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

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

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 :

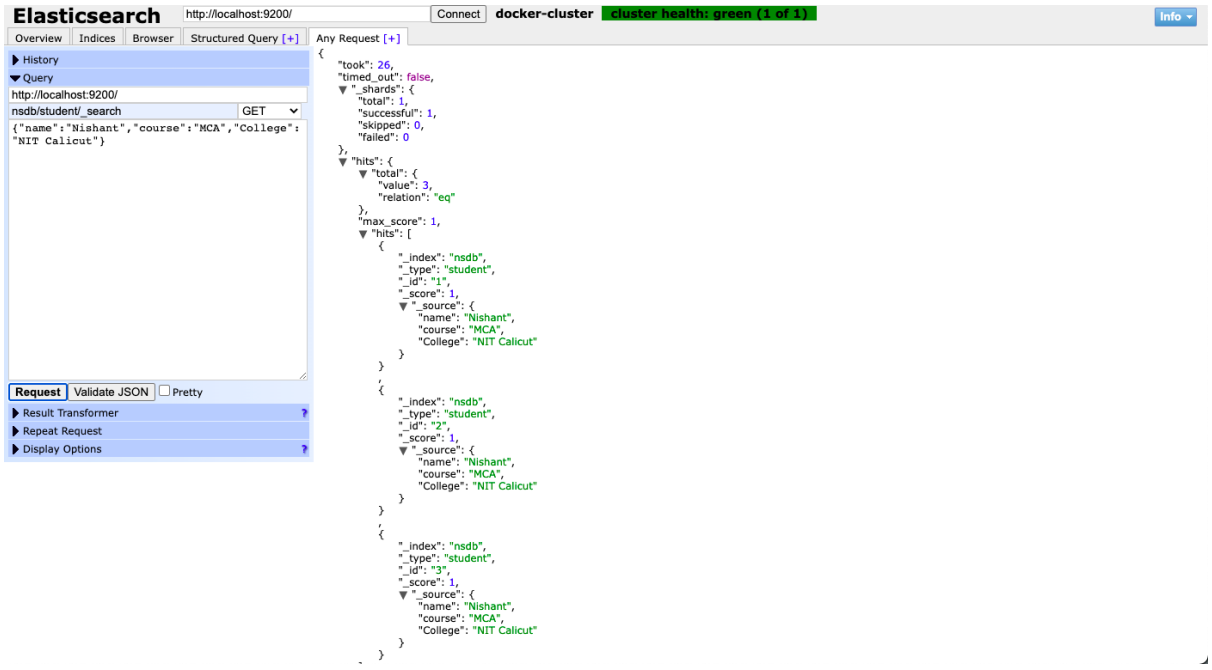
3.1 Step 1:

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

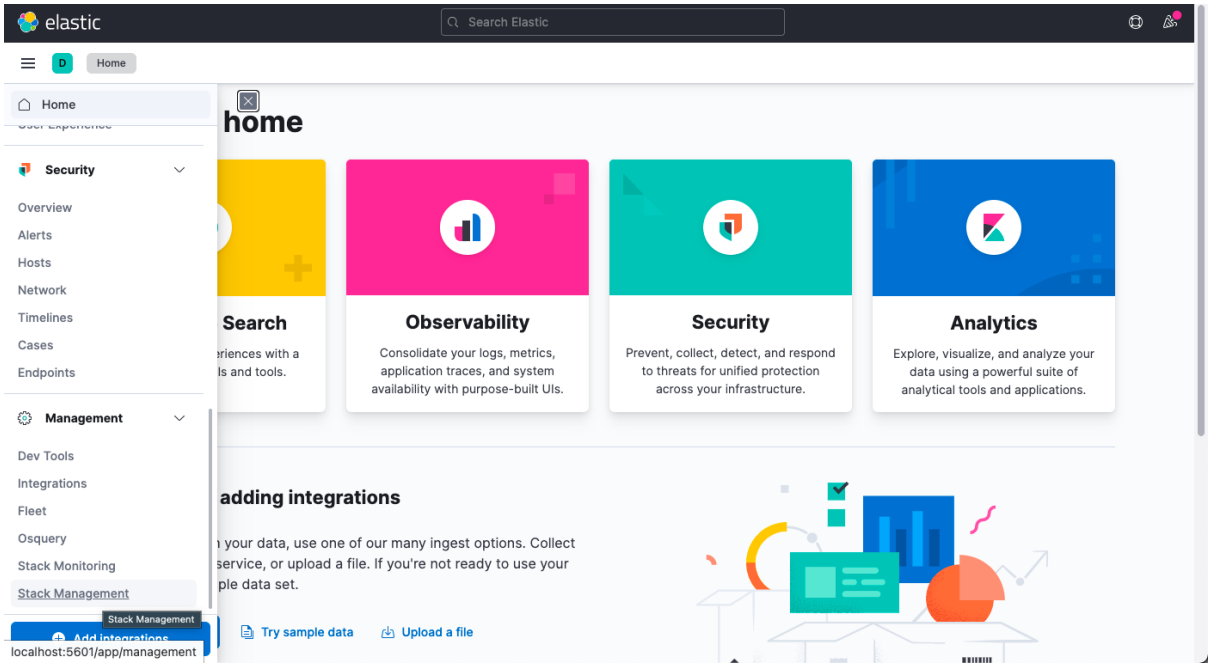
docker run -dit --name kibana --net elk -p 5601:5601 kibana:7.17.0

Opened the kibana on browser

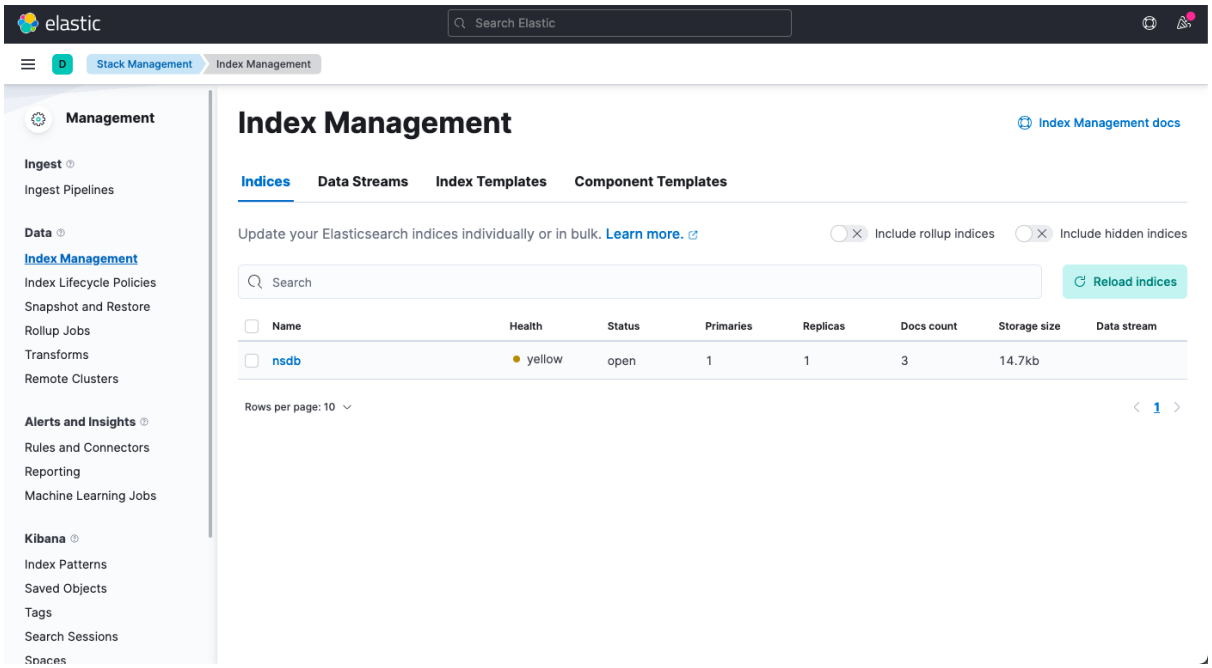
http:/localhost:5601/app/kibana

3.3 Step 3:

Opened the stack management section



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



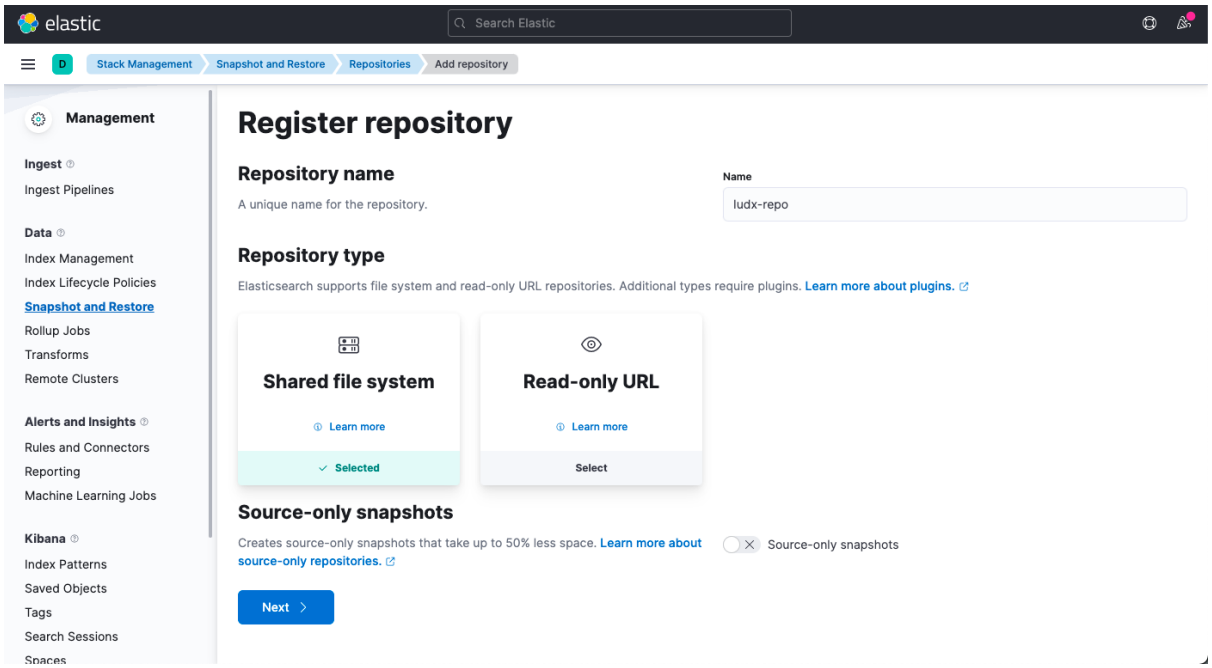
3.4 Step 4:

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/

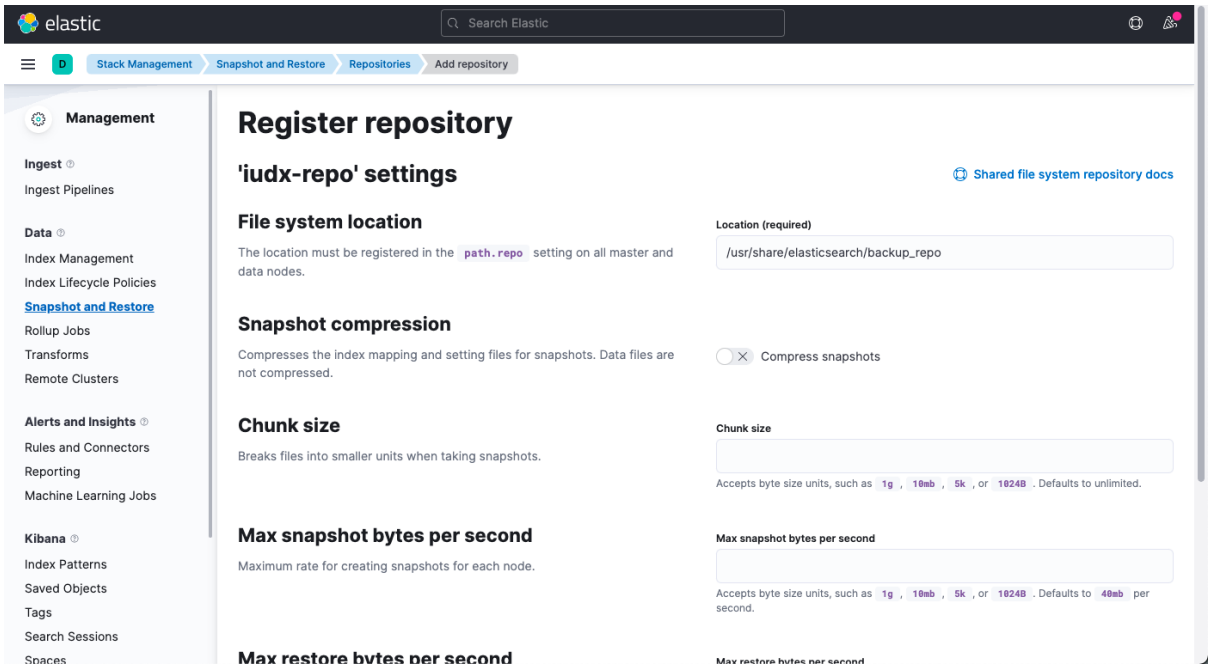
Update the elastic search configuration file and add the following line

Vim config/elasticsearch.yml

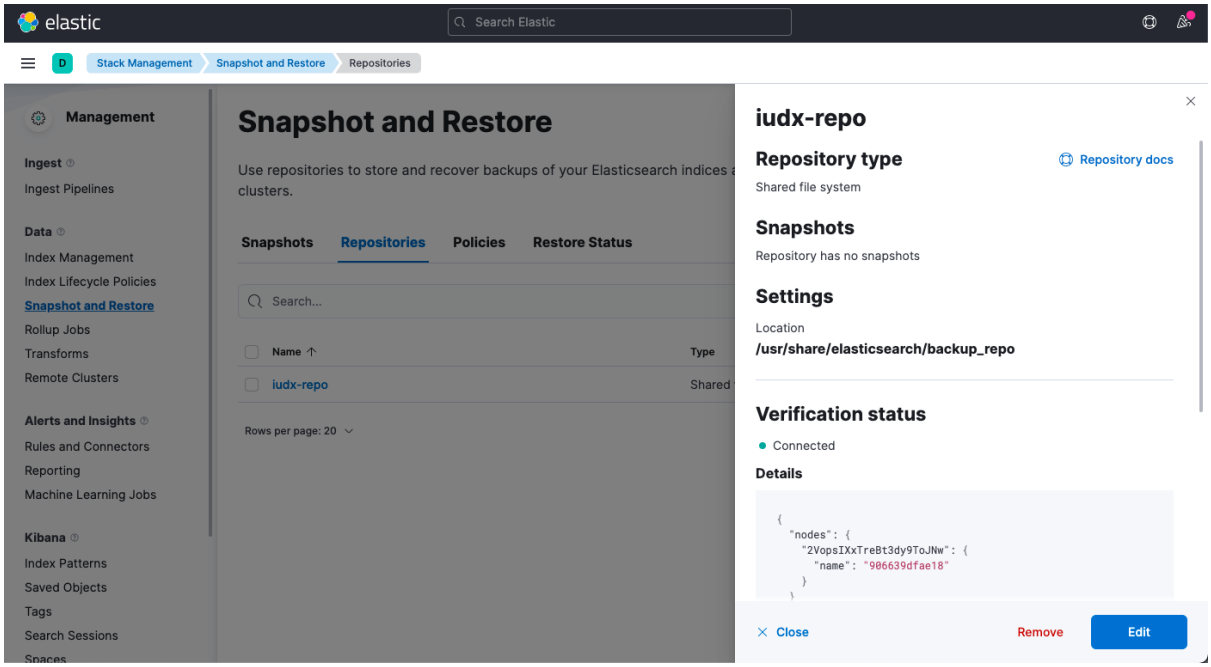
path.repo: /usr/share/elasticsearch/backup_repo

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

After this register repository with this setting



Check connectivity

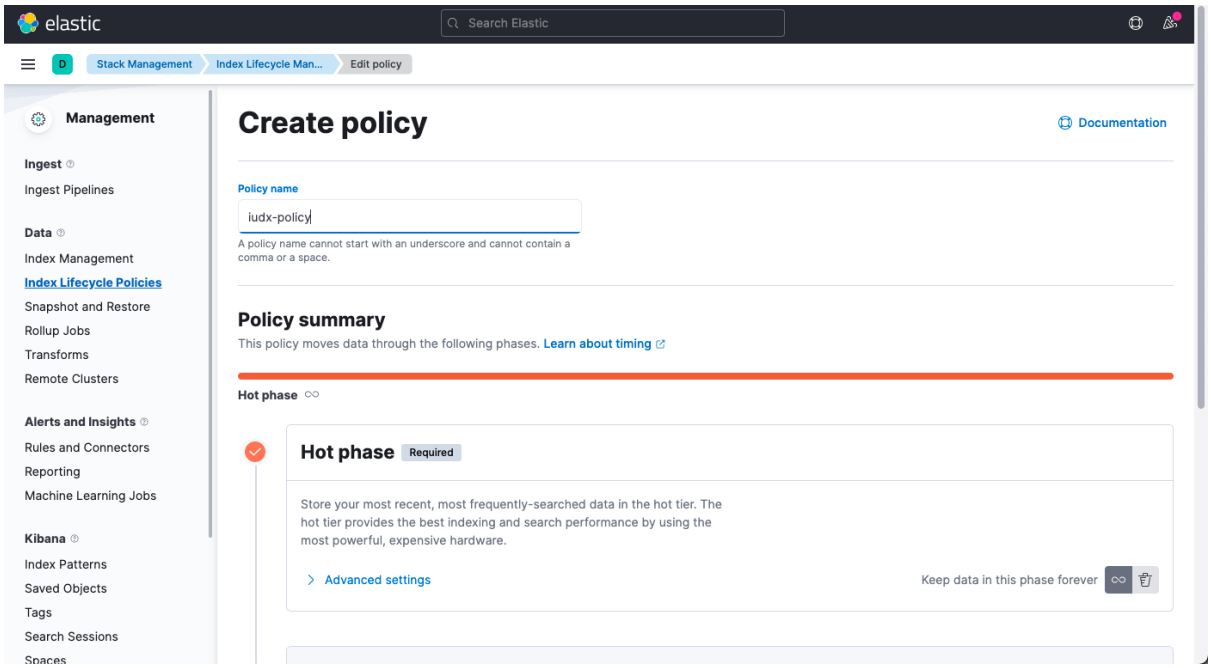


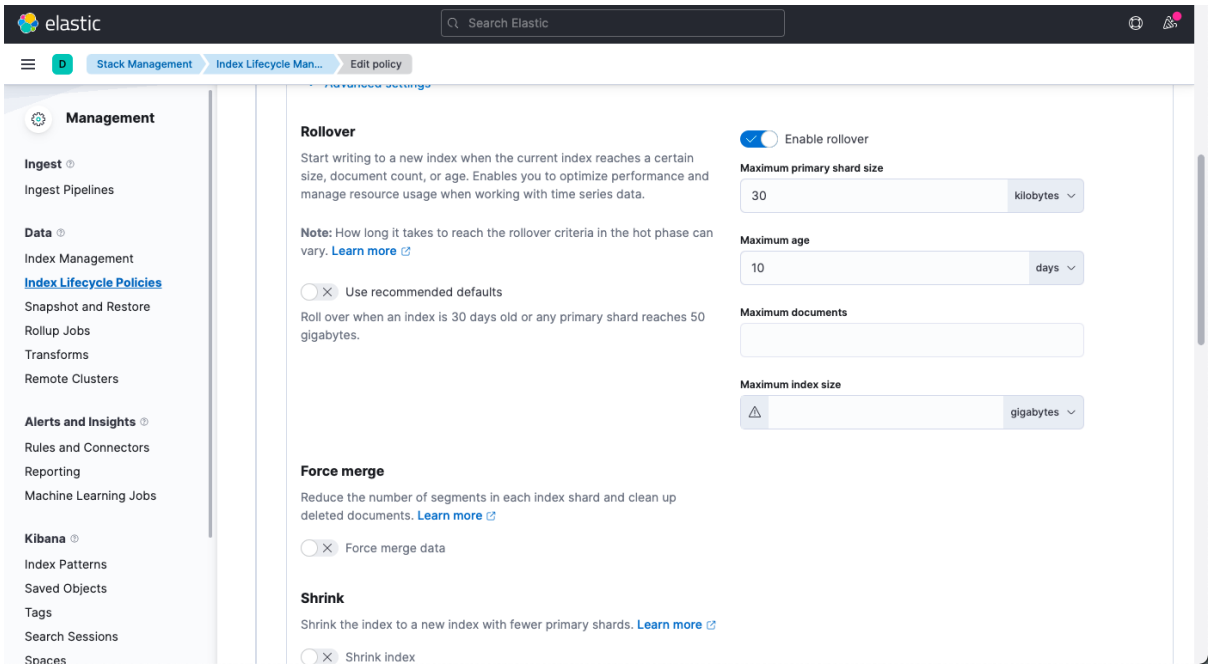
3.5 Step 5:

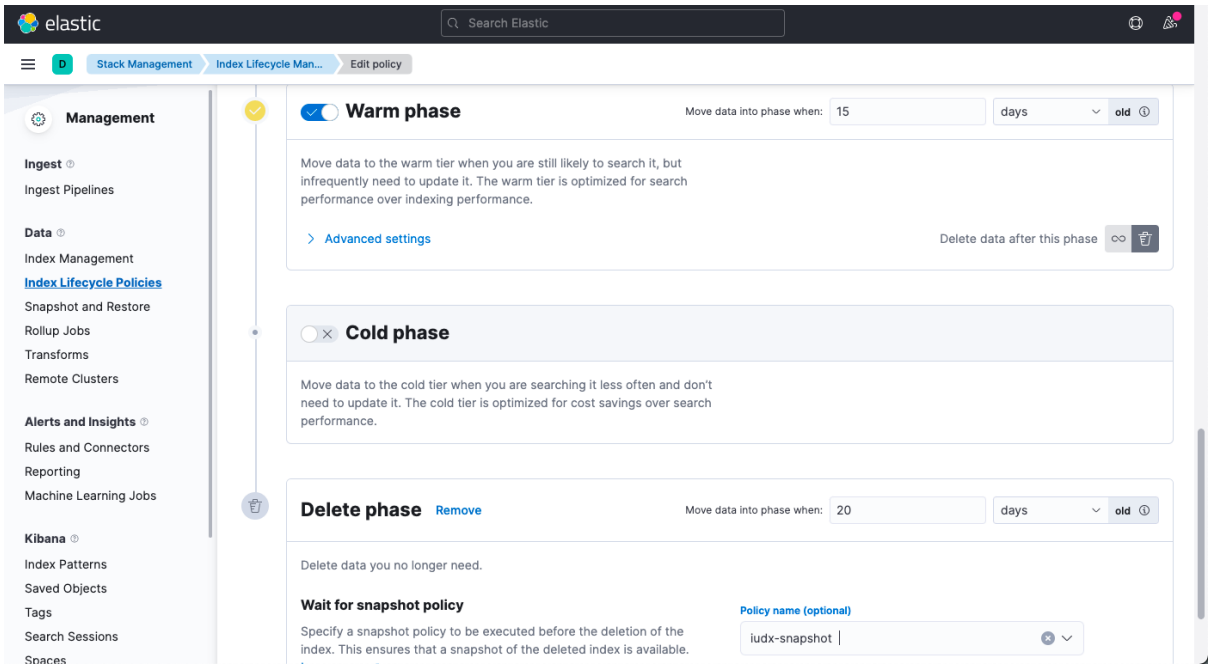
Now I will create index life cycle management for data management so that searching will be faster and old data is deleted and stored somewhere in the form of a snapshot.

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





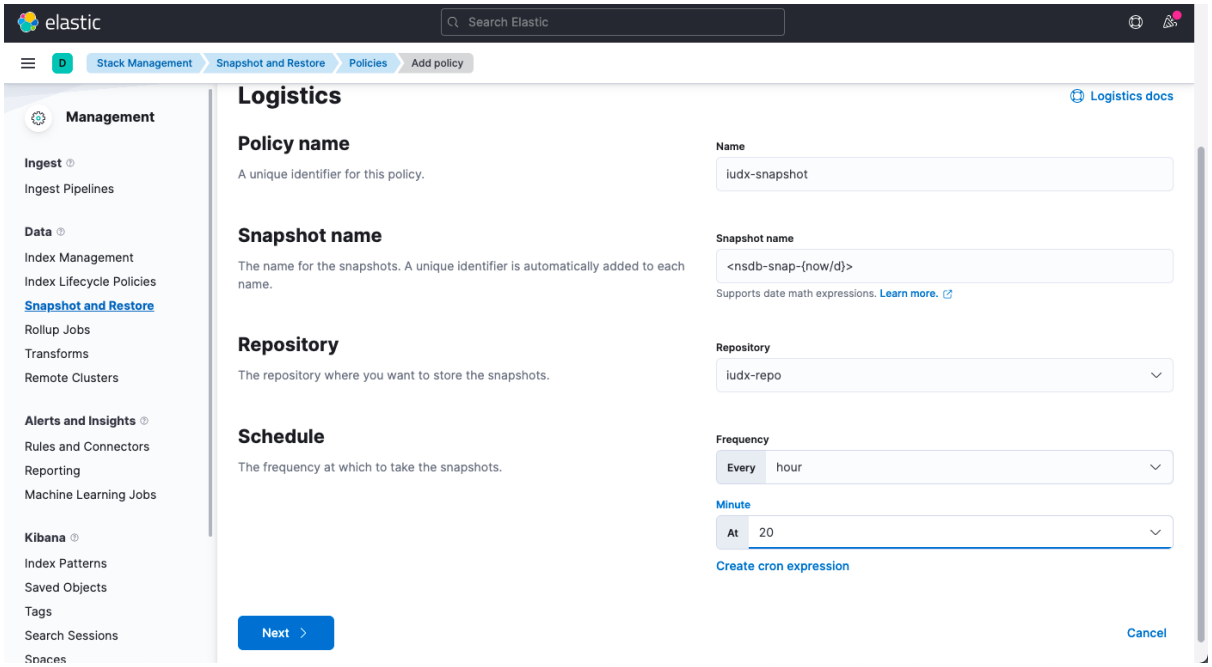


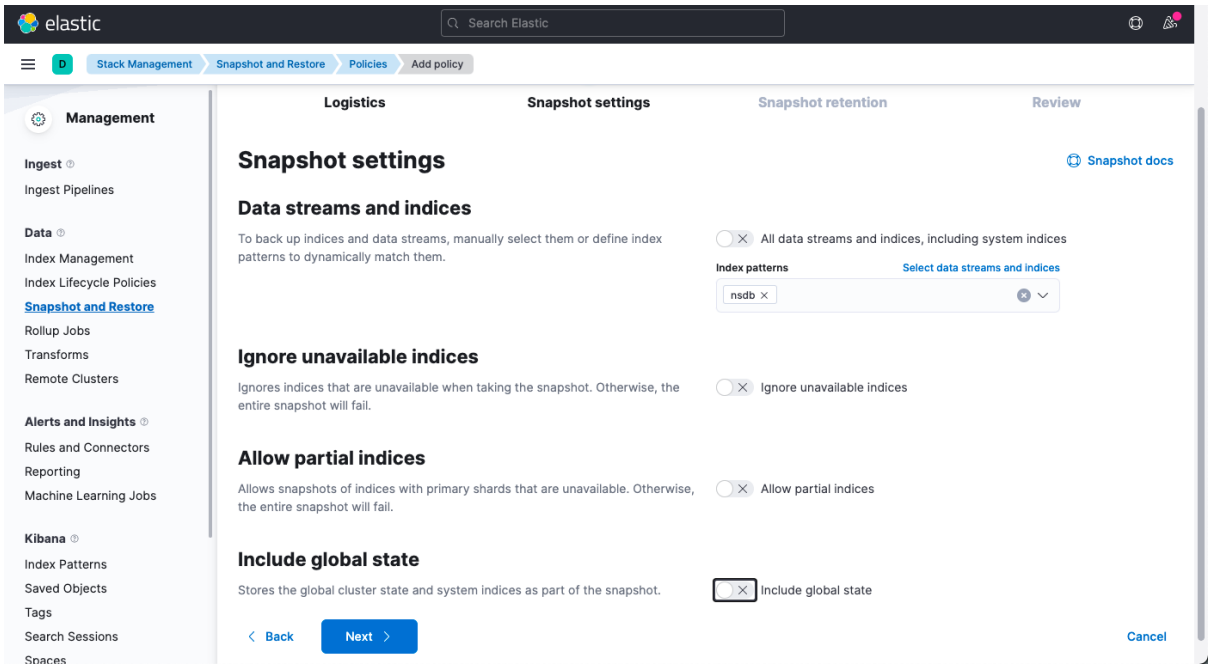
Policy created

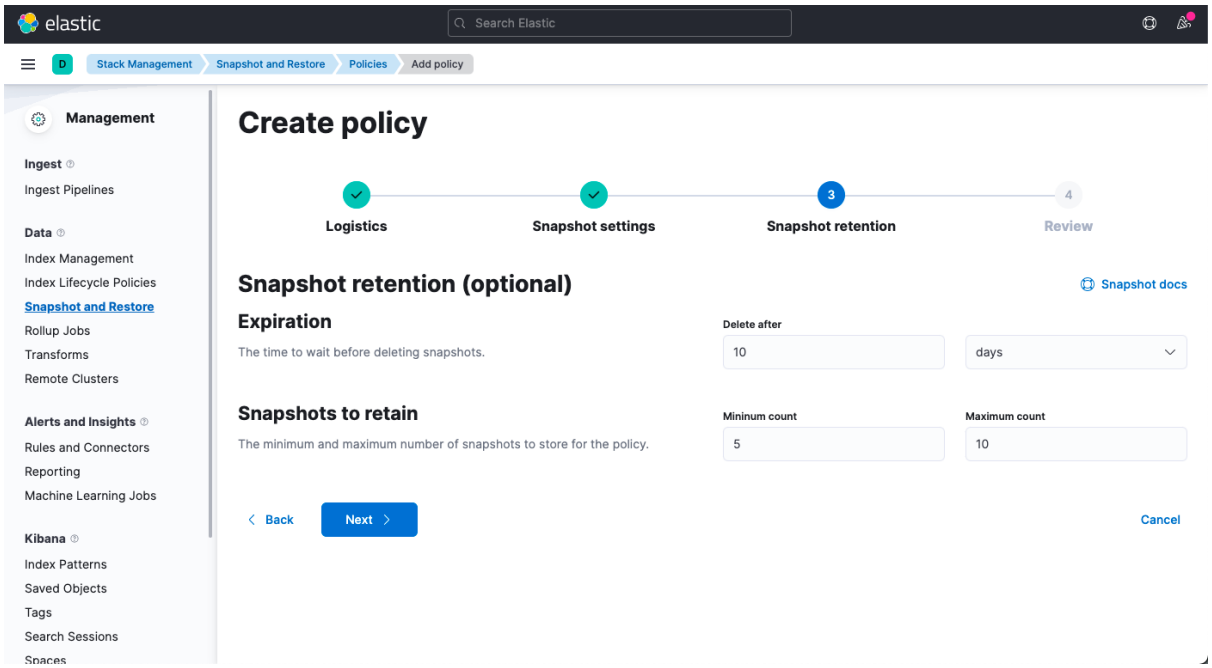
For policy creation we need to create a snapshot such that before deleting the data we can store it.In the future from this snapshot we can restore the data by curator or Kibana GUI.

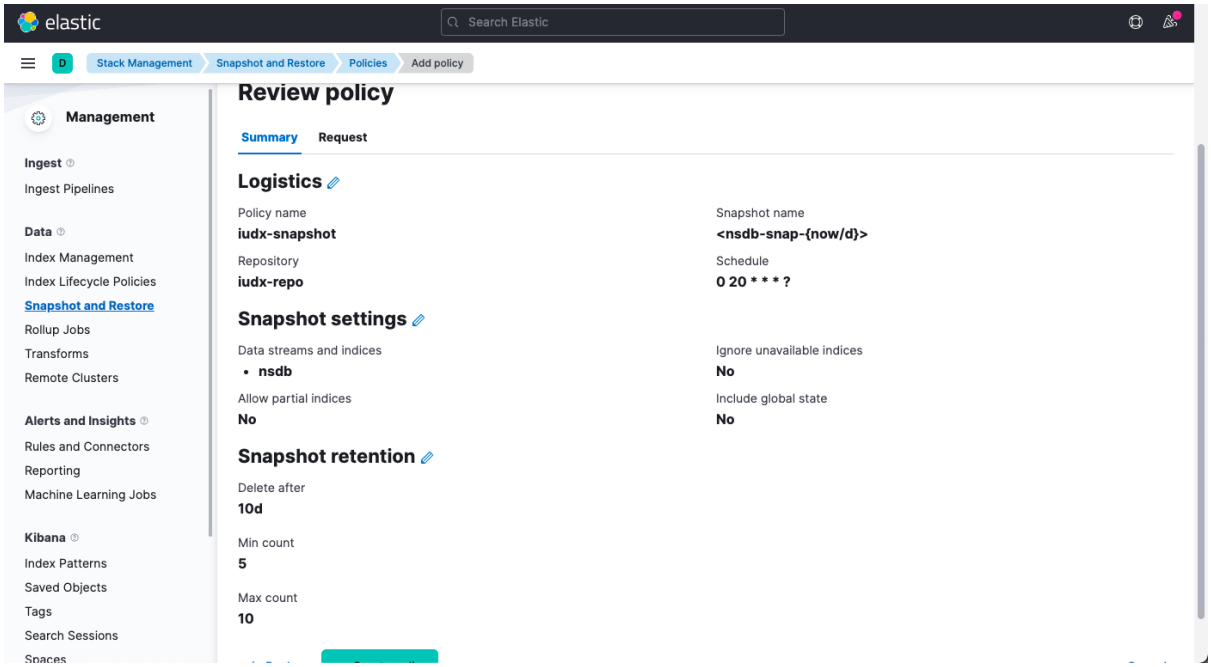
3.6 Step 6:

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

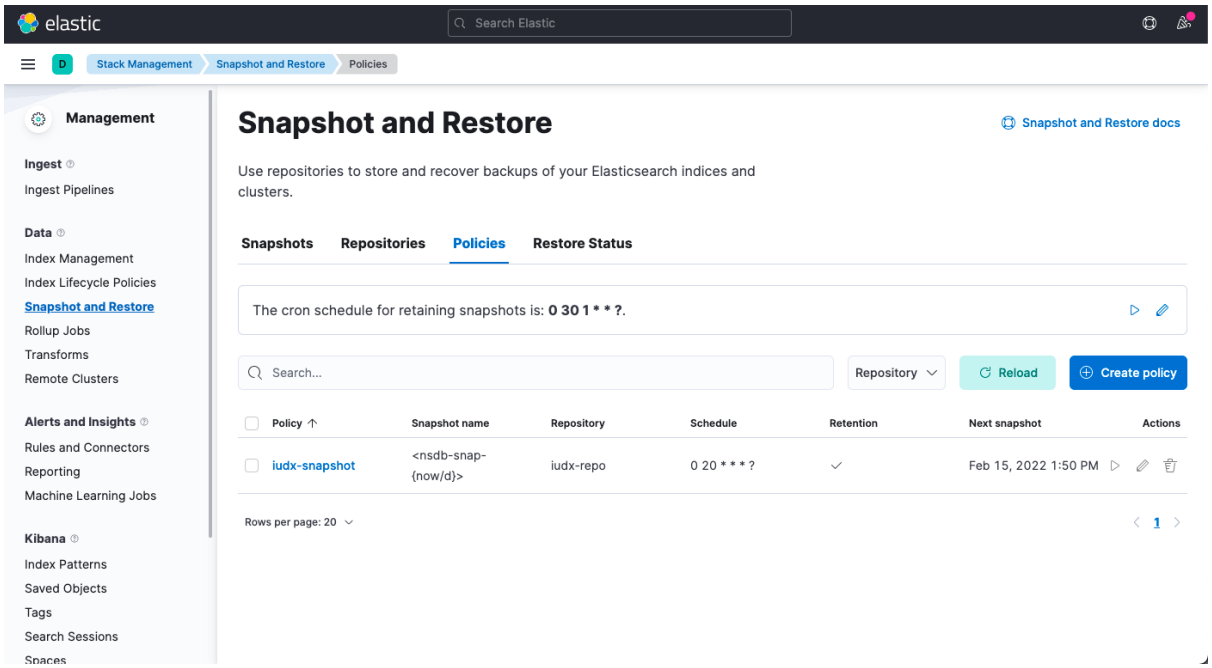




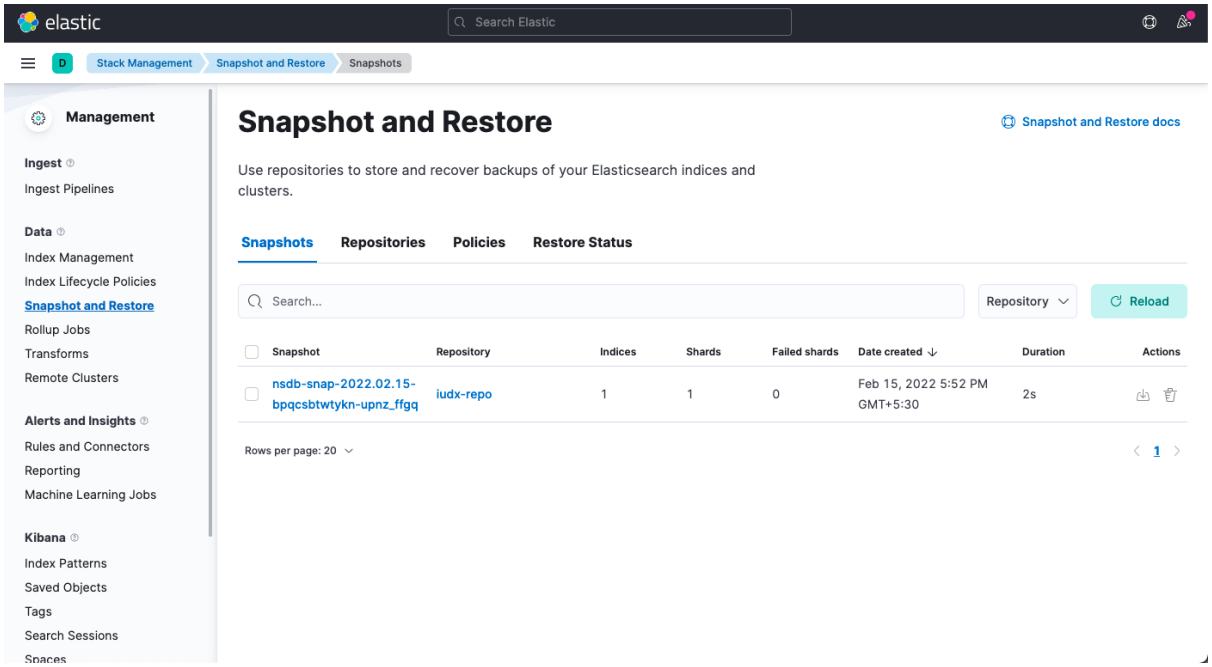

elastic
Search Elastic
Stack Management
Snapshot and Restore
Policies
Add policy
Management
Ingest
Ingest Pipelines
Data
Index Management
Index Lifecycle Policies
Snapshot and Restore
Rollup Jobs
Transforms
Remote Clusters
Alerts and Insights
Rules and Connectors
Reporting
Machine Learning Jobs
Kibana
Index Patterns
Saved Objects
Tags
Search Sessions
Spaces
Create policy
Logistics
Snapshot settings
Snapshot retention
Review
Snapshot retention (optional)
Snapshot docs
Expiration
The time to wait before deleting snapshots.
Delete after
10
days
Snapshots to retain
The minimum and maximum number of snapshots to store for the policy.
Mininum count
5
Maximum count
10
Back
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
Spaces
Review policy
Summary
Request
Logistics
Policy name
iudx-snapshot
Snapshot name
<nsdb-snap-{now/d}>
Repository
iudx-repo
Schedule
0 20 * * * ?
Snapshot settings
Data streams and indices
nsdb
Ignore unavailable indices
No
Allow partial indices
No
Include global state
No
Snapshot retention
Delete after
10d
Min count
5
Max count
10

So successfully it's created here.



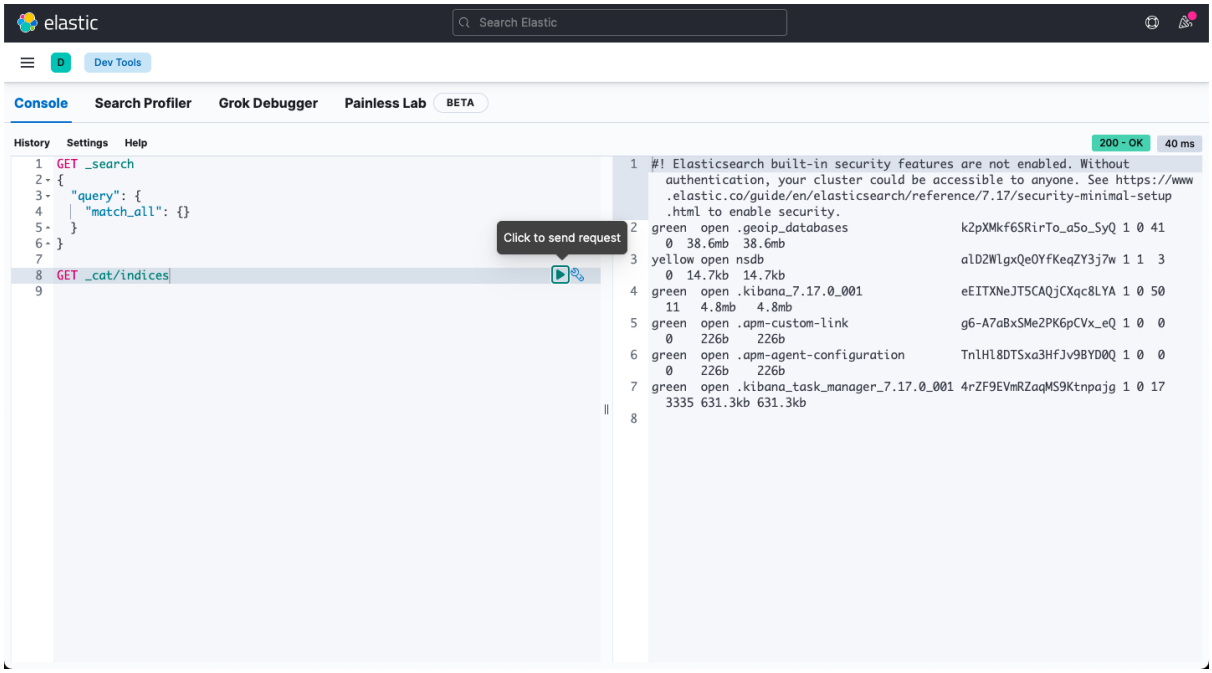
It will take snapshot according to define period
Here we can see taken snapshot



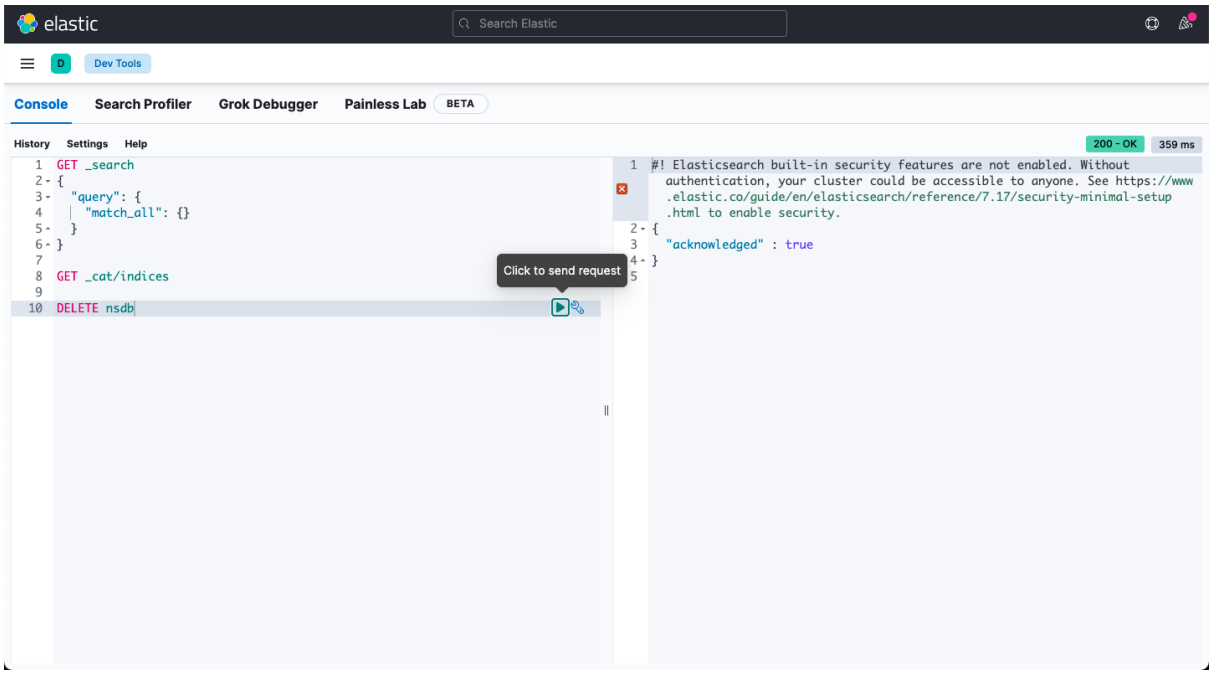
Now I am going to restore data from the snapshot, make sure your index should be deleted . otherwise during restore the data conflict comes up.

3.7 Step 7:

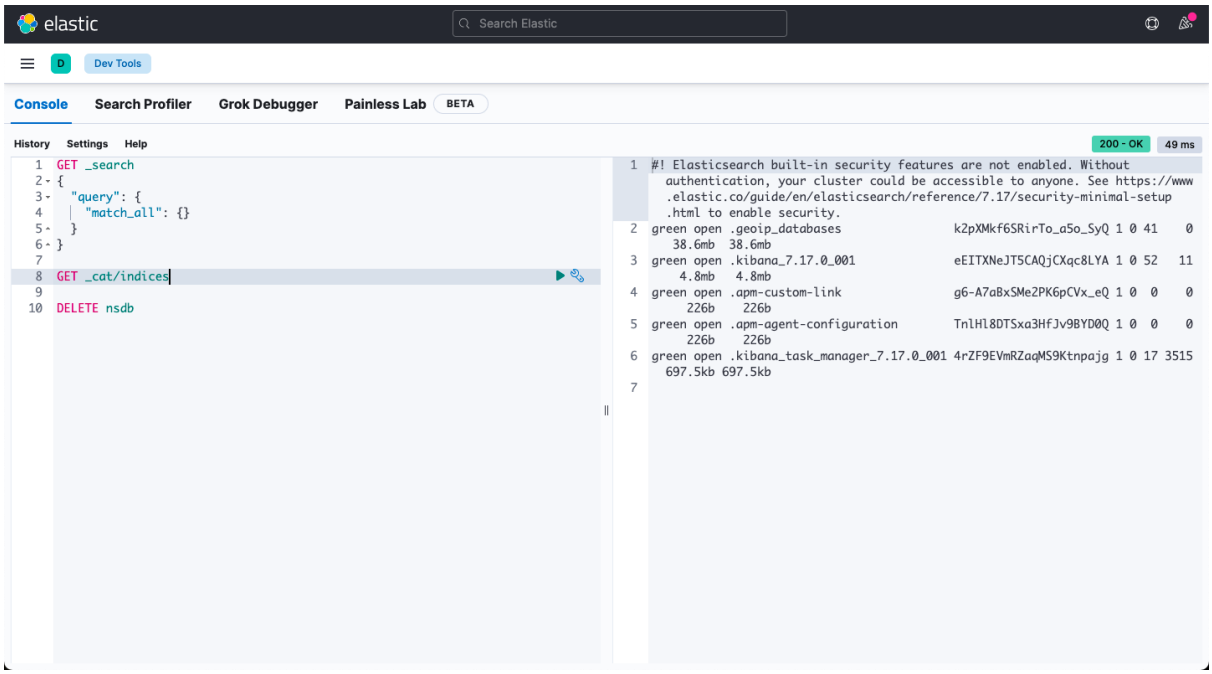
We can see nsdb index is present right now by the following query

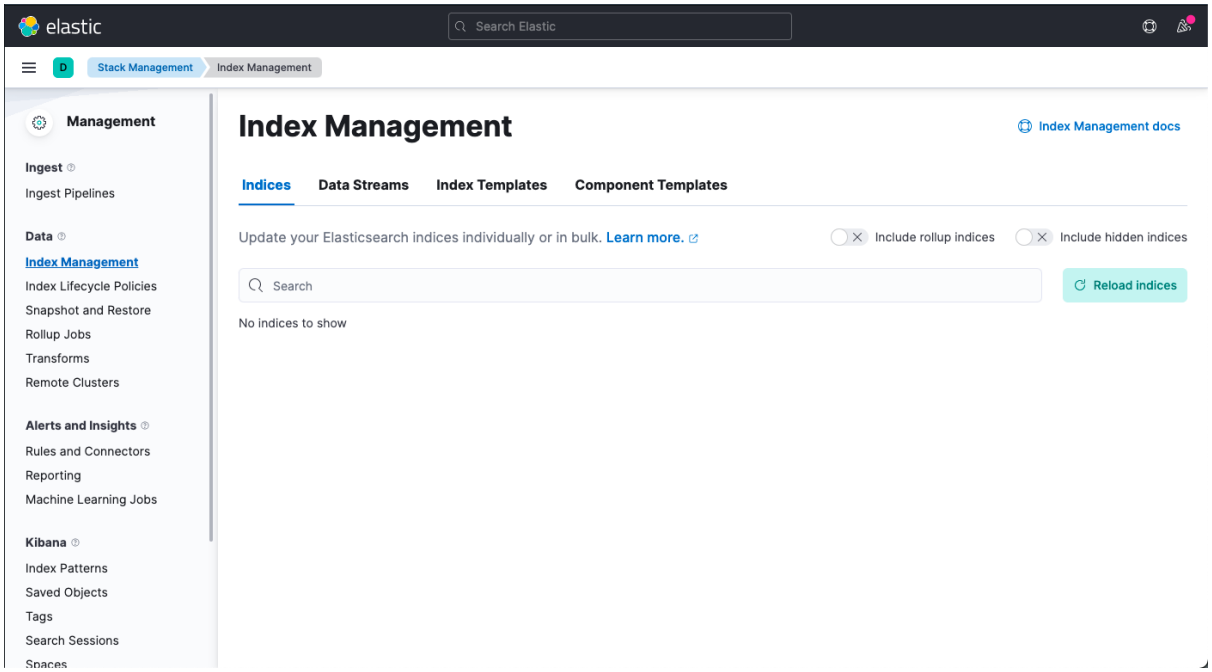


Let's delete the nsdb index for further operation, such that we can test the restore.

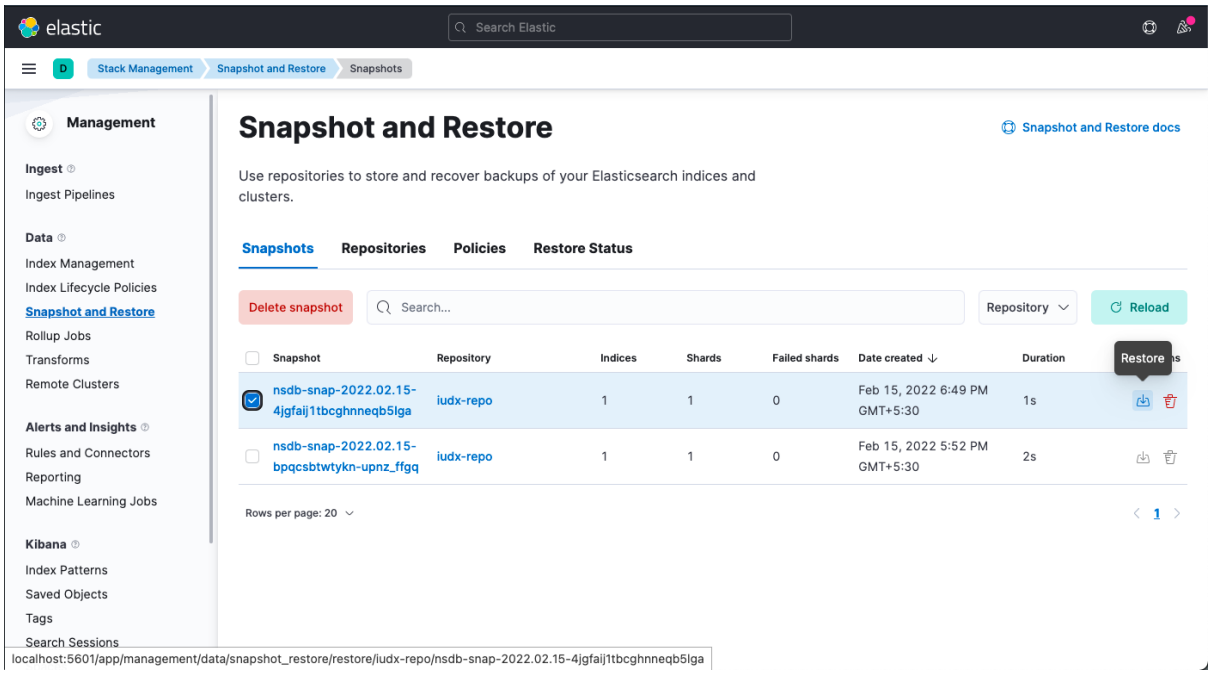


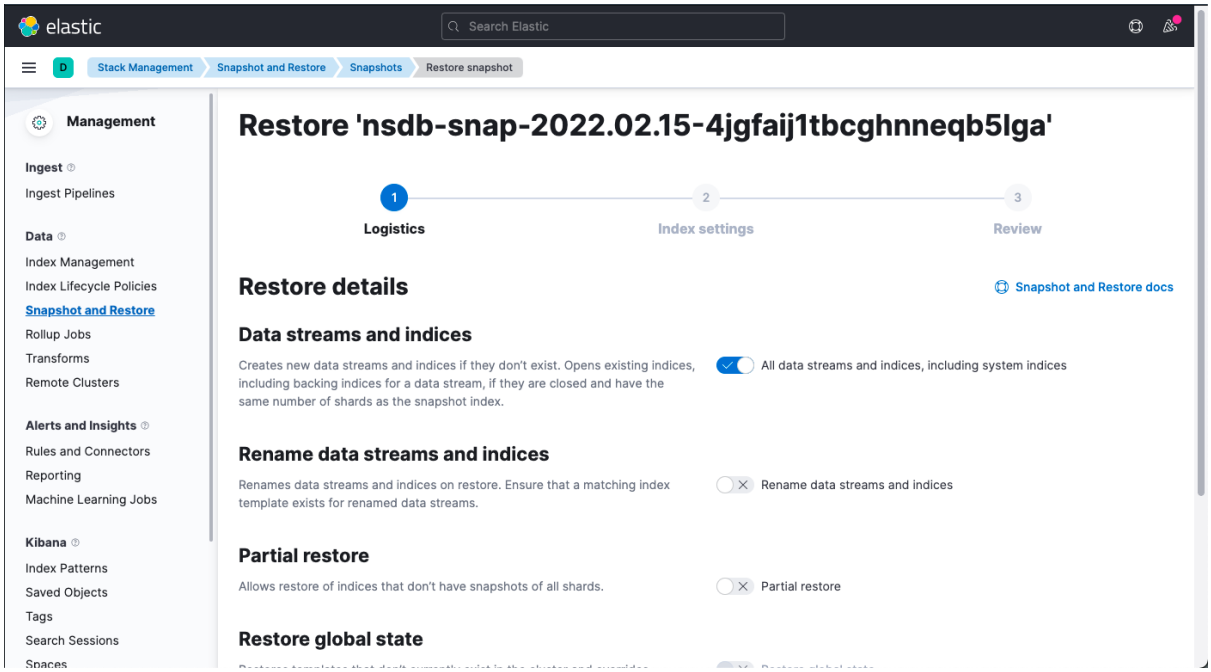
Again check index and you can see nsdb is deleted successfully.

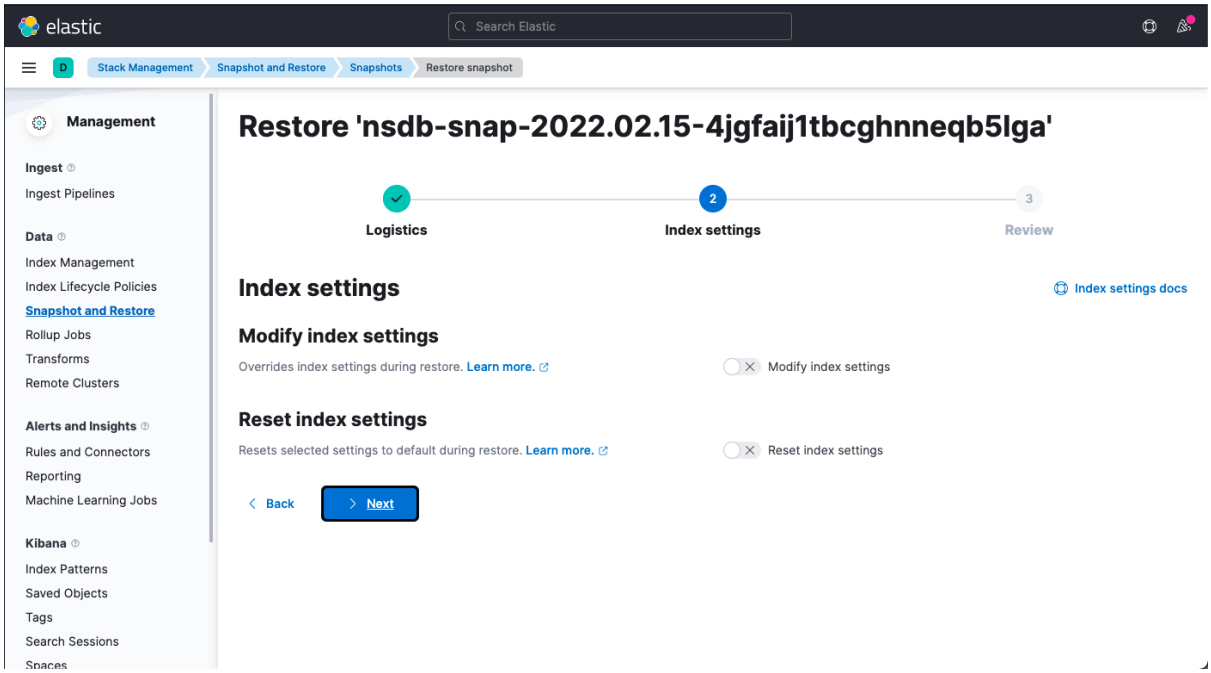




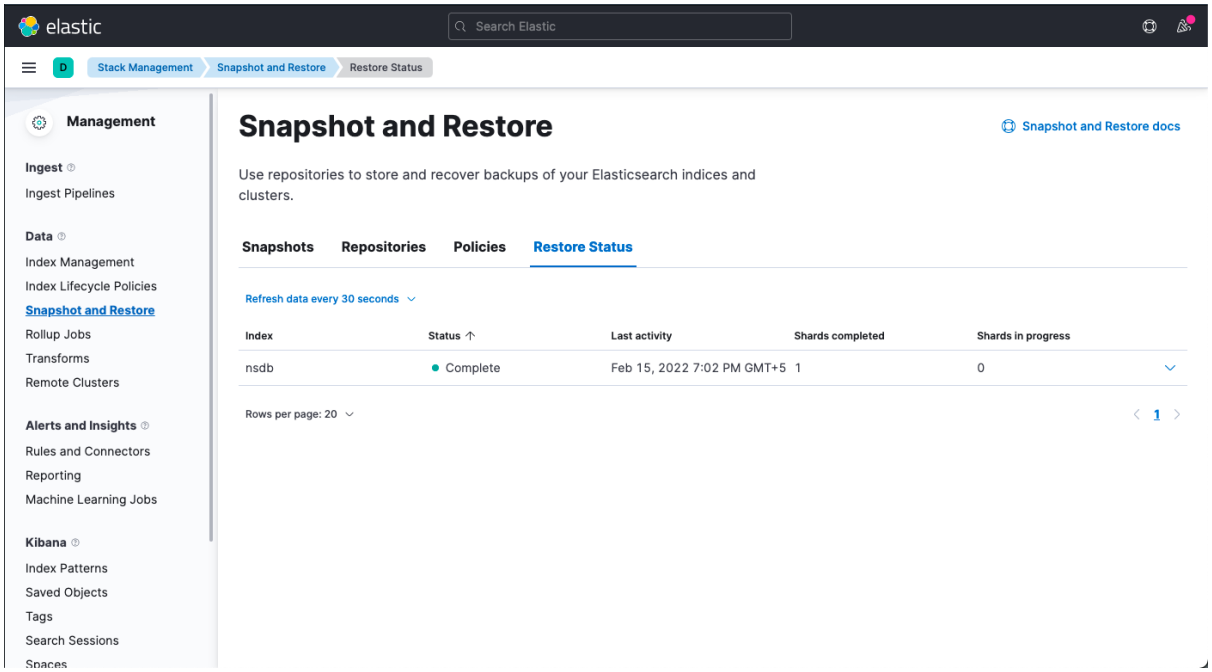
Let's restore this nsdb index again

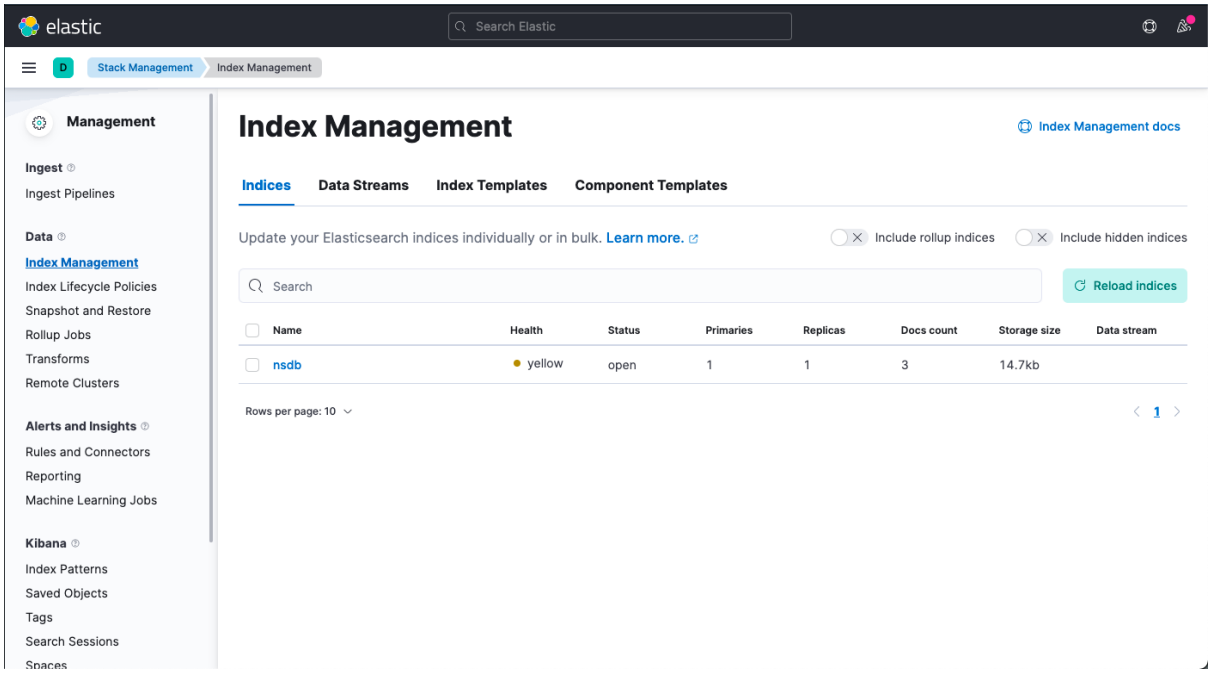






Restored successfully, you can check the status in following images





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

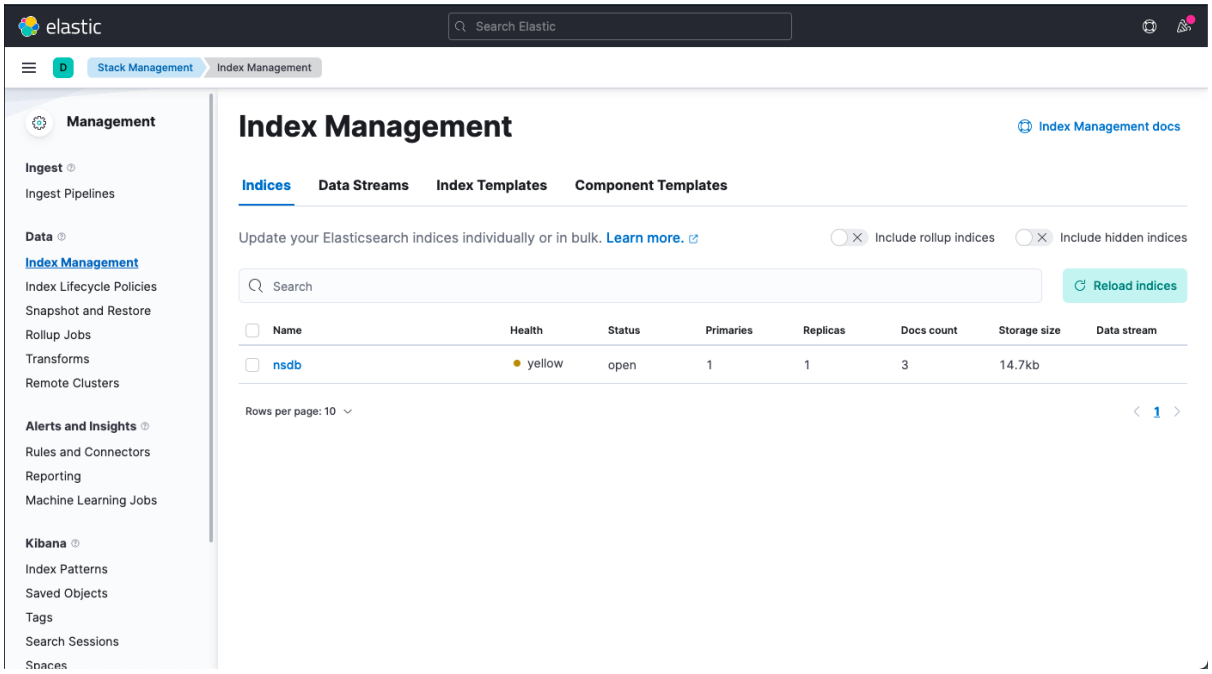
Now we have both the file curator action and curator config, let's store the deleted index.

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