NEW CLUETR SET-UP

**Prepare yamls**

We need to create new namespace eg. Spc-dev & spc-uat-one is deployed in sosh cluster only.  
to create namespace, we need to choose **create namespace** pipeline

A screenshot of a computer

Description automatically generated  
as it is asking for Specify Environment and Specify infrastructure.  
Before we are creating namespace, need to create prerequisites such as creating environment, infrastructure, and override.

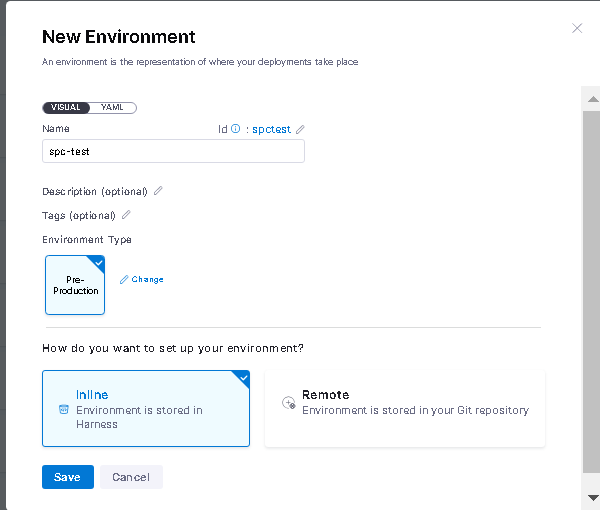
Under deployment, we can see all these options to create

A screenshot of a computer

Description automatically generated

First, we need to create environment, A blue and white box with white text

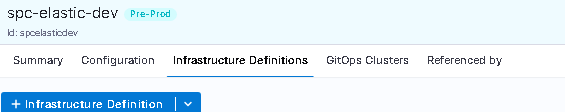
Description automatically generated



**then need to create new infrastructure :**

we need to create two infrastructure: one is by selecting Kubernetes and another is helm chart.

<+service.name> this is parameter need to use.



A screenshot of a computer

Description automatically generated

this is for native helm, for Kubernetes choose k8

Then need to create override like below: values in override should be same as folder name of qa/dev in bitbucket like spc-elastic-dev/spc-elastic-qa.

A screenshot of a computer

Description automatically generated

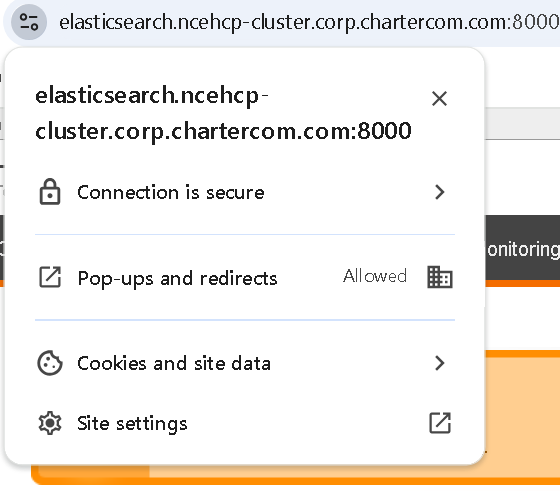
After this store all certificate in Centrify

We need deploy 3 certificates

Folder structure should be like: we need to create one folder like **s3-secrets** within this folder we need to store 2 secrets

**one** **is custom-s3-trustore**: this secrete should inside s3 folder. this we can download from hitachi .

Click on connection is secure then certification valid



Then go to details click on export, cacerts certificate get downloaded.

A screenshot of a computer

Description automatically generated

**Second one is** **s3-ncehcp-credential**: this also should be inside s3 folder.this we can downloade from existing one go to any folder open **s3-ncehcp-credential secrets,**click on action --?retrive. it det doenload, so we can uploaded same downloaded file to our new file in S3 for **s3-ncehcp-credential.**

it has password & access key credential

**Third one is elastic-certificate-ca:** this also we can downlaode from existing one and this secrete should be is outside of s3 folder

While adding new secrets : click on add permission, this will ask for name like we have specify **elastic-certificate-ca, then with same page click on permission**

A screenshot of a computer

Description automatically generated

Then upload respective file

A screenshot of a computer

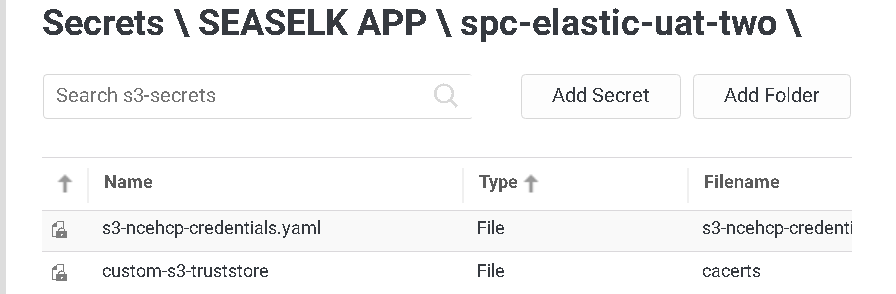
Description automatically generated

After this clik on save, in next UI search for [svc-ara-caas@corp.charter.com](mailto:svc-ara-caas@corp.charter.com) it will comes in drop down select and update it.

After this click on permission make sure to grant permissions like: GRANT, VIEW AND RETRIEVE for all three files.

A screenshot of a computer

Description automatically generated



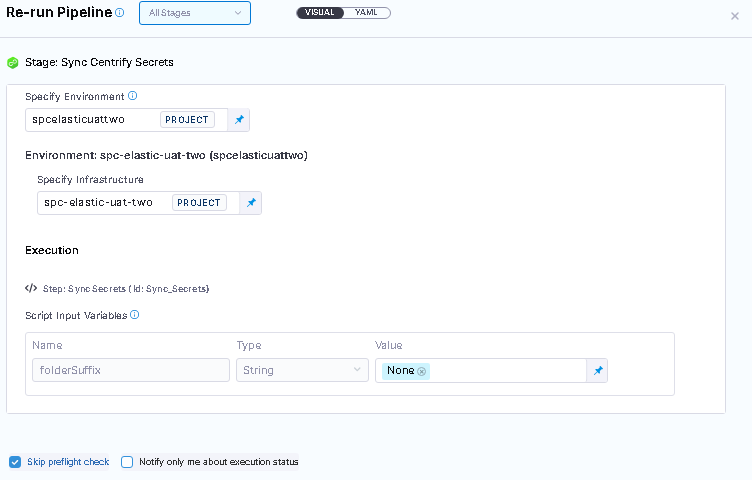
Now certificate all we stoed in centrify. Next we have to run this using Harness.

After this we need deploy all these Below **pipeline Sync centrify secrets**

A screenshot of a computer

Description automatically generated

First we need reun this pipeline first by selecting none in value and then by selecting s3-secrets like below



A screenshot of a computer

Description automatically generated

After all these set up we are ready to deploy yaml’s

If we are deploying to existing cluster no need to deploy eck operator yaml. Only multiple\_eck\_operator ,elastic & kiban is enough.

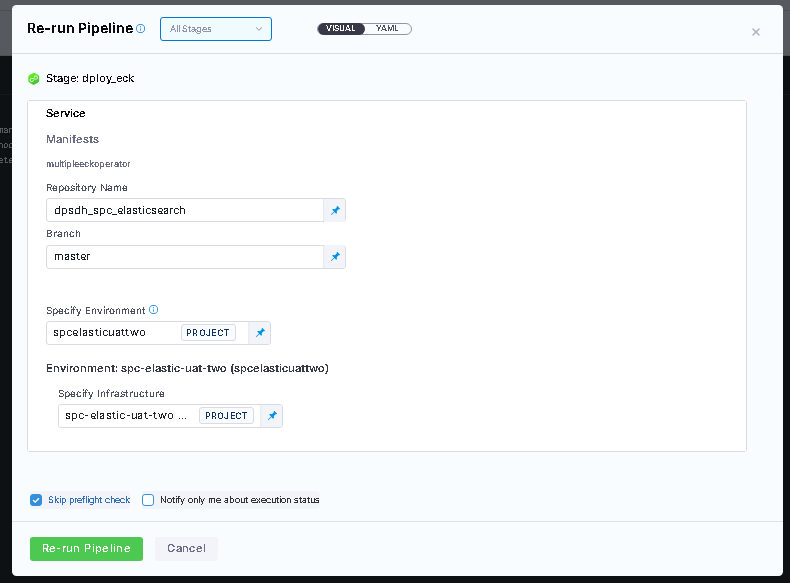
If we are setting up for new cluster need to deploy eck operator, multiple eck operator and elastic, kibana yaml.

Here to deploy eck operator chose **deploy yaml pipeline** in Harness but select service as eck\_operatot. Select Repository Name, Branch and Specify Environment as per requirements.

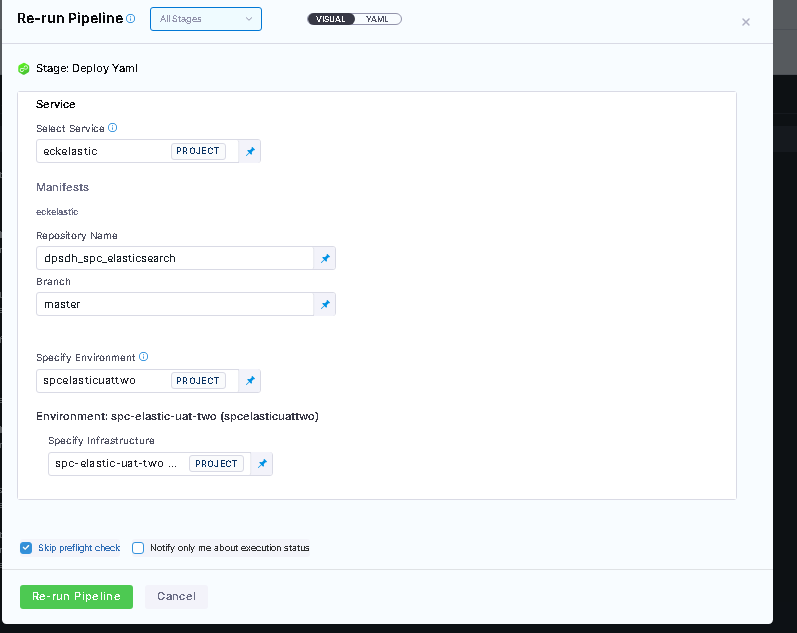
A screenshot of a computer

Description automatically generated

To deploy multiple\_eck\_operator : we need to choose **multiple\_eck\_operator** pipelineand Select Repository Name, Branch and Specify Environment and run it.



Next we need to deploy yamls for ellasticseaarch, kibana: to ths also select **Deploy yaml** Pipeline but select service as eck\_elastic. Select Repository Name, Branch and Specify Environment as per requirements. Once we select eck\_elastic as service it will only deploy elastic & kibana yaml.



After all yaml deployed verify urr working once working start work on set up

1. Setup self monitoring in SPC-DEV cluster.

2. Create rules and connectors. Snapshot setup.

3. Create logstash pipeline for the alerts.

4. Deploy hearbeat changes in SOSH-UAT\_ONE environment to monitor SPC URLs.

5. Create Monitoring dashboards as per standard template.

**Setup self monitoring in SPC-DEV cluster.**

PUT \_cluster/settings  
{  
"persistent": {  
"xpack.monitoring.collection.enabled": true,  
"xpack.monitoring.elasticsearch.collection.enabled": true  
}  
}

**This is a dynamic setting**, meaning **no restart is required**.

If we mention only this "xpack.monitoring.collection.enabled": true, we need to restart cluster, instead of that what we can do is, enable "xpack.monitoring.elasticsearch.collection.enabled": true. Then no need to restart cluster it is directly fetch monitoring data from elasticsearch.

"xpack.monitoring.collection.enabled": true, It does **not configure where to store monitoring data** (i.e., it doesn’t automatically set up self-monitoring).

If xpack.monitoring.elasticsearch.collection.enabled is **not** set, it won’t automatically collect and send data to a monitoring cluster.

xpack.monitoring.collection.enabled: true → Enables monitoring collection at the **cluster level**.

xpack.monitoring.elasticsearch.collection.enabled: true → Ensures monitoring data is **collected and stored in Elasticsearch** (instead of needing an external monitoring cluster like Metricbeat).

Ref link: <https://www.elastic.co/guide/en/elasticsearch/reference/current/monitoring-settings.html>

**Rules & connectors**

After set self-monitoring,We need to create rules for this instead of creating each and every rules manually , we can go to stack monitoring click on alerts & rules then click on create default rule.

A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated

First create connectors

A screenshot of a computer

AI-generated content may be incorrect.

After this we need to go to each and evrey rule & select connector and put query like below

{

  "date": "{{date}}",

  "Error Name Header": "{{rule.name}}",

  "subject": "{{context.clusterName}} - {{rule.name}}",

  "Alert Cluster Name": "{{context.clusterName}}",

  "Additional Links": "{{context.action}}",

  "message": "{{context.internalShortMessage}}",

  "Error details": "{{context.internalFullMessage}}"

}

.We need to specify that storing all rule in cluster-health-log index. For this first create index, define mapping like below. Name of index is **cluster-health-logs**

put cluster-health-logs  
GET cluster-health-logs/\_search  
PUT cluster-health-logs/\_mapping  
{  
"properties": {  
"Additional Links": {  
"type": "text",  
"fields": {  
"keyword": {  
"type": "keyword",  
"ignore\_above": 256  
}  
}  
},  
"Alert Cluster Name": {  
"type": "text",  
"fields": {  
"keyword": {  
"type": "keyword",  
"ignore\_above": 256  
}  
}  
},  
"Error Name Header": {  
"type": "text",  
"fields": {  
"keyword": {  
"type": "keyword",  
"ignore\_above": 256  
}  
}  
},  
"Error details": {  
"type": "text",  
"fields": {  
"keyword": {  
"type": "keyword",  
"ignore\_above": 256  
}  
}  
},  
"date": {  
"type": "date"  
},  
"message": {  
"type": "text",  
"fields": {  
"keyword": {  
"type": "keyword",  
"ignore\_above": 256  
}  
}  
},  
"subject": {  
"type": "text",  
"fields": {  
"keyword": {  
"type": "keyword",  
"ignore\_above": 256  
}  
}  
}  
}  
}  
after creation of rules & connector we can test rules are storing in index or not.  
eg: set disk usage or cpu usage to 0% in rule save it.

Run this GET cluster-health-logs/\_search  
will see that rules are alert are storing in index.

We need to prepare logstash pipeline, so **refer** [**dpsdh\_sosh\_logstash**](https://bitbucket.corp.chartercom.com/projects/DP-ELK/repos/dpsdh_sosh_logstash/browse?at=refs%2Fheads%2Ffeature%2Fspc-logstash) repo. In spc-logtsash branch : Cass\_spc\_qa like this after preparing this yaml add this in main yaml, pushed to bitbucket.

  
To test alerts storing or not before trigger, use below api in dev tol

POST cluster-health-logs/\_doc/3

{

"date": "2025-02-11T15:10:20.991Z", //mention current time when we are running this .

"Error Name Header": "CPU Usage",

"subject": "spc-elastic-uat-one - CPU Usage",

"Alert Cluster Name": "spc-elastic-uat-one",

"Additional Links": "[View node](https://spc-uat1-kibana.corp.chartercom.com:443/app/monitoring#/elasticsearch/nodes/6spd8hPeTVqzQWmEEeJ0xw?\_g=(cluster\_uuid:HwlVrKW1TMasu2SfxjYFjQ))",

"message": "CPU usage alert is firing for node spc-elastic-uat-one-es-master-1 in cluster: spc-elastic-uat-one. Verify CPU level of node.",

"Error details": "CPU usage alert is firing for node spc-elastic-uat-one-es-master-1 in cluster: spc-elastic-uat-one. [View node](https://spc-uat1-kibana.corp.chartercom.com:443/app/monitoring#/elasticsearch/nodes/6spd8hPeTVqzQWmEEeJ0xw?\_g=(cluster\_uuid:HwlVrKW1TMasu2SfxjYFjQ))"

}

GET cluster-health-logs/\_search

{

"query": {

"range": {

"date": {

"gte": "now-15m"

}

}

}

}//it will show above alert we ingested in Put api, which mention it is correct  
then delete all stored alert in index, before deploy logstash yaml

POST cluster-health-logs/\_delete\_by\_query

{

"query": {

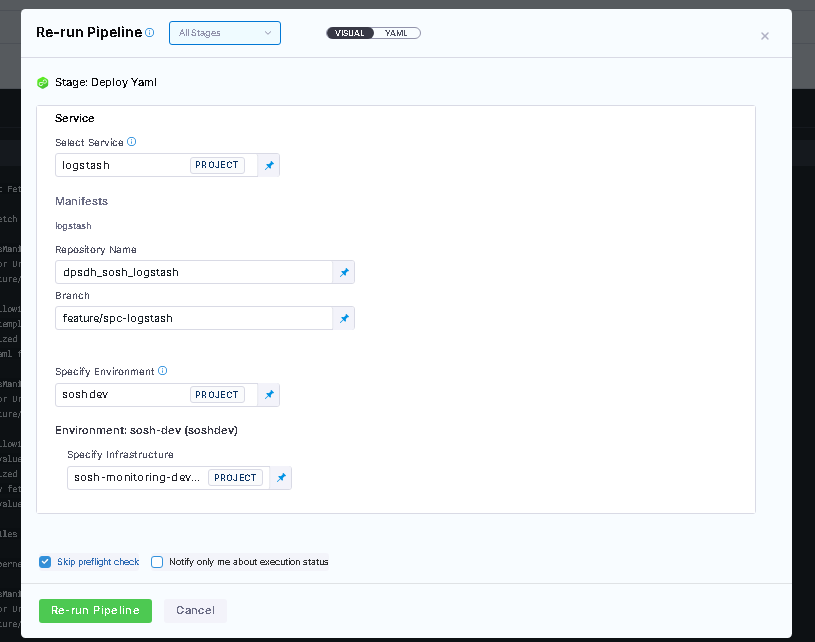
"match\_all": {}

}

}

To trigger this alert to our mail & Webex, deploy prepared logstash pipeline

Refer below figure. Once alerts start trigget to Webex or mail, go back rules where we set disk usga ot cpu usage to 0. Again set it to85% only.



1. **Heartbeat set up   
   repo:** [**dpsdh\_sosh\_heartbeat**](https://bitbucket.corp.chartercom.com/projects/DP-ELK/repos/dpsdh_sosh_heartbeat/browse) **in sosh-dev**

**Source branch: feature/spc-heartbeat**

[**dpsdh\_sosh\_heartbeat**](https://bitbucket.corp.chartercom.com/projects/DP-ELK/repos/dpsdh_sosh_heartbeat/browse?at=feature%2Fspc-heartbeat)**/[sosh-dev](https://bitbucket.corp.chartercom.com/projects/DP-ELK/repos/dpsdh_sosh_heartbeat/browse/sosh-dev?at=feature%2Fspc-heartbeat)/**[**templates**](https://bitbucket.corp.chartercom.com/projects/DP-ELK/repos/dpsdh_sosh_heartbeat/browse/sosh-dev/templates?at=feature%2Fspc-heartbeat)**/[monitors.d](https://bitbucket.corp.chartercom.com/projects/DP-ELK/repos/dpsdh_sosh_heartbeat/browse/sosh-dev/templates/monitors.d?at=feature%2Fspc-heartbeat)**

**this spc-qa heartbeat watcher in sosh-dev, in watcher search for http alerting then look for spc non production.**

A screenshot of a computer

AI-generated content may be incorrect.

**To delete heartbeat choose delete k8 resorce**

A screenshot of a computer

AI-generated content may be incorrect.

**After uninstall check in backend weather it is deleted or not then redeploy it while redeploying choose re-run pipeline**

**Roles and User**

[dpsdh\_llm\_elasticsearch](https://bitbucket.corp.chartercom.com/projects/DP-ELK/repos/dpsdh_llm_elasticsearch/browse) repo: for roles and user

create it in dev toll only by run this api

instead of creating manually

PUT \_security/role/spc\_account\_idx\_role\_rw

{

"cluster": ["monitor"],

"indices": [

{

"names": ["account-v1"],

"privileges": ["read","write"]

}

]

}

PUT \_security/role/spc\_idx\_all\_role\_superuser

{

"cluster": ["monitor"],

"indices": [

{

"names": ["account-v1"],

"privileges": ["all"]

}

]

}

POST \_security/user/spc\_admin

{

"password": "L1Lw+61BFTd{",

"roles": ["superuser"],

"full\_name": "Data Platform Team",

"email": "DL-DPSDH-Elastic@charter.com",

"enabled": true

}

POST \_security/user/spc\_app\_esi

{

"password": "8k<X6UX9&JEb",

"roles": ["spc\_account\_idx\_role\_rw"],

"full\_name": "User for SPC ES Ingest Application",

"email": "DL-SPC-Dev-Team@charter.com",

"enabled": true

}

POST \_security/user/spc\_app\_spces

{

"password": "VzH90)W0^mma",

"roles": ["spc\_account\_idx\_role\_ro"],

"full\_name": "User for SPC Application to consume from ES",

"email": "DL-SPC-Dev-Team@charter.com",

"enabled": true

}

POST \_security/user/spc\_app\_superuser

{

"password": "5Nx4v>O~I80y",

"roles": ["monitoring\_user","spc\_idx\_all\_role\_superuser"],

"full\_name": "SuperUser for SPC Application with access to all indices",

"email": "DL-SPC-Dev-Team@charter.com",

"enabled": true

}

POST \_security/user/spc\_sdit

{

"password": "rIVN0Pm7[K8Y",

"roles": ["spc\_account\_idx\_role\_ro"],

"full\_name": "User EAA team admin activities",

"email": "DLEnterpriseApplicationAdministrators@charter.com",

"enabled": true

}

[dpsdh\_keepass](https://bitbucket.corp.chartercom.com/projects/DP-ELK/repos/dpsdh_keepass/browse) repo to set up keypass : SoshElastic

**Ingest Pipeline:**

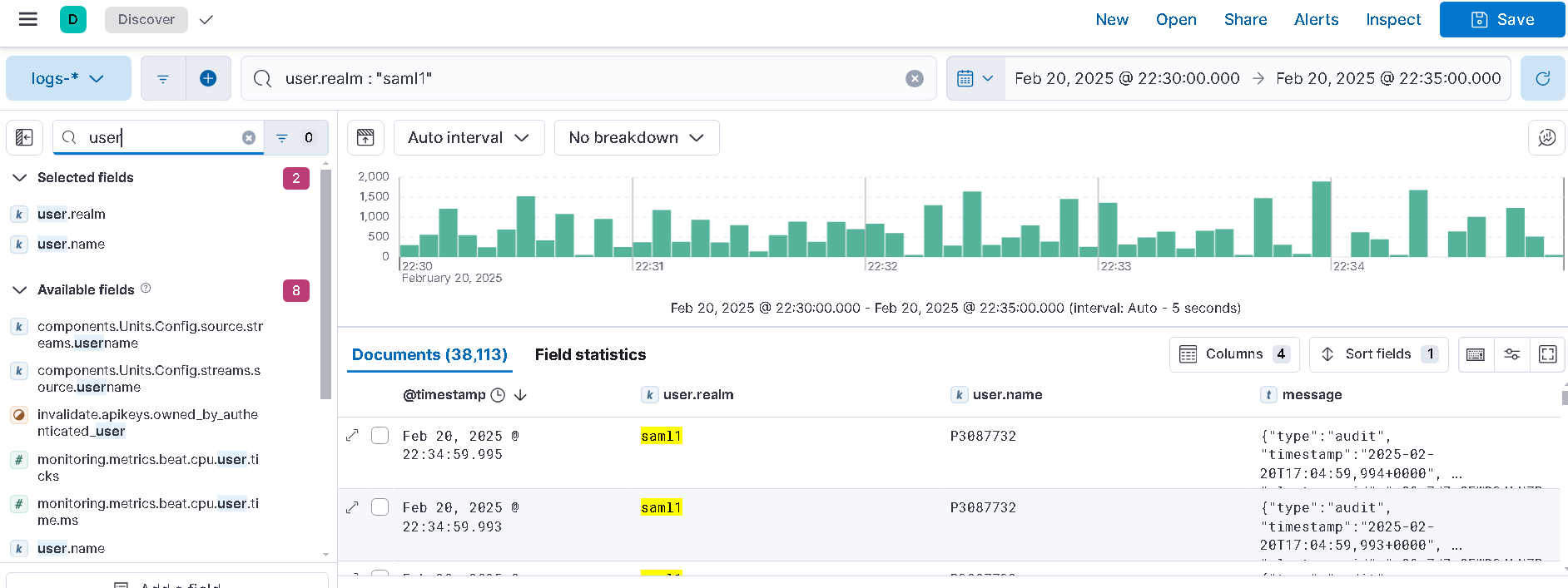
* **Requires manual application** while indexing each document (?pipeline=<pipeline\_name>).
* **Used for selective processing** when not all documents need transformation.
* Example: Adding a default **status** field only when indexing specific documents.

**Final Pipeline:**

* **Automatically applied to all documents** in an index without needing ?pipeline=.
* **Defined at the index level**, so it ensures uniform processing for every document.
* It is set at the index level using the final\_pipeline setting.
* Best for **mandatory transformations** like adding timestamps or enforcing field consistency.

A screenshot of a computer

AI-generated content may be incorrect.



Monitoring work on Saturday & Sunday

Change time interval from 15 to 1min.

Then monitor.name is UP change this to DOWN

**SOSH-UAT : SPC space FOR INGEST PIPELINE**

1. [Stack Management](https://sosh-uat-kibana.corp.chartercom.com/s/sps/app/management/)
2. [Ingest Pipelines](https://sosh-uat-kibana.corp.chartercom.com/s/sps/app/management/ingest/ingest_pipelines/)

**AUDIT LOGGING**

[dpsdh\_sosh\_elasticsearch](https://bitbucket.corp.chartercom.com/projects/DP-ELK/repos/dpsdh_sosh_elasticsearch/browse?at=refs%2Fheads%2Faudit-logging)/[sosh-dev](https://bitbucket.corp.chartercom.com/projects/DP-ELK/repos/dpsdh_sosh_elasticsearch/browse/sosh-dev?at=refs%2Fheads%2Faudit-logging)/[templates](https://bitbucket.corp.chartercom.com/projects/DP-ELK/repos/dpsdh_sosh_elasticsearch/browse/sosh-dev/templates?at=refs%2Fheads%2Faudit-logging)/**eck\_es.yaml branch is** audit logging

* 1. **Metricbeat** – **Collects system and service metrics**

Gathers **system-level metrics** (CPU, memory, disk usage, network traffic, etc.)

Monitors services like **Redis, MySQL, Kubernetes, Docker, and Elasticsearch**

Sends metrics to **Elasticsearch or Logstash** for storage and analysis

Use Case:

Monitoring **server health**, **application performance**, and **infrastructure metrics**

**2. Filebeat – Collects and ships log files**

* Reads logs from various sources (e.g., system logs, application logs, web server logs)
* Supports structured and unstructured log data
* Can **forward logs** to Elasticsearch, Logstash, or Kafka

**Use Case:**

* Centralized **log collection and forwarding**
* Used in logging pipelines for applications and services

**3. Heartbeat – Monitors service uptime**

* Periodically pings **HTTP, TCP, or ICMP endpoints**
* Tracks service **availability, response time, and status**
* Useful for **monitoring APIs, websites, and network services**

**Use Case:**

* **Uptime monitoring** for critical services and infrastructure
* Alerts when services go **down or become unreachable**

In Heartbeat, the **records** refer to the **monitoring events** that it generates every time it checks the status of a service.

Each record contains:

1. **Timestamp** – When the check was performed.
2. **Monitor name** – The name of the service being monitored.
3. **URL or Host** – The target being checked (e.g., a website, API, or server).
4. **Status** – Whether the service is **up (available)** or **down (unreachable)**.
5. **Response time** – How long it took for the service to respond.

So, if Heartbeat runs **every 5 minutes**, it will create **one record every 5 minutes**. If you check for 15 minutes, you’ll have **3 records** (one at 0 min, one at 5 min, and one at 10 min).

**How Logs Flow from Splunk to Elasticsearch in Your Project**

1. **Logs are stored in Splunk** – Your project already has logs in Splunk.
2. **Filebeat Reads Logs from Splunk**
   * Filebeat is configured to **pull logs from Splunk** using the **Splunk input module**.
   * It fetches logs in **real-time** or at **scheduled intervals**.
3. **Filebeat Processes and Forwards Logs**
   * Filebeat **parses and structures** the logs (if needed).
   * It then **forwards logs to Elasticsearch** for indexing.
4. **Elasticsearch Stores and Indexes the Logs**
   * Logs are stored in **Elasticsearch indices** based on their timestamp, log type, or source.
   * You can search, analyze, and visualize these logs in Kibana.

**Summary**

👉 **Filebeat acts as a "bridge" between Splunk and Elasticsearch.**  
👉 It **reads logs from Splunk**, processes them, and then **forwards them to Elasticsearch** for storage and analysis.

**CLUSTERS**

**A node is a single running instance of Elasticsearch.**

**A cluster is a group of nodes working together.**

**Each node can have one or more roles (Master, Data, Ingest, etc.).**

**Nodes communicate with each other to store and process data.**

**Types of Nodes**

| **Node Type** | **Role** | **Function** |
| --- | --- | --- |
| **Master Node** | master | Manages cluster state, node coordination. |
| **Data Node** | data | Stores actual data and handles queries. |
| **Ingest Node** | ingest | Preprocesses data before indexing. |
| **Coordinating Node** | coordinating | Routes search requests to data nodes. |

**2. What Are Shards?**

Shards are the **smallest unit of data storage** in Elasticsearch.  
When you create an **index**, Elasticsearch automatically divides it into multiple shards.

🔹 **Why Use Shards?**

* Helps distribute **data** across multiple nodes.
* Increases **performance** by parallelizing searches.
* Provides **fault tolerance** using replicas.

**Types of Shards**

| **Shard Type** | **Description** |
| --- | --- |
| **Primary Shard** | Stores original data. |
| **Replica Shard** | Copy of primary shard (used for failover and load balancing). |
| **GET \_cat/shards?v**  **Shows which node holds each shard.**  **Helps debug unassigned shards.** |  |

Sematic search bit bucket repo: [llm\_keyahead\_api](https://bitbucket.corp.chartercom.com/projects/DP-ELK/repos/llm_keyahead_api/browse)

Branch: feature/pvc-mount

Branch actions

[llm\_keyahead\_api](https://bitbucket.corp.chartercom.com/projects/DP-ELK/repos/llm_keyahead_api/browse?at=feature%2Fpvc-mount)/[src](https://bitbucket.corp.chartercom.com/projects/DP-ELK/repos/llm_keyahead_api/browse/src?at=feature%2Fpvc-mount)/[main](https://bitbucket.corp.chartercom.com/projects/DP-ELK/repos/llm_keyahead_api/browse/src/main?at=feature%2Fpvc-mount)/[java](https://bitbucket.corp.chartercom.com/projects/DP-ELK/repos/llm_keyahead_api/browse/src/main/java?at=feature%2Fpvc-mount)/[com](https://bitbucket.corp.chartercom.com/projects/DP-ELK/repos/llm_keyahead_api/browse/src/main/java/com?at=feature%2Fpvc-mount)/[charter](https://bitbucket.corp.chartercom.com/projects/DP-ELK/repos/llm_keyahead_api/browse/src/main/java/com/charter?at=feature%2Fpvc-mount)/[icy](https://bitbucket.corp.chartercom.com/projects/DP-ELK/repos/llm_keyahead_api/browse/src/main/java/com/charter/icy?at=feature%2Fpvc-mount)/[addresssearch](https://bitbucket.corp.chartercom.com/projects/DP-ELK/repos/llm_keyahead_api/browse/src/main/java/com/charter/icy/addresssearch?at=feature%2Fpvc-mount)/service /**ChatBotController.java**

**search tempalte for fulltext and semanticsearch**

GET \_scripts/fulltext-search-template  
GET \_scripts/pano-search-spectrum-template  
GET search-spectrum/\_search

POST \_scripts/fulltext-search-template-V1  
{  
  "script": {  
    "lang": "mustache",  
    "source": """  
    {  
      "\_source": ["title", "meta\_description", "url"],  
      "query": {  
        "bool": {  
          "should": [  
            {  
              "match": {  
                "title": {  
                  "query": "{{query\_string}}"  
                }  
              }  
            },  
            {  
              "match": {  
                "meta\_description": {  
                  "query": "{{query\_string}}"  
                }  
              }  
            }  
          ]  
        }  
      }  
    }  
    """,  
    "options": {  
      "content\_type": "application/json;charset=utf-8"  
    }  
  }  
}

POST \_render/template  
{  
  "id": "fulltext-search-template-V1",  
  "params": {  
    "query\_string": "iPhone 14 Plus"  
  }  
}

GET pano\_uat\_v1\_semanticsearch/\_count  
{

  "\_source": ["title","meta\_description","url"],

  "query" :{

    "bool": {

      "should": [

{

"text\_expansion":{

"title\_embedding" : {

"model\_id": ".elser\_model\_2\_linux-x86\_64",

 "model\_text" : "iphone",

"boost" : 1

}

}

}

]

}}}

GET search-spectrum/\_count  
{

  "\_source": ["title", "meta\_description", "url"],

  "query": {

"bool": {

"should": [

       {

          "match": {

            "title": "iphone"

          }

}

      ]

     }

  }

}

Uninstall fleet agents for spc-qa  
A screenshot of a computer

AI-generated content may be incorrect.

To delete any yaml means uninstall need to select [Delete K8s Resource pipeline and select respective env and infrastucture then in valyue we need to add like kind/metadata. As this kind and metadata we don’t know, we can check the yaml file. Here as i am uninstalling fleet agents for spc-qa, i went to particular repo: dpsdh\_fleet\_deployment and select dleet\_elastic\_agents and spc\_elastic\_qa the in templates/elastic\_agents.yml this yaml i find kinf and meta data like](https://charter.harness.io/ng/account/YdcrtZUpT0Wb462nro8tJw/home/orgs/Data_Management/projects/SEASELK/pipelines/Delete_K8s_Resource/pipeline-studio/?storeType=INLINE)

Ex: kind: DaemonSet

metadata:

name: spc-elastic-agent-qa

so values will be : DaemonSet/spc-elastic-agent-qa the run pipeline check in backend to confirm weather it is uninstall or not.

Login spc-qa cluster

./kubectl get pods -n spc-monitoring-qa : it will shows like No resources found in spc-monitoring-qa namespace. Meand it confirm that fleet agents are uninstalled.  
then delete namespace spc-monitoring-qa

Fleet agents installation

A screenshot of a computer

AI-generated content may be incorrect.

Delete Namespace

A screenshot of a computer

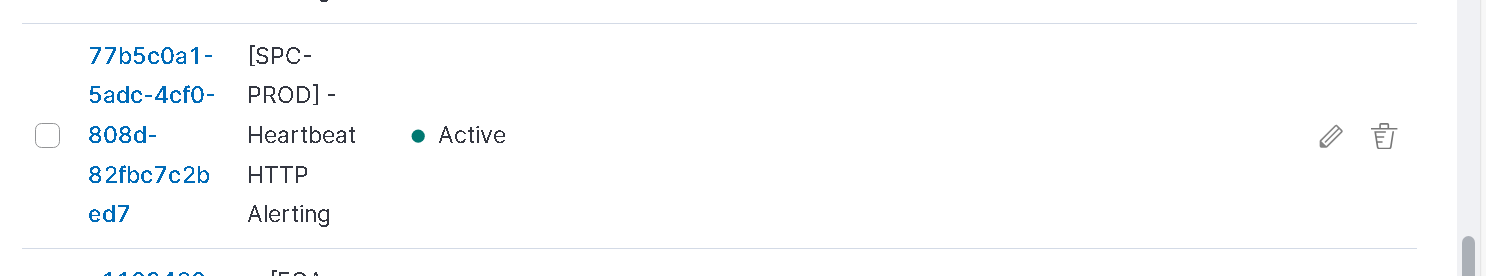
AI-generated content may be incorrect.

<https://grokdebugger.com/> : to check greeddata

weekends monitoring : trigger test alert for production

in Sosh-dev, search for http alerting and look for any production heatbeat like below

SPC-PROD HEARTBEAT Http alerting



starting we have time interval, click on edit and change 15m to 1m

{

  "trigger": {

    "schedule": {

      "interval": "15m"

A screen shot of a computer program

AI-generated content may be incorrect.

A computer screen shot of a computer

AI-generated content may be incorrect.

    And look for monitor status: UP should be DOWN  
must\_not": [

                {

                  "match": {

                    "monitor.status": "up"

                  }

                }

              ],

Once we get alert in Webex, revert back all changes to up and to 15m.

<https://chalk.charter.com/pages/viewpage.action?spaceKey=ITSDDM&title=ITDP+SDH+ELK+Spectrum+Core+%28SPC%29+Elasticsearch+Information#ITDPSDHELKSpectrumCore(SPC)ElasticsearchInformation-UserDetails> : this is a list of VM   
user and password

**User Details**

**PROD and DR user:**

dpsdh\_elastic\_team

LlzO5T3F0kEzl9rr3RieVrvGd6A1o9sv

**NON-PROD (DEV/QA/UAT1/UAT2) user:**

dpsdh\_elastic\_team

g9rMgNubWwLxjyzCHtZnyMujppDQ5I3B

base64 -----🡪to retrieve password

**How to close work Iteam:**

[**https://charter.cherwellondemand.com/CherwellClient/Access?\_=51150f55#0**](https://charter.cherwellondemand.com/CherwellClient/Access?_=51150f55#0)

**Type Wi ticket number:**

Click on **set as in Progress, Below status**

Then click on **close** where need to select in dropdown as **completed, then Ok.**

Then verify it once under Tasks

Check with my name it should marked as completed and no assignment

**Alerts in Webex**

When we receive cpu usage alert got o link if not recovered ot anything or like continuously receiving alert, then check dashboard, name of dashboard is FOA Modified Date Time Dashboard. In first visual check with changing time interval, will able to see document count is increasing then reply to alert like   
Hello team,  
Alert was triggered due to ongoing data ingestion and has recovered automatically.

**Snapshot restore to new cluster**

When we build new cluster like seaselk-nce-uat, then need to move all data from sosh-0uat to seaselek-nce-uat. At that time first need to check in sosh-uat that, name of snampshot and it registerd with bucket,

We have to use the seam s3 bucket in new cluster asfter rigester with bucket in new cluster, to restore all data, run below api.  
here check with latest snapshot because it may have more indices and shards.

Old cluster scr

A screenshot of a phone

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

Here above scr apr 14th snapshot has 448 indices and 531 shards  
but when I check latest snapshot is 15th apr with 1 more indices 449 and 532, so we need to restore with latest one so we could have all data.

POST /\_snapshot/my\_repository/snapshot\_2/\_restore?wait\_for\_completion=true  
{  
"indices": "index\_1,index\_2",  
"ignore\_unavailable": true,  
"include\_global\_state": false,  
"rename\_pattern": "index\_(.+)",  
"rename\_replacement": "restored\_index\_$1",  
"include\_aliases": false  
}

**Data streem roolover**

Api: POST /sps\_application\_logs-8.17.1/\_rollover

A screenshot of a computer program

AI-generated content may be incorrect.

Rollover to new index:

GET /\_cat/indices/.ds-sps\_application\_logs-8.17.1\*?v --check this api to confirm rollover

A screenshot of a computer

AI-generated content may be incorrect.

**logs-kubernetes.container\_logs\_sps\* :this index is from fleet**

**sps\_application\_logs-8.17.1 : this is from filebeat**

Indices with higher priority are recovered **first**, making them available earlier.

**ES rally set up for performance testing**

**F**or this one Es rally yaml is prepared by subhashree,  
need to check virtual service available in which cluster, in cdp poc cluster VS isnot avavilble so we set up in sosh-dev cluster.

Log into backend to sosh-dev. And in sosh-monitoring-dev namespace wr are setting up this

Run

./kubectl get ns

./kubectl get pods -n sosh-monitoring-dev

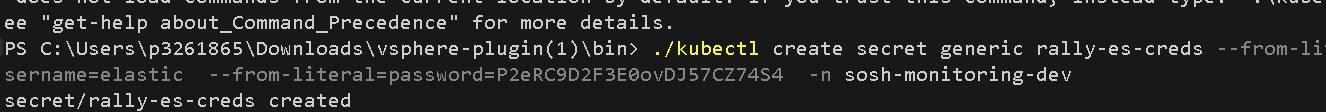
./kubectl get virtualservice -n sosh-monitoring-dev -- this is because in yaml need to specify the name vs, here take elasticsearch vs.

./kubectl get virtualservice -n sps-elastic-uat

In yaml, we hardcoded username and password. So need to deploy to userame and password. To that, run below comd

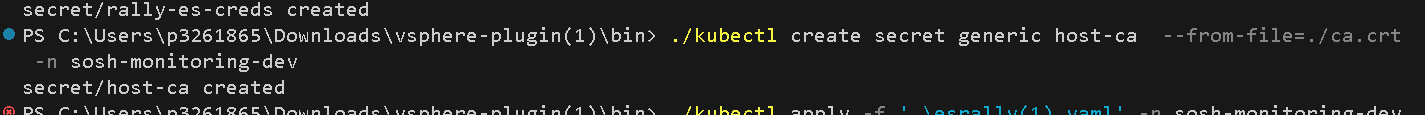
./kubectl create secret generic rally-es-creds --from-literal=username=elastic --from-literal=password=P2eRC9D2F3E0ovDJ57CZ74S4 -n sosh-monitoring-dev

This will show as created.



Then need to add CA certificate means basic charter certificate also need to add to this namespace. Make sure that CA certicate in that VS shpere path, run below cmd, to steployore certificatesd

./kubectl create secret generic host-ca --from-file=./ca.crt -n sosh-monitoring-dev



Then we need to add yaml file path to this Vspgere folder only, then make sure to mount this certificate in volume and volume mounts.

Then to deploy yaml use below cmd:

./kubectl apply -f '.\esrally(1).yaml' -n sosh-monitoring-dev

Then check weather it is deployed or not using below cmd

./kubectl get pods -n sosh-monitoring-dev

Then check logs to make sure we getting expected results, run below cmd  
./kubectl logs esrally-benchmark -n sosh-monitoring-dev

./kubectl apply -f '.\esrally(1).yaml' -n dpsdh-elastic

./kubectl apply -f '.\esrally(3).yaml' -n dpsdh-elastic

./kubectl get pods -n dpsdh-elastic

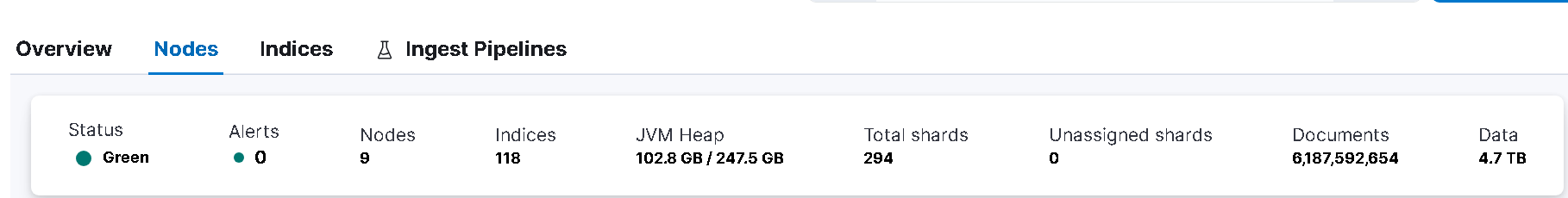
./kubectl logs esrally-benchmark -n dpsdh-elk

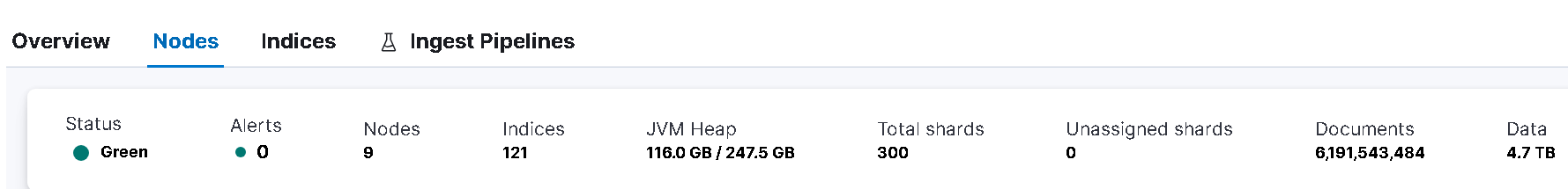
./kbectl delete pod esrally-benchmark -n dpsdh-elk

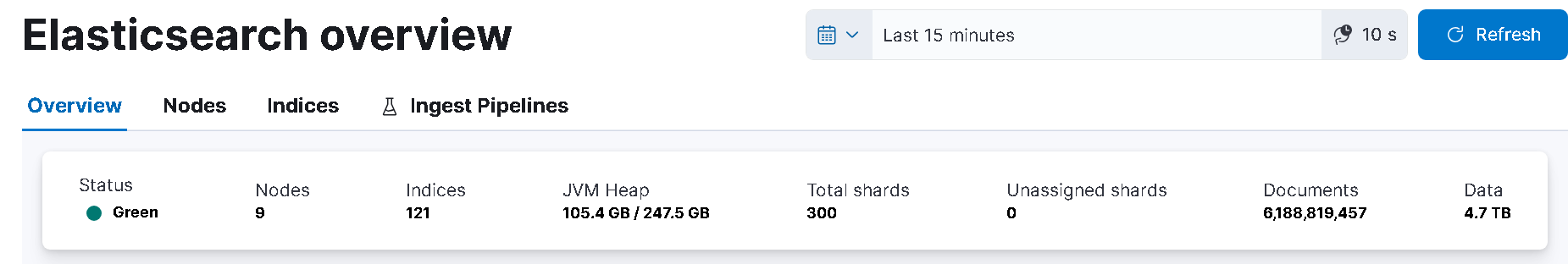
./kubectl apply -f '.\esrally(3).yaml'

./kubectl apply -f '.\esrally(1).yaml' -n dpsdh-elastic

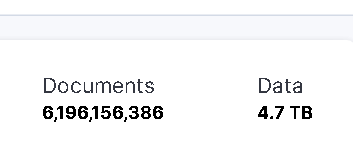
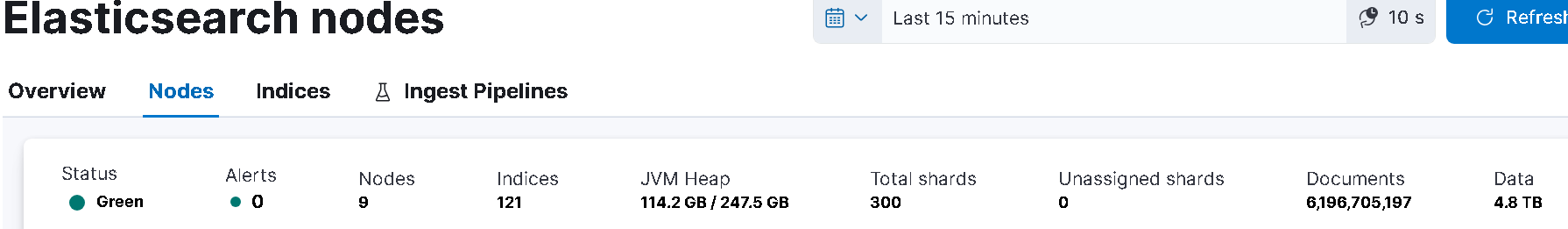
Seaselk

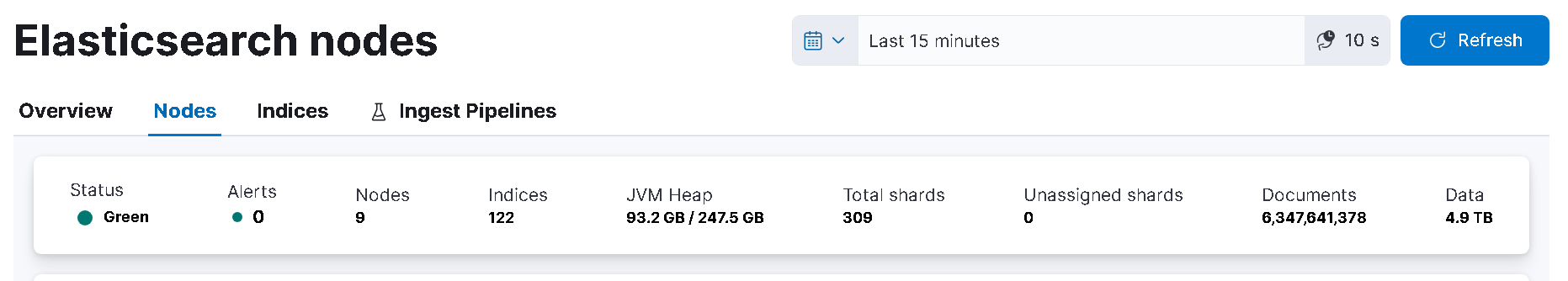






After 20 minutes





[INFO] Downloading track data (252.9 MB total size) [100.0%]

[INFO] Decompressing track data from [/rally/.rally/benchmarks/data/geonames/documents-2.json.bz2] to [/rally/.rally/benchmarks/data/geonames/documents-2.json] (resulting size: [3.30] GB) ... [OK]

[INFO] Preparing file offset table for [/rally/.rally/benchmarks/data/geonames/documents-2.json] ... [OK]

[WARNING] merges\_total\_time is 613905471 ms indicating that the cluster is not in a defined clean state. Recorded index time metrics may be misleading.

[WARNING] merges\_total\_throttled\_time is 497954957 ms indicating that the cluster is not in a defined clean state. Recorded index time metrics may be misleading.

[WARNING] indexing\_total\_time is 179148183 ms indicating that the cluster is not in a defined clean state. Recorded index time metrics may be misleading.

[WARNING] refresh\_total\_time is 35539194 ms indicating that the cluster is not in a defined clean state. Recorded index time metrics may be misleading.

[WARNING] flush\_total\_time is 10954718 ms indicating that the cluster is not in a defined clean state. Recorded index time metrics may be misleading.

Running delete-index [100% done]

Running create-index [100% done]

Running check-cluster-health [100% done]

Running index-append [100% done]

Running refresh-after-index [100% done]

Running force-merge [100% done]

Running refresh-after-force-merge [100% done]

Running wait-until-merges-finish [100% done]

Running index-stats [100% done]

PS C:\Users\p3261865\Downloads\vsphere-plugin(1)\bin>