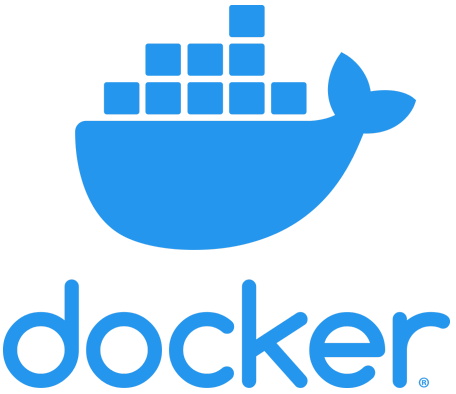


TP Adaptabilité : Cloud et gestion autonomique

Groupe B1-5

Kilian Desportes (login : desporte)
Yanis Imekraz (login : imekraz)

Theorical Part

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

1) **Understand the figure above, and elaborate on it.**

The architecture between VM and CT is different. When using a VM, there's a hypervisor which will make a link between our OS and the VM OS. It allows deployment of an application on a specific OS. On CT, there's only one OS, the host one, which will be used to host every application at the same time.

Compare the two types of hosts based on two perspectives: from an application developer's point of view, and from an infrastructure administrator point of view.

2) **For each one of these perspectives, the comparison should be based on the following criteria :**

   a) **Virtualization cost, taking into consideration memory size and CPU,**

VM are heavier to deploy due to the multiple OS architecture. OS will take a larger part of the memory and CPU will be more solicitated due to the multiple OS utilisation (host+guest one). The virtualization cost is greater on VM due to the 'reservation' of ressources by this one, resources are 'wasted' if the VM isnt using them are their maximum. On container architecture, containers will use appropriate ressources for their usage, and virtualization cost is smaller.

From an application developer point of view, architecture won't change anything, he just has to know how many ressources are allowed to its application. From an infrastructure administrator point of view, VM are heavier to deploy due to hypervisor and multiple OS configuration. It has to know how much ressources should be allowed to every VM to allow applications to run properly and to minimize the waste of ressources.

   b) **Usage of CPU, memory and network for a given application,**

For a given application, on CT, every app will theoricaly have the same access to CPU, memory and network ressources. On VM, each app is on a different OS which can be configured separately with different cpu ressource, memory size or network parameters. This will make every app have potential access to different ressources. If the guest OS ressources allow it, a given application will use the same ressources as if it was running on a container.

From an application developer point of view, architecture won't change anything, he just has to know how many ressources are allowed to its system and how much ressources this application can take. From an infrastructure administrator point of view, the network is easier to configure on VM because every OS are separated and they allocated their own ressources.

c) Security for the application (access right, resources sharing, etc.),

Using a VM, each OS being separated, the security of application is greater. Each application has its own file system and it's own ressources. Each application is isolated from the others via separated guest OS.

With containers, it's the same OS that is running every application. The security is not induced by the architecture. There can be potential conflict between applications concerning resources sharing on the OS.

From an application developer point of view, using containers, he has to be carefull about file system rights and root running mode for applications because this isnt safe.
From an infrastructure administrator point of view, VM are much safer due to the separated architecture. On containers, he has to secure the file system and users privilige because the OS and the file system are common to every application.

d) Performances (response time),

VM will have higher response time compared to containers because the VM is simulating hardware with their given ressources and this will decrease performance.
On containers, applications will run at their maximum potential.

From an application developer point of view, VM have given ressources and this can cause performance issues if ressources are smaller than application needs. VM will have more response time due to the double OS between hardware and applications. On containers, applications should run with smaller response time.

From an infrastructure administrator point of view, he doesn't have to develop applications with response time constraints, performances are the same for a given hardware. He have to correctly balance them with the needs of applications running on VM.

e) Tooling for the continuous integration support

Continuous integration support is much easier on containers because they are running on the same architecture. So changing the code of applications, reusing it, or migrating it will be easier.

On VM, each OS being separated, applications will harder be reused or migrated because the architecture (OS) has an important impact on how the application is developped, especially its relation with hardware, plugins, OS management.

From an application developer point of view, developing on containers is much easier due to the same environment for every application on it. It's easier to set up a continuous integration environment on a container. VM are isolated from each other and therefore it's harder to develop multiple applications and maintain them.

From an infrastructure administrator point of view, as he doesn't develop applications he doesn't have to use continuous integration tools.

2. Similarities and differences between the existing CT types

1) **Elaborate on the proposed criteria,**

   a) **Application isolation and resources, from a multi-tenancy point of view,**

On the server, each container will be isolated from each other even if the underlying structure is the same for every container. They'll have isolated resources allowed by the process that manages containers.

   b) **Containerization level (e.g. operating system, application),**

Each container type manages its containers and the application on it differently. A container can contain OS specific files to allow application running. And it's possible to run different Linux distros based containers on a single host for example, if this one which shares the same kernet space.

   c) **Tooling (e.g. API, continuous integration, or service composition).**

Containers offer tools to handle continuous integration for application and can also integrate multiple API or services which will be available for applications.

2) **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.**

Container technology	App isolation and resources	Containerization level	Tooling	Setup
Docker	Same kernel. Same hardware as the host is a thin layer of separation. Easy isolation. Don't need to manage Ram and space.	Application virtualization. Applications running on the same architecture. Hardware not duplicated virtualy. Run on a single node.	Docker API for docker images management. Downloadable OS images. Intercommunication API. Jenkins for CI (installed separately). Package and distrubiton of containerized applications. Require cloud storage for sizable systems.	Easy to set up (as a standard application) and manage.

LXC	Same kernel. Same hardware. Isolation with namespace and contol groups. Each container having it's own system setup.	Full system virtualization.	Don't require cloud storage (Linux provides the feature) Same tooling as bare-metal server (SSH access, script..etc)	Close to classic Linux VM deployment. Simulation of complete Linux OS.
Kubernetes	Same kernel. Same hardware as the host is a thin layer of separation. Time-consumin g isolation.	Container orchestrator. Runs on a cluster.	Handle docker images. Features to manage workload. Jenkins for CI(installed separately). Allow scale run and monitor applications.	Manual configuration to tie components. Need to know network configuration during setup.

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

1) **Based on the lecturer's slides (slides 14, 15 and 16), summarize and elaborate on each of these types.**

On type 1, there is no main OS between VM and hardware, the hypervisor is directly ahead of the hardware and makes the link with VMs.

On type 2, there is a main OS between hardware and hypervisor. This allows it to run directly on the main OS without having to use a VM.

2) **Identify to which architecture type VirtualBox and OpenStack belong to.**

VirtualBox, running on our main OS, is a type 2 architecture. We can run a specific OS in VirtualBox process.

OpenStack is also a type 2 architecture and needs an OS to execute commands related to scripts transmited by OpenStack Instance.
(source : https://www.redhat.com/fr/topics/openstack)

Practical Part

VirtualBox :

Identify the IP address attributed to the VM using ifconfig (in Linux command line), and compare it to the host address (ip config in the command line). What do you observe?

VM IP : 10.0.2.15/24
Host IP: 10.1.5.86/16

The IPs are not on the same network.

Check the connectivity from the VM to the outside. What do you observe?

We can ping the host from the VM. We can also ping distant server (google.fr)

Check the connectivity from your neighbour's host to your hosted VM. What do you observe?

We can ping our neighbour's machine from our VM but the reverse way doesn't work.

Check the connectivity from your host to the hosted VM. What do you observe?

We can't ping our VM from host.

Docker :

What is the Docker IP address → *Docker ip adr = 172.17.0.2*

Ping an Internet resource from Docker→ *Ping google.fr worked*

Ping the VM from Docker → *Ping 10.0.2.15 worked*

Ping the Docker from the VM → *Ping 172.17.0.2 worked*

Do you still have nano installed on CT3? Explain why.

We still have nano installed because it was installed when we committed the ct2 image. We load the ct3 from this ct2 image, which is containing nano application, so nano application is already installed.

OpenStack :

After instance creation, we observe the IP Address directly on the dashboard.

We can also create a snapshot directly and we have access to other information like power state, age, status.

OpenStack connectivity test :

What do you think about this address?

It's a private address, on network 192.168.1.0/24, address is 192.168.1.66.

Check the connectivity from the VM to the desktop.

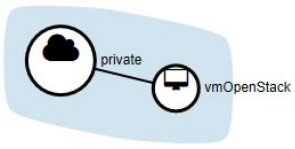
Pinging from the VM to the Desktop is not working (destination host unreachable).

Check the connectivity from the desktop to the VM. Write down your comments.

Pinging from desktop to VM is working.

Graphical view of topology :

Private network where the VM is located is not connected to the external public, that's why the VM cannot ping the host which is through this network.



After creating a router to connect private and public network, the VM can ping the desktop properly :

```
user@tutorial-vm:~$ ping 10.1.5.154
PING 10.1.5.154 (10.1.5.154) 56(84) bytes of data.
64 bytes from 10.1.5.154: icmp_seq=1 ttl=126 time=4.81 ms
64 bytes from 10.1.5.154: icmp_seq=2 ttl=126 time=1.30 ms
64 bytes from 10.1.5.154: icmp_seq=3 ttl=126 time=0.953 ms
```

Snapshot, restore and resize a VM

When we resize an instance, her status is 'Resize/migrate' and we then need to confirm the resize/migrate by a click to allow her to be active again.

When the VM is shutdown, status is 'resize/migrate' then back to active without having to confirm.

Technical limitation is the fact a VM needs a given amount of ressources to work, which cannot be necesseraly given with low ram/memory flavor (so we cannot downsize the VM as we want to).

If the VM is shutdown, needed ressources are hiden and then downsizing the ressources can be risky because it could crash when restarting the VM.

Normally there is no difference between the current image of the VM and the snapshot we just created.

After creating the snapshot, we restarted the VM and it started correctly.

OpenStack client installation

We created 5 vm, 4 to heberge microservices operations (vm_microservices), one to heberge calculator micro service (vm_1) and we use vmOpenStack to request the calculator.

☐	Instance Name	Image Name	IP Address
☐	vm_microservices-4	vm_uservices	192.168.1.131
☐	vm_microservices-3	vm_uservices	192.168.1.77
☐	vm_microservices-2	vm_uservices	192.168.1.140
☐	vm_microservices-1	vm_uservices	192.168.1.181
☐	vm_1	ubuntu4CLV	192.168.1.126, 192.168.37.131
☐	test	ubuntu4CLV	192.168.1.29
☐	vmOpenStack	ubuntu4CLV	192.168.1.66

We executed the command

```
curl -d "(5+6)" -X POST http://192.168.1.126:50015
```

and got

```
result = 11
```

which is correct.

If we modify the application sources it won't stop running but the changes won't be applied.

If we restart the services, changes are applied.

Docker :

Script de création des containers pour la mise en place du service Calculator :

```
sudo docker build -t userv:v2 -f dockerServices.dockerfile .
sudo docker run -d --name uservSum -it userv:v2
sudo docker run -d --name uservSub -it userv:v2
sudo docker run -d --name uservDiv -it userv:v2
sudo docker run -d --name uservMul -it userv:v2
sudo docker run -d --name uservCal -it userv:v2
sudo docker ps (verifiy everything is ok)
sudo docker exec -d uservSum node SumService.js
sudo docker exec -d uservSub node SubService.js
sudo docker exec -d uservDiv node DivService.js
sudo docker exec -d uservMul node MulService.js
sudo docker exec uservCal node CalculatorService.js
```

Test final :

```
user@tutorial-vm:~$ curl -d "(5+6)*2" -X POST http://172.17.0.6:50015
result = 22
```

NaaS :

Test the connectivity between the VM hosting the calculator front end and any machine from the ones hosting the arithmetic operations services.
What do you observe?

We can't ping an arithmetic machine from the calculator due to the lack of route to our router.

Elaborate on this?

We need to add a route to guide the packet through the needed network.

We won't need to add a route for the network hosting arithmetic operations services as it is only connected to one router, which will be the default gateway.

After setting up the route on the calculator VM, we are able to ping (and use cURL) in both directions from VM hosting calculator to machine hosting arithmic operations and vice versa.