Potiers Léo
Seronie Paul

TP1: Theoretical Part

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

We can see that on the VM figure, multiple OS are running on the same physical server system thanks to the hypervisor of type 2, that enables virtualization.

The container enables the developer to use isolated and individual workspace easily and quickly. The main advantage of such a technique is that by writing a single file (ex:Dockerfile), the developer can generate a specific container with all the modules, dependencies and libraries he needs. In opposition, with a VM, the user needs a configuration file to install the needed tools, or he can do it himself after creating the VM.

Talking about storage, the VM enables the user to assign the amount of RAM memory depending on the developed application needs. With the VM, the CPU is shared between all OS, furthermore the response time will be shorter in the VM but longer in the main OS because it has dedicated pieces of CPU : that is why it is important to turn off any unused VM to free its CPU dedicated part. The container uses all the CPU, so it can be much faster if you do not use many containers, but if there are too many, the CPU will deal with demands using its own protocol (ex : FIFO, LILO etc…). In the end, the virtual cost of a VM is more important due to the CPU sharing.

Virtualization ensures isolation of the resources. With VM, the virtualization occurs at the hardware layer whereas for the container, it happens at the applicative level.

The container is more flexible thanks to its ability to create several applications with the same particular combination of modules and libraries. From a developer point of view, using containers allows to save time whereas using a VM implies downloading all the needed tools over again. However, a malware or a bad installation on a specific library can impact several applications while using containers. That is one of the reasons why VMs are prefered by administrators.

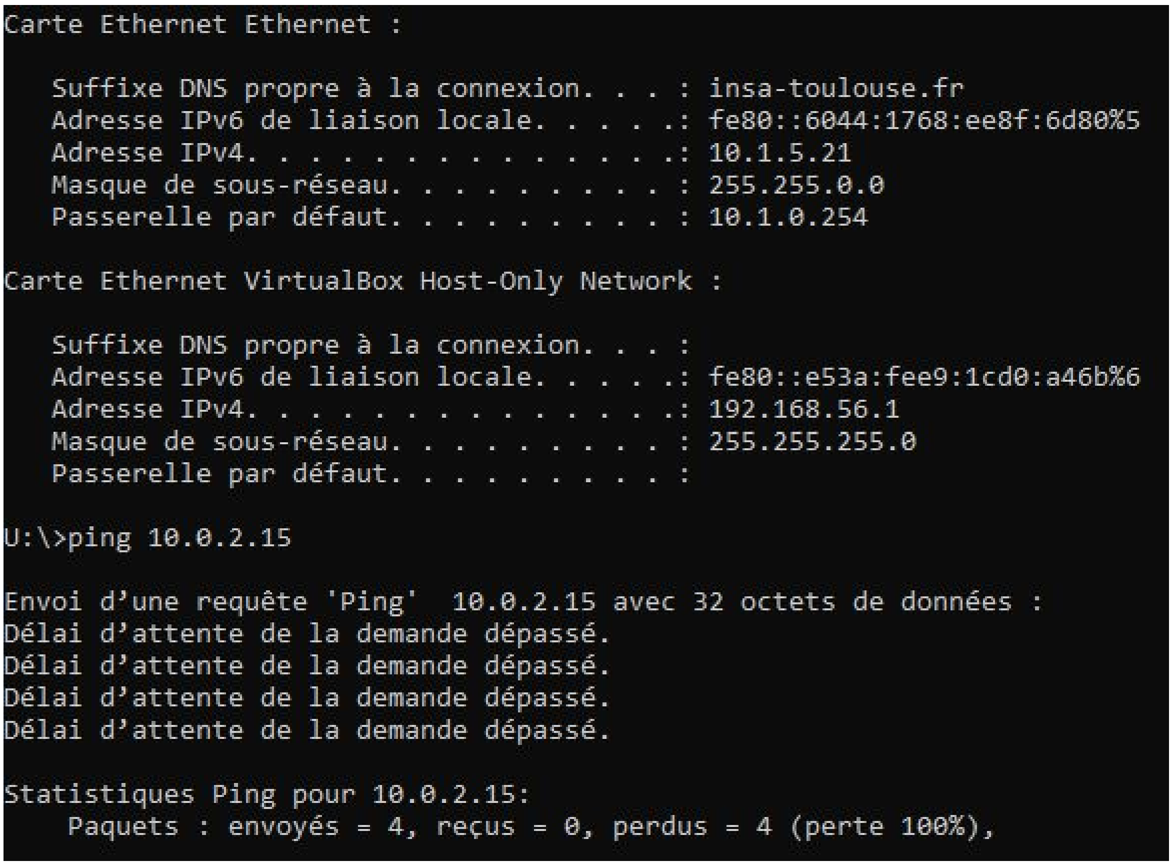
For DevOps and tooling for continuous integration, the containers are known for their flexibility because a program running in a container can be quickly deployed. The containers give stability and are more consistent : we can expect that system will run the same no matter where they are deployed. Containers enable more production, they allow faster deployment, patching, or scaling of applications. Using containers for DevOps is more affordable because they require fewer system resources than conventional machine environments or virtual machine hardware because they do not include images from the operating system.

TP2 : Practical part

our host machine’s IP : 10.1.5.21
our VM machine’s IP : 10.0.2.15

Test a
Commande : **ping 8.8.8.8 it WORKS !**

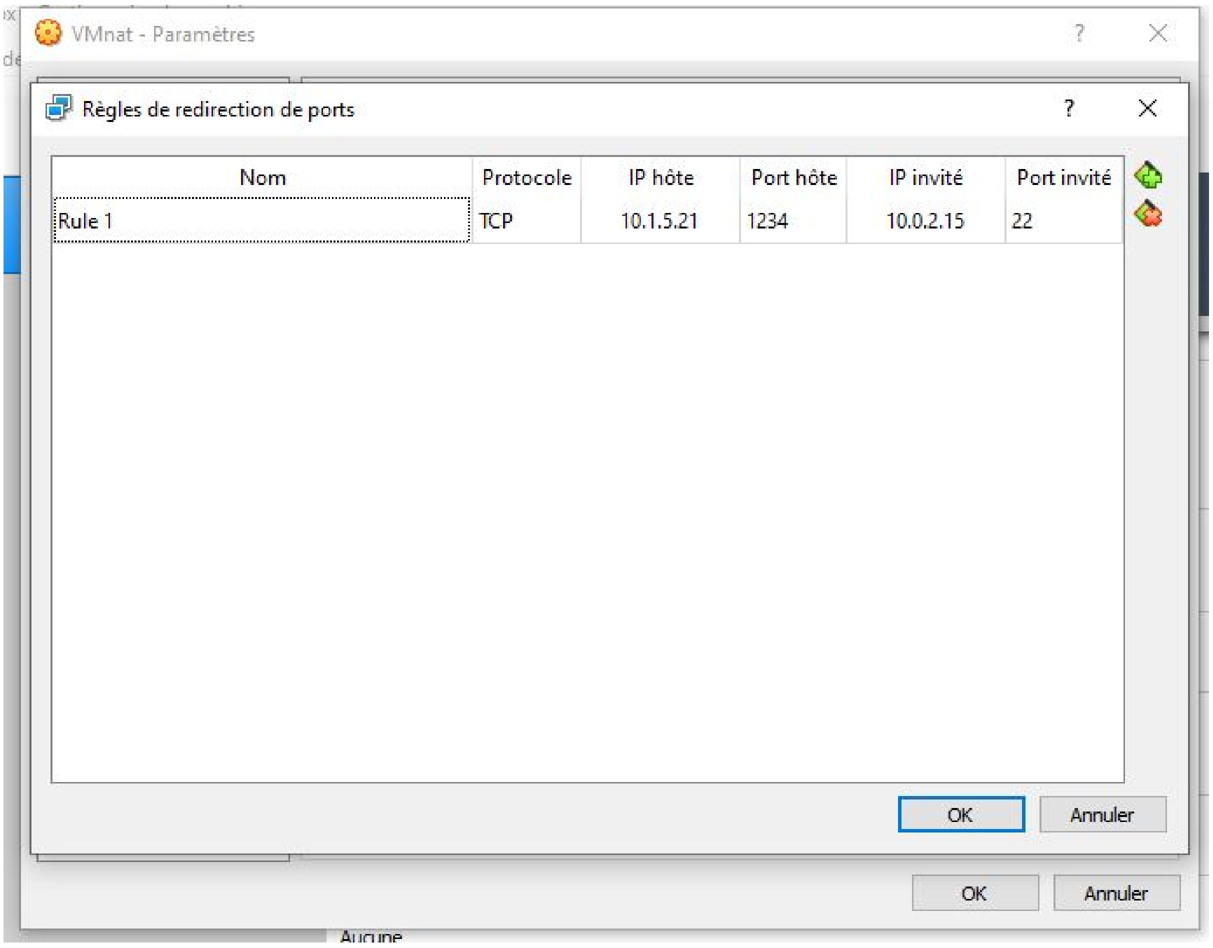
Ping from our host to our hosted VM :



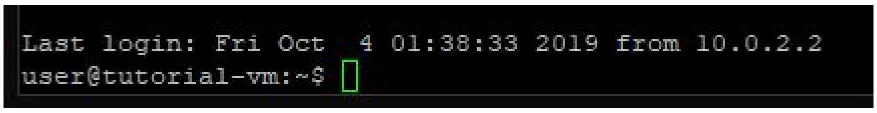
```
Carte Ethernet Ethernet :

   Suffixe DNS propre à la connexion. . . : insa-toulouse.fr
   Adresse IPv6 de liaison locale. . . . .: fe80::6044:1768:ee8f:6d80%5
   Adresse IPv4. . . . . . . . . . . . . .: 10.1.5.21
   Masque de sous-réseau. . . . . . . . . : 255.255.0.0
   Passerelle par défaut. . . . . . . . . : 10.1.0.254

Carte Ethernet VirtualBox Host-Only Network :

   Suffixe DNS propre à la connexion. . . :
   Adresse IPv6 de liaison locale. . . . .: fe80::e53a:fee9:1cd0:a46b%6
   Adresse IPv4. . . . . . . . . . . . . .: 192.168.56.1
   Masque de sous-réseau. . . . . . . . . : 255.255.255.0
   Passerelle par défaut. . . . . . . . . :

U:\>ping 10.0.2.15

Envoi d’une requête 'Ping'  10.0.2.15 avec 32 octets de données :
Délai d’attente de la demande dépassé.
Délai d’attente de la demande dépassé.
Délai d’attente de la demande dépassé.
Délai d’attente de la demande dépassé.

Statistiques Ping pour 10.0.2.15:
    Paquets : envoyés = 4, reçus = 0, perdus = 4 (perte 100%),
```

It does **not work !** Because each VM has the same IP address, instead of sending a ping to every VM, it does not send anything. 10.0.2.15 is not a routable address, it does not appear on the network.

But we can use the Port Forwarding, that redirects every ping/SSH connection/etc… directed to the VM's IP address, to the host machine's IP. Via the host IP, we can finally join the VM. After defining a rule for port forwarding, we used PuTTy to try a SSH connection to the VM :



Nom	Protocole	IP hôte	Port hôte	IP invité	Port invité
Rule 1	TCP	10.1.5.21	1234	10.0.2.15	22



```
Last login: Fri Oct  4 01:38:33 2019 from 10.0.2.2
user@tutorial-vm:~$
```

Docker :
Container's IP address : 172.17.0.2

Docker does the port forwarding itself. Even if every container has the same IP address, it is reachable from the VM. Of course, it is also possible to ping from the container, to the outside, for example internet.

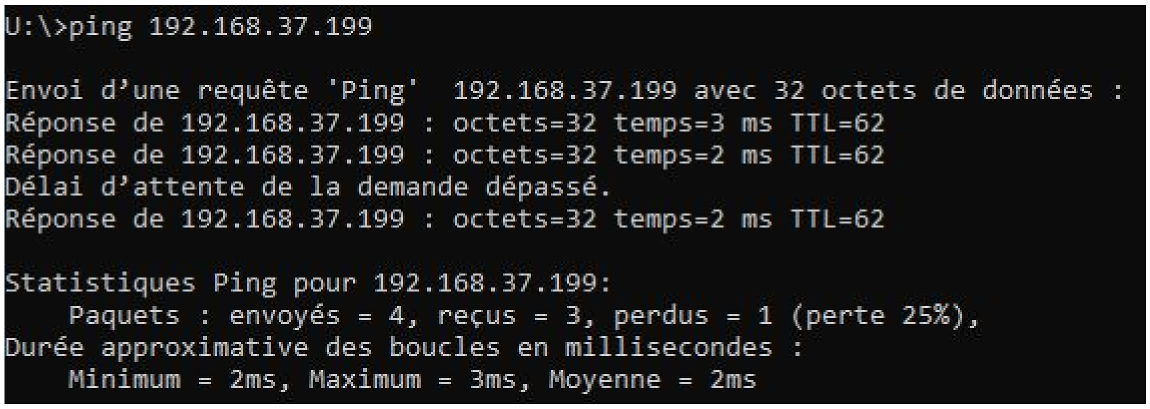
TP3 : Practical Part

First Part :

First the VM is created on OpenStack and then it's possible to start it. The keyboard has to be changed because it is configured as QWERTY.
Then, we added some rules to allow the traffic between virtual networks. This lets us use the Ping and SSH traffic.

Second Part :

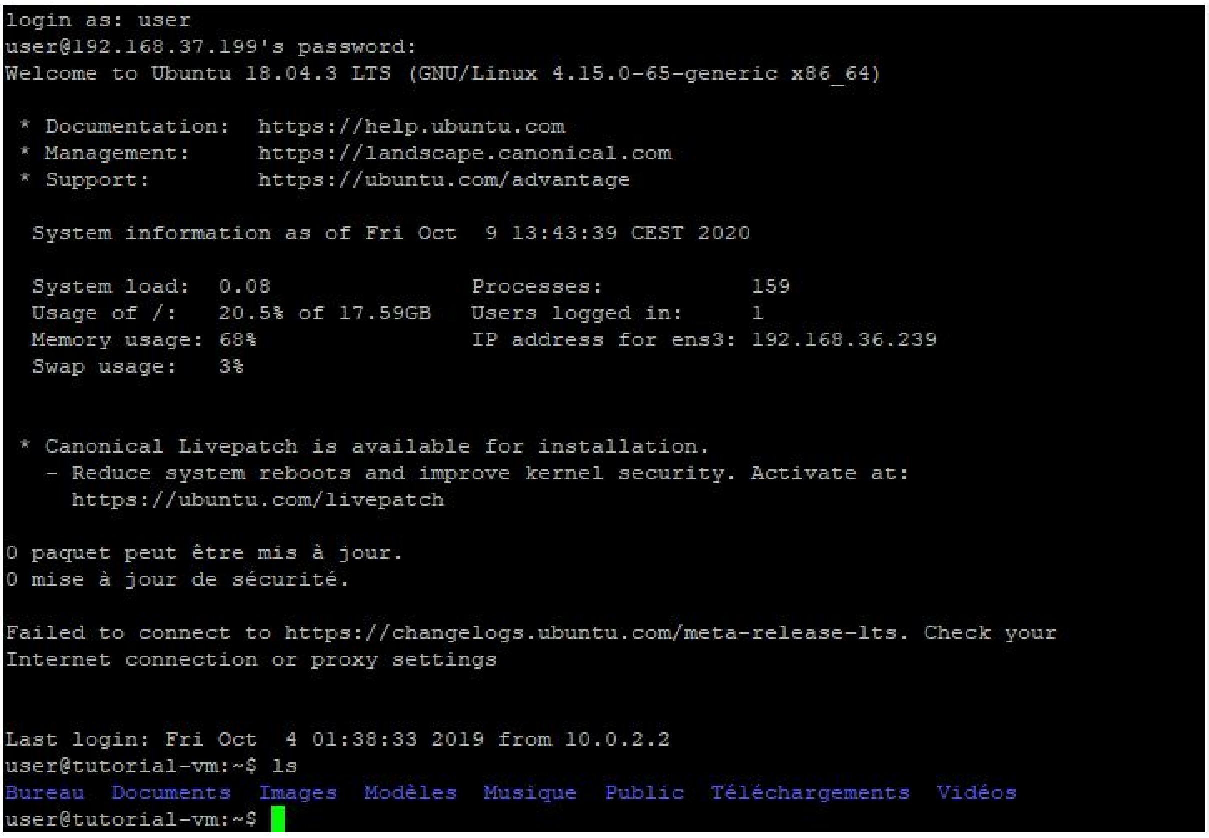
At first, the VM is isolated and not connected to any network. We must create a private one in order to put the VM on it. After doing that, we still can not ping from the VM to the host and vice versa. To solve the problem, a router is created to connect the public network and our private network. At this point, it is possible to reach the Internet from the VM but not the other way. To make the VM reachable from the outside, we implemented a floating IP associated with the VM. Then, we are able to reach the VM from the outside even using an SSH client (Putty).



```
U:\>ping 192.168.37.199

Envoi d'une requête 'Ping'  192.168.37.199 avec 32 octets de données :
Réponse de 192.168.37.199 : octets=32 temps=3 ms TTL=62
Réponse de 192.168.37.199 : octets=32 temps=2 ms TTL=62
Délai d'attente de la demande dépassé.
Réponse de 192.168.37.199 : octets=32 temps=2 ms TTL=62

Statistiques Ping pour 192.168.37.199:
    Paquets : envoyés = 4, reçus = 3, perdus = 1 (perte 25%),
Durée approximative des boucles en millisecondes :
    Minimum = 2ms, Maximum = 3ms, Moyenne = 2ms
```

Third Part :

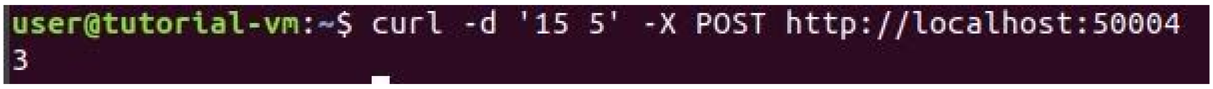
Resizing the VM while it is running creates a loss of connection with the server. You need to stop the VM and restart it in order to make it work. The problem we encountered with OpenStack is that you can't change the size of the VM from a bigger one to a smaller one. You always have to choose a bigger one in order to resize it.
The main difference between the original VM and the snapshot is that the snapshot contains all the space that was allocated to the original VM. It means that if you want to reuse the snapshot you made to create a new VM, you will need to use a lot of memory even if it was not really used in the first place.



```
login as: user
user@192.168.37.199'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 as of Fri Oct  9 13:43:39 CEST 2020

  System load:  0.08               Processes:           159
  Usage of /:   20.5% of 17.59GB   Users logged in:     1
  Memory usage: 68%                IP address for ens3: 192.168.36.239
  Swap usage:   3%


 * Canonical Livepatch is available for installation.
   - Reduce system reboots and improve kernel security. Activate at:
     https://ubuntu.com/livepatch

0 paquet peut être mis à jour.
0 mise à jour de sécurité.

Failed to connect to https://changelogs.ubuntu.com/meta-release-lts. Check your
Internet connection or proxy settings


Last login: Fri Oct  4 01:38:33 2019 from 10.0.2.2
user@tutorial-vm:~$ ls
Bureau  Documents  Images  Modèles  Musique  Public  Téléchargements  Vidéos
user@tutorial-vm:~$
```

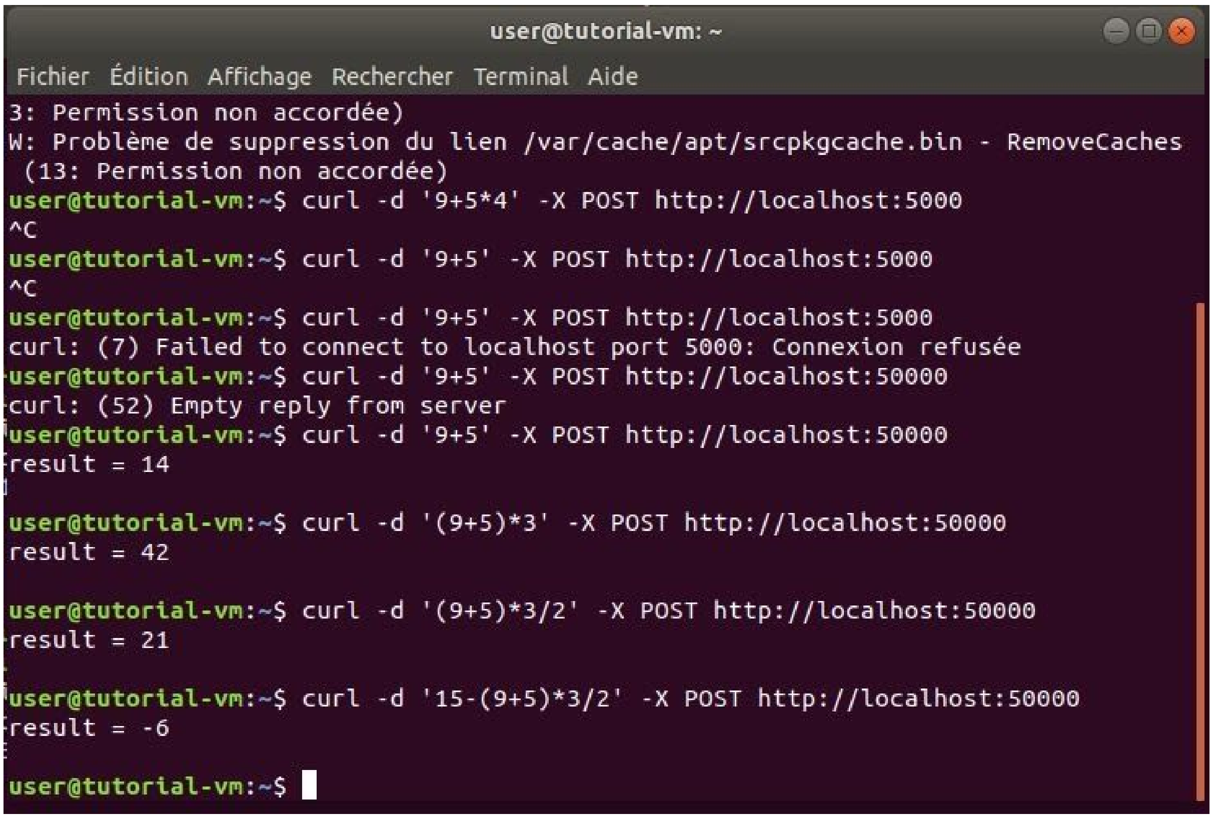
TP4: Practical Part

The first step is to configure the client on an Ubuntu VM with Oracle. We needed to use the rc file available on OpenStack.
After that, we created several VMs in order to realize the different operation (four in total) and one that centralizes everything. They are all implemented on the same private network. We installed everything needed on each one of them and ran all the different services. We tested all of them on a local scale.
Here is an example of local test for the division :



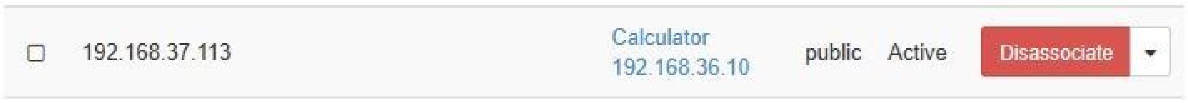
```
user@tutorial-vm:~$ curl -d '15 5' -X POST http://localhost:50004
3
```

Once we verified that they all worked correctly individually, we needed to see if the CalculatorService was able to interact with every web service. In order to do that, we had to modify the port in the CalculatorService.js document and to add the different addresses of all the VMs hosting the different operations. After a few tests, we verified that the whole calculator was working from the Calculator VM.



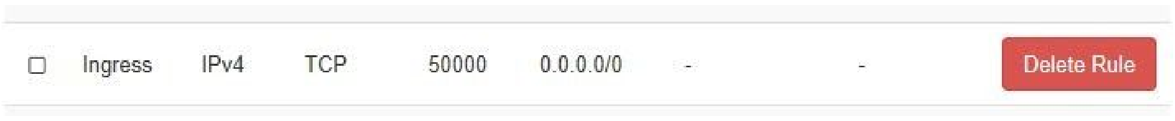
```
3: Permission non accordée)
W: Problème de suppression du lien /var/cache/apt/srcpkgcache.bin - RemoveCaches
 (13: Permission non accordée)
user@tutorial-vm:~$ curl -d '9+5*4' -X POST http://localhost:5000
^C
user@tutorial-vm:~$ curl -d '9+5' -X POST http://localhost:5000
^C
user@tutorial-vm:~$ curl -d '9+5' -X POST http://localhost:5000
curl: (7) Failed to connect to localhost port 5000: Connexion refusée
user@tutorial-vm:~$ curl -d '9+5' -X POST http://localhost:50000
curl: (52) Empty reply from server
user@tutorial-vm:~$ curl -d '9+5' -X POST http://localhost:50000
result = 14

user@tutorial-vm:~$ curl -d '(9+5)*3' -X POST http://localhost:50000
result = 42

user@tutorial-vm:~$ curl -d '(9+5)*3/2' -X POST http://localhost:50000
result = 21

user@tutorial-vm:~$ curl -d '15-(9+5)*3/2' -X POST http://localhost:50000
result = -6

user@tutorial-vm:~$
```

In order to make the Calculator Service work from the client, we had to associate to the Calculator VM a floating IP, because they were not on the same network:



☐	192.168.37.113	Calculator 192.168.36.10	public	Active	Disassociate ▾

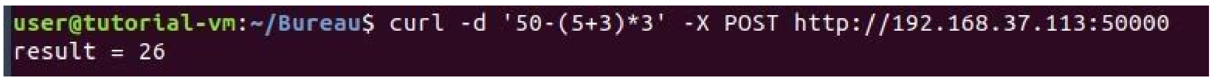
The last step was authorize the data traffic between the Client and the Calculator VM, to do so, we created a security group :



☐	Ingress	IPv4	TCP	50000	0.0.0.0/0	-	-

If the client would have been on the same private network, this rule would not have been needed because it only concerns the out of network traffic.

Finally, the Calculator Service was accessible and operative from the client VM :



```
user@tutorial-vm:~/Bureau$ curl -d '50-(5+3)*3' -X POST http://192.168.37.113:50000
result = 26
```