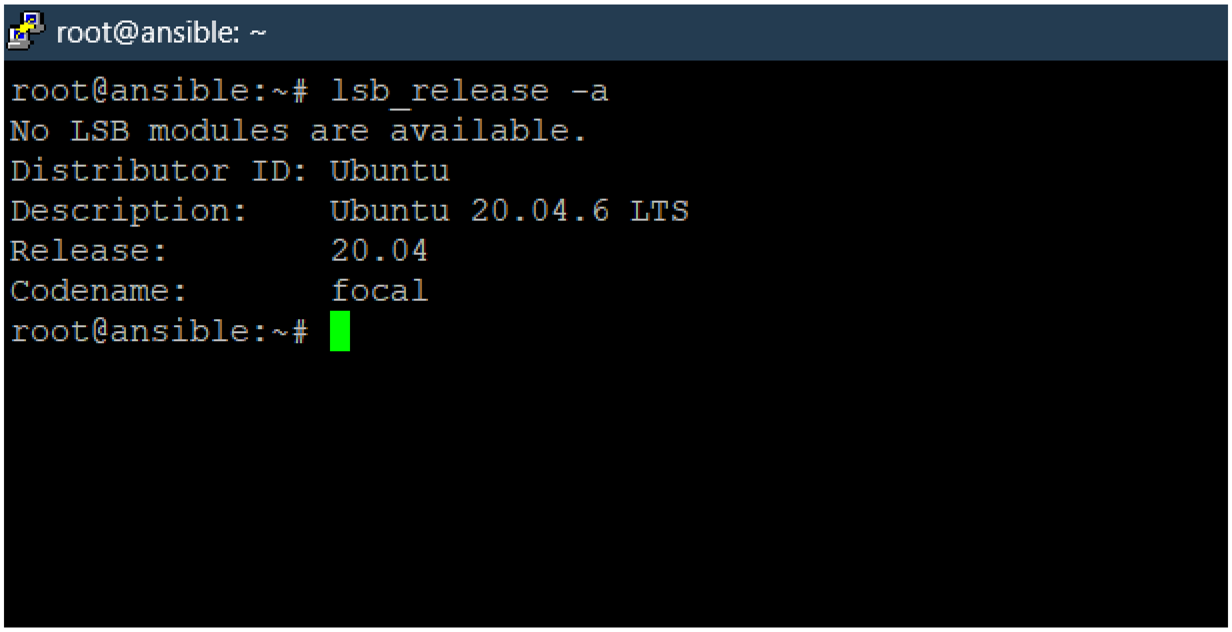
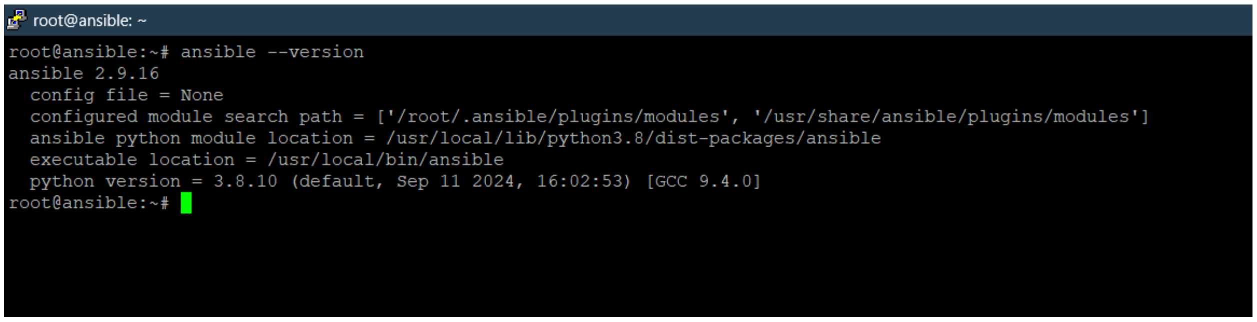
Suggested environment: Ubuntu 20 LTS, ansible 2.9.16,



root@ansible: ~

```
root@ansible:~# lsb_release -a
No LSB modules are available.
Distributor ID: Ubuntu
Description:    Ubuntu 20.04.6 LTS
Release:        20.04
Codename:       focal
root@ansible:~#
```

ansible 2.9.16



root@ansible: ~

```
root@ansible:~# ansible --version
ansible 2.9.16
  config file = None
  configured module search path = ['/root/.ansible/plugins/modules', '/usr/share/ansible/plugins/modules']
  ansible python module location = /usr/local/lib/python3.8/dist-packages/ansible
  executable location = /usr/local/bin/ansible
  python version = 3.8.10 (default, Sep 11 2024, 16:02:53) [GCC 9.4.0]
root@ansible:~#
```

1) Which ansible command can display all ansible_ configuration for a host.

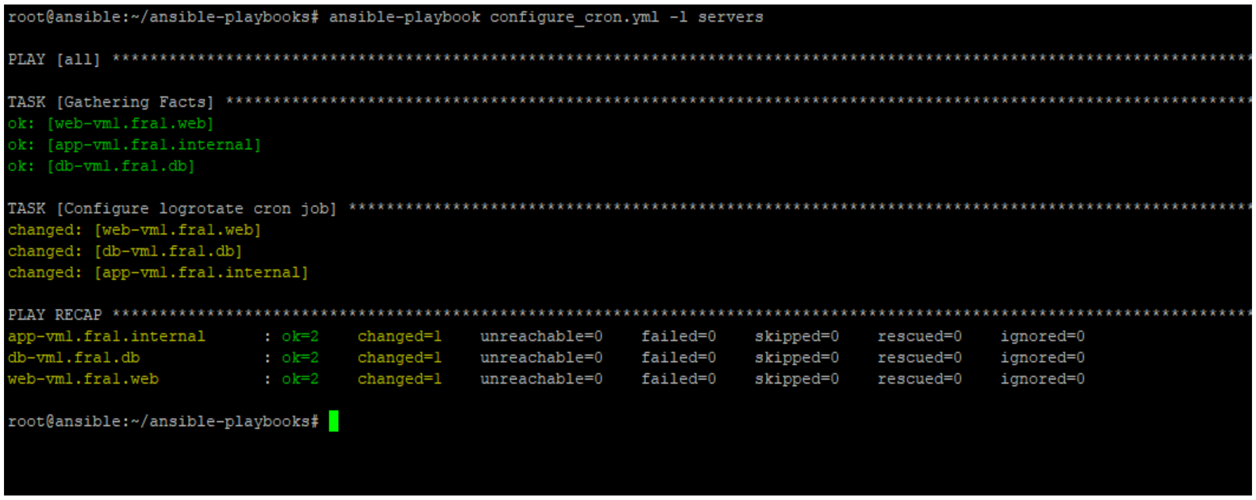
```
#ansible all -m setup
```

1) Please configure a cron job that runs logrotate on all machines every 10 minutes between 2h - 4h.

Ansible Playbook:

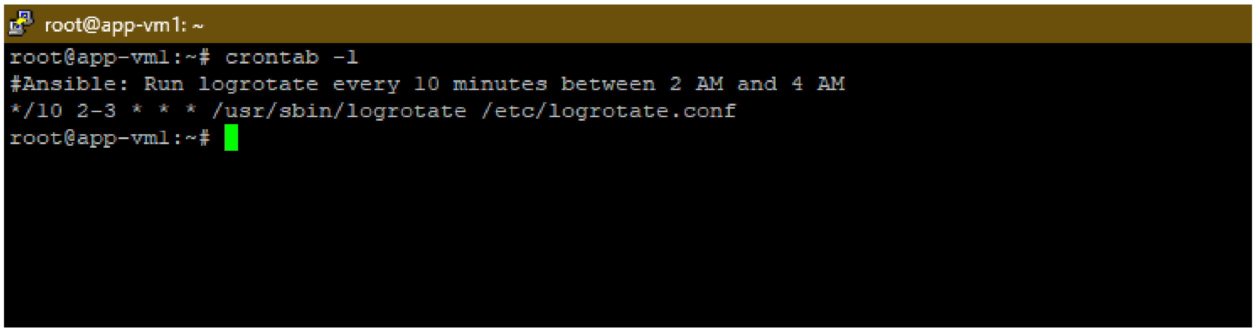
```
- hosts:
    - app-vm1.fra1.internal
    - db-vm1.fra1.db
    - web-vm1.fra1.web
  tasks:
    - name: Install ntpd
      apt:
        name: ntp
        state: present

    - name: Deploy custom ntpd.conf
      copy:
        src: /home/dnet/ntpd.conf  # Ensure this points to the actual
file
        dest: /etc/ntp.conf
        owner: root
        group: root
        mode: '0644'
```

```
- hosts:
     - all
  tasks:
     - name: Configure logrotate cron job
     cron:
     name: "Run logrotate every 10 minutes between 2 AM and 4 AM"
     minute: "*/10"
     hour: "2-3"  # 2 AM to 3:59 AM
     job: "/usr/sbin/logrotate /etc/logrotate.conf"
     state: present
```



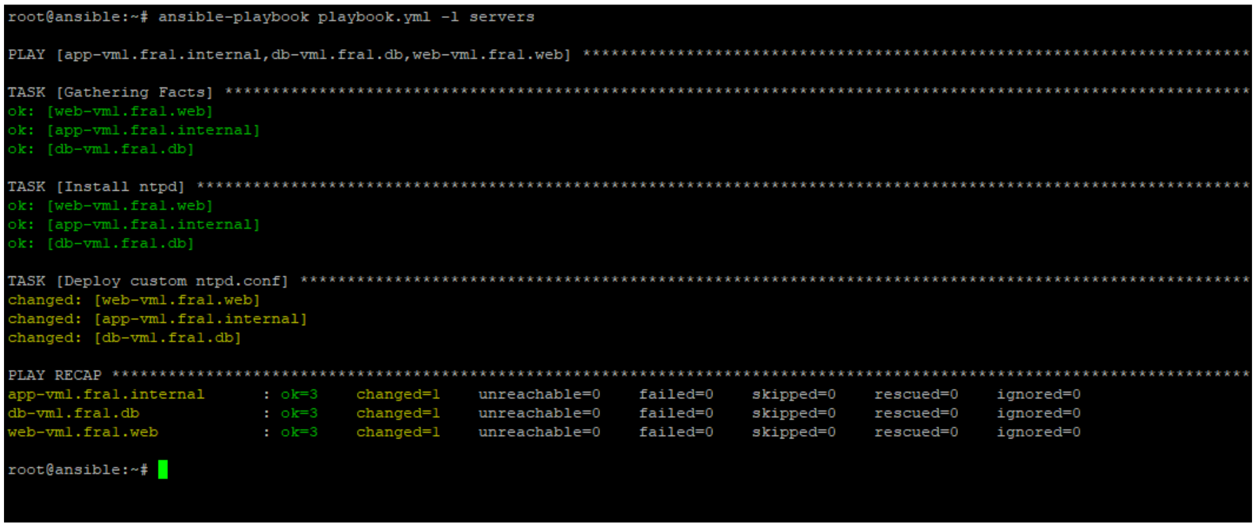
```
root@ansible:~/ansible-playbooks# ansible-playbook configure_cron.yml -l servers

PLAY [all] ************************************************************************************************************************

TASK [Gathering Facts] ************************************************************************************************************
ok: [web-vm1.fra1.web]
ok: [app-vm1.fra1.internal]
ok: [db-vm1.fra1.db]

TASK [Configure logrotate cron job] ***********************************************************************************************
changed: [web-vm1.fra1.web]
changed: [db-vm1.fra1.db]
changed: [app-vm1.fra1.internal]

PLAY RECAP ************************************************************************************************************************
app-vm1.fra1.internal      : ok=2    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
db-vm1.fra1.db             : ok=2    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
web-vm1.fra1.web           : ok=2    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0

root@ansible:~/ansible-playbooks#
```

Its set to every server



```
root@app-vm1:~# crontab -l
#Ansible: Run logrotate every 10 minutes between 2 AM and 4 AM
*/10 2-3 * * * /usr/sbin/logrotate /etc/logrotate.conf
root@app-vm1:~#
```

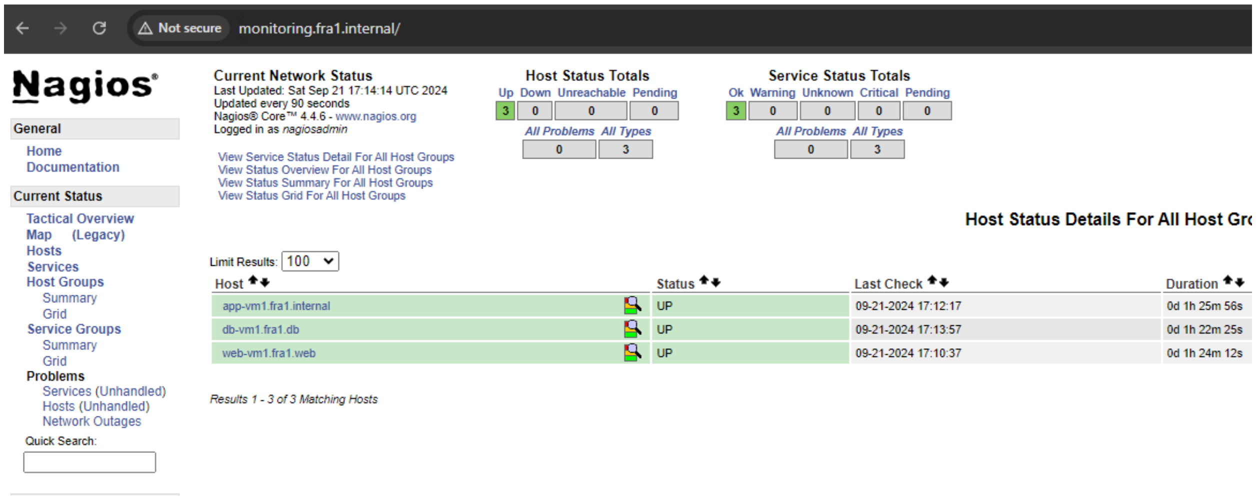
```
#ansible-playbook playbook.yml -l servers
```



```
root@ansible:~# ansible-playbook playbook.yml -l servers

PLAY [app-vm1.fra1.internal,db-vm1.fra1.db,web-vm1.fra1.web] **********************************************************************

TASK [Gathering Facts] ************************************************************************************************************
ok: [web-vm1.fra1.web]
ok: [app-vm1.fra1.internal]
ok: [db-vm1.fra1.db]

TASK [Install ntpd] ***************************************************************************************************************
ok: [web-vm1.fra1.web]
ok: [app-vm1.fra1.internal]
ok: [db-vm1.fra1.db]

TASK [Deploy custom ntpd.conf] ****************************************************************************************************
changed: [web-vm1.fra1.web]
changed: [app-vm1.fra1.internal]
changed: [db-vm1.fra1.db]

PLAY RECAP ************************************************************************************************************************
app-vm1.fra1.internal      : ok=3    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
db-vm1.fra1.db             : ok=3    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
web-vm1.fra1.web           : ok=3    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0

root@ansible:~#
```

We also need to deploy monitoring template onto our nagios server “monitoring.fra1.internal”, each of the above machines should use the following nagios templates:

```
# Host Definitions
define host {
  host_name                       app-vm1.fra1.internal
  address                         192.168.0.2
  check_command                   check_ping
  active_checks_enabled           1
  passive_checks_enabled          1
  max_check_attempts              3
}

define host {
  host_name                       db-vm1.fra1.db
  address                         192.168.0.3
  check_command                   check_ping
  active_checks_enabled          1
  passive_checks_enabled         1
  max_check_attempts             3
}

define host {
  host_name                      web-vm1.fra1.web
  address                        192.168.0.4
  check_command                  check_ping
  active_checks_enabled          1
  passive_checks_enabled         1
  max_check_attempts             3
}

# Service Definitions
define service {
  service_description            ntp_process
  host_name                      app-vm1.fra1.internal
  check_command                  check_ntp
  check_interval                 10
  max_check_attempts             3
}

define service {
  service_description            ntp_process
  host_name                      db-vm1.fra1.db
  check_command                  check_ntp
  check_interval                 10
  max_check_attempts             3
}

define service {
  service_description            ntp_process
  host_name                      web-vm1.fra1.web
  check_command                  check_ntp
  check_interval                 10
  max_check_attempts             3
}
```



Current Network Status
Last Updated: Sat Sep 21 17:14:14 UTC 2024
Updated every 90 seconds
Nagios® Core™ 4.4.6 - www.nagios.org
Logged in as *nagiosadmin*

View Service Status Detail For All Host Groups
View Status Overview For All Host Groups
View Status Summary For All Host Groups
View Status Grid For All Host Groups

Host Status Totals

Up	Down	Unreachable	Pending
3	0	0	0

All Problems	*All Types*
0	3

Service Status Totals

Ok	Warning	Unknown	Critical	Pending
3	0	0	0	0

All Problems	*All Types*
0	3

Host Status Details For All Host Gro

Limit Results: 100

Host	Status	Last Check	Duration
app-vm1.fra1.internal	UP	09-21-2024 17:12:17	0d 1h 25m 56s
db-vm1.fra1.db	UP	09-21-2024 17:13:57	0d 1h 22m 25s
web-vm1.fra1.web	UP	09-21-2024 17:10:37	0d 1h 24m 12s

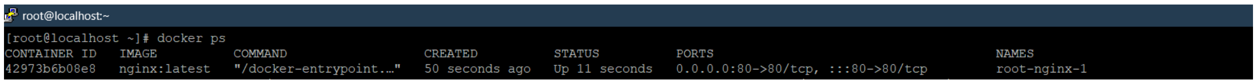
Results 1 - 3 of 3 Matching Hosts

1) Prepare a docker-compose for a nginx server.

Requirements:

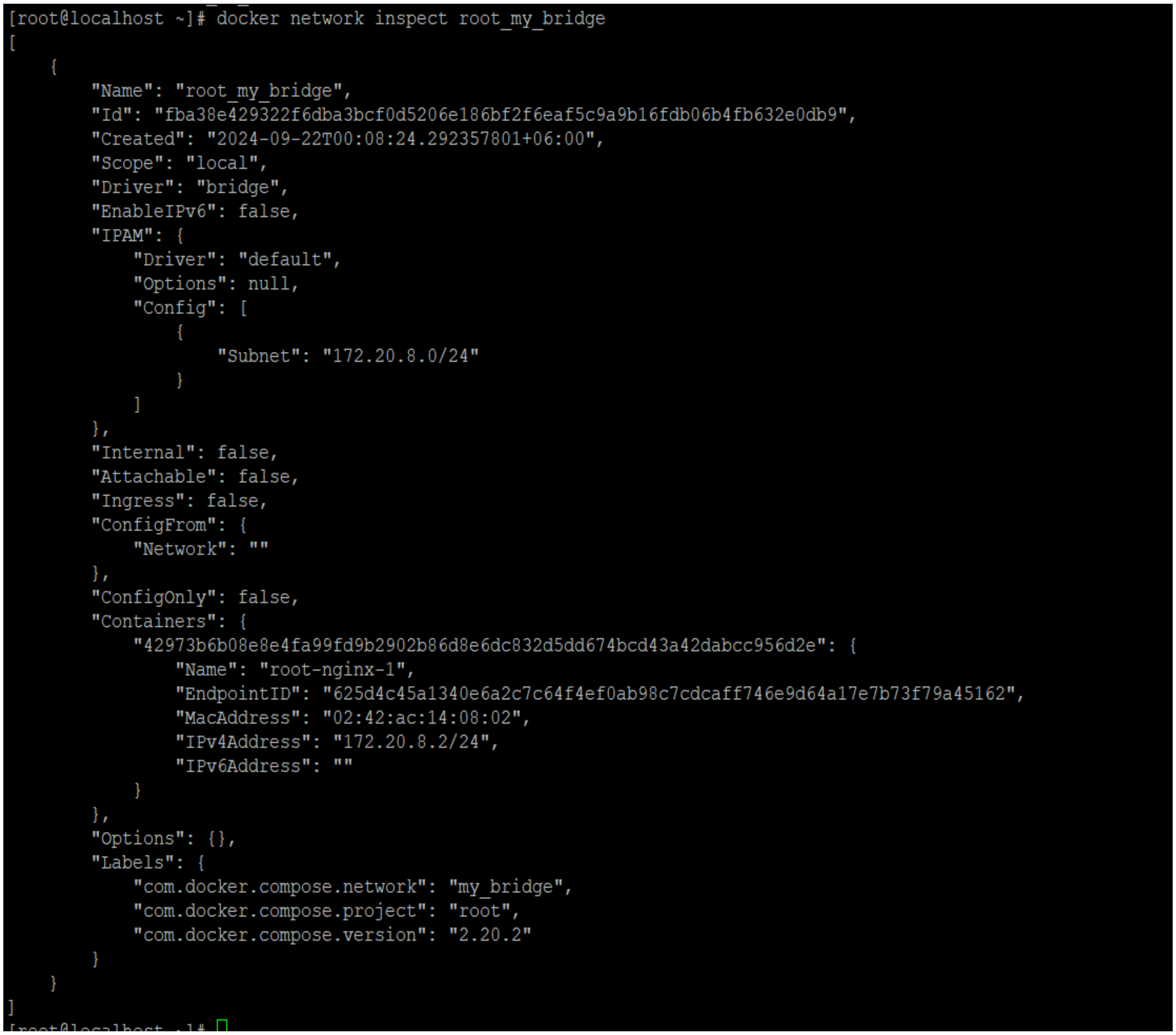
- nginx logs need to survive between nginx container restarts
- docker should use network bridge subnet 172.20.8.0/24

```
version: '3.8'

services:
  nginx:
    image: nginx:latest
    volumes:
      - nginx_logs:/var/log/nginx
    networks:
      my_bridge:
        ipv4_address: 172.20.8.2
    ports:
      - "80:80"

volumes:
  nginx_logs:
    driver: local

networks:
  my_bridge:
    driver: bridge
    ipam:
      config:
        - subnet: 172.20.8.0/24
```



```
root@localhost:~
[root@localhost ~]# docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED          STATUS          PORTS                                 NAMES
42973b6b08e8   nginx:latest   "/docker-entrypoint.…"   50 seconds ago   Up 11 seconds   0.0.0.0:80->80/tcp, :::80->80/tcp     root-nginx-1
```

```
[root@localhost ~]# docker network inspect root_my_bridge
```



```
[root@localhost ~]# docker network inspect root_my_bridge
[
    {
        "Name": "root_my_bridge",
        "Id": "fba38e429322f6dba3bcf0d5206e186bf2f6eaf5c9a9b16fdb06b4fb632e0db9",
        "Created": "2024-09-22T00:08:24.292357801+06:00",
        "Scope": "local",
        "Driver": "bridge",
        "EnableIPv6": false,
        "IPAM": {
            "Driver": "default",
            "Options": null,
            "Config": [
                {
                    "Subnet": "172.20.8.0/24"
                }
            ]
        },
        "Internal": false,
        "Attachable": false,
        "Ingress": false,
        "ConfigFrom": {
            "Network": ""
        },
        "ConfigOnly": false,
        "Containers": {
            "42973b6b08e8e4fa99fd9b2902b86d8e6dc832d5dd674bcd43a42dabcc956d2e": {
                "Name": "root-nginx-1",
                "EndpointID": "625d4c45a1340e6a2c7c64f4ef0ab98c7cdcaff746e9d64a17e7b73f79a45162",
                "MacAddress": "02:42:ac:14:08:02",
                "IPv4Address": "172.20.8.2/24",
                "IPv6Address": ""
            }
        },
        "Options": {},
        "Labels": {
            "com.docker.compose.network": "my_bridge",
            "com.docker.compose.project": "root",
            "com.docker.compose.version": "2.20.2"
        }
    }
]
[root@localhost ~]#
```

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

```
kubectl describe pod my-pod -n production
```

1) Consider the followings:

POD	NAME	CPU(cores)	MEMORY(bytes)
java-app-7d9d44ccbf-lmvbc	java-app	3m	951Mi
java-app-7d9d44ccbf-lmvbc	java-app-logrotate	1m	45Mi
java-app-7d9d44ccbf-lmvbc	java-app-fluentd	1m	84Mi
java-app-7d9d44ccbf-lmvbc	mongos	4m	62Mi

Application pod has the following resource quota:

· Memory request & limit: 1000 & 1500

· CPU request & limit: 1000 & 2000

· Xmx of 1000M

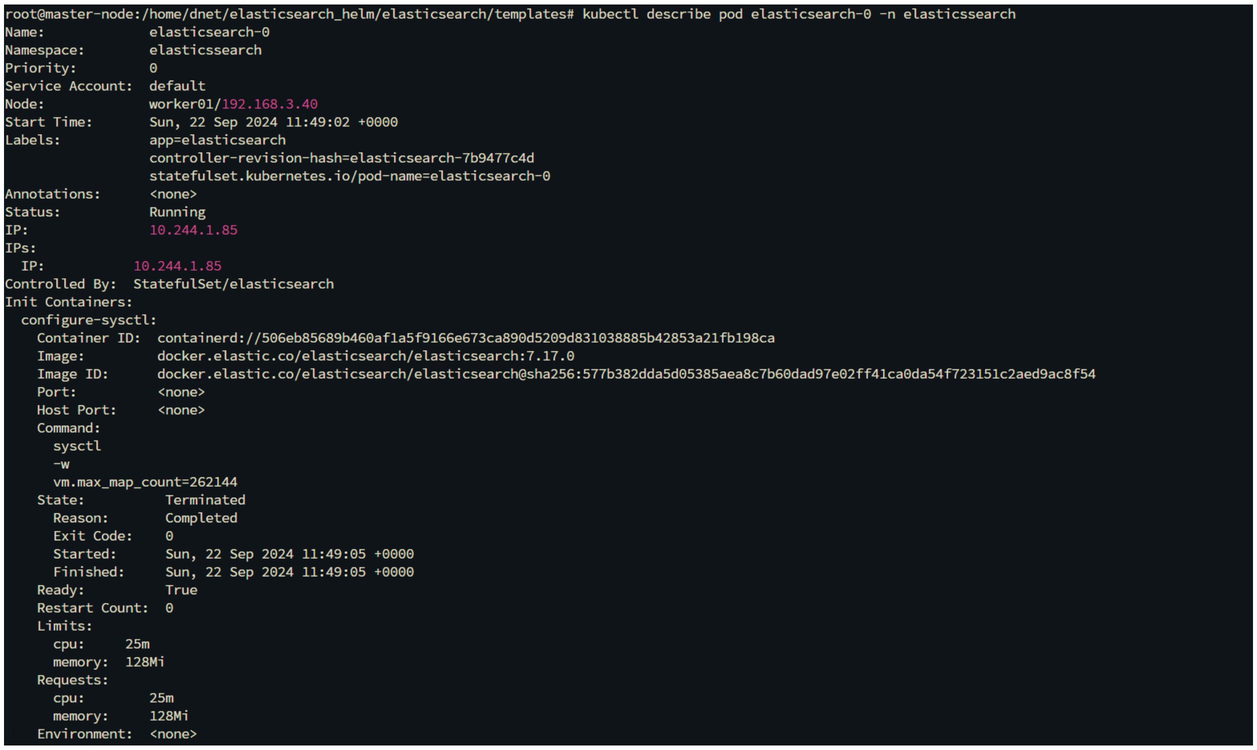
Java-app keep restarting at random. From Kubernetes configuration perspective, what are the possible reasons for the pod restarts?

Possible Reasons for Pod Restarts:

If the pod exceeds its memory limit, the Linux kernel will invoke the Out-Of-Memory (OOM) killer, causing the pod to restart with the reason OOMKilled.

Please use the accompanied elasticsearch helm template to create a Kubernetes deployment of elasticsearch.

Provide a screenshot & deployment yaml of the resultant deployment in Kubernetes.



```
root@master-node:/home/dnet/elasticsearch_helm/elasticsearch/templates# kubectl describe pod elasticsearch-0 -n elasticssearch
Name:             elasticsearch-0
Namespace:        elasticssearch
Priority:         0
Service Account:  default
Node:             worker01/192.168.3.40
Start Time:       Sun, 22 Sep 2024 11:49:02 +0000
Labels:           app=elasticsearch
                  controller-revision-hash=elasticsearch-7b9477c4d
                  statefulset.kubernetes.io/pod-name=elasticsearch-0
Annotations:      <none>
Status:           Running
IP:               10.244.1.85
IPs:
  IP:           10.244.1.85
Controlled By:  StatefulSet/elasticsearch
Init Containers:
  configure-sysctl:
    Container ID:  containerd://506eb85689b460af1a5f9166e673ca890d5209d831038885b42853a21fb198ca
    Image:         docker.elastic.co/elasticsearch/elasticsearch:7.17.0
    Image ID:      docker.elastic.co/elasticsearch/elasticsearch@sha256:577b382dda5d05385aea8c7b60dad97e02ff41ca0da54f723151c2aed9ac8f54
    Port:          <none>
    Host Port:     <none>
    Command:
      sysctl
      -w
      vm.max_map_count=262144
    State:          Terminated
      Reason:       Completed
      Exit Code:    0
      Started:      Sun, 22 Sep 2024 11:49:05 +0000
      Finished:     Sun, 22 Sep 2024 11:49:05 +0000
    Ready:          True
    Restart Count:  0
    Limits:
      cpu:     25m
      memory:  128Mi
    Requests:
      cpu:        25m
      memory:     128Mi
    Environment:  <none>
```

https://github.com/zeesaan/DevOps.git

Explain how Prometheus work

Prometheus collects data in the form of time series through a pull model. The Prometheus server scrapes a list of data sources (often called exporters) at a specified polling frequency.
Data is stored as metrics, each identified by a unique name used for referencing and querying. Prometheus stores data locally on disk, which allows for fast data storage and querying, but it also has the capability to store metrics in remote storage systems.
Each Prometheus server is standalone, operating independently without relying on network storage or other remote services.

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

```
groups:
  - name: kubernetes-alerts
    rules:
      - alert: HighCpuUsage
        expr:
sum(rate(container_cpu_usage_seconds_total{job="kubelet", cluster="",
image!="", container!="POD"}[5m])) by (pod) > 0.8
        for: 5m
        labels:
          severity: warning
        annotations:
          summary: "High CPU usage detected"
          description: "Pod {{ $labels.pod }} is using more than 80%
CPU."
```

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

Grafana using Prometheus, use the following query:

```
rate(your_counter_metric_name[5m])
```

Create a New Dashboard in Grafana.

Add a Panel and select Graph.

Choose Prometheus as the data source.

Enter your query (e.g., rate `(http_requests_total[5m]))`.

Save your dashboard.