Methods For Service Health Check

**What is API Endpoint ?**

* Endpoint of a communication channel
* When an api interacts with another system or service the communication point is called as service endpoint or api endpoint
* Example : url of a server or a service
* Use requests and response to query api

**Openstack Services**

* KEYSTONE- provides API for client authentication, service discovery and multi-tenant

authorization.

* GLANCE - provide services for discovering, registering, and retrieving virtual machine

images.

* NOVA - provides services for scaling and accessing compute resources on demand
* NEUTRON - provides networking-as-a-service (NaaS) in virtual compute

environments.

* SWIFT - provides highly available, distributed and consistent object/blob storage.
* CINDER - provides block storage service for openstack, virtualizes the management of block storage and consumes the storage resource without the user having any knowledge where the actual storage is located and what type of storage is used.
* HORIZON - provides a web-based interface for openstack services to the end-user.

**Different type of health checks to verify the cloud health**

1. Operator test

* Check all services are running and listening on the ports .
* Check if the docker containers are up in the nodes.
* Check if the nodes are reachable.
* Check the galera cluster status.
* Check if the ceph cluster is in the healthy state.
* Check the cluster status of infra components rabbit and percona (mysql ‘wsrep’ and rabbitmqctl cluster\_status)
* If Openstack is in HA mode, test the HAProxy and each of the services behind the HAProxy (run 'a' and 'b’)
* If pacemaker is installed, use 'crm status' or ‘pcs status'

Requires cloud-admin and operator access.

2. Endpoint test

* keystone service-list
* glance image-list
* cinder list
* nova list
* neutron net-list
* login to horizon page

3. Functional test

* Create tenant, create network, upload an image, create two VMs and run ping between the VMs.
* Create VM, create volume, attach volume to the VM.
* Detach VM, delete volume and delete VM
* Clean up all resources

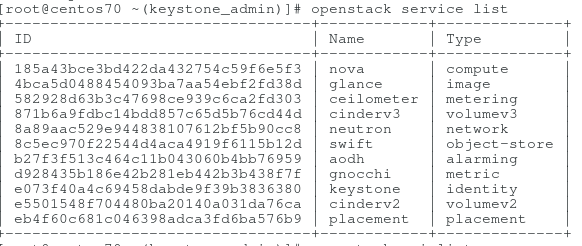
4. Comprehensive health test

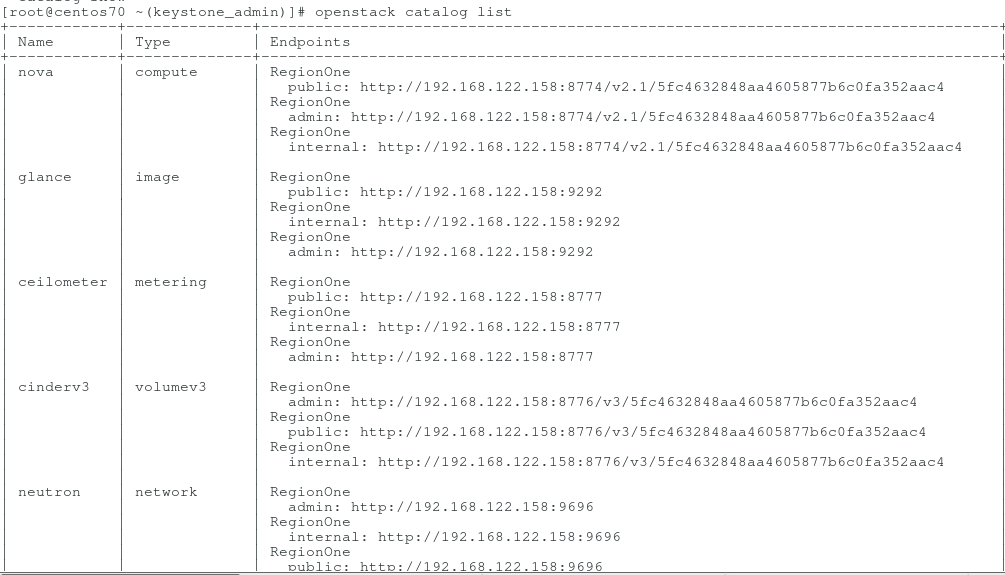
* Create a VM on each compute node and ping the gateway.
* Determine max MTU and check jumbo packets (optional)
* Check security groups (ping, ssh and http traffic)

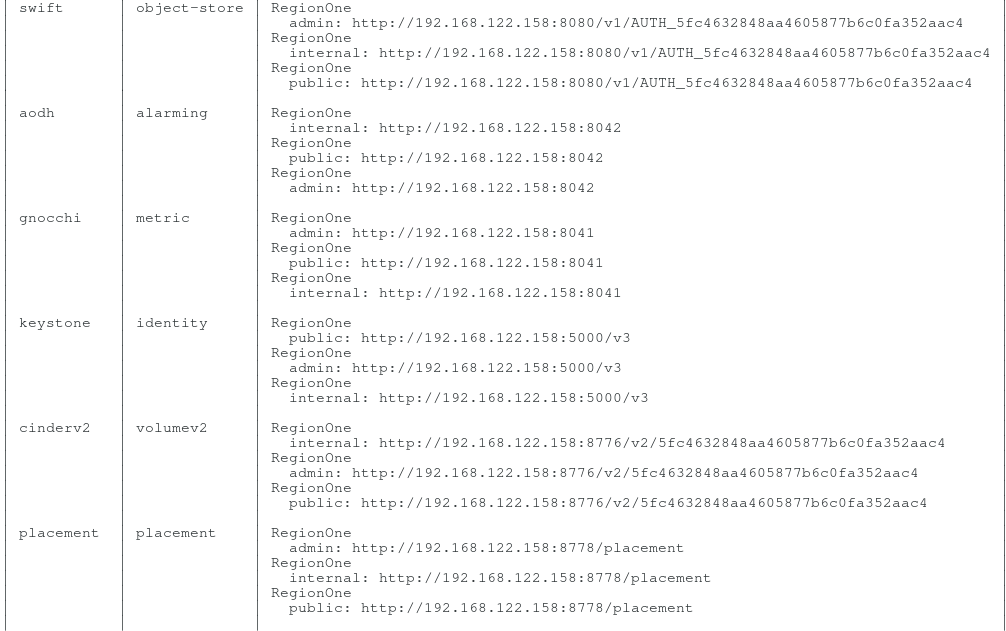
5. Upgrade test

* Create or snapshot the state of existing openstack resources such as tenants/routers/VMs/Loadbalancers
* After upgrade check if the created/snapshotted resources are in operational state
* Check security groups after upgrade (ping, ssh and http)

Command to list down all the Services

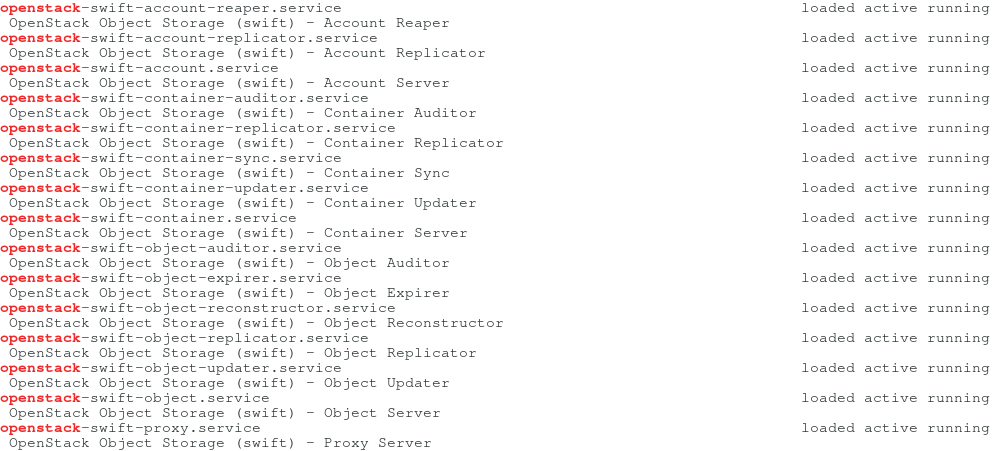


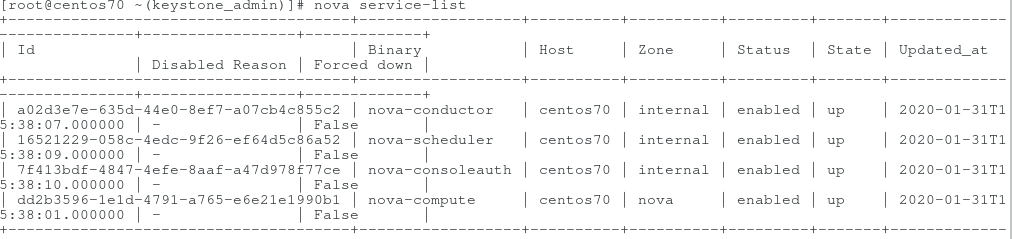


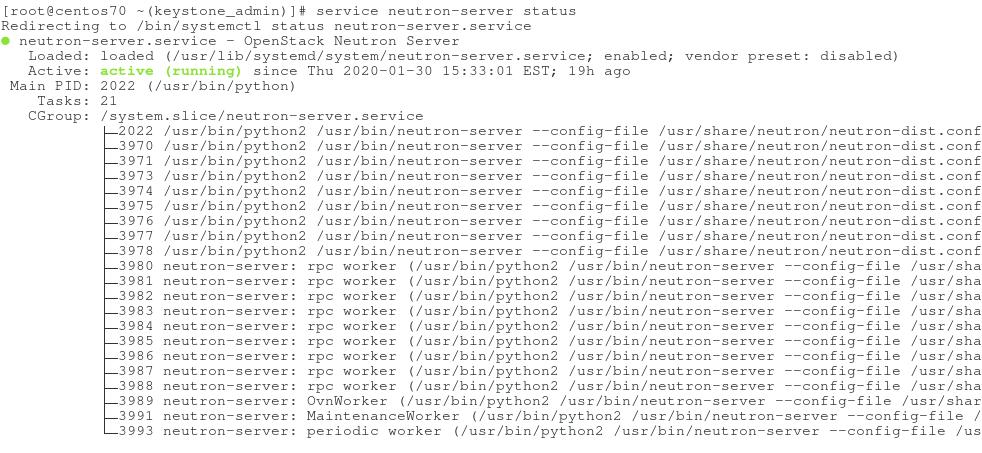


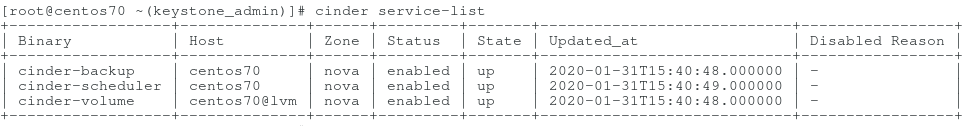
Command to run On Controller node to check the status of openstack services running

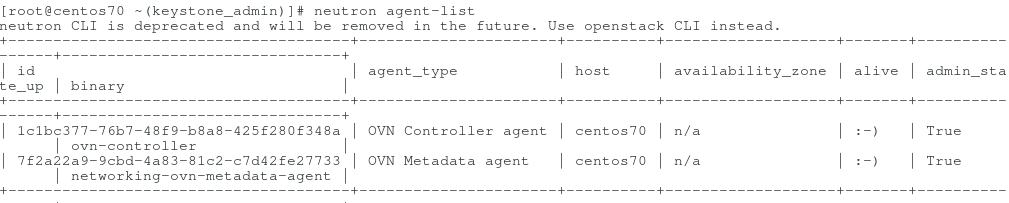


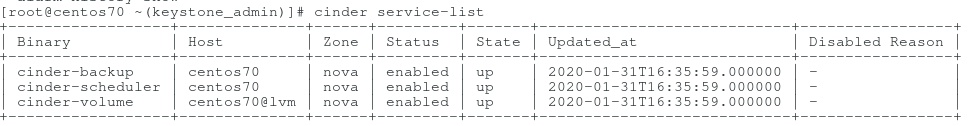












**Tools that can used for health check**

**Nagios**

* provides dashboard
* custom plugins
* can use ansible to run periodic checks
* Requires admin access

**CloudPulse**

* Requires admin access
* command line utility,
* use rest APIs to query the openstack services and returns a JSON response

**Ansible**

* Write playbooks to run service check commands on each node

**ELK stack + Heartbeat**

1. Heartbeat can operate from inside or outside of your network. All it needs is network access to the desired HTTP, TCP, or ICMP endpoint.
2. Configuration is as simple as supplying Heartbeat with the list of URLs you want to monitor.
3. Heartbeat executes periodic checks to verify whether the endpoint is up or down, then reports this information, along with other useful metrics, to Elasticsearch. This information is automatically displayed in pre-built Kibana dashboards for turnkey uptime monitoring.

**How to Setup ELK Stack and Heartbeat to monitor the health of services using api endpoints**

* **Elasticsearch:** This is an open source, distributed, RESTful, JSON-based search engine. It is scalable, easy to use, and flexible
* **Logstash :** This is a server‑side data processing pipeline that ingests data from multiple sources simultaneously, transforms it, and then sends it to a “stash” like Elasticsearch.
* **Kibana** lets users visualize data with charts and graphs in Elasticsearch.

For the experimentation purpose I installed ELK stack on Centos 7 VM on virtual box.

## Install ELK stack

### Step 1: Install Java

One of the main dependencies for elastic search is JAVA, so you need to install JAVA first on your Centos system before moving ahead. Run the following command to install it :

sudo yum -y install java-openjdk-devel java-openjdk

### Step 2 : Add ELK stack repository in your system

Copy paste the following code on your terminal

cat <<EOF | sudo tee /etc/yum.repos.d/elasticsearch.repo

[elasticsearch-7.x]

name=Elasticsearch repository for 7.x packages

baseurl=<https://artifacts.elastic.co/packages/7.x/yum>

gpgcheck=1

gpgkey=<https://artifacts.elastic.co/GPG-KEY-elasticsearch>

enabled=1

autorefresh=1

type=rpm-md

EOF

After that , now import the GPG key

sudo rpm --import<https://artifacts.elastic.co/GPG-KEY-elasticsearch>

Clear and update your YUM package index

sudo yum clean all

sudo yum makecache

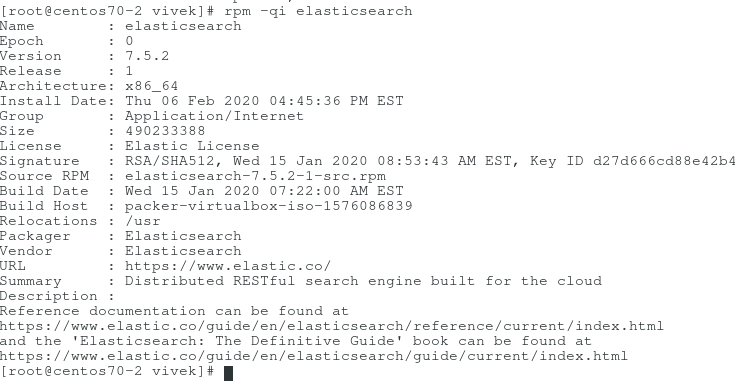
### Step 3: Install and Configure Elasticsearch

Since you have already added the repository in previous steps, you can run below command on the terminal to install elasticsearch

sudo yum -y install elasticsearch

Execute the below command to confirm the installation

rpm -qi elasticsearch



You can set JVM options like memory limits by editing the file:

/etc/elasticsearch/jvm.options



To access elasticsearch on your IP , edit the elasticsearch.yml in location /etc/elasticsearch/ and edit following parameters as shown below:

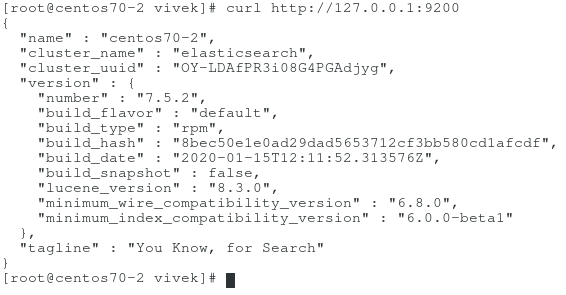
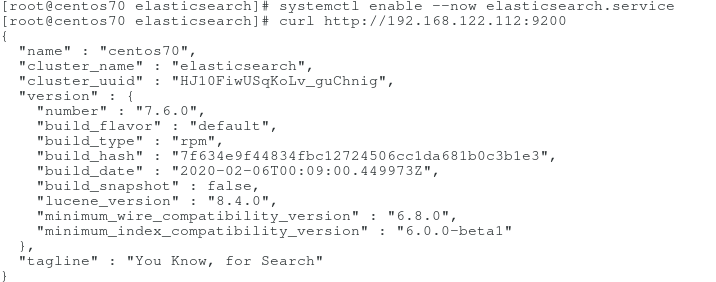
network.host: 127.0.0.1

http.host: 0.0.0.0

Execute the below command to start and enable the elasticsearch service:

$ sudo systemctl enable --now elasticsearch.service

To verify the above process , curl the localhost on port 9200 as shown below:



### Step 4: Configure and Install kibana

Since we have already added the elasticsearch repository, we just need to yum install the kibana package as shown below:

sudo yum -y install kibana

Configure kibana, edit the kibana.yml file in location - /etc/kibana/

Edit the following fields as shown below or uncomment if already present :

server.host: "localhost" or “0.0.0.0”

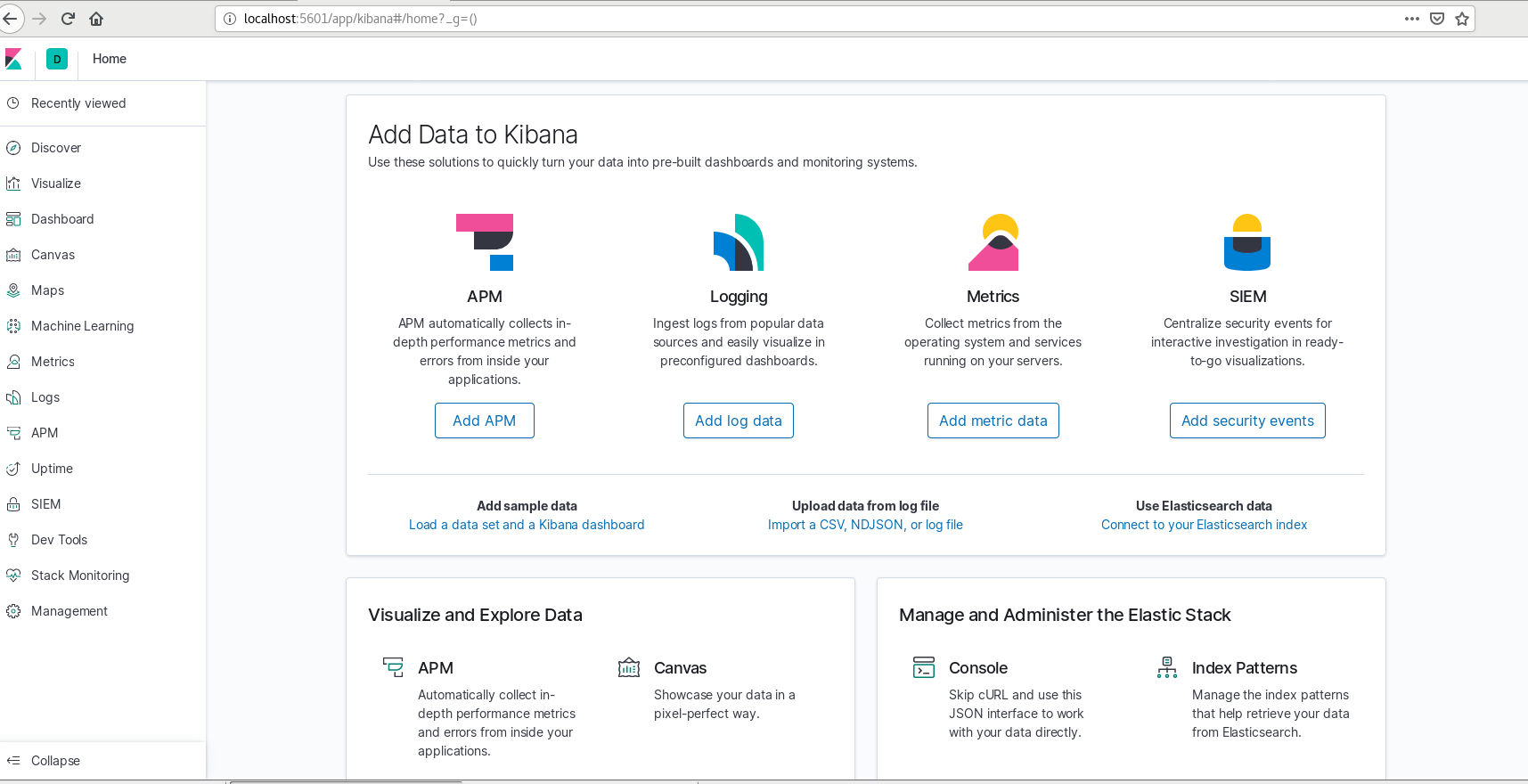
server.name: "localhost"

elasticsearch.url: "[http://localhost:9200](http://localhost:9200/)"

Change any settings as required for your purpose and execute below command to start kibana service

sudo systemctl enable --now kibana

Now access the kibana dashboard as <http://localhost:5601> or <http://your-ip:5601> (if server.host parameter is configured as “0.0.0.0”)



If you have firewall enabled then please add the tcp post 5601 to it , as follows

sudo firewall-cmd --add-port=5601/tcp --permanent

sudo firewall-cmd --reload

### Step 5 : Install Logstash

Logstash works as a centralised log server for all the logs that you collect.

Execute below command to install :

sudo yum -y install logstash

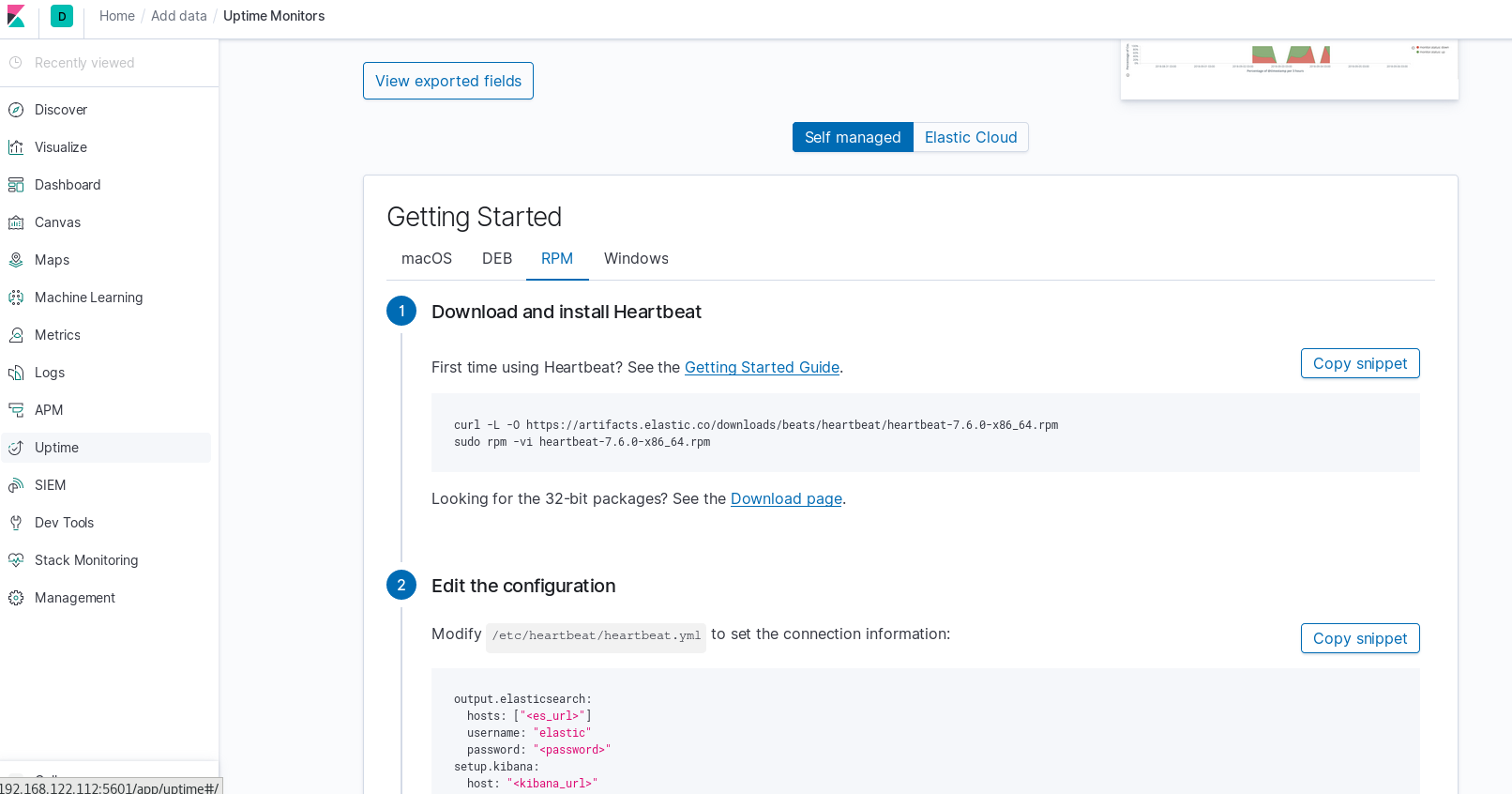
### 

### Step 6 : Install heartbeat and configure the urls for uptime monitoring

To install and configure heartbeat you can type the following url

<http://your_kibana_server_ip:5601/app/kibana#/home/tutorial/uptimeMonitors>

Or you can go to uptime monitors on your kibana app as show below



Once you have heartbeat on your system, edit the heartbeat.yml file in location /etc/heartbeat/

And edit the monitors as show below:

There are two ways of writing monitors, the 1st one is inline , inside heartbeat.yml 2nd is external by writing a yml file inside monitors.d directory. The content will be the same as show below:

# Configure monitors inline

Heartbeat.monitors:

* type: http

name: "elasticsearch"

# List or urls to query

urls: ["<http://localhost:9200>"] // here you can provide the IP and port of the server

// where your services are located instead of the localhost

# Configure task schedule

schedule: '@every 10s'

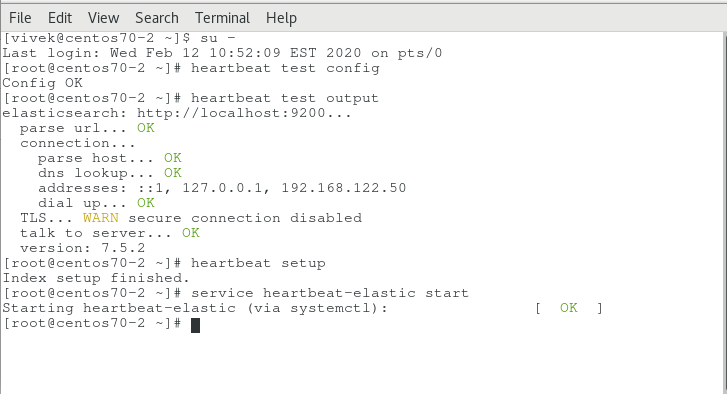
# Total test connection and data exchange timeout

timeout: 16s

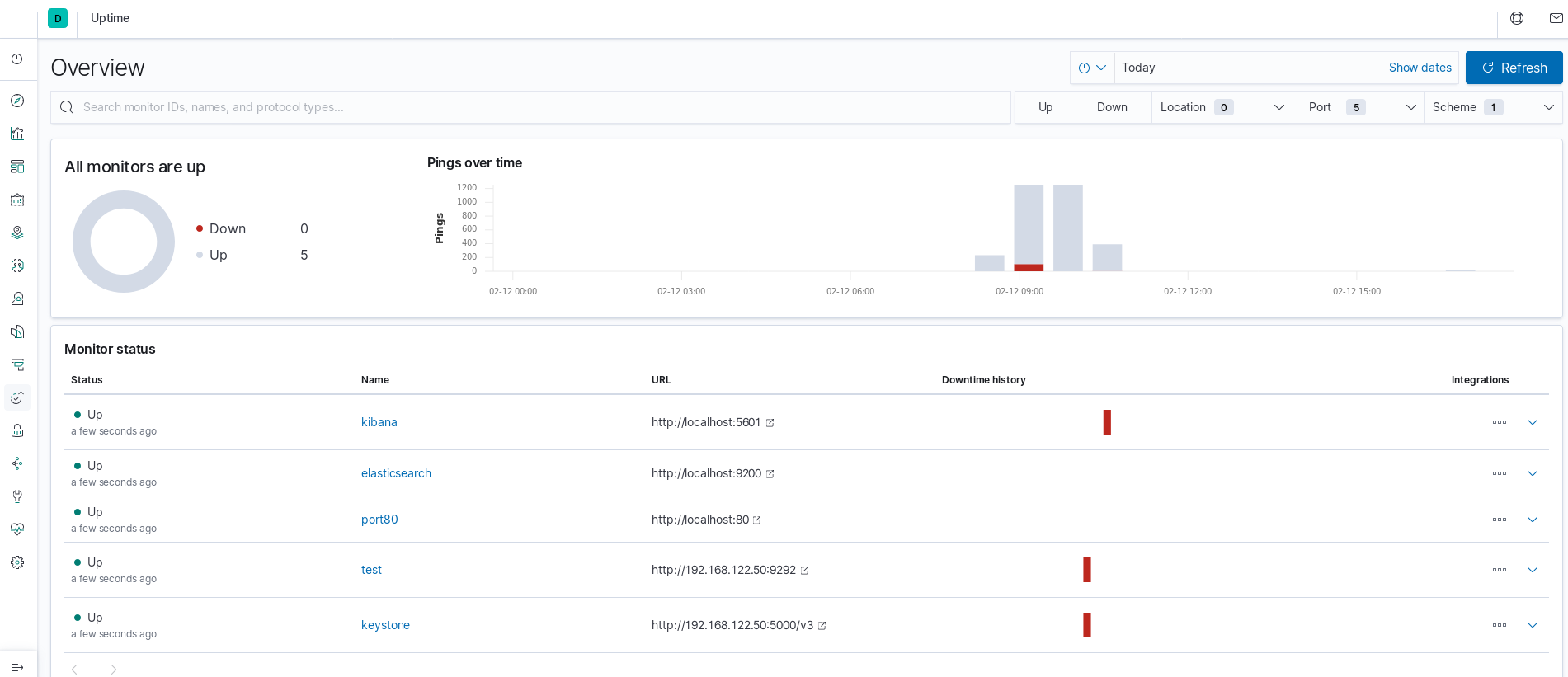
After that execute heartbeat test config command to verify the yaml syntax and the execute

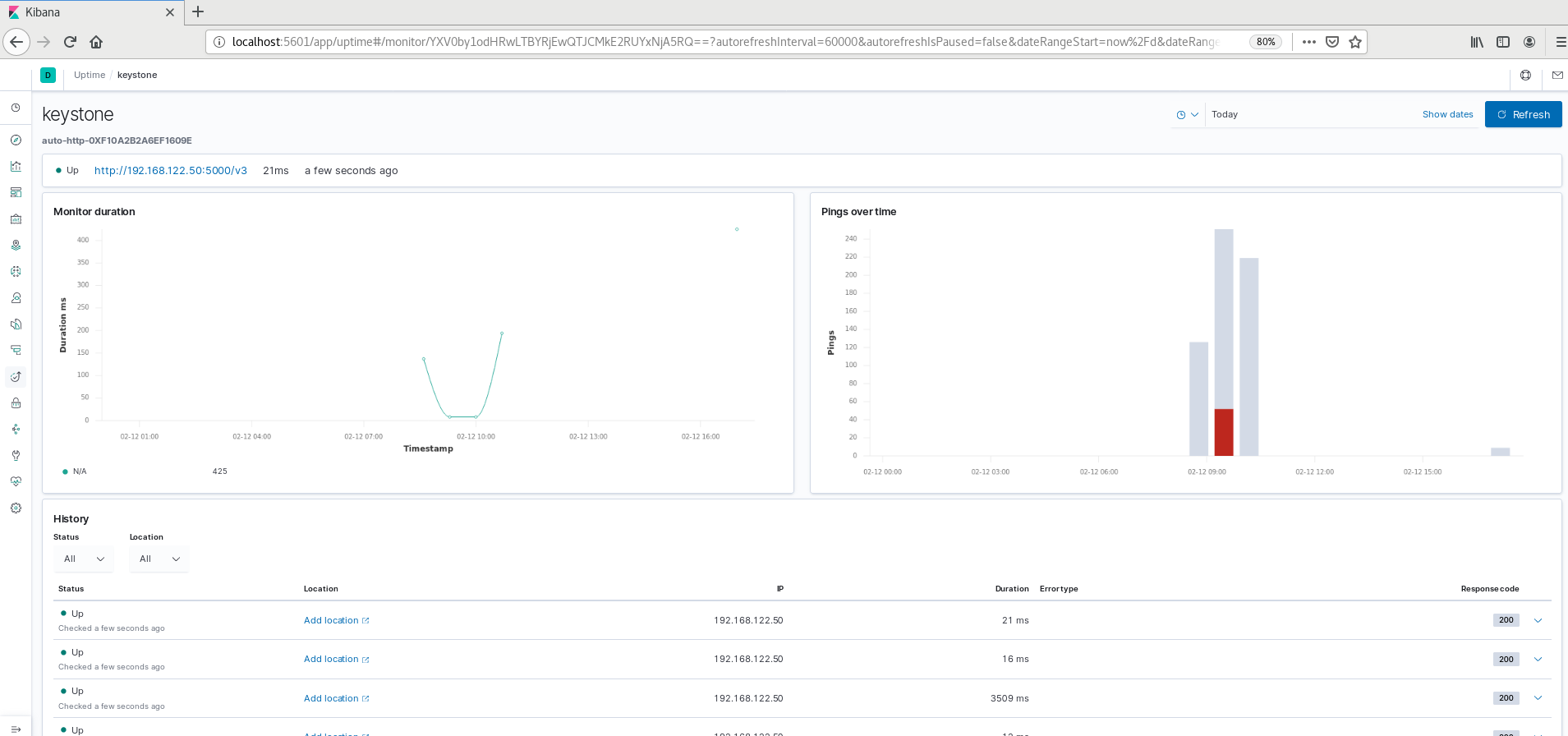
heartbeat test output command to test the config.

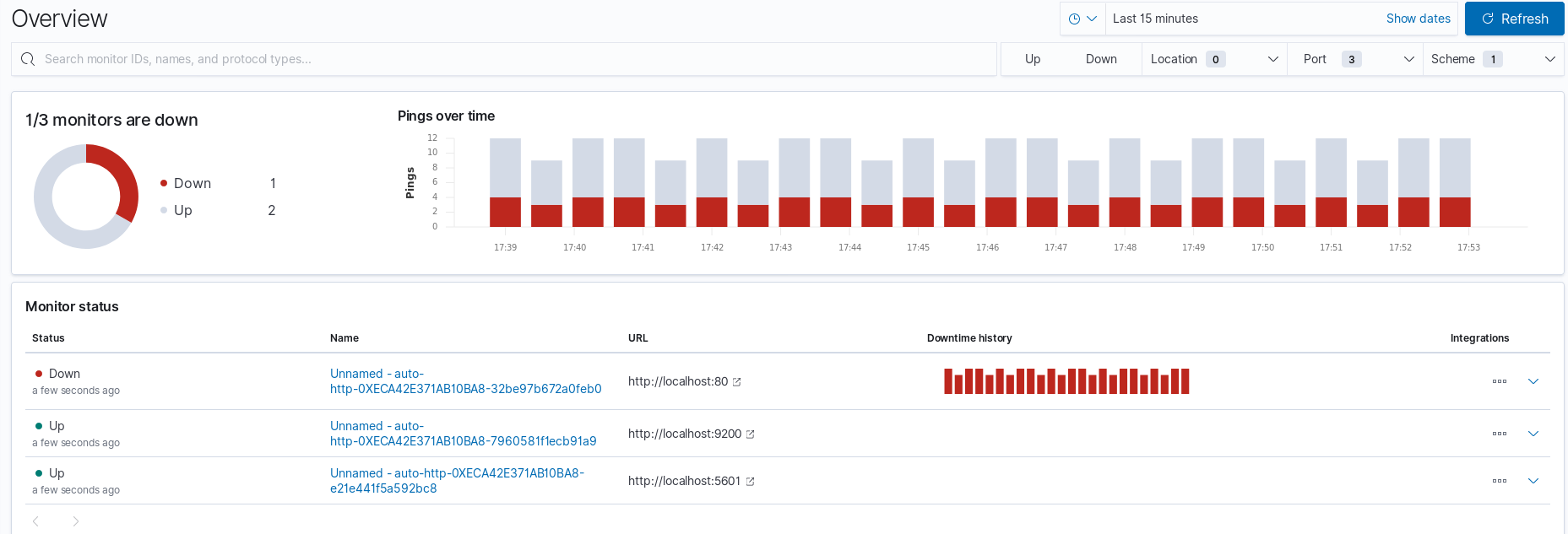
Execute heartbeat setup and service heartbeat-elastic start to run the heartbeat service as shown below.



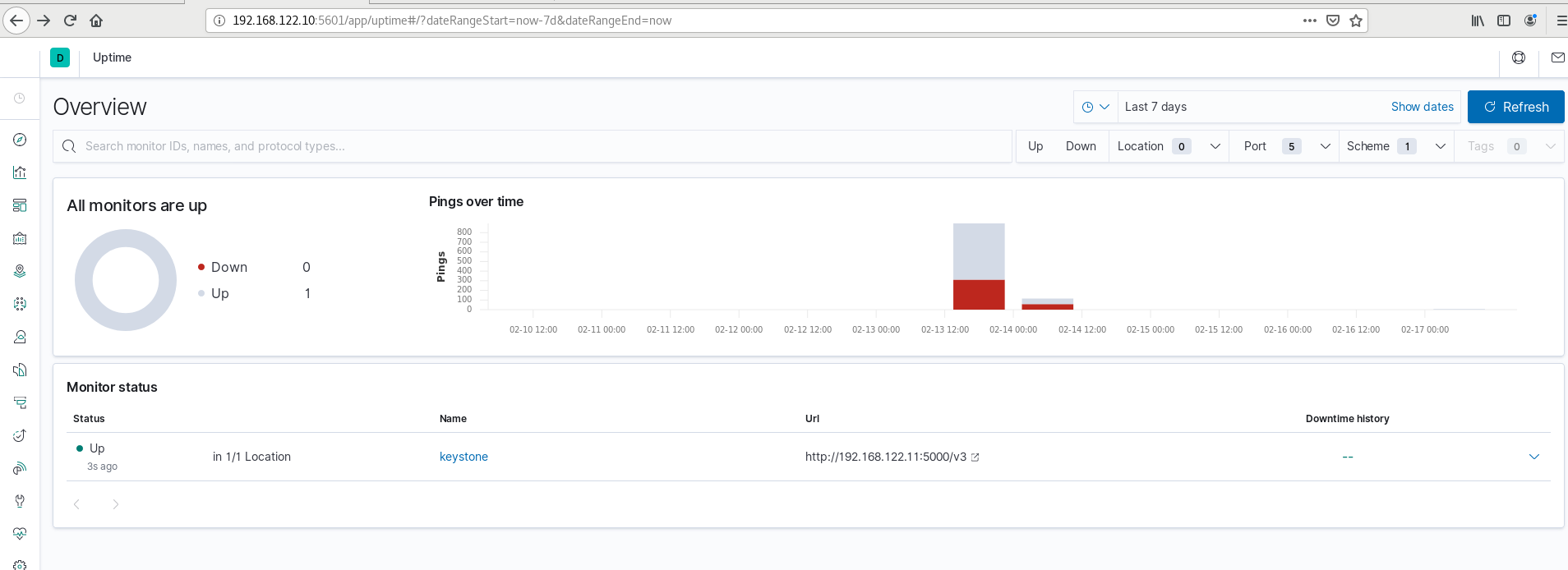
**Kibana Uptime dashboard after configuring the service endpoints**

****

****

****

**Keystone service running on a different VM**

****

**References**

1. <https://wiki.openstack.org/wiki/Cloudpulse>
2. <https://learning.oreilly.com/library/view/openstack-administration-with/9781787121638/ch10s03.html>
3. <https://www.elastic.co/blog/uptime-monitoring-with-heartbeat-and-the-elastic-stack>
4. <https://computingforgeeks.com/how-to-install-elk-stack-on-centos-fedora/>
5. <https://www.youtube.com/watch?v=NGfbcmQoVYc>