BUYO - System Manual

Table of Contents

Android

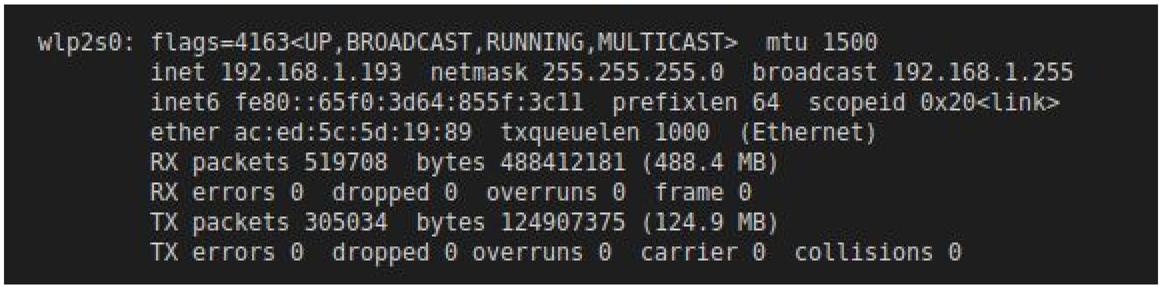
- You need to have an Android development environment that supports Kotlin. You need Android 5.0 and above (API Level 21). Suggested IDE is Android Studio 3.0 and up.
- When you have a suitable IDE, clone the project from Github Repository, open the source code in IDE. Then, Install a virtual device or connect an Android phone.
- No additional setup necessary to build and run BUYO app. All operations are common to usual Android project operations.

Backend

Requirements
- Docker
- Redis

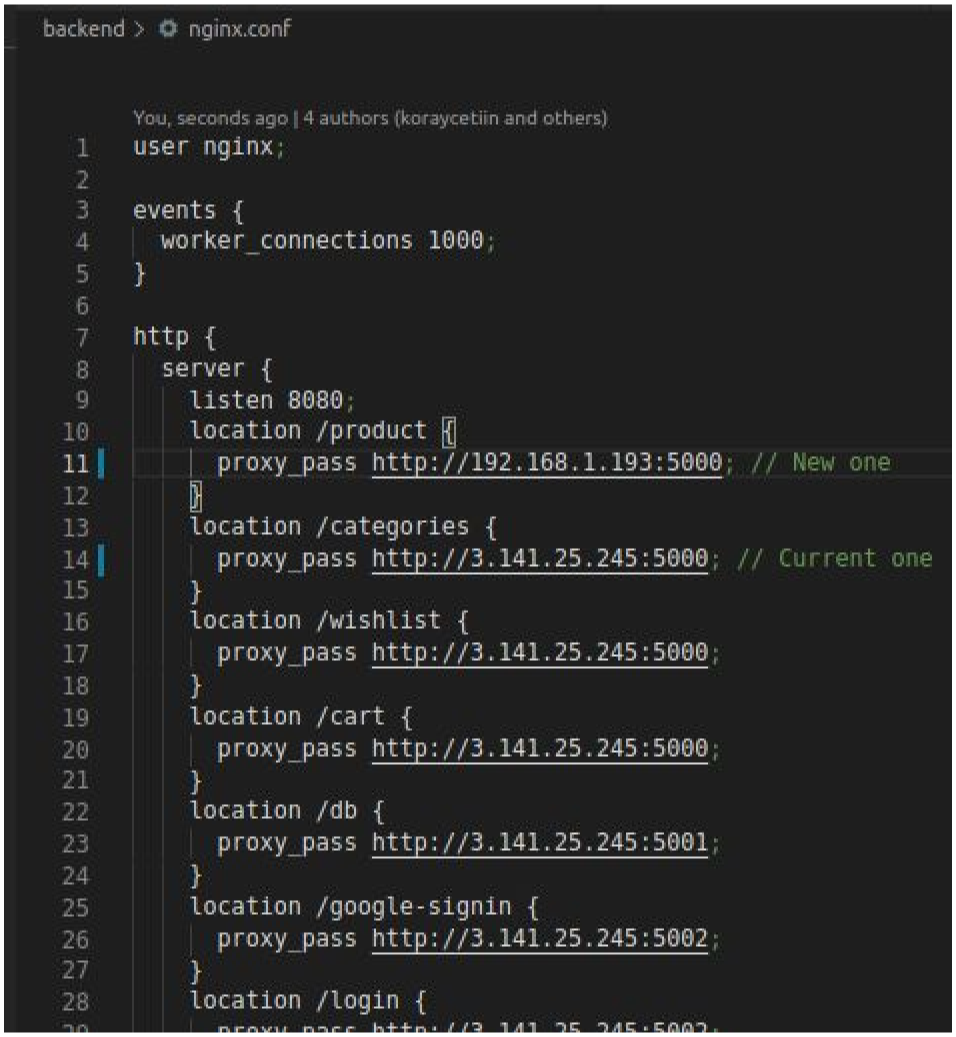
Deployment
1. Clone the repository from this url:
   git clone https://github.com/bounswe/bounswe2020group4.git
2. Go to backend directory by using this command
   cd ./bounswe2020group4/backend

3. It is very important to update nginx file with your current IP address.
   Check your ip address with this command: ifconfig



```
wlp2s0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.1.193  netmask 255.255.255.0  broadcast 192.168.1.255
        inet6 fe80::65f0:3d64:855f:3c11  prefixlen 64  scopeid 0x20<link>
        ether ac:ed:5c:5d:19:89  txqueuelen 1000  (Ethernet)
        RX packets 519708  bytes 488412181 (488.4 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 305034  bytes 124907375 (124.9 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

4. Type the value (192.168.1.193 for our case) instead of new one for every service:

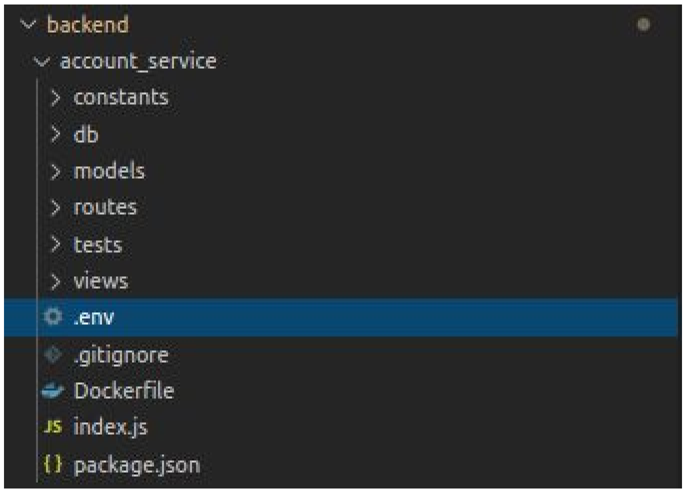


```
user nginx;

events {
  worker_connections 1000;
}

http {
  server {
    listen 8080;
    location /product {
      proxy_pass http://192.168.1.193:5000; // New one
    }
    location /categories {
      proxy_pass http://3.141.25.245:5000; // Current one
    }
    location /wishlist {
      proxy_pass http://3.141.25.245:5000;
    }
    location /cart {
      proxy_pass http://3.141.25.245:5000;
    }
    location /db {
      proxy_pass http://3.141.25.245:5001;
    }
    location /google-signin {
      proxy_pass http://3.141.25.245:5002;
    }
    location /login {
      proxy_pass http://3.141.25.245:5002;
```

5. Make sure that you added the .env files in each service



```
backend
  account_service
    constants
    db
    models
    routes
    tests
    views
    .env
    .gitignore
    Dockerfile
    index.js
    package.json
```

.env file :
PORT = 9876
MONGO_URL =
"mongodb+srv://buyo-dev:buyoapp123@buyo-app.aycr0.mongodb.net/buyo-app?retryWrites=true&w=majority"
TEST_MONGO_URL =
"mongodb+srv://buyo-dev:buyoapp123@buyo-app.aycr0.mongodb.net/test-db?retryWrites=true&w=majority"

Warning : For the **interaction service**, please add below to .env file as well:
REDIS_URL = "redis://cache"

6. Make sure that there is no package-lock.json in the services.
7. Install redis
    - on Mac

    brew install redis
    - on Ubuntu

    sudo apt install redis-server

8. Run redis
    - On Mac

    brew services start redis

- On Ubuntu

  redis-server

9. Build docker containers

   sudo docker-compose build

10. Make docker containers up and running

    sudo docker-compose up

Running Tests

1. Visit the root file of backend
2. Give execution permission to run_tests.sh

   chmod +x run_tests.sh
3. Run tests with the bash file

   ./run_tests.sh

Frontend

Running the application in local development mode:

1. Pull the repository to your local machine
2. Inside the repository, run 'npm install' to install necessary npm packages that is not stored on the git repository.
3. Then run 'npm start' to run the react application in development mode. This will serve the application on port 3000 (or some other port if it is being used) where you can access it from any web browser by going to the url localhost:3000

Deploying the application to 'http://ec2-3-17-180-100.us-east-2.compute.amazonaws.com': A CI/CD workflow is implemented using github actions. So when you make a commit or merge a pull request to the master branch, our github actions workflow will automatically do the necessary steps to deploy the application on our server.

Deploying the application to another machine:

- Pull the repository to your desired machine.
- Make sure there are no images named webimage or running containers remaining.
- Inside the repository, build the docker image with command: ‘docker build -t webimage web’
- Run the docker image with command: ‘sudo docker run -p 80:80 -d webimage’
- The application is now live on your machines port 80

The same steps can be followed to deploy the admin project (with different image names and different port if you like)

Linting:

- We are using the package eslint for linting. Eslint rules are configured in .eslintrc.json
- To use it normally: ./node_modules/.bin/eslint src
- To automatically fix most of the common cases: ./node_modules/.bin/eslint src --fix

Running the Packaged Web Application on the Submission Delivery Package:

- Unzip web-build file
- Run ‘npm install -g serve’ (You may need npm for this. Also on some platforms, you may need to grant access to some folders)
- Then run ‘serve -s build’
- You can follow the same steps for the admin application