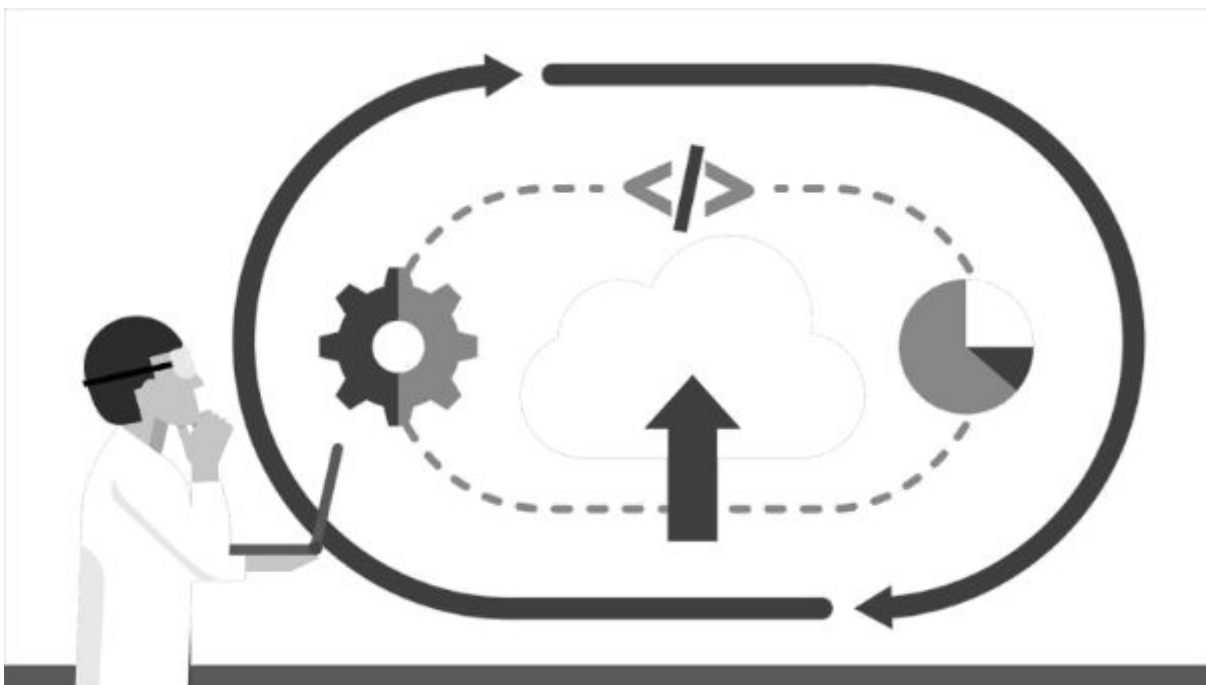


CLOUD LABS

5ISS



Vincent Laurens

Samir Fayçal Traoré IR - IoT
Valley

Sami Yanguï

yanguï@laas.fr

Sami.Yanguï@insa-toulouse.fr

Khalil Drira

<https://homepages.laas.fr/khalil/page/index.php>

Plan

Introduction	3
Theoretical part	3
Virtualization	3
Hypervisor	4
Virtualization Techniques	5
Virtual Machine vs Containerization	5
Containerization technologies : LCX vs Docker	7
Network connection modes for virtualization hosts : Bridge vs NAT	9
Practical part	10
VirtualBox VM and Docker containerization	10
[VM] Creating and configuring a VM	10
[VM] Testing the VM connectivity	11
[VM] Set up the “missing” connectivity	12
[VM] VM duplication	13
[Container] Docker containers provisioning	14
Application of Virtualization technologies in Cloud infrastructure	23
CT creation and configuration on OpenStack	23
Creation of Network infrastructure	25
Connectivity Tests	26
Configuring security groups	30
Snapshot, restore and resize a VM	33
Resizing a VM	33
Creation of a snapshot/Restore	35
Web 2-tier application topology and specification	36
OpenStack client installation	36
Test of hosting infrastructure and if application is accessible from public Network	38
NaaS (Network as-a-Service) provider,	39
Conclusion	43
Bibliography/Webography:	43

Introduction

This document is a formal debriefing of our labs experimentation during Cloud Labs. We will first introduce the virtualisation concept, then we will compare different concepts and technologies of virtualization techniques. Next we will apply our theoretical knowledge to implement, manage and deploy dynamic and distributed virtualized software systems on a cloud infrastructure.

Theoretical part

In this section, we will recognize the fundamental differences between the types of hypervisors' architectures (type 1/type 2).

First to start we will define what is virtualization and what different kind of virtualization we find, then we will define what is a hypervisor and identify its role, finally we will compare to techniques of virtualization (VM and containers).

Virtualization

Virtualization [1] is the process of creating a software (or virtual) version of a physical entity, such as applications, servers, storage systems and virtual networks. It is the most effective way to reduce IT expenses while boosting efficiency and agility for all businesses, regardless of size. There exist several classes of virtualization :

- storage virtualization
- hardware virtualization
- desktop virtualization
- network virtualization
- application virtualization

Hypervisor

A hypervisor, also known as a virtual machine monitor, is a process that creates and runs virtual machines (VMs). A hypervisor allows one host computer to support multiple guest VMs by virtually sharing its resources, like memory and processing.

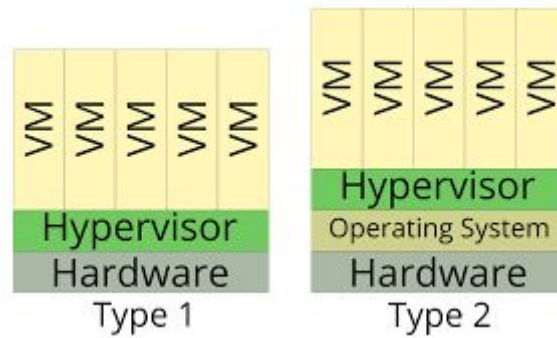


figure 1 : Description of hypervisor type architecture

There are two types of hypervisors as describe in figure 1. Type 1 hypervisors, called “bare metal,” run directly on the host’s hardware. Type 2 hypervisors, called “hosted,” run as a software layer on an operating system, like other computer programs. Now we will try to compare them and give an example of each.

The main difference between both is that in type 1 the hypervisor can run directly on hard machine rather than type 2 hypervisor need a os.

Type 1 is the most used because we generally run Type 1 is ran directly on the machine. So that is the most used because we generally run directly a virtual machine on a server machine. So it is widely used by cloud providers which can host differents hosts on same physical machine.

VirtualBox is a type 2 hypervisor, i.e. it must be installed on an operating system and not directly on a computer as an operating system.

OpenStack is a type 1 hypervisor because he can run on a specific OS.

Virtualization Techniques

Virtual Machine vs Containerization

We will now compare the two main virtualization techniques which are virtual machine and containers.

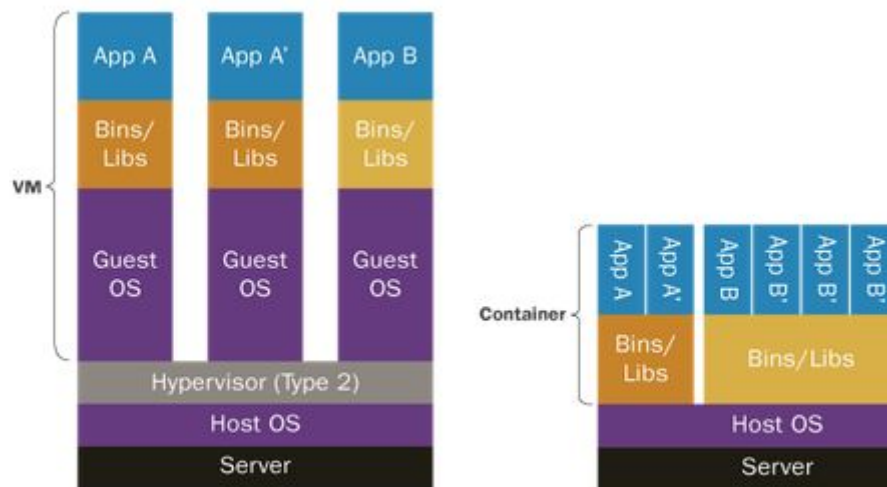


Figure 2: Virtual Machine vs Container

In the figure 2 scheme we can observe the case of Virtual Machine. In this case we can see that Hypervisor create a hypervisor of type 2 because VM is an OS (guest OS) which runs inside another OS which can be the native OS of the machine or another VM. Then as shown on the picture the hypervisor will allow to host one or more guest OS and will share resources of machines between different guest OS.

The second scheme describe the architecture based on virtual containers. A virtual container is a data structure which aims to organize others virtual objects.

We have to know that containers can be run on Virtual machine. It is used for example when we want to increase isolation between two services in a application. An example is application which need to be both scalable and secure.

	Virtual Machine	Containers
Virtualization cost	Is RAM and CPU consuming. We know because if we run multiple virtual machine in Virtualbox our PC can lag.	Less costly because lighter and based on existing OS.
memory size	More consuming because of hypervisor	Less consuming.
CPU	Need more CPU	Need not a lot of CPU
CPU Usage	More consuming	Less consuming
Security for the application	The hypervisor represent a secure layer for VM. So that why hypervisor has to be protected against for exemple DOS or DDOS. Because if a pirate access hypervisor he as access to all guests OS.	Software so the isolation between different layers is not right delimited. So it is less secure than VM
Performance	More longer to install but more stable	Scalable up which allow to add or remove instance of container in function of demand and thus the charge of the system.
Flexibility if failure	Rebuild of system after failure is longer	more reactive to failure because of his size. And it is easy to make a copy of container and deploy it.
Tooling for continuous integration support	In my opinion use a jenkins job or pipeline to deploy a virtual machine will ask to provision an image VM which can be heavy.	For container the docker image can be found on docker hub and is very light. Otherwise we have to develop a dockerfile to implement services on docker container.
Portable	Not so much	easily just a script which call a configuration script code we run.

As bilan of this comparison, for us, none of both technologies are better than the other. It is depending on what type of application we will host and deploy. For example, if we develop an application sharing sensitive data we will prefer virtual machines, if we develop an application based on IoT we will may be prefer containers because we want it to be scalable to the demand.

Containerization technologies : LCX vs Docker

In this part we will describe to containerization technologies that are at the time concurrent but also LCX is part of Docker technology.

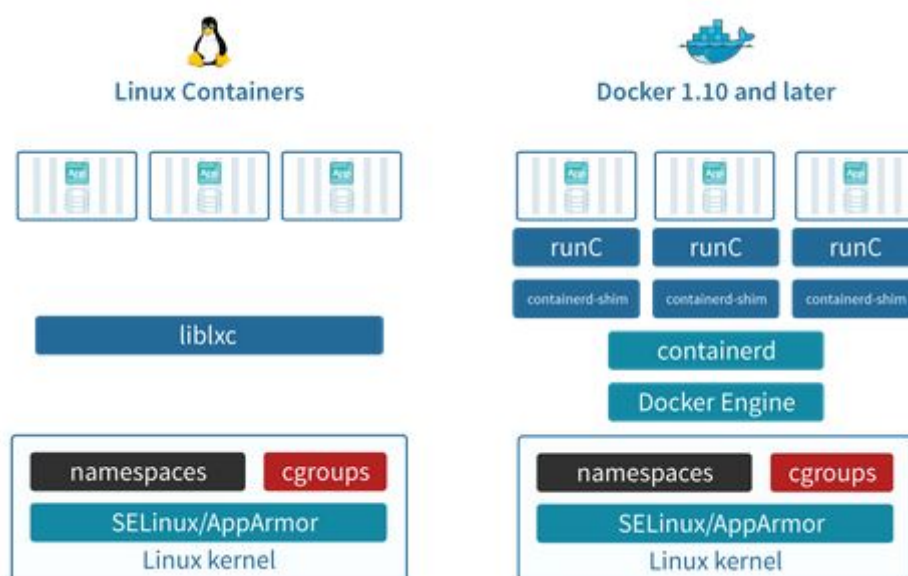


figure 3: LCX and Docker layer architecture

We saw that container, at the contrary of VM, is a software space allocated in os memory. If we introduce Linux Containers and Docker on a Linux OS we can see that they are the same base definition in the Linux Kernel. Indeed, they are composed by a namespace which is an abstract place designed to accommodate sets of terms belonging to the same repertoire, the same memory area. CGroups (control groups) is a feature of the Linux kernel to limit, count and isolate resource usage (CPU, memory, disk usage, etc.). Then we can see that LCX has a liblxc which help to create containers on it rather than Docker propose a more complex layer description that allow Docker containers to run on any OS.

Once we have an overview of main differences in layer architecture we can show in a table what are the main differences between the two technologies but we have to remember that there LCX can be a layer of Docker. Indeed LCX is the Linux containerization daemon where

a docker can rely on a Linux infrastructure. For example we will use both in practical labs because we will use a ubuntu OS.

	LCX	Docker
ressources	Dépend of OS Linux and also physical	More efficient because it is an improvement of LcX can be barely deployed
isolation	Have a lot of dependencies with linux OS	More because it not dependant of any OS types.
containerization level	Type 2 run in Linux OS	Type 2 run on an OS
storage management	simple Store backup on a single folder.	more sophisticated solution for container storage and image management.
configuration	More files to config in linux treemap, so it is more difficult to set up	A single recipe file make it easier for developer to manage.
security	is not a secure virtualization technology for multi-tenant environments we need to had additional containment technologies, such as AppArmor, may be used to provide better isolation between containers,	More secure more isolation between multi-tenant environment but less than VM. In entreprise version it provides default configurations that offer greater protection for applications running on top of Docker Engine and across both orchestrators Docker Swarm and Kubernetes. The platform establishes strong secure defaults, while still leaving the controls with the admin to change configurations and policies as needed.
portability	Portable but depend on OS linux host	Totally portable independent from any OS layers
Production env	Can not be deploy in production because no isolation and not secure	Can provide more security and performances in deployment or failure recovery.

As a bilan of this table so Docker actually is based on LCX for linux containerization. We can say that Docker is an improvement of LCX.

Network connection modes for virtualization hosts : Bridge vs NAT

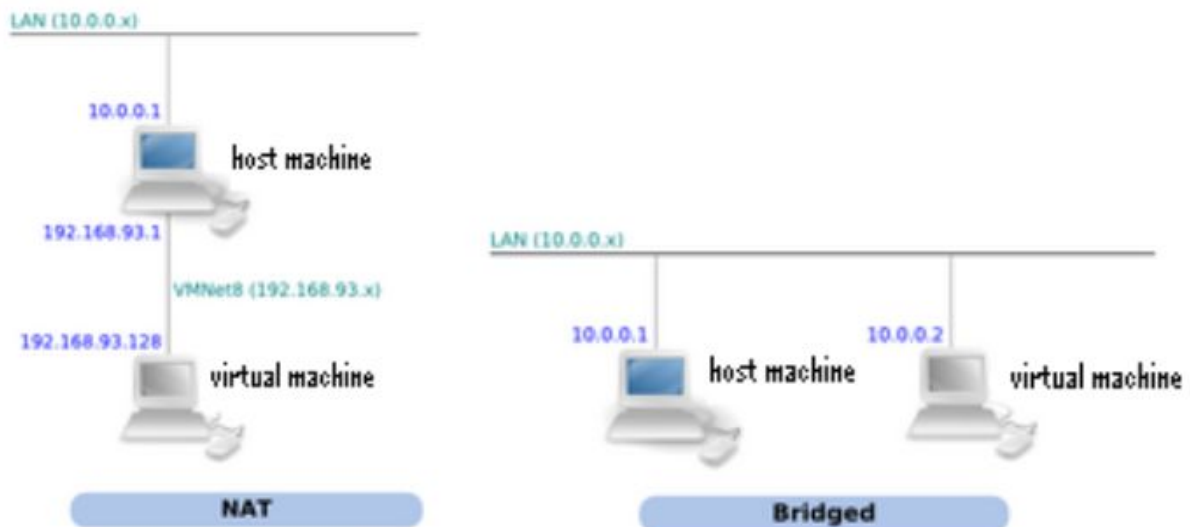


figure 4 : NAT vs Bridge connexion

An external address, usually routable, is the "outside" of the NAT. The machines behind the NAT have an "inside" address that is usually non-routable. So as you can see on the figure the Virtual machine is considered behind host machine. We thus need of an association table which mach "inside" address with a "outside" address to be able to go over the Internet or just over the outside network.

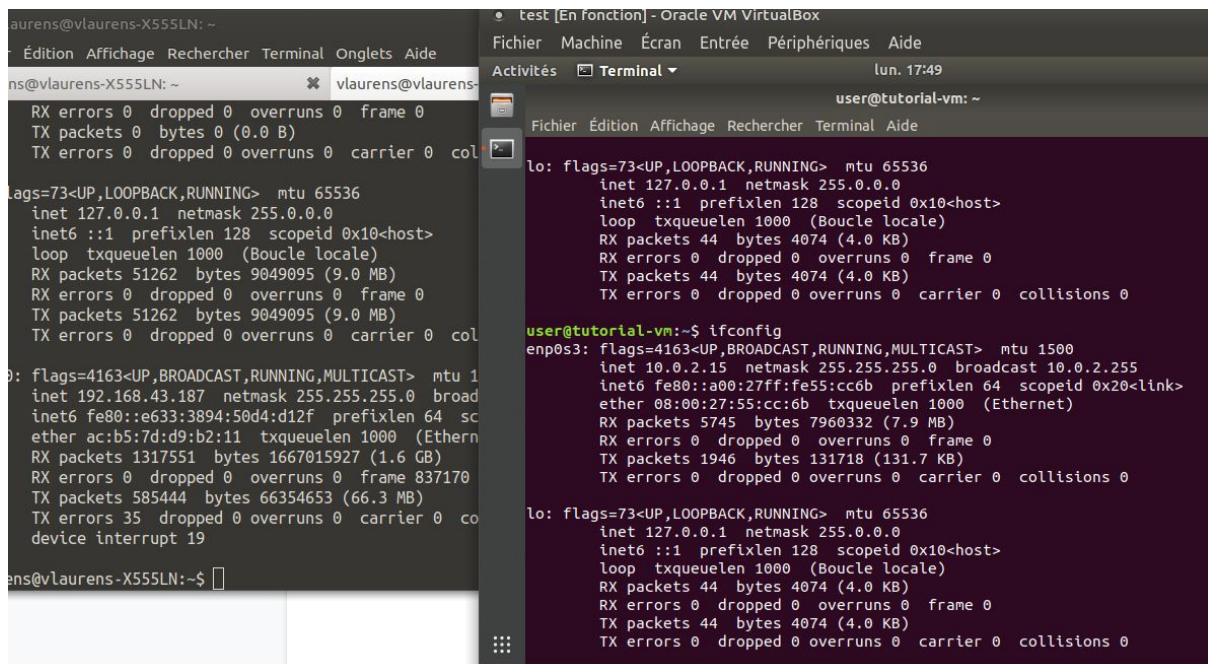
Bridged mode acts just like a switch between VM interface and host machine interface and the VM is plugged into a port on it. Everything acts the same as if it were another regular machine attached to that network.

Practical part

VirtualBox VM and Docker containerization

[VM] Creating and configuring a VM

In this lab to create a VM we are on a ubuntu OS on what we we add VirtualBox hypervisor and we create a ubuntu 18.0.4 virtual machine. Once created the first step was to test VM connectivity to The Internet. So first step is to configure IP address to its interface (here eth0) As you can see the IP address of the VM is not in the same network as host machine. It is because, as we saw in previous section, we settled a NAT connexion between VM and host machine it means that VM as a non-routable address and can't be accessed from host machine and outside but VM can access the outside and go over the Internet.



```

laurens@vlaurens-X555LN: ~
ns@vlaurens-X555LN: ~
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 0 bytes 0 (0.0 B)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
inet 127.0.0.1 netmask 255.0.0.0
inet6 ::1 prefixlen 128 scopeid 0x10<host>
loop txqueuelen 1000 (Boucle locale)
RX packets 44 bytes 4074 (4.0 KB)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 44 bytes 4074 (4.0 KB)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
inet 192.168.43.187 netmask 255.255.255.0 broadcast 192.168.43.255
inet6 fe80::e633:3894:50d4:d12f prefixlen 64 scopeid 0x20<link>
ether ac:b5:7d:d9:b2:11 txqueuelen 1000 (Ethernet)
RX packets 1317551 bytes 1667015927 (1.6 GB)
RX errors 0 dropped 0 overruns 0 frame 837170
TX packets 585444 bytes 66354653 (66.3 MB)
TX errors 35 dropped 0 overruns 0 carrier 0 collisions 0
device interrupt 19

laurens@vlaurens-X555LN:~$

test [En fonction] - Oracle VM VirtualBox
Fichier Machine Écran Entrée Périphériques Aide
Activités Terminal lun. 17:49
user@tutorial-vm: ~
Fichier Édition Affichage Rechercher Terminal Aide

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
inet 127.0.0.1 netmask 255.0.0.0
inet6 ::1 prefixlen 128 scopeid 0x10<host>
loop txqueuelen 1000 (Boucle locale)
RX packets 44 bytes 4074 (4.0 KB)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 44 bytes 4074 (4.0 KB)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

user@tutorial-vm:~$ ifconfig
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
inet 10.0.2.15 netmask 255.255.255.0 broadcast 10.0.2.255
inet6 fe80::a00:27ff:fe55:cc6b prefixlen 64 scopeid 0x20<link>
ether 08:00:27:55:cc:6b txqueuelen 1000 (Ethernet)
RX packets 5745 bytes 7960332 (7.9 MB)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 1946 bytes 131718 (131.7 KB)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
inet 127.0.0.1 netmask 255.0.0.0
inet6 ::1 prefixlen 128 scopeid 0x10<host>
loop txqueuelen 1000 (Boucle locale)
RX packets 44 bytes 4074 (4.0 KB)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 44 bytes 4074 (4.0 KB)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

```

figure 5 : Interface ip address of VM and host.

[VM] Testing the VM connectivity

As we said we will test if we can ping from VM the Internet (Results in figure 6).

```
user@tutorial-vm:~$ ping www.google.com
PING www.google.com (216.58.213.132) 56(84) bytes of data.
64 bytes from par21s03-in-f132.1e100.net (216.58.213.132): icmp_seq=1 ttl=63 time=
60.0 ms
64 bytes from par21s03-in-f132.1e100.net (216.58.213.132): icmp_seq=2 ttl=63 time=
78.9 ms
^C
--- www.google.com ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1002ms
rtt min/avg/max/mdev = 60.096/69.519/78.942/9.423 ms
```

figure 6 : Ping test access Internet from VM

Ouf we can access the Internet from VM. Now let's demonstrate that the VM can be accessed from outside network it is means from the host.

<pre>vlaurens@vlaurens-X555LN:~\$ ping 10.0.2.15 PING 10.0.2.15 (10.0.2.15) 56(84) bytes of data. ^C --- 10.0.2.15 ping statistics --- 8 packets transmitted, 0 received, 100% packet loss, time vlaurens@vlaurens-X555LN:~\$</pre>	<pre>user@tutorial-vm:~\$ ping 192.168.43.187 PING 192.168.43.187 (192.168.43.187) 56(84) bytes of data. 64 bytes from 192.168.43.187: icmp_seq=1 ttl=63 time=0.335 ms 64 bytes from 192.168.43.187: icmp_seq=2 ttl=63 time=0.496 ms 64 bytes from 192.168.43.187: icmp_seq=3 ttl=63 time=0.410 ms 64 bytes from 192.168.43.187: icmp_seq=4 ttl=63 time=0.377 ms ^C --- 192.168.43.187 ping statistics --- 4 packets transmitted, 4 received, 0% packet loss, time 3054ms rtt min/avg/max/mdev = 0.335/0.404/0.496/0.062 ms</pre>
---	---

figure 7 : Ping test connexion from host to guest VM

We see that host machine can't access VM because there is no match in forwarding table so the public IP address of host can't be translated into private IP address understandable by VM. So that explain why ping from host to VM failed as pictured on the left black terminal rather than ping from the VM to the host on right terminal succeeded.

[VM] Set up the "missing" connectivity

As we observe previously NAT offer a one-way connectivity from VM to host so now we will activate the second way from Host to the VM.

In order to offer a bidirectional access we activate the forwarding table as already explained in first section to match a public IP address and port with a private IP address and port.

The configuration is done as describe on figure 8.

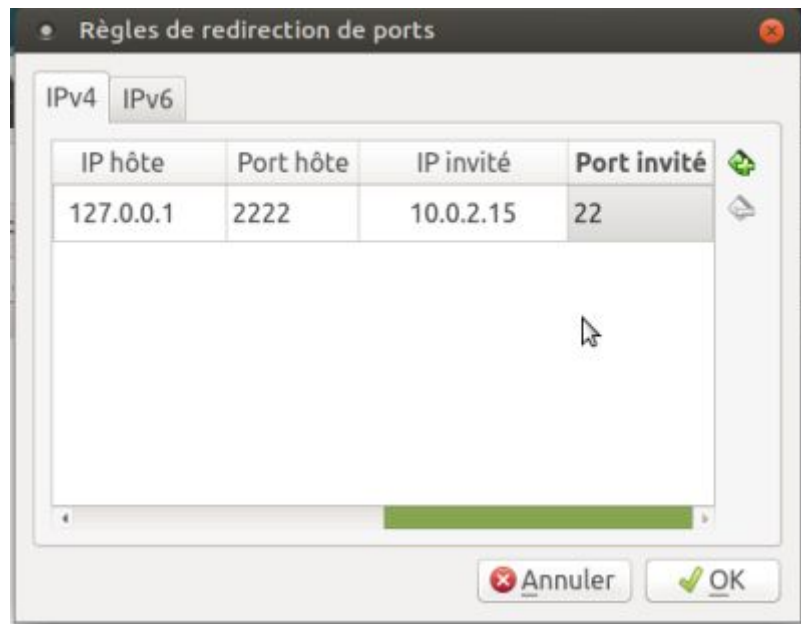


figure 8 : Forwarding table

We try one more time to access VM from Host by opening a SSH connexion and we see that now connexion is possible as shown in figure 9. That ssh connexion is possible because as describe on figure 8, we create a match between localhost address of host and port 2222 with VM IP address on port 22. To explain more clearly, when we try to connect VM we will target localhost IP and port number 2222 and it will be automatically redirect on address 10.0.2.15 of VM and on port 22 for SSH.

```
vlaurens@vlaurens-X555LN:~/Documents/5ISS/Cloud$ ssh user@127.0.0.1
ssh: connect to host 127.0.0.1 port 22: Connection refused
vlaurens@vlaurens-X555LN:~/Documents/5ISS/Cloud$ ssh -p 2222 user@127.0.0.1
The authenticity of host '[127.0.0.1]:2222 ([127.0.0.1]:2222)' can't be established.
ECDSA key fingerprint is SHA256:RwbyEuLL7lk++LikmWRQzNhQLfvIBjMtMQV/WHg8lo.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '[127.0.0.1]:2222' (ECDSA) to the list of known hosts
.
user@127.0.0.1's password:
Welcome to Ubuntu 18.04.3 LTS (GNU/Linux 4.15.0-65-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

System information disabled due to load higher than 1.0

 * Overheard at KubeCon: "microk8s.status just blew my mind".

https://microk8s.io/docs/commands#microk8s.status
```

figure 9 : SSH access possible

[VM] VM duplication

Now I would to show that we can create as much as VM we want by cloning the first VM we created with command :

VBoxManage clonemedium "chemin\vers\disk.vmdk" "chemin\vers\disk-copy.vmdk"

We can then access them by using localhost IP, because the most important is to allocate a unique port number to localhost address in order to the pair be unique. As VM side the port is always 22 because is the TCP port allocated to SSH protocol but the IP address allocated dynamically by DHCP or manually has to be uniq. To sum up in our forwarding table we have to be on both side a unique pair address:port to process match between public address and private address.


```
vlaurens@vlaurens-X555LN: ~
65 mises à jour de sécurité.

Last login: Mon Jan  6 19:01:29 2020 from 10.0.2.2
user@tutorial-vm:~$ ifconfig
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
inet 10.0.2.4 netmask 255.255.255.0 broadcast 10.0.2.255
inet6 fe80::a00:27ff:fe2d:b7d2 prefixlen 64 scopeid 0x2
ether 08:00:27:2d:b7:d2 txqueuelen 1000 (Ethernet)
RX packets 152 bytes 18545 (18.5 KB)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 150 bytes 20161 (20.1 KB)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
inet 127.0.0.1 netmask 255.0.0.0
inet6 ::1 prefixlen 128 scopeid 0x10<host>
loop txqueuelen 1000 (Boucle locale)
RX packets 8 bytes 480 (480.0 B)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 8 bytes 480 (480.0 B)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions

user@tutorial-vm:~$ https://trendy.letudiant

user@tutorial-vm:~$
```

figure 10 : IP address of two host VM

We can with forwarding rule run two VM simultaneously but we have to give a different host port I took 2222 for first VM and VM 2223 for copy

Règles de redirection de ports					
IPv4 IPv6					
Nom	Protocole	IP hôte	Port hôte	IP invité	Port invité
Rule 1	TCP	127.0.0.1	2222	10.0.2.15	22
Rule 2	TCP	127.0.0.1	2223	10.0.2.4	22

figure 11 : Forwarding table with two host VM

Now we are able to establish a connexion between host machine and each VM using ssh command:

ssh -P 2222 root@127.0.0.1 and

ssh -P 2223 root@127.0.0.1

[Container] Docker containers provisioning

In this part we will learn to retrieve a Docker container image from docker hub : <https://hub.docker.com/>, and create a container using the image.

So first we use the command "Docker pull <imageName> to retrieve the image artifact.

```
user@tutorial-vm:~$ docker pull ubuntu
Using default tag: latest
Got permission denied while trying to connect to the Docker daemon socket at unix:
//var/run/docker.sock: Post http://%2Fvar%2Frun%2Fdocker.sock/v1.39/images/create?fromImage=ubuntu&tag=latest: dial unix /var/run/docker.sock: connect: permission denied
user@tutorial-vm:~$ sudo docker pull ubuntu
Using default tag: latest
latest: Pulling from library/ubuntu
2746a4a261c9: Pull complete
4c1d20cdee96: Pull complete
0d3160e1d0de: Pull complete
c8e37668deea: Pull complete
Digest: sha256:250cc6f3f3ffc5cdaa9d8f4946ac79821aafb4d3afc93928f0de9336eba21aa4
Status: Downloaded newer image for ubuntu:latest
```

In a second time, we run a container named ct1 that we build with ubuntu image that we have previously pulled.

```
user@tutorial-vm:~$ sudo docker run --name ct1 -it ubuntu
root@128c993bb90f:/#
```

Once the command run we see that identification of machine and user user@machineName in terminal changed because we are actually connected to the container as root.

So we can process any action as on the ubuntu host we can see that container has its own ip address (figure 11) and can be pinged by host or can ping host (figure 13 and 14). It can also ping the Internet (figure 12).

```
root@128c993bb90f:/# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 172.17.0.2 netmask 255.255.0.0 broadcast 172.17.255.255
    ether 02:42:ac:11:00:02 txqueuelen 0 (Ethernet)
    RX packets 9531 bytes 18266562 (18.2 MB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 5810 bytes 318974 (318.9 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    loop txqueuelen 1000 (Local Loopback)
    RX packets 0 bytes 0 (0.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 0 bytes 0 (0.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

figure 11: IP Parameters of container

```
root@128c993bb90f:/# ping www.google.fr
PING www.google.fr (172.217.19.227) 56(84) bytes of data.
64 bytes from par21s11-in-f3.1e100.net (172.217.19.227): icmp_seq=1 ttl=55 time=
17.4 ms
64 bytes from par21s11-in-f3.1e100.net (172.217.19.227): icmp_seq=2 ttl=55 time=
18.5 ms
64 bytes from par21s11-in-f3.1e100.net (172.217.19.227): icmp_seq=3 ttl=55 time=
19.0 ms
^C
--- www.google.fr ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2003ms
rtt min/avg/max/mdev = 17.433/18.359/19.066/0.693 ms
```

figure 12 : Ping an Internet resource from Docker

```
root@128c993bb90f:/# ping 10.0.2.4
PING 10.0.2.4 (10.0.2.4) 56(84) bytes of data.
64 bytes from 10.0.2.4: icmp_seq=1 ttl=64 time=0.058 ms
64 bytes from 10.0.2.4: icmp_seq=2 ttl=64 time=0.076 ms
64 bytes from 10.0.2.4: icmp_seq=3 ttl=64 time=0.092 ms
^C
--- 10.0.2.4 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2052ms
rtt min/avg/max/mdev = 0.058/0.075/0.092/0.015 ms
```

figure 13: Ping the VM from Docker

```
user@tutorial-vm:~$ ping 172.17.0.2
PING 172.17.0.2 (172.17.0.2) 56(84) bytes of data.
64 bytes from 172.17.0.2: icmp_seq=1 ttl=64 time=0.053 ms
64 bytes from 172.17.0.2: icmp_seq=2 ttl=64 time=0.063 ms
^C
--- 172.17.0.2 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1019ms
rtt min/avg/max/mdev = 0.053/0.058/0.063/0.005 ms
```

figure 14 : Ping the Docker from the VM

The docker is accessible from host VM and VM can access docker, Docker container can access Internet as the VM did.

Then we create a second instance of container as describe on figure 15 where we install nano on the container.

```
user@tutorial-vm:~$ sudo docker ps
[sudo] Mot de passe de user :
CONTAINER ID   IMAGE      COMMAND                  CREATED
STATUS        PORTS      NAMES
967dba00ff20   ubuntu    "/bin/bash"             2 minutes ago
Up 2 minutes   0.0.0.0:2223->22/tcp    ct2
128c993bb90f   ubuntu    "/bin/bash"             31 minutes ago
Up 31 minutes                ct1
```

figure 15 : Container list

Now we will duplicate container instance 2 as we did with VM. So first step is to create snapshot of docker container.

So we start to create a tag to identify changes to be made in the image that we can identify by processing "sudo docker images" command and then use the image ID to tage changes on this image. We also need to define a tag name and a version number.

```
sudo docker tag 549b9b86cb8d ubuntu/test:version1.0
```

```
user@tutorial-vm:~$ sudo docker tag 549b9b86cb8d ubuntu/test:version1.0
user@tutorial-vm:~$
```

Once the tag is created we can apply changes on a given image by committing tag.

```
sudo docker ps
```

```
sudo docker commit 967dba00ff20 ubuntu/test:version1.0
```

```
user@tutorial-vm:~$ sudo docker commit 967dba00ff20 ubuntu/test:version1.0
sha256:c619d52a6b3d494da99a19d51f1f7bca69efa6dc10ea01faeda0b11226d048fa
```

We can see that the image committed before has been created

correctly.

```
user@tutorial-vm:~$ sudo docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED
ubuntu/test	version1.0	c619d52a6b3d	8 minutes ago
ubuntu	latest	549b9b86cb8d	2 weeks ago

Once the image has been created we can run a new container which is exactly the same as container 1 with the same software installed.

We can also decide to delete container using its ID but before we need to stop it:

```
user@tutorial-vm:~$ sudo docker stop 967dba00ff20
967dba00ff20
user@tutorial-vm:~$ sudo docker rm 967dba00ff20
967dba00ff20
```

Now we have created a snapshot of container instance 2 and delete ct1 we will create an third instance which is the copy of instance 2.

```
sudo docker run --name ct3 -it ubuntu/test:version1.0
```

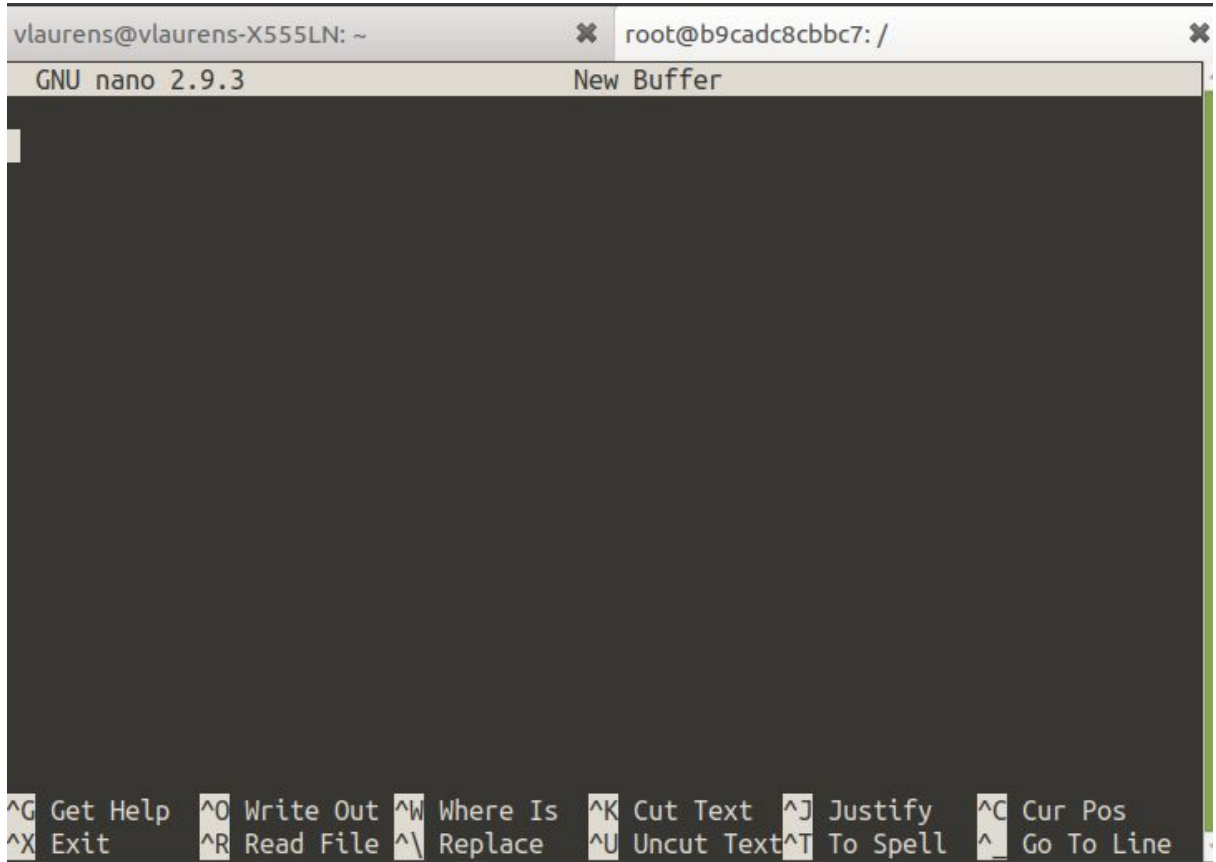
We try to use “nano” editor we have installed on ct2 which is the source of the snapshot.

```
user@tutorial-vm:~$ sudo docker run --name ct3 -it ubuntu/test:version1.0
[sudo] Mot de passe de user :
root@b9cad8cbbc7:/# nano
```

So hopefully we have nano on ct3 the copy of ct2.

So when I was connected I start to write na.. on prompt and did a tabulation and nano was printed it is mean the command is correctly installed.

Furthermore we can edit new document with nano.



As bilan we can say that we learn to create copies of an existing container instance and deploy those copies.

Now we learnt how to create container instance, how to test their connectivity, to add packages on it and finally learnt to make copy of containers instances, we will now try to automate creation of docker container.

To do so, we will write in a file (DockerFile) the recipe of container creation.

The DockerFile is composed as follow of a set of command we describe:

```
FROM ubuntu
RUN apt update -y
RUN apt install -y nano
CMD ["/bin/bash"]
```

So “From” target the image to use/deploy

“RUN” represent the command to do after creation of container as we did on previous questions.

“CMD” means that command ran are bash commands

we can see also “script” if we want to run a script.

```
sudo docker build -t ubuntu/test:version1.0 -f myDocker.dockerfile .
```

Docker build will build container , using image Repo:TAG and apply recipe described on “myDocker.dockerfile.

Indeed option “-t” allow to target a container image using is tag.

Then “-f” option allows to specify a given recipe i.e. a specific DockerFile.

Finally the “.” is the PATH specifies where to find the files for the “context” of the build on the Docker daemon

So at the and we should obtain a container CT4 with exactly the same configuration than CT3.

So below the result of command. Notice that during the build process a step has already been builded before so it will not be build one more time.

```
user@tutorial-vm:~$ sudo docker build -t ubuntu/test:version1.0 -f myDocker.dockerfile .
[sudo] Mot de passe de user :
Sending build context to Docker daemon 1.987MB
Step 1/4 : FROM ubuntu
--> 549b9b86cb8d
Step 2/4 : RUN apt update -y
--> Running in 5354d3437db5

WARNING: apt does not have a stable CLI interface. Use with caution in scripts.

Get:1 http://security.ubuntu.com/ubuntu bionic-security InRelease [88.7 kB]
Get:2 http://archive.ubuntu.com/ubuntu bionic InRelease [242 kB]
Get:3 http://archive.ubuntu.com/ubuntu bionic-updates InRelease [88.7 kB]
Get:4 http://security.ubuntu.com/ubuntu bionic-security/multiverse amd64 Packages [6781 B]
Get:5 http://security.ubuntu.com/ubuntu bionic-security/main amd64 Packages [761 kB]
Get:6 http://archive.ubuntu.com/ubuntu bionic-backports InRelease [74.6 kB]
Get:7 http://security.ubuntu.com/ubuntu bionic-security/restricted amd64 Packages [19.2 kB]
Get:8 http://security.ubuntu.com/ubuntu bionic-security/universe amd64 Packages [795 kB]
Get:9 http://archive.ubuntu.com/ubuntu bionic/universe amd64 Packages [11.3 MB]
Get:10 http://archive.ubuntu.com/ubuntu bionic/restricted amd64 Packages [13.5 kB]
Get:11 http://archive.ubuntu.com/ubuntu bionic/main amd64 Packages [1344 kB]
Get:12 http://archive.ubuntu.com/ubuntu bionic/multiverse amd64 Packages [186 kB]
Get:13 http://archive.ubuntu.com/ubuntu bionic-updates/universe amd64 Packages [1324 kB]
Get:14 http://archive.ubuntu.com/ubuntu bionic-updates/main amd64 Packages [1072 kB]
Get:15 http://archive.ubuntu.com/ubuntu bionic-updates/multiverse amd64 Packages [10.8 kB]
Get:16 http://archive.ubuntu.com/ubuntu bionic-updates/restricted amd64 Packages [35.0 kB]
Get:17 http://archive.ubuntu.com/ubuntu bionic-backports/universe amd64 Packages [4244 B]
Get:18 http://archive.ubuntu.com/ubuntu bionic-backports/main amd64 Packages [2496 B]
Fetched 17.4 MB in 20s (871 kB/s)
```

....

```
Reading state information...
Suggested packages:
  spell
The following NEW packages will be installed:
  nano
0 upgraded, 1 newly installed, 0 to remove and 0 not upgraded.
Need to get 231 kB of archives.
After this operation, 778 kB of additional disk space will be used.
Get:1 http://archive.ubuntu.com/ubuntu bionic/main amd64 nano amd64 2.9.3-2 [231 kB]
debconf: delaying package configuration, since apt-utils is not installed
Fetched 231 kB in 2s (125 kB/s)
Selecting previously unselected package nano.
(Reading database ... 4046 files and directories currently installed.)
Preparing to unpack .../nano_2.9.3-2_amd64.deb ...
Unpacking nano (2.9.3-2) ...
Setting up nano (2.9.3-2) ...
update-alternatives: using /bin/nano to provide /usr/bin/editor (editor) in auto mode
update-alternatives: warning: skip creation of /usr/share/man/man1/editor.1.gz because associated file /usr/share/man/man1/
itor) doesn't exist
update-alternatives: using /bin/nano to provide /usr/bin/pico (pico) in auto mode
update-alternatives: warning: skip creation of /usr/share/man/man1/pico.1.gz because associated file /usr/share/man/man1/r
) doesn't exist
Removing intermediate container 1d5617248729
--> a511faa17656
Step 4/4 : CMD ["/bin/bash"]
--> Running in 6dc87c85e195
Removing intermediate container 6dc87c85e195
--> 2988aa41e7a4
Successfully built 2988aa41e7a4
Successfully tagged ubuntu/test:version1.0
```


Once we run the Docker file and build new container we can see it appeared on container list and the image we use in docker image list:

```
user@tutorial-vm:~$ sudo docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
ubuntu/test	version1.0	2988aa41e7a4	5 minutes ago	93.1MB
<none>	<none>	c619d52a6b3d	About an hour ago	93.1MB
ubuntu	latest	549b9b86cb8d	2 weeks ago	64.2MB

```
user@tutorial-vm:~$ sudo docker ps -a
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
aebf0c8b5407	ubuntu	"/bin/bash"	About a minute ago	Exited (0) About a minute ago		ct4
b9cad8cbbc7	c619d52a6b3d	"/bin/bash"	45 minutes ago	Exited (0) 25 minutes ago		ct3
128c993bb90f	ubuntu	"/bin/bash"	2 hours ago	Exited (127) 45 minutes ago		ct1

Now that we know how to install containers, create, configure and remove them. Idem for VM.

We will deploy them on OpenStack

In this part we created a private network of machine but our machine in the network are not physical but it is our VM and containers.

Now we will migrate this network on remote servers (openstack at INSAT).

Application of Virtualization technologies in Cloud infrastructure

CT creation and configuration on OpenStack

In this section we will learn to deploy different virtualization technologies on an INSA cloud infrastructure using OpenStack.

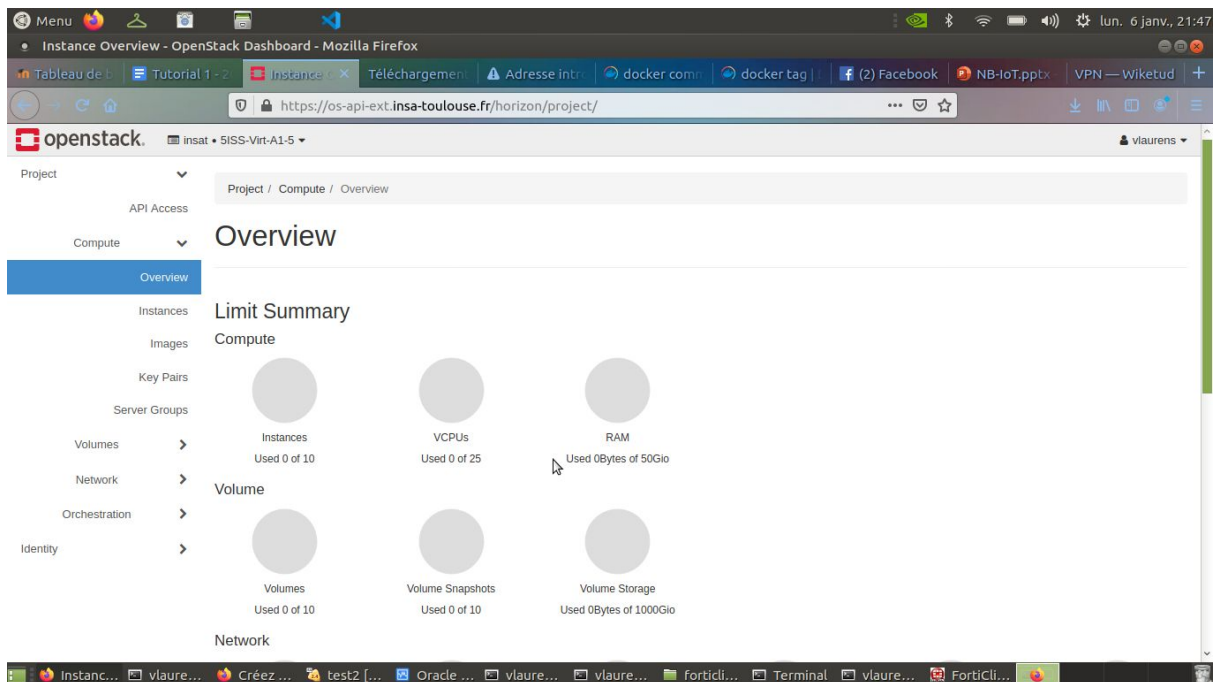


Figure 16 : first look of openStack

Create Image

Image Details

Metadata

Image Details

Specify an image to upload to the Image Service.

Image Name*

myImageUbuntu

Image Description

My Labs image

Image Source

File*

Browse...

disk2.vmdk

Format*

VMDK - Virtual Machine Disk

Image Requirements

Kernel

Choose an image

Ramdisk

Choose an image

Architecture

Minimum Disk (GB)

4

Minimum RAM (MB)

512

figure 17 : Interface to create a new VM image

We first added our OS Image on OpenStack to be able to create VM instances.

Creation of Network infrastructure

Then we created a Private Network that we connected to Public network by a router.
When we create a Router we can go on interface because of following error



We have created the private network and an instance de machine cirros

The machine console is accessible on menu instance/console.

We can observe that DHCP give an address automatically to our machine instance.

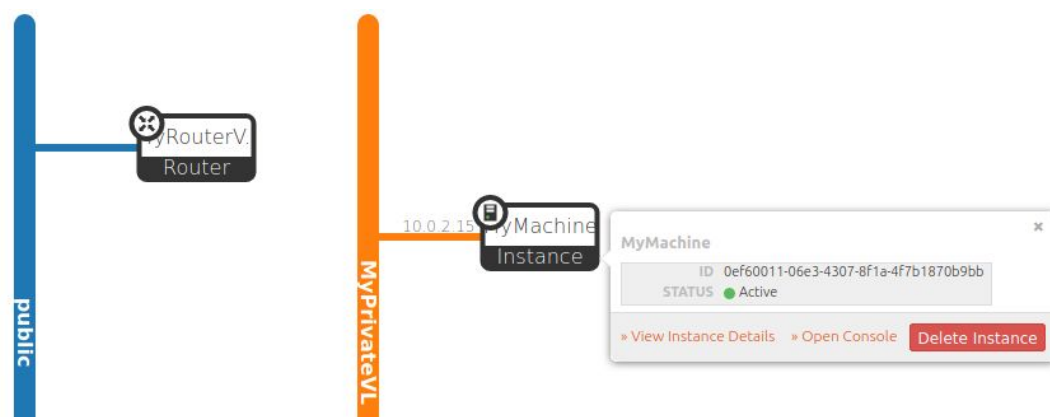


figure 18 : Network topology in construction

Then we will connect a private network hosting our VM instances via Router to public network. So we will need to set up security group to limit access and protect access to our private network.

I open Console on Chromium navigator to can access the prompt as shown on figure 19.

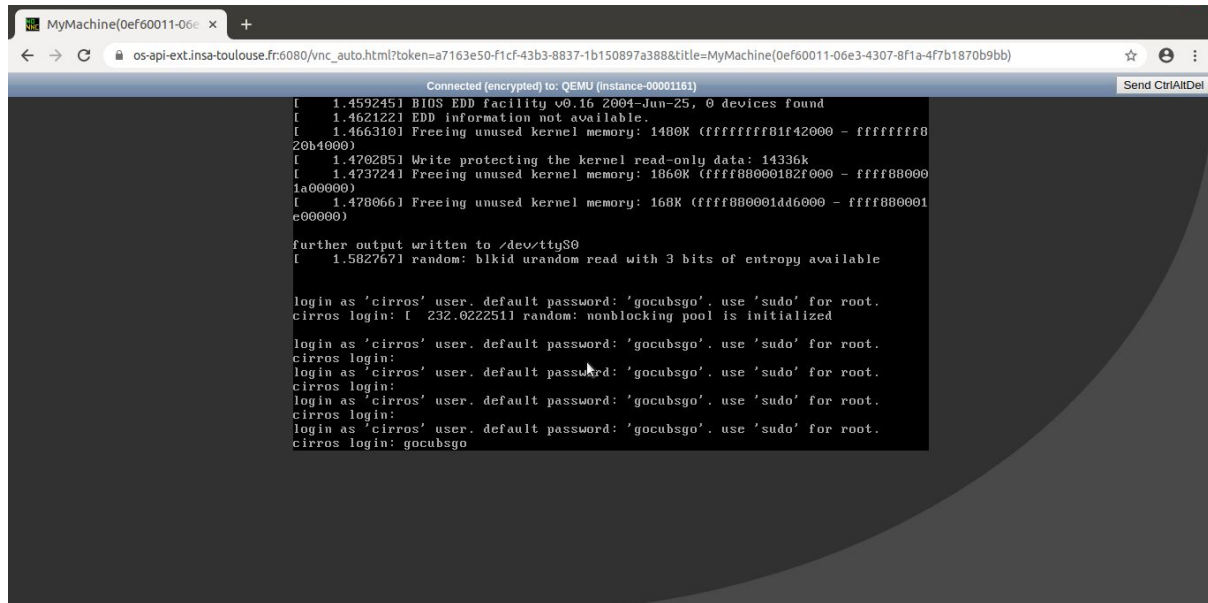


figure 19 : Console of a VM in OpenStack

Connectivity Tests

Now we will do some connectivity tests:

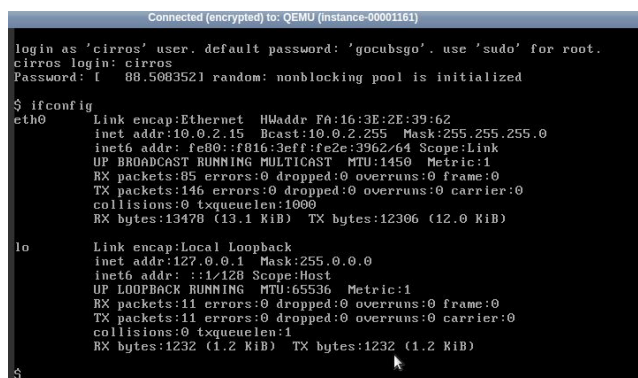


figure 20: VM instance private IP address

Now let's test the connectivity from machine to internet

Ping to internet doesn't work. But it is normal we have not define the connection between the

private and public networks with router.

So we install connectivity to internet by installing a router which link private and public network together because it offer a route between the two networks.

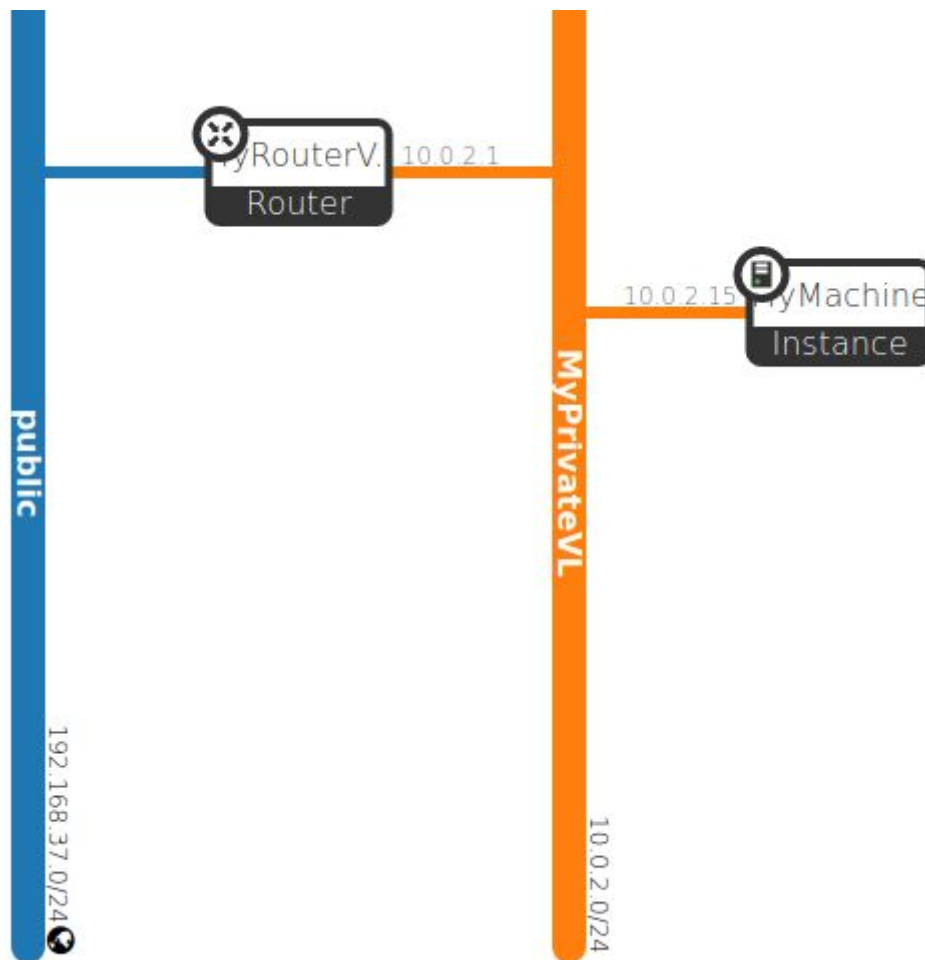


Figure 21: Final Topology : Connexion of private network hosting VM to public network

We start to test if we can ping our gateway

```
$ ping 10.0.2.1
PING 10.0.2.1 (10.0.2.1): 56 data bytes
64 bytes from 10.0.2.1: seq=0 ttl=64 time=3.050 ms
64 bytes from 10.0.2.1: seq=1 ttl=64 time=0.520 ms
64 bytes from 10.0.2.1: seq=2 ttl=64 time=0.360 ms
^C
--- 10.0.2.1 ping statistics ---
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max = 0.360/1.310/3.050 ms
```

figure 22 : Successful ping from VM to Input interface (private network side) of the Router

Then that we are sure the gateway is accessible we test the connectivity to the Internet by launching following ping command.

```
$ ping 8.8.8.8
PING 8.8.8.8 (8.8.8.8): 56 data bytes
64 bytes from 8.8.8.8: seq=0 ttl=51 time=10.810 ms
64 bytes from 8.8.8.8: seq=1 ttl=51 time=7.408 ms
^C
--- 8.8.8.8 ping statistics ---
2 packets transmitted, 2 packets received, 0% packet loss
round-trip min/avg/max = 7.408/9.109/10.810 ms
```

figure 23 : Successful ping from VM to the Internet

So we have an access from machine to the internet

Now we assign a floating ip address to the VM on private network to be accessed from Public network. It is like a forwarding Table in NAT network, the floating IP is a public address which is bound to a private address and which allows a VM instance to be accessed from public network.

We can see bellow the process of creation and allocation of a floating IP address.

Allocate Floating IP

Pool *

public

Description

MyFloatIP

DNS Domain

|

DNS Name

Description:

Allocate a floating IP from a given floating IP pool.

Project Quotas

Floating IP

0 of 50 Used

Cancel

Allocate IP

Figure 24 : Floating IP Creation interface

Then we attach it to the instance via the interface describe in figure 25. For accessing this interface we have to go on menu "instance" then choice our instance and in the dropdown menu of the instance select the "Assign a floating IP address to instance".

Manage Floating IP Associations ✕

IP Address *

192.168.37.26
+

Select the IP address you wish to associate with the selected instance or port.

Port to be associated *

MyMachine: 10.0.2.15

Cancel

Associate

figure 25 : Allocation interface of a floating IP address to a specific VM

and we finish association of our floating address to the private IP address of the instance.

Displaying 1 item

<input type="checkbox"/>	Instance Name	Image Name	IP Address	Flavor	Key Pair	Status	Availability Zone	Task	Power State
<input type="checkbox"/>	MyMachine	cirros	10.0.2.15 Floating IPs: 192.168.37.26	nano	-	Active	nova	None	Running

Displaying 1 item

figure 26 : Result of floating IP association

We see association made.

Configuring security groups

Once the floating IP address is associated to our VM, then our VM is accessible from the public network we have thus to define security rules to limit access to the VM and prevent possible malicious behavior.

So we show own to add a security group to authorize ssh connexion on the VM.

So we need to add a security group to allow SSH by clicking on “Network” menu and selecting “Security Group”

<input type="checkbox"/>	Name	Security Group ID	Description	Actions
<input type="checkbox"/>	All ICMP	0f9ceb17-72ad-4c11-824f-45dfddb0fec		Manage Rules ▼
<input type="checkbox"/>	SSH	6910ad7f-9341-46c5-b100-b96dbee3cac3		Manage Rules ▼
<input type="checkbox"/>	default	eb8c65e6-135a-4b4e-9279-b9b5bc517d74	Default security group	Manage Rules

figure 27 : Creation of Security group

Then we associate the security group to a VM instance by going on the instance list interface and we choice our VM instance then in the dropdown menu we select “Edit Security Groups”

Edit Instance ✕

Information *
Security Groups

Add and remove security groups to this instance from the list of available security groups.

Warning: If you change security groups here, the change will be applied to all interfaces of the instance. If you have multiple interfaces on this instance and apply different security groups per port, use "Edit Port Security Groups" action instead.

All Security Groups

No security groups found.

Instance Security Groups

SSH	-
default	-

Cancel
Save

figure 28: Association of Security group to the VM instance

Then we test connexion by ssh to the machine and we can access it:


```
vlaurens@vlaurens-X555LN:~$ ssh cirros@192.168.37.26
The authenticity of host '192.168.37.26 (192.168.37.26)' can't be established.
ECDSA key fingerprint is SHA256:1G5lb8J99jeSNVKDUMKwVKmp2jowxKgBrilGqbD1PQM.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.37.26' (ECDSA) to the list of known hosts.
cirros@192.168.37.26's password:
$
```

figure 29 : SSH connexion from public machine to the VM on private network (Openstack) without using SSH key

We can see that we use the address public of machine and we can connect to the VM by only giving the user password is totally insecure and can't be accepted on a production machine. So to encrypted SSH connexion and don't use a simple password but by downloading a public key and generate associated private key. Then the connexion will be done by an exchange of public key encrypted by private key. It is the best security solution to protect a ssh connexion.

Now we try to connect via ssh to the machine. The banner warning show us that probably the private secret key has been create with a non secure access on OpenStack.

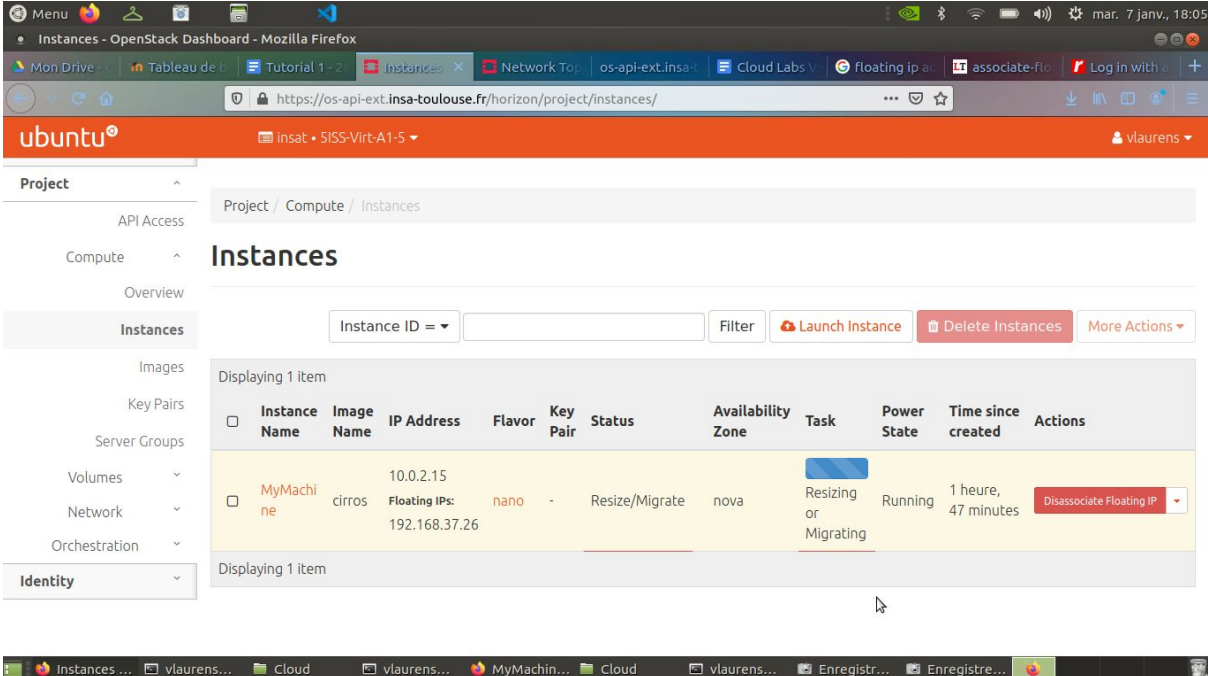
```
vlaurens@vlaurens-X555LN:~/Documents/5ISS/Cloud$ ssh -i ./MuKey.pem ssh cirros@192.168.37.26
ssh: Could not resolve hostname ssh: Temporary failure in name resolution
vlaurens@vlaurens-X555LN:~/Documents/5ISS/Cloud$ ssh ssh cirros@192.168.37.26
ssh: Could not resolve hostname ssh: Temporary failure in name resolution
vlaurens@vlaurens-X555LN:~/Documents/5ISS/Cloud$ ssh -i MuKey.pem cirros@192.168.37.26
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@                WARNING: UNPROTECTED PRIVATE KEY FILE!                @
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Permissions 0664 for 'MuKey.pem' are too open.
It is required that your private key files are NOT accessible by others.
This private key will be ignored.
Load key "MuKey.pem": bad permissions
cirros@192.168.37.26's password:
```

figure 30 : SSH connexion from public machine to the VM on private network (Openstack) using SSH key

Snapshot, restore and resize a VM

Resizing a VM

During labs we demonstrate one of the strength of a cloud infrastructure and OpenStack. Indeed, we demonstrate that we can resize a VM up/down while it is running and OpenStack will automatically restart the VM and process the size change as we can see on figure 31.



The screenshot shows the OpenStack Horizon dashboard in a Mozilla Firefox browser. The page title is 'Instances - OpenStack Dashboard - Mozilla Firefox'. The URL is 'https://os-api-ext.insa-toulouse.fr/horizon/project/instances/'. The dashboard shows the 'Instances' page with a table of instances. The instance 'MyMachine' is highlighted, showing it is in the 'Resizing or Migrating' state. The table columns include Instance Name, Image Name, IP Address, Flavor, Key Pair, Status, Availability Zone, Task, Power State, Time since created, and Actions.

Instance Name	Image Name	IP Address	Flavor	Key Pair	Status	Availability Zone	Task	Power State	Time since created	Actions
MyMachine	cirros	10.0.2.15 Floating IPs: 192.168.37.26	nano	-	Resize/Migrate	nova	Resizing or Migrating	Running	1 heure, 47 minutes	Disassociate Floating IP

figure 31 : Resizing up of VM instance size

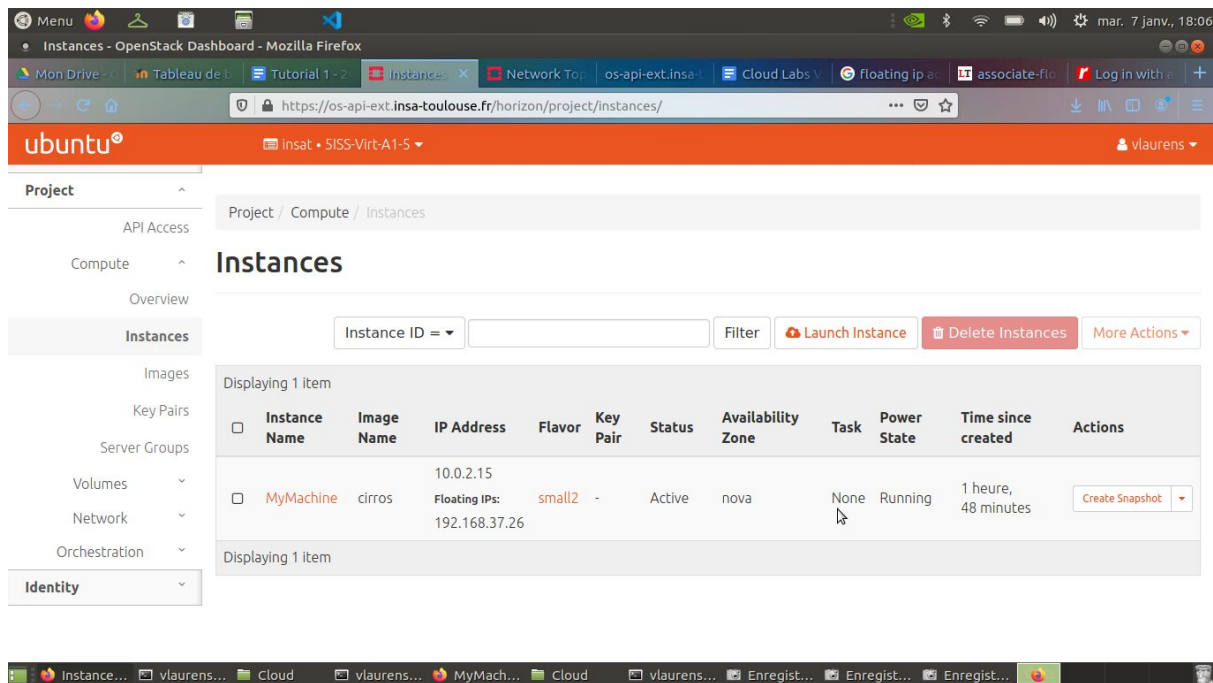
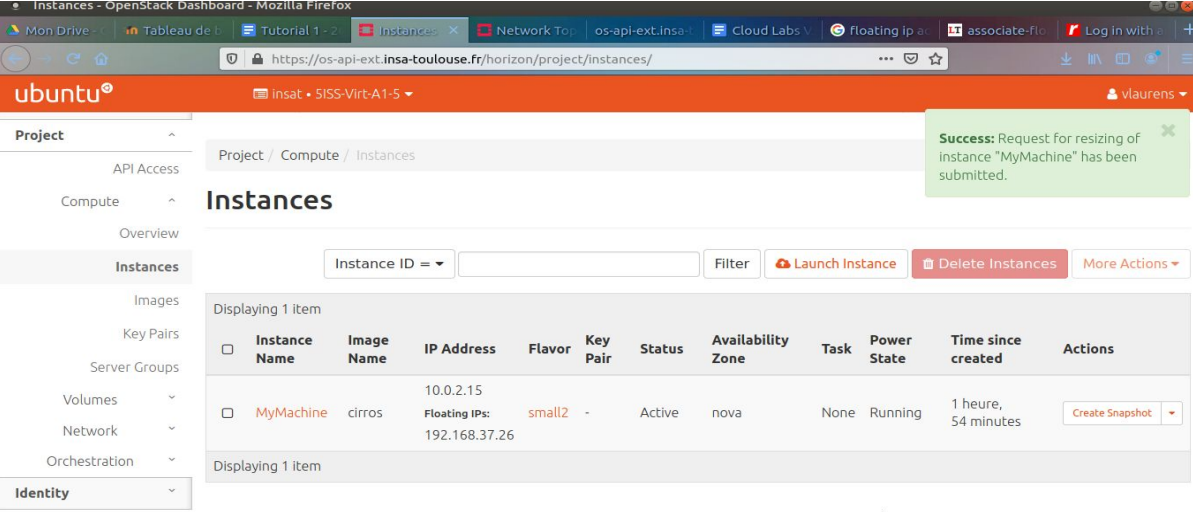


figure 32 : Result/State of the VM instance after reboot and resize

After to launch the resize of VM openStack ask to the administrator to confirm the reboot of the VM. If the size of the machine require more space than the space allocated after the resize. openstack write us a warning message.

We try next to resize down the VM and we have to know that the openstack process ended well as shown on figure 33.



Project / Compute / Instances

Instances

Instance ID = Filter Launch Instance Delete Instances More Actions

Displaying 1 item

	Instance Name	Image Name	IP Address	Flavor	Key Pair	Status	Availability Zone	Task	Power State	Time since created	Actions
<input type="checkbox"/>	MyMachine	cirros	10.0.2.15 Floating IPs: 192.168.37.26	small2	-	Active	nova	None	Running	1 heure, 54 minutes	Create Snapshot

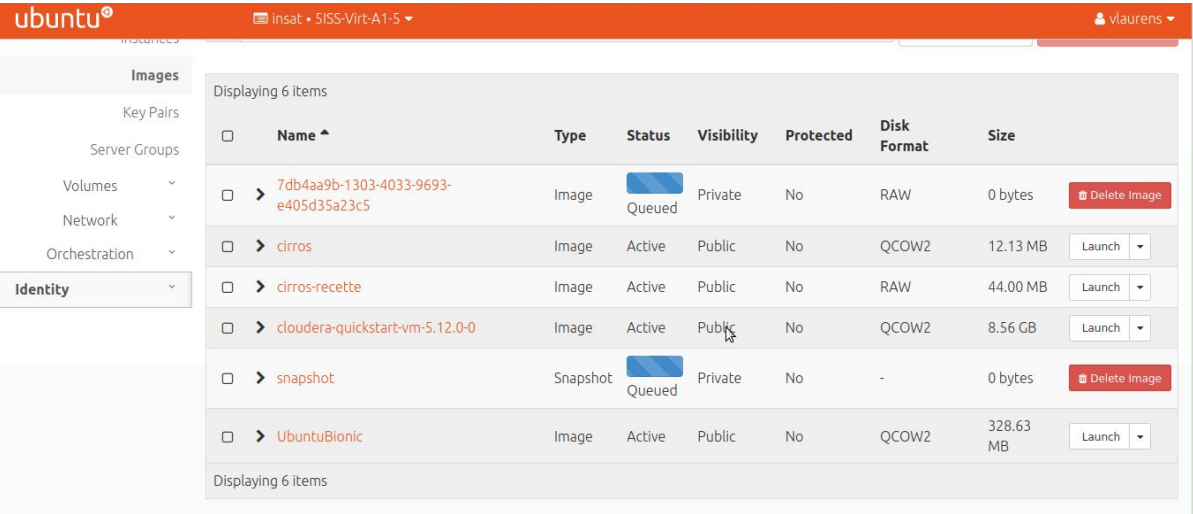
Displaying 1 item

Figure 33 : New size of instance : before tiny now small2

However sometimes the resize down of an instance is not effective because many reasons no rights to resize, no enough hard disk space available, etc.

Creation of a snapshot/Restore

In a cloud infrastructure we have to create new instance which are similar to another or simply create a backup of a machine to prevent any failure and prevent data lost.



Displaying 6 items

	Name	Type	Status	Visibility	Protected	Disk Format	Size	Actions
<input type="checkbox"/>	7db4aa9b-1303-4033-9693-e405d35a23c5	Image	Queued	Private	No	RAW	0 bytes	Delete Image
<input type="checkbox"/>	cirros	Image	Active	Public	No	QCOW2	12.13 MB	Launch
<input type="checkbox"/>	cirros-recette	Image	Active	Public	No	RAW	44.00 MB	Launch
<input type="checkbox"/>	cloudera-quickstart-vm-5.12.0-0	Image	Active	Public	No	QCOW2	8.56 GB	Launch
<input type="checkbox"/>	snapshot	Snapshot	Queued	Private	No	-	0 bytes	Delete Image
<input type="checkbox"/>	UbuntuBionic	Image	Active	Public	No	QCOW2	328.63 MB	Launch

Displaying 6 items

Figure 34 : Creation of Snapshot image

☐ > snapshot	Snapshot	Active	Private	No	QCOW2	97.38 MB	Launch	▼
--------------	----------	--------	---------	----	-------	----------	--------	---

We create instance from snapshot image called backup.

<input type="checkbox"/>	backup	cirros	10.0.2.5	nano	MuKey	Active	nova	None	Running	2 minutes	Create Snapshot
<input type="checkbox"/>	MyMachine	cirros	10.0.2.15 Floating IPs: 192.168.37.26	small2	-	Active	nova	None	Running	2 heures, 11 minutes	Create Snapshot

Figure 35 : Result showing new instance “backup”

Now we are able to manage a network and instances. Now we know how to open an access to the outside and protect and limit access we will deploy an infrastructure that host application first on a single private network and next we will deploy a Network as a Service infrastructure to isolate front and API of our application and we will see why we have to implement such an infrastructure Naas.

Web 2-tier application topology and specification

OpenStack client installation

We add a Image Alpine

☐ > Alpine	Image	Active	Private	No	ISO	121.00 MB	Launch	▼
------------	-------	--------	---------	----	-----	-----------	--------	---

Then we create first API VM where we install nodejs

After we create an instance alpine-node and we had http/https rights in security group to be able to access the Internet and install curl package.

Furthermore we faced a DNS problem and it was impossible to install curl command

To solve the issue we remove the dns information in the subnet configuration of our private network. I spend two hours to solve this issue. However I am happy to met this trouble because the most interesting part is to know how to troubleshoot a network.

Once we installed curl and nodejs. We added a floating IP address and a http/https rules in security group to allow connexion from my computer located on public INSA network to my “CalculatorService” service located in private network as seen on figure below on the network schemes.

We changed in CalculatorService.js the IP address of services with VM private IP then we launch a calculation.

We tested that with CalculatorService and SumService

```
vlaurens@vlaurens-X555LN:~$ curl -d "(5+6)" -X POST http://192.168.37.10:80
result = 11
```

The command is ran on my computer on the INSA Network, so application is accessed from outside (INSA Network).

Picture of final network:

<input type="checkbox"/>	Instance Name	Image Name	IP Address	Flavor	Key Pair	Status	Availability Zone	Task	Power State	Time since created	Actions
<input type="checkbox"/>	DivService	alpine-node	10.0.2.11	nano	MuKey	Active	nova	None	Running	5 minutes	Create Snapshot ▼
<input type="checkbox"/>	MulService	alpine-node	10.0.2.24	nano	MuKey	Active	nova	None	Running	8 minutes	Create Snapshot ▼
<input type="checkbox"/>	SubService	alpine-node	10.0.2.13	nano	MuKey	Active	nova	None	Running	9 minutes	Create Snapshot ▼
<input type="checkbox"/>	CalculatorService	alpine-node	10.0.2.6 Floating IPs: 192.168.37.10	nano	MuKey	Active	nova	None	Running	13 heures, 8 minutes	Create Snapshot ▼
<input type="checkbox"/>	sumService	alpine-node	10.0.2.8	nano	MuKey	Active	nova	None	Running	3 jours, 21 heures	Create Snapshot ▼

Figure 36 : Creation of all instances for hosting different services of the application

Notice on figure 36 the floating IP on CalculatorService VM which as we describe our front-end service to be accessed from public network.

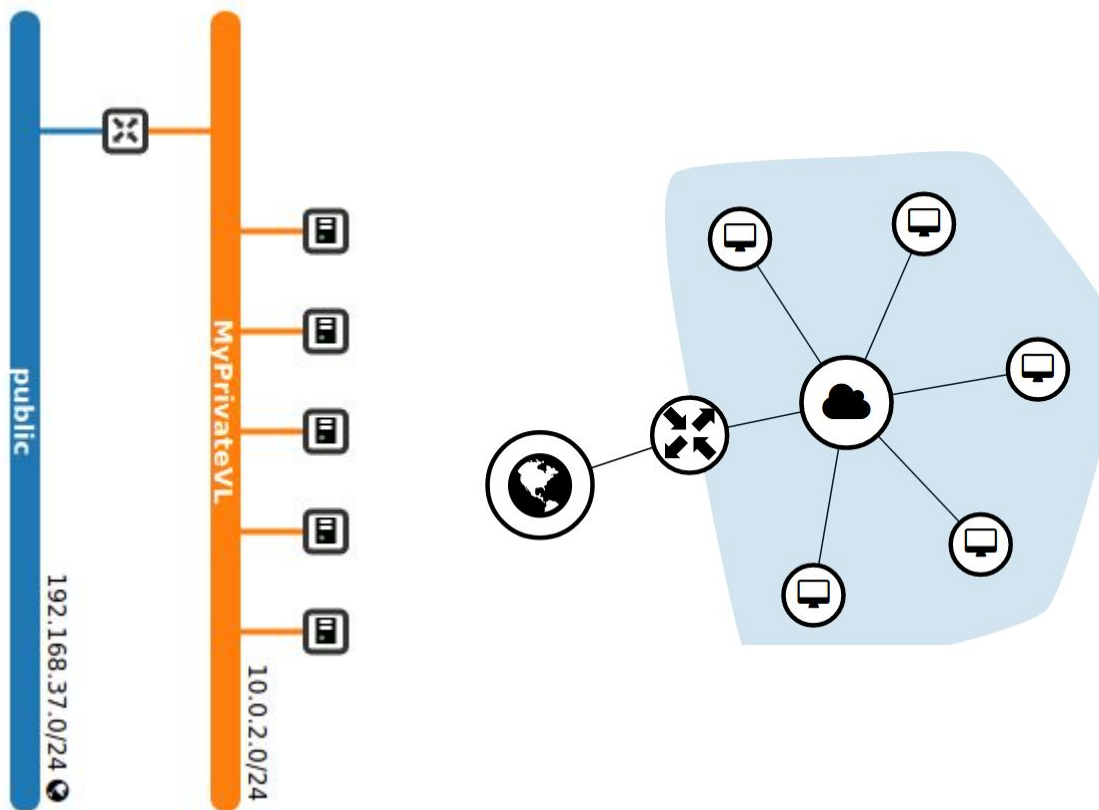


Figure 37 : Topology of network infrastructure on OpenStack

Test of hosting infrastructure and if application is accessible from public Network

We create a calculation which invoke all the services and we get the result as expected:

```
vlaurens@vlaurens-X555LN:~$ curl -d "(5*6)+3/4-2" -X POST http://192.168.37.10:80
result = 28.75
```

Figure 38 : Test to use application from public network using port 80

If we a service we can stop it, modify it and restart it without affecting the rest of the application. And we know when we added the new line "It is Working !! I have been added!"

```
Listening on port : 50004
New request :
A = 3
B = 4
A / B = 0.75
It is Working !! I have been added!
```

Figure 39 : Test to change the code while others services keep running

It is the advantage of micro-service different developer can update the application code part without affecting another part. It is not at all a monolithic development and so increase the efficiency and productivity. We can say that some services or area are isolated or protected from failures and we can compare that with Netflix which push some modifications on the application in some world area without affecting the other areas. It is a good point to go faster in test phase and test on a real scale.

NaaS (Network as-a-Service) provider,

Now we will isolate the front-end and services in two private interconnected networks. So we can protect services which have not to be open on the outside. Moreover if we look with a security point of view we will can protect our network with firewall by filtering all the protocol to accept only http request. We can do that to avoid any attacks.

We created the following infrastructure:

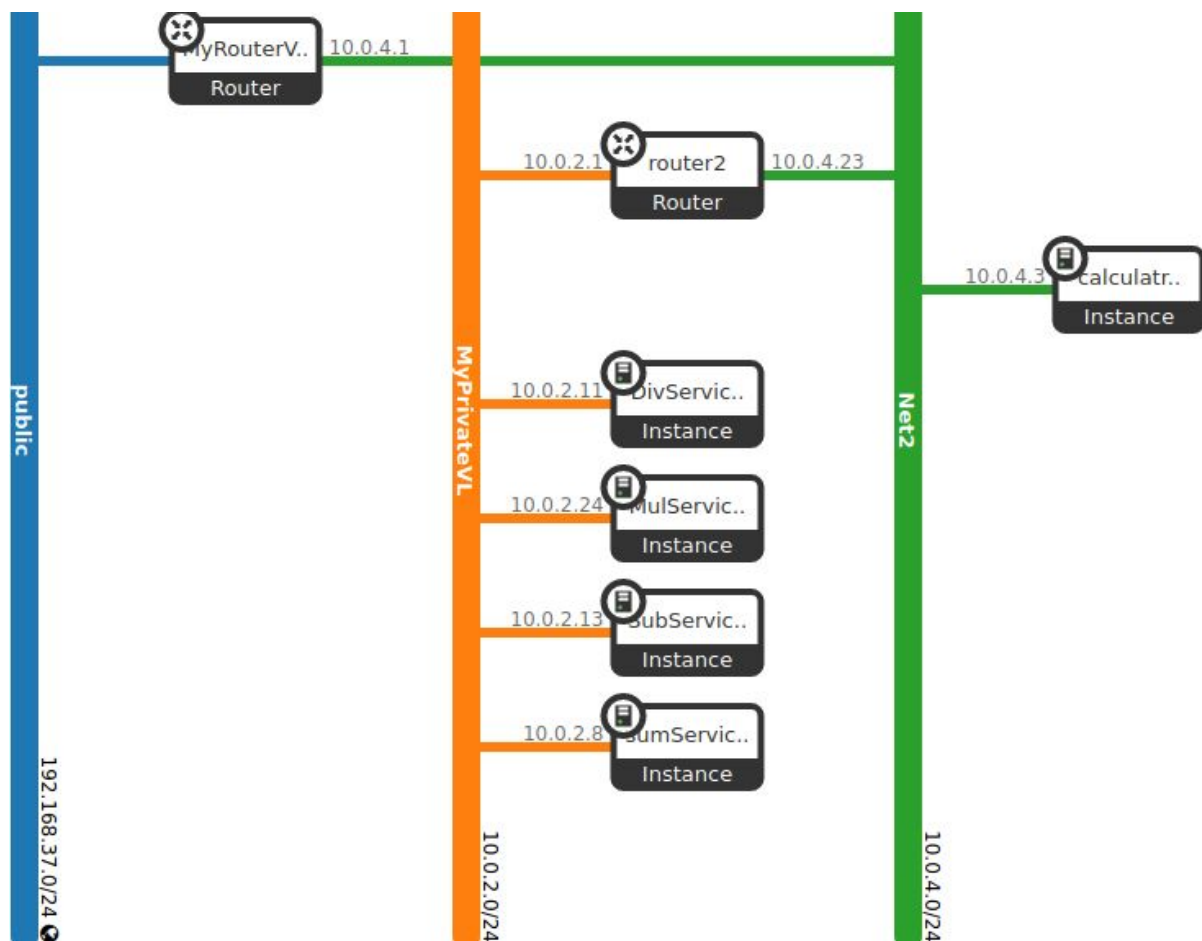


Figure 40: Granular Architecture

So on the Net2 we have the FrontService which can access other services by a routing implementation between both networks (Router).

Then we create a router which link Public network to Net2.

Here we can implement a proxy server between public network and Net2 because the link represent a critic access point open to the public network. So to detect attacks and secure system we need to install firewall and proxy to filter the incoming flow and limit the outcoming flow.

We don't forget to add a floating ip to the VM and security group to accept http.

		10.0.4.3								
<input type="checkbox"/>	calculatriceService	alpine-node	Floating IPs:	nano	MuKey	Active	nova	None	Running	16 minutes
		192.168.37.26								

Instance Security Groups	
Filter	<input type="text"/>
HTTP	<input type="checkbox"/>
SSH	<input type="checkbox"/>
default	<input type="checkbox"/>

We tested routing between two networks : FrontService and micro-services.

```
alpine-node:~# route
Kernel IP routing table
Destination      Gateway         Genmask         Flags Metric Ref    Use Iface
default          host-10-0-4-1.1 0.0.0.0         UG    202   0      0 eth0
10.0.2.0         host-10-0-4-23. 255.255.255.0   UG    0     0      0 eth0
10.0.4.0         *              255.255.255.0   U     0     0      0 eth0
alpine-node:~# ping 10.0.2.13
PING 10.0.2.13 (10.0.2.13): 56 data bytes
64 bytes from 10.0.2.13: seq=0 ttl=63 time=3.139 ms
64 bytes from 10.0.2.13: seq=1 ttl=63 time=1.257 ms
^C
--- 10.0.2.13 ping statistics ---
2 packets transmitted, 2 packets received, 0% packet loss
round-trip min/avg/max = 1.257/2.198/3.139 ms
```

The application run well! We invoke a service from FrontVM and we get the result.

```
alpine-node:~# curl -d "3 2" -X POST http://10.0.2.8:50001
5
```

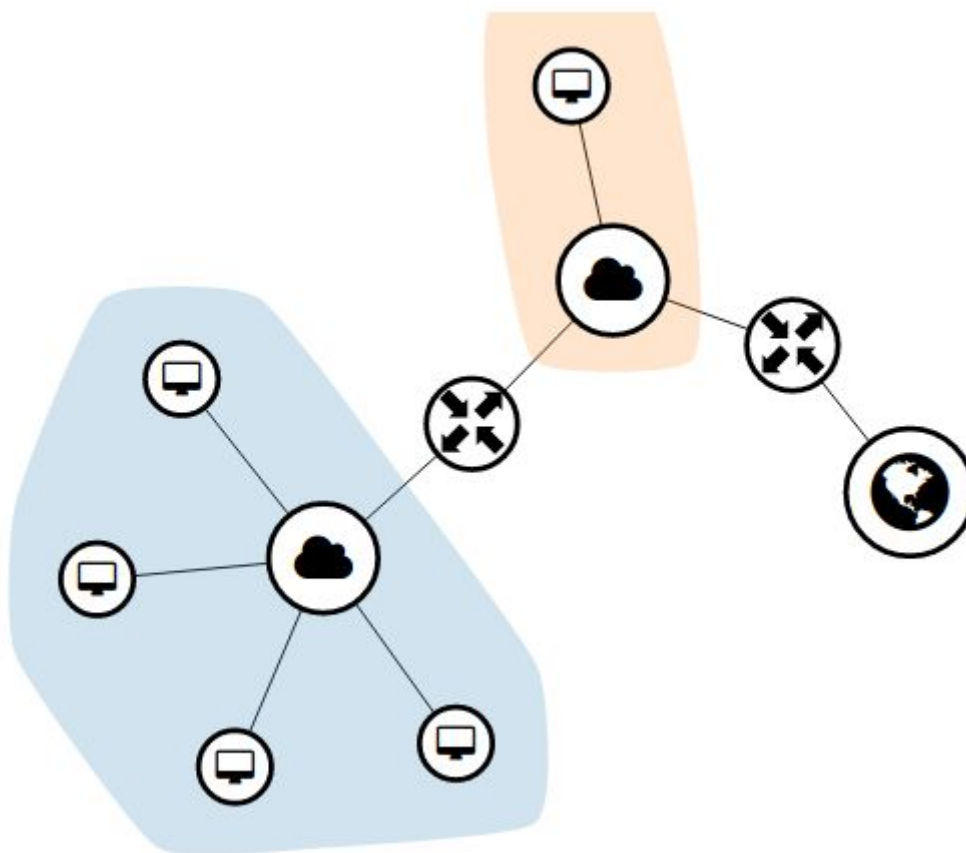



Figure 41 : Granular architecture topology schematic view

As we can observe the architecture offer a granularity level to the network which help to isolate some crucial application part. API (in blue) is directly hosting database (containing data) and by the consequence we need to protect it from public network. However, the front-end (in red) is a light application that can be developed and deploy or redeploy faster and which makes the link with public network so we put the VM in a different private network which is connected to the public network as describe on figure 41.

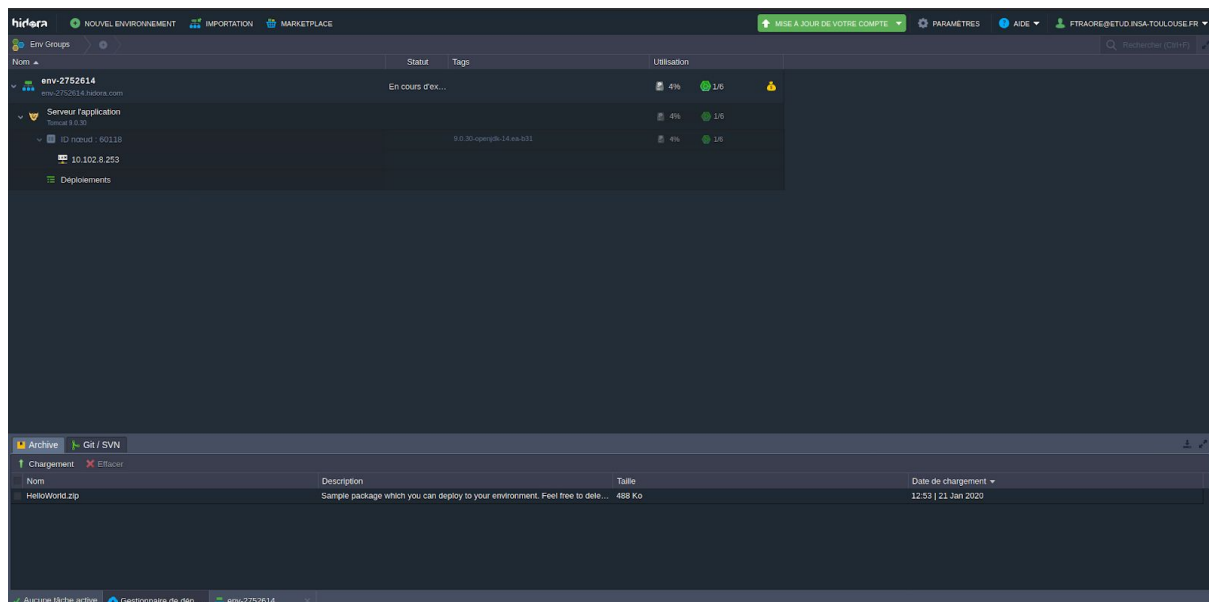
Conclusion

During those labs we learnt the different techniques and technologies to virtualize resources. We then learn to deploy an application on a cloud infrastructure (IaaS) and protect some part using a NaaS.

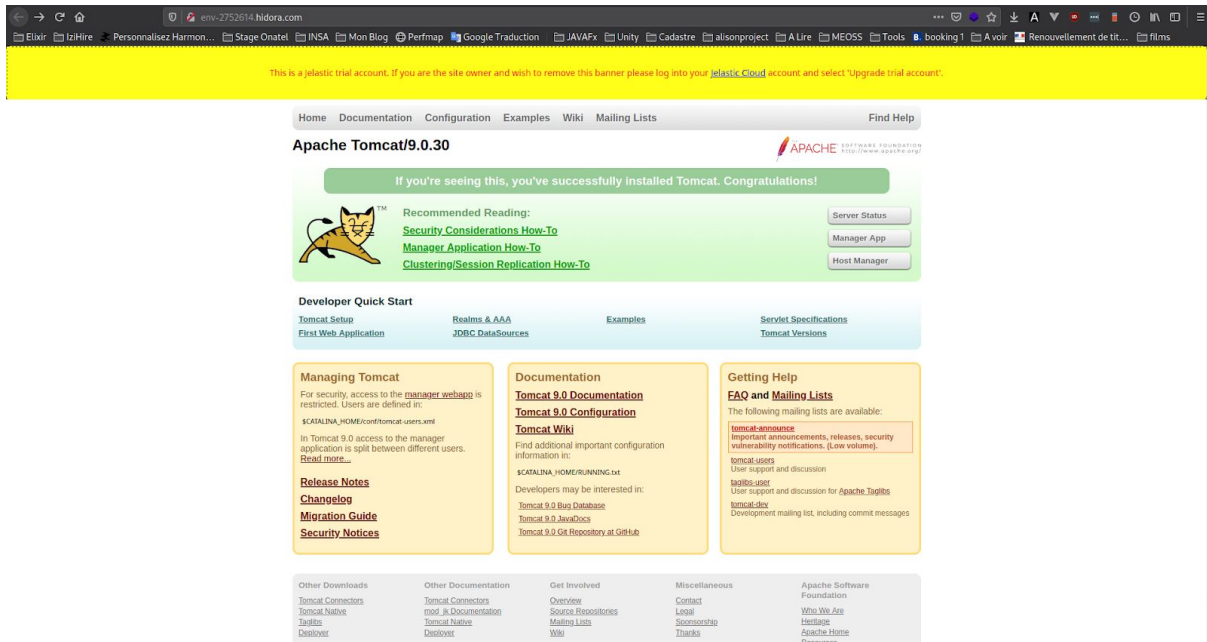
LABS 2 Provisioning End-user Application in Cloud Platforms

Theoretical part

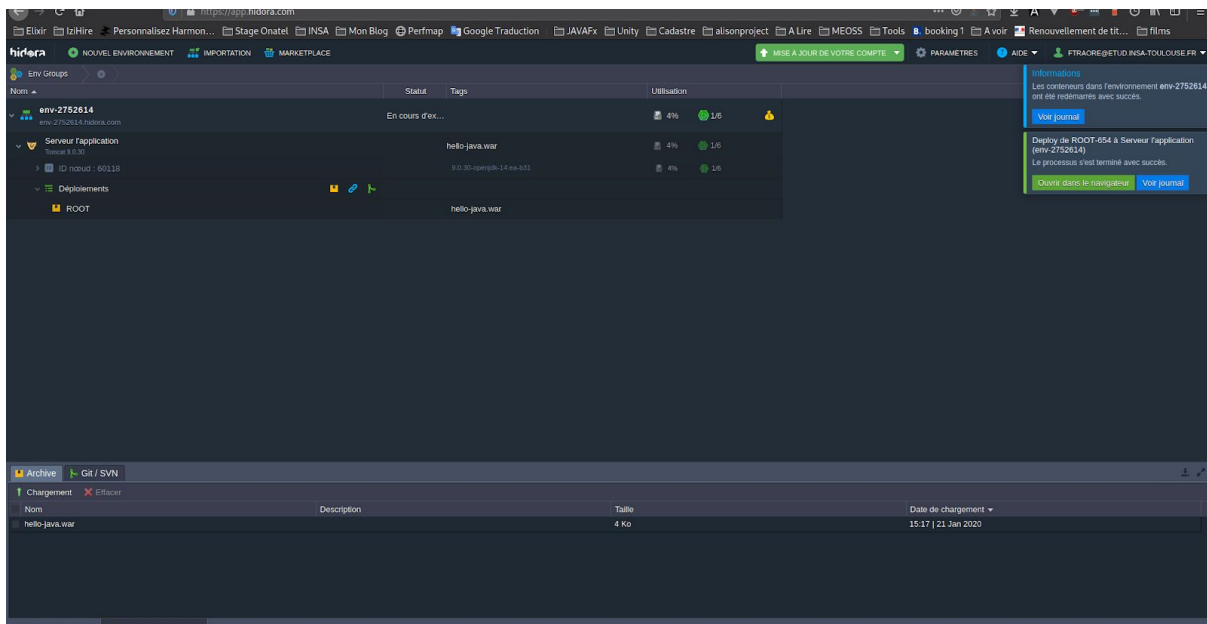
Practical part



Test



we deployed the application archive on the jelastic



After the deployment the application works fine!

Cloudfundary client

```
vlaurens@vlaurens-X555LN:~$ cf api api.run.pivotal.io
Définition du noeud final d'API api.run.pivotal.io...
OK

noeud final d'API : https://api.run.pivotal.io
version d'API : 2.144.0
Not logged in. Use 'cf login' or 'cf login --sso' to log in.
```

```
noeud final d'API : https://api.run.pivotal.io
version d'API : 2.144.0
Not logged in. Use 'cf login' or 'cf login --sso' to log in.
vlaurens@vlaurens-X555LN:~$ cf login
Noeud final d'API : https://api.run.pivotal.io

Email: vlaurens@etud.insa-toulouse.fr
Mot de passe:
Authentification...
OK

Targeted org vlaurens-insa-tp1

Targeted space development

Noeud final d'API : https://api.run.pivotal.io (API version: 2.144.0)
Utilisateur : vlaurens@etud.insa-toulouse.fr
Organisation : vlaurens-insa-tp1
Espace : development
```

Pivotal Web Services

Search apps, services, spaces, & orgs

press 17

vlaurens@etud.insa-toulouse.fr

Home <<

Marketplace

Recently Accessed Apps

No apps recently accessed

Orgs

CREATE ORG

Org Name	Quota	Spaces	Domains
vlaurens-insa-tp1	0% (0 Bytes / 2 GB)	1	0

Home

ORG
vlaurens-insa-tp1

MEMORY	AI COUNT	SI COUNT
0 %	0 %	0 %
0 / 2GB	0 / 32	0 / 10

Spaces CREATE NEW SPACE

Name	Apps	App Status	Services	Org Quota Usage
development	0	● 0 ■ 0 ▼ 0	0	0% (0 Bytes / 2 GB)

With command lines

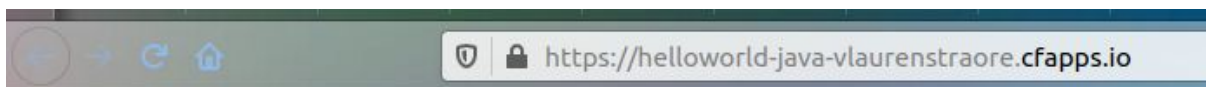
```
vlaurens@vlaurens-X555LN:~$ cf target -o vlaurens-insa-tp1 -s development
noeud final d'API : https://api.run.pivotal.io
version d'API : 2.144.0
utilisateur : vlaurens@etud.insa-toulouse.fr
organisation : vlaurens-insa-tp1
espace : development
```

Before to be able to run the application. We have pushed the application and deployment failed because they was not enough memory.

So in the manifest Cloud we increase the space capacity and we relaunch the application and the deployment succeeded and we could print the application on a browser.


```
état demandé : started
instances : 1/1
syntaxe : 1G x 1 instances
adresses URL : helloworld-java-vlaurensTraore.cfapps.io
dernier téléchargement : Tue Jan 21 13:24:14 UTC 2020
pile : cflinuxfs3
pack de construction : client-certificate-mapper=1.11.0_RELEASE container-security-provider=1.16.0_RELEASE java-buildpack=v4.26-offline-https://github.com/cloudfoundry/java-buildpack.git#e06e00b java-opts java-security jvmskill-agent=1.16.0_RELEASE open-jdk-like-jre=...
```

	état	depuis	unité centrale	mémoire
#0	en cours d'exécution	2020-01-21 02:25:20 PM	0.0%	41.9M sur
1G				116.9M sur 1G



Java is an island

SPACE RUNNING STOPPED CRASHED
development ● 1 ■ 1 ▼ 0


Apps					
Status	Name	Instances	Memory	Last Push	Route
■ STOPPED	hello-java-vlaurens	1	256 MB	20 min	no bound route
● RUNNING	hello-java-vlaurensTraore	1	1 GB	8 min	https://helloworld-java-... 



In the navigator on cloud foundry we can see that our application state changed to “RUNNING”.

In the TP it is really difficult to test performances of application because she process only a print. Otherwise we learn to scale up application memory space and we up to 2GB. We should implement an application which retrieve data from database or call database. We need a heavy application to see changes on performances.

Otherwise we can implement JMeter software to process performances tests on application by emulating numerous simultaneous request to the application.

SPACE RUNNING STOPPED CRASHED
development ● 1 ■ 1 ▼ 0



Apps					
Status	Name	Instances	Memory	Last Push	Route
■ STOPPED	hello-java-vlaurens	1	256 MB	20 min	no bound route
● RUNNING	hello-java-vlaurensTraore	1	1 GB	8 min	https://helloworld-java-... 

hello-java-vlaurensTraore
■


● RUNNING
Buildpack: client-certificate-map...

Bound Services

BIND SERVICE

NEW SERVICE

Service	Plan	Instance Name	Binding Name
 ElephantSQL	free - Tiny Turtle	DB	

We bind a service for storing data in a base.

Bibliography/Webography:

Yangui Samir (2019). « Labs subject », *Google Document*, https://docs.google.com/document/d/185KDFMGJmbGZthB_zldjw7tTw1-37xNskJHoGFIngKA/edit (2019/2020)
[1] : VMware website, available online :
<https://www.vmware.com/fr/solutions/virtualization.html> , visited 18 of january 2020