TheCork - High-Level Architecture - Group 06

Students

- Afonso Lopes (95528)
- Eduardo Claudino (95567)
- Tomás Pereira (95682)

Architecture

For the chosen scenario, we have decided that the system will be separated into 3 Virtual Machines:

- VM-DB: The Database machine, running **MySQL**
- VM-API: The API machine, running a **Java** application
- VM-Client: The client machine, also running **Java**

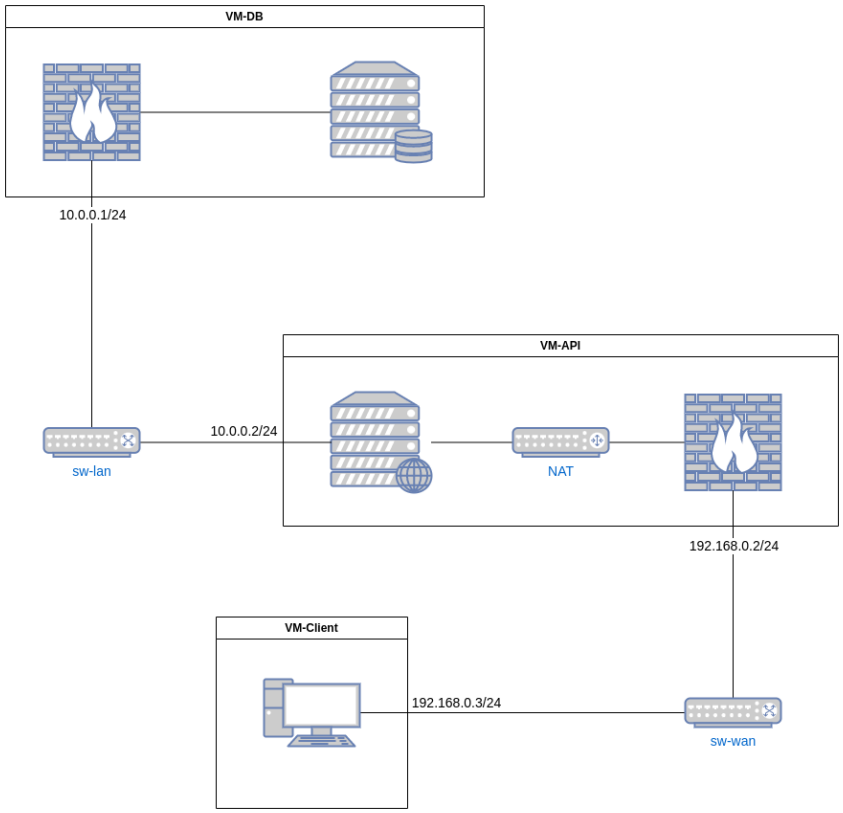
VM-DB is connected to VM-API through a switch we denominated `sw-lan`, to simulate a LAN connection. As a convention, we decided to connect VM-API to VM-Client through another switch we denominated `sw-wan`, to simplify connections from a client outside the *LAN*.

That said, we can then define the VM network **interfaces** as such:

- ***VM-DB:***
  - `enp0s3`: Internal Network connected to `sw-lan`
- ***VM-API:***
  - `enp0s3`: Internal Network connected to `sw-lan`
  - `enp0s8`: Internal Network connected to `sw-wan`
  - `enp0s9`: NAT (host network)
- ***VM-Client:***
  - `enp0s8`: Internal Network connected to `sw-wan`
  - `enp0s9`: NAT (host network)

For convenience sake for the simulated environment, we're allowing VM-API and VM-Client to connect to the internet independently through a NAT interface. VM-DB can have net access by routing requests through VM-API. These details will not be taken into consideration for the solution's diagram.

Finally, we can now define the **network diagram** as such:



Firewalls

As seen in the diagram, this system has two **firewalls**:

- One for the **DB machine**, to only allow valid incoming requests from the API through. (In MySQL, by default, the database receives **TCP** requests on **port 3306**)
- One for the **API**, to only allow valid HTTP or HTTPS requests through.

These firewalls are configured using *Uncomplicated Firewall* , **UFW** for short. The configuration commands used in this setup are the following (assuming default starting conditions):

For VM-DB (Database):

```
sudo ufw enable
sudo ufw default deny INCOMING
sudo ufw default deny ROUTED
sudo ufw allow from 10.0.0.2 to any port 3306
sudo ufw reload
```

For VM-API (API):

```
sudo ufw enable
sudo ufw default deny INCOMING
sudo ufw default allow ROUTED        #so VM-DB has net access
sudo ufw allow http
sudo ufw allow https
sudo ufw reload
```

As mentioned before, for convenience sake, we're allowing VM-DB to connect to the internet through VM-API. To permit this behaviour in a **NAT** context with **UFW**, the following block of code was added to `/etc/ufw/before.rules`:

```
# NAT table rules
*nat
:POSTROUTING ACCEPT [0:0]

# Forward traffic through enp0s9 - Change to match your out-interface
-A POSTROUTING -s 10.0.0.1 -o enp0s9 -j MASQUERADE

# don't delete the 'COMMIT' line or these nat table rules won't
# be processed
COMMIT
```

Tecnologies

Both the **API** and the **Client App** will be developed in **Java**, as to share the same cryptography libraries and to facilitate communications.

In terms of communications, the API will be implemented in **REST** for client-server communications, supported by **Spring Boot** framework. For the API to communicate with the DB, the system will be using the **JDBC** API.

For the remainder tecnologies, the firewall policies will be configured using **UFW** and the Database will be running on a **MySQL** engine.