**ElasticSearch Cluster on AWS**

**Task Achieved**

Launch 3 EC2 instances (in free tier) and setup a cluster of 3 ElasticSearch nodes on it.

ElasticSearch is configured in such a way that requires credentials and provides encrypted communication the demonstration of the functioning is also shown.

The cross-cutting concerns addresses are security (encrypted communication and credentials required for ElasticSearch and also on the Ansible EC2 Server IAM role is assigned for EC2 access instead providing AWS Secret and Access Key), reusability (in the playbook, roles are designed to offer reusability) and cost optimization (use of free tier only)

Commands:

Run using commands:

ansible-playbook InstallAWS.yml

ansible-playbook InstallElasticSearch.yml

Pre-requisites

* Install Ansible on an AWS EC2 Server

sudo yum update -y

sudo yum install -y <https://dl.fedoraproject.org/pub/epel/epel-release-latest-7.noarch.rpm>

sudo yum install ansible -y

ansible –version

optionally create user for ansible eg ansadmin and setup passwordless SSH authenticated for it

and generate its keys

sudo yum install python-pip

pip install boto boto3 –user

pip install awscli –user

assign the EC2 server IAM role for EC2

Changes made

* In ansible.cfg file

set

host\_key\_checking = False

* Made a backup of the original host file

hosts.orig

* ec2.ini and ec2.py files taken from github

<https://github.com/ansible/ansible/tree/stable-2.9/contrib/inventory>

Approach to the Solution

**Approach**

We first launch 3 EC2 instance and use user data to ensure java and elasticsearch is already installed on it.

Then we install discovery-ec2 plugin on all the 3 instances and set bootstrap password for superuser elastic. This is done so that we ensure credentials are required for ElasticSearch. In jvm.options set heap space to 512m as we use t2.micro instance with 1g RAM to remain in free tier.

On one server we generate certificate for encrypted TCP and HTTP communications with in the node as well as communicating outside. This certificate is copied on all other nodes.

Ensuring elasticsearch.yml has configurations set for cluster, security and encrypted communication.

Starting elasticsearch service and demonstrating functionality using curl.

The are mainly two playbooks a) InstallAWS.yml and b)InstallElasticSearch.yml

InstallAWS.yml

---

- name: Launch of EC2 instances

hosts: localhost

connection: local

gather\_facts: false

become: yes

roles:

- role: installAWS

- role: forES

tags: forESonly

- role: sshAWS

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_-

The basic purpose of the InstallAWS.yml file is to launch 3 AWS EC2 instances in free tier.

role – installAWS :

Uses ansible module ec2 to launch EC2 instances.

The parameters for bringing up the EC2 instances such as ami-id, key-pair, security group, region, tag name have been passed through the vars/main.yml for the ansible role – installAWS.

User data ensures java and elasticsearch is installed on all the EC2 instances at boot time.

The security group used has the following inbound and outbound rules:

| **Type** | **Protocol** | **Port range** | **Destination** | **Description - optional** |
| --- | --- | --- | --- | --- |
| HTTP | TCP | 80 | 0.0.0.0/0 | - |
| HTTP | TCP | 80 | ::/0 | - |
| SSH | TCP | 22 | 0.0.0.0/0 | - |
| SSH | TCP | 22 | ::/0 | - |
| HTTPS | TCP | 443 | 0.0.0.0/0 | - |
| HTTPS | TCP | 443 | ::/0 | - |
| All ICMP - IPv4 | ICMP | All | 0.0.0.0/0 | - |
| All ICMP - IPv4 | ICMP | All | ::/0 | - |
| Custom TCP | TCP | 9200 - 9300 | 0.0.0.0/0 | - |
| Custom TCP | TCP | 9200 - 9300 | ::/0 | - |
| Custom TCP | TCP | 1024 - 65535 | 0.0.0.0/0 | - |
| Custom TCP | TCP | 1024 - 65535 | ::/0 | - |

These values can be modified hence making the script reusable.

forES role

It saves the private ip addresses of the 3 EC2 instances in variables using set\_fact module.

Adds private ip’s the newly launched instances in the hosts file and puts them under groups in the hosts file using template module. Template is in the template folder of the role.

hosts.j2

[master]

{{ ec2.instances[0].private\_ip }}

[node1]

{{ ec2.instances[1].private\_ip }}

[node2]

{{ ec2.instances[2].private\_ip }}

[deploy:children]

master

node1

node2

[deploy:vars]

ansible\_user=ec2-user

ansible\_ssh\_private\_key\_file=/etc/ansible/LocalKP.pem

It then replaces these ip’s in the elasticsearch.yml file template on the local machine for each node to form a cluster.

Example

cluster.name: demo-cluster

node.name: master

network.host: {{ master\_priIP }}

discovery.seed\_hosts: ["{{master\_priIP }}", "{{ node1\_priIP }}", "{{ node2\_priIP }}"]

cluster.initial\_master\_nodes: ["{{ master\_priIP }}"]

node.master: true

xpack.security.enabled: true

xpack.security.transport.ssl.enabled: true

xpack.security.transport.ssl.verification\_mode: certificate

xpack.security.transport.ssl.keystore.path: elastic-certificates.p12

xpack.security.transport.ssl.truststore.path: elastic-certificates.p12

http.host: 0.0.0.0 # accept request from remote

xpack.security.http.ssl.enabled: true

xpack.security.http.ssl.keystore.path: elastic-certificates.p12

xpack.security.http.ssl.truststore.path: elastic-certificates.p12

\*for reusability if we want to use InstallAWS.yml to just launch EC2 instances then we can modify the user data in vars/main.yml in ansible role – installAWS. And skip forES role \*

ansible-playbook InstallAWS.yml --skip-tags forESonly

sshAWS role waits until EC2 instances are ready to be SSH into and then it pauses again so that the instances pass the initial check.

InstallElasticSearch.yml

---

- name: Installing AWS plugin, set bootstrap password and replace JVM file

hosts: deploy

become: yes

gather\_facts: true

roles:

- configcluster

- name: Generate the Certificates on master node and fetch them to copy on all other nodes, replace the elasticsearch.yml file

hosts: master

become: yes

gather\_facts: true

roles:

- setmaster

- name: node1 instance replace elasticsearch.yml file and placing certificate file

hosts: node1

become: yes

gather\_facts: true

roles:

- setnode1

- name: node2 instance replace elasticsearch.yml file and placing certificate file

hosts: node2

become: yes

gather\_facts: true

roles:

- setnode2

- name: Demonstrate Functionality

hosts: deploy

become: yes

gather\_facts: true

roles:

- demo

Role: configcluster on all 3 nodes

Installing AWS discovery-ec2 plugin on all the three nodes

set bootstrap password as we want to ensure ElasticSearch requires credentials so it sets the password for superuser elastic as elastic. The script can be modified to change this password.

replace jvm.options file with a template to change minimum and maximum heap search to 512m as we use t2.micro to remain in free tier.

Role: setmaster

For encrypted communication at TCP and HTTP

Generate the Certificates on master node using certificate utility and copying the generated certificate to /etc/elasticsearch and setting appropriate permissions for it.

Using the fetch module copy the certificate file to the local machine so that it can be copied on all other nodes at later stages

replace the elasticsearch.yml file with the one formed in the forES role tasks having

network.host: <current node private IP>

Start elasticsearch service.

Role : setnode1 and setnode2

Copying the certificate generated from master node to the other two nodes and

replace the elasticsearch.yml file with the one formed in the forES role tasks having

network.host: <current node private IP>

Start elasticsearch service.

Role: demo

Ensure elasticsearch service is started on all 3 nodes.

Demonstrate security and encrypted communication is working using curl command

curl -k -uelastic:elastic <https://localhost:9200/_cluster/health?pretty>

If we do not provide credentials ie -uelastic:elastic it do not give desired output or if we provide wrong credentials it will not work.

Also if we use http instead of https it will not work.

OUTPUTS:

[ec2-user @ip-172-31-35-169 ~]#

**curl -k -uelastic:elastic https://localhost:9200/\_cluster/health?pretty**

{

"cluster\_name" : "demo-cluster",

"status" : "green",

"timed\_out" : false,

"number\_of\_nodes" : 3,

"number\_of\_data\_nodes" : 3,

"active\_primary\_shards" : 0,

"active\_shards" : 0,

"relocating\_shards" : 0,

"initializing\_shards" : 0,

"unassigned\_shards" : 0,

"delayed\_unassigned\_shards" : 0,

"number\_of\_pending\_tasks" : 0,

"number\_of\_in\_flight\_fetch" : 0,

"task\_max\_waiting\_in\_queue\_millis" : 0,

"active\_shards\_percent\_as\_number" : 100.0

}

WITHOUT CREDENTIALS :

[ec2-user@ip-172-31-35-169 ~]# **curl -k https://localhost:9200/\_cluster/health?pretty**

{

"error" : {

"root\_cause" : [

{

"type" : "security\_exception",

"reason" : "missing authentication credentials for REST request [/\_cluster/health?pretty]",

"header" : {

"WWW-Authenticate" : [

"Bearer realm=\"security\"",

"ApiKey",

"Basic realm=\"security\" charset=\"UTF-8\""

]

}

}

],

"type" : "security\_exception",

"reason" : "missing authentication credentials for REST request [/\_cluster/health?pretty]",

"header" : {

"WWW-Authenticate" : [

"Bearer realm=\"security\"",

"ApiKey",

"Basic realm=\"security\" charset=\"UTF-8\""

]

}

},

"status" : 401

}

WITHOUT HTTPS

[ec2-user@ip-172-31-35-169 ~]$ **curl -k -uelastic:elastic http://localhost:9200/\_cluster/health?pretty**

curl: (52) Empty reply from server

Demonstrating elasticsearch functionality:

[ec2-user@ip-172-31-35-169 ~]$ **curl -X POST -k -uelastic:elastic "https://localhost:9200/people/person/1" -H 'Content-Type: application/json' -d '{"Fname":"John","Lname":"Smith","Job":"Administrator"}'**

{"\_index":"people","\_type":"person","\_id":"1","\_version":1,"result":"created","\_shards":{"total":2,"successful":1,"failed":0},"\_seq\_no":0,"\_primary\_term":1}

[ec2-user@ip-172-31-35-169 ~]$ **curl -X POST -k -uelastic:elastic "https://localhost:9200/people/person/2" -H 'Content-Type: application/json' -d '{"Fname":"Alice","Lname":"Stevenson","Job":"Manager"}'**

{"\_index":"people","\_type":"person","\_id":"2","\_version":1,"result":"created","\_shards":{"total":2,"successful":2,"failed":0},"\_seq\_no":1,"\_primary\_term":1}[ec2-user@ip-172-31-35-169 ~]

[ec2-user@ip-172-31-35-169 ~]$ **curl -X POST -k -uelastic:elastic "https://localhost:9200/people/person/3" -H 'Content-Type: application/json' -d '{"Fname":"Leila","Lname":"thomson","Job":"Director"}'**

[ec2-user@ip-172-31-35-169 ~]$ **curl -XGET -k -uelastic:elastic 'https://localhost:9200/\_search?q=\*'**

{"took":679,"timed\_out":false,"\_shards":{"total":1,"successful":1,"skipped":0,"failed":0},"hits":{"total":{"value":3,"relation":"eq"},"max\_score":1.0,"hits":[{"\_index":"people","\_type":"person","\_id":"1","\_score":1.0,"\_source":{"Fname":"John","Lname":"Smith","Job":"Administrator"}},{"\_index":"people","\_type":"person","\_id":"2","\_score":1.0,"\_source":{"Fname":"Alice","Lname":"Stevenson","Job":"Manager"}},{"\_index":"people","\_type":"person","\_id":"3","\_score":1.0,"\_source":{"Fname":"Leila","Lname":"thomson","Job":"Director"}}]}}

[ec2-user@ip-172-31-35-169 ~]$ **curl -XGET -k -uelastic:elastic 'https://localhost:9200/\_search?q=leila'**

{"took":486,"timed\_out":false,"\_shards":{"total":1,"successful":1,"skipped":0,"failed":0},"hits":{"total":{"value":1,"relation":"eq"},"max\_score":0.9808292,"hits":[{"\_index":"people","\_type":"person","\_id":"3","\_score":0.9808292,"\_source":{"Fname":"Leila","Lname":"thomson","Job":"Director"}}]}}

* Resources:

ElasticSearch Guide : <https://www.elastic.co/guide/index.html>

AWS official documentation

Ansible documentation for modules and other information.

* Only AWS free tier used.
* ElasticSearch access and communication is secure.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

What did you choose to automate the provisioning and bootstrapping of the instance? Why?

To provision the instances used AWS EC2 service in free tier as it is easy to provision, ansible has modules to do this task and also it is free of charge and provides all functionalities required. Bootstraping the instances used user data while provisioning the instances so that the instance softwares are up to date, java and elasticsearch is installed on them and they are ready for further configurations.

How did you choose to secure ElasticSearch? Why?

ElasticSearch was made secure as it required credential and all communication was encrypted.

Without giving credentials or with wrong credentials we get error as demonstrated in the output section.

type" : "security\_exception",

"reason" : "missing authentication credentials for REST request [/\_cluster/health?pretty]"

This can be done by using x pack provided by elasticsearch.

Configurations have to be set in elasticsearch.yml

xpack.security.enabled: true

For encrypted communication we need to generate certificate and Configurations have to be set in elasticsearch.yml

For example:

xpack.security.transport.ssl.enabled: true

xpack.security.transport.ssl.verification\_mode: certificate

How would you monitor this instance? What metrics would you monitor?

AWS CloudWatch could be used to monitor the instances.

Using default metrics like CPUUtilization and Status Checks and creating custom metrics to monitor RAM Utilizations

 Could you extend your solution to replace a running ElasticSearch instance with little or no downtime? How?

AWS AutoScaling can be used to ensure that a minimum number of EC2 server as ElasticSearch nodes are always running. The Amazon machine image (AMI) can be a custom image we create of a elasticsearch node.

Was it a priority to make your code well structured, extensible, and reusable?

Yes. Focus was given to make the code well structured so that it is understandable and names of tasks are self-descriptive, extensible and also attempted to make the code reusable using ansible roles.