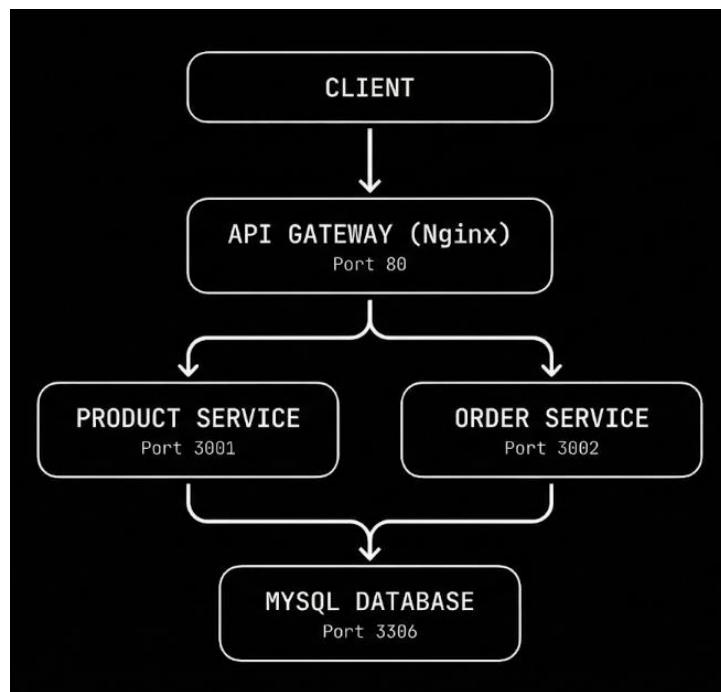


NAMA : FIDELIA PING
NIM : 245410012
KELAS : INFORMATIKA 1

MODUL 13

Implementasi Microservices Sederhana dengan Docker di AWS EC2

DASAR TEORI



- **VM 1 (Server - docker-host)**: Menjalankan 4 container:
 1. **API Gateway**: Pintu masuk tunggal (menggunakan Nginx).
 2. **Service 1 (Product Service)**: Menangani data produk.
 3. **Service 2 (Order Service)**: Menangani pemesanan.
 4. **Database**: MySQL (Shared database untuk kesederhanaan praktikum, atau dedicated untuk service tertentu).
- **VM 2 (Client - client-tester)**: Digunakan untuk melakukan request HTTP (testing) ke VM 1.

PRAKTIKUM

Bagian 1: Provisioning Infrastruktur (AWS EC2)

Langkah 1: Membuat Instance EC2

Buat 2 Instance EC2 dengan spesifikasi berikut:

- AMI: Ubuntu Server 22.04 LTS (Free Tier).
- Instance Type: t2.micro (Free Tier).

Langkah 2: Konfigurasi Security Group (Firewall)

Ini adalah langkah krusial agar VM 2 bisa bicara ke VM 1.

1. Security Group untuk VM 1 (Server):

The screenshot shows the 'Edit aturan masuk' (Inbound Rules) page for a security group named 'sg-07be91e2dfadc0525'. There are five rules listed:

- SSH (TCP port 22) - Sumber: Kustom (0.0.0.0/0)
- HTTP (TCP port 80) - Sumber: IPv4-Di... (0.0.0.0/0)
- MySQL/Aurora (TCP port 3306) - Sumber: IPv4-Di... (0.0.0.0/0)
- ICMP Kustom - IPv4 - Sumber: Semua (0.0.0.0/0)
- Another ICMP rule - Sumber: IPv4-Di... (0.0.0.0/0)

A warning message at the bottom states: 'Aturan dengan sumber 0.0.0.0/0 atau ::/0 memungkinkan semua alamat IP mengakses instans Anda. Sebaiknya atur aturan grup keamanan untuk mengizinkan akses hanya dari alamat IP yang diketahui.'

2. Security Group untuk VM 2 (Client):

The screenshot shows the 'Edit aturan masuk' (Inbound Rules) page for a security group named 'sg-0ed168f2698cf6afa'. There are two rules listed:

- SSH (TCP port 22) - Sumber: Kustom (0.0.0.0/0)
- ICMP Kustom - IPv4 - Sumber: Semua (0.0.0.0/0)

A warning message at the bottom states: 'Aturan dengan sumber 0.0.0.0/0 atau ::/0 memungkinkan semua alamat IP mengakses instans Anda. Sebaiknya atur aturan grup keamanan untuk mengizinkan akses hanya dari alamat IP yang diketahui.'

Bagian 2: Konfigurasi VM 1 (Server)

Login ke VM 1 menggunakan EC2 Instance Connect:

The screenshot shows the 'Hubungkan ke instans' (Connect to instance) page for an instance with ID 'i-0a7ef532db89a6b20'. The 'EC2 Instance Connect' tab is selected. The 'ID Instans' section shows the instance ID. The 'Tipe koneksi' section has two options:

- Hubungkan menggunakan IP Publik (selected)
- Hubungkan menggunakan IP Privat

The 'Alamat IPv4 publik' section shows the IP address '34.228.165.170'. The 'Nama pengguna' section has the name 'ubuntu' entered. A note at the bottom says: 'Catatan: Dalam kebanyakan kasus, nama pengguna default, ubuntu, benar. Namun, baca instruksi penggunaan AMI Anda untuk memeriksa apakah pemilik AMI telah mengubah nama pengguna AMI default.' At the bottom right are 'Batalkan' and 'Hubungkan' buttons.

Langkah 1: Install Docker & Docker Compose

Jalankan perintah berikut baris demi baris untuk menginstal Docker engine:

Update dan Install Dependencies

```
sudo apt-get update  
sudo apt-get install -y ca-certificates curl gnupg
```

```
ubuntu@ip-172-31-24-48:~$ sudo apt-get update  
Get:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy InRelease [128 kB]  
Get:2 http://security.ubuntu.com/ubuntu jammy-updates InRelease [129 kB]  
Get:3 http://security.ubuntu.com/ubuntu jammy-security InRelease [127 kB]  
Get:4 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-backports InRelease [127 kB]  
Get:5 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy/universe amd64 Packages [14.1 MB]  
Get:6 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy/universe Translation-en [5652 kB]  
Get:7 http://security.ubuntu.com/ubuntu jammy-security/main amd64 Packages [2900 kB]  
Get:8 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy/universe amd64 c-n-f Metadata [286 kB]  
Get:9 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy/multiverse amd64 Packages [217 kB]  
Get:10 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy/multiverse Translation-en [112 kB]  
Get:11 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy/multiverse amd64 c-n-f Metadata [8372 B]  
Get:12 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 Packages [3162 kB]  
Get:13 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/main Translation-en [485 kB]  
Get:14 http://security.ubuntu.com/ubuntu jammy-security/main Translation-en [417 kB]  
Get:15 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 c-n-f Metadata [19.1 kB]  
Get:16 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/restricted amd64 Packages [5043 kB]  
Get:17 http://security.ubuntu.com/ubuntu jammy-security/main amd64 c-n-f Metadata [14.0 kB]  
Get:18 http://security.ubuntu.com/ubuntu jammy-security/restricted amd64 Packages [4983 kB]  
Get:19 http://security.ubuntu.com/ubuntu jammy-security/restricted Translation-en [917 kB]  
Get:20 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/restricted Translation-en [944 kB]  
Get:21 http://security.ubuntu.com/ubuntu jammy-security/restricted amd64 c-n-f Metadata [652 B]  
Get:22 http://security.ubuntu.com/ubuntu jammy-security/universe amd64 Packages [1007 kB]  
Get:23 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/restricted amd64 c-n-f Metadata [644 B]  
Get:24 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/universe amd64 Packages [1245 kB]  
Get:25 http://security.ubuntu.com/ubuntu jammy-security/universe Translation-en [221 kB]  
Get:26 http://security.ubuntu.com/ubuntu jammy-security/universe amd64 c-n-f Metadata [22.4 kB]  
Get:27 http://security.ubuntu.com/ubuntu jammy-security/multiverse amd64 Packages [50.5 kB]  
Get:28 http://security.ubuntu.com/ubuntu jammy-security/multiverse Translation-en [10.2 kB]  
Get:29 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/main Translation-en [485 kB]
```

i-0a7ef532db89a6b20 (Vm-Server docker host)

PublicIPs: 34.228.165.170 PrivateIPs: 172.31.24.48

```
ubuntu@ip-172-31-24-48:~$ sudo apt-get install -y ca-certificates curl gnupg  
Reading package lists... Done  
Building dependency tree... Done  
Reading state information... Done  
ca-certificates is already the newest version (20240203-22.04.1).  
ca-certificates is set to manually installed.  
curl is already the newest version (7.81.0-1ubuntu1.21).  
curl is set to manually installed.  
The following additional packages will be installed:  
  dimngr gnupg-110 gnupg-utils gpg gpg-agent gpg-wks-client gpgconf gpgsm gpgv  
Suggested packages:  
  pinentry-gnome3 tor parmcionne xloadimage scdaemon  
The following packages will be upgraded:  
  dimngr gnupg gnupg-110 gnupg-utils gpg gpg-agent gpg-wks-client gpg-wks-server gpgconf gpgsm gpgv  
11 upgraded, 0 newly installed, 0 to remove and 39 not upgraded.  
Need to get 2248 kB of archives.  
After this operation, 0 B of additional disk space will be used.  
Get:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 gpg-wks-client amd64 2.2.27-3ubuntu2.5 [62.7 kB]  
Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 dimngr amd64 2.2.27-3ubuntu2.5 [293 kB]  
Get:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 gpg-wks-server amd64 2.2.27-3ubuntu2.5 [57.6 kB]  
Get:4 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 gnupg-utils amd64 2.2.27-3ubuntu2.5 [309 kB]  
Get:5 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 gpg-agent amd64 2.2.27-3ubuntu2.5 [209 kB]  
Get:6 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 gpgconf amd64 2.2.27-3ubuntu2.5 [109 kB]  
Get:7 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 gnupg-110 all 2.2.27-3ubuntu2.5 [94.3 kB]  
Get:8 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 gnupg-all 2.2.27-3ubuntu2.5 [54.5 kB]  
Get:9 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 gpgsm amd64 2.2.27-3ubuntu2.5 [197 kB]  
Get:10 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 gpgv amd64 2.2.27-3ubuntu2.5 [137 kB]  
Fetched 2248 kB in 0s (41.6 MB/s)
```

i-0a7ef532db89a6b20 (Vm-Server docker host)

PublicIPs: 34.228.165.170 PrivateIPs: 172.31.24.48

Setup GPG Key Docker (Jalankan per blok) Buat direktori keyring:

```
sudo install -m 0755 -d /etc/apt/keyrings  
ubuntu@ip-172-31-24-48:~$ sudo install -m 0755 -d /etc/apt/keyrings  
ubuntu@ip-172-31-24-48:~$ 
```

i-0a7ef532db89a6b20 (Vm-Server docker host)

PublicIPs: 34.228.165.170 PrivateIPs: 172.31.24.48

Download key (pastikan copy seluruh baris ini sekaligus):

```
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o  
/etc/apt/keyrings/docker.gpg  
ubuntu@ip-172-31-24-48:~$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /etc/apt/keyrings/docker.gpg  
ubuntu@ip-172-31-24-48:~$ 
```

i-0a7ef532db89a6b20 (Vm-Server docker host)

PublicIPs: 34.228.165.170 PrivateIPs: 172.31.24.48

Ubah permission file key:

```
sudo chmod a+r /etc/apt/keyrings/docker.gpg
ubuntu@ip-172-31-24-48:~$ sudo chmod a+r /etc/apt/keyrings/docker.gpg
ubuntu@ip-172-31-24-48:~$
```

i-0a7ef532db89a6b20 (Vm-Server docker host)

PublicIPs: 34.228.165.170 PrivateIPs: 172.31.24.48

Tambahkan Repository Docker Copy dan jalankan perintah panjang ini sebagai satu kesatuan:

```
echo "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.gpg] https://download.docker.com/linux/ubuntu $(. /etc/os-release & echo "$VERSION_CODENAME") stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
ubuntu@ip-172-31-24-48:~$ echo "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.gpg] https://download.docker.com/linux/ubuntu $(. /etc/os-release & echo "$VERSION_CODENAME") stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
ubuntu@ip-172-31-24-48:~$
```

i-0a7ef532db89a6b20 (Vm-Server docker host)

PublicIPs: 34.228.165.170 PrivateIPs: 172.31.24.48

Install Docker Engine Update ulang repository (wajib setelah menambah repo baru):

```
sudo apt-get update
ubuntu@ip-172-31-24-48:~$ sudo apt-get update
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy InRelease
Hit:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates InRelease
Hit:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-backports InRelease
Get:4 https://download.docker.com/linux/ubuntu jammy InRelease [48.5 kB]
Get:5 http://security.ubuntu.com/ubuntu jammy-security InRelease [129 kB]
Get:6 https://download.docker.com/linux/ubuntu jammy/stable amd64 Packages [66.2 kB]
Get:7 http://security.ubuntu.com/ubuntu jammy-security/main amd64 c-n-f Metadata [14.0 kB]
Get:8 http://security.ubuntu.com/ubuntu jammy-security/universe amd64 c-n-f Metadata [22.4 kB]
Fetched 280 kB in 1s (279 kB/s)
Reading package lists... Done
ubuntu@ip-172-31-24-48:~$
```

i-0a7ef532db89a6b20 (Vm-Server docker host)

PublicIPs: 34.228.165.170 PrivateIPs: 172.31.24.48

Install paket Docker:

```
sudo apt-get install -y docker-ce docker-ce-cli containerd.io docker-buildx-plugin
docker-compose-plugin
ubuntu@ip-172-31-24-48:~$ sudo apt-get install -y docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin
Reading package list... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  docker-ce-rootless-extras libslirp0 pigz slirp4netns
Suggested packages:
  cgroups-mount cgroup-lite docker-model-plugin
The following NEW packages will be installed:
  containerd.io docker-buildx-plugin docker-ce docker-ce-rootless-extras docker-compose-plugin libslirp0 pigz slirp4netns
0 upgraded, 9 newly installed, 0 to remove and 39 not upgraded.
Need to get 91.3 MB of archives.
After this operation, 364 MB of additional disk space will be used.
Get:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy/universe amd64 pigz amd64 2.6-1 [63.6 kB]
Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy/main amd64 libslirp0 amd64 4.6.1-build1 [61.5 kB]
Get:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy/universe amd64 slirp4netns amd64 1.0+git-2 [28.2 kB]
Get:4 https://download.docker.com/linux/ubuntu jammy/stable amd64 containerd.io amd64 1.2.1-1~jammy [23.4 kB]
Get:5 https://download.docker.com/linux/ubuntu jammy/stable amd64 docker-buildx-plugin amd64 5:29.1.3-1~ubuntu.22.04-jammy [16.3 MB]
Get:6 https://download.docker.com/linux/ubuntu jammy/stable amd64 docker-ce amd64 5:29.1.3-1~ubuntu.22.04-jammy [21.1 MB]
Get:7 https://download.docker.com/linux/ubuntu jammy/stable amd64 docker-ce-rootless-extras amd64 5:29.1.3-1~ubuntu.22.04-jammy [16.4 MB]
Get:8 https://download.docker.com/linux/ubuntu jammy/stable amd64 docker-ce-rootless-extras amd64 5:29.1.3-1~ubuntu.22.04-jammy [6385 kB]
Get:9 https://download.docker.com/linux/ubuntu jammy/stable amd64 docker-compose-plugin amd64 5:0.1-1~ubuntu.22.04-jammy [7714 kB]
Fetched 91.3 MB in 1s (79.6 MB/s)
Selecting previously unselected package containerd.io.
(Reading database ... 65993 files and directories currently installed.)
Preparing to unpack .../0-containerd.io_2.2.1-1~ubuntu.22.04-jammy_amd64.deb ...
Unpacking containerd.io (2.2.1-1~ubuntu.22.04-jammy) ...
Selecting previously unselected package docker-ce-cli.
Preparing to unpack .../1-docker-ce-cli_53a329.1.3-1~ubuntu.22.04-jammy_amd64.deb ...

```

i-0a7ef532db89a6b20 (Vm-Server docker host)

PublicIPs: 34.228.165.170 PrivateIPs: 172.31.24.48

Konfigurasi User Permission Tambahkan user ubuntu ke grup docker agar tidak perlu ketik sudo terus menerus:

```
sudo usermod -aG docker $USER  
ubuntu@ip-172-31-24-48:~$ sudo usermod -aG docker $USER  
ubuntu@ip-172-31-24-48:~$
```

i-0a7ef532db89a6b20 (Vm-Server docker host)

PublicIPs: 34.228.165.170 PrivateIPs: 172.31.24.48

Penting: Setelah perintah ini, Logout dari terminal lalu Login kembali agar perubahan grup aktif.

Langkah 2: Clone Project Microservice

Jalankan perintah di bawah ini untuk cloning repository yang sudah di sediakan.

```
git clone https://github.com/akbarwjyy/microstore.git  
cd microstore  
ubuntu@ip-172-31-24-48:~$ git clone https://github.com/akbarwjyy/microstore.git  
Cloning into 'microstore'...  
remote: Enumerating objects: 16, done.  
remote: Counting objects: 100% (16/16), done.  
remote: Compressing objects: 100% (12/12), done.  
remote: Total 16 (delta 2), reused 16 (delta 2), pack-reused 0 (from 0)  
Receiving objects: 100% (16/16), 5.13 KiB | 1.28 MiB/s, done.  
Resolving deltas: 100% (2/2), done.  
ubuntu@ip-172-31-24-48:~$ cd microstore/  
ubuntu@ip-172-31-24-48:~/microstore$
```

i-0a7ef532db89a6b20 (Vm-Server docker host)

PublicIPs: 34.228.165.170 PrivateIPs: 172.31.24.48

Langkah 3: Membuat Docker Network (Penting)

Karena kita akan menjalankan container secara terpisah, kita harus membuat Network Bridge secara manual agar container satu bisa "memanggil" container lainnya (misal: API Gateway memanggil Product Service).

```
docker network create micro-ne  
ubuntu@ip-172-31-24-48:~/microstore$ docker network create micro-ne  
89ed815259831bf5fb1b4d6b29f419f57343ccbc9ffe447874c00ea3cb86befc  
ubuntu@ip-172-31-24-48:~/microstore$
```

i-0a7ef532db89a6b20 (Vm-Server docker host)

PublicIPs: 34.228.165.170 PrivateIPs: 172.31.24.48

Langkah 4: Menjalankan Database (MySQL)

Service database harus jalan duluan karena service lain bergantung padanya. Kita tidak perlu build karena menggunakan image public (mysql:8.0), tapi kita perlu me-mount script SQL.

Perintah:

```
docker run -d \
--name mysqlDb \
--network micro-net \
-p 3306:3306 \
-e MYSQL_ROOT_PASSWORD=password \
-e MYSQL_DATABASE=microstore \
-v $(pwd)/database/init.sql:/docker-entrypoint-initdb.d/init.sql \
mysql:8.0
ubuntu@ip-172-31-24-48:~/microstore$ docker run -d \
--name mysqlDb \
--network micro-net \
-p 3306:3306 \
-e MYSQL_ROOT_PASSWORD=password \
-e MYSQL_DATABASE=microstore \
-v $(pwd)/database/init.sql:/docker-entrypoint-initdb.d/init.sql \
mysql:8.0
Unable to find image 'mysql:8.0' locally
8.0: Pulling from library/mysql
3758af62b8ea: Pull complete
ad9d782f3f87: Pull complete
8310b139f7c3: Pull complete
f34fb977c6ce: Pull complete
a2e746e7b2e3: Pull complete
750e74778c5d: Pull complete
75d963f8b5a9: Pull complete
acd41a114e50: Pull complete
b5f4bc928a89: Pull complete
082df7bb4f09: Pull complete
2583e052860a: Pull complete
a7ee027e6dce: Download complete
daf507697d1d: Download complete
Digest: sha256:271aa4b4c84411d52790b340039dc6cea8637ab273546ff15639a77fae6b29f6
Status: Downloaded newer image for mysql:8.0
8aa975932d68b84ff121a61af5101b82391a53b5dc58a2e0d6f677a680e806fb
docker: Error response from daemon: failed to set up container networking: network micro-net not found
Run 'docker run --help' for more information
```

i-0a7ef532db89a6b20 (Vm-Server docker host)

PublicIPs: 34.228.165.170 PrivateIPs: 172.31.24.48

Penjelasan:

- `--name mysqlDb`: Memberi nama container "mysqlDb" (nama ini akan dipakai service lain sebagai host).
- `--network micro-net`: Menyambungkan ke network yang kita buat tadi.
- `-v ...`: Memasukkan file init.sql agar tabel dibuat otomatis saat start.

Langkah 5: Build & Run Product Service

Service ini menggunakan Node.js dan harus di-build dari Dockerfile yang ada di folder product-service.

A. Build Image:

```
docker build -t image-product-service ./product-service
```

- B. Run Container: Perhatikan kita harus mengoper environment variable agar bisa connect ke database.

```
docker run -d \
--name product-service \
--network micro-net \
-p 3001:3000 \
-e DB_HOST=mysqldb \
-e DB_USER=root \
-e DB_PASS=password \
-e DB_NAME=microstore \
image-product-service
```

```
ubuntu@ip-172-31-24-48:~/microstore$ docker run -d \
  --name product-service \
  --network micro-net \
  -p 3001:3000 \
  -e DB_HOST=mysqldb \
  -e DB_USER=root \
  -e DB_PASS=password \
  -e DB_NAME=microstore \
  image-product-service
19da0d683a6d1d628ba4ad18426cb36180363a2220ad95ca6f8a6eafcc6aad13
```

i-0a7ef532db89a6b20 (Vm-Server docker host)

Public IPs: 34.228.165.170 Private IPs: 172.31.24.48

Catatan: Nama container --name product-service harus persis seperti itu, karena di nginx.conf tertulis server product-service:3000.

Langkah 6: Build & Run Order Service

Sama seperti product service, kita build dulu dari folder order-service.

- #### A. Build Image:

```
docker build -t image-order-service ./order-service
```

```

ubuntu@ip-172-31-24-48:~/microstore$ docker build -t image-order-service ./order-service
[*] Building 7.0s (10/10) FINISHED
[*]   [internal] load build context
[*]   [internal] transfer context: 27B
[*]   [internal] load metadata for docker.io/library/node:18-alpine
[*]   [internal] load .dockerignore
[*]   => transferring context: 2B
[*]   [1/5] FROM docker.io/library/node:18-alpine@sha256:8d6421d663b4c28fd3ebc498332f249011d118945588d0a35cb9bc4b8ca09d9e
[*]   => resolve docker.io/library/node:18-alpine@sha256:8d6421d663b4c28fd3ebc498332f249011d118945588d0a35cb9bc4b8ca09d9e
[*]   [internal] load build context
[*]   => transferring context: 3.38kB
[*]   => CACHED (2/5) WORKDIR /app
[*]   [3/5] COPY package*.json .
[*]   [4/5] RUN npm install --production
[*]   [5/5] COPY .
[*]   => exporting to image
[*]   => exporting layers
[*]   => exporting manifest sha256:bebfbaz90b79de30dc308ad02504db1507b2e22a1bac9e9a41a14130d9b785c7
[*]   => exporting config sha256:a28efb5d857421dd34ec3f9d1a732be08f90837720557fe1db2flac3d83015d2
[*]   => exporting attestation manifest sha256:a3b04caaa3b42aeaf385b59ce33b1dd6b30dc3839ced7f39ec90143d0ac04d
[*]   => exporting manifest list sha256:9b933ed6d1a7fb27372f37665bd7d5adc608dc9803ad26474ea64961ec7ff8fe
[*]   => naming to docker.io/library/image-order-service:latest
[*]   => unpacking and dockerizing docker.io/library/image-order-service:latest
ubuntu@ip-172-31-24-48:~/microstore$ 

```

i-0a7ef532db89a6b20 (Vm-Server docker host)

PublicIPs: 34.228.165.170 PrivateIPs: 172.31.24.48

B. Run Container:

```

docker run -d \
  --name order-service \
  --network micro-net \
  -p 3002:3000 \
  -e DB_HOST=mysql \
  -e DB_USER=root \
  -e DB_PASS=password \
  -e DB_NAME=microstore \
  image-order-service
ubuntu@ip-172-31-24-48:~/microstore$ docker run -d \
  --name order-service \
  --network micro-net \
  -p 3002:3000 \
  -e DB_HOST=mysql \
  -e DB_USER=root \
  -e DB_PASS=password \
  -e DB_NAME=microstore \
  image-order-service
a2e5dd7e8cdf03666a1458a56e9972457cf1b07fef19f9ebb09db6ce1032a602

```

i-0a7ef532db89a6b20 (Vm-Server docker host)

PublicIPs: 34.228.165.170 PrivateIPs: 172.31.24.48

Langkah 7: Menjalankan API Gateway (Nginx)

Terakhir, kita jalankan Nginx sebagai gerbang utama. Kita akan menggunakan image nginx:alpine dan menimpa konfigurasinya dengan file nginx.conf kita.

Perintah:

```

docker run -d \
  --name api-gateway \
  --network micro-net \
  -p 80:80 \
  -v $(pwd)/api-gateway/nginx.conf:/etc/nginx/nginx.conf:ro \
  nginx:alpine

```

```

ubuntu@ip-172-31-24-48:~/microstore$ docker run -d \
--name api-gateway \
--network micro-net \
-p 80:80 \
-v $(pwd)/api-gateway/nginx.conf:/etc/nginx/nginx.conf:ro \
nginx:alpine
Unable to find image 'nginx:alpine' locally
alpine: Pulling from library/nginx
de54cb821236: Pull complete
1074353eec0d: Pull complete
25f453064fd3: Pull complete
567f84da6fdb: Pull complete
da7c973d8b92: Pull complete
33f95a0f3229: Pull complete
085c5e5aaa8e: Pull complete
0abf9e567266: Pull complete
54c5bfcc22277: Download complete
35e741720152: Download complete
Digest: sha256:8491795299c8e739b7fcc6285d531d9812ce2666e07bd3dd8db00020ad132295
Status: Downloaded newer image for nginx:alpine
5e8602ade6b64e7e256b81373e738816a3b8b259eacf1f03f43db79b8849b255
docker: Error response from daemon: failed to set up container networking: network micro-net not found

Run 'docker run --help' for more information
ubuntu@ip-172-31-24-48:~/microstore$ 

```

i-0a7ef532db89a6b20 (Vm-Server docker host)

PublicIPs: 34.228.165.170 PrivateIPs: 172.31.24.48

Verifikasi (Testing)

- Cek Container Berjalan: Pastikan ada 4 container yang statusnya "Up".

```

docker ps
ubuntu@ip-172-31-24-48:~/microstore$ docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS                 NAMES
4b435e0f4ade        nginx:alpine        "docker-entrypoint..."   10 seconds ago    Up 9 seconds         0.0.0.0:80->80/tcp, :::80->80/tcp
d4af3e4afe7         image-order-service   "docker-entrypoint.s..." 22 seconds ago    Up 22 seconds       0.0.0.0:3002->3000/tcp, :::3002->3000/tcp
ce
adf1bcbffee4       image-product-service "docker-entrypoint.s..." 33 seconds ago    Up 32 seconds       0.0.0.0:3001->3000/tcp, :::3001->3000/tcp
vice
b194b13a7c8a        mysql:8.0           "docker-entrypoint.s..."  About a minute ago Up About a minute  0.0.0.0:3306->3306/tcp, :::3306->3306/tcp, 33060/tcp  mysqldb
ubuntu@ip-172-31-24-48:~/microstore$ 

```

i-0a7ef532db89a6b20 (Vm-Server docker host)
 PublicIPs: 3.90.88.52 PrivateIPs: 172.31.24.48

Bagian 3: Pengujian dari VM 2 (Client)

Sekarang kita berpindah peran menjadi client. Buka terminal baru.

Langkah 1: Persiapan Tools

Install tool sederhana untuk HTTP request jika belum ada:

```

sudo apt update && sudo apt install curl jq -y
ubuntu@ip-172-31-21-102:~$ sudo apt update && sudo apt install curl jq -y
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy InRelease
Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates InRelease [128 kB]
Get:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-backports InRelease [127 kB]
Get:4 http://security.ubuntu.com/ubuntu jammy-security InRelease [129 kB]
Get:5 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy/universe amd64 Packages [14.1 MB]
Get:6 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy/universe Translation-en [5652 kB]
Get:7 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy/universe amd64 c-n-f Metadata [286 kB]
Get:8 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy/multiverse amd64 Packages [217 kB]
Get:9 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy/multiverse Translation-en [112 kB]
Get:10 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy/multiverse amd64 c-n-f Metadata [8372 kB]
Get:11 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 Packages [3162 kB]
Get:12 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/main Translation-en [485 kB]
Get:13 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 c-n-f Metadata [19.1 kB]
Get:14 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/restricted amd64 Packages [5043 kB]
Get:15 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-security/main amd64 Packages [2901 kB]
Get:16 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/restricted Translation-en [944 kB]
Get:17 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/restricted amd64 c-n-f Metadata [644 B]
Get:18 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/universe amd64 Packages [1245 kB]
Get:19 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/universe Translation-en [310 kB]
Get:20 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/universe amd64 c-n-f Metadata [30.0 kB]
Get:21 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/multiverse amd64 Packages [57.6 kB]
Get:22 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/multiverse Translation-en [13.2 kB]
Get:23 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/multiverse amd64 c-n-f Metadata [600 B]
Get:24 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-backports/main amd64 Packages [69.4 kB]
Get:25 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-backports/main Translation-en [11.5 kB]
Get:26 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-backports/main amd64 c-n-f Metadata [412 B]
Get:27 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-backports/restricted amd64 c-n-f Metadata [116 B]
Get:28 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-backports/universe amd64 Packages [31.7 kB]

```

i-0b7db3beaafa2f5f (Vm-client tester)

PublicIPs: 54.226.135.221 PrivateIPs: 172.31.21.102

Langkah 2: Testing Endpoint

Lakukan request ke IP Public VM 1. API Gateway (Nginx) di VM 1 akan mem-forward request ke service yang sesuai berdasarkan path url.

Skenario 1: Mengakses Service Produk

Ganti <IP-PUBLIC-VM-1> dengan IP asli

```
curl http://3.90.88.52/products
```

```
ubuntu@ip-172-31-21-102:~$ curl http://3.90.88.52/products
{"success":true,"data":[{"id":1,"name":"Laptop Asus","price":12500000.00,"stock":10}, {"id":2,"name":"Mouse Logitech","price":350000.00,"stock":50}]}ubuntu@ip-172-31-21-102:~$ █
i-0b7db3beaaafa2f5f (Vm-client tester)
PublicIPs: 54.226.135.221 PrivateIPs: 172.31.21.102
```

Skenario 2: Mengakses Service Order

```
curl http://<IP-PUBLIC-VM-1>/orders
```

```
ubuntu@ip-172-31-21-102:~$ curl http://3.90.88.52/orders
{"success":true,"data":[{"id":1,"product_id":1,"quantity":2,"total_price":25000000.00,"product_name":"Laptop Asus"}, {"id":2,"product_id":2,"quantity":5,"total_price":175000.00,"product_name":"Mouse Logitech"}]}ubuntu@ip-172-31-21-102:~$ █
i-0b7db3beaaafa2f5f (Vm-client tester)
PublicIPs: 54.226.135.221 PrivateIPs: 172.31.21.102
```

KESIMPULAN

Praktikum implementasi microservices sederhana dengan Docker di AWS EC2 berhasil menunjukkan bahwa aplikasi dapat dibagi menjadi beberapa service yang berjalan secara terpisah dalam container. Penggunaan Docker mempermudah proses deployment, sedangkan AWS EC2 menyediakan lingkungan server yang fleksibel dan dapat diakses secara cloud. Praktikum ini membantu memahami konsep microservices, container, dan penerapannya di cloud secara nyata.