Contents

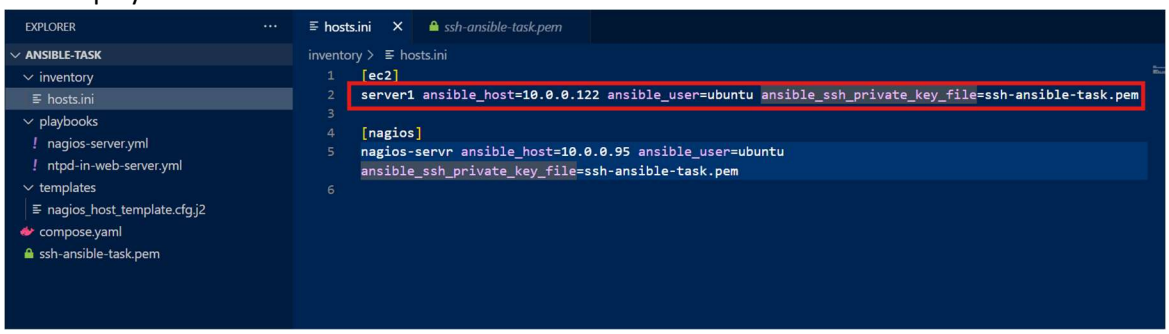
Configuration Management

Which ansible command can display all ansible_ configuration for a host?

'ansible -i <inventory-file> -m <server-name-for-which-we-want-to-check-ansible-configuation>'

The following command will show the ansible-configurations for the indicated server declared in hosts.ini file '***ansible -i ./inventory/hosts.ini -m setup server1*** '

This displays all the facts of the host



To have a reduced number of outputs we can only filter with ansible_related facts

ansible -i ./inventory/hosts.ini -a 'filter=ansible_*' -m setup server1

Output-of-the-command: Uploaded in github **server1-ansible-host-configurations.json** file

Logrotate-cronjob, ntpd configuration & nagios monitoring

3 EC2 instances has been created to complete this task.

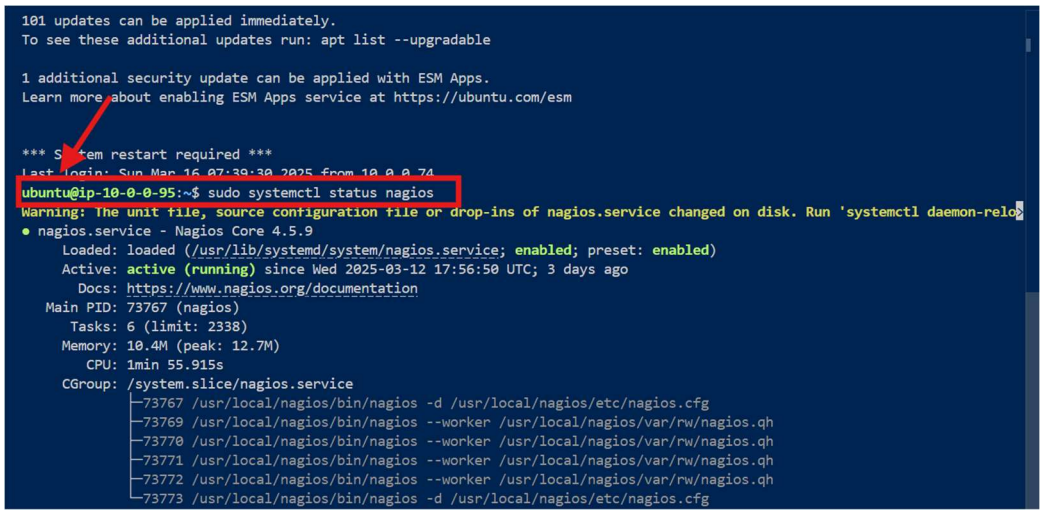
1. 1 EC2 machine to run ansible playbook.
    a. Ansible is installed
    b. If Github-Actions/Gitlab-CI integrated, this EC2 machine will be able to run the playbook through automation
2. 1 nagios server
    a. Security group rule
        i. All ports are open for selected ip's (who will monitor the nagios server)
    b. Public IP assigned
3. 1 web-server
    a. No public IP assigned
    b. SSH port is open for Ansible-host EC2
    c. All ports are open for nagios server

Nagios Server setup:

The machine is **ubuntu 24.04** based.

Based on this set of instructions

Nagios Service status:



Nagios Configuration Changes :

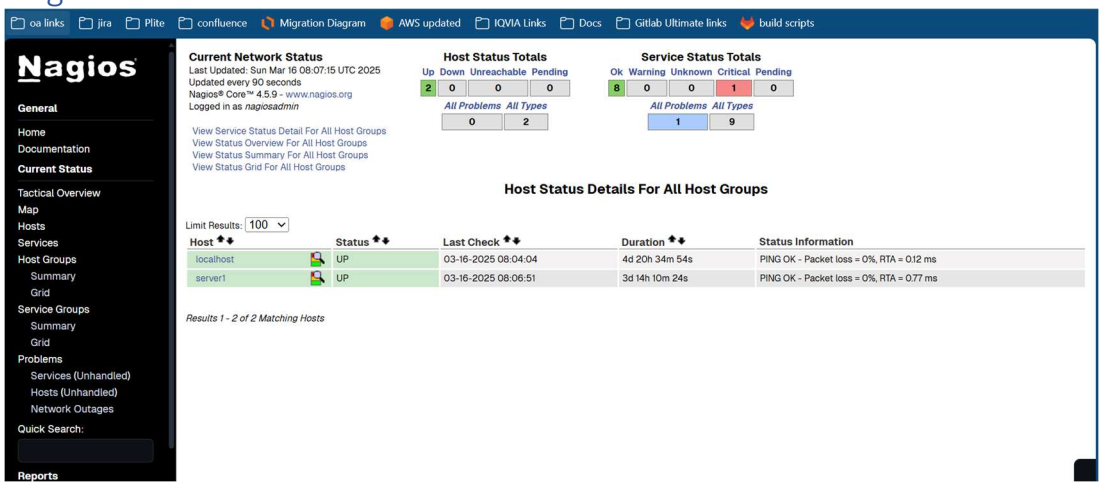
1. **Host directory:** Updated **/usr/local/nagios/etc/nagios.cfg** file to allow /usr/local/nagios/etc/hosts as valid config directory.
   cfg_dir=/usr/local/nagios/etc/hosts
2. All hosts will have a separate configuration file under /usr/local/nagios/etc/hosts which will be **generated by ansible.**
   a.

```
root@ip-10-0-0-95:/usr/local/nagios/etc# tree
.
├── cgi.cfg
├── hosts
│   └── server1.cfg
├── htpasswd.users
├── nagios.cfg
├── objects
│   ├── commands.cfg
│   ├── contacts.cfg
│   ├── localhost.cfg
│   ├── printer.cfg
│   ├── switch.cfg
│   ├── templates.cfg
│   ├── timeperiods.cfg
│   └── windows.cfg
└── resource.cfg
```

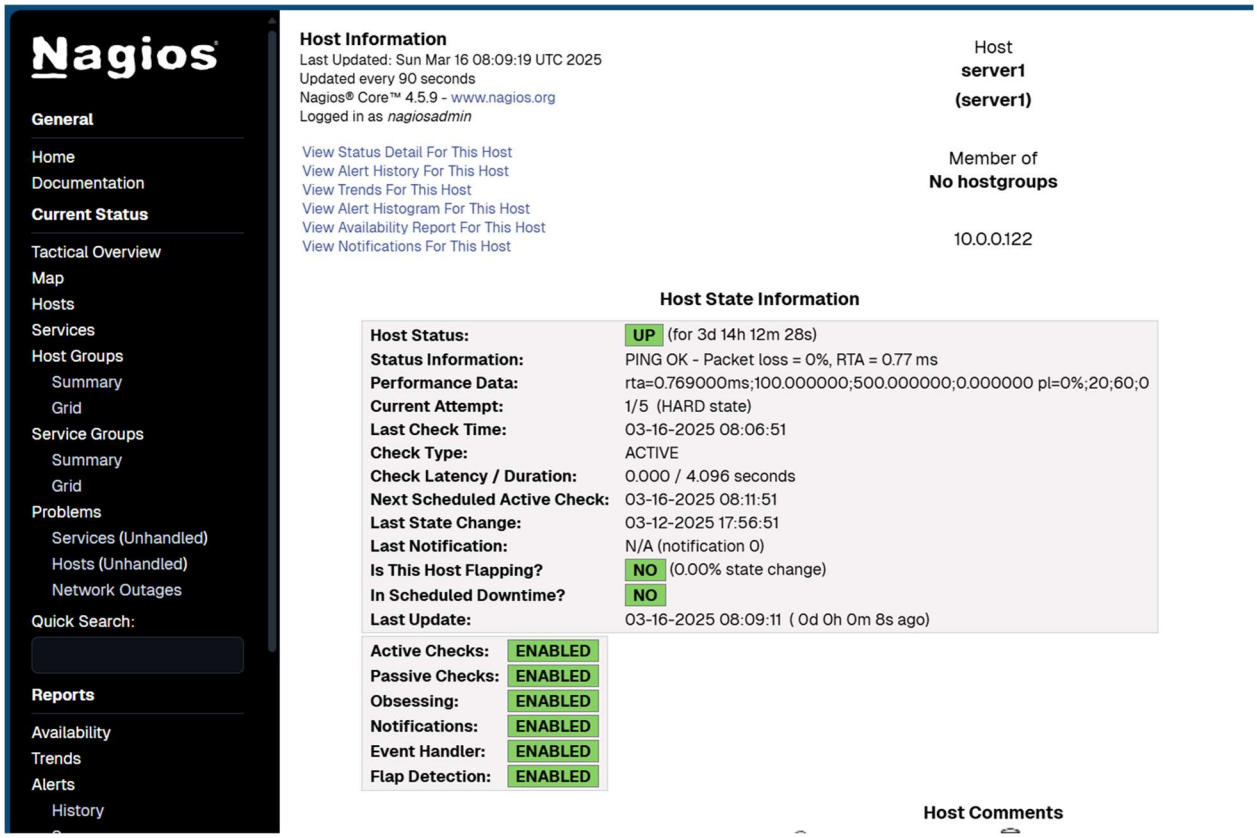
3. **Adding Commands**: Updated objects/commands.cfg file to add check-ping and check_ntp command

```
define command {

    command_name    notify-service-by-email
    command_line    /usr/bin/printf "%b" "***** Nagios *****\n\nNotification Type: $NOTIFICATIONTYPE$\n\nServ
SC$\nHost: $HOSTALIAS$\nAddress: $HOSTADDRESS$\nState: $SERVICESTATE$\n\nDate/Time: $LONGDATETIME$\n\nAdditio
ERVICEOUTPUT$\n" | /bin/mail -s "** $NOTIFICATIONTYPE$ Service Alert: $HOSTALIAS$/$SERVICEDESC$ is $SERVICEST
CTEMAIL$
}


define command {
    command_name    check_ntp
    command_line    $USER1$/check_ntp_time -H $HOSTADDRESS$ -w 5 -c 10
}


define command {
    command_name    check-ping
    command_line    $USER1$/check_ping -H $HOSTADDRESS$ -w 100.0,20% -c 500.0,60% -p 5
}
```

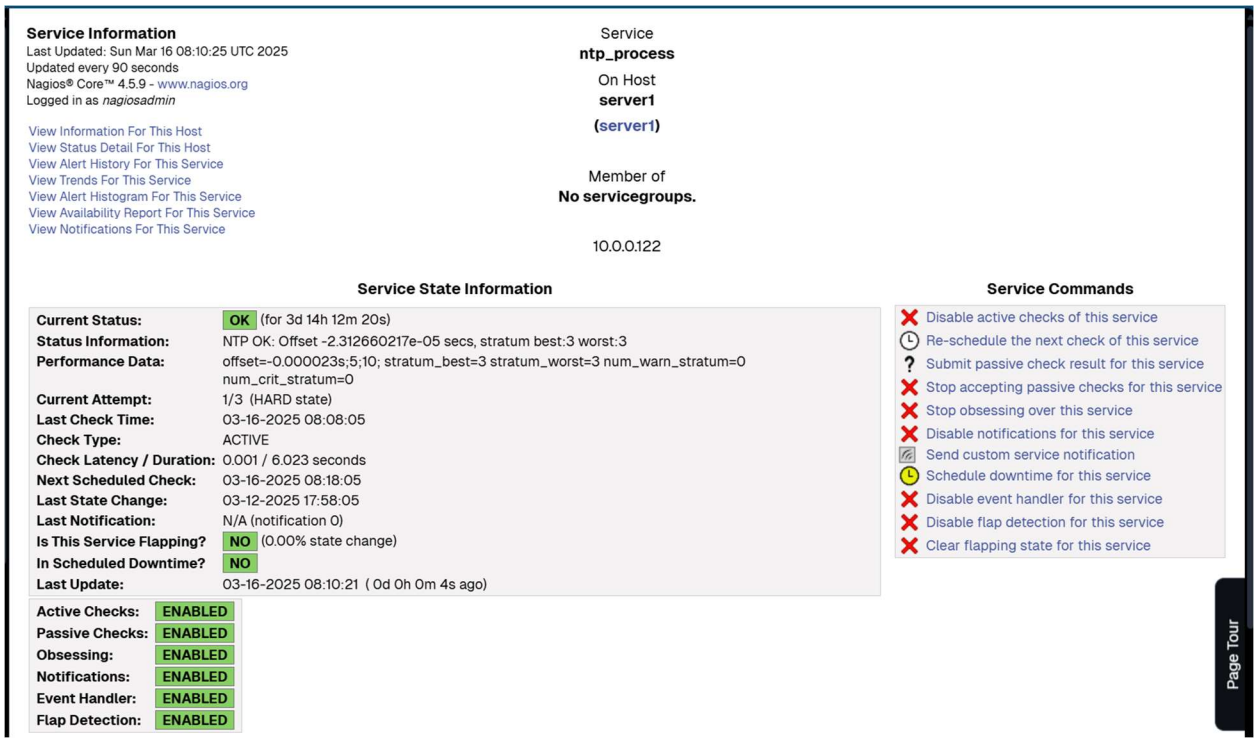
Nagios UI



Server1 Information



Server1 NTP service status



Service Information
Last Updated: Sun Mar 16 08:10:25 UTC 2025
Updated every 90 seconds
Nagios® Core™ 4.5.9 - www.nagios.org
Logged in as *nagiosadmin*

View Information For This Host
View Status Detail For This Host
View Alert History For This Service
View Trends For This Service
View Alert Histogram For This Service
View Availability Report For This Service
View Notifications For This Service

Service
ntp_process
On Host
server1
(server1)

Member of
No servicegroups.

10.0.0.122

Service State Information

Current Status:	OK (for 3d 14h 12m 20s)
Status Information:	NTP OK: Offset -2.312660217e-05 secs, stratum best:3 worst:3
Performance Data:	offset=-0.000023s;5;10; stratum_best=3 stratum_worst=3 num_warn_stratum=0 num_crit_stratum=0
Current Attempt:	1/3 (HARD state)
Last Check Time:	03-16-2025 08:08:05
Check Type:	ACTIVE
Check Latency / Duration:	0.001 / 6.023 seconds
Next Scheduled Check:	03-16-2025 08:18:05
Last State Change:	03-12-2025 17:58:05
Last Notification:	N/A (notification 0)
Is This Service Flapping?	NO (0.00% state change)
In Scheduled Downtime?	NO
Last Update:	03-16-2025 08:10:21 (0d 0h 0m 4s ago)

Active Checks:	ENABLED
Passive Checks:	ENABLED
Obsessing:	ENABLED
Notifications:	ENABLED
Event Handler:	ENABLED
Flap Detection:	ENABLED

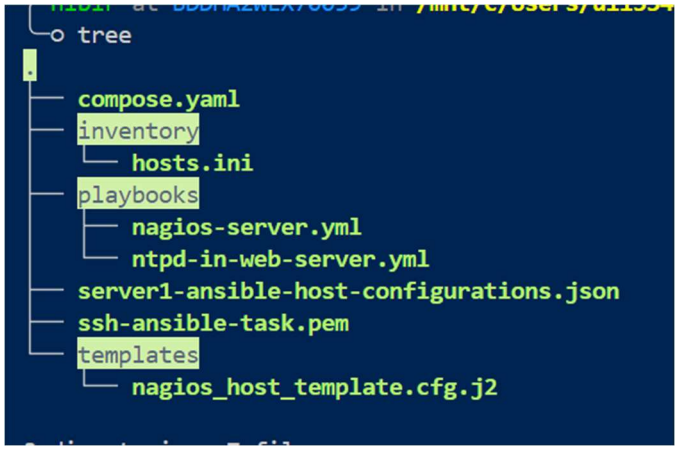
Service Commands

- Disable active checks of this service
- Re-schedule the next check of this service
- Submit passive check result for this service
- Stop accepting passive checks for this service
- Stop obsessing over this service
- Disable notifications for this service
- Send custom service notification
- Schedule downtime for this service
- Disable event handler for this service
- Disable flap detection for this service
- Clear flapping state for this service

Page Tour

Asnsible Structure

Directory

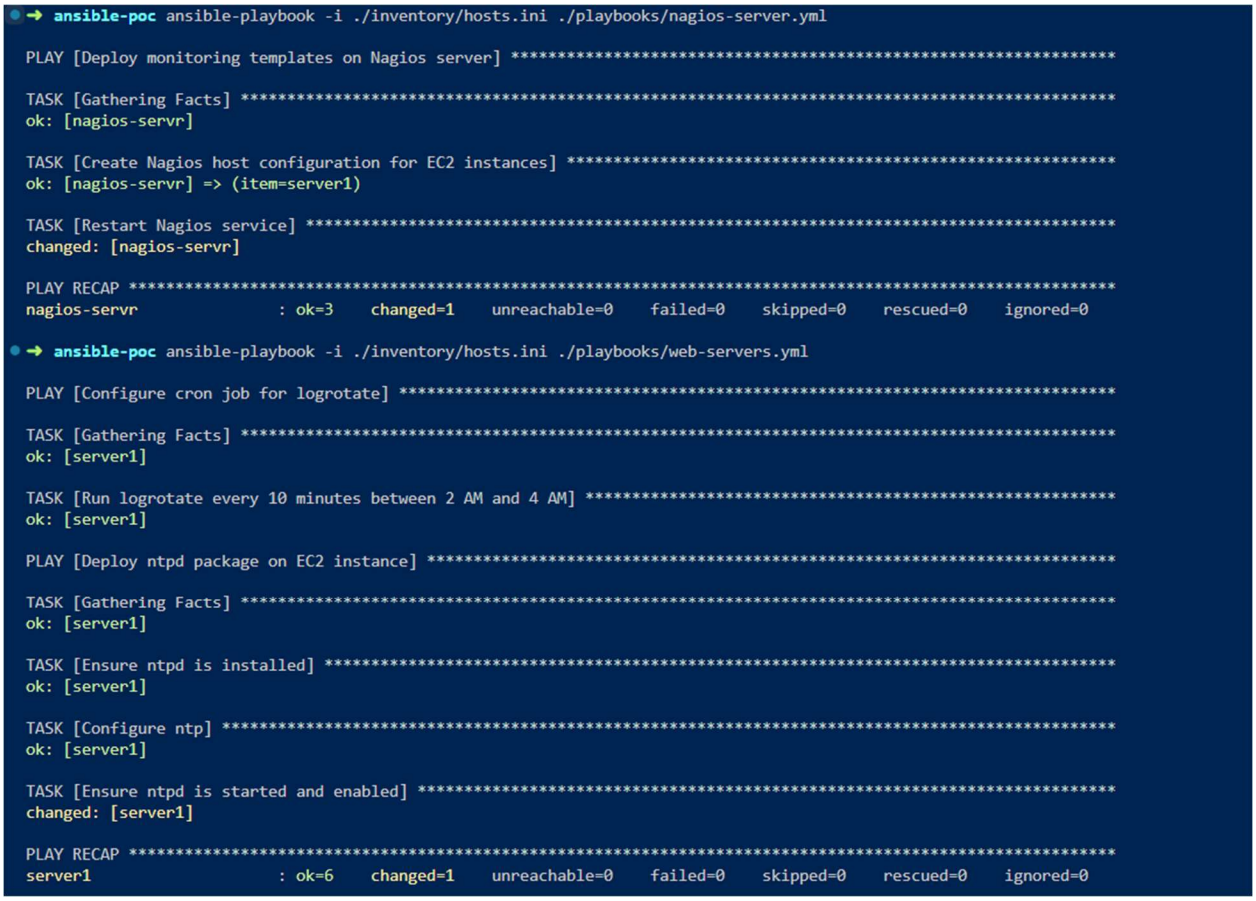


- **Inventory**
  - **All hosts are defined**
- **Playbooks**
  - **1 for the web-server**
  - **1 for the nagios server**
- **Templates**
  - **Nagios configuration jinja template**

Ansible commands to execute

1. **ansible-playbook -i ./inventory/hosts.ini ./playbooks/web-servers.yml**
2. **ansible-playbook -i ./inventory/hosts.ini ./playbooks/nagios-server.yml**

Ansible Output:



Docker -Kubernetes

- Prepare a docker-compose for a nginx server.

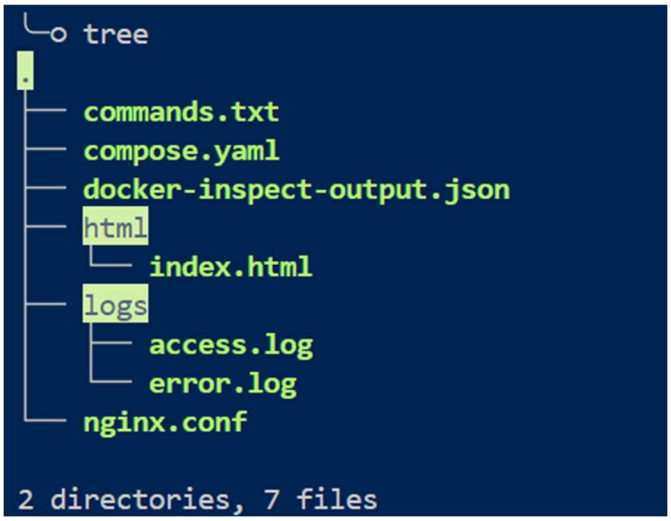
  Requirements:

  - nginx logs need to survive between nginx container restarts
  - docker should use network bridge subnet 172.20.8.0/24

Directory Preparation

For the nginx server a simple **nginx.conf** in root dir and **index.html** file has been created inside html folder .

The docker compose is prepared in **compose.yaml** file



Docker-compose file

- **Volume mounting** : To persist logs between container restart a volume has been mounted from local machine
    - **logs** folder is the mounted path receiving logs
- **Custom Bridge Network**: To have the container an ip from the subnet 172.20.8.0/24, a custom bridge network for docker has been established.

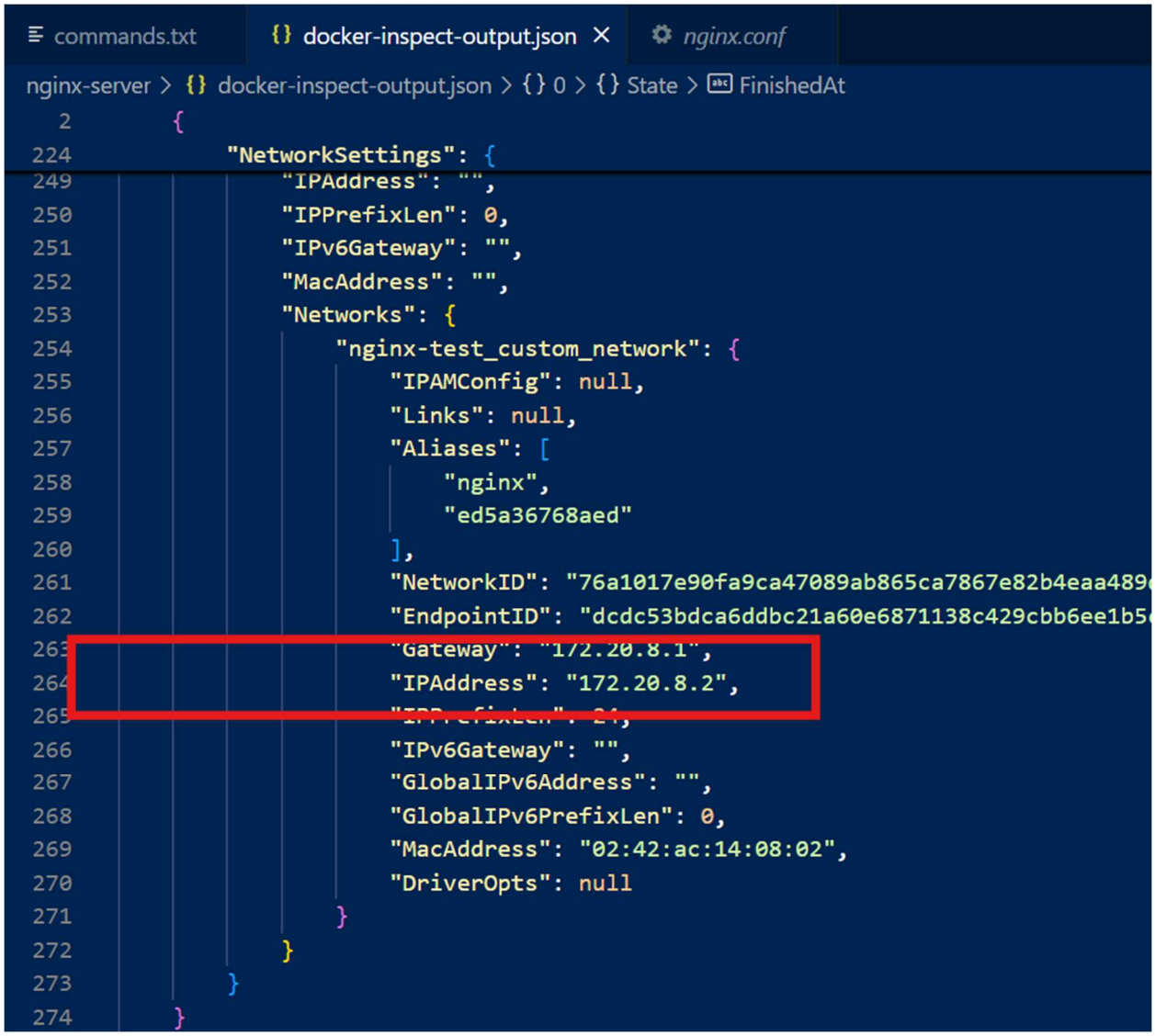
Command to up docker-compose: docker-compose up

Test and validation

Nginx logs:

- Multiple times ***docker compose up*** and ***docker compose down*** have been executed.
- The localhost server has been hit multiple times localhost:80/asr , localhost:80/aserfaaf, localhost:80 in this way. The error logs and access logs of the nginx are always there in the logs file **access.log** and **error.log.**

IP Check: A file has been attached docker-inspect-output.json. This file points out the ip of the container which is taken from the container.



Which Kubernetes command you will use to identify the reason for a pod restart in the project "internal" under namespace "production".

kubectl describe pod <pod-name> -n production

This will provide the events. From events it is most likely few things

- Crashloopbackoff
    i. Application error. Pod log checking afterwards.
- OOM (Out of memory)

kubectl logs <pod-name> -n production –all-cotainers=true

3) **Java-app keep restarting at random. From Kubernetes configuration perspective, what are the possible reasons for the pod restarts?**

The most likely scenario is JVM exceeding the memory limit of 1500. As it already has a heap memory of 1000M allocated it can impact the container runtime with memory spike for other java processes.

However standard debugging process needs to performed side by side

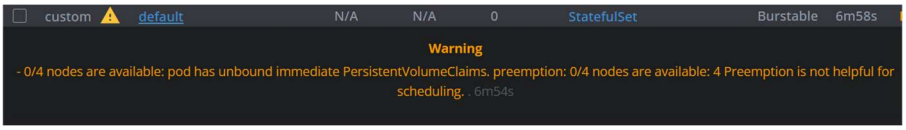
- CPU usage check if CPU is getting throttled or not
- Application crashloopback is happening or not
- Livenessprobe/readinessprobe checks

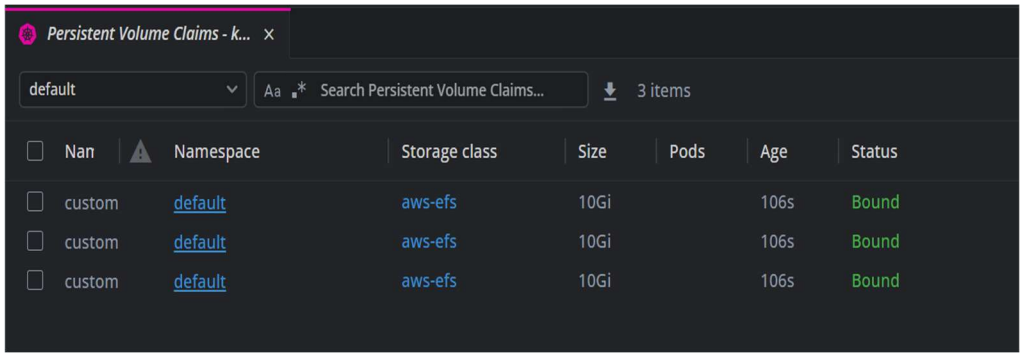
Helm chart

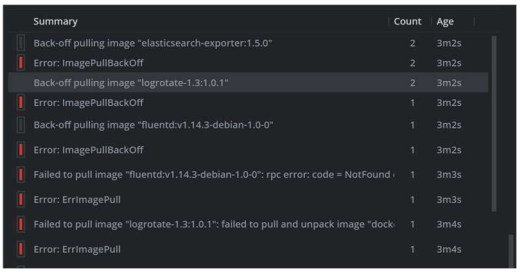
Kubernetes Cluster Details

Kuberentes Cluster: AWS EKS 1.31 version
Cluster Node AMI : AmazonLinux2
PVC/Storageclass: EFS CSI driver

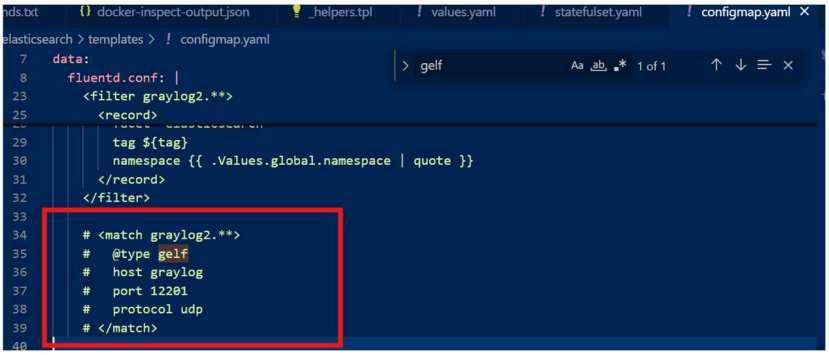
Errors and problems faced during the helm chart deployment

1. Deployed the helm chart with the following command
    - helm upgrade -i elasticsearch ./elasticsearch -f ./elasticsearch/envs/qa/customer-abc-elasticsearch/values.yaml
    - Error1:
        - Error: parse error at (elasticsearch/templates/_helpers.tpl:20): unexpected <define> in command
        - Resolution: Updating helpers.tpl function -> removing custom logic
    - Error2:
        - 

        - Resolution:
            - helm uninstall elasticsearch
            - Updated the storage-class to aws-efs (Which was already present in the eks cluster)
        - PVC looks fine now
            - 

- Error 3:
    - ImagepullBackOff
        - 

        Resolution:

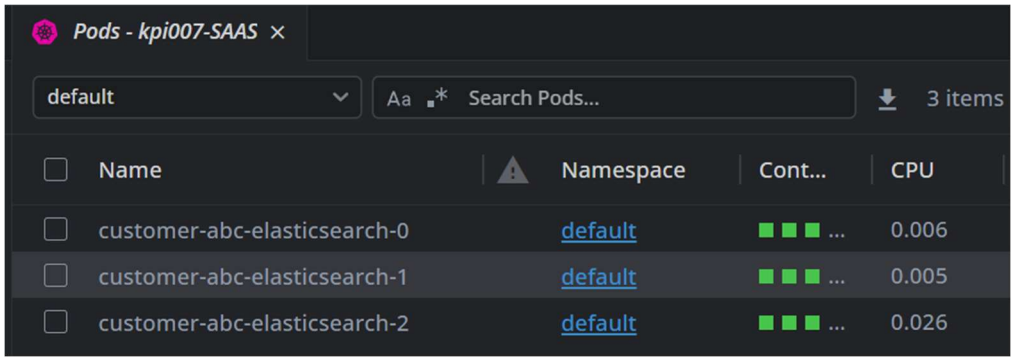
            - Updated the statefulset template with different images available in dockerhub
                - fluentd:v1.18.0-debian-1.0
                - blacklabelops/logrotate:1.3
                - bitnami/elasticsearch-exporter:1.5.0
            - Containers started afterwards
- Error4:
    - gelf is missing from the fluentd container.
    - Temporary resolution: Commented out gelp part from the fluentd config
    - 

    - **Actual resolution:** Install the gelf package in fluentd image

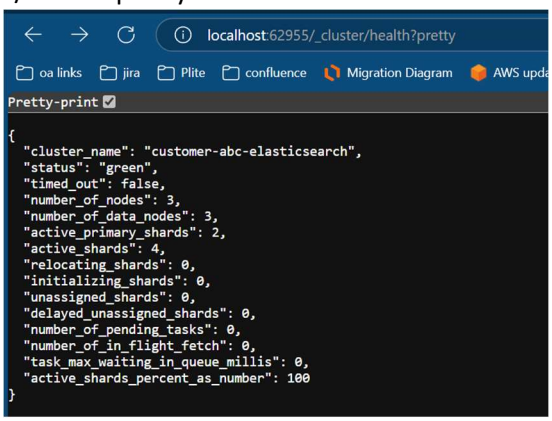
Final output and validation

Final Output

- **Deployed Resources**
  - **1 stateful with 3 replicas**
  - **1 service: http://customer-abc-headless:9200**
  - **Configmaps and secrets**
-



Validation

- **After port-forwarding the service http://customer-abc-headless:9200 to local-browser, triggered the following endpoints**
  - /_cluster/health?pretty
    - 

  - **_cat/nodes?v**

```
ip           heap.percent ram.percent cpu load_1m load_5m load_15m node.role   master name
10.0.141.101           67          68   1    0.89    0.72     0.55 cdfhilmrstw *      customer-abc-elasticsearch-1
10.0.135.81            66          69   1    1.04    0.85     0.59 cdfhilmrstw -      customer-abc-elasticsearch-0
10.0.130.26            26          69   1    1.04    0.85     0.59 cdfhilmrstw -      customer-abc-elasticsearch-2
```

Metrics

Explain how Prometheus work

As a complete monitoring toolkit, with Prometheus, sophisticated and ideal monitoring system can be implemented.

To simply put, how Prometheus works in a Kubernetes cluster, understanding 3 flows will be sufficient

- Scraping metrics with Service discovery: k8s services need to expose metrics for Prometheus. Prometheus service discovery will point these as target endpoints to scrape metrics from
- Time-Series Database: Prometheus aggregates all recorded metrics in a time-series database through Prometheus server
- Leverage promql & alertmanager: With flexible query language like promql, metrics can be parsed and multiple tools are needed to be integrated with Prometheus like alertmanager, Grafana.

How do you create custom Prometheus alerts and alerting rules for Kubernetes monitoring? Provide an example alert rule and its configuration.

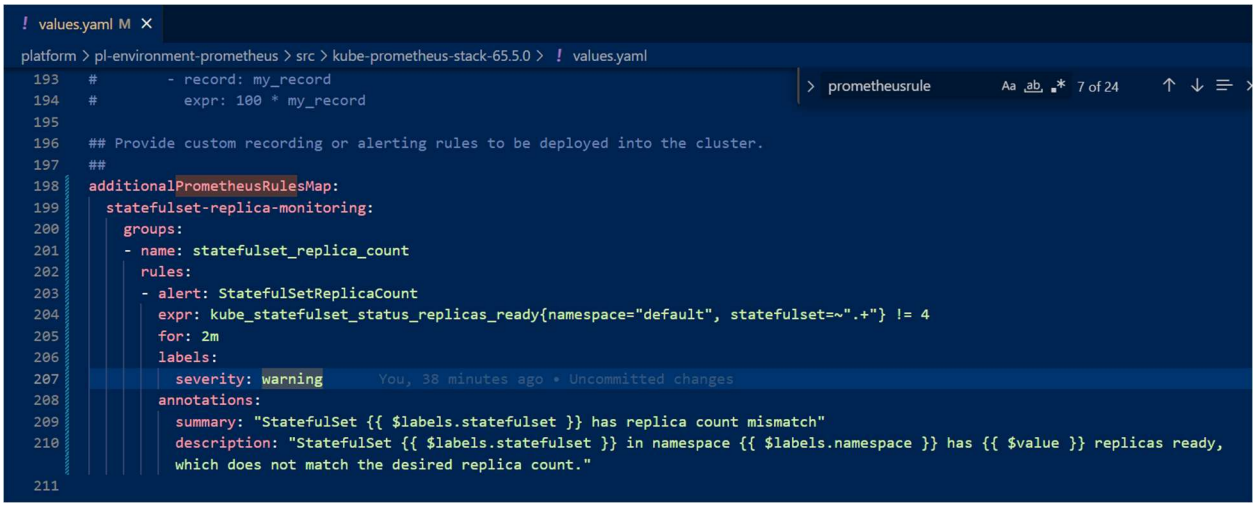
Explaining the answer with the context of most popular and community standard Prometheus helm chart Kube-prometheus-stack. [helm-charts/charts/kube-prometheus-stack at main · prometheus-community/helm-charts]

Ideally to configure an alert, a promql needs to be placed against a specific service. The promql's responsibility will be to monitor scraped metrics against a specific time frame. The alert needs to be placed in helm chart from automation perspective.

Example Alert

In the previous task, elasticsearch statefulset had 3 replicas created. So, a dummy alert has been created where if the statefulset replica count is 4, an alert will be triggered.

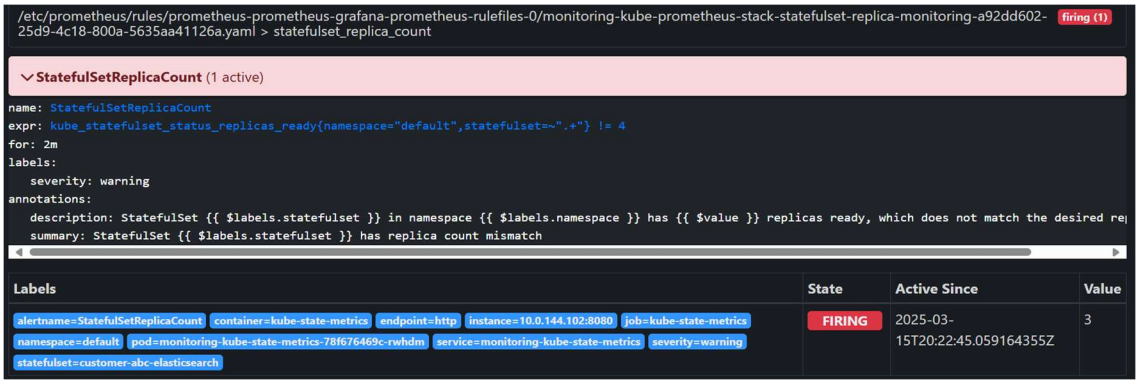
Alert syntax added in the values.yaml file of the helm chart



```
values.yaml M
platform > pl-environment-prometheus > src > kube-prometheus-stack-65.5.0 > values.yaml
193  #         - record: my_record
194  #           expr: 100 * my_record
195
196  ## Provide custom recording or alerting rules to be deployed into the cluster.
197  ##
198  additionalPrometheusRulesMap:
199    statefulset-replica-monitoring:
200      groups:
201      - name: statefulset_replica_count
202        rules:
203        - alert: StatefulSetReplicaCount
204          expr: kube_statefulset_status_replicas_ready{namespace="default", statefulset=~".+"} != 4
205          for: 2m
206          labels:
207            severity: warning        You, 38 minutes ago • Uncommitted changes
208          annotations:
209            summary: "StatefulSet {{ $labels.statefulset }} has replica count mismatch"
210            description: "StatefulSet {{ $labels.statefulset }} in namespace {{ $labels.namespace }} has {{ $value }} replicas ready,
               which does not match the desired replica count."
211
```

prometheusrule Aa ab .* 7 of 24

Created alert in alertmanager UI



/etc/prometheus/rules/prometheus-prometheus-grafana-prometheus-rulefiles-0/monitoring-kube-prometheus-stack-statefulset-replica-monitoring-a92dd602-25d9-4c18-800a-5635aa41126a.yaml > statefulset_replica_count firing (1)

StatefulSetReplicaCount (1 active)

```
name: StatefulSetReplicaCount
expr: kube_statefulset_status_replicas_ready{namespace="default",statefulset=~".+"} != 4
for: 2m
labels:
   severity: warning
annotations:
   description: StatefulSet {{ $labels.statefulset }} in namespace {{ $labels.namespace }} has {{ $value }} replicas ready, which does not match the desired re
   summary: StatefulSet {{ $labels.statefulset }} has replica count mismatch
```

Labels	State	Active Since	Value
alertname=StatefulSetReplicaCount, container=kube-state-metrics, endpoint=http, instance=10.0.144.102:8080, job=kube-state-metrics, namespace=default, pod=monitoring-kube-state-metrics-78f676469c-rwhdm, service=monitoring-kube-state-metrics, severity=warning, statefulset=customer-abc-elasticsearch	FIRING	2025-03-15T20:22:45.059164355Z	3

What is the Prometheus query you can use in Granfana to properly show usage trend of an application metric that is a counter?

For a counter type metric, we need to use rate() type promql. Let's say for a metric name app_hits is a counter whose value gets scraped through Prometheus-operator. For the trend of this metric, rate(app_hits)[10m] will be an ideal query.

As example,

One promql is prepared dedicated to monitor a pod of the statefulset where the metric *container_cpu_usage_seconds_total* is a counter

rate(container_cpu_usage_seconds_total{container="elasticsearch",namespace="default",pod="customer-abc-elasticsearch-0"}[5m])

Database

Cassandra - > Not familiar with Cassandra

Mongo -> Replicaset and Sharding

In the problem statement, due to performance issue, a new replicaset has been provisioned.

Multiple replicaset for mongo:

In a mongo cluster we can have multiple replicaset. The primary node and other replicas. However only the primary node is responsible for the write operations whereas other nodes are only present to handle the replica data and will be used only for the read operations.

Sharding

By sharding, storage will be distributed from the primary node. That means, let's say previously primary node had a total amount of 500 GB of a db. Now it will be in two parts. Depends on the key, let's assume 300gb will stay in one node and the rest will be moved to other replicaset.

How to perform sharding

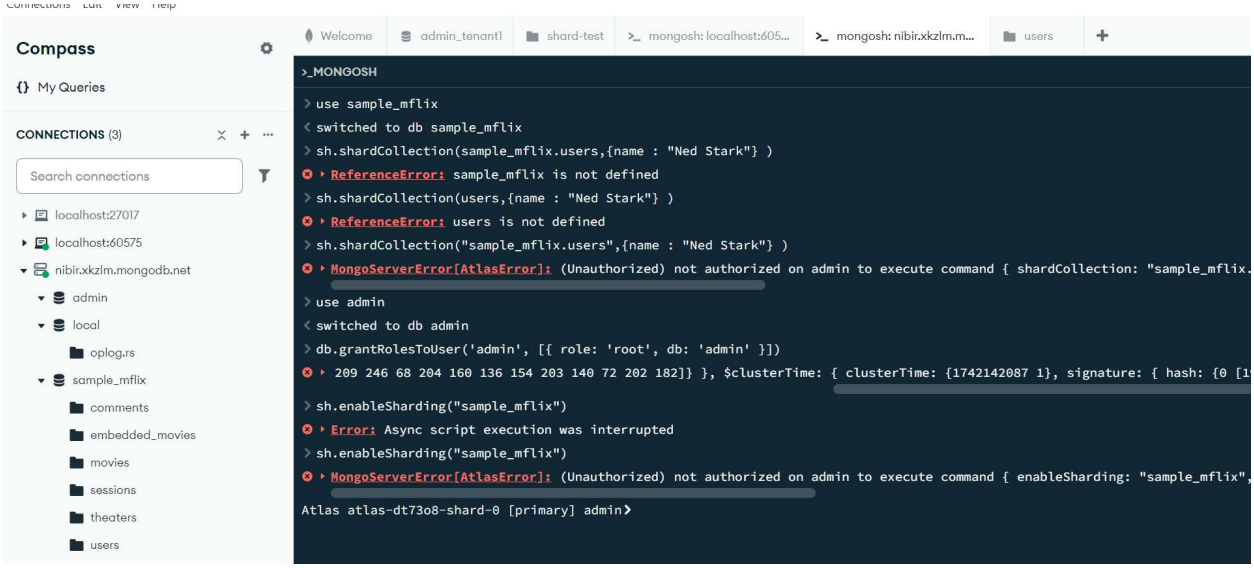
- **Sharding keys:** Index needs to be prepared on the sharding key. As based on the index, sharding modes will be applied (hashed/ranged).
- **Config server and mongos service:** Config servers hold metadata of the shards and **mongos service** routes queries to appropriate shard.
- Executables
    a. Enable sharding on the db :
    sh.enableSharding("sample_db")
    b. Shard collection based on the key:
    sh.shardCollection("sample_db.sample_collection ", { _id: "hashed_key" })

Problems faced for simulating the scenario

1. **Mongo Kubernetes operator:** Deployed mongo as a Kubernetes operator. However, the community mongo operator doesn't support sharding. [mongodb/mongodb-kubernetes-operator: MongoDB Community Kubernetes Operator]

    Mongos service doesn't get deployed with this operator and hence sharding isn't possible.

2.**MongoDB Atlas cluster free tier:** An mongodb atlas cluster was prepared to simulate sharding. In the free tier, it doesn't support sharding.



AWS DocumentDB(Mongo Compatible) : Not tested, but seems like possible [Choose shard keys to optimize Amazon DocumentDB Elastic Clusters | AWS Database Blog]