Title: EMart App: A Microservice Showcase Leveraging NGINX and Docker

Abstract

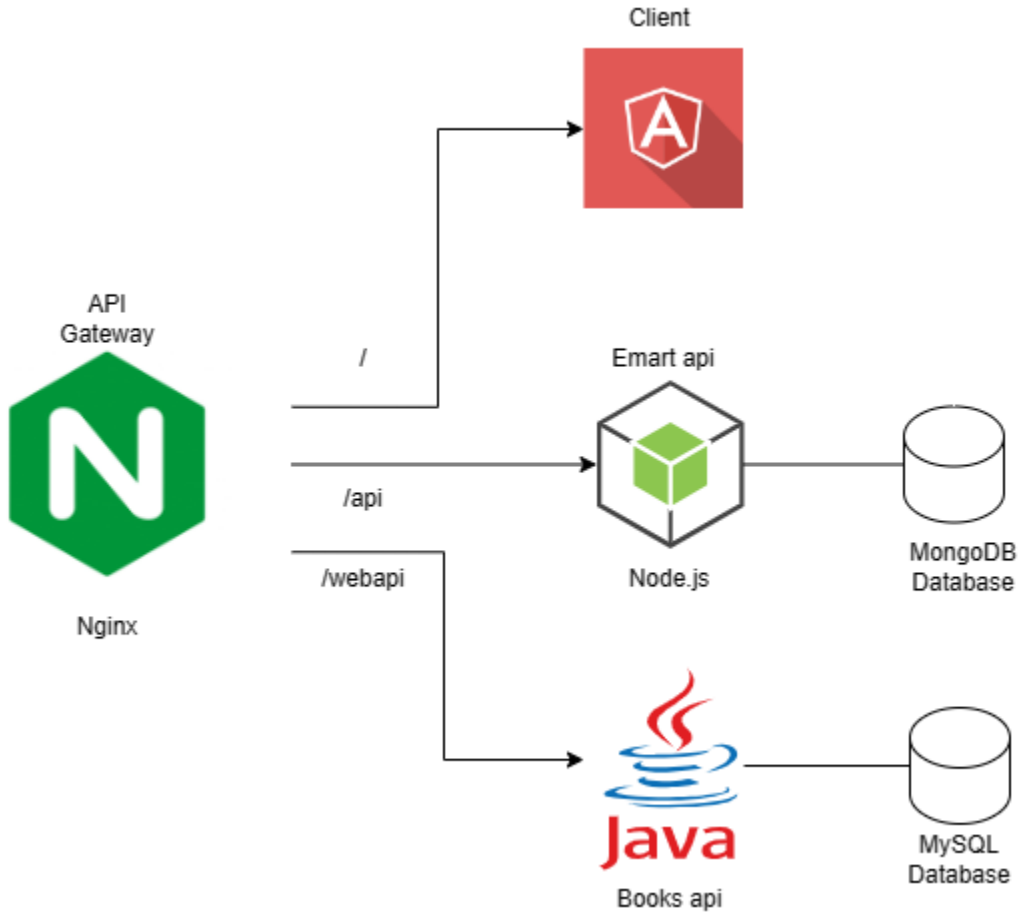
The EMart App demonstrates the power of microservice architecture in building a scalable and maintainable application. This report delves into the app's architecture, focusing on the NGINX API gateway and its role in routing traffic to independent microservices written in Node.js (EMart API) and Java (Books API). Each microservice interacts with its dedicated database (MongoDB for EMart API and MySQL for Books API), fostering loose coupling and independent scaling.

Introduction

Microservice architecture is an increasingly popular approach for building complex applications. It decomposes the application into smaller, self-contained services, each with well-defined responsibilities. This approach offers numerous benefits, including:

- **Scalability:** Individual microservices can be scaled independently based on their specific load requirements.
- **Maintainability:** Independent codebases with smaller footprints are easier to understand, maintain, and test.
- **Fault Isolation:** An issue in one microservice is less likely to cascade and impact the entire application.
- **Deployment Flexibility:** Microservices can be deployed and updated independently, streamlining the development and release process.

EMart App Architecture



The EMart App utilizes a layered architecture:

- **Frontend:** The user interface is built with Angular, a modern JavaScript framework.
- **API Gateway:** NGINX, a versatile web server and reverse proxy, acts as the API gateway. It listens on three predefined endpoints (/, /api, and /webapp) and routes incoming requests accordingly.
- **Microservices:**
    - EMart API: Developed in Node.js, this service handles EMart-related functionalities and interacts with the MongoDB database.
    - Books API: Written in Java, this service manages book-related operations and connects to the MySQL database.

Detailed Breakdown

- **API Gateway:** NGINX efficiently routes requests based on the path:
  - /: Redirects to the Angular frontend (client app).
  - /api: Forwards requests to the EMart API (Node.js) for EMart-specific operations.
  - /webapp: Directs requests to the Books API (Java) for book-related interactions.
- **EMart API (Node.js):** This service manages EMart-specific functionalities and connects to the MongoDB database.
- **Books API (Java):** This service handles book-related operations and communicates with the MySQL database.

Docker for Containerization

Docker empowers efficient containerization of the EMart App's microservices. The docker-compose.yaml file defines the service configuration, including dependencies and ports. Using docker-compose up -d, the developer can seamlessly bring up all containers from a single command.

Running the EMart App

1. **Clone the source code:**

   Bash

   git clone https://github.com/devopshydclub/emartapp.git

   Use code with caution.

   content_copy

2. **Navigate to the project directory:**

   Bash

   cd emartapp/

   Use code with caution.

   content_copy

3. **Review the docker-compose.yaml file using preferred code editor (e.g., Vim).**
4. **Start the containers in detached mode:**

   Bash

   docker-compose up -d

Use code with caution.

content_copy

5. **Verify running containers:**

   Bash

   ```
   docker compose ps  # List running containers
   docker ps -a       # List all containers, including stopped ones
   ```

   Use code with caution.

   content_copy

6. **Access the application:** Open a web browser and navigate to http://<VMIp>:80. Replace <VMIp> with the Virtual Machine IP address, which can be determined using ip addr show.

Cleaning Up:

1. **Stop running containers:**

   Bash

   ```
   docker compose down
   ```

   Use code with caution.

   content_copy

2. **Remove unused containers and networks:**

   Bash

   ```
   docker system prune -a   # Use with caution, may remove unwanted data
   ```

   Use code with caution.

   content_copy

Demo

```
aditi@ADITI MINGW64 ~/OneDrive/文档/DevOps
$ ls
 Microservice/                VProject.docx   vprofile-project/
'Microservice Project.docx'   VProject.pdf

aditi@ADITI MINGW64 ~/OneDrive/文档/DevOps
$ cd Microservice/

aditi@ADITI MINGW64 ~/OneDrive/文档/DevOps/Microservice
$ ls
MacOSM1Chip/  compose/  emartapp/  windowsAndMacIntel/

aditi@ADITI MINGW64 ~/OneDrive/文档/DevOps/Microservice
$ cd windowsAndMacIntel/

aditi@ADITI MINGW64 ~/OneDrive/文档/DevOps/Microservice/windowsAndMacIntel
$ ls
Vagrantfile

aditi@ADITI MINGW64 ~/OneDrive/文档/DevOps/Microservice/windowsAndMacIntel
$
```

```
aditi@ADITI MINGW64 ~/OneDrive/文档/DevOps/Microservice/windowsAndMacIntel
$ vagrant global-status --prune
id       name   provider   state    directory
-----------------------------------------------------------------------------
-----------------------------------------------
0768cb4  db01   virtualbox running  C:/Users/aditi/OneDrive/文档/DevOps/vprofile
-project/vagrant/Manual_provisioning_WinMacIntel
5e67979  mc01   virtualbox running  C:/Users/aditi/OneDrive/文档/DevOps/vprofile
-project/vagrant/Manual_provisioning_WinMacIntel
310ccfa  rmq01  virtualbox running  C:/Users/aditi/OneDrive/文档/DevOps/vprofile
-project/vagrant/Manual_provisioning_WinMacIntel
f99419c  app01  virtualbox running  C:/Users/aditi/OneDrive/文档/DevOps/vprofile
-project/vagrant/Manual_provisioning_WinMacIntel
9239b28  web01  virtualbox running  C:/Users/aditi/OneDrive/文档/DevOps/vprofile
-project/vagrant/Manual_provisioning_WinMacIntel
171a978  db01   virtualbox poweroff C:/Users/aditi/OneDrive/文档/DevOps/vprofile
-project/vagrant/Automated_provisioning_WinMacIntel

The above shows information about all known Vagrant environments
on this machine. This data is cached and may not be completely
up-to-date (use "vagrant global-status --prune" to prune invalid
entries). To interact with any of the machines, you can go to that
directory and run Vagrant, or you can use the ID directly with
Vagrant commands from any directory. For example:
"vagrant destroy 1a2b3c4d"
```

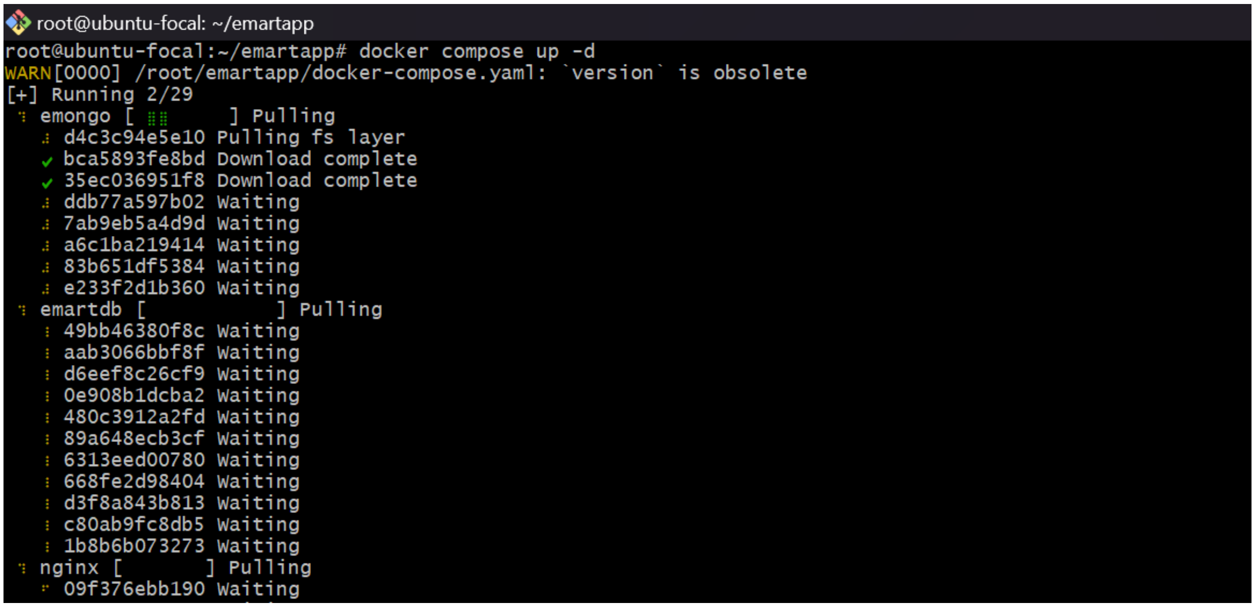
```
aditi@ADITI MINGW64 ~/OneDrive/文档/DevOps/Microservice/windowsAndMacIntel
$ vagrant up
Bringing machine 'default' up with 'virtualbox' provider...
==> default: Box 'ubuntu/focal64' could not be found. Attempting to find and ins
tall...
    default: Box Provider: virtualbox
    default: Box Version: >= 0
==> default: Loading metadata for box 'ubuntu/focal64'
    default: URL: https://vagrantcloud.com/ubuntu/focal64
==> default: Adding box 'ubuntu/focal64' (v20240531.0.0) for provider: virtualbo
x
    default: Downloading: https://vagrantcloud.com/ubuntu/boxes/focal64/versions
/20240531.0.0/providers/virtualbox/unknown/vagrant.box
==> default: Box download is resuming from prior download progress
Download redirected to host: cloud-images.ubuntu.com
    default:
==> default: Successfully added box 'ubuntu/focal64' (v20240531.0.0) for 'virtua
lbox'!
==> default: Importing base box 'ubuntu/focal64'...
==> default: Matching MAC address for NAT networking...
==> default: Checking if box 'ubuntu/focal64' version '20240531.0.0' is up to da
te...
==> default: Setting the name of the VM: windowsAndMacIntel_default_171751089059
3_71643
==> default: Clearing any previously set network interfaces...
==> default: Preparing network interfaces based on configuration...
    default: Adapter 1: nat
    default: Adapter 2: hostonly
    default: Adapter 3: bridged
==> default: Forwarding ports...
```

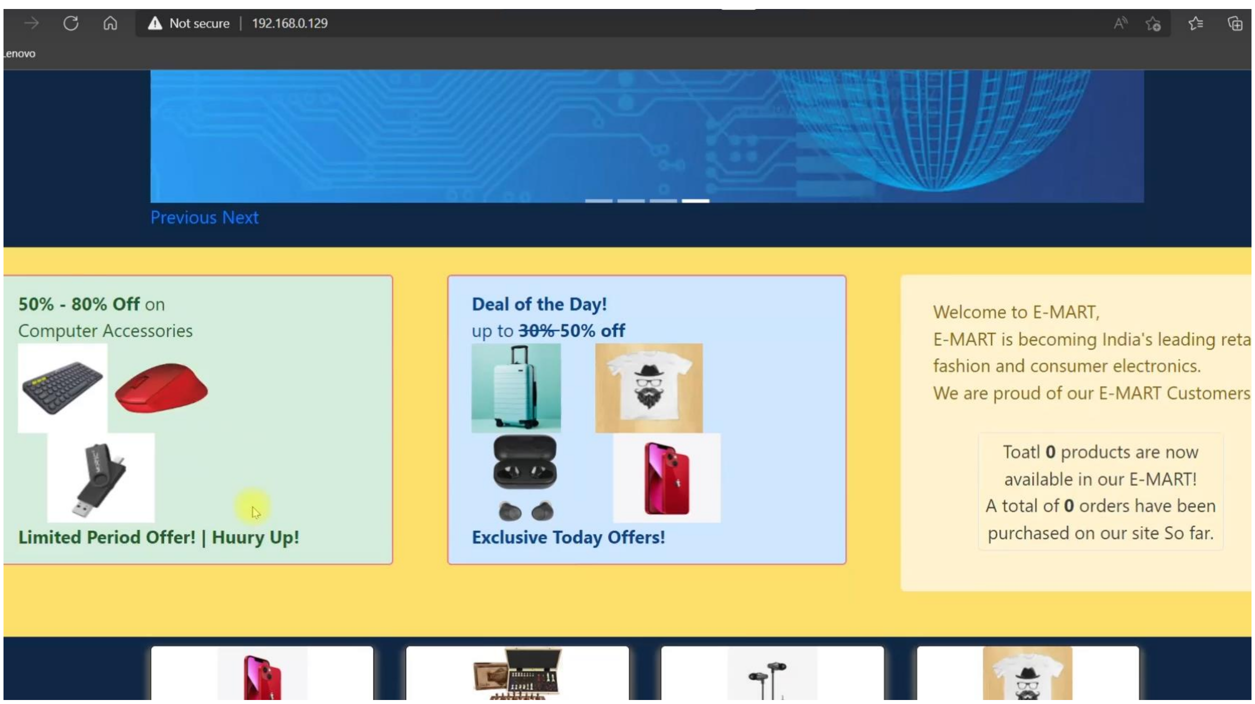
```
aditi@ADITI MINGW64 ~/OneDrive/文档/DevOps/Microservice/windowsAndMacIntel
$ vagrant ssh
Welcome to Ubuntu 20.04.6 LTS (GNU/Linux 5.4.0-182-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/pro

 System information as of Tue Jun  4 15:15:54 UTC 2024

  System load:             0.37
  Usage of /:              5.4% of 38.70GB
  Memory usage:            14%
  Swap usage:              0%
  Processes:               127
  Users logged in:         0
  IPv4 address for enp0s9: 192.168.140.237
  IPv6 address for enp0s9: 2409:4080:9107:24b9:a00:27ff:fe6e:d43


Expanded Security Maintenance for Applications is not enabled.
```

```
vagrant@ubuntu-focal:~$ sudo -i
root@ubuntu-focal:~# cd emartapp/
-bash: cd: emartapp/: No such file or directory
root@ubuntu-focal:~# ls
snap
root@ubuntu-focal:~# git clone https://github.com/devopshydclub/emartapp.git
Cloning into 'emartapp'...
remote: Enumerating objects: 333, done.
remote: Counting objects: 100% (75/75), done.
remote: Compressing objects: 100% (51/51), done.
remote: Total 333 (delta 33), reused 24 (delta 24), pack-reused 258
Receiving objects: 100% (333/333), 3.80 MiB | 1.26 MiB/s, done.
Resolving deltas: 100% (46/46), done.
root@ubuntu-focal:~# cd emartapp/
root@ubuntu-focal:~/emartapp# ls
Dockerfile   README.md  docker-compose.yaml  kkartchart  nodeapi
Jenkinsfile  client     javaapi              nginx       package-lock.json
root@ubuntu-focal:~/emartapp# vim docker-compose.yaml
```

root@ubuntu-focal: ~/emartapp

```
version: "3.8"

services:
  client:
    build:
      context: ./client
    ports:
      - "4200:4200"
    container_name: client
    depends_on:
      - api
      - webapi

  api:
    build:
      context: ./nodeapi
    ports:
      - "5000:5000"
    restart: always
    container_name: api
    depends_on:
      - nginx
      - emongo

  webapi:
    build:
      context: ./javaapi
    ports:
      - "9000:9000"
    restart: always
    container_name: webapi
    depends_on:
      - emartdb

  nginx:
    restart: always
    image: nginx:latest
    container_name: nginx
    volumes:
      - "./nginx/default.conf:/etc/nginx/conf.d/default.conf"
    ports:
      - "80:80"

  emongo:
    image: mongo:4
    container_name: emongo
    environment:
      - MONGO_INITDB_DATABASE=epoc
    ports:
      - "27017:27017"

  emartdb:
    image: mysql:8.0.33
    container_name: emartdb
    ports:
      - "3306:3306"
    environment:
```


root@ubuntu-focal: ~/emartapp
root@ubuntu-focal:~/emartapp# docker compose up -d
WARN[0000] /root/emartapp/docker-compose.yaml: `version` is obsolete
[+] Running 2/29
emongo [ ] Pulling
d4c3c94e5e10 Pulling fs layer
bca5893fe8bd Download complete
35ec036951f8 Download complete
ddb77a597b02 Waiting
7ab9eb5a4d9d Waiting
a6c1ba219414 Waiting
83b651df5384 Waiting
e233f2d1b360 Waiting
emartdb [ ] Pulling
49bb46380f8c Waiting
aab3066bbf8f Waiting
d6eef8c26cf9 Waiting
0e908b1dcba2 Waiting
480c3912a2fd Waiting
89a648ecb3cf Waiting
6313eed00780 Waiting
668fe2d98404 Waiting
d3f8a843b813 Waiting
c80ab9fc8db5 Waiting
1b8b6b073273 Waiting
nginx [ ] Pulling
09f376ebb190 Waiting


Not secure | 192.168.0.129
Previous Next
50% - 80% Off on
Computer Accessories
Limited Period Offer! | Huury Up!
Deal of the Day!
up to 30% 50% off
Exclusive Today Offers!
Welcome to E-MART,
E-MART is becoming India's leading reta
fashion and consumer electronics.
We are proud of our E-MART Customers
Toatl 0 products are now
available in our E-MART!
A total of 0 orders have been
purchased on our site So far.

Conclusion

The EMart App effectively showcases the benefits of a microservice architecture. NGINX's flexibility as an API gateway seamlessly routes requests to independent services built with Node.js and Java. The use of Docker simplifies deployment and management. By leveraging these principles, developers can create applications that are scalable, maintainable, and resilient.