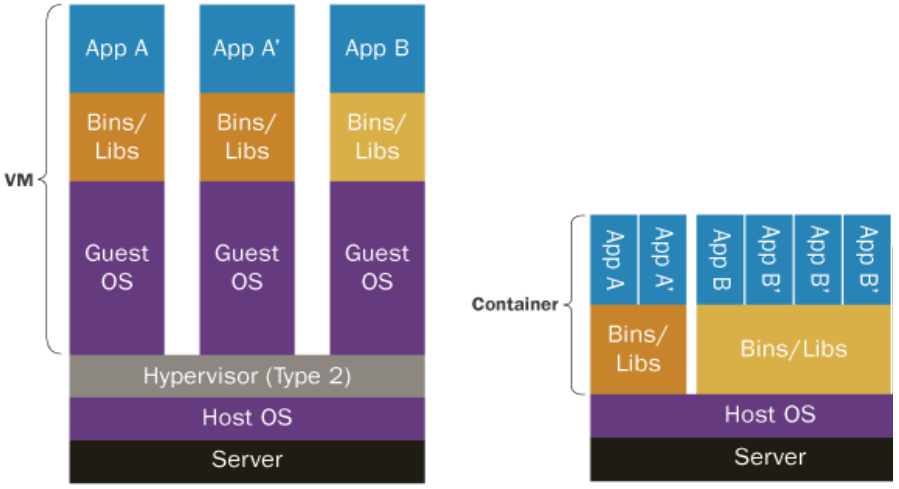
**CLOUD COMPUTING:**

**Groupe:** B1
Abderrazzak Mahraye, Ilhame Hadouch

**I- Theoretical work**

## Objectives 1, 2, and 3:

### 1. Similarities and differences between the main virtualisation hosts (VM et CT)




**Elaboration:**
In containers the hardware is not virtualized, it can be set on the host OS. So it does not need guest hosts and hypervisors to handle the ressources. Consequently, several applications can share the same libraries.

**Comparison VM vs Container**:

- virtualization cost, taking into consideration memory size and CPU:

Containers are more efficient about cost/CPU/memory insofar as it does not have to launch several OSs to use several applications.

| | VM | CT |
|---|---|---|
| Virtualisation cost | Allocation has to be done (VM1 2CPUs… VM2 2CPUs) If both are running they can not use the resources that are not allocated<br><br>developer:<br>administrator: | Allocation is not necessary. Each application can use the resources directly.<br><br>developer: CT is better because, the developer does not have to allocate the resources<br>administrator:CT is better because, the administrator does not have to allocate the resources |
| CPU/Memory | For a given application, each app is on a different OS which can be configured separately with different cpu resources, memory size or network parameters. So the application has access to all this resource<br><br>developer:<br>administrator: VM is better for the admin because he can handle the ressources as he wants | For a given application, every app will theoretically have the same access to CPU, memory and network resources. So they share the ressource.<br><br>developer: containers are better because the process manage themselves<br>administrator: |
| security | VM is more secure because all application are isolated.<br><br>developer: better to avoid security issues<br>administrator: better to avoid security issues | Containers are process that are launched on the same OS, consequently they may have access to sensitive data.<br><br>developer:<br>administrator: |
| performance | VM is lower in terms of time responses because there are already resources that must be allocated to the main OS to launch other OSs. So, these resources cannot be used by the generated OSs.<br><br>developer:<br>administrator: he doesn't have to develop applications with response time constraints, performances are the same for a given hardware | Containers can access directly to ressources without hypervisors. It is faster in terms of time. They can be launched faster.<br><br>developer: better because if the developer want to develop several application, he does not have to switch from an os to an other<br>administrator: |
| continuous integration | If several OSs are needed VMs are better to access to | The containers are launched on a single OS. If the user |

| | | |
|---|---|---|
| | services provided by the OSs<br><br>developer:<br>administrator: better because he needs differents services provided by different VMs | needs services of an OS which is not the main OS, he has to install it and set up all the processes again.<br><br>developer: better: some containers contains already all the tools needed to develop application administrator. |

### 2. Similarities and differences between the existing CT types

Different CT technologies are available in the market (e.g. LXC/LXD, Docker, Rocket, OpenVZ, runC, containerd, systemd-nspawn). Their respective positioning is not obvious, but comparative analyses are available online, such as the one displayed in Figure 2.

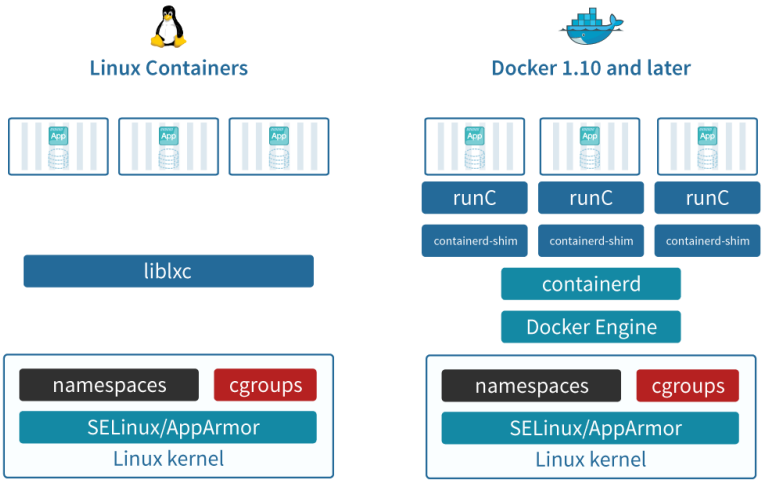



Figure 2 : Linux Lxc vs Docker

These technologies can however be compared based on the following criteria (non-exhaustive list):

- Application isolation and resources, from a multi-tenancy point of view,
- Containerization level (e.g. operating system, application),
- Tooling (e.g. API, continuous integration, or service composition).

**Tasks (You need to write down your answers in the shared document):**

- Elaborate on the proposed criteria,
- By doing your own research on the Web, classify the existing CT technologies (LXC, Docker, Rocket, …) based on these criteria, and eventually, on additional criteria that you need to identify by yourself.

| | LXC | Docker | Rocket |
|---|---|---|---|
| Application isolation and resources | LXC is to provide isolated application environments that closely resemble those of full-blown virtual machines (VMs) but without the overhead of running their own kernel. | It uses the capabilities provided by liblxc. | |
| Containerization level | Only works on ubuntu | | |
| Tooling | run more than just the one process in an LXC container, | Docker is designed for running a single process in each container.<br>Ex: Dockerfile, Docker Hub, Docker Registry, Docker Compose... | Rocket does not have a daemon<br><br>flexibility in publishing or sharing images<br><br>moving to another container system |
| Security | Due to the fact that LXC has a good isolation as we said before and that it is closed to VMs, we can say that it is secured. | Root privileges: containers are created from root privileges → privileges escalation attacks.<br><br>Authenticity: before version 1.12 , Authenticity verification is not done so we were not safe from issues. | Root privileges: containers are not created from root privileges → privileges escalation attacks safety. |

https://bobcares.com/blog/docker-vs-rkt-rocket/2/

### 3. Similarities and differences between Type 1 & Type 2 of hypervisors' architectures

**Type 1 & Type 2 of hypervisors' architectures:**

Hypervisor type 1 runs directly on the host whereas the type 2 one runs on the OS.
Virtual Box: type 2
OpenStack: type 1

**II - Practical part**

## Objectives 4 and 5:

### First part: Creating and configuring a VM

We created and ran VM on virtual box.

### Second part: Testing the VM connectivity

IP adresses:
The addresses are not the same between the VM and the Host machine.

We can ping from the VM to the host and to the neighbour's host. However our host can not ping our VM. That is normal because the VM goes through the NAT to reach our host and the neighbour host. But our host does not go through the NAT to reach the VM.

### Third part: Set up the "missing" connectivity

We set up a port forwarding to fix the missing connectivity. Consequently, we can now do an ssh connection with our VM by ssh our host. The port forwarding is done on port 22 and the access to the VM is done. However we still can not ping the VM because we do not know the port of the ping.

### Fifth part: Docker containers provisioning

Testing the pings:

We can ping over the internet from the docker.
We can ping our VM.
We can ping from the VM to the docker.
We can ping from the docker to the host.

### Fifth part: Docker containers provisioning

After creating a new container ct3 from the snapshot of ct2 we still have nano. It is evident because the generated snapshot still contains the installed nano.

We created a persistent image that contains the installed nano. So if we create a new container from this image, we will have nano installed on it.

## Objectives 6 and 7:

### First part : CT creation and configuration on OpenStack

We created a new VM using Ubuntu by launching a new instance on OpenStack.

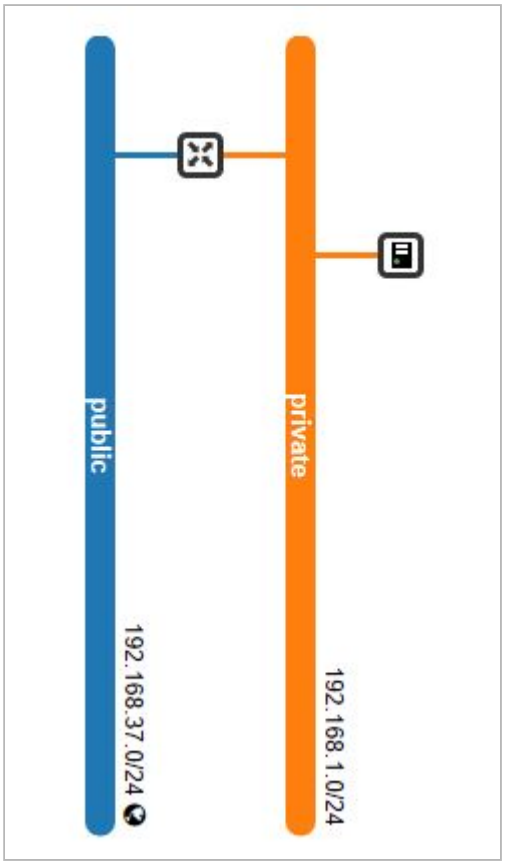
**Configuring the security rules:**
By default, OpenStack blocks the network traffic on the virtualized private network. Consequently we added a new security group rule to allow SSH and ICMP connection.

**Connectivity test:**
The IP address of the instance is: 192.168.1.229

We can not ping our host from the VM and the host can not ping the VM. This is normal because our VM is not visible in the network our the host, the VM is not the public network of INSA.

We needed to create a router in order to connect the private network to the insa public one. To do that, we created a new interface on the router, and then connected it to our private network.

In order to allow the VM to be visible from outside, this one must have a floating IP. That's what we have done, and then we are allowed to SSH our VM from our host.




**Third part: Snapshot, restore and resize a VM**

We resized the VM without turning off the machine. That worked but we have been disconnected from the console.
We resized the VM after the extinction and that worked to.

That shows the flexibility of openstack ( the resize with the machine turned off or on) in opposition to the VirtualBox (we must stop the VM)..

However there is a limitation: we can not resize the VM.

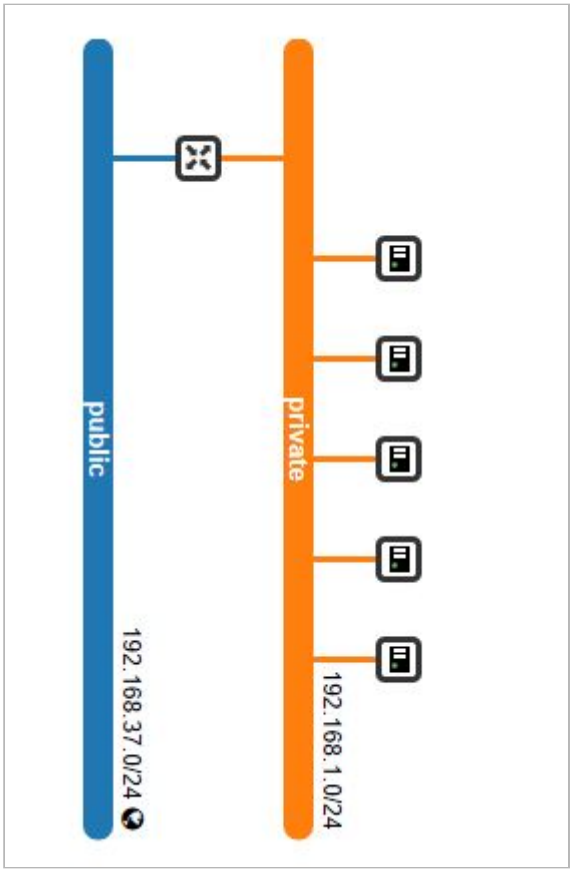
## Objectives 8 and 9:

**Part three: Deploy the Calculator application on OpenStack**

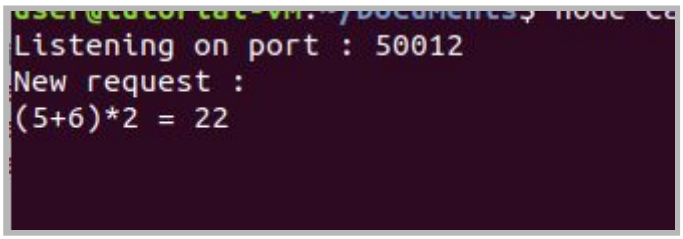
We created 5 VMs: SumService, SubService, MulService, DivService and CalculatorService (CS).

One of them is reachable from our host (Floating IP), it represents the CS. It is also in the network of the other VMs. So, we can reach each service by passing through the CS which can reach the VMs.

Each service is listening on a port which is in a range between 50000 and 50050. We added this rule in the openstack securityGroup rules to open the ports.

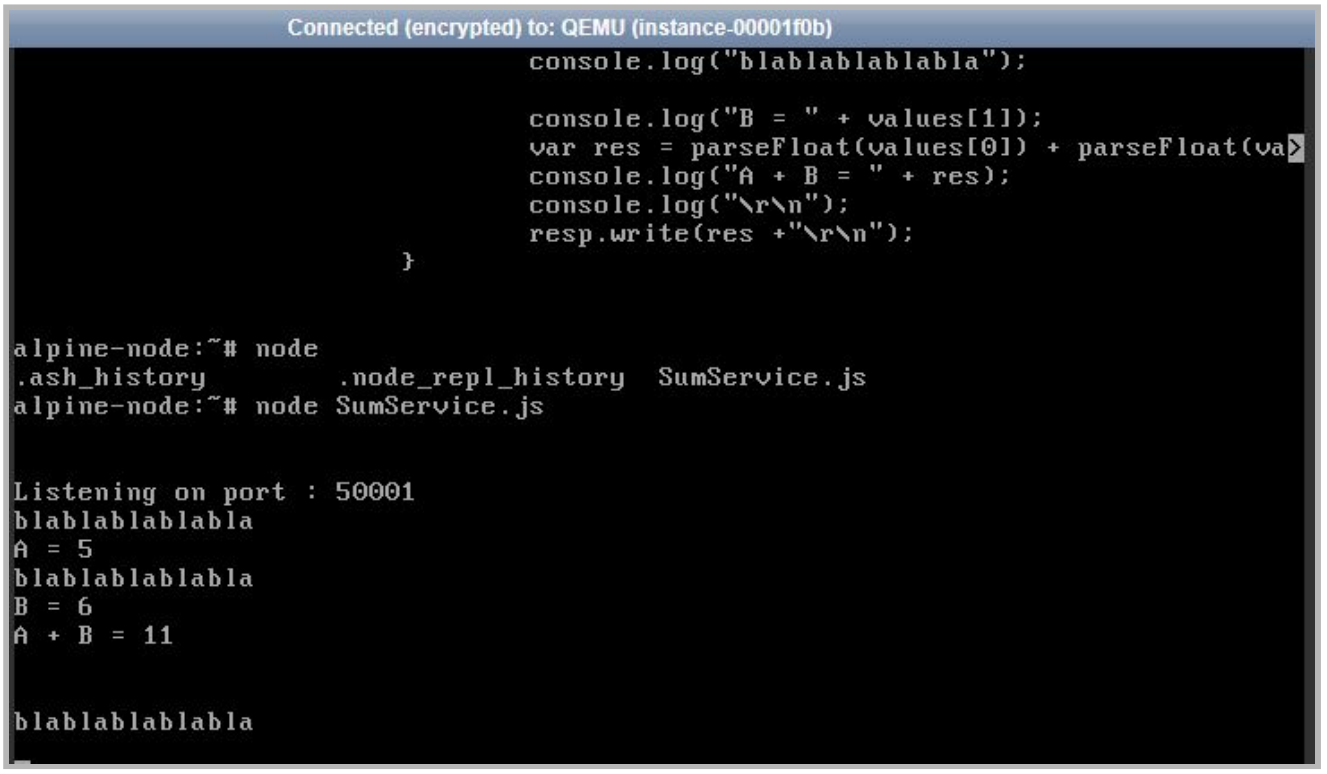



So we tested our services, by doing a Curl to our CS.



```
^C
user@tutorial-vm:~/Documents$ curl -d "(5+6)*2" -X POST http://192.168.37.198:50012
result = 22
```

By adding text visualization on a service, the added text was displayed as expected.




IP adresses:

Mul: 172.17.0.2
Sub : 172.17.0.4
CS: 172.17.0.3
Div : 172.17.0.5
Sum: 172.17.0.6

**Part four: Automate/Orchestrate the application deployment:**

We did the same work with containers. We created 5 containers with an image built with a dockerFile.

METTRE SCREEN DOCKERFILE.

After setting up all things we needed, we can run the five dockers. Then, when we did a curl command, it successfully worked as we can see on the next screenshot.

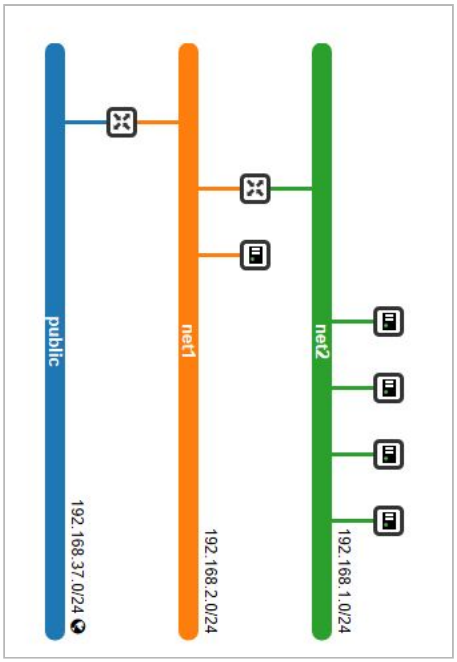
```
usr
var
user@tutorial-vm:~/Documents$ sudo docker exec ctSub node SubService.js
Listening on port : 50002
^C
user@tutorial-vm:~/Documents$ sudo docker exec -d ctSub node SubService.js
user@tutorial-vm:~/Documents$ sudo docker exec -d ctMul node MulService.js
user@tutorial-vm:~/Documents$ sudo docker exec -d ctDiv node DivService.js
user@tutorial-vm:~/Documents$ sudo docker exec -d ctSum node SumService.js
user@tutorial-vm:~/Documents$ curl -d "(5+6)*2" -X POST http://<ip>:50012
bash: ip: Aucun fichier ou dossier de ce type
user@tutorial-vm:~/Documents$ curl -d "(5+6)*2" -X POST http://172.17.0.3:50012
result = 22

user@tutorial-vm:~/Documents$
```

## Objectives 10 and 11:

In this step, as NaaS (Network as-a-Service) provider, we propose to process a prospective client request and deliver on-demand virtualized network topologies that include virtual hosts, routers, and communication functions implementing network levels from 3 to 7.

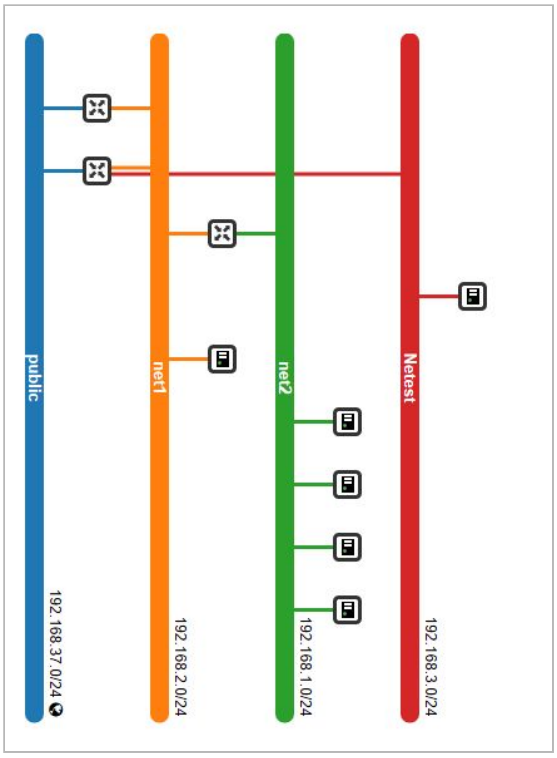
### Part one: The client requirements and target network topology

We created the topology required: the two subnet networks of alpines VMs and the routers.




Weirdly, without adding a traffic route, we were able to ping from the CS to the VMs and reversely.

However, we added a  third subnet to test, and, cursiously, the ping did not work. That is normal because the CS do not know the path to access to the other VMs. So, we must indicate the next router where the VMs are to the CS. To do that we added a new route indicating the network wanted to reach and the gateway (the router) that allows access to this network.




```
user@tutorial-vm:~/Documents$ gedit CalculatorService.js
user@tutorial-vm:~/Documents$ ping 192.168.3.59
PING 192.168.3.59 (192.168.3.59) 56(84) bytes of data.
From 192.168.2.1 icmp_seq=1 Destination Net Unreachable
From 192.168.2.1 icmp_seq=2 Destination Net Unreachable
From 192.168.2.1 icmp_seq=3 Destination Net Unreachable
From 192.168.2.1 icmp_seq=4 Destination Net Unreachable
^C
--- 192.168.3.59 ping statistics ---
8 packets transmitted, 0 received, +4 errors, 100% packet loss, time 7133ms

user@tutorial-vm:~/Documents$ sudo route add -net 192.168.3.0/24 gw 192.168.2.38
/24
[sudo] Mot de passe de user :
192.168.2.38/24: Hôte inconnu
user@tutorial-vm:~/Documents$ sudo route add -net 192.168.3.0/24 gw 192.168.2.38
user@tutorial-vm:~/Documents$ ping 192.168.3.59
PING 192.168.3.59 (192.168.3.59) 56(84) bytes of data.
^C
--- 192.168.3.59 ping statistics ---
18 packets transmitted, 0 received, 100% packet loss, time 17389ms
```

We tested a curl command and it worked.

```
U:\>curl -d "(5+6)*2" -X POST http://192.168.37.198:50012
result = 22
```