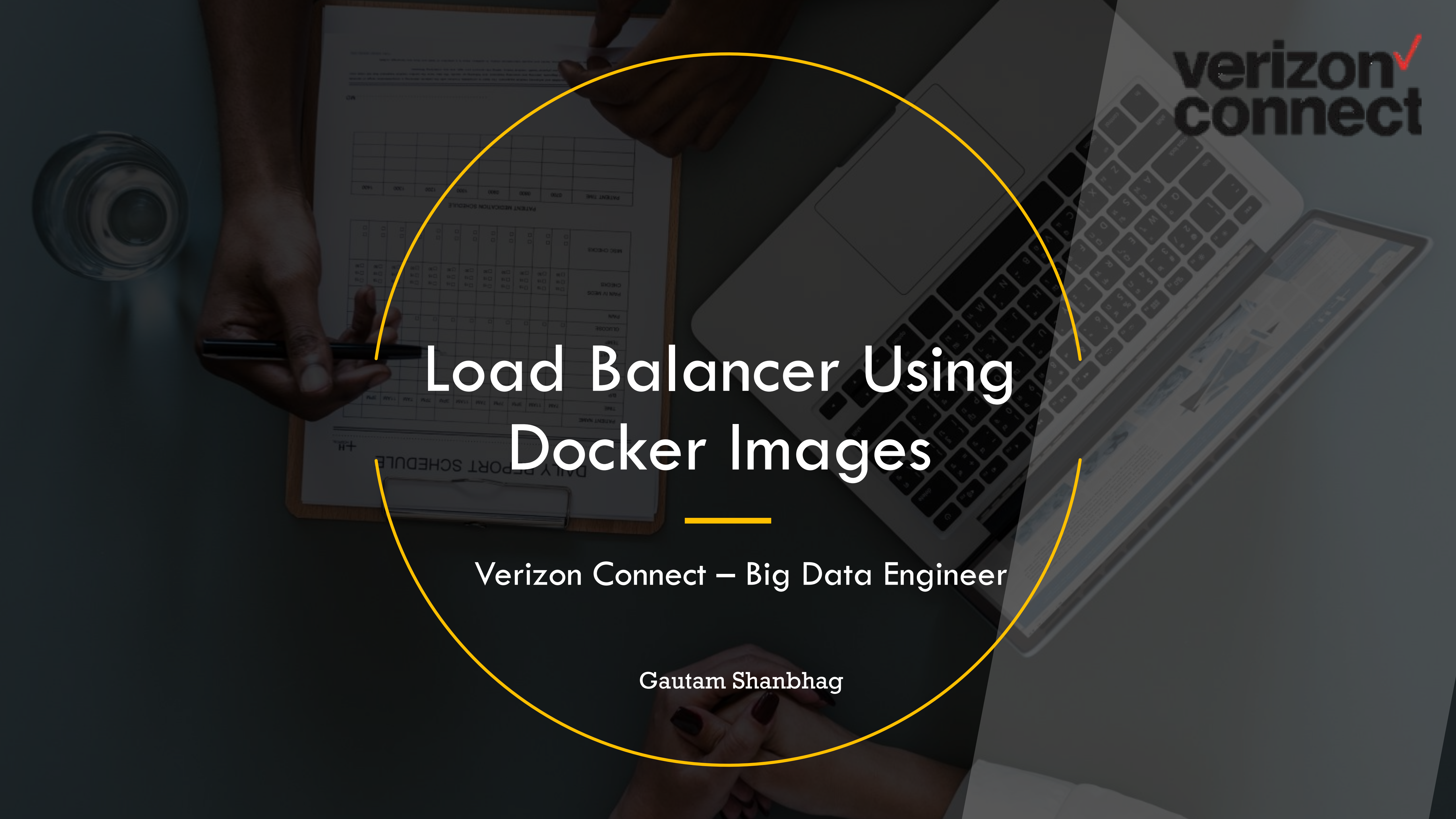


verizon connect

# Load Balancer Using Docker Images

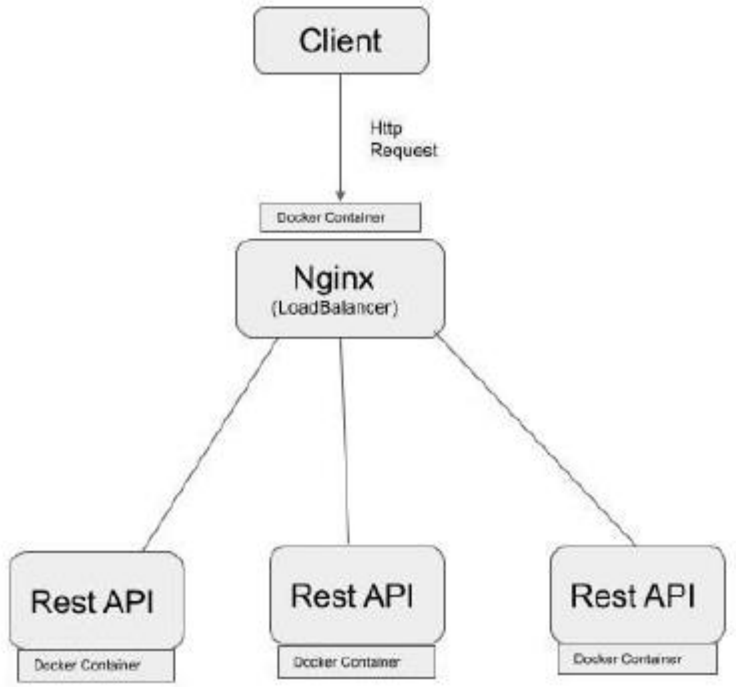
Verizon Connect – Big Data Engineer

Gautam Shanbhag

## Contents :

## Problem Statement :

- Create Load Balancer Using multiple Docker Containers
- Nginx to act as a Load Balancer (round –robin algo) in one docker container
- 3 Spring boot java application hosted on individual docker containers to serve as backend servers
- Spring boot application to return machine ip:port and hostname



## Prerequisites / Tools Required :

- Docker
- Nginx
- Spring boot application built on Eclipse / Intelli J

## Description :

**Docker** is a set of platform-as-a-service products that use OS-level virtualization to deliver software in packages called containers. Containers are isolated from one another and bundle their own software, libraries and configuration files; they can communicate with each other through well-defined channels.

**Nginx** is a web server which can also be used as a reverse proxy, load balancer, mail proxy and HTTP cache.

**Spring Boot** is an open source Java-based framework used to create a Micro Service. It is easy to create a stand-alone and production ready spring applications using Spring Boot.

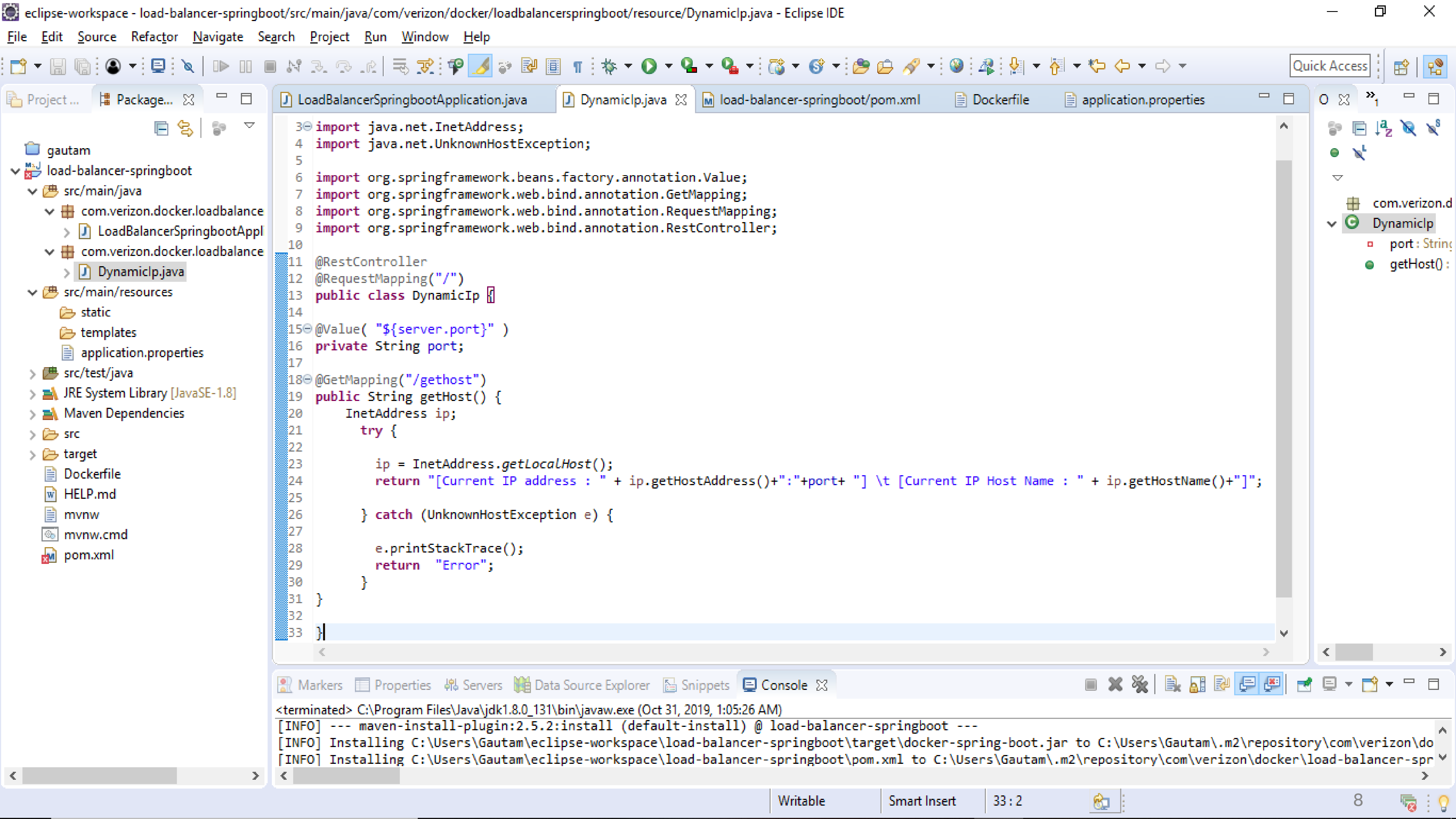
## Proposed Approach :

1. Create a Spring boot rest application which returns machine ip and host name on api call
2. Start nginx standalone server with load balancer configuration
3. Test connection between sprint boot application and nginx
4. Use docker to create individual docker images for spring boot application
5. Use docker to create nginx image and copy local nginx configuration to the docker image
6. Use docker-compose.yml to establish communication between the 4 docker images

## Step By Step Process :

- ❖ Installation of Docker
  - https://docs.docker.com/docker-for-windows/install/

- ❖ Spring Boot Application
  - Create a spring boot application using https://start.spring.io/
  - Use favorable editor (eclipse / IntelliJ)to open the spring boot app
  - Create a Simple Class with Rest Mapping which would read the host machine's ip port and hostname
  - Build java jar using maven commands (maven clean install)

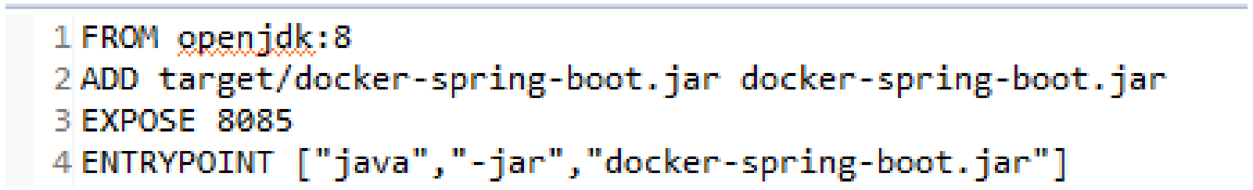


eclipse-workspace - load-balancer-springboot/src/main/java/com/verizon/docker/loadbalancerspringboot/resource/DynamicIp.java - Eclipse IDE

```
import java.net.InetAddress;
import java.net.UnknownHostException;

import org.springframework.beans.factory.annotation.Value;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

@RestController
@RequestMapping("/")
public class DynamicIp {

@Value( "${server.port}" )
private String port;

@GetMapping("/gethost")
public String getHost() {
    InetAddress ip;
      try {

        ip = InetAddress.getLocalHost();
        return "[Current IP address : " + ip.getHostAddress()+":"+port+ "] \t [Current IP Host Name : " + ip.getHostName()+"]";

      } catch (UnknownHostException e) {

        e.printStackTrace();
        return  "Error";
      }
}

}
```

```
<terminated> C:\Program Files\Java\jdk1.8.0_131\bin\javaw.exe (Oct 31, 2019, 1:05:26 AM)
[INFO] --- maven-install-plugin:2.5.2:install (default-install) @ load-balancer-springboot ---
[INFO] Installing C:\Users\Gautam\eclipse-workspace\load-balancer-springboot\target\docker-spring-boot.jar to C:\Users\Gautam\.m2\repository\com\verizon\do
[INFO] Installing C:\Users\Gautam\eclipse-workspace\load-balancer-springboot\pom.xml to C:\Users\Gautam\.m2\repository\com\verizon\docker\load-balancer-spr
```

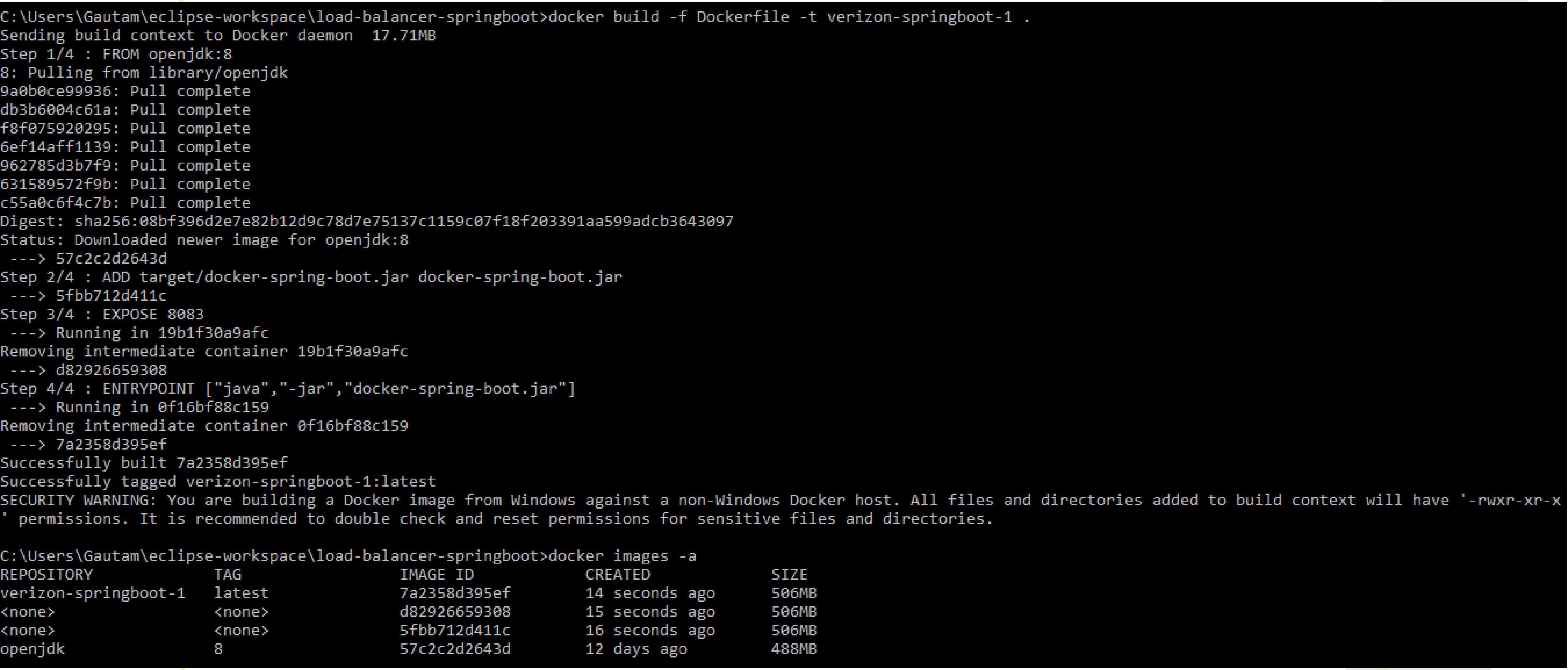
- Create a Dockerfile in the repository



```
FROM openjdk:8
ADD target/docker-spring-boot.jar docker-spring-boot.jar
EXPOSE 8085
ENTRYPOINT ["java","-jar","docker-spring-boot.jar"]
```

- Build Docker images using the Dockerfile
  - ➢ This  will download base java image from docker hub
  - ➢ Copy the spring boot jar from host machine to docker image
  - ➢ Expose docker port 8085 for external use
  - ➢ And run the java jar using “java –jar {jar_name}”

```
docker build -f Dockerfile -t verizon-springboot-1 .
```


```
C:\Users\Gautam\eclipse-workspace\load-balancer-springboot>docker build -f Dockerfile -t verizon-springboot-1 .
Sending build context to Docker daemon  17.71MB
Step 1/4 : FROM openjdk:8
8: Pulling from library/openjdk
9a0b0ce99936: Pull complete
db3b6004c61a: Pull complete
f8f075920295: Pull complete
6ef14aff1139: Pull complete
962785d3b7f9: Pull complete
631589572f9b: Pull complete
c55a0c6f4c7b: Pull complete
Digest: sha256:08bf396d2e7e82b12d9c78d7e75137c1159c07f18f203391aa599adcb3643097
Status: Downloaded newer image for openjdk:8
 ---> 57c2c2d2643d
Step 2/4 : ADD target/docker-spring-boot.jar docker-spring-boot.jar
 ---> 5fbb712d411c
Step 3/4 : EXPOSE 8083
 ---> Running in 19b1f30a9afc
Removing intermediate container 19b1f30a9afc
 ---> d82926659308
Step 4/4 : ENTRYPOINT ["java","-jar","docker-spring-boot.jar"]
 ---> Running in 0f16bf88c159
Removing intermediate container 0f16bf88c159
 ---> 7a2358d395ef
Successfully built 7a2358d395ef
Successfully tagged verizon-springboot-1:latest
SECURITY WARNING: You are building a Docker image from Windows against a non-Windows Docker host. All files and directories added to build context will have '-rwxr-xr-x
' permissions. It is recommended to double check and reset permissions for sensitive files and directories.

C:\Users\Gautam\eclipse-workspace\load-balancer-springboot>docker images -a
REPOSITORY             TAG                 IMAGE ID            CREATED             SIZE
verizon-springboot-1   latest              7a2358d395ef        14 seconds ago      506MB
<none>                 <none>              d82926659308        15 seconds ago      506MB
<none>                 <none>              5fbb712d411c        16 seconds ago      506MB
openjdk                8                   57c2c2d2643d        12 days ago         488MB
```

- Similarly create 2 more docker spring boot images to serve as backend server by simply changing image names
- change ports if required

## ❖ Nginx Configuration

- Install nginx on local server and modify nginx.conf to setup load balancer

```
upstream verizonLB{
server verizon-springboot-1:8083;
server verizon-springboot-2:8084;
server verizon-springboot-3:8085;
}

server {
    listen       80;
    server_name  localhost;

    #charset koi8-r;

    #access_log  logs/host.access.log  main;
    #access_log  /var/log/nginx/access.log  main;


    location / {
       ## root   html;
       ## index  index.html index.htm;
       proxy_pass http://verizonLB;
    }
```

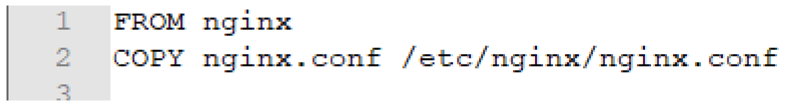
Nginx supports 3 types of load balancing mechanism

- Round-robin (default)
- Least-connected
- Ip-hash

Proper logging format when using load balancer

```
log_format main '[$time_local] $remote_addr $remote_user $server_name  '
       'to: $upstream_addr: $request upstream_response_time '
       '$upstream_response_time msec $msec request_time $request_time';
```

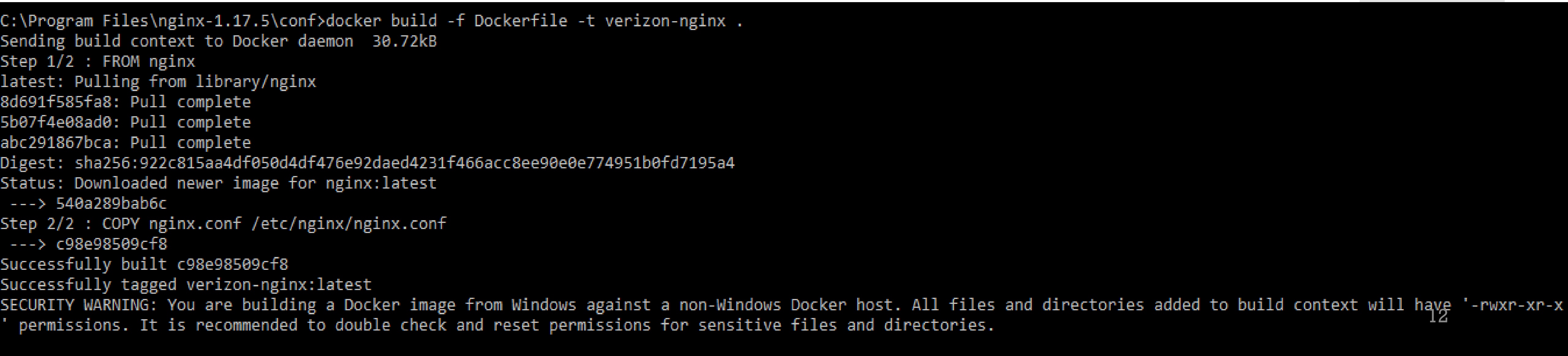
- Create a nginx Dockerfile in the conf repository



```
FROM nginx
COPY nginx.conf /etc/nginx/nginx.conf
```

- Build Docker images using the Dockerfile
  - ➢ This  will download base nginx image from docker hub
  - ➢ Copy/replace the nginx.conf file in linux docker environment at etc/nginx/nginx.conf

```
docker build -f Dockerfile -t verizon-nginx .
```

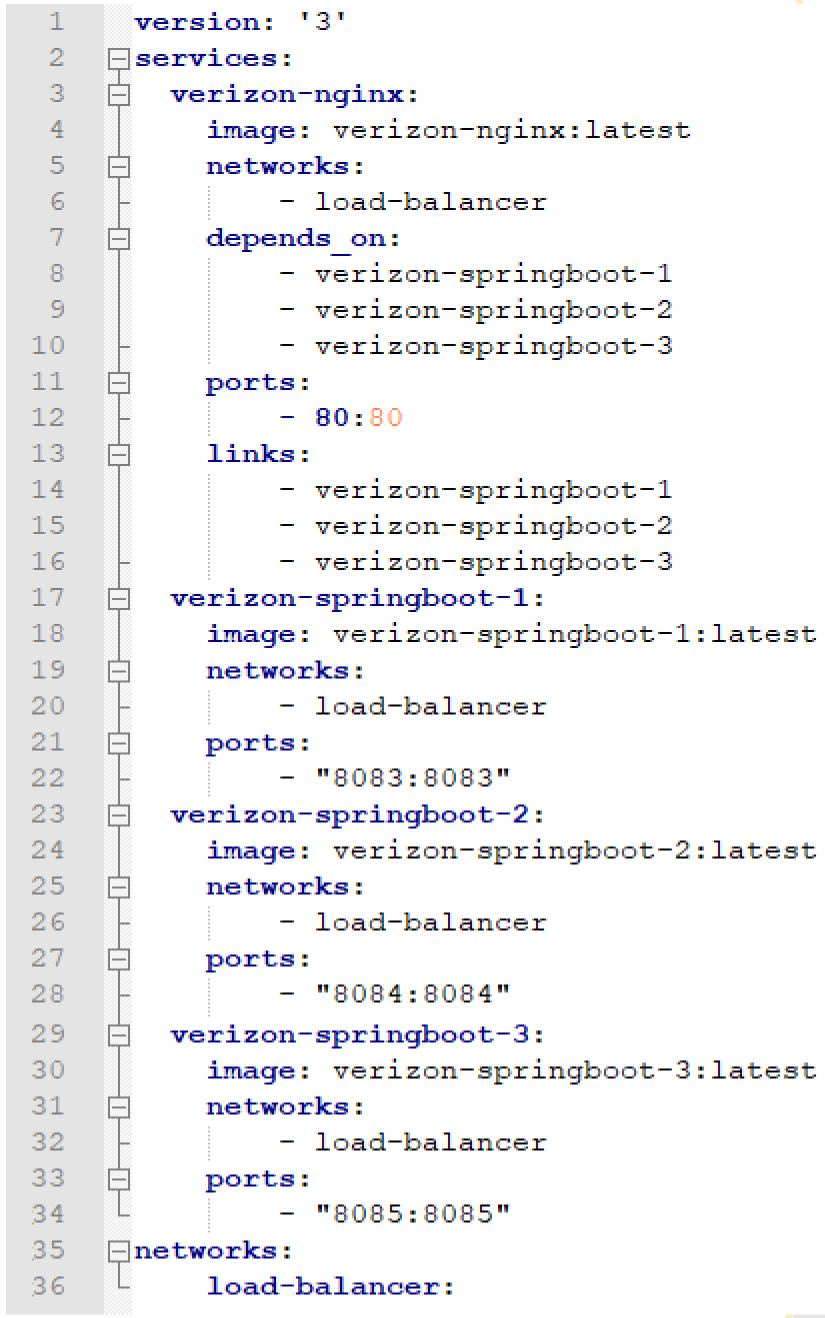


```
C:\Program Files\nginx-1.17.5\conf>docker build -f Dockerfile -t verizon-nginx .
Sending build context to Docker daemon  30.72kB
Step 1/2 : FROM nginx
latest: Pulling from library/nginx
8d691f585fa8: Pull complete
5b07f4e08ad0: Pull complete
abc291867bca: Pull complete
Digest: sha256:922c815aa4df050d4df476e92daed4231f466acc8ee90e0e774951b0fd7195a4
Status: Downloaded newer image for nginx:latest
 ---> 540a289bab6c
Step 2/2 : COPY nginx.conf /etc/nginx/nginx.conf
 ---> c98e98509cf8
Successfully built c98e98509cf8
Successfully tagged verizon-nginx:latest
SECURITY WARNING: You are building a Docker image from Windows against a non-Windows Docker host. All files and directories added to build context will have '-rwxr-xr-x
' permissions. It is recommended to double check and reset permissions for sensitive files and directories.
```

- Create a docker-compose.yaml file

docker-compose.yaml

- ➢ Creates Services Verizon-nginx, Verizon-springboot-1, Verizon-springboot-2, Verizon-springboot-3
- ➢ Verizon nginx depends on the other 3 containers and will be built at the end
- ➢ External ports are exposed
- ➢ Verizon-nginx are linked to spring boot applications so that it can communicate with external resources
- ➢ All services are bundled in a single load-balancer network



```yaml
version: '3'
services:
  verizon-nginx:
    image: verizon-nginx:latest
    networks:
        - load-balancer
    depends_on:
        - verizon-springboot-1
        - verizon-springboot-2
        - verizon-springboot-3
    ports:
        - 80:80
    links:
        - verizon-springboot-1
        - verizon-springboot-2
        - verizon-springboot-3
  verizon-springboot-1:
    image: verizon-springboot-1:latest
    networks:
        - load-balancer
    ports:
        - "8083:8083"
  verizon-springboot-2:
    image: verizon-springboot-2:latest
    networks:
        - load-balancer
    ports:
        - "8084:8084"
  verizon-springboot-3:
    image: verizon-springboot-3:latest
    networks:
        - load-balancer
    ports:
        - "8085:8085"
networks:
    load-balancer:
```

- Build and run docker-compose.yml file using

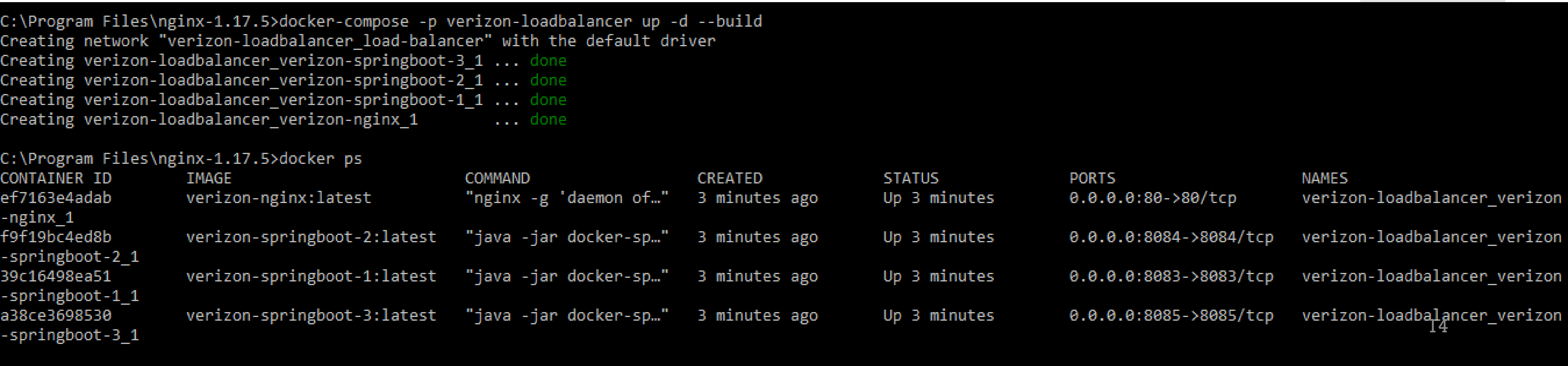
```
docker-compose -p verizon-loadbalancer up -d –build
```

- Check running docker states using

```
docker ps
```

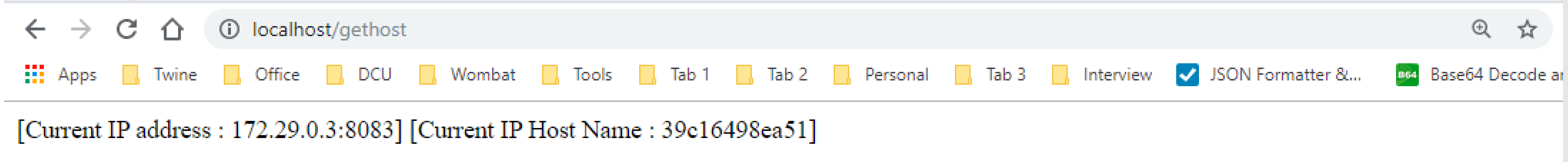
- Stop docker images using

```
docker-compose verizon-loadbalancer down
```



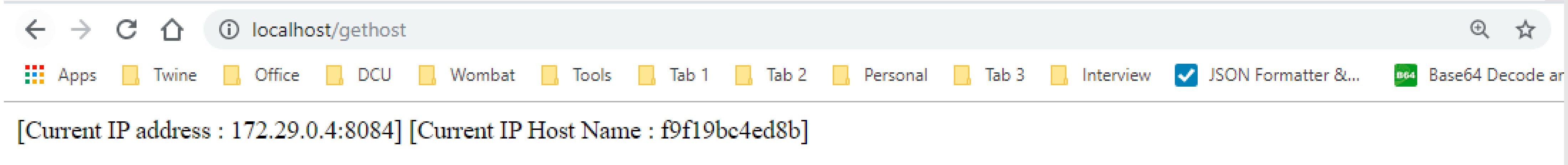
```
C:\Program Files\nginx-1.17.5>docker-compose -p verizon-loadbalancer up -d --build
Creating network "verizon-loadbalancer_load-balancer" with the default driver
Creating verizon-loadbalancer_verizon-springboot-3_1 ... done
Creating verizon-loadbalancer_verizon-springboot-2_1 ... done
Creating verizon-loadbalancer_verizon-springboot-1_1 ... done
Creating verizon-loadbalancer_verizon-nginx_1        ... done

C:\Program Files\nginx-1.17.5>docker ps
CONTAINER ID        IMAGE                         COMMAND                  CREATED             STATUS              PORTS                    NAMES
ef7163e4adab        verizon-nginx:latest          "nginx -g 'daemon of…"   3 minutes ago       Up 3 minutes        0.0.0.0:80->80/tcp       verizon-loadbalancer_verizon
-nginx_1
f9f19bc4ed8b        verizon-springboot-2:latest   "java -jar docker-sp…"   3 minutes ago       Up 3 minutes        0.0.0.0:8084->8084/tcp   verizon-loadbalancer_verizon
-springboot-2_1
39c16498ea51        verizon-springboot-1:latest   "java -jar docker-sp…"   3 minutes ago       Up 3 minutes        0.0.0.0:8083->8083/tcp   verizon-loadbalancer_verizon
-springboot-1_1
a38ce3698530        verizon-springboot-3:latest   "java -jar docker-sp…"   3 minutes ago       Up 3 minutes        0.0.0.0:8085->8085/tcp   verizon-loadbalancer_verizon
-springboot-3_1
```

# Output :



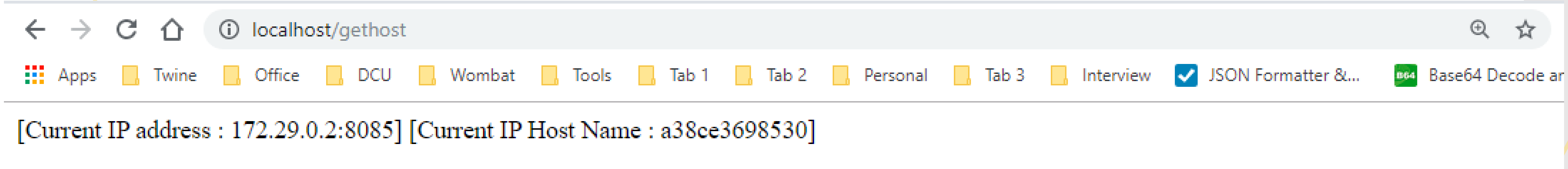
localhost/gethost
Apps Twine Office DCU Wombat Tools Tab 1 Tab 2 Personal Tab 3 Interview JSON Formatter &... Base64 Decode ar

```
[Current IP address : 172.29.0.3:8083] [Current IP Host Name : 39c16498ea51]
```



localhost/gethost
Apps Twine Office DCU Wombat Tools Tab 1 Tab 2 Personal Tab 3 Interview JSON Formatter &... Base64 Decode ar

```
[Current IP address : 172.29.0.4:8084] [Current IP Host Name : f9f19bc4ed8b]
```



localhost/gethost
Apps Twine Office DCU Wombat Tools Tab 1 Tab 2 Personal Tab 3 Interview JSON Formatter &... Base64 Decode ar

```
[Current IP address : 172.29.0.2:8085] [Current IP Host Name : a38ce3698530]
```

## Problems Faced :

1. Connecting docker images to communicate with each other
2. Copying nginx.conf contents from windows machine to linux vm and starting nginx server
3. Debugging into docker containers to view logs
4. Portability - Deploying / sharing docker images

## Learnings :

1. Understanding nginx
2. Load-balancing using nginx
3. Docker from scratch
4. Docker configurations and commands

## References :

- https://docs.docker.com/docker-for-windows/install/
- https://www.youtube.com/watch?v=FlSup_eelYE
- https://hostry.com/blog/nginx-configuration-of-a-simple-load-balancing/
- https://www.youtube.com/watch?v=4ZHhYQuzL2g/
- http://nginx.org/en/docs/windows.html
- https://docs.nginx.com/nginx/admin-guide/monitoring/logging/
- https://serverfault.com/questions/842423/how-to-completely-kill-nginx-in-windows
- https://medium.com/@nirgn/load-balancing-applications-with-haproxy-and-docker-d719b7c5b231
- https://www.callicoder.com/spring-boot-mysql-react-docker-compose-example/
- https://forums.docker.com/t/nginx-getting-proxy-pass-to-work/32889
- https://medium.com/@ainaleke/spinning-up-and-managing-multi-container-apps-using-docker-compose-d0d9f1e4d8ae
- https://hackernoon.com/docker-commands-the-ultimate-cheat-sheet-994ac78e2888
- https://phase2.github.io/devtools/common-tasks/ssh-into-a-container/
- https://www.mkyong.com/java/how-to-get-ip-address-in-java/
- https://runnable.com/docker/rails/manage-share-docker-images
- http://blog.thoward37.me/articles/where-are-docker-images-stored/
- https://www.baeldung.com/properties-with-spring
- https://docs.docker.com/compose/reference/overview/
- https://medium.com/elements/docker-application-packages-83c141d8cb0f

# THANK YOU