Elasticsearch Watcher Script Analysis

Generated on: 2025-03-26 19:53:33

## User

Watcher or ILM for implementation of lifecycle policy

## Assistant

When choosing between Watcher and Index Lifecycle Management (ILM) in Elasticsearch for implementing a lifecycle policy, ILM is the right and recommended approach.  
  
Here's why:  
  
| Feature | Index Lifecycle Management (ILM) | Watcher |  
|--------|----------------------------------|---------|  
| Purpose | Built-in for managing index lifecycle | General-purpose alerting and automation |  
| Use Case | Automating index rollover, shrink, delete, etc. | Sending alerts, triggering actions based on conditions |  
| Lifecycle Policy | Directly supports hot-warm-cold-delete phases | Not built for lifecycle; would need custom logic |  
| Ease of Use | Simple configuration via policy APIs or Kibana | Requires writing JSON watches, custom logic |  
| Efficiency | Native integration with index management | More overhead; not optimized for index lifecycle |  
  
Example ILM Use Case:  
You want to:  
- Keep logs in hot nodes for 7 days,  
- Move to warm nodes after 7 days,  
- Delete after 30 days.  
This is perfect for ILM with a simple policy.  
  
When would you use Watcher?  
Use Watcher if you need:  
- Email/Slack alerts when disk usage is high,  
- Alert when index size exceeds a limit,  
- Custom actions beyond what ILM provides.  
  
Conclusion:  
Use ILM for managing the lifecycle of indices (rollover, delete, etc.) — it's purpose-built for it. Use Watcher for alerting and monitoring.

## User

Uploaded images of index\_watcher.json

## Assistant

Your script is an Elasticsearch Watcher JSON configuration named index\_watcher.json. Let me break it down and explain what it does and why it might have been created in your environment:  
  
What the Script Does  
  
1. Trigger:  
 - Runs daily at midnight.  
  
2. Input (Search Query):  
 - Searches all indices ("\_all").  
 - Filters indices:  
 - That are older than 30 days ("creation\_date": { "lt": "now-30d/d" }).  
 - Excludes indices matching:  
 - \*data-reconciliation\*, \*aai\*, \*snap-outputs\*, \*dnb21\*, \*resolver-data\*, \*resolver-match\*, \*qssmetrics\*  
 - Also excludes indices that have aliases.  
  
3. Condition:  
 - Checks if any indices match the query (hits.total.value > 0).  
  
4. Actions:  
 - Deletes the matched indices.  
 - Limits deletion to 10 indices per run.  
  
Why This Script Was Created in Your Env  
- Likely implemented to automate deletion of old indices that are:  
 - Older than 30 days,  
 - Not critical,  
 - Not tied to active aliases.  
- It helps manage disk usage and keep Elasticsearch clean.  
  
Issues in the Script  
1. Incorrect Wildcard Format (Minor):  
 - Wildcards like "\*resolver-data\*" are fine, but could be costly for large clusters.  
  
2. Lack of Index Sorting:  
 - No sorting mechanism — if the search returns >10 results, it randomly deletes the first 10.  
  
3. Hardcoded "\_all":  
 - "indices": ["\_all"] is not recommended. It can be expensive and is deprecated in newer Elasticsearch versions.  
  
4. No Logging or Alerting:  
 - The watcher deletes data, but does not log actions or notify anyone.  
  
5. Aliases Check Could Be Insufficient:  
 - "exists": { "field": "aliases" } may not be valid, as aliases are metadata.  
  
Recommendation  
- Use Index Lifecycle Management (ILM) if possible.  
- ILM handles rollover and deletion safely, with minimal risk and better performance.  
  
Would you like me to help convert this logic into an ILM policy? Or add alerting to this Watcher?