# PART I: Concepts and Theory, Algorithms [60 points]

1. Watch this tutorial and make notes:

Then, explain the 12 to 15 specific STEPS you learned from this tutorial which are useful for you to build the complete eCommerce SaaS web app you desiegned earlier. Follw the exact steps detailed in this particular tutorial and expalin how you'd use them to you build your eCommerce App SaaS.

For the given link, it shows the first tier (the Presentation tier) of 3 tiers of the web application.

1. Set up the bundles (using PowerShell: npx create-react-app demo-app)

2. Launch the app (Powershell: npm start)

3. Create an optimize production build (PowerShell: npm run build)

4. Create S3 buckets on console (www and non-www)

5. Upload files from the build folder

6. Edit bucket policy

7. Edit static website hosting

8. Create record for Route 53

9. Set up the certificate (for https enabled)

9. Set up the CloudFront distribution

10. Change DNS settings

11. Test and Check everything

12. Show the changes for redeployment

And following the given link, I've expanded a little bit on the scope of the requirements in this question. The other links show the process for the second tier (the App tier) using serverless with DynamoDB and S3, and for the third tier (the Data tier) using RDS. With these process, there are still some steps for configuration and IAM setting.

For my eCommerce SaaS web application, I would use S3, CloudFront, ACM, Route53 as the video shows. The video actually shows the process for how to properly deploy a static web site on AWS. It contains some useful methods which could help the website better such as redirection, https encryption, distributions, domain records and something others. Using these techniques, what I

only need to do for my eCommerce SaaS web application frontend is to write the eCommerce html website on my own computer.

2. Study the book chapter on Scalability of Web Apps

Scalable and High-Performance Web Applications.pdf

Then, provide a basic design showing scalability calculations for your 3 tier eCommerce Web app SaaS. What are the key metrics of scalability for this SaaS and how can you achive scalability. Provide a detailed 1-2 page design with calcs and a diagram.

*Provide a basic design showing scalability calculations for your 3 tier eCommerce Web app SaaS*

*How to define scalability*

Informally, engineers describe the challenge of dealing with large audiences and high demand as a problem of scalability. More specifically, a Web application can scale if it continues to be available and functional at consistent speeds as the number of users and requests continues to grow, even to very high numbers.

*The key metrics of scalability for this SaaS*

• Throughput— the rate at which transactions are processed by the system

• Resource usage – the usage levels for the various resources involved (CPU, memory, disk, bandwidth)

• Cost— the price per transaction

*How to achieve scalability*

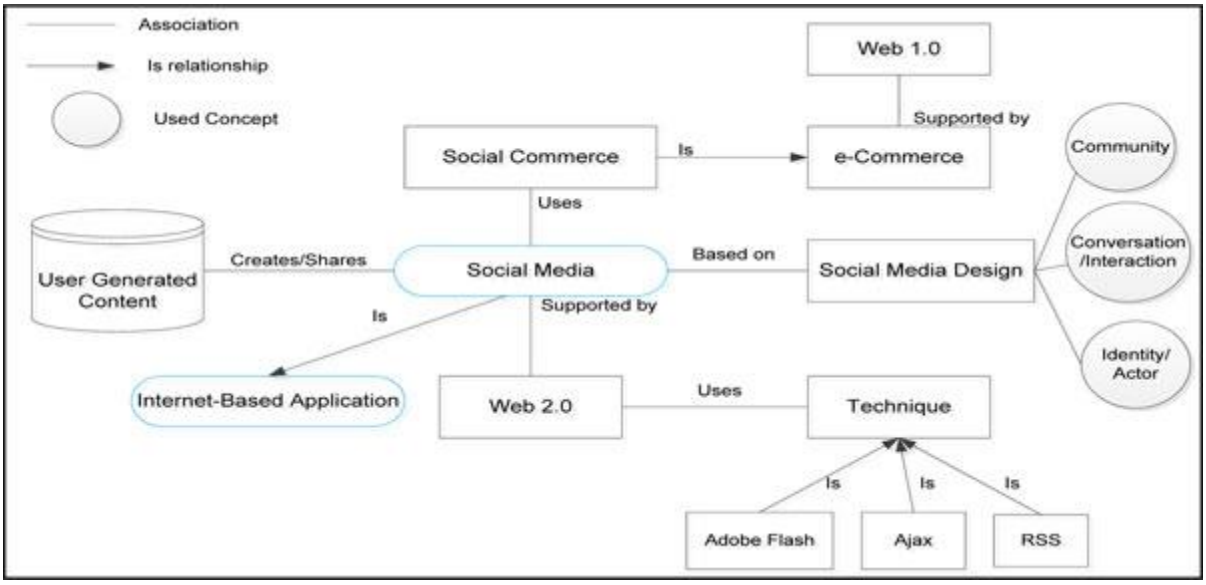
There are three techniques can be employed to achieve scalability

• Increase the resources (bigger machines, more disk, more memory).

• Improve the software.

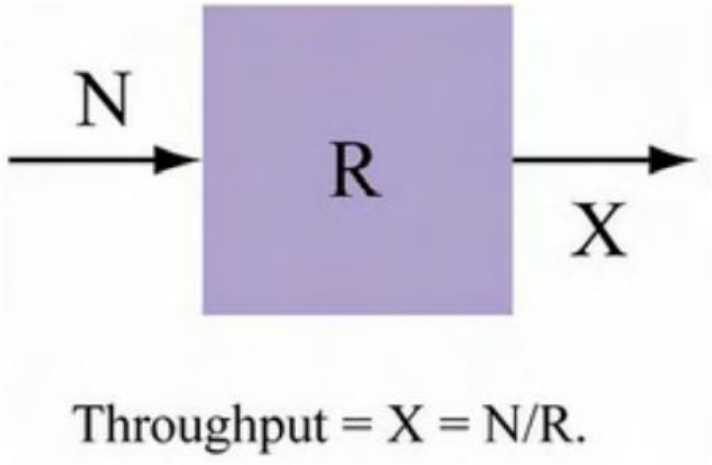
• Increase the resources and improve the software.

*Design with calcs and a diagram*




*scalability calculations*

Little's law: if this box contains an average of N users, and the average user spends R seconds in that box, then the throughput X of that box is roughly



For example,

Concurrent Users(N) is 10, Average Response Time(R) is 10 ms,

then the Throughput = N/R = 10 tps

Concurrent Users(N) is 100, Average Response Time(R) is 1200 ms,

then the Throughput = N/R = 83.333 tps

3. What is DevOps? Why do you need it for the scalability growth of your eCommerce Web app SaaS?

Explain a CI/CD with DevOps design you would follow within AWS to enable this for your eCom app.

List any 4 Devops tools we discussed in class (see references). How can you enable CI/CD nd devOps for your App using these instaed of AWS? Provide a basic design.
https://www.eginnovations.com/blog/top-devops-tools/

*What is DevOps*

DevOps is the combination of cultural philosophies, practices, and tools that increases an organization's ability to deliver applications and services at high velocity: evolving and improving products at a faster pace than organizations using traditional software development and infrastructure management processes.

A typical DevOps process consists of 8 stages: plan, code, build, test, release, deploy, operate and monitor. Adoption of new techniques, better tools, and improved collaboration methods continue to be on the rise in the DevOps universe.

*Why do you need it for the scalability growth of your eCommerce Web app SaaS?*

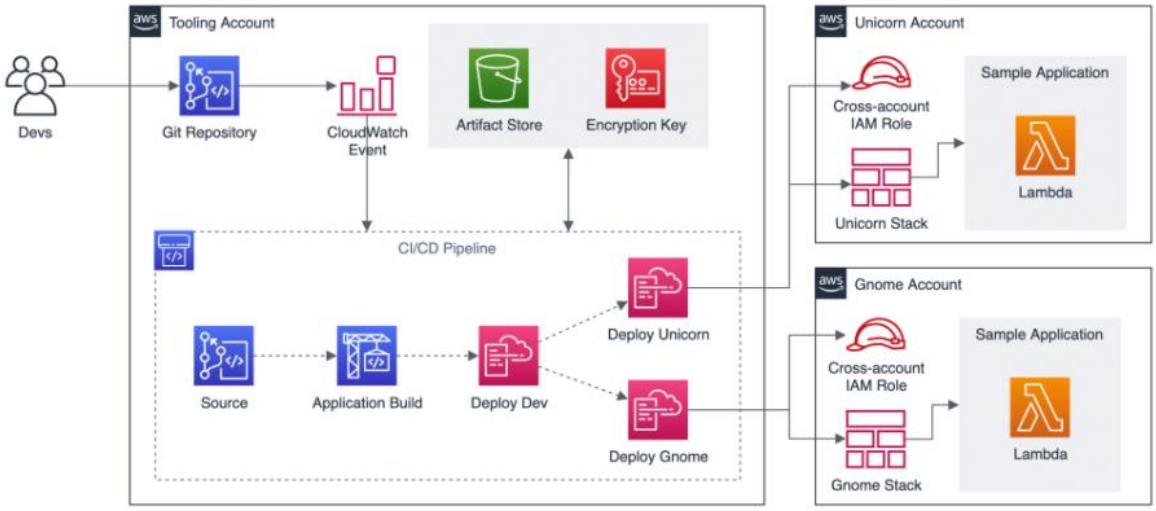
For my eCommerce SaaS Web Application, there are lots of benefits for using DevOps.

The first one is speed. The DevOps model enables move at high velocity so I can innovate for customers faster, adapt to changing markets better, and grow more efficient at driving business results. The DevOps model enables the developers and operations teams to achieve these results.

For this question - the scalability, The DevOps model enables operate and manage the infrastructure and development processes at scale. Automation and consistency help manage complex or changing systems efficiently and with reduced risk.

For example, infrastructure as code helps manage the development, testing, and production environments in a repeatable and more efficient manner.

*A CI/CD with DevOps design*




For my eCommerce SaaS web application, I consider a fictitious single-tenant ISV with two customers: Unicorn and Gnome to demonstrate the solution. It uses one central account where the tools reside (Tooling account), and two other accounts, each representing a tenant (Unicorn and Gnome accounts). As depicted in the following architecture diagram, when a developer pushes code changes to CodeCommit, Amazon CloudWatch Events  triggers the CodePipeline CI/CD pipeline, which automatically deploys a new version on each tenant's AWS account. It ensures that the fictitious ISV doesn't have the operational burden to manually re-deploy the same version for each end-customers.

*List any 4 Devops tools:*

1. Git
2. Docker
3. Kubernetes
4. Jenkins

*How can you enable CI/CD devOps for your App using these instaed of AWS?*

Git is a widely used DevOps tool across the software industry. Git is very easy to use and maintain version control artifacts. It's like AWS service 'CodeCommit'.

Docker is a forerunner in containerization. It automates application deployment and provides integrated security along with agile operations for legacy and cloud-native applications. The most significant advantage of Docker is that it efficiently separates apps into containers to make them more secure and transferable. Docker helps in stimulating massive changes in delivery workflows.

Whereas Kubernetes is the most popular open source container orchestration platform. It automates the deployment, management, scaling, networking, and availability of container-based applications. Deployment automation is the biggest benefit of Kubernetes in the DevOps world.

Using Docker and Kubernetes, I can replace some AWS services such as EKS, container and EC2.

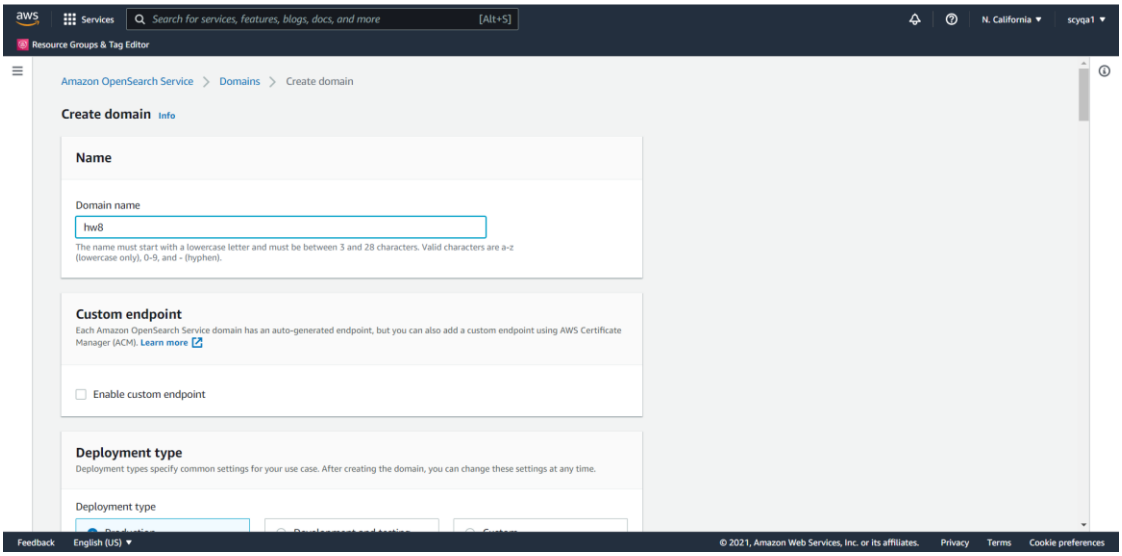
Jenkins is an open source solution for continuous integration that orchestrates and automates sequence of actions enabling developers to reliably build, test, and deploy their software.

Jenkins is an open-source automation server that integrates with a number of AWS Services, such as AWS CodeCommit, AWS CodeDeploy, Amazon EC2 Spot, and Amazon EC2 Fleet.

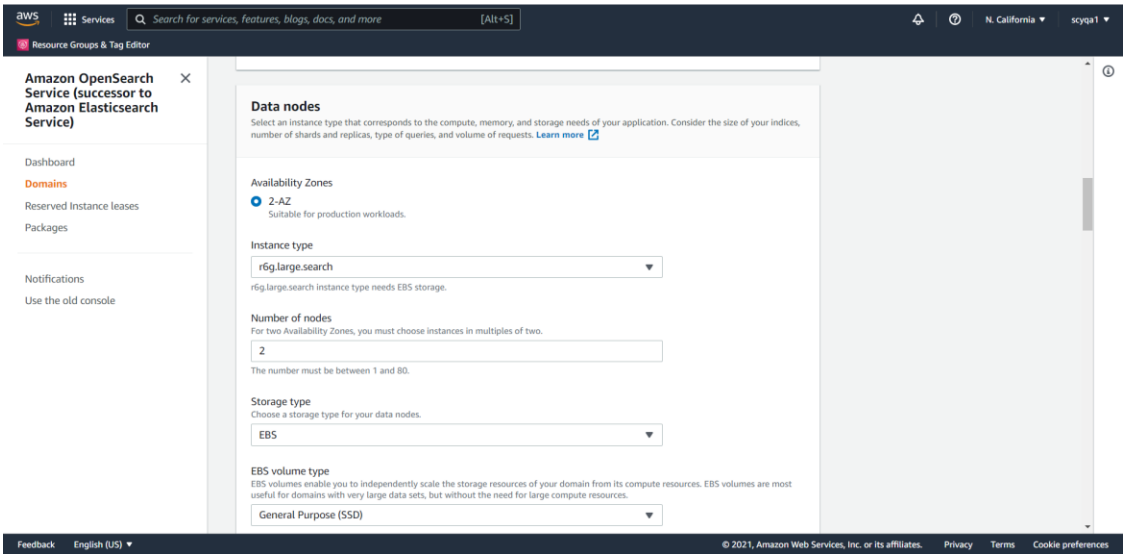
# PART II: LAB [40 points]

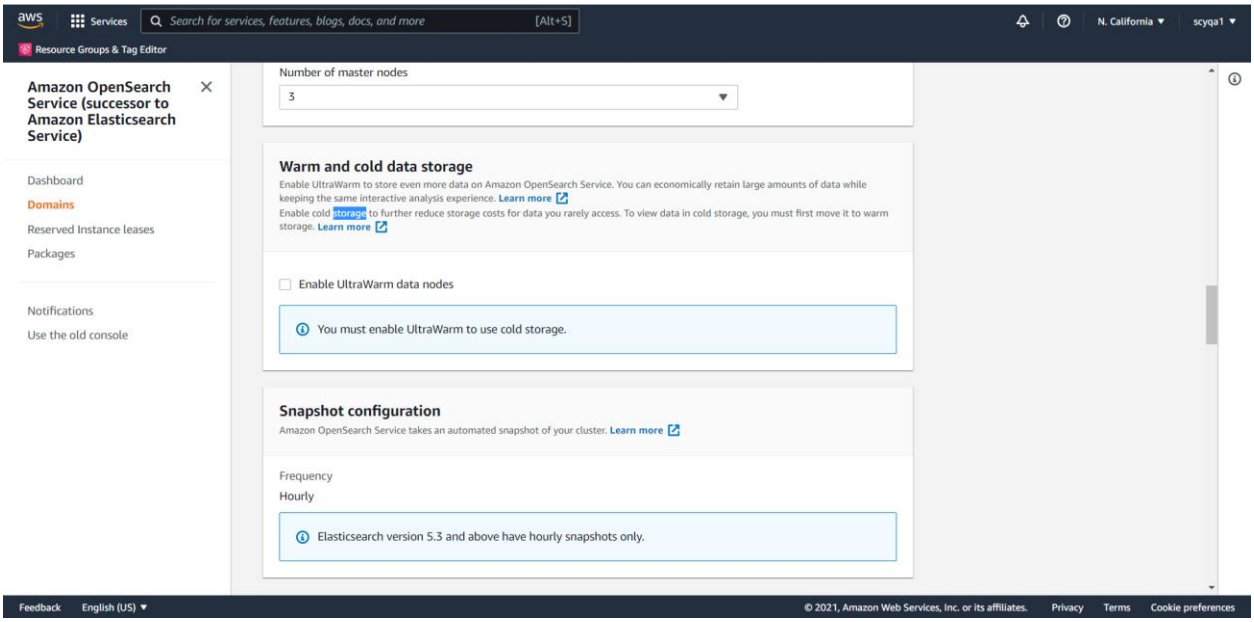
1. Following the examples in Chapter 10 (page 554) section on AWS Elasticsearch service. Then, setup a basic Elasticsearch on AWS and show serach and results. It is now called Amazon OpenSearch Service (successor to Amazon Elasticsearch Service). See the tutorials and Book's exmple to complete this. Clusters etc is not necessary, a basic install and Search use case is sufficient.

Create Elasticsearch domain

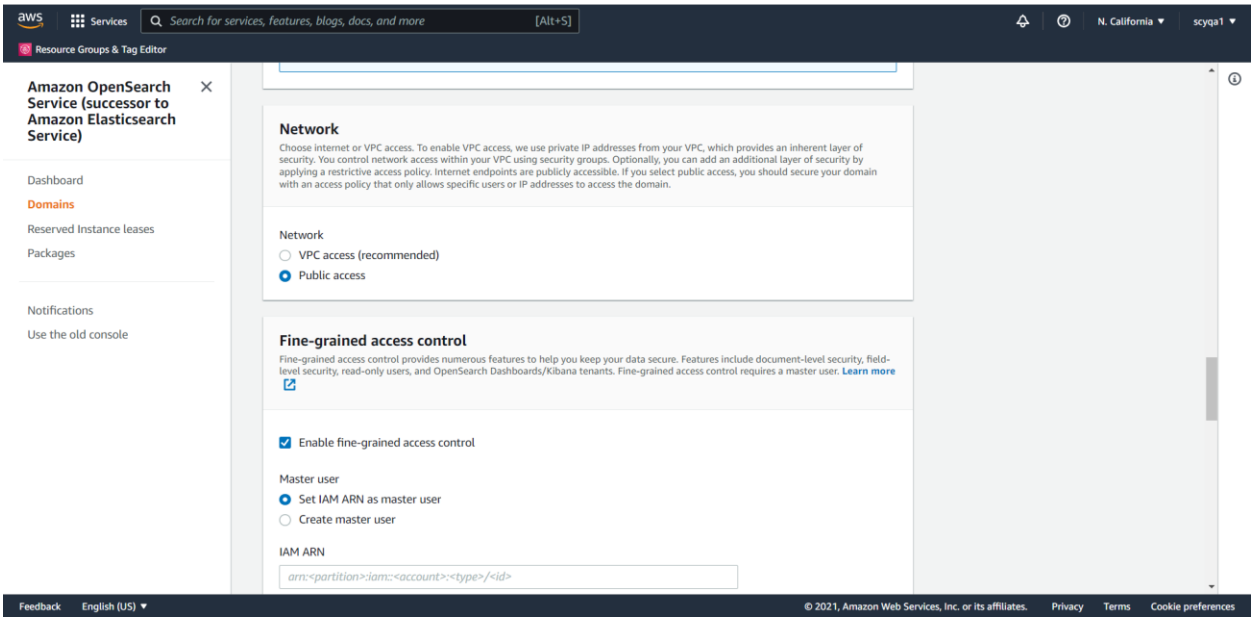



Node configuration and storage configuration

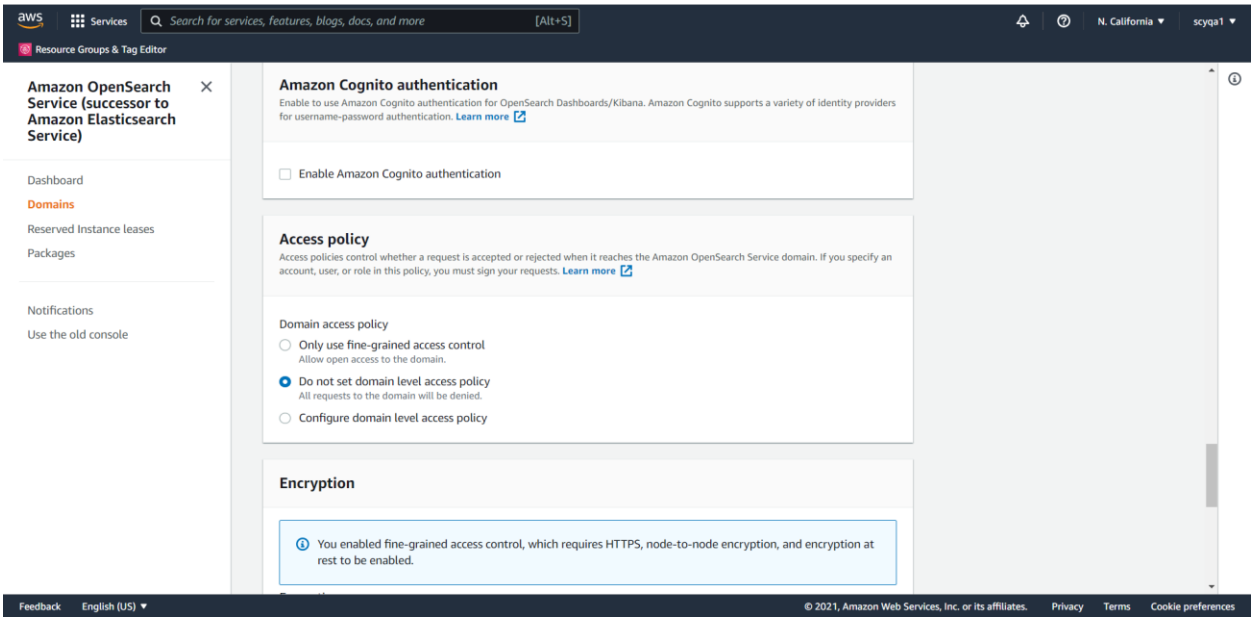




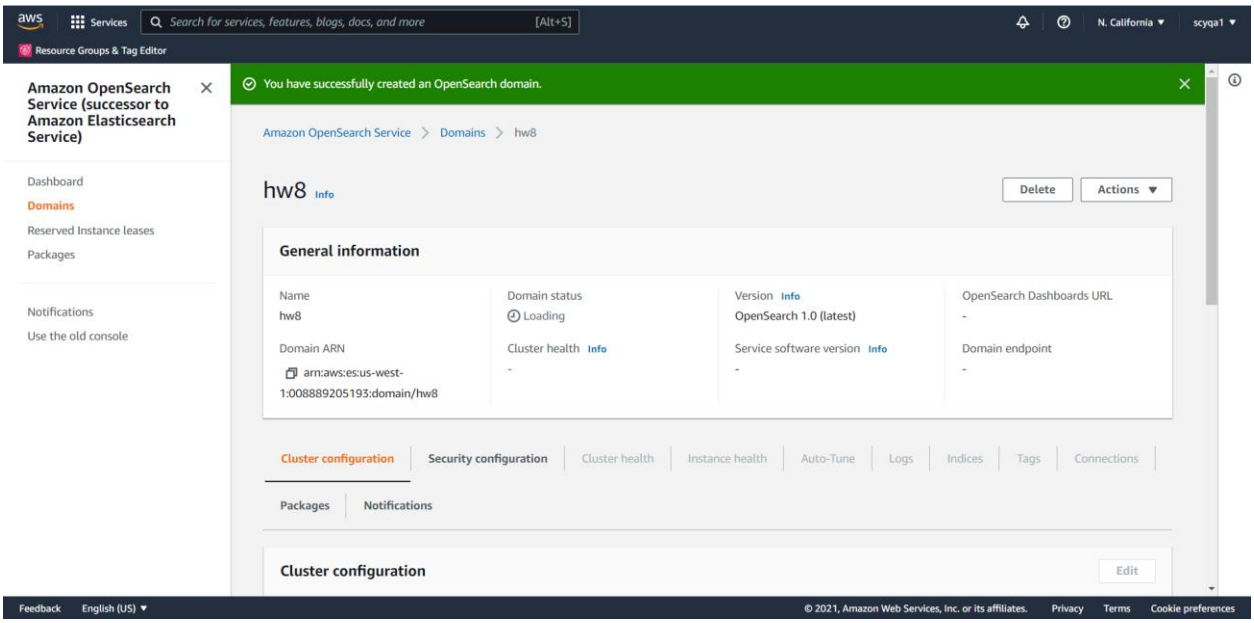

Network configuration



Access policy




Result



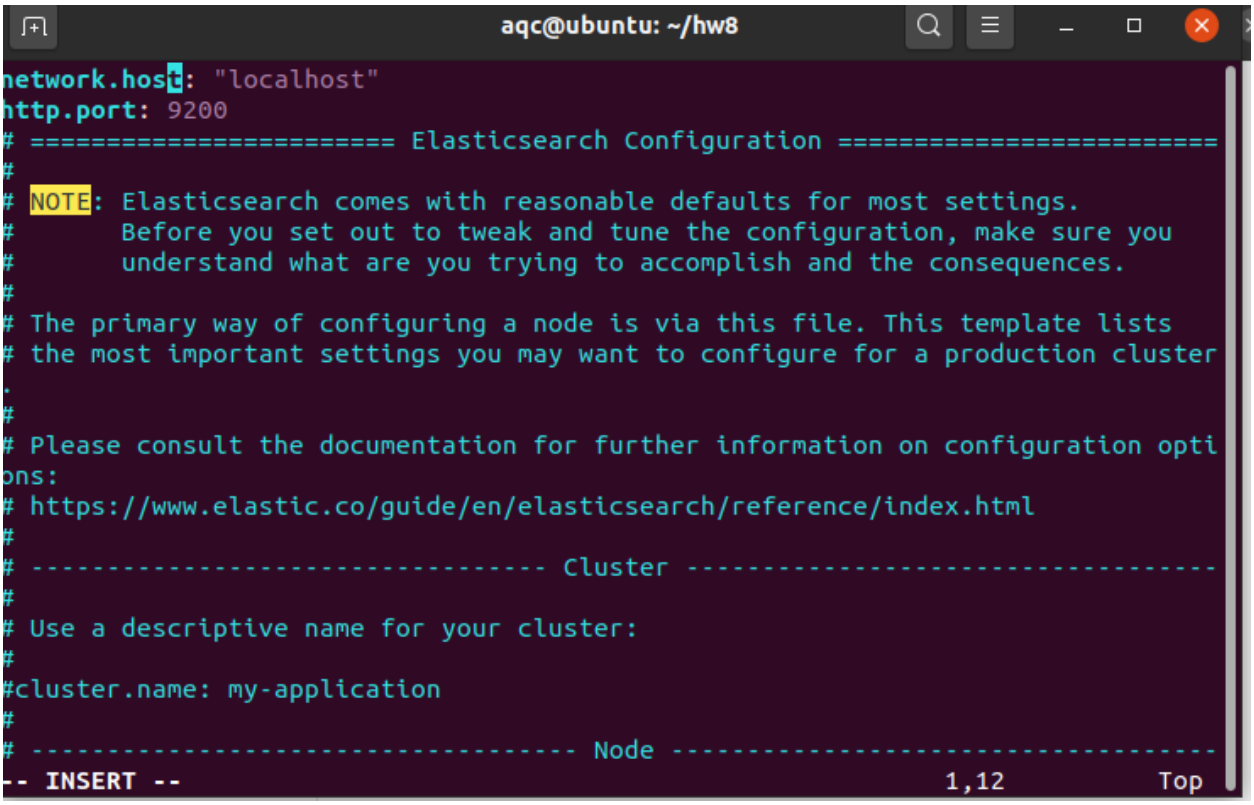
Install Elasticsearch

```
(base) aqc@ubuntu:~/hw8$ sudo apt-get install apt-transport-https
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following NEW packages will be installed:
  apt-transport-https
0 upgraded, 1 newly installed, 0 to remove and 324 not upgraded.
Need to get 4,680 B of archives.
After this operation, 162 kB of additional disk space will be used.
Get:1 http://us.archive.ubuntu.com/ubuntu focal-updates/universe amd64 apt-trans
port-https all 2.0.6 [4,680 B]
Fetched 4,680 B in 0s (15.4 kB/s)
Selecting previously unselected package apt-transport-https.
(Reading database ... 190567 files and directories currently installed.)
Preparing to unpack .../apt-transport-https_2.0.6_all.deb ...
Unpacking apt-transport-https (2.0.6) ...
Setting up apt-transport-https (2.0.6) ...
```

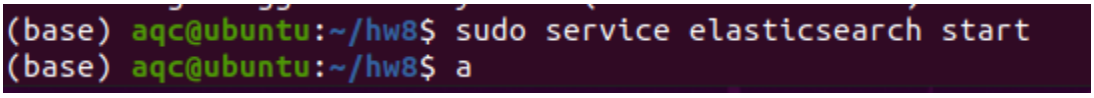
```
(base) aqc@ubuntu:~/hw8$ sudo apt-get install elasticsearch
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following NEW packages will be installed:
  elasticsearch
0 upgraded, 1 newly installed, 0 to remove and 341 not upgraded.
Need to get 149 MB of archives.
After this operation, 240 MB of additional disk space will be used.
Get:1 https://artifacts.elastic.co/packages/6.x/apt stable/main amd64 elasticsea
rch all 6.8.20 [149 MB]
12% [1 elasticsearch 21.9 MB/149 MB 15%][a
```

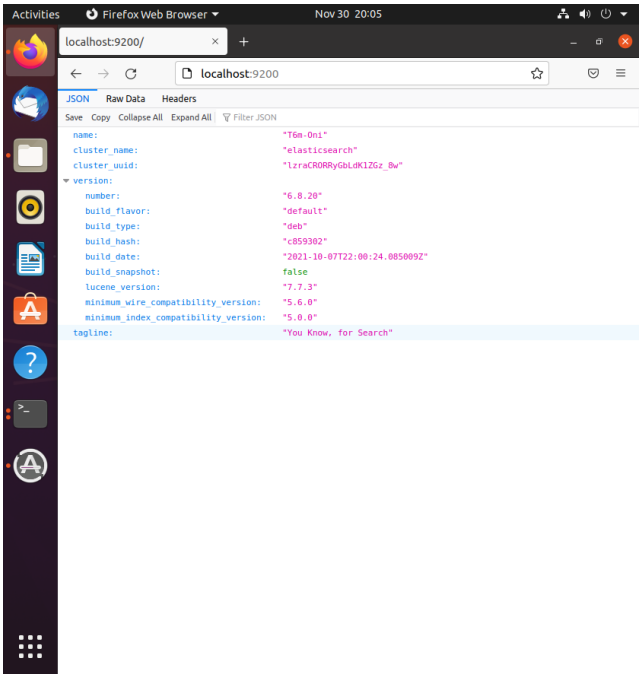
Configure Elasticsearch

```
(base) aqc@ubuntu:~/hw8$ vim /etc/elasticsearch/elasticsearch.yml
```




Start Elasticsearch



REST API

Create an Elasticsearch Index with response

```
(base) aqc@ubuntu:~/hw8$ curl -XPOST 'localhost:9200/logs/my_app' -H 'Cont
ent-Type: application/json' -d'
> {
> "timestamp": "2018-01-24 12:34:56",
> "message": "User logged in",
> "user_id": 4,
> "admin": false
> }
> '
{"_index":"logs","_type":"my_app","_id":"AqpCdH0BpAoYcH35HUcH","_version":
1,"result":"created","_shards":{"total":2,"successful":1,"failed":0},"_seq
```

```
_no":0,"_primary_term":1}(base) aqc@ubuntu:~/hw8$ curl -X PUT 'localhost:9
200/app/users/4' -H 'Content-Type: application/json' -d '
> {
>   "id": 4,
>   "username": "john",
>   "last_login": "2018-01-25 12:34:56"
> }
> '
{"_index":"app","_type":"users","_id":"4","_version":1,"result":"created",
"_shards":{"total":2,"successful":1,"failed":0},"_seq_no":0,"_primary_term
":1}(base) aqc@ubuntu:~/hw8$
```

Search with response

```
":1}(base) aqc@ubuntu:~/hw8$ curl -XGET 'localhost:9200/app/users/4?pretty
'
{
  "_index" : "app",
  "_type" : "users",
  "_id" : "4",
  "_version" : 1,
  "_seq_no" : 0,
  "_primary_term" : 1,
  "found" : true,
  "_source" : {
    "id" : 4,
    "username" : "john",
    "last_login" : "2018-01-25 12:34:56"
  }
}
(base) aqc@ubuntu:~/hw8$
```