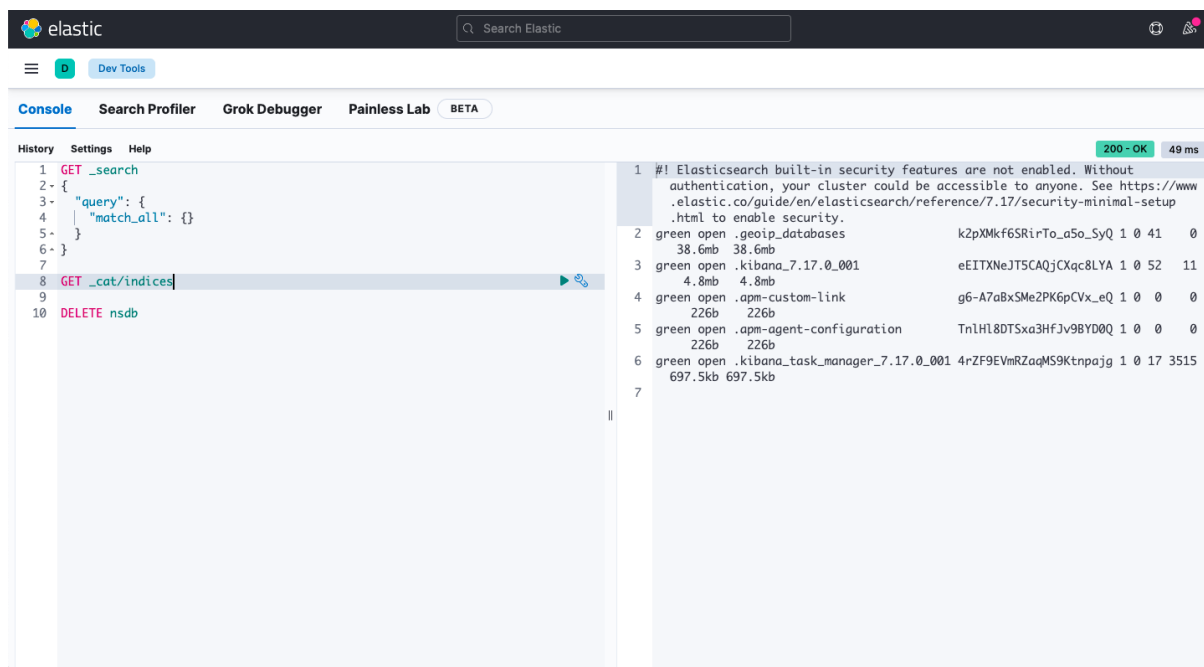
1 #! Elasticsearch built-in security features are not enabled. Without
2 authentication, your cluster could be accessible to anyone. See https://www
3 .elastic.co/guide/en/elasticsearch/reference/7.17/security-minimal-setup
4 .html to enable security.
5 green open .geoip_databases k2pXMkf6SRirTo_a5o_SyQ 1 0 41
6 0 38.6mb 38.6mb
7 yellow open nsdb aLD2WlGxQe0YfKeqZY3j7w 1 1 3
8 0 14.7kb 14.7kb
9 green open .kibana_7.17.0_001 eEITXNeJT5CAQjCXqc8LYA 1 0 50
10 11 4.8mb 4.8mb
11 green open .apm-custom-link g6-A7aBx5Me2PK6pCVx_eQ 1 0 0
12 0 226b 226b
13 green open .apm-agent-configuration Tn1Hl8DTSxa3HfJv98YD0Q 1 0 0
14 0 226b 226b
15 green open .kibana_task_manager_7.17.0_001 4rZF9EVmRZaqMS9Ktnpajg 1 0 17
16 3335 631.3kb 631.3kb
17
18

```

Let's delete the nsdb index for further operation, such that we can test the restore.



Again check index and you can see nsdb is deleted successfully.



elastic Search Elastic

Stack Management Index Management

Management

- Ingest
 - Ingest Pipelines
- Data
 - Index Management**
 - Index Lifecycle Policies
 - Snapshot and Restore
 - Rollup Jobs
 - Transforms
 - Remote Clusters
- Alerts and Insights
 - Rules and Connectors
 - Reporting
 - Machine Learning Jobs
- Kibana
 - Index Patterns
 - Saved Objects
 - Tags
 - Search Sessions
 - Snapses

Index Management

[Index Management docs](#)

Indices Data Streams Index Templates Component Templates

Update your Elasticsearch indices individually or in bulk. [Learn more.](#)

☐ Include rollup indices ☐ Include hidden indices

Search

Reload indices

No indices to show

Let's restore this nsdb index again

elastic Search Elastic

Stack Management Snapshot and Restore Snapshots

Management

- Ingest
 - Ingest Pipelines
- Data
 - Index Management
 - Index Lifecycle Policies
 - Snapshot and Restore**
 - Rollup Jobs
 - Transforms
 - Remote Clusters
- Alerts and Insights
 - Rules and Connectors
 - Reporting
 - Machine Learning Jobs
- Kibana
 - Index Patterns
 - Saved Objects
 - Tags
 - Search Sessions

Snapshot and Restore

[Snapshot and Restore docs](#)

Use repositories to store and recover backups of your Elasticsearch indices and clusters.

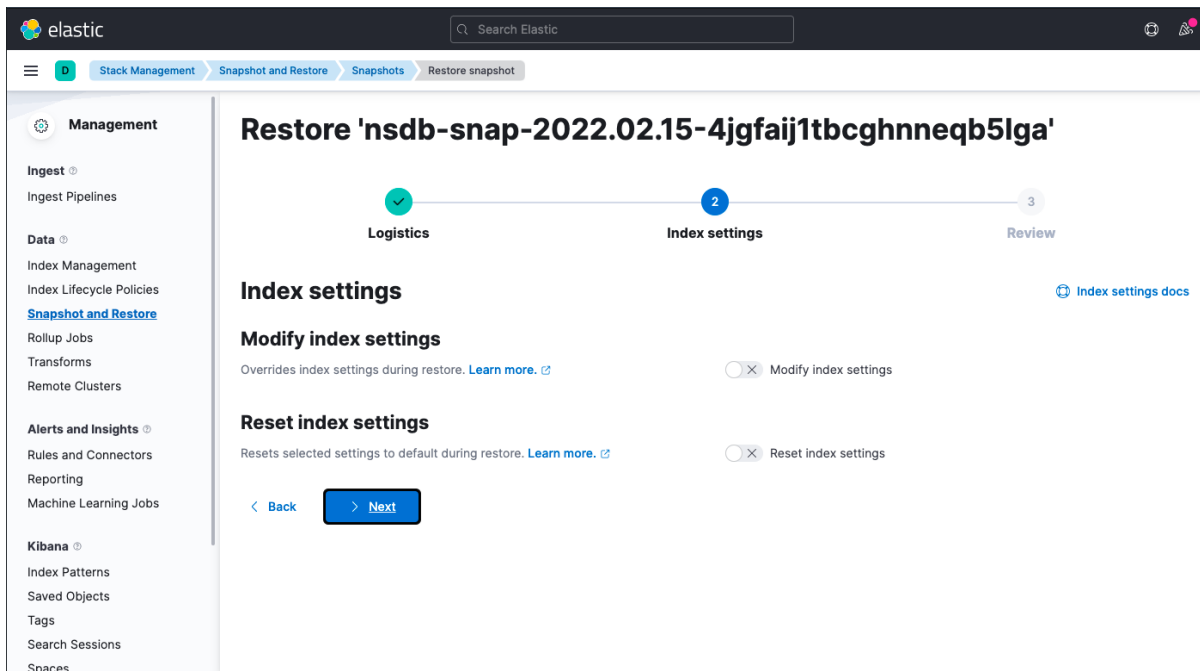
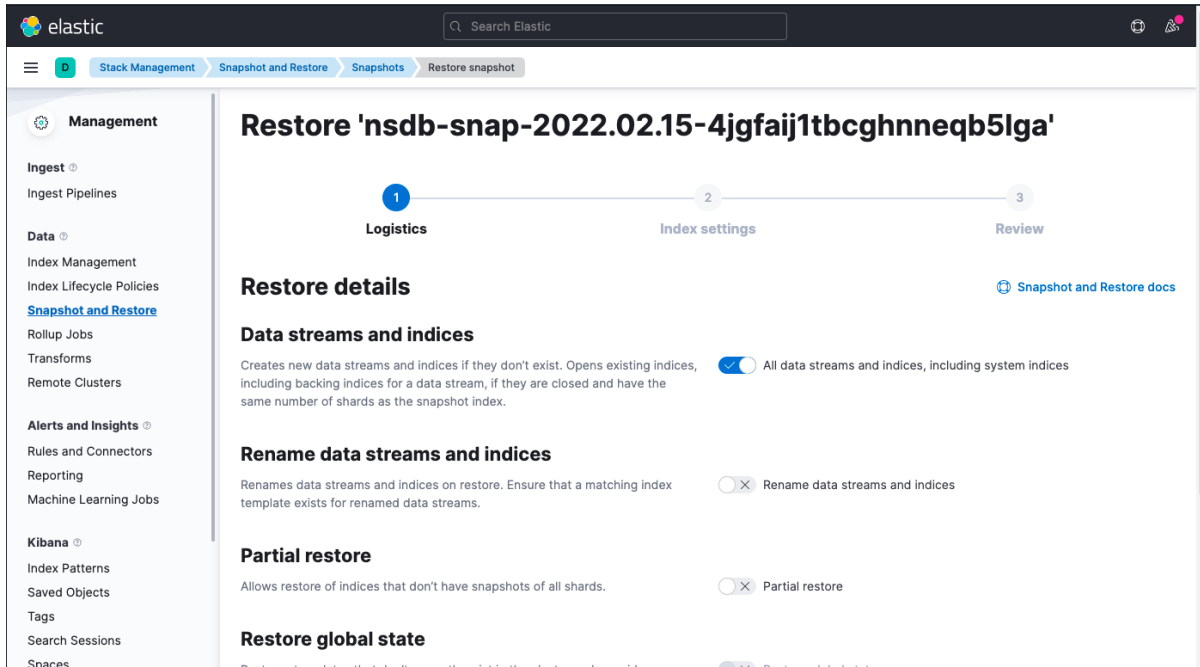
Snapshots Repositories Policies Restore Status

Delete snapshot Search... Repository Reload

<input type="checkbox"/> Snapshot	Repository	Indices	Shards	Failed shards	Date created ↓	Duration	Restore
<input checked="" type="checkbox"/> nsdb-snap-2022.02.15-4jgfaj1tbcghnneqb5lga	iudx-repo	1	1	0	Feb 15, 2022 6:49 PM GMT+5:30	1s	Restore
<input type="checkbox"/> nsdb-snap-2022.02.15-bpqcsbtwtykn-upnz_ffgq	iudx-repo	1	1	0	Feb 15, 2022 5:52 PM GMT+5:30	2s	Restore

Rows per page: 20

localhost:5601/app/management/data/snapshot_restore/restore/iudx-repo/nsdb-snap-2022.02.15-4jgfaj1tbcghnneqb5lga



Restored successfully, you can check the status in following images

Snapshot and Restore

Use repositories to store and recover backups of your Elasticsearch indices and clusters.

Snapshots Repositories Policies Restore Status

Refresh data every 30 seconds ▾

Index	Status ↑	Last activity	Shards completed	Shards in progress
nsdb	● Complete	Feb 15, 2022 7:02 PM GMT+5	1	0

Rows per page: 20 ▾

< 1 >

Index Management

Update your Elasticsearch indices individually or in bulk. [Learn more.](#)

Indices Data Streams Index Templates Component Templates

Search

☐ Include rolup indices ☐ Include hidden indices

[Reload indices](#)

Name	Health	Status	Primaries	Replicas	Docs count	Storage size	Data stream
nsdb	● yellow	open	1	1	3	14.7kb	

Rows per page: 10 ▾

< 1 >

So this is all about my approach and solution with the Help of Graphical User Interface .

Chapter 4

Code Approach:

Now I will Restore index with the Curator. If we want to restore data through code then we use the technology curator.

First go inside the Elastic search container by Following Command, because we are storing snapshots in ElasticSearch

```
docker exec -it elasticsearch bash
```

Update the old package of the Elastic search container

```
apt-get update
```

Now we will install Curator through pip library but first we need to install pip library

```
apt-get install pip -y
```

Now pip library is installed, let's install curator by following command :

```
pip install elasticsearch-curator
```

Now we installed curator successfully, if you want to check that it's successfully installed or not the hit following command

```
Curator --help
```

And output will be like

Curator for Elasticsearch indices.

See <http://elastic.co/guide/en/elasticsearch/client/curator/current>

Options:

- config PATH Path to configuration file. Default: ~/.curator/curator.yml
- dry-run Do not perform any changes.
- version Show the version and exit.
- help Show this message and exit.

So you successfully installed curator

Now for storing index with the help of curator we need two things

1: curator configuration file

2: curator action file

Configuration file will tell us from where we want to store the index and the action file will tell us what to store.

Let's write configuration file

Vim curator_config.yaml

```
client:
  hosts:
    - 127.0.0.1
  port: 9200
  url_prefix:
  use_ssl: False
  certificate:
  client_cert:
  client_key:
  ssl_no_validate: False
  username:
  password:
  timeout: 30
  master_only: False
```

```

logging:
  loglevel: INFO
  logfile:
  logformat: default
  blacklist:

```

Save it.

Now let's write an action file such as which index we want to store and from which repo .

Vim curator_action.yaml

```

actions:
1:
  action: restore
  description: >-
    Restore all indices in the most recent snapshot with state SUCCESS. Wait
    for the restore to complete before continuing. Do not skip the repository
    filesystem access check. Use the other options to define the index/shard
    settings for the restore.
  options:
    repository: nitc
    # If name is blank, the most recent snapshot by age will be selected
    name:
    # If indices is blank, all indices in the snapshot will be restored
    indices:
    wait_for_completion: True
    max_wait: 3600
    wait_interval: 10
  filters:
    - filtertype: state
    state: SUCCESS
    exclude:

```

Save the configuration file

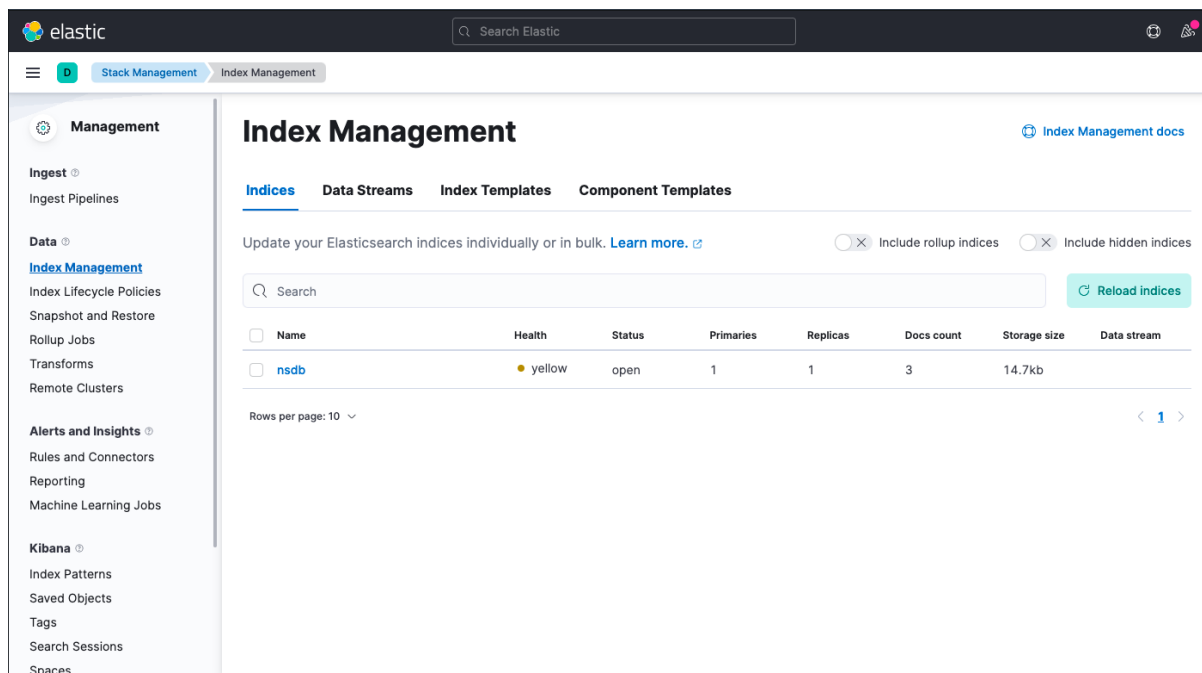
Now we have both the file curator action and curator config, let's store the deleted index.

Make sure the index is deleted before restore .

Apply the following command

```
curator --config curator_config.yaml curator_action.yaml
```

You will see the job completed, You can see that index is restored successfully.



So this is all about my code approach.

Chapter 5

Results:

- Elastic Search Head Plugin working Successfully for data ingestion in ElasticSearch.
- Successfully launched the ElasticSearch and kibana container and exposed it.
- Applied Index life cycle successfully for transferring the data into different phases
- Successfully Created snapshots policy for taking snapshots before deleting the data.
- Deleted the index successfully by Dev tools.
- Restored the index successfully by graphical User interface and code Also.
- Successfully register the local directory for storing Snapshots.

Chapter 6

Conclusion :

Up to now i performed all steps Successfully in the project like bringing up operating system for elasticsearch and kibana, learned about data format, stored the data, ingested data into elasticsearch, founded the approach for transferring data from one phase to another, deleting the data, restoring the data, taking snapshots successfully.

we are able to meet the objectives. And if we talk about benefits they can be following

- Cost saving
- Better data management
- Better utilization of resources.
- Security of data in the matter of loss.

If we talk about disadvantages then one is we are storing snapshots locally. In the future we can store snapshots on cloud because local storage can be corrupted some time but cloud storage is so much more reliable in the matter of storage and security.

References:

1: Docker Container : <https://docs.docker.com/desktop/>

Accessed between : Jan 2022 - May 2022

2: Elastic Search

<https://www.elastic.co/guide/en/elasticsearch/reference/current/index.html>

Accessed between : Jan 2022 - May 2022

3: Kibana : <https://www.elastic.co/guide/en/kibana/current/index.html>

Accessed between : Jan 2022 - May 2022

4: Curator:

<https://www.elastic.co/guide/en/elasticsearch/client/curator/current/restore.html>

Accessed between : Jan 2022 - May 2022

5: Data Format : JSON (Javascript Object notation)

6: Plugin : Elasticsearch Head plugin