

Databases, queries and a load balancer

Paolo Testa
(20038424)

This tutorial is part of the education materials from the “Teaching Cloud Computing” website. For info, comment or support, please visit <http://tcc.uniupo.it>.

Introduction

The goal of this tutorial is to create two databases on two separate and distinct virtual machines.

Afterwards, a load balancer will be configured through which queries will be executed and distributed between the two databases. It will be demonstrated that, even if one of the two VMs is shut down, the queries will continue to function correctly thanks to the second VM still being active.

Furthermore, the tutorial will show how to create a load balancer from scratch using the terminal, including the configuration of the pool and the listener.

Basic assumptions:

It is assumed that two instances have already been created on Chameleon Cloud, each with an associated floating IP, and that the security group has also been configured.

Inside the instance

At this point, the connection is made to the first virtual machine.

```
paolo@DESKTOP-S3E91SI:~$ ssh -i ~/chiave/id_ed25519 cc@129.114.27.205
```

And proceed with the installation of MySQL inside the first virtual machine.

```
cc@testapaolo-db1:~$ sudo apt install
```

```
cc@testapaolo-db1:~$ sudo apt update
```

```
cc@testapaolo-db1:~$ sudo apt install mysql-server -y
```

Once MySQL is installed, access it by entering:

```
cc@testapaolo-db1:~$ sudo mysql
```

And create a database:

```
mysql> create database pagamenti;  
Query OK, 1 row affected (0.01 sec)
```

In my case, I decided to create a database to manage the payments that are made, but any database could be created to manage anything.

Next, access the previously created database:

```
mysql> use pagamenti;  
Database changed
```

And within it, create a table containing 2 columns (data and importo).

```
mysql> create table transazioni (  
-> data date,  
-> importo decimal (10,2)  
-> );  
Query OK, 0 rows affected (0.05 sec)
```

In this case too, the columns could have had different names or types, and there could have been more than two if another line had been added before the “;”.

Next, proceed with creating the user:

```
mysql> create user 'paolo'@'%' identified by 'Qwertyuiop12!?';  
Query OK, 0 rows affected (0.03 sec)
```

Grant the user Paolo all privileges on all databases and all tables (.); the with grant option allows the user to in turn grant privileges to other users.

On the other hand, flush privileges; reloads the privilege tables to make the recent changes effective.

```
mysql> grant all privileges on *.* to 'paolo'@'%' with grant option;  
Query OK, 0 rows affected (0.01 sec)  
  
mysql> flush privileges;  
Query OK, 0 rows affected (0.01 sec)
```

Then exit MySQL:

```
mysql> exit  
Bye
```

And open this file:

```
cc@testapaolo-db1:~$ sudo nano /etc/mysql/mysql.conf.d/mysqld.cnf
```

Inside the file, look for the line bind-address to allow MySQL to listen on all network interfaces, not just on localhost.

```
GNU nano 4.8 /etc/mysql/mysql.conf.d/mysqld.cnf  
#  
# Instead of skip-networking the default is now to listen only on  
# localhost which is more compatible and is not less secure.  
bind-address            = 127.0.0.1  
mysqlx-bind-address     = 127.0.0.1
```

Modify the bind-address line to 0.0.0.0.

```
GNU nano 4.8 /etc/mysql/mysql.conf.d/mysqld.cnf  
#  
# Instead of skip-networking the default is now to listen only on  
# localhost which is more compatible and is not less secure.  
bind-address            = 0.0.0.0  
mysqlx-bind-address     = 127.0.0.1
```

To save the changes to the file and exit, press Ctrl+O, Enter, and Ctrl+X.

Restart MySQL to apply the changes made.

```
cc@testapaolo-db1:~$ sudo systemctl restart mysql
```

And finally, exit the VM:

```
cc@testapaolo-db1:~$ exit
logout
Connection to 129.114.27.205 closed.
```

The exact same operations are repeated on the other VM that will be created.

Be careful to create the user identically in both databases, with the same name, same privileges, and the same password, because this way queries can later be made from the load balancer to both VMs.

Openstack

It is assumed that the OpenStack RC file has already been downloaded from Chameleon, and a password has already been created.

Then, connect to OpenStack and enter the personal password chosen earlier:

```
paolo@DESKTOP-S3E91SI:~$ cd "/mnt/c/Users/Utente/Desktop/secondo semestre/cloud/chiave"
paolo@DESKTOP-S3E91SI:/mnt/c/Users/Utente/Desktop/secondo semestre/cloud/chiave$ source CHI-251436-openrc.sh
(20038424@studenti.uniupo.it) Please enter your Chameleon CLI password:
```

Then proceed with installing the Octavia client: first, update the system, then proceed with the installation of Octavia, which is the OpenStack load balancing service that distributes traffic among multiple servers, ensuring high availability and scalability of applications.

```
paolo@DESKTOP-S3E91SI:/mnt/c/Users/Utente/Desktop/secondo semestre/cloud/chiave$ sudo apt update
paolo@DESKTOP-S3E91SI:/mnt/c/Users/Utente/Desktop/secondo semestre/cloud/chiave$ sudo apt install python3-octaviaclient
```

Create the load balancer

First, you need to get the ID of our subnet, and to do this, run the command:

```
paolo@DESKTOP-S3E91SI:/mnt/c/Users/Utente/Desktop/secondo semestre/cloud/chiave$ openstack subnet list
```

ID	Name	Network	Subnet
06c725cd-feef-4bf3-a56a-457583f00217	sharednet1-subnet	50073c73-5817-49c3-8e3a-69b8c357e158	10.56.0.0/22

Once the ID is obtained, the load balancer will be created (in my case, I called it "testapaolo_lb") and in the command, you insert the subnet ID:

```
paolo@DESKTOP-S3E91SI:/mnt/c/Users/Utente/Desktop/secondo semestre/cloud/chiave$ openstack loadbalancer create --name testapaolo_lb --vip-subnet-id 06c725cd-feef-4bf3-a56a-457583f00217
```

Field	Value
admin_state_up	True
availability_zone	None
created_at	2025-06-19T09:49:56
description	
flavor_id	None
id	e4682c62-5757-457f-9b3a-df327ee98628
listeners	
name	testapaolo_lb
operating_status	OFFLINE
pools	
project_id	d57a2227d43e4cc29fc00547b4fa3f2a
provider	amphora
provisioning_status	PENDING_CREATE
updated_at	None
vip_address	10.56.2.253
vip_network_id	50073c73-5817-49c3-8e3a-69b8c357e158
vip_port_id	d84ee0b9-4a5b-4c1b-9ce5-dac9a2185a95
vip_qos_policy_id	None
vip_subnet_id	06c725cd-feef-4bf3-a56a-457583f00217
vip_vnic_type	normal
tags	
additional_vips	[]

Once created, it is necessary to verify that it appears in the list of various load balancers, and that it is active and online:

```
paolo@DESKTOP-S3E91SI:/mnt/c/Users/Utente/Desktop/secondo semestre/cloud/chiave$ openstack loadbalancer list
```

id	name	project_id	vip_address	provisioning_status	operating_status	provider
e4682c62-5757-457f-9b3a-df327ee98628	testapaolo_lb	d57a2227d43e4cc29fc00547b4fa3f2a	10.56.2.253	ACTIVE	ONLINE	amphora

(for clarity, I have cut out the rest of the other LB list)

Once the load balancer is created and shows as active and online, proceed with creating the listener (called testapaolo-listener in my case, and it is associated with my load balancer, testapaolo_lb):

```
paolo@DESKTOP-S3E91SI:/mnt/c/Users/Utente/Desktop/secondo semestre/cloud/chiave$ openstack loadbalancer listener create --name testapaolo-listener --protocol TCP --protocol-port 3306 testapaolo_lb
```

Field	Value
admin_state_up	True
connection_limit	-1
created_at	2025-06-19T09:52:24
default_pool_id	None
default_tls_container_ref	None
description	
id	07571c90-9670-4022-a132-e856f99a586b
insert_headers	None
l7policies	
loadbalancers	e4682c62-5757-457f-9b3a-df327ee98628
name	testapaolo-listener
operating_status	OFFLINE
project_id	d57a2227d43e4cc29fc00547b4fa3f2a
protocol	TCP
protocol_port	3306
provisioning_status	PENDING_CREATE
sni_container_refs	[]
timeout_client_data	50000
timeout_member_connect	5000
timeout_member_data	50000
timeout_tcp_inspect	0
updated_at	None
client_ca_tls_container_ref	None
client_authentication	NONE
client_crl_container_ref	None
allowed_cidrs	None
tls_ciphers	None
tls_versions	None
alpn_protocols	None
tags	
hsts_max_age	
hsts_include_subdomains	
hsts_preload	

Then proceed with creating the pool (in my case called testapaolo-pool), which uses round_robin as the algorithm type, and this is also associated with my load balancer:

```
pao@DESKTOP-53E9151: /mnt/c/Users/Utente/Desktop/secondo semestre/cloud/chiave$ openstack loadbalancer pool create --name testapaolo-pool --lb-algorithm ROUND_ROBIN --listener testapaolo-listener --protocol TCP
```

Field	Value
admin_state_up	True
created_at	2025-06-19T09:54:02
description	
healthmonitor_id	
id	0f42f5d7-aceb-4dfe-87d8-1a6533a89171
lb_algorithm	ROUND_ROBIN
listeners	07571c90-9670-4822-a132-e856f99a586b
loadbalancers	e4682c62-5757-457f-9b3a-df327ee98628
members	
name	testapaolo-pool
operating_status	OFFLINE
project_id	d57a2227d43e4cc29fc00547b4fa3f2a
protocol	TCP
provisioning_status	PENDING_CREATE
session_persistence	None
updated_at	None
tls_container_ref	None
ca_tls_container_ref	None
crl_container_ref	None
tls_enabled	False
tls_ciphers	None
tls_versions	None
tags	
alpn_protocols	None

(Round robin is a load balancing algorithm used in load balancers, that distributes requests cyclically and sequentially among the available servers).

Once the pool is created, create a health-monitor associated with my pool, so that it can provide information about the status of the associated members, whether they are active and working or if a failure has occurred.

```
pao@DESKTOP-53E9151: /mnt/c/Users/Utente/Desktop/secondo semestre/cloud/chiave$ openstack loadbalancer healthmonitor create --delay 5 --timeout 3 --max-retries 3 --type TCP testapaolo-pool
```

Field	Value
project_id	d57a2227d43e4cc29fc00547b4fa3f2a
name	
admin_state_up	True
pools	0f42f5d7-aceb-4dfe-87d8-1a6533a89171
created_at	2025-06-19T09:56:06
provisioning_status	PENDING_CREATE
updated_at	None
delay	5
expected_codes	None
max_retries	3
http_method	None
timeout	3
max_retries_down	3
url_path	None
type	TCP
id	d510ff3c-6c53-47d6-a361-558943de2f4f
operating_status	OFFLINE
http_version	None
domain_name	None
tags	

Next, proceed with creating the members in my pool. In this case, you need to specify the subnet ID, which was found earlier through openstack subnet list, use the public IP of our VM as the address, and finally indicate the pool to which it should be associated:

```
pao@DESKTOP-53E9151: /mnt/c/Users/Utente/Desktop/secondo semestre/cloud/chiave$ openstack loadbalancer member create --subnet-id 06c725cd-feef-4bf3-a56a-457583f00217 --address 129.114.27.205 --protocol-port 3306 testapaolo-pool
```

Field	Value
address	129.114.27.205
admin_state_up	True
created_at	2025-06-19T09:58:43
id	ee8a2774-c087-4bd4-8ba2-1fefbe0123a8
name	
operating_status	OFFLINE
project_id	d57a2227d43e4cc29fc00547b4fa3f2a
protocol_port	3306
provisioning_status	PENDING_CREATE
subnet_id	06c725cd-feef-4bf3-a56a-457583f00217
updated_at	None
weight	1
monitor_port	None
monitor_address	None
backup	False
tags	

The exact same operation is performed for the other VM to add it as a member.

Verify that both members are correctly added to the pool so they can be used, and thanks to the monitor created earlier, it can be checked that both are online.

```
paolo@DESKTOP-S3E91SI:/mnt/c/Users/Utente/Desktop/secondo semestre/cloud/chiave$ openstack loadbalancer member list testapaolo-pool
```

id	name	project_id	provisioning_status	address	protocol_port	operating_status	weight
ee8a2774-c087-4bd4-8ba2-1fefbe0123a8		d57a2227d43e4cc29fc00547b4fa3f2a	ACTIVE	129.114.27.205	3306	ONLINE	1
2ebd2492-86a4-42e6-bce7-e9c30fc8f917		d57a2227d43e4cc29fc00547b4fa3f2a	ACTIVE	129.114.26.146	3306	ONLINE	1

Proceed with installing mysql-client so that you can access the databases inside the VMs through the load balancer:

```
paolo@DESKTOP-S3E91SI:/mnt/c/Users/Utente/Desktop/secondo semestre/cloud/chiave$ sudo apt update
paolo@DESKTOP-S3E91SI:/mnt/c/Users/Utente/Desktop/secondo semestre/cloud/chiave$ sudo apt install mysql-client
```

Now you need to associate the security group to the load balancer, and to do this, first get the ID of our security group:

```
paolo@DESKTOP-S3E91SI:/mnt/c/Users/Utente/Desktop/secondo semestre/cloud/chiave$ openstack security group list
```

ID	Name	Description	Project	Tags
ea710afb-a6c0-4ae0-bf9e-1a38a40d4cdb	paolot		d57a2227d43e4cc29fc00547b4fa3f2a	[]

Create the rule to add to the security group if it is not already present:

```
paolo@DESKTOP-S3E91SI:/mnt/c/Users/Utente/Desktop/secondo semestre/cloud/chiave$ openstack security group rule create --proto tcp --dst-port 3306 --ingress --remote-ip 0.0.0.0/0 ea710afb-a6c0-4ae0-bf9e-1a38a40d4cdb
```

Field	Value
created_at	2025-06-19T10:36:43Z
description	ingress
direction	
ether_type	IPv4
id	371c265d-ae6b-4926-bd90-cff769117ba7
name	None
normalized_cidr	0.0.0.0/0
port_range_max	3306
port_range_min	3306
project_id	d57a2227d43e4cc29fc00547b4fa3f2a
protocol	tcp
remote_address_group_id	None
remote_group_id	None
remote_ip_prefix	0.0.0.0/0
revision_number	0
security_group_id	ea710afb-a6c0-4ae0-bf9e-1a38a40d4cdb
tags	[]
updated_at	2025-06-19T10:36:43Z

Check which security group is associated with the load balancer's port (the ID is obtained by looking at the "vip_port_id" line in the output when the load balancer was created)

vip_port_id	d84ee0b9-4a5b-4c1b-9ce5-dac9a2185a95
-------------	--------------------------------------

```

paolo@DESKTOP-S3E91SI:/mnt/c/Users/Utente/Desktop/secondo semestre/cloud/chiave$ openstack port show d84ee0b9-4a5b-4c1b-9ce5-dac9a2185a95
+-----+-----+
| Field | Value |
+-----+-----+
| admin_state_up | DOWN |
| allowed_address_pairs | None |
| binding_host_id | None |
| binding_profile | None |
| binding_vif_details | None |
| binding_vif_type | None |
| binding_vnic_type | normal |
| created_at | 2025-06-19T09:49:56Z |
| data_plane_status | None |
| description | None |
| device_id | lb-e4682c62-5757-457f-9b3a-df327ee98628 |
| device_owner | Octavia |
| device_profile | None |
| dns_assignment | None |
| dns_domain | None |
| dns_name | None |
| extra_dhcp_opts | None |
| fixed_ips | ip_address='10.56.2.253', subnet_id='06c725cd-feef-4bf3-a56a-457583f00217' |
| hardware_offload_type | None |
| hints | None |
| id | d84ee0b9-4a5b-4c1b-9ce5-dac9a2185a95 |
| ip_allocation | None |
| mac_address | fa:16:3e:70:2b:4b |
| name | octavia-lb-e4682c62-5757-457f-9b3a-df327ee98628 |
| network_id | 50073c73-5817-49c3-8e3a-69b8c357e158 |
| numa_affinity_policy | None |
| port_security_enabled | True |
| project_id | d57a2227d43e4cc29fc00547b4fa3f2a |
| propagate_uplink_status | None |
| resource_request | None |
| revision_number | 2 |
| qos_network_policy_id | None |
| qos_policy_id | None |
| security_group_ids | f8f85e0f-c3b8-41a2-a420-9c6273df02cf |
| status | DOWN |
| tags | None |
| trunk_details | None |
| updated_at | 2025-06-19T09:49:57Z |
+-----+-----+

```

Almost certainly, a different security group will be associated, so proceed to remove the current security group and insert the security group of our interest:

```

paolo@DESKTOP-S3E91SI:/mnt/c/Users/Utente/Desktop/secondo semestre/cloud/chiave$ openstack port set --no-security-group d84ee0b9-4a5b-4c1b-9ce5-dac9a2185a95
paolo@DESKTOP-S3E91SI:/mnt/c/Users/Utente/Desktop/secondo semestre/cloud/chiave$ openstack port set --security-group ea710afb-a6c0-4ae0-bf9e-1a38a40d4cdb d84ee0b9-4a5b-4c1b-9ce5-dac9a2185a95

```

(when adding the security group, you need to enter both the security group ID and the load balancer port ID).

Now if you check, the security group of our interest will appear:

```
paolo@DESKTOP-S3E91SI:/mnt/c/Users/Utente/Desktop/secondo semestre/cloud/chiave$ openstack port show d84ee0b9-4a5b-4c1b-9ce5-dac9a2185a95
```

Field	Value
admin_state_up	DOWN
allowed_address_pairs	
binding_host_id	None
binding_profile	None
binding_vif_details	None
binding_vif_type	None
binding_vnic_type	normal
created_at	2025-06-19T09:49:56Z
data_plane_status	None
description	
device_id	lb-e4682c62-5757-457f-9b3a-df327ee98628
device_owner	Octavia
device_profile	None
dns_assignment	None
dns_domain	None
dns_name	None
extra_dhcp_opts	
fixed_ips	ip_address='10.56.2.253', subnet_id='06c725cd-feef-4bf3-a56a-457583f00217'
hardware_offload_type	None
hints	
id	d84ee0b9-4a5b-4c1b-9ce5-dac9a2185a95
ip_allocation	None
mac_address	fa:16:3e:70:2b:4b
name	octavia-lb-e4682c62-5757-457f-9b3a-df327ee98628
network_id	50073c73-5817-49c3-8e3a-69b8c357e158
numa_affinity_policy	None
port_security_enabled	True
project_id	d57a2227d43e4cc29fc00547b4fa3f2a
propagate_uplink_status	None
resource_request	None
revision_number	4
qos_network_policy_id	None
qos_policy_id	None
security_group_ids	ea710afb-a6c0-4ae0-bf9e-1a38a40d4cdb
status	DOWN
tags	
trunk_details	None
updated_at	2025-06-19T11:00:11Z

Next, proceed with creating a public floating IP:

```
paolo@DESKTOP-S3E91SI:/mnt/c/Users/Utente/Desktop/secondo semestre/cloud/chiave$ openstack floating ip create public
```

Field	Value
created_at	2025-06-19T11:01:12Z
description	
dns_domain	None
dns_name	None
fixed_ip_address	None
floating_ip_address	129.114.24.229
floating_network_id	69adad42-e10e-4e34-ab68-62cbe7fc23b1
id	26967692-1087-4beb-8914-a9467d443b04
name	129.114.24.229
port_details	None
port_id	None
project_id	d57a2227d43e4cc29fc00547b4fa3f2a
qos_policy_id	None
revision_number	0
router_id	None
status	DOWN
subnet_id	None
tags	[]
updated_at	2025-06-19T11:01:12Z

And once created, associate it with the load balancer:

```
paolo@DESKTOP-S3E91SI:/mnt/c/Users/Utente/Desktop/secondo semestre/cloud/chiave$ openstack floating ip set --port d84ee0b9-4a5b-4c1b-9ce5-dac9a2185a95 129.114.24.229
```

Now everything is set to connect to MySQL and run queries through the load balancer that will be distributed between the two databases.

Execute queries through the load balancer

Connect to the load balancer's public IP, and entering the user data created previously ('paolo' with password 'Qwertyuiop12!?'), will allow making queries.

```
paolo@DESKTOP-S3E91SI:/mnt/c/Users/Utente/Desktop/secondo semestre/cloud/chiave$ mysql -u paolo -p -h 129.114.24.229 -P 3306
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
```

Enter the "pagamenti" database:

```
mysql> use pagamenti;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A
```

And proceed with making 2 queries:

```
mysql> insert into transazioni (data, importo) values ('2024-06-14', 32.20);
Query OK, 1 row affected (0.15 sec)

mysql> insert into transazioni (data, importo) values ('2022-06-14', 73.20);
Query OK, 1 row affected (0.15 sec)
```

Once the queries are done, exit MySQL inside the load balancer, and connect to one of the two VMs:

```
paolo@DESKTOP-S3E91SI:/mnt/c/Users/Utente/Desktop/secondo semestre/cloud/chiave$ ssh -i ~/chiave/id_ed25519 cc@129.114.26.146
```

Entering the "pagamenti" database and selecting the table, it is noted that the two queries were successful ("sudo mysql" to enter mysql).

```
mysql> use pagamenti;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> select * from transazioni;
+-----+-----+
| data      | importo |
+-----+-----+
| 2024-06-14 | 32.20   |
| 2022-06-14 | 73.20   |
+-----+-----+
2 rows in set (0.00 sec)

mysql> exit
Bye
```

Now shut down this VM:

```
cc@testapalo-db2:~$ sudo shutdown now

Broadcast message from root@testapalo-db2 on pts/1 (Thu 2025-06-19 11:10:05 UTC):

The system will power off now!
```

Access the load balancer again through the public IP:

```
pao1o@DESKTOP-S3E91SI:/mnt/c/Users/Utente/Desktop/secondo semestre/cloud/chiave$ mysql -u pao1o -p -h 129.114.24.229 -P 3306
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
```

Enter the “pagamenti” database again and proceed with making other queries, and note that no error messages appear despite one of the two VMs being shut down, because the other is still running:

```
mysql> use pagamenti;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> insert into transazioni (data, importo) values ('2005-06-14', 10.20);
Query OK, 1 row affected (0.16 sec)

mysql> insert into transazioni (data, importo) values ('2042-06-14', 26849.20);
Query OK, 1 row affected (0.15 sec)

mysql> exit
Bye
```

Finally, connect to the other VM:

```
pao1o@DESKTOP-S3E91SI:/mnt/c/Users/Utente/Desktop/secondo semestre/cloud/chiave$ ssh -i ~/chiave/id_ed25519 cc@129.114.27.205
Welcome to Ubuntu 24.04.2 LTS (GNU/Linux 6.8.0-59-generic x86_64)
```

```
cc@testapao1o-db1:~$ sudo mysql
Welcome to the MySQL monitor.  Commands end with ; or \g.
```

Entering the payments database and selecting the table, it is noted that the last two queries were successful.

```
mysql> use pagamenti;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> select * from transazioni;
+-----+-----+
| data      | importo |
+-----+-----+
| 2005-06-14 |    10.20 |
| 2042-06-14 | 26849.20 |
+-----+-----+
2 rows in set (0.00 sec)

mysql> exit
Bye
```