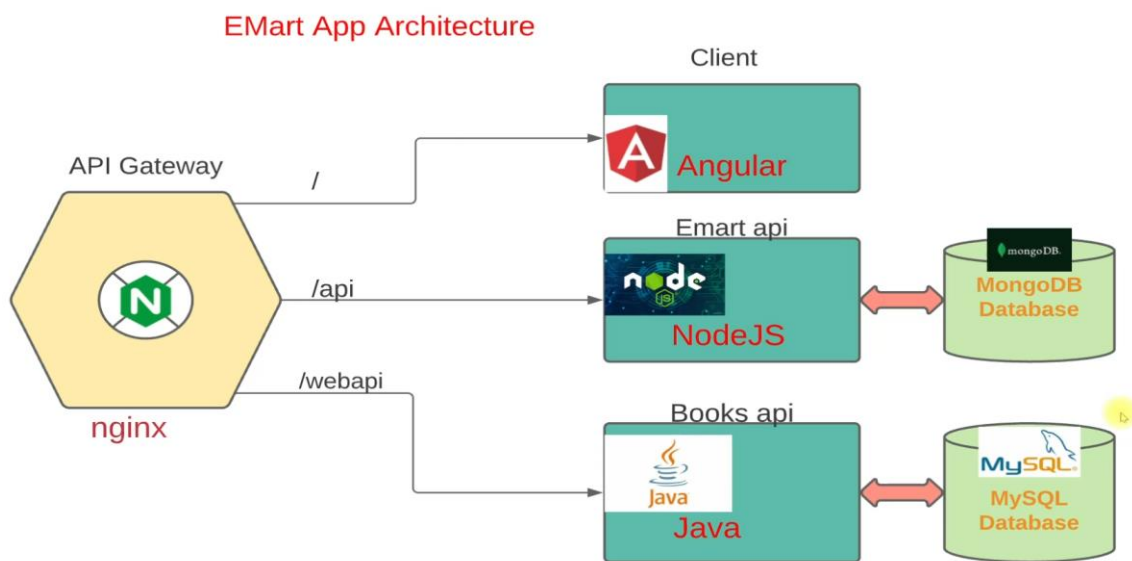


Container Microservice Architecture

This project explains the setup of container Microservice architecture, there are 4 services used in this project api gateway which is in nginx service, it listens for request and route based on the url. If the request is on root, then the Angular client home page will be loading, for backend we have two api's Nodejs and Java, and two databases one for Mongo Db and one for MySQL DB. Active Containers living in the project are nginx, Angular client, Node JS, Java Api, Mongo Db and MySQL DB.



1. Launch a EC2 instance with user data for Docker, Docker compose and add user to the dockergroup.

EC2 > Instances > Launch an instance

Launch an instance [Info](#)

Amazon EC2 allows you to create virtual machines, or instances, that run on the AWS Cloud. Quickly get started by following the simple steps below.

Name and tags [Info](#)

Name

MicroserviceContainer

Add additional tags

▼ Application and OS Images (Amazon Machine Image) [Info](#)

An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. Search or Browse for AMIs if you don't see what you are looking for below

Q Search our full catalog including 1000s of application and OS images

Recents

Quick Start

Amazon Linux

aws

macOS

Mac

Ubuntu

ubuntu

Windows

Microsoft

Red Hat

RedHat

⋮

Browse more AMIs

Including AMIs from AWS, Marketplace and the Community

Amazon Machine Image (AMI)

Ubuntu Server 22.04 LTS (HVM), SSD Volume Type

ami-08c40ec9ead489470 (64-bit (x86)) / ami-0f69dd1d0d03ad669 (64-bit (Arm))

Virtualization: hvm ENA enabled: true Root device type: ebs

Free tier eligible

Description

Canonical, Ubuntu, 22.04 LTS, amd64 jammy image build on 2022-09-12

Architecture

AMI ID

64-bit (x86)

ami-08c40ec9ead489470

Verified provider

▼ Summary

Number of instances [Info](#)

1

Software Image (AMI)

Canonical, Ubuntu, 22.04-ami-08c40ec9ead489470

Virtual server type (instance type)

t2.large

Firewall (security group)

New security group

Storage (volumes)

1 volume(s) - 20 GiB

Free tier: In your first hours of t2.micro (or Regions in which t2 instance usage on first month, 30 GiB of EBS I/Os, 1 GB of snapshot bandwidth to the internet)

Cancel

▼ Network settings

Info

Edit

Network

Info

vpc-0f7768310869e62ba | DefaultVPC

Subnet

Info

No preference (Default subnet in any availability zone)

Auto-assign public IP

Info

Enable

Firewall (security groups)

Info

A security group is a set of firewall rules that control the traffic for your instance. Add rules to allow specific traffic to reach your instance.

Create security group

Select existing security group

We'll create a new security group called 'launch-wizard-1' with the following rules:

✓

Allow SSH traffic from

Helps you connect to your instance

Anywhere

0.0.0.0/0

☐

Allow HTTPS traffic from the internet

To set up an endpoint, for example when creating a web server

✓

Allow HTTP traffic from the internet

To set up an endpoint, for example when creating a web server

⚠

Rules with source of 0.0.0.0/0 allow all IP addresses to access your instance. We recommend setting security group rules to allow access from known IP addresses only.

×

▼ Configure storage

Info

Advanced

1x

25

GiB

gp2

Root volume (Not encrypted)

Free tier eligible customers can get up to 30 GB of EBS General Purpose (SSD) or Magnetic storage

×

Add the below user data,

```
#!/bin/bash

# Install docker on Ubuntu
sudo apt-get update
sudo apt-get install \
  ca-certificates \
  curl \
  gnupg \
  lsb-release -y
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /usr/share/keyrings/docker-archive-keyring.gpg
echo \
"deb [arch=$(dpkg --print-architecture) signed-by=/usr/share/keyrings/docker-archive-keyring.gpg] https://download.docker.com/linux/ubuntu \
$(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null

# Install docker-compose
sudo apt-get update
sudo apt-get install docker-ce docker-ce-cli containerd.io -y
sudo curl -L "https://github.com/docker/compose/releases/download/1.29.2/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose
sudo chmod +x /usr/local/bin/docker-compose

# Add ubuntu user into docker group
sudo usermod -a -G docker ubuntu
```

2. Login to the ubuntu machine

```
ubuntu@ip-172-31-27-42:~$ id
uid=1000(ubuntu) gid=1000(ubuntu) groups=1000(ubuntu),4(adm),20(dialout),24(cdrom),25(floppy),27(sudo),29(audio),30(dip),44(video),46(plugdev),118(netdev),119(lxd),999(docker)
ubuntu@ip-172-31-27-42:~$ docker-compose --version
docker-compose version 1.29.2, build 5becea4c
ubuntu@ip-172-31-27-42:~$
```

3. Clone the source code from Git repo, and execute docker-compose build <https://github.com/devopshydclub/emartapp.git>

```
docker-compose version 1.29.2, build 5becea4c
ubuntu@ip-172-31-27-42:~$ git clone https://github.com/devopshydclub/emartapp.git
Cloning into 'emartapp'...
remote: Enumerating objects: 310, done.
remote: Counting objects: 100% (310/310), done.
remote: Compressing objects: 100% (251/251), done.
remote: Total 310 (delta 36), reused 301 (delta 31), pack-reused 0
Receiving objects: 100% (310/310), 3.78 MiB | 40.34 MiB/s, done.
Resolving deltas: 100% (36/36), done.
ubuntu@ip-172-31-27-42:~$ ls
emartapp
ubuntu@ip-172-31-27-42:~$ cd emartapp/
ubuntu@ip-172-31-27-42:~/emartapp$ ls
Dockerfile  README.md  docker-compose.yaml  kkartchart  nodeapi
Jenkinsfile  client  javaapi  nginx  package-lock.json
ubuntu@ip-172-31-27-42:~/emartapp$ docker-compose build
```

4. Show the docke images created in the machine,

```
# Use official nginx image as the base image
FROM nginx:latest

# Copy the build output to replace the default nginx contents.
COPY --from=web-build /usr/src/app/client/dist/client/ /usr/share/nginx/html
COPY nginx.conf /etc/nginx/conf.d/default.conf

# Expose port 4200
EXPOSE 4200
ubuntu@ip-172-31-27-42:~/emartapp/client$ cd ..
ubuntu@ip-172-31-27-42:~/emartapp$ docker images
REPOSITORY          TAG             IMAGE ID        CREATED         SIZE
emartapp_client      latest          a6b345c726ac   13 minutes ago  148MB
<none>               <none>          8bbe3a096924   13 minutes ago  1.44GB
emartapp_webapi      latest          c626454cb7de   15 minutes ago  564MB
<none>               <none>          917527bb6c1f   15 minutes ago  957MB
emartapp_api         latest          1b18259e1408   16 minutes ago  934MB
<none>               <none>          b8891e719a82   16 minutes ago  939MB
node                 14             7a8bb5c30f83   4 days ago     916MB
nginx                latest          88736fe82739   4 days ago     142MB
openjdk              8              b273004037cc   3 months ago   526MB
ubuntu@ip-172-31-27-42:~/emartapp$
```

```

emartapp_api    latest    1b18259e1408    16 minutes ago    934MB
<none>         <none>    b8891e719a82    16 minutes ago    939MB
node           14       7a8bb5c30f83    4 days ago        916MB
nginx          latest    88736fe82739    4 days ago        142MB
openjdk        8        b273004037cc    3 months ago      526MB
ubuntu@ip-172-31-27-42:~/emartapp$ docker-compose up
Creating network "emartapp_default" with the default driver
Pulling emongo (mongo:4)...
4: Pulling from library/mongo
eaead16dc43b: Pull complete
8a00eb9f68a0: Pull complete
f683956749c5: Pull complete
b33b2f05ea20: Pull complete
3a342baa915a: Pull complete

```

5. All the required docker instances are up and running, and the web application is loading from the EC2 machine.

