Task 3. Search Engine and Deployment in Cluster

Javier García, Jesús Matos, Liam Mahmud, Krish Sadhwani

January 3, 2023

Abstract

Search engine project implementation and the idea of Load Balancing in modern day usage.

1 Introduction

Considering the previous tasks of manifesting the idea of an inverted index and with it a crawler in an application, the next step is to advance more into this topic and dive deep into a real world scenario when it comes to Big Data.

2 Task Statement

The idea is to develop a Search Engine that will have an API functionality, to make consultations and queries from different clients. The Indexer and Crawler modules will be connected and the data received will be stored in a datamart and datalake respectively.

But, the application as such will present a problem, which is scalability and availability. As more users utilize the program, the inefficiency grows. For this the engine will be presented in an image, ready to be deployed into a docker container, so that each system is able to execute its own search engine independently.

Finally to distribute the load, a balancer will be inserted. Which is crucial in modern day applications, the next section will speak more over this topic.

3 Why Load Balancing?

This tool is fundamentally designed to distribute traffic among different resources and avoid overload of a single resource. There are several criterias it considers, being such as; less busy, most healthy, located in a specific region..etc which helps route the traffic.[IBM]

3.1 Brief History

Load balancing came to prominence in the 1990s as hardware appliances came about distributing traffic across a network. As internet technologies and connectivity improved rapidly, web applications became more complex and their demands exceeded the capabilities of individual servers. There was a need to find better ways to take multiple requests for similar resources and distribute them effectively across servers. This was the genesis of load balancers. [Net]

Since load balancing allowed web applications to avoid relying on individual servers, it also helped in scaling these applications easily beyond what a single server could support. Soon, there were other functionalities that evolved, including the ability to provide continuous health checks, intelligent distribution based on the application's content and other specialized functions in the modern era.

3.2 It is especially important in cloud based applications

The main goal of preventing a specific server getting overloaded is crucial in any cloud environment. So that all incoming traffic gets is dealt in a coordinated fashion. If not, scalability will be a major issue, not being able to manage high workloads from a wide range of clients.

Thousands of users have accessed a website at a particular time. It is challenging for applications to manage the load that comes from all these requests all at once. Sometimes, it may result in a breakdown of your entire system.

For this, high traffic websites requires highly efficient load balancing for a smooth operation of their business. Load balancing helps in maintaining system firmness, performance and protection against system failures.[1]

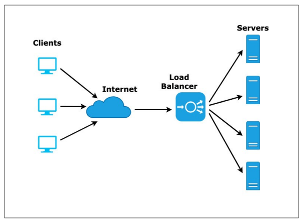


Figure 1: Load Balancing Architecture

3.3 Combining Clustering and Load Balancing is Powerful

Splitting the tasks upon different nodes and introducing a mechanism to maintain a healthy load between the different servers is a good way to increase the overall performance and make it available at a large scale. This is why many large and financially well established entities in current times are taking advantage of these tools.

3.4 How does it work?

Load in load balancing refers not only to website traffic but also comprises of memory capacity, network and CPU load on the server. The primary function of load balancing technique is to ensure that each system of the network is equipped with the same amount of work. It means neither of the system goes overloaded or underutilized.

The load balancer equally distributes the data depending on how busy the server is. Without load balancer, the client would wait long to process their data that might be frustrating for them.

During this load balancing process, information like job arrival rate and CPU processing rate are exchanged among the processors. Any failures in the application of load balancers can head to some severe consequences such as data loss.

Various companies use different load balancers along with multiple load balancing algorithms. One of the most commonly used methods or algorithms is the "Round Robin" load balancing.

3.5 Insight on Round Robin Search

Round-robin load balancing is one of the simplest methods for distributing client requests across a group of servers. Going down the list of servers in the group, the round-robin load balancer forwards a client request to each server in turn. When it reaches the end of the list, the load balancer loops back and goes down the list again (sends the next request to the first listed server, the one after that to the second server, and so on).

It is the simplest algorithm to implement, it cycles through your targets in order, so each target should receive an equal share of requests.[2]

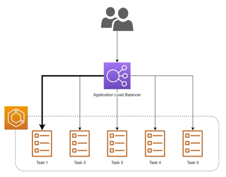
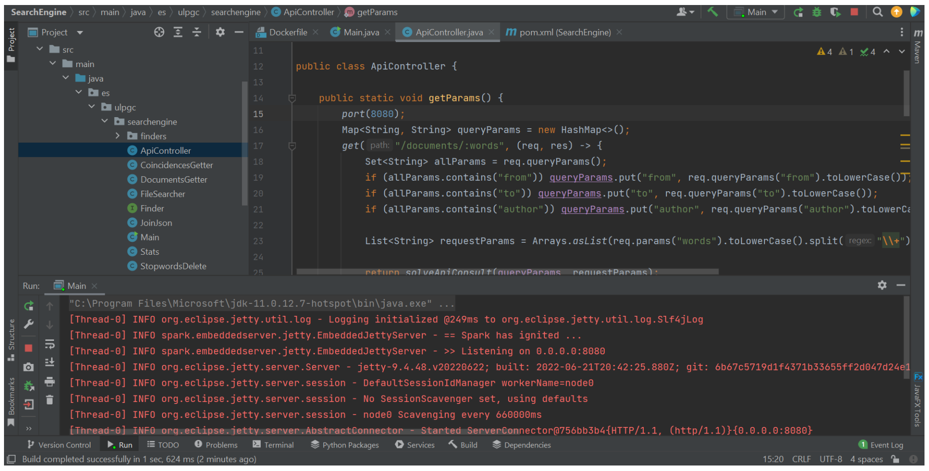


Figure 2: Diagram RRS

4 Brief Procedural Overview of Assignment

Before creating the Docker image, it was important to evaluate if the program was running in of itself, to avoid any disparities later on. As well as evaluating requests briefly made to the API, for testing purposes, using localhost with a specific port denomination, for an insight on the execution. 3 4 5

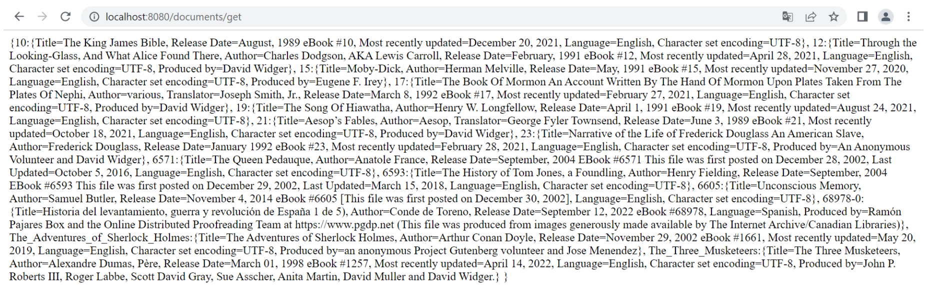


```java
public class ApiController {

    public static void getParams() {
        port(8080);
        Map<String, String> queryParams = new HashMap<>();
        get( path: "/documents/:words", (req, res) -> {
            Set<String> allParams = req.queryParams();
            if (allParams.contains("from")) queryParams.put("from", req.queryParams("from").toLowerCase());
            if (allParams.contains("to")) queryParams.put("to", req.queryParams("to").toLowerCase());
            if (allParams.contains("author")) queryParams.put("author", req.queryParams("author").toLowerCa

            List<String> requestParams = Arrays.asList(req.params("words").toLowerCase().split( regex: "\\+")

            return solveApiConsult(queryParams, requestParams);
```

```
"C:\Program Files\Microsoft\jdk-11.0.12.7-hotspot\bin\java.exe" ...
[Thread-0] INFO org.eclipse.jetty.util.log - Logging initialized @249ms to org.eclipse.jetty.util.log.Slf4jLog
[Thread-0] INFO spark.embeddedserver.jetty.EmbeddedJettyServer - == Spark has ignited ...
[Thread-0] INFO spark.embeddedserver.jetty.EmbeddedJettyServer - >> Listening on 0.0.0.0:8080
[Thread-0] INFO org.eclipse.jetty.server.Server - jetty-9.4.48.v20220622; built: 2022-06-21T20:42:25.880Z; git: 6b67c5719d1f4371b33655ff2d047d24e1
[Thread-0] INFO org.eclipse.jetty.server.session - DefaultSessionIdManager workerName=node0
[Thread-0] INFO org.eclipse.jetty.server.session - No SessionScavenger set, using defaults
[Thread-0] INFO org.eclipse.jetty.server.session - node0 Scavenging every 660000ms
[Thread-0] INFO org.eclipse.jetty.server.AbstractConnector - Started ServerConnector@756bb3b4{HTTP/1.1, (http/1.1)}{0.0.0.0:8080}
```

Build completed successfully in 1 sec, 624 ms (2 minutes ago)

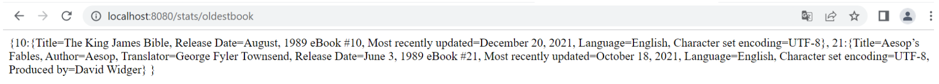
Figure 3: Intellij Testing



localhost:8080/documents/get

{10:{Title=The King James Bible, Release Date=August, 1989 eBook #10, Most recently updated=December 20, 2021, Language=English, Character set encoding=UTF-8}, 12:{Title=Through the Looking-Glass, And What Alice Found There, Author=Charles Dodgson, AKA Lewis Carroll, Release Date=February, 1991 eBook #12, Most recently updated=April 28, 2021, Language=English, Character set encoding=UTF-8, Produced by=David Widger}, 15:{Title=Moby-Dick, Author=Herman Melville, Release Date=May, 1991 eBook #15, Most recently updated=November 27, 2020, Language=English, Character set encoding=UTF-8, Produced by=Eugene F. Irey}, 17:{Title=The Book Of Mormon An Account Written By The Hand Of Mormon Upon Plates Taken From The Plates Of Nephi, Author=various, Translator=Joseph Smith, Jr., Release Date=March 8, 1992 eBook #17, Most recently updated=February 27, 2021, Language=English, Character set encoding=UTF-8, Produced by=David Widger}, 19:{Title=The Song Of Hiawatha, Author=Henry W. Longfellow, Release Date=April 1, 1991 eBook #19, Most recently updated=August 24, 2021, Language=English, Character set encoding=UTF-8}, 21:{Title=Aesop's Fables, Author=Aesop, Translator=George Fyler Townsend, Release Date=June 3, 1989 eBook #21, Most recently updated=October 18, 2021, Language=English, Character set encoding=UTF-8, Produced by=David Widger}, 23:{Title=Narrative of the Life of Frederick Douglass An American Slave, Author=Frederick Douglass, Release Date=January 1992 eBook #23, Most recently updated=February 28, 2021, Language=English, Character set encoding=UTF-8, Produced by=An Anonymous Volunteer and David Widger}, 6571:{Title=The Queen Pedauque, Author=Anatole France, Release Date=September, 2004 EBook #6571 This file was first posted on December 28, 2002, Last Updated=October 5, 2016, Language=English, Character set encoding=UTF-8}, 6593:{Title=The History of Tom Jones, a Foundling, Author=Henry Fielding, Release Date=September, 2004 EBook #6593 This file was first posted on December 29, 2002, Last Updated=March 15, 2018, Language=English, Character set encoding=UTF-8}, 6605:{Title=Unconscious Memory, Author=Samuel Butler, Release Date=November 4, 2014 eBook #6605 [This file was first posted on December 30, 2002], Language=English, Character set encoding=UTF-8}, 68978-0:{Title=Historia del levantamiento, guerra y revolución de España 1 de 5), Author=Conde de Toreno, Release Date=September 12, 2022 eBook #68978, Language=Spanish, Produced by=Ramón Pajares Box and the Online Distributed Proofreading Team at https://www.pgdp.net (This file was produced from images generously made available by The Internet Archive/Canadian Libraries)}, The_Adventures_of_Sherlock_Holmes:{Title=The Adventures of Sherlock Holmes, Author=Arthur Conan Doyle, Release Date=November 29, 2002 eBook #1661, Most recently updated=May 20, 2019, Language=English, Character set encoding=UTF-8, Produced by=an anonymous Project Gutenberg volunteer and Jose Menendez}, The_Three_Musketeers:{Title=The Three Musketeers, Author=Alexandre Dumas, Père, Release Date=March 01, 1998 eBook #1257, Most recently updated=April 14, 2022, Language=English, Character set encoding=UTF-8, Produced by=John P. Roberts III, Roger Labbe, Scott David Gray, Sue Asscher, Anita Martin, David Muller and David Widger.} }

Figure 4: Consultation 1

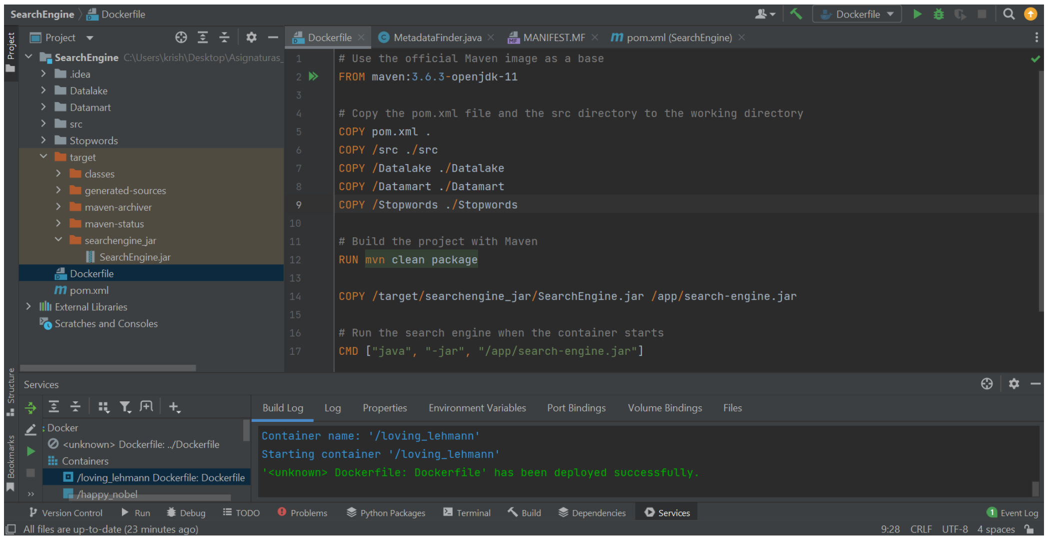


localhost:8080/stats/oldestbook

{10:{Title=The King James Bible, Release Date=August, 1989 eBook #10, Most recently updated=December 20, 2021, Language=English, Character set encoding=UTF-8}, 21:{Title=Aesop's Fables, Author=Aesop, Translator=George Fyler Townsend, Release Date=June 3, 1989 eBook #21, Most recently updated=October 18, 2021, Language=English, Character set encoding=UTF-8, Produced by=David Widger} }

Figure 5: Consultation 2

After the verification, the Dockerfile along with all of its requiring files where created, having had made sure that all of the necessary dependencies were stated to be exported in the deliverance of the image. 6

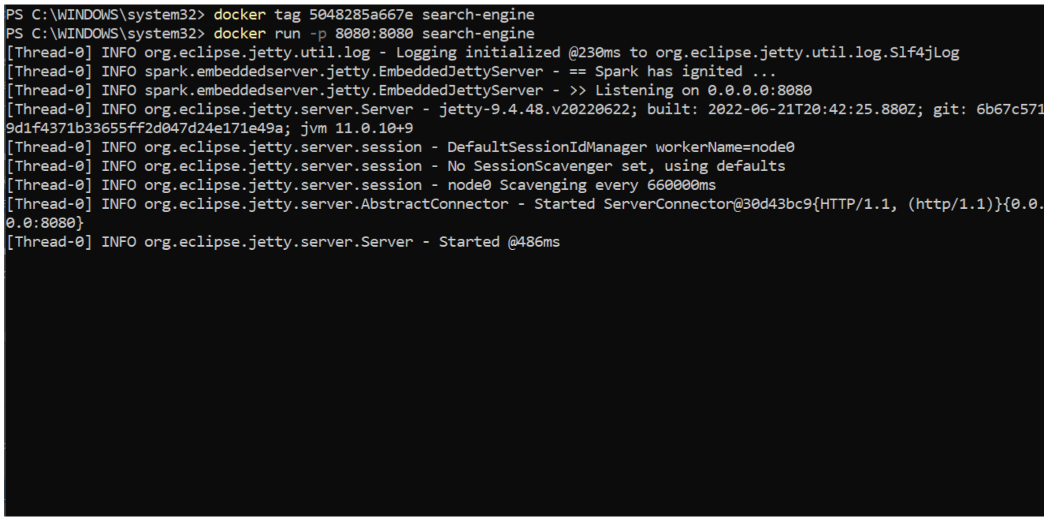


```
# Use the official Maven image as a base
FROM maven:3.6.3-openjdk-11

# Copy the pom.xml file and the src directory to the working directory
COPY pom.xml .
COPY /src ./src
COPY /Datalake ./Datalake
COPY /Datamart ./Datamart
COPY /Stopwords ./Stopwords

# Build the project with Maven
RUN mvn clean package

COPY /target/searchengine_jar/SearchEngine.jar /app/search-engine.jar

# Run the search engine when the container starts
CMD ["java", "-jar", "/app/search-engine.jar"]
```

```
Container name: '/loving_lehmann'
Starting container '/loving_lehmann'
'<unknown> Dockerfile: Dockerfile' has been deployed successfully.
```

Figure 6: Dockerfile Revision

Using localhost with port 8080, the requests to the API could be made by running in command line the image within a container. 7



```
PS C:\WINDOWS\system32> docker tag 5048285a667e search-engine
PS C:\WINDOWS\system32> docker run -p 8080:8080 search-engine
[Thread-0] INFO org.eclipse.jetty.util.log - Logging initialized @230ms to org.eclipse.jetty.util.log.Slf4jLog
[Thread-0] INFO spark.embeddedserver.jetty.EmbeddedJettyServer - == Spark has ignited ...
[Thread-0] INFO spark.embeddedserver.jetty.EmbeddedJettyServer - >> Listening on 0.0.0.0:8080
[Thread-0] INFO org.eclipse.jetty.server.Server - jetty-9.4.48.v20220622; built: 2022-06-21T20:42:25.880Z; git: 6b67c571
9d1f4371b33655ff2d047d24e171e49a; jvm 11.0.10+9
[Thread-0] INFO org.eclipse.jetty.server.session - DefaultSessionIdManager workerName=node0
[Thread-0] INFO org.eclipse.jetty.server.session - No SessionScavenger set, using defaults
[Thread-0] INFO org.eclipse.jetty.server.session - node0 Scavenging every 660000ms
[Thread-0] INFO org.eclipse.jetty.server.AbstractConnector - Started ServerConnector@30d43bc9{HTTP/1.1, (http/1.1)}{0.0.
0.0:8080}
[Thread-0] INFO org.eclipse.jetty.server.Server - Started @486ms
```

Figure 7: Execution of Image

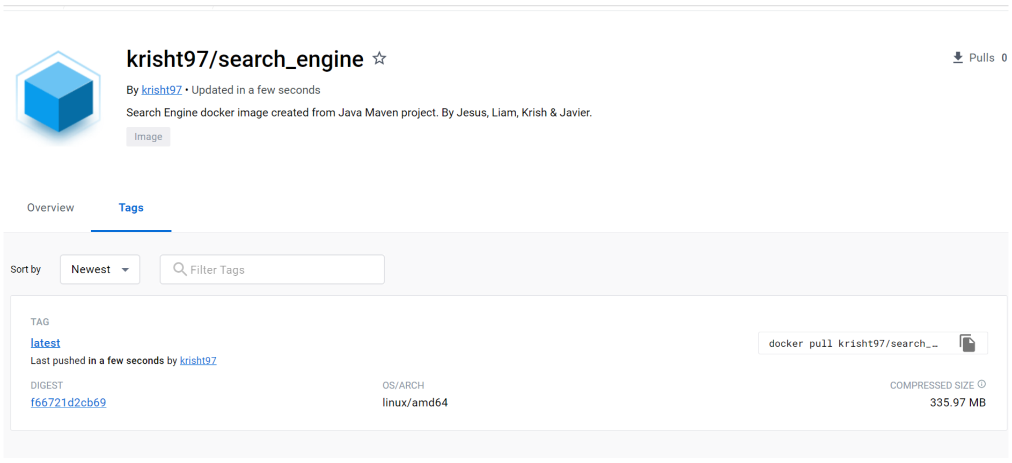
The next step was to push the image, with the latest tag, after having had logged in through the command line. So that any system were to possess the ability to be able to pull the image and execute within its own terms. 8

When it comes to Load Balancing for this implementation, the idea is to create a Docker network for the containers to use. Later multiple instances of the docker image would be created. 9

Next, an Nginx container needs to be started on the same network. Considering that the Nginx needs to act as the load balancer for the instances of the Docker image previously created. 10

For that a configuration file has to be edited and a server block needs to be introduced for each instance of the image. The file would be edited with nano through bash. 11 12

The final restart ensures that the changes made to the configuration file are applied. 13 14


krisht97/search_engine

By krisht97 • Updated in a few seconds

Search Engine docker image created from Java Maven project. By Jesus, Liam, Krish & Javier.

TAG
latest
Last pushed in a few seconds by krisht97

docker pull krisht97/search_…

DIGEST
f66721d2cb69

OS/ARCH
linux/amd64

COMPRESSED SIZE
335.97 MB

Figure 8: Image in DockerHub

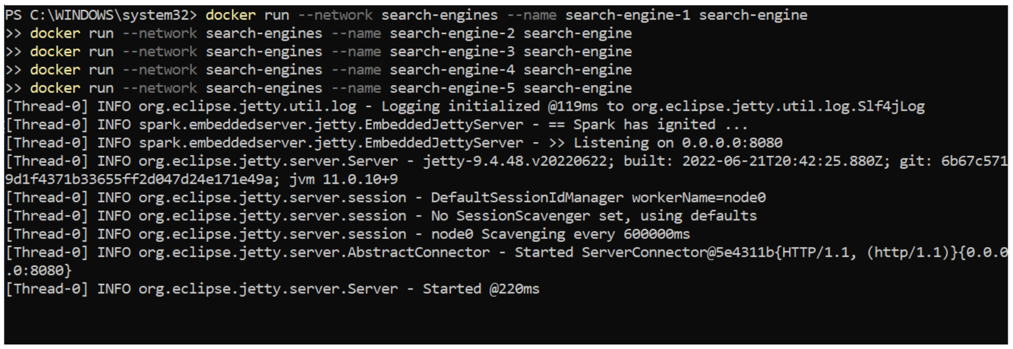

```
PS C:\WINDOWS\system32> docker run --network search-engines --name search-engine-1 search-engine
>> docker run --network search-engines --name search-engine-2 search-engine
>> docker run --network search-engines --name search-engine-3 search-engine
>> docker run --network search-engines --name search-engine-4 search-engine
>> docker run --network search-engines --name search-engine-5 search-engine
[Thread-0] INFO org.eclipse.jetty.util.log - Logging initialized @119ms to org.eclipse.jetty.util.log.Slf4jLog
[Thread-0] INFO spark.embeddedserver.jetty.EmbeddedJettyServer - == Spark has ignited ...
[Thread-0] INFO spark.embeddedserver.jetty.EmbeddedJettyServer - >> Listening on 0.0.0.0:8080
[Thread-0] INFO org.eclipse.jetty.server.Server - jetty-9.4.48.v20220622; built: 2022-06-21T20:42:25.880Z; git: 6b67c571
9d1f4371b33655ff2d047d24e171e49a; jvm 11.0.10+9
[Thread-0] INFO org.eclipse.jetty.server.session - DefaultSessionIdManager workerName=node0
[Thread-0] INFO org.eclipse.jetty.server.session - No SessionScavenger set, using defaults
[Thread-0] INFO org.eclipse.jetty.server.session - node0 Scavenging every 600000ms
[Thread-0] INFO org.eclipse.jetty.server.AbstractConnector - Started ServerConnector@5e4311b{HTTP/1.1, (http/1.1)}{0.0.0
.0:8080}
[Thread-0] INFO org.eclipse.jetty.server.Server - Started @220ms
```

Figure 9: Multiple Instances of Image

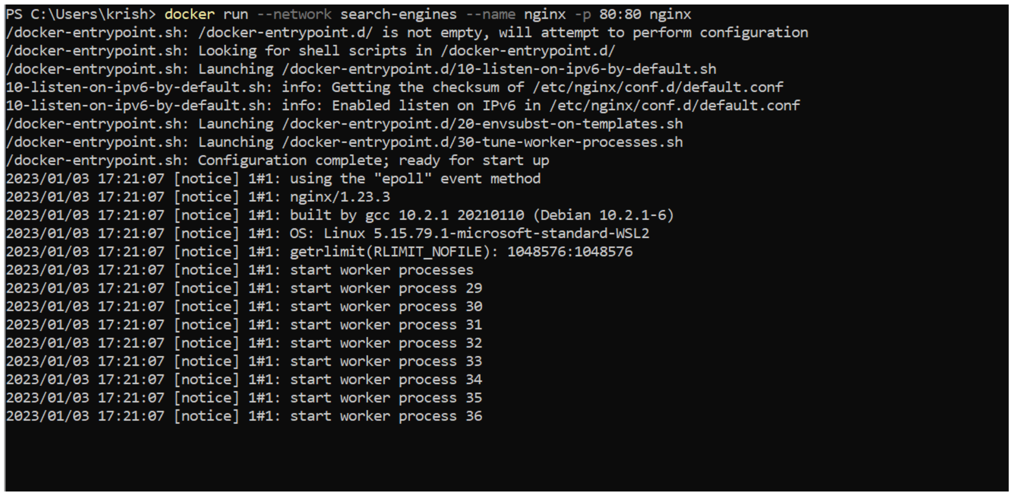
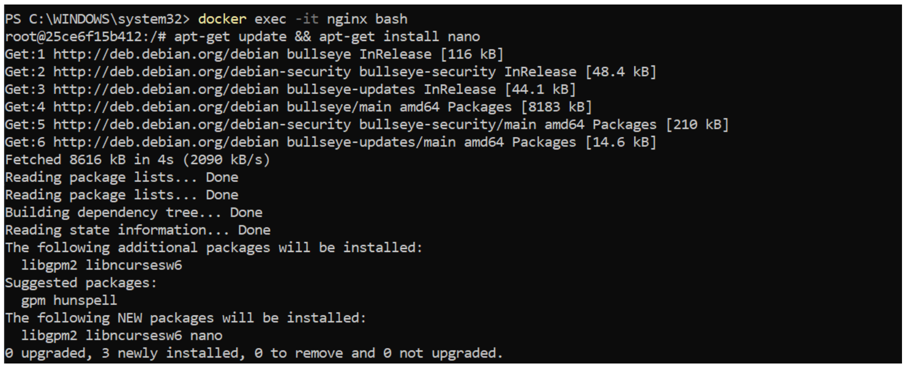

```
PS C:\Users\krish> docker run --network search-engines --name nginx -p 80:80 nginx
/docker-entrypoint.sh: /docker-entrypoint.d/ is not empty, will attempt to perform configuration
/docker-entrypoint.sh: Looking for shell scripts in /docker-entrypoint.d/
/docker-entrypoint.sh: Launching /docker-entrypoint.d/10-listen-on-ipv6-by-default.sh
10-listen-on-ipv6-by-default.sh: info: Getting the checksum of /etc/nginx/conf.d/default.conf
10-listen-on-ipv6-by-default.sh: info: Enabled listen on IPv6 in /etc/nginx/conf.d/default.conf
/docker-entrypoint.sh: Launching /docker-entrypoint.d/20-envsubst-on-templates.sh
/docker-entrypoint.sh: Launching /docker-entrypoint.d/30-tune-worker-processes.sh
/docker-entrypoint.sh: Configuration complete; ready for start up
2023/01/03 17:21:07 [notice] 1#1: using the "epoll" event method
2023/01/03 17:21:07 [notice] 1#1: nginx/1.23.3
2023/01/03 17:21:07 [notice] 1#1: built by gcc 10.2.1 20210110 (Debian 10.2.1-6)
2023/01/03 17:21:07 [notice] 1#1: OS: Linux 5.15.79.1-microsoft-standard-WSL2
2023/01/03 17:21:07 [notice] 1#1: getrlimit(RLIMIT_NOFILE): 1048576:1048576
2023/01/03 17:21:07 [notice] 1#1: start worker processes
2023/01/03 17:21:07 [notice] 1#1: start worker process 29
2023/01/03 17:21:07 [notice] 1#1: start worker process 30
2023/01/03 17:21:07 [notice] 1#1: start worker process 31
2023/01/03 17:21:07 [notice] 1#1: start worker process 32
2023/01/03 17:21:07 [notice] 1#1: start worker process 33
2023/01/03 17:21:07 [notice] 1#1: start worker process 34
2023/01/03 17:21:07 [notice] 1#1: start worker process 35
2023/01/03 17:21:07 [notice] 1#1: start worker process 36
```

Figure 10: Nginx Container Instantiation

To test the load balancer by making requests to the Nginx server, one could use a tool such as "curl". As such:

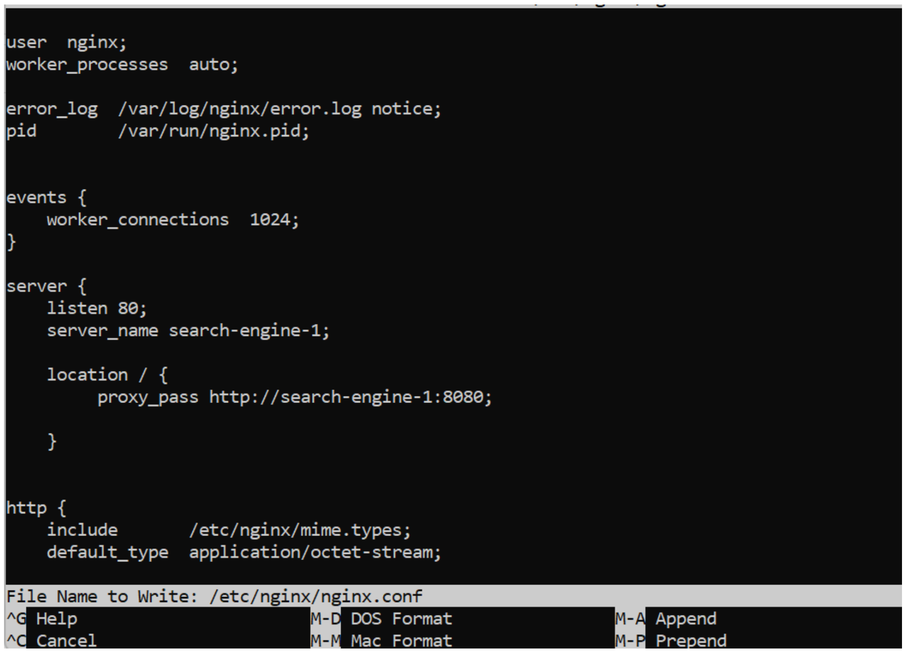
- curl http://localhost/

However, it might possibly give a "Unable to connect to the remote server" error. This is most probably due to firewall or other network configuration issue from the device preventing "curl" from reaching the Nginx server.



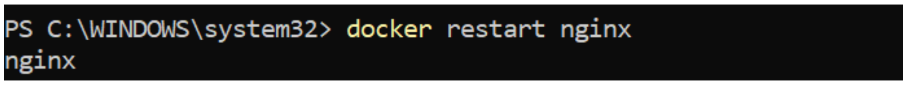
```
PS C:\WINDOWS\system32> docker exec -it nginx bash
root@25ce6f15b412:/# apt-get update && apt-get install nano
Get:1 http://deb.debian.org/debian bullseye InRelease [116 kB]
Get:2 http://deb.debian.org/debian-security bullseye-security InRelease [48.4 kB]
Get:3 http://deb.debian.org/debian bullseye-updates InRelease [44.1 kB]
Get:4 http://deb.debian.org/debian bullseye/main amd64 Packages [8183 kB]
Get:5 http://deb.debian.org/debian-security bullseye-security/main amd64 Packages [210 kB]
Get:6 http://deb.debian.org/debian bullseye-updates/main amd64 Packages [14.6 kB]
Fetched 8616 kB in 4s (2090 kB/s)
Reading package lists... Done
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  libgpm2 libncursesw6
Suggested packages:
  gpm hunspell
The following NEW packages will be installed:
  libgpm2 libncursesw6 nano
0 upgraded, 3 newly installed, 0 to remove and 0 not upgraded.
```

Figure 11: Nginx Bash Command



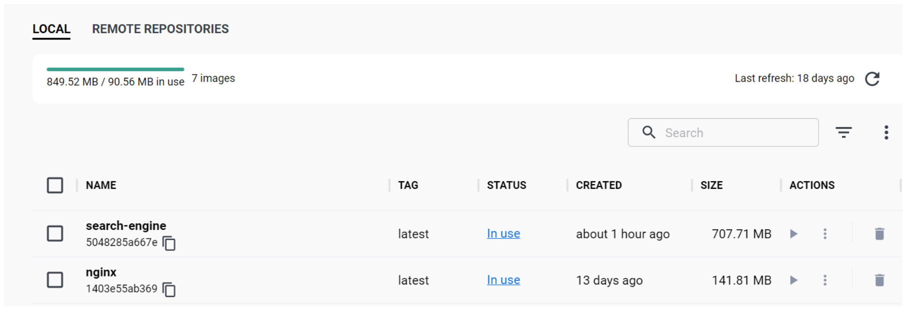
```
user  nginx;
worker_processes  auto;

error_log  /var/log/nginx/error.log notice;
pid        /var/run/nginx.pid;


events {
    worker_connections  1024;
}

server {
    listen 80;
    server_name search-engine-1;

    location / {
         proxy_pass http://search-engine-1:8080;

    }


http {
    include       /etc/nginx/mime.types;
    default_type  application/octet-stream;

File Name to Write: /etc/nginx/nginx.conf
^G Help                          M-D DOS Format                   M-A Append
^C Cancel                        M-M Mac Format                   M-P Prepend
```

Figure 12: Nginx Configuration File



```
PS C:\WINDOWS\system32> docker restart nginx
nginx
```

Figure 13: Nginx Restart Command



NAME	TAG	STATUS	CREATED	SIZE	ACTIONS
search-engine 5048285a667e	latest	In use	about 1 hour ago	707.71 MB	
nginx 1403e55ab369	latest	In use	13 days ago	141.81 MB	

Figure 14: Docker Desktop Revision

5 Conclusion

Docker is a tool that is vital when it comes to executing fully-fledged applications in a lightweight and efficient environment. Without the needs to install specific libraries, frameworks and tools, or have a specific OS system. All the dependencies are incorporated upon the execution of the image, and its no wonder why many are moving from the use of hypervisors to containerization methodology.

In terms of load balancing, it has become an effective way for businesses to keep up with rising demand and ensure that their applications are up and running for users. Today's businesses receive hundreds and thousands of client requests per minute to their websites and applications. During peak season or hours, this traffic can spike up even more. Servers are under pressure to keep up and respond with high-quality media including photos, videos and other application data. Any interruptions or downtime in these applications can result in below par experiences and turn away users, resulting in loss of profits.

References

[IBM] IBM. What is load balancing?

[Net] AVI Networks. History of load balancing.