



ElasticSearch



caosao@techmaster

Data Management on elasticsearch

- **Index Lifecycle Management (ILM)**
- **Data streams**
- **Searchable snapshots**

Data Management

- Data management needs different strategy depending on the type of data you are collecting

	Static data	Time series Metrics	Time series Logs
Data grows ...	Slowly, predictably	Fast, predictably	Fast, unpredictably
Updates ...	May happen	Never happen	Never happen
Old data is read...	Frequently	Infrequently	Infrequently

Scaling indices

- Indices scale by adding more shards
 - Increasing the number of shards of an index requires reindexing
- **Solution:** A new index has more shards



Using aliases

- Use **index aliases** to simplify your access to the growing number of indices



One alias to multiple indices

- An alias can point to multiple indices
 - Specify the **write index** using **is_write_index**

```
POST _aliases
```

```
{  
  "actions": [ {  
    "add": {  
      "index": "my_logs-*",  
      "alias": "my_logs"  
    } },  
  {  
    "add": {  
      "index": "my_logs-2021-07",  
      "alias": "my_logs",  
      "is_write_index": true  
    } } ] }
```

Reads will search all
matching indices

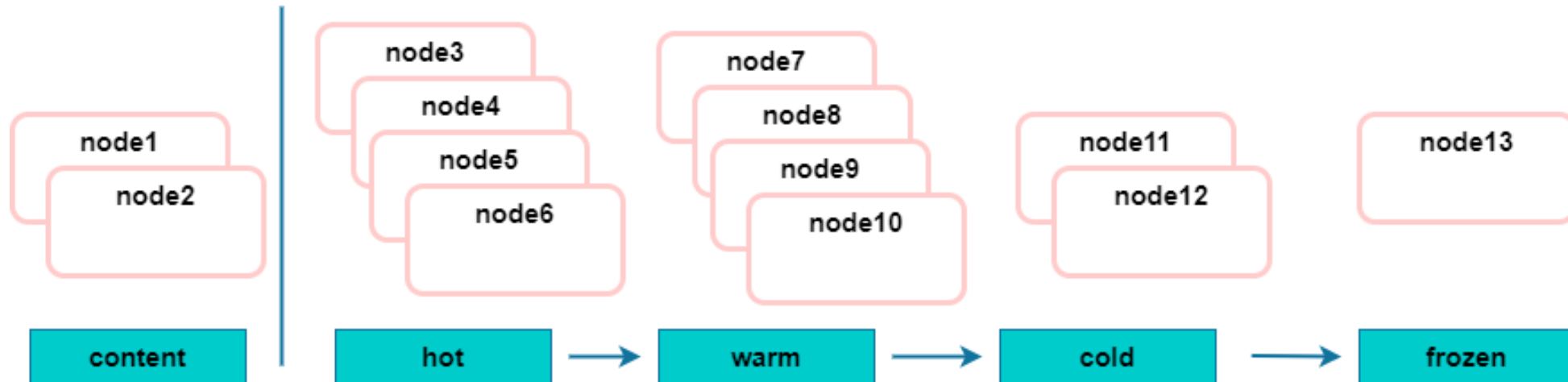
Write requests
will go to this
index

Data grows old

- Data gets older and loses value over time
- Older less frequently accessed data can be moved to less expensive hardware
 - Increases the data density on existing resources
- We can also move it to less expensive hardware while it still has some value
- **Solution:** Data tiers

What is a data tier

- A data tier is a collection of nodes with the same data role
 - That typically share the same hardware profile
- There are five types of data tiers:



Overview of the five data tiers

- The **content tier** is useful for static datasets
- Implementing a **hot → warm → cold → frozen architecture** can be achieved using the following data tiers:
 - **hot tier**: have the fastest storage for writing data and for frequent searching
 - **warm tier**: for read-only data that is searched less often
 - **cold tier**: for data that is searched sparingly
 - **frozen tier**: for data in searchable snapshots

Assigning nodes to a data tier

- Every node is **all** data tiers by default
 - Set using the **node.roles** parametes
- Node roles are handled for you automatically in an Elastic Cloud deployment
 - you can assign node roles manually in elasticsearch.yml:

```
node.roles: ["data_hot", "data_content"]
```

Assigning nodes to a data tier

- Set the node role with the desired configuration parameter
 - a node can be in multiple tiers

Tier Type	Configuration parametes
content	data_content
hot	data_host
warm	data_warm
cold	data_cold
frozen	data_frozen

Configuring an index to prefer a data tier

- Set the data tier preference of an index using the `routing.allocation.include._tier_preference` property
 - `data_content` is the default for all indices
 - you can update the property at any time
 - ILM can manage this setting for you

```
PUT logs-2021-03
{
  "settings": {
    "index.routing.allocation.include._tier_preference" : "data_hot"
  }
}
```

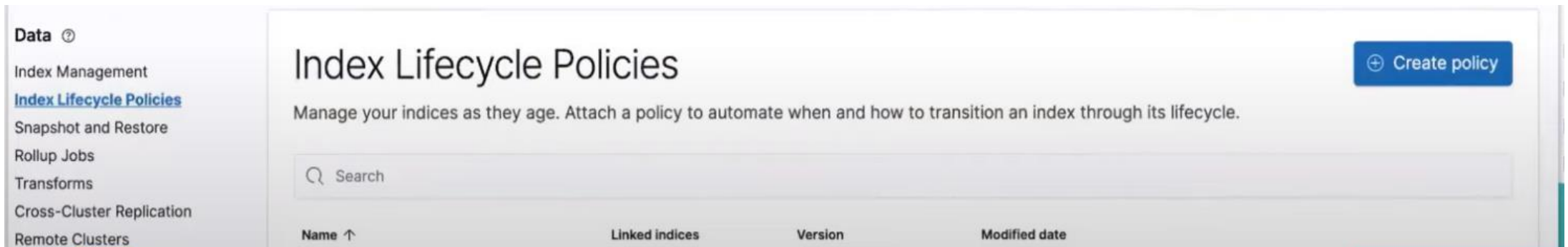
Index Lifecycle Management

- Index Lifecycle Management (ILM) allows you to easily manage your indices according to your specific requirements
- ILM consists of policies that trigger actions, such as:

Action	Description
rollover	create a new index based on age, size, or doc count
shrink	reduce the number of primary shards
force merge	merge the segments of the shards
freeze	saves memory on rarely used indices
delete	permanently remove an index

Lifecycle phases, actions and policies

- The lifecycle is broken up into five **phases**:
 - hot, warm, cold, frozen, and delete
- Each phase has a set of available **actions**:
 - rollover, allocate, freeze, etc.
- Combine the phases with the actions to create a **policy**
 - policies are managed through API or the Kibana UI



ILM policy example

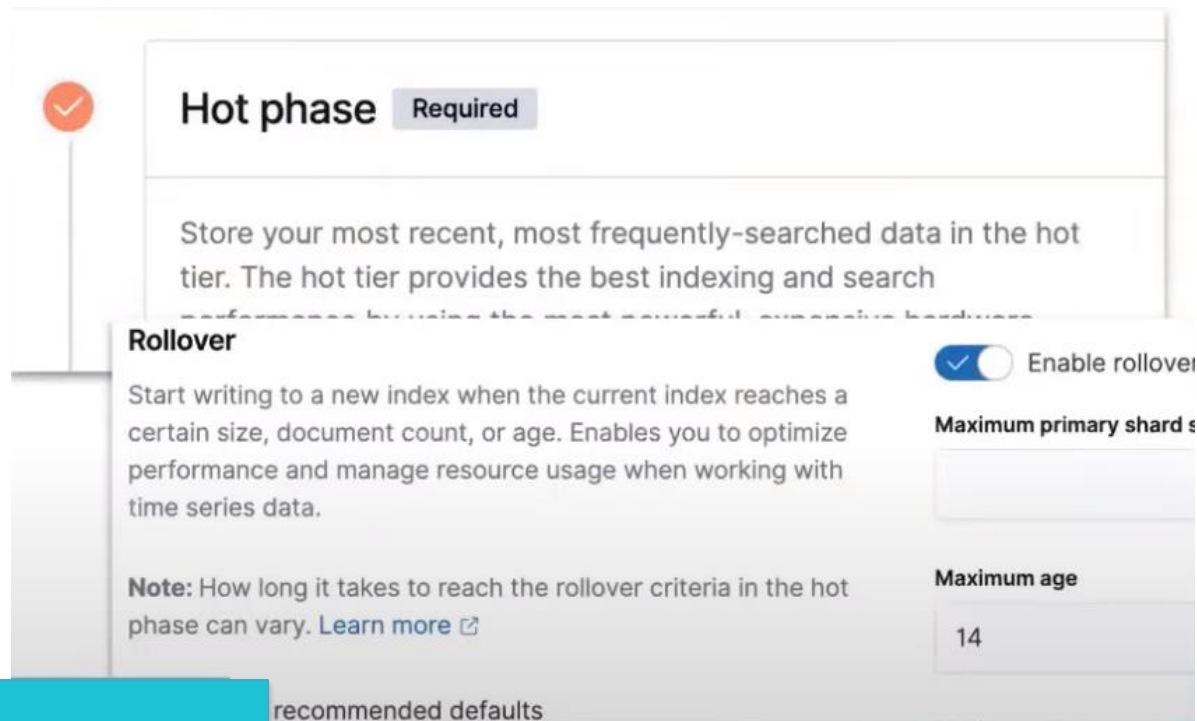
- During the hot phase you might:
 - Roll an alias over to a new index every two weeks
- In the warm phase you might:
 - make the index read-only and move to warm for one week
- In the cold phase you might:
 - freeze the index, decrease the number of replicas, and move to cold for three weeks
- In the delete phase:
 - the only action allowed is to delete the 6-week-old index

Define the hot phase

We want documents in the hot phase for 2 weeks

```
PUT _ilm/policy/my-hwcd-policy
{
  "policy": {
    "phases": {
      "hot": {
        "actions": {
          "rollover": {
            "max_age": "14d"
          }
        },
        "min_age": "0ms"
      }
    }
  }
}
```

Rollover to a new index after
14 days



The screenshot shows the 'Hot phase' configuration in the Elasticsearch UI. It includes a 'Required' label, a description of the hot tier, a 'Rollover' section with a description and a note, and input fields for 'Maximum primary shards' and 'Maximum age' (set to 14). A 'recommended defaults' link is at the bottom.

Hot phase Required

Store your most recent, most frequently-searched data in the hot tier. The hot tier provides the best indexing and search performance by using the most powerful, expensive hardware.

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](#)

☒ Enable rollover

Maximum primary shards

Maximum age

14

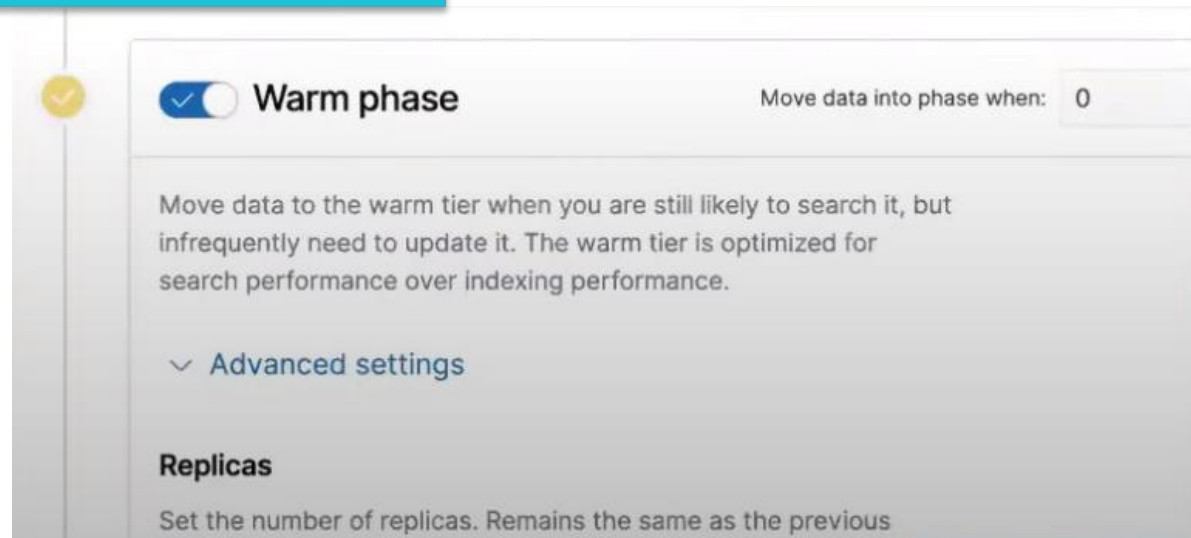
[recommended defaults](#)

Define the warm phase

- We want the old index to move to the warm tier immediately and set the index as read-only:
 - **data age** is calculated **from the time of rollover**

Move the warm tier
immediately after rollover

```
"warm": {  
  "min_age": "0d",  
  "actions": {  
    "readonly": {}  
  }  
},
```



Define the cold phase

After one week of warm, move the index to the cold phase, reduce the number of replicas, and freeze the data

Move a cold 7 days after rollover

```
"cold": {  
  "min_age": "7d",  
  "actions": {  
    "migrate": {  
      "enabled": false  
    },  
    "allocate": {  
      "number_of_replicas": 1  
    },  
    "freeze": {}  
  }  
},
```



Cold phase

Move data into phase when: 7

Move data to the cold tier when you are searching it less often and

Replicas

Set the number of replicas. Remains the same as the previous phase by default.

Number of replicas

1



Set replicas

Freeze

Make the index read-only and minimize its memory footprint.

[Learn more](#)

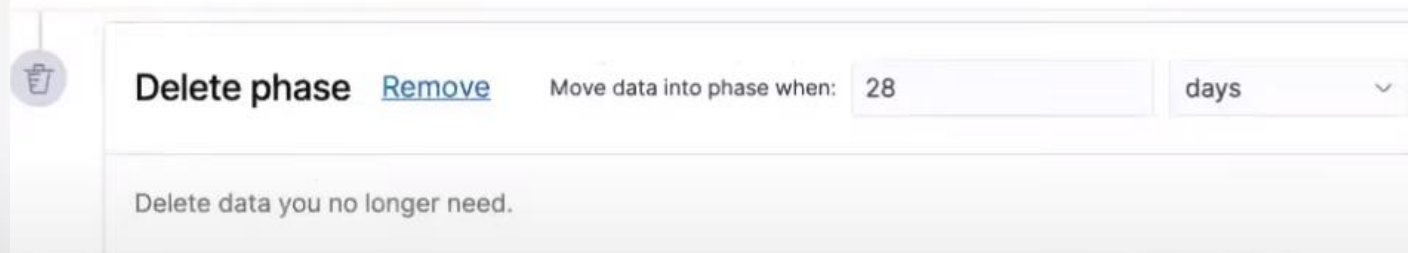


Freeze index

Define the delete phase

- Let's delete the data four weeks **after rollover**.
 - which means the documents lived for 14 days in hot
 - then 7 days in warm
 - then 21 days in cold

```
"delete": {  
  "min_age": "28d",  
  "actions": {  
    "delete": {}  
  }  
}
```



The image shows a snippet of the Elasticsearch UI for configuring a delete phase. It includes a trash icon, the text "Delete phase" followed by a "Remove" link, and a field "Move data into phase when:" with the value "28" and a unit dropdown set to "days". Below this is a light gray box with the text "Delete data you no longer need."

Delete phase [Remove](#) Move data into phase when: 28 days

Delete data you no longer need.

Delete 28 days from
rollover

Applying the policy

- Set two settings in your index:
 - `index.lifecycle.name`
 - `index.lifecycle.rollover_alias`
- The simplest way to do this is with an index template

```
PUT _index_template/my_logs_template
{
  "index_patterns": ["my-logs*"],
  "template": {
    "settings": {
      "index.lifecycle.name": "my_hwcd_policy",
      "index.lifecycle.rollover_alias": "my-logs-alias"
    }
  }
}
```

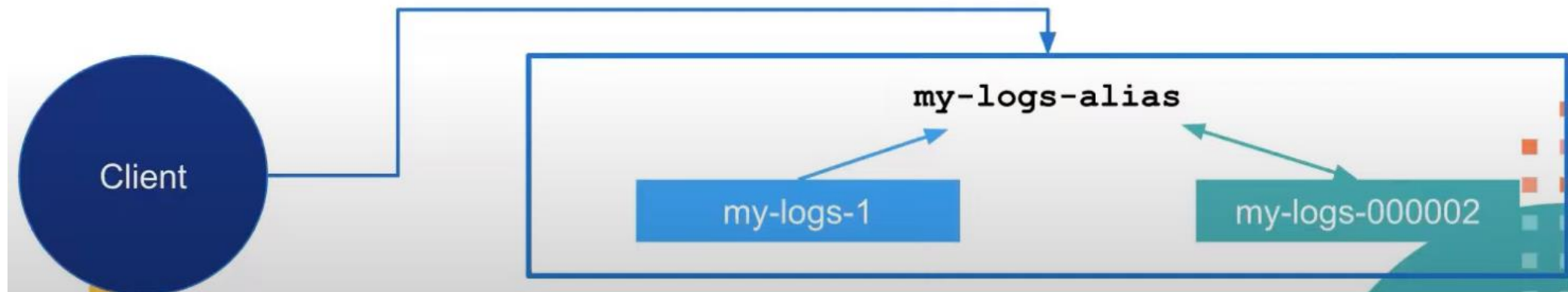
Creating the first index

- Make sure the name ends with a “-” and a number
 - make sure the name matches the template
 - make sure the new index is the write index for the alias

```
PUT my-logs-1
{
  "aliases": {
    "my-logs-alias": {
      "is_write_index": true
    }
  }
}
```

Start indexing documents

- **ILM takes over from here!**
 - all interactions should go through the alias
- When a rollover happens, the number of the index is incremented
 - the new index is set as the write index of the alias
 - all indices are still part of the alias for reads



Troubleshooting lifecycle rollover

- If the index is not green, it will not progress
- The default polling interval for the cluster is 10 minutes
 - can set with **indices.lifecycle.poll_interval**
- Check the server log for errors
- Make sure you have the appropriate data tiers for migration
- **Reminder:** name your first index with a number
- **Reminder:** use a template to apply a policy to new indices
- Get detailed information about ILM status with `GET <alias>/_ilm/explain`

Beats and ILM

- All Beats use ILM policies to manage rollover
- By default, Beats policies:
 - remain in the hot phase forever
 - never delete
 - indices are rolled over after 30 days or 50GB
- The default Beats policies can be edited within the Kibana UI

Data tiers and ILM

- Data tiers can be configured within an ILM policy

☒ Warm phase

Move data into phase when: 30 days

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](#)

Data allocation

Move data to nodes optimized for less-frequent, read-only access.

Data tier options

Use warm nodes (recommended) ▼

Hand-on labs