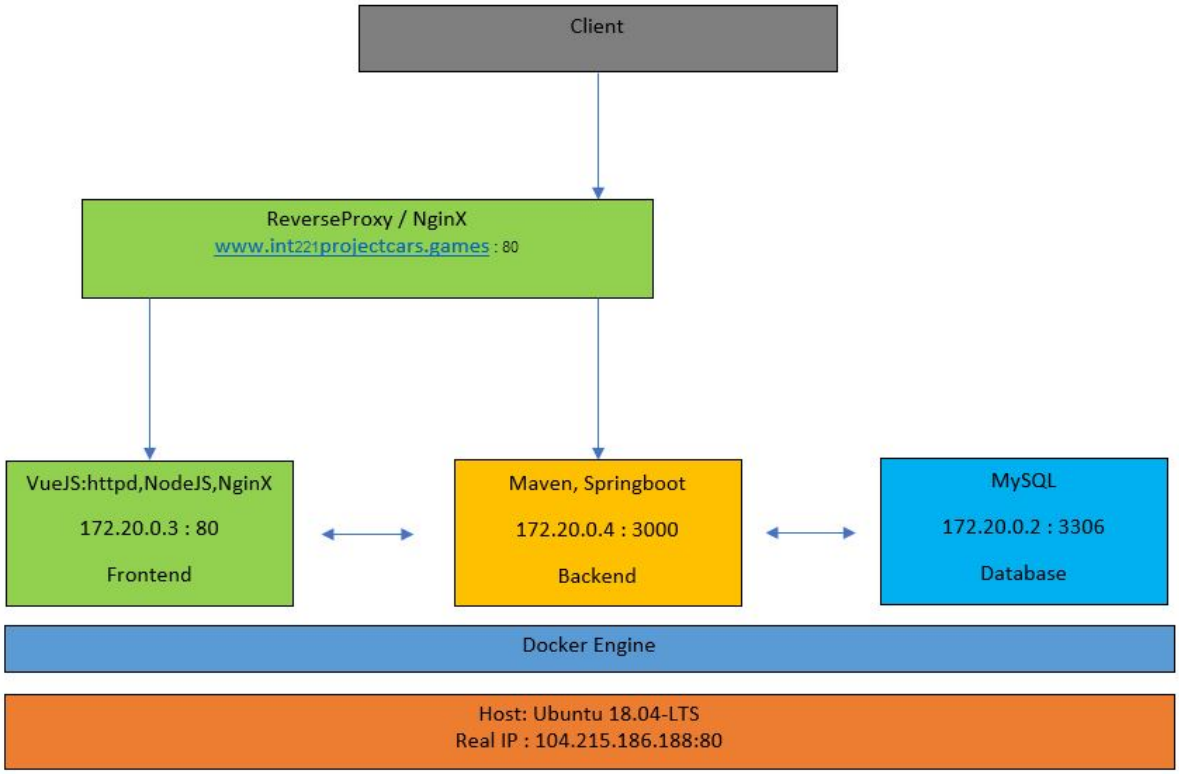
DevOps

1. Infrastructure Architecture Diagram



1.1.proxy : http://104.215.186.188/

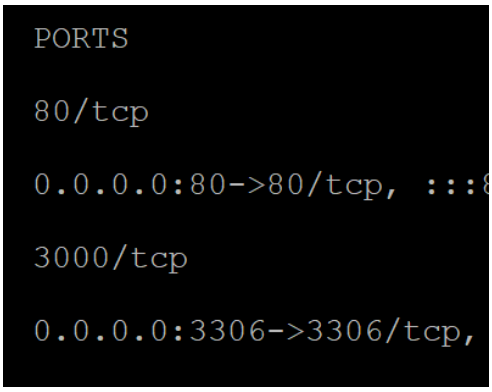
1.2.containers :

```
CONTAINER ID   IMAGE

e99b820a8b52   frontend

299c8c0dc65f   nginx
0/tcp
fef2cdf581dc   backend

d35d0379ba9c   database
```

1.3.IP addresses (host, containers) :


```
"Containers": {
    "299c8c0dc65fbe9b086b07b91261fd7b0eea592fe09870946666d724b0629a77": {
        "Name": "int221integratedprojectproxy_reverseproxy_1",
        "EndpointID": "4b0b84e78cb0b30341f04d45b889078518f5477c023b12c225e92946250e04a7",
        "MacAddress": "02:42:ac:14:00:05",
        "IPv4Address": "172.20.0.5/16",
        "IPv6Address": ""
    },
    "d35d0379ba9ca2f3fa5165bcf2fc7b1845daa75adc827b53134f96dfa4f3cf9e": {
        "Name": "database",
        "EndpointID": "b5bf1a426832e026391f88151ea9ea80025e1ca34392c64c4a0520e3bc6f4490",
        "MacAddress": "02:42:ac:14:00:02",
        "IPv4Address": "172.20.0.2/16",
        "IPv6Address": ""
    },
    "e99b820a8b526d70e479623736f193c6a06fca83ce005b547fc569157cc04bb5": {
        "Name": "frontend",
        "EndpointID": "6fbd71c49ba72efb85dfa7a1a8aae6ba5c7c6779558f9ad9c6406931d4344b5d",
        "MacAddress": "02:42:ac:14:00:03",
        "IPv4Address": "172.20.0.3/16",
        "IPv6Address": ""
    },
    "fef2cdf581dc34056f0710567b5f6f9ba2cca527bf0bb1c62bd12ac2286d7eb9": {
        "Name": "backend",
        "EndpointID": "cf49c4f748411c4a95e55aeddf6433b8dcb1fcd7b006581525f616f1409c15c1",
        "MacAddress": "02:42:ac:14:00:04",
        "IPv4Address": "172.20.0.4/16",
        "IPv6Address": ""
    }
```


1.4.open ports (host, containers) :



1.5. App URL (at proxy) :

http://www.int221projectcars.games/ @ http://104.215.186.188/

2. List of Docker images that you used with versions/tags

REPOSITORY	TAG
frontend	latest
database	latest
backend	latest
node	latest
openjdk	11.0-slim
mysql	latest
node	14.16.1-alpine3.10
nginx	1.19.10-alpine
nginx	latest
maven	3.6.1-jdk-11-slim

```
REPOSITORY   TAG                 IMAGE ID       CREATED          SIZE
frontend     latest              8085921948a8   56 minutes ago   152MB
database     latest              6c4efbcffb0c   8 hours ago      556MB
backend      latest              ff374474b0a8   39 hours ago     464MB
phpmyadmin   latest              5357004ce1de   7 days ago       477MB
node         latest              6817534de6bd   12 days ago      907MB
openjdk      11.0-slim           1a70900cdaaa   2 weeks ago      421MB
mysql        latest              0627ec6901db   2 weeks ago      556MB
node         14.16.1-alpine3.10  dad9ba894bb5   3 weeks ago      116MB
nginx        1.19.10-alpine      a64a6e03b055   3 weeks ago      22.6MB
nginx        latest              62d49f9bab67   3 weeks ago      133MB
mysql        8.0.23              cbe8815cbea8   4 weeks ago      546MB
caddy        2.0.0-alpine        313ad00e425a   10 months ago    40.3MB
maven        3.6.1-jdk-11-slim   b67032e30f89   21 months ago    418MB
maven        3.6.0-jdk-11-slim   c7428be691f8   2 years ago      489MB
```

3. Configuration/script files with explanations where required

3.1. Dockerfile(s)

Frontend

```
FROM node:latest as build-stage
WORKDIR /app
COPY package*.json ./
RUN npm install
COPY ./ .
RUN npm run build

FROM nginx as production-stage
RUN mkdir /app
COPY --from=build-stage /app/dist /app
COPY nginx.conf /etc/nginx/nginx.conf
EXPOSE 80
```

Backend

```
FROM maven:3.6.1-jdk-11-slim AS build
COPY src /workspace/src
COPY pom.xml /workspace
WORKDIR /workspace
RUN mvn clean install

FROM openjdk:11.0-slim
EXPOSE 3000
COPY --from=build /workspace/target/*.jar app.jar
ENTRYPOINT ["java","-jar","app.jar"]
```

Database

```
FROM mysql
COPY ./scripts ./scripts
ENV  MYSQL_ROOT_PASSWORD=Newnismo_2001
EXPOSE 3306
```

3.2. Docker compose file

Frontend :

```yaml
version: "3"
services:
    frontend:
        container_name: frontend
        build:
            context: .
        image: frontend

networks:
    default:
       external:
          name: allnetwork
```

Backend :

```yaml
version: "3"
services:
  backend:
    container_name: backend
    build: .
    image: backend
    environment:
      - MYSQL_DATABASE=carsdb
      - MYSQL_USER=root
      - MYSQL_ROOT_PASSWORD=Newnismo_2001
    volumes:
      - ~/backend_data:/public/product-img

networks:
  default:
    external:
      name: allnetwork

```

Database :

```yaml
version: '3'
services:
  database:
    container_name: database
    build: .
    image: database
    environment:
      MYSQL_ROOT_USER: root
      MYSQL_ROOT_PASSWORD: Newnismo_2001
      MYSQL_DATABASE: carsdb
    volumes:
      - "./scripts/carsdb.sql:/docker-entrypoint-initdb.d/carsdb.sql"
      - "./scripts/user_grant.sql:/docker-entrypoint-initdb.d/user_grant.sql"
    ports:
      - "3306:3306"

networks:
    default:
        external:
            name: allnetwork
```

3.3. Proxy configuration file

Docker-compose.yml

```yaml
version: '3'
services:
  reverseproxy:
    container_name: reverseproxy
    image: nginx
    volumes:
          - ./nginx.conf:/etc/nginx/conf.d/default.conf
    ports:
          - "80:80"
networks:
    default:
       external:
         name: allnetwork
```

Nginx.conf

```
server{
    listen 80;
    location / {
        proxy_pass http://172.20.0.3;
    }
    location /backend {
        proxy_pass http://172.20.0.4:3000/;
        }
}
```

3.4. other files that you use with explanation of what you did

.dockerignore(frontend)

Setting up the .dockerignore file prevents node_modules and any intermediate build artifacts from being copied to the image which can cause issues during building.

.env(frontend)

Set api to backend

Nginx.conf(frontend)

```
user  nginx;
worker_processes  1;
error_log  /var/log/nginx/error.log warn;
pid        /var/run/nginx.pid;
events {
    worker_connections  1024;
}
http {
    include       /etc/nginx/mime.types;
    default_type  application/octet-stream;
    log_format  main  '$remote_addr - $remote_user [$time_local] "$request" '
                      '$status $body_bytes_sent "$http_referer" '
                      '"$http_user_agent" "$http_x_forwarded_for"';
    access_log  /var/log/nginx/access.log  main;
    sendfile        on;
    keepalive_timeout  65;
    server {
        listen       80;
        server_name  localhost;
        location / {
            root   /app;
```

```
        index  index.html;
        try_files $uri $uri/ /index.html;
      }
      error_page   500 502 503 504  /50x.html;
      location = /50x.html {
        root   /usr/share/nginx/html;
      }
    }
}
```

Nginx is an HTTP(s) server that will run in your docker container. It uses a configuration file to determine how to serve content/which ports to listen on/etc. See the nginx configuration documentation for an example of all of the possible configuration options.

The following is a simple nginx configuration that serves your vue project on port 80. The root index.html is served for page not found / 404 errors which allows us to use pushState() based routing.

Application.properties(backend)

```
server.port=3000
spring.datasource.url=jdbc:mysql://172.20.0.2:3306/carsdb
spring.datasource.username=root
spring.datasource.password=Newnismo_2001
spring.datasource.platform=mysql
spring.datasource.driver-class-name=com.mysql.jdbc.Driver
spring.datasource.initialization-mode=always
spring.jpa.database-platform=org.hibernate.dialect.MySQL5InnoDBDialect

spring.servlet.multipart.enabled=true
spring.servlet.multipart.file-size-threshold=2KB
spring.servlet.multipart.max-file-size=200MB
spring.servlet.multipart.max-request-size=215MB

cars.storage.location=./public/product-img
cars.origin.host=http://104.215.186.188:8080,http://104.215.186.188,http://www.int221projectcars.games
cars.origin.method=GET,POST,PUT,HEAD,DELETE,PATCH,OPTIONS

```

Use to set origin host fom frontend and lock inner IP of MySQL database

4. Describe configurations that you did to set the environment for FE, BE, proxy

- **Frontend**

For Dockerfile we use the node image version latest and nginx image for create the image name's "frontend" and copy the environment from frontend source code to build the image. Use port 80 for container port.

For docker-compose.yml we build container name "frontend" with using image name's "frontend" and docker network name's "allnetwork".

- **Backend**

For Dockerfile we use maven image for mvn clean install and use openjdk for build image with .jar file that we got from mvn clean install. The image name's "backend". Use port 3000 for container port.

For docker-compose.yml we build container name "backend" with using image name's "backend" and build the environment with database connection. We kept the product-picture in the backend_data file.

And finally using docker network name's "allnetwork".

Application.properties we set server port for 3000 and set the database connect to inner IP network connection with platform mysql.

- **Database**

For Dockerfile we use mysql image name's "database" with container port 3306 and set mysql root password.

For docker-compose.yml we build container name's "database" with using image name's "database" and set database connection. We run scripts from the scripts file and set server port to 3306.

- **Proxy**

For docker-compose.yml we build the container name's "reverseproxy" with using image nginx and kept nginx.conf to default.conf in root. We use port 80 for server and port 80 for container.

For nginx.conf file use server port 80 go to proxy pass to home page website with the path "/" and proxy pass to backend page with path "/backend".