Portfolio-exam-2-Cloud

OBS! Have to use HioA VPN solution to get access with openstack from outside the school's network: https://ansatt.oslomet.no/vpn-koble-filserver

This is the second portfolio exam which must be done and delivered in a group. This contributes 50% for your final grade. In this work, you will setup a cloud-based application architecture using LEMP stack in OpenStack (ALTO). It has a load balancer, three web servers, a database proxy and three database servers as shown in the figure below.

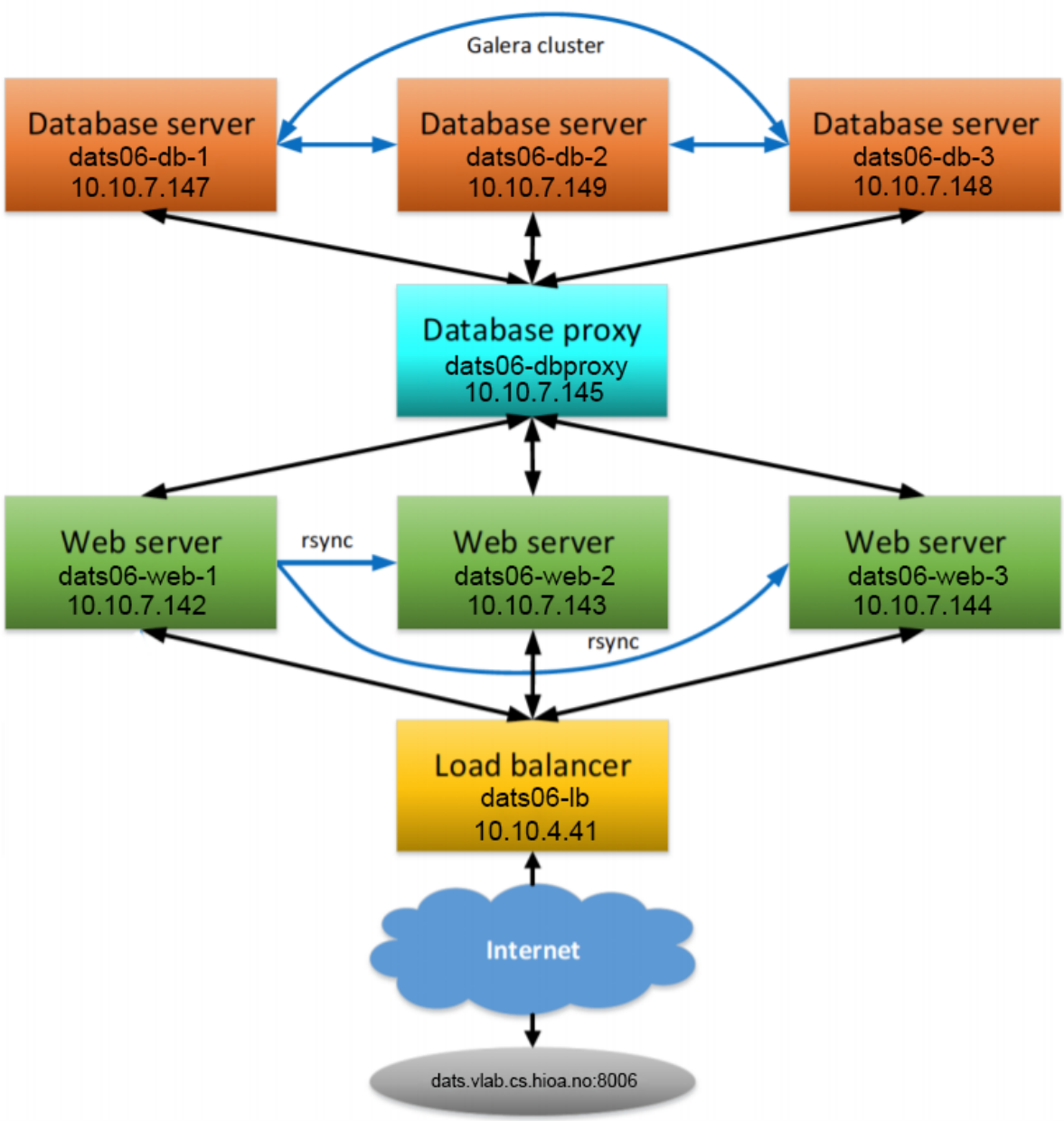
The VM created in Lab-C1 (named initially as datsXX-m1 and later changed to datXX-lb) whose IP was provided during the approval of the lab must be used as the load balancer. Reminding again not to terminate this VM! If the VM is somehow terminated, the new IP may not get configured again on time, and you might risk not being unable to complete the assignment on time. Also note that if in case ALTO crashes, all your VMs may disappear (from previous experience). Therefore, it is advised to keep notes of all the required information, steps followed, scripts etc. so that you will be able to redo everything quickly in case such a situation arises.

Follow the naming conventions for the virtual machines (VMs) as shown in the figure, whereyour two-digit group number should be used in place of XX. Load balancer should be of **m1.1GB** flavor and all other VMs should be of **m1.512MB4GB** flavor. Use the following OS and software in the setup. • OS: Ubuntu 16.04 • Web server: Nginx • Load balancer: HAProxy • Database server: MariaDB v10.2 • Server-side programming: PHP v7.x • Database proxy: MariaDB MaxScale 2.2

The whole work is divided into different tasks listed below, which include implementing in ALTO cloud and providing etails as asked in the submitted report. The report should provide properly labelled or captioned diagrams, configuration details and screenshots as asked in these tasks.

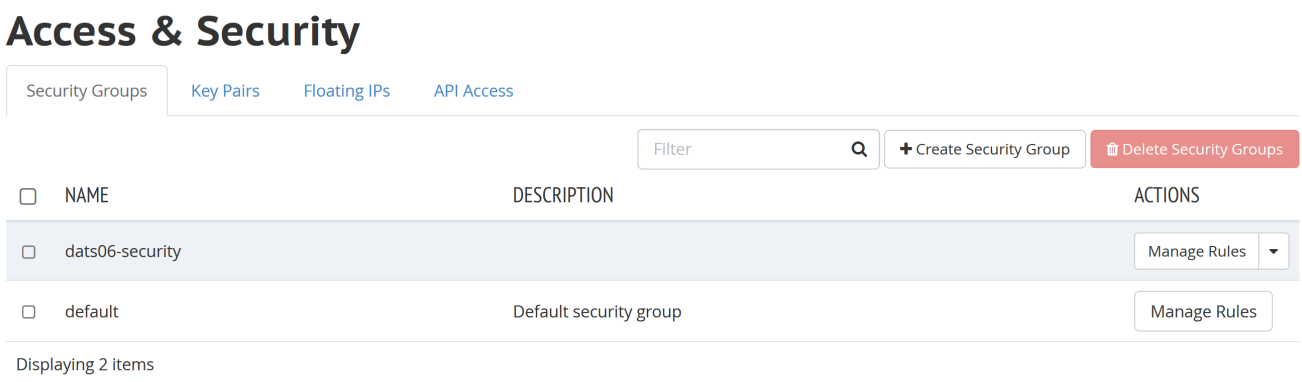
1. **VM setup**: This task consists of creating a ssh key (datsXX-key) and a security group (datsXXsecurity), creating VMs with desired flavors, doing minimal required common configurations in VMs such as **naming of hosts**, and setup **locale to Norwegian**. From the security point of view, only the required outside access (such as ssh, web, etc.) to the VMs must be given. This means the required ports only should be opened in the security group. The report should provide the followings:

- An architecture diagram of your cloud setup, where all the VMs are labeled with VM names, and IPs.

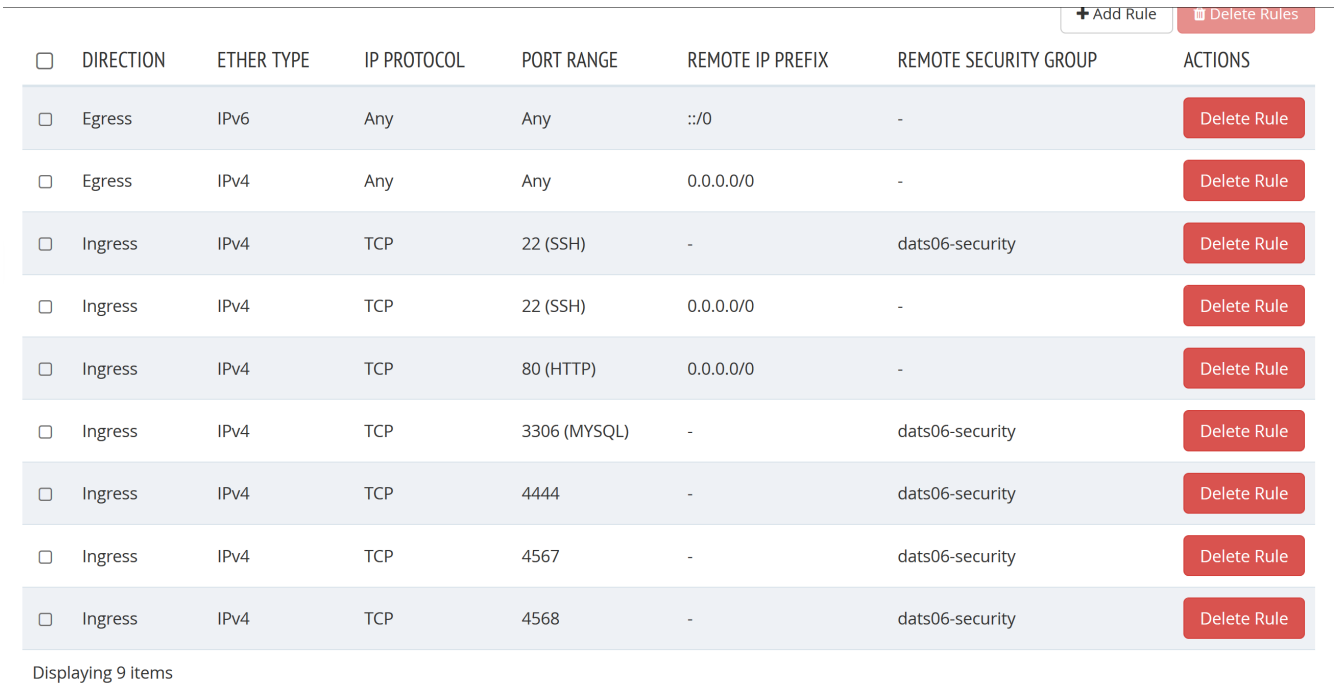


- Screenshots from ALTO showing created ssh key, security group and security group rules. Label security group rules indicating purpose of the rules.

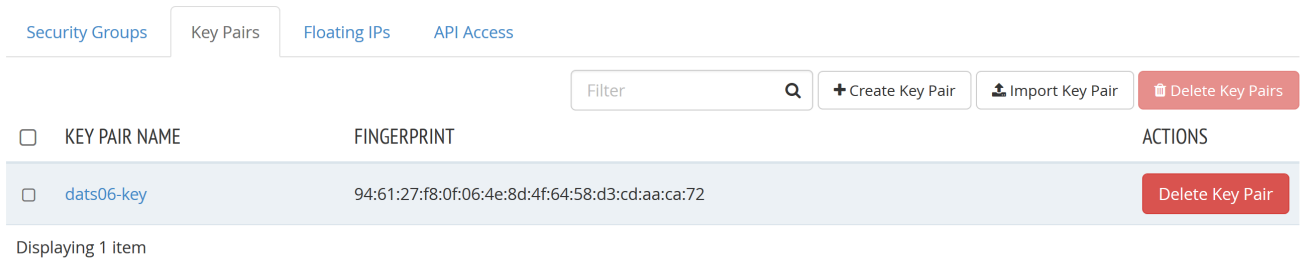
  Security groups:



Rules of dats06-security:



Keypairs:



A screenshot of the list of VMs created.

☐	dats06-db-3	Ubuntu16.04	10.10.7.148	m1.512MB4GB	dats06-key	Active	nova	None	Running	1 week, 1 day	Create Snapshot
☐	dats06-db-2	Ubuntu16.04	10.10.7.149	m1.512MB4GB	dats06-key	Active	nova	None	Running	1 week, 1 day	Create Snapshot
☐	dats06-db-1	Ubuntu16.04	10.10.7.147	m1.512MB4GB	dats06-key	Active	nova	None	Running	1 week, 1 day	Create Snapshot
☐	dats06-dbproxy	Ubuntu16.04	10.10.7.145	m1.512MB4GB	dats06-key	Active	nova	None	Running	1 week, 1 day	Create Snapshot
☐	dats06-web-3	Ubuntu16.04	10.10.7.144	m1.512MB4GB	dats06-key	Active	nova	None	Running	1 week, 1 day	Create Snapshot
☐	dats06-web-2	Ubuntu16.04	10.10.7.143	m1.512MB4GB	dats06-key	Active	nova	None	Running	1 week, 1 day	Create Snapshot
☐	dats06-web-1	Ubuntu16.04	10.10.7.142	m1.512MB4GB	dats06-key	Active	nova	None	Running	1 week, 1 day	Create Snapshot
☐	dats06-lb	Ubuntu16.04	10.10.4.41	m1.1GB	dats06-key	Active	nova	None	Running	1 month, 3 weeks	Create Snapshot

0

- A screenshot of the host names defined in /etc/hosts of one of the servers (say, datsXX-lb). Give short hostnames to the servers here, such as lb, web1, web2, web3, db1, db2, db3, and maxscale and use names in all the configurations instead of hard coded IPs.

  Load balancer's hosts file:

```
ubuntu@dats06:~$ cat /etc/hosts
10.10.4.41 dats06-lb lb
10.10.7.142 dats06-web-1 web1
10.10.7.143 dats06-web-2 web2
10.10.7.144 dats06-web-3 web3
10.10.7.145 dats06-dbproxy maxscale
10.10.7.147 dats06-db-1 db1
10.10.7.149 dats06-db-2 db2
10.10.7.148 dats06-db-3 db3

127.0.0.1 localhost

# The following lines are desirable for IPv6 capable hosts
::1 ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
ff02::3 ip6-allhosts
```

- A table listing the VMs with these information: VM name, hostname, IP, flavor, software you installed in the VM, and ports used for specific purpose(s).

VMs	Hostname	IP	Flavor	Software	Ports
dats06-lb	lb	10.10.4.41	m1.1GB	HAProxy	22, 80
dats06-web-1	web1	10.10.7.142	m1.512MB4GB	Nginex MariaDB-client MySQL php Git Cron	22, 80, 3306
dats06-web-2	web2	10.10.7.143	m1.512MB4GB	Nginex MariaDB-client MySQL php Git Cron	22, 80, 3306
dats06-web-3	web3	10.10.7.144	m1.512MB4GB	Nginex MariaDB-client MySQL php Git Cron	22, 80, 3306
dats06-db-1	db1	10.10.7.147	m1.512MB4GB	MariaDB 10.1	22, 3306, 4444, 4567, 4568
dats06-db-2	db2	10.10.7.149	m1.512MB4GB	MariaDB 10.1	22, 3306, 4444, 4567, 4568
dats06-db-3	db3	10.10.7.148	m1.512MB4GB	MariaDB 10.1	22, 3306, 4444, 4567, 4568
dats06-dbproxy	dbproxy	10.10.7.145	m1.512MB4GB	MaxScale	22, 3306, 4444, 4567, 4568

2. **HAProxy setup**: Setup HAProxy for the load balancer and monitoring. Load balancer should use round robin algorithm with equal weights. HAProxy monitoring page should be configured such that it can be accessed from the url, dats.vlab.cs.hioa.no:80XX/stats. Use your ALTO credentials for the authentication purpose.

- Provide screenshots of the HAProxy configuration.

  HAproxy config:

```
global
        log /dev/log    local0
        log /dev/log    local1 notice
        chroot /var/lib/haproxy
        stats socket /run/haproxy/admin.sock mode 660 level admin
        stats timeout 30s
        user haproxy
        group haproxy
        daemon

        # Default SSL material locations
        ca-base /etc/ssl/certs
        crt-base /etc/ssl/private

        # Default ciphers to use on SSL-enabled listening sockets.
        # For more information, see ciphers(1SSL). This list is from:
        #  https://hynek.me/articles/hardening-your-web-servers-ssl-ciphers/
        ssl-default-bind-ciphers ECDH+AESGCM:DH+AESGCM:ECDH+AES256:DH+AES256:ECDH+AES128:DH+AES:ECDH+3DES:DH+3DES:RSA+AESGCM:RSA+AES:RSA+3DES:!aNULL:!MD
5:!DSS
        ssl-default-bind-options no-sslv3

defaults
        log     global
        mode    http
        option  httplog
        option  dontlognull
        timeout connect 5000
        timeout client  50000
        timeout server  50000
        errorfile 400 /etc/haproxy/errors/400.http
        errorfile 403 /etc/haproxy/errors/403.http
        errorfile 408 /etc/haproxy/errors/408.http
        errorfile 500 /etc/haproxy/errors/500.http
        errorfile 502 /etc/haproxy/errors/502.http
        errorfile 503 /etc/haproxy/errors/503.http
        errorfile 504 /etc/haproxy/errors/504.http
```

```
frontend web-frontend
    bind *:80
    mode http
    default_backend web-backend

backend web-backend
    balance roundrobin
    mode http
    option httpchk HEAD / HTTP/1.1\r\nHost:\ localhost
    server web-3 web1:80 check weight 10
    server web-2 web2:80 check weight 10
    server web-1 web3:80 check weight 10


    # Monitoring
    stats enable
    stats refresh 30s
    stats uri /stats
    stats realm Haproxy\ Statistics
    stats auth dats06:"thrown similar river"
```

- Provide screenshot of the resulting monitoring web page.

HAProxy version 1.6.3, released 2015/12/25

Statistics Report for pid 2837

> General process information

pid = 2837 (process #1, nbproc = 1)
uptime = 1d 11h52m34s
system limits: memmax = unlimited; ulimit-n = 4034
maxsock = 4034; **maxconn =** 2000; **maxpipes =** 0
current conns = 4; current pipes = 0/0; conn rate = 0/sec
Running tasks: 1/11; idle = 100 %

active UP
active UP, going down
active DOWN, going up
active or backup DOWN
active or backup DOWN for maintenance (MAINT)
active or backup SOFT STOPPED for maintenance
backup UP
backup UP, going down
backup DOWN, going up
not checked
Note: "NOLB"/"DRAIN" = UP with load-balancing disabled.

Display option:
- Scope :
- Hide 'DOWN' servers
- Disable refresh
- Refresh now
- CSV export

External resources:
- Primary site
- Updates (v1.5)
- Online manual

web-frontend

	Queue			Session rate			Sessions						Bytes		Denied		Errors			Warnings		Server								
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Dwntme	Thrtle
Frontend				0	22	-	4	5	2 000	614			1 094 430	12 331 445	0	0	511					OPEN								

web-backend

	Queue			Session rate			Sessions						Bytes		Denied		Errors			Warnings		Server								
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Dwntme	Thrtle
web-1	0	0	-	0	14		0	1	-	79	79	1h25m	139 158	28 018		0		0	0	0	0	1m7s UP	L7OK/301 in 1ms	10	Y	-	3	1	34m16s	-
web-2	0	0	-	0	14		0	1	-	108	108	4m15s	146 826	35 029		0		0	0	0	0	1d11h UP	L7OK/301 in 1ms	10	Y	-	0	0	0s	-
web-3	0	0	-	0	14		0	1	-	107	107	4m16s	148 772	34 165		0		0	0	0	0	1d11h UP	L7OK/301 in 1ms	10	Y	-	0	0	0s	-
Backend	0	0		1	42		1	1	200	1 029	294	0s	1 094 430	12 235 252	0	0		0	0	0	0	1d11h UP		30	3	0		0	0s	

- Show screenshots of the test results confirming working load balancer by curling a web page testlb.php in a loop from a local computer (e.g., your laptop), which shows alternately changing the web server IP.

```
groot@DESKTOP-RV0FT5D:~/Git/Portfolio-exam-2-Cloud/scripts$ while true; do curl http://dats.vlab.cs.hioa.no:8006/test.php; echo ""; done
Server IP: 10.10.7.143
Server IP: 10.10.7.144
Server IP: 10.10.7.142
Server IP: 10.10.7.143
Server IP: 10.10.7.144
Server IP: 10.10.7.142
Server IP: 10.10.7.143
Server IP: 10.10.7.144
Server IP: 10.10.7.142
Server IP: 10.10.7.143
Server IP: 10.10.7.144
Server IP: 10.10.7.142
Server IP: 10.10.7.143
Server IP: 10.10.7.144
```

3. **Web server setup**: Setup all the web servers using Nginx with the support for dynamic web development with PHP and MariaDB and give a proper ownership and permission to the web root folder for the 'ubuntu' user.

- Provide a screenshot of Nginx configuration you updated to enable PHP support in one of the web servers (say, datsXX-web-1).

```
[ubuntu@dats06-web-1:~$ cat /etc/nginx/sites-available/default
server {
    listen 80 default_server;
    listen [::]:80 default_server;

    root /var/www/html;
    index index.php test.php students-grades.php index.html index.htm index.nginx-debian.html;

    server_name web1;

    location / {
        try_files  / =404;
    }

    location ~ \.php$ {
        include snippets/fastcgi-php.conf;
        fastcgi_pass unix:/run/php/php7.0-fpm.sock;
    }

    location ~ /\.ht {
        deny all;
    }
}
```

Setup a simple web deployment mechanism, where datsXX-web-1 is considered as the primary web server and whenever an updated web application is deployed to this server, the application is synchronized (pushed) to the other web servers automatically in every 3 minutes (using rsync and crontab).

- Provide a screenshot of the crontab showing the rsync commands used for synchronizing the web servers.

```
ubuntu@dats06-web-1:~$ cat rsyncScript.sh
#!/bin/bash
webs="web3 web2"
cd /var/www/html
git pull
for web in $webs; do
    rsync -chavz --delete --exclude ".*" -e "ssh -i ~/.ssh/dats06-key.pem -o StrictHostKeyChecking=no -o UserKnownHostsFile=/dev/null" /var/www/html ubuntu@$web:/var/www/
done
```

```
[ubuntu@dats06-web-1:~$ crontab -l
*/3 * * * * /bin/sh /home/ubuntu/rsyncScript.sh
ubuntu@dats06-web-1:~$
```

4. **HA database setup**: Setup a cluster-based high availability database with Galera cluster of three MariaDB database servers (nodes) and a MaxScale database proxy. Create a database named 'student_grades' with two tables as in the lecture slide and add some test data. Create a students-grades.php page that lists the grades of the students on the web, dats.vlab.cs.hioa.no:80XX/students-grades.php. For those who do not know much PHP, the example PHP code given in the lecture slide can be used. This web page should also show the host name or IP of the web server serving the page, at the bottom. Use the **same user name and password as your ALTO** to access the database from the PHP code.

   NB: We had some issues adding the repository for MariaDB 10.2 and have then decided to use 10.1

- Provide screenshots of the configurations (only for the changes you made) of the 3 database servers in the Galera cluster and the MaxScale proxy server.

  db1:

```
ubuntu@dats06-db-1:~$ cat /etc/mysql/conf.d/galera.cnf

[mysqld]
binlog_format=ROW
default-storage-engine=innodb
innodb_autoinc_lock_mode=2
bind-address=0.0.0.0

# Galera Provider Configuration
wsrep_on=ON
wsrep_provider=/usr/lib/galera/libgalera_smm.so

# Galera Cluster Configuration
wsrep_cluster_name="galera_cluster"
wsrep_cluster_address="gcomm://db3,db2,db1"

# Galera Synchronization Configuration
wsrep_sst_method=rsync

# Galera Node Configuration
wsrep_node_address="db1"
wsrep_node_name="db1"
ubuntu@dats06-db-1:~$
```

db2:

```
ubuntu@dats06-db-2:~$ cat /etc/mysql/conf.d/galera.cnf

[mysqld]
binlog_format=ROW
default-storage-engine=innodb
innodb_autoinc_lock_mode=2
bind-address=0.0.0.0

# Galera Provider Configuration
wsrep_on=ON
wsrep_provider=/usr/lib/galera/libgalera_smm.so

# Galera Cluster Configuration
wsrep_cluster_name="galera_cluster"
wsrep_cluster_address="gcomm://db3,db2,db1"

# Galera Synchronization Configuration
wsrep_sst_method=rsync

# Galera Node Configuration
wsrep_node_address="db2"
wsrep_node_name="db2"
ubuntu@dats06-db-2:~$
```

db3:

```
ubuntu@dats06-db-3:~$ cat /etc/mysql/conf.d/galera.cnf

[mysqld]
binlog_format=ROW
default-storage-engine=innodb
innodb_autoinc_lock_mode=2
bind-address=0.0.0.0

# Galera Provider Configuration
wsrep_on=ON
wsrep_provider=/usr/lib/galera/libgalera_smm.so

# Galera Cluster Configuration
wsrep_cluster_name="galera_cluster"
wsrep_cluster_address="gcomm://db3,db2,db1"

# Galera Synchronization Configuration
wsrep_sst_method=rsync

# Galera Node Configuration
wsrep_node_address="db3"
wsrep_node_name="db3"
ubuntu@dats06-db-3:~$
```

MaxScale:

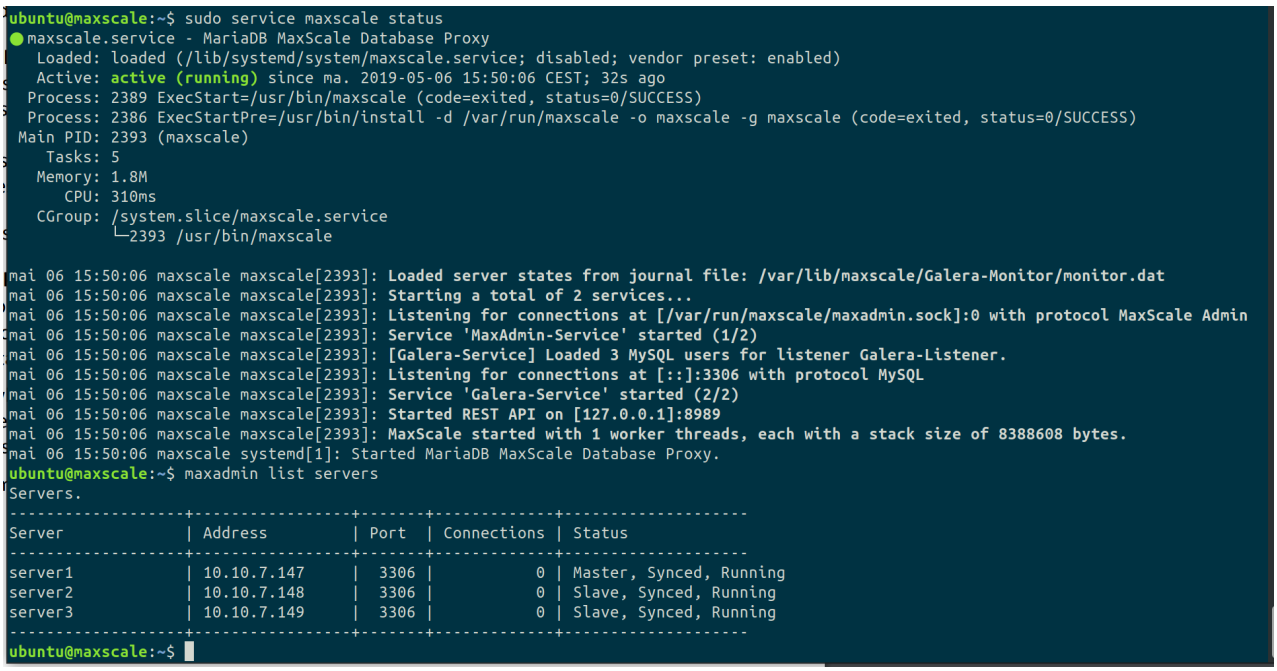
```
ubuntu@dats06-dbproxy:~$ cat /etc/maxscale.cnf

# Globals
[maxscale]
threads=4

# Servers
[server1]
type=server
address=db3
port=3306
protocol=MySQLBackend

[server2]
type=server
address=db2
port=3306
protocol=MySQLBackend

[server3]
type=server
address=db1
port=3306
protocol=MySQLBackend
```

```
# Monitoring for the servers
[Galera Monitor]
type=monitor
module=galeramon
servers=server1,server2,server3
user=maxScaleUsr
passwd=thrown similar river
monitor_interval=1000

# Galera router service
[Galera Service]
type=service
router=readwritesplit
servers=server1,server2,server3
user=maxScaleUsr
passwd=thrown similar river

# MaxAdmin Service
[MaxAdmin Service]
type=service
router=cli

# Galera cluster listener
[Galera Listener]
type=listener
service=Galera Service
protocol=MySQLClient
address=0.0.0.0
port=3306

# MaxAdmin listener
[MaxAdmin Listener]
type=listener
service=MaxAdmin Service
protocol=maxscaled
address=0.0.0.0
socket=default

ubuntu@dats06-dbproxy:~$
```

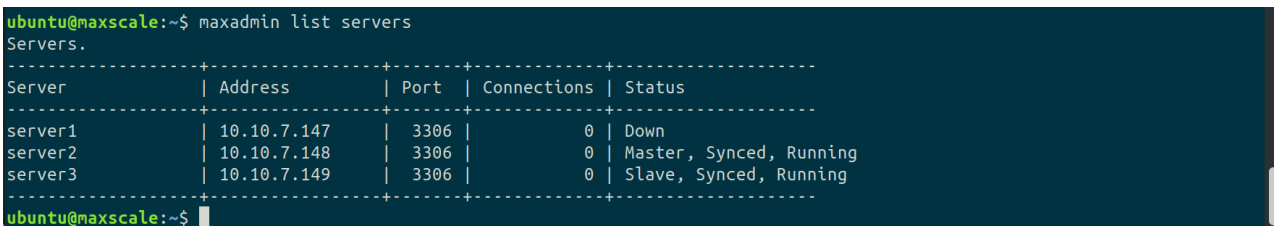
- Provide a screenshot confirming Galera cluster size to 3.

```
ubuntu@db3:~$ mysql -u root -p -e "show status like 'wsrep_cluster_size'"
Enter password:
+--------------------+-------+
| Variable_name      | Value |
+--------------------+-------+
| wsrep_cluster_size | 3     |
+--------------------+-------+
ubuntu@db3:~$
```

- Show the list of servers from the command *maxadmin list servers* in MaxScale confirming that the Galera cluster is working as expected.



- Also, provide a screenshot of the list of servers showing successful change of master role when the current master database is stopped



5. **Automation with scripts**: Automate all the setup tasks above (1 to 4) using bash shell scripts, and name the script files as: vm_setup.sh, lb_setup.sh, web_setup.sh, hadb_setup.sh respectively. Create one more script file, cloud_setup_all.sh which runs all the scripts and do all the tasks automatically by running this script. Only bash, python and OpenStack API commands are allowed in the scripts. All the shell scripts should be fully parameterized to avoid any hard coding of parameter values inside the scripts so that it can be used in any other projects just by modifying the relevant parameters in a parameter file and/or passing command-line parameters, but without modifying the script. Use a single parameter file, datsXX-params.sh to have most of the common parameters for all the scripts. Describe each script file briefly in the report about its usage and what it does.

   Script files should be **well documented with appropriate comments.**

   NB: Run the main script from the scripts directory, we have also made a instanceRubuildUbunutu16_04.sh to revert all VMs back to base Ubuntu 16.04 state to preserve IP on the ALTO cloud.

   NB: Make sure to have parallel-ssh installed (pssh)

Step by step explanation about the web deployment:

For the deployment part off the assignment we decided to make a automatic deployment using **crontab** and **GIT** that we found very useful when testing the "students-grades.php" script.

Make a public git repository (can be done with a secret repo but we decided to make it public for ease of access)

Limitations and thoughts:

We are aware that this solution have limitations but in a real developer environment we would use tools to automate the continues delivery pipeline. Originally we where going to use web hooks but made a decision to not and use crontab to auto pull from our Github repo on a set interval (3 minuets). This setup have more limitations that is linked to Git.

Initial clone from GitHub:

clone down the gitrepo from your preferd service on the "master webserver"

$GITPHPDEPLOYMENT = git repo clone link;

```
git clone $GITPHPDEPLOYMENT /var/www/html;
```

Create script that pushes and pulls the code from GitHub:

```
webs=""

git pull

for web in $webs;

do

rsync -chavz --delete --exclude ".*" -e "ssh -i ~/.ssh/KEY.pem -o StrictHostKeyChecking=no
-o UserKnownHostsFile=/dev/null" /var/www/html ubuntu@$web:/var/www/

done
```

This scripts pulls down the latest version of the git repo from our preferred service, then iterate over X numbers of web servers and pushes the newly updated Git repo folder from the master web server. This is done by using rsync and is done for every webserver except the master web server its executed on.

Nginx config setup:

Since we use nginx for the web server we need to edit the config to display a new site if added.

Got to: `/etc/nginx/sites-available/default`

Add your new website:

```
root /var/www/html;

index index.php "add your script"."the extention of choise" test.php index.html students-grades.php index.htm index.nginx-debian.html;
```

We made this parameterized and adds all files to the parameter file insted off hard coding it like here.

Make crontab execute every X minutes:

```
*/3 * * * * /bin/sh /home/ubuntu/"git pull script / rsync push script"
```

The crontab script over tells the vm to execute the script /home/ubuntu/"name" every 3 minuets.

So in short every 3 minutes we pull down the latest version from the git repo and pushes it out to the "slave" web servers.

Decisions: Explanation for the various decisions we have made throughout the project

- When making these scripts we assumed that the load balancer VM is already present and at a base ubuntu 16.04 setup with no added software. When the load balancer is revert to base Ubuntu the hostname is reset to dats06 and we then change it to lb.
- Keygen: We have decided to not create a new key since our load balancer, because it is never deleted, will have the dats06-key.pem key that existed before we ran the script. Therefore we continue using this key for any subsequent virtual machines we create in the scripts. If we were to make a key we would do it like this:

  ```
  # Creates keypair and puts it in the .ssh folder and gives it permission 400 so that no one except the owner can read it
  openstack keypair create $KEYPAIRNAME > $KEYLOCATION
  chmod 400 $KEYLOCATION
  ```

  This creates the key on the ALTO cloud and redirects it into a file on your computer and gives it proper permissions
- Security group: We, for this assignment, have only made a single security group, however we see that there is a possibility to have a second security group since the load balancer, for example, does not need any access to MySQL or the galera cluster. We think this will

Manual Setup:

Webserver setup manual

Basic websetup that installs dependencies, configure Nginx and setting up rsync with crontab deployment.

Install dependencies and file permissions:

First we install all the dependencies needed to setup the web server.

```
sudo apt-get update -y;

sudo apt-get upgrade -y;

sudo apt-get install nginx -y;

sudo systemctl start nginx.service;

sudo adduser ubuntu www-data;

sudo chown -R www-data:www-data /var/www;

sudo chmod -R g+rw /var/www;

sudo apt-get install php-fpm -y;

sudo apt-get install php-mysql -y
```

Setup Nginx:

To setup the Nginx config we overwrite the /etc/nginx/sites-availabel/default with the use off

`sudo bash -C "echo '"`

```
sudo bash -C "echo 'server {

listen 80 default_server;

listen [::]:80 default_server;

root /var/www/html;

index "ADD SITES" index.html index.htm index.nginx-debian.html;

server_name "Web server name";

location / {

    try_files $uri  $uri/ =404;

}

location ~ \.php$ {

    include snippets/fastcgi-php.conf;

   fastcgi_pass unix:/run/php/php7.0-fpm.sock;

}

location ~ /\.ht {

    deny all;

    }
}' > /etc/nginx/sites-available/default"
```

Note: after setting the Nginx config you have to restart Nginx.service

```
sudo systemctl restart nginx.service;
```

Setup rsync script

```
rsyncScript=('#!/bin/bash
webs=""
for web in $webs; do
    rsync -chavz --delete --exclude ".*" -e "ssh -i ~/.ssh/KEY.pem -o
StrictHostKeyChecking=no -o UserKnownHostsFile=/dev/null" /var/www/html
ubuntu@$web:/var/www/
done
')
```

We chose to make a rsyncScript file because in the automation script we implemented a automatic file sync from GitHub and it was more structured and was easier to edit this way.

```
(crontab -l ; echo \"*/3 * * * * /bin/sh /home/ubuntu/rsyncScript.sh\") | crontab -;
```

Adding websites

To add websites you add the `"name"."extetion"` to `/var/www/html` and add it to sites in the nginx config.

Steps to do a manual configuration of the DBs:

On all db servers run the following sequence of commands:

First we add the MariaDB repository key:

```
sudo apt-key adv --recv-keys --keyserver hkp://keyserver.ubuntu.com:80 0xF1656F24C74CD1D8
```

Next we add the repository, followed by a update of apt cache:

```
sudo add-apt-repository 'deb [arch=amd64,i386,ppc64el]
http://ftp.utexas.edu/mariadb/repo/10.1/ubuntu xenial main'
sudo apt-get update -y
```

When the repository is updated we install MariaDB and rsync. This is done using the noninteractive option to avoid being prompted for input during the installation. This will cause mariadb to be setup without a password for the root user. This is beneficial for the rest of the setup process (especially when scripting) but obviously bad from a security standpoint. Therefor it is advised to set a password using the following commands:

```
mysql -u root;
UPDATE mysql.user SET password = PASSWORD('new_password') WHERE user = 'root';
```

Note that we are not doing this now, but would otherwise do so. Now, onwards with the installation.

```
sudo DEBIAN_FRONTEND=noninteractive apt-get install mariadb-server rsync -y;
```

On MariaDB Galera cluster db1 do the following:

Instead of editing the config file found at /etc/mysql/my.cnf , we create a new configuration file.

```
sudo nano /etc/mysql/conf.d/galera.cnf
```

Next we append the following lines to the file:

```
[mysqld]
binlog_format=ROW
default-storage-engine=innodb
innodb_autoinc_lock_mode=2
bind-address=0.0.0.0

# Galera Provider Configuration
wsrep_on=ON
wsrep_provider=/usr/lib/galera/libgalera_smm.so

# Galera Cluster Configuration
wsrep_cluster_name="galera_cluster"
wsrep_cluster_address="gcomm://db1,db2,db3"

# Galera Synchronization Configuration
wsrep_sst_method=rsync

# Galera Node Configuration
wsrep_node_address="db1"
wsrep_node_name="db1"
```

Next we do the same operation on db2 and db3:

In this step we run the exact same command on both servers, the only difference is in the last two lines of the config file:

db2:

```
# Galera Node Configuration
wsrep_node_address="db2"
wsrep_node_name="db2"
```

db3:

```
# Galera Node Configuration
wsrep_node_address="db3"
wsrep_node_name="db3"
```

Screenshots of the resulting config files on all db servers. Note that we are not making changes to the my.cnf template file, as this is not a good approach for production. Instead we are creating new config files at /etc/mysql/conf.d/galera.cnf.

db1:

```
ubuntu@dats06-db-1:~$ cat /etc/mysql/conf.d/galera.cnf

[mysqld]
binlog_format=ROW
default-storage-engine=innodb
innodb_autoinc_lock_mode=2
bind-address=0.0.0.0

# Galera Provider Configuration
wsrep_on=ON
wsrep_provider=/usr/lib/galera/libgalera_smm.so

# Galera Cluster Configuration
wsrep_cluster_name="galera_cluster"
wsrep_cluster_address="gcomm://db3,db2,db1"

# Galera Synchronization Configuration
wsrep_sst_method=rsync

# Galera Node Configuration
wsrep_node_address="db1"
wsrep_node_name="db1"
ubuntu@dats06-db-1:~$
```

db2:

```
ubuntu@dats06-db-2:~$ cat /etc/mysql/conf.d/galera.cnf

[mysqld]
binlog_format=ROW
default-storage-engine=innodb
innodb_autoinc_lock_mode=2
bind-address=0.0.0.0

# Galera Provider Configuration
wsrep_on=ON
wsrep_provider=/usr/lib/galera/libgalera_smm.so

# Galera Cluster Configuration
wsrep_cluster_name="galera_cluster"
wsrep_cluster_address="gcomm://db3,db2,db1"

# Galera Synchronization Configuration
wsrep_sst_method=rsync

# Galera Node Configuration
wsrep_node_address="db2"
wsrep_node_name="db2"
ubuntu@dats06-db-2:~$
```

db3:

```
ubuntu@dats06-db-3:~$ cat /etc/mysql/conf.d/galera.cnf

[mysqld]
binlog_format=ROW
default-storage-engine=innodb
innodb_autoinc_lock_mode=2
bind-address=0.0.0.0

# Galera Provider Configuration
wsrep_on=ON
wsrep_provider=/usr/lib/galera/libgalera_smm.so

# Galera Cluster Configuration
wsrep_cluster_name="galera_cluster"
wsrep_cluster_address="gcomm://db3,db2,db1"

# Galera Synchronization Configuration
wsrep_sst_method=rsync

# Galera Node Configuration
wsrep_node_address="db3"
wsrep_node_name="db3"
ubuntu@dats06-db-3:~$
```

Next - starting the Galera cluster:

We run this command on all nodes:

```
sudo systemctl stop mysql
```

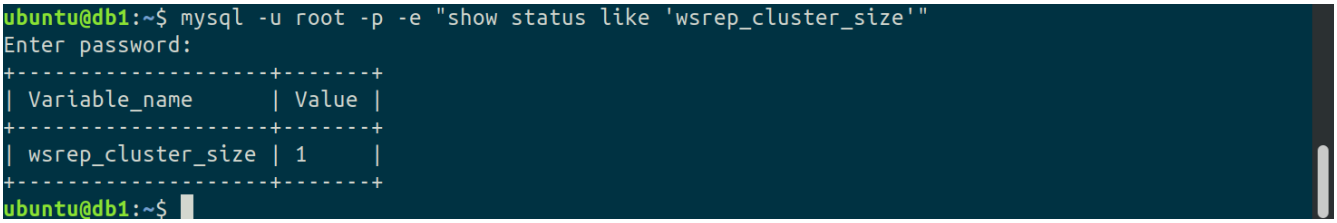
On the first node (db1), we start the cluster using the following command:

```
sudo galera_new_cluster
```

Next we check if the cluster is running using the following command:

```
mysql -u root -p -e "show status like 'wsrep_cluster_size'"
```

This produces the following output:



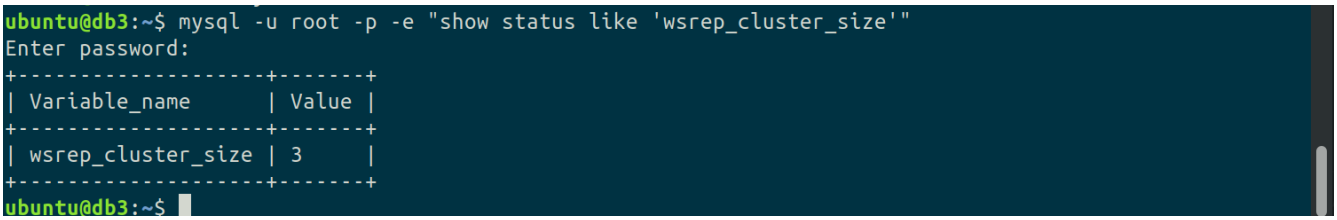
After verifying that the result is as expected, we run the following command on BOTH db2 and db3:

```
sudo systemctl start mysql
```

Then we do another check for cluster size, this time we expect size = 3 (this command can be run on any of the db servers):

```
mysql -u root -p -e "show status like 'wsrep_cluster_size'"
```

This produces the following output:



As evident by the screenshot, all nodes are part of the cluster.

Next we create the test database as specified. We are using a script to accomplish this. To avoid unnecessary clutter, this script is not shown here. But it is provided with the other script files if you wish to inspect it. To deploy via the script we run the following command:

```
mysql -u root < database-init-script.txt;
```

Then we create a user for web server access and grant the necessary permissions:

```
mysql -u root
create user 'datsO6'@'%' identified by '$thrown similar river';
grant select on mysql.user to 'datsO6'@'%';
grant select on student_grades.* to 'datsO6'@'%';
```

Now that the Galera cluster is configured, we move on to setting up MaxScale.

First we set up a user that MaxScale can use to connect to the cluster. We only need to run these commands on db1, as they are replicated to all servers by galera:

```
mysql -u root;
```

```
create user 'maxScaleUsr'@'maxscale' identified by 'thrown similar river';
grant select on mysql.user to 'maxScaleUsr'@'maxscale';
grant select on mysql.db to 'maxScaleUsr'@'maxscale';
grant select on mysql.tables_priv to 'maxScaleUsr'@'maxscale';
grant show databases on *.* to 'maxScaleUsr'@'maxscale';
```

On the MaxScale dbProxy instance, run the following commands:

Install maxscale:

```
sudo apt-get -y install maxscale
```

For sanity testing later, we will also install the mariaDB client:

```
apt-get -y install mariadb-client
```

Next we need to configure the MaxScale config file located at */etc/maxscale.cnf*. This is a bare minimum of what we need, but it is sufficient for this application:

```
# Globals
[maxscale]
threads=4

# Servers
[server1]
type=server
address=db1
port=3306
protocol=MySQLBackend

[server2]
type=server
address=db2
port=3306
protocol=MySQLBackend

[server3]
type=server
address=db3
port=3306
protocol=MySQLBackend

# Monitoring for the servers
[Galera Monitor]
type=monitor
module=galeramon
servers=server1,server2,server3
user=maxScaleUsr
passwd=thrown similar river
monitor_interval=1000

# Galera router service
[Galera Service]
type=service
router=readwritesplit
servers=server1,server2,server3
user=maxScaleUsr
passwd=thrown similar river

# MaxAdmin Service
[MaxAdmin Service]
type=service
router=cli

# Galera cluster listener
[Galera Listener]
type=listener
service=Galera Service
protocol=MySQLClient
```

```
# This needs to be here for ipv6 bug in maxscale
address=0.0.0.0
port=3306

# MaxAdmin listener
[MaxAdmin Listener]
type=listener
service=MaxAdmin Service
protocol=maxscaled
# This needs to be here for ipv6 bug in maxscale
address=0.0.0.0
socket=default
```

Screenshot of the resulting config file on the maxscale dbproxy server:

```
ubuntu@dats06-dbproxy:~$ cat /etc/maxscale.cnf

# Globals
[maxscale]
threads=4

# Servers
[server1]
type=server
address=db3
port=3306
protocol=MySQLBackend

[server2]
type=server
address=db2
port=3306
protocol=MySQLBackend

[server3]
type=server
address=db1
port=3306
protocol=MySQLBackend
```

```
# Monitoring for the servers
[Galera Monitor]
type=monitor
module=galeramon
servers=server1,server2,server3
user=maxScaleUsr
passwd=thrown similar river
monitor_interval=1000

# Galera router service
[Galera Service]
type=service
router=readwritesplit
servers=server1,server2,server3
user=maxScaleUsr
passwd=thrown similar river

# MaxAdmin Service
[MaxAdmin Service]
type=service
router=cli

# Galera cluster listener
[Galera Listener]
type=listener
service=Galera Service
protocol=MySQLClient
address=0.0.0.0
port=3306

# MaxAdmin listener
[MaxAdmin Listener]
type=listener
service=MaxAdmin Service
protocol=maxscaled
address=0.0.0.0
socket=default

ubuntu@dats06-dbproxy:~$
```

To avoid being asked for a password when starting the maxscale service, we need to add the user ubuntu to the maxscale group:

```
sudo adduser ubuntu maxscale
```

We start the maxscale service:

```
sudo systemctl start maxscale.service
```

Then we confirm everything running as expected:

```
Last login: Tue May  7 10:00:30 2019 from 10.10.0.14
ubuntu@maxscale:~$ maxadmin list servers
Servers.
-------------------+-----------------+-------+-------------+--------------------
Server             | Address         | Port  | Connections | Status
-------------------+-----------------+-------+-------------+--------------------
server1            | db1             |  3306 |           0 | Master, Synced, Running
server2            | db2             |  3306 |           0 | Slave, Synced, Running
server3            | db3             |  3306 |           0 | Slave, Synced, Running
-------------------+-----------------+-------+-------------+--------------------
ubuntu@maxscale:~$
```

To check change of master role, we disable server 1 by running the following command on db1:

```
sudo systemctl stop mysql
```

Next we run the following command on dbproxy and observe the result:

```
maxadmin list servers
```

```
ubuntu@maxscale:~$ maxadmin list servers
Servers.
-------------------+-----------------+-------+-------------+--------------------
Server             | Address         | Port  | Connections | Status
-------------------+-----------------+-------+-------------+--------------------
server1            | db1             |  3306 |           0 | Down
server2            | db2             |  3306 |           0 | Master, Synced, Running
server3            | db3             |  3306 |           0 | Slave, Synced, Running
-------------------+-----------------+-------+-------------+--------------------
ubuntu@maxscale:~$
```

6. **Group work details**: Provide following details on your group work.

How you worked as a team [How often did you meet, how tasks were distributed among your group members, whether you managed to make everyone participate and known about all the tasks (not just what s/he did), etc.].

- We have been working very well as a team, we have met almost every day the last 2 weeks of the assignment to work on it. In the beginning we assigned tasks to each member, one "part" to each one of us: Michael(VM-setup) Ole-Martin(LB-setup), Jakob(Web-setup) and Fredrik(Database-setup). We all know what everyone is doing because we help each other. The group has worked together in a good way which has contributed to

State who did what in terms of concrete tasks and contribution of the individual members in percentage (not for individual tasks, but as a whole project) using the one who contributed the most as a reference (i.e., 100%).

	Ole-Martin (s325905)	**Michael (s325903)**	**Jakob(s325908)**	**Fredrik(s325853)**
Task 1	x	x		
Task 2	x			
Task 3	x		x	
Task 4				x
vm_setup.sh	x	x		
lb_setup.sh	x			
web_setup.sh	x		x	
hadb_setup.sh	x			x
cloud_setup_all.sh	x	x	x	x
Task 6.		x		
Task 7.		x		

Due to the lb_setup.sh script being very small and easy to setup, Ole-Martin helped out the other scripts in addition to doing it.

Due to the interconnectivity of the tasks, everyone has contributed to some degree to every task. The ones marked are the main contributers to each task.

Work percentage:

Ole-Martin: 100%

Michael: 100%

Jakob: 100%

Fredrik: 100%

Problems or difficulties faced (if any) regarding working in the group.

- We have had prior experience working as a group, so the first thing we did when we started working on the assignment was to distribute the tasks among us and create a git repo. Due to all of this we have had very few issues working together as a group

7. **Self-evaluation**: Evaluate your own submission by filling up the table below. It has two parts: section-wise expected scores (out of the given full scores in brackets), and comments. Comments should be given point-wise whether you have done what has been asked properly (+), or not done or if any issues/weaknesses (-), and any other comments worth mentioning (*).

Tasks (points possible)	**Expected points**	**Comments**
VM-setup (25)	25	(+) All VMs setup correctly with proper hosts files
HAProxy setup (10)	10	(+) HAProxy setup to roundrobin traffic between web servers
Web server setup (15)	15	(+) DevOps: Automatic web deployment using Git (+) Web servers working correctly (+) Web servers displaying grades using PHP
HA DB setup (17)	17	(+) Galera setup properly (+) MaxScale setup properly (+) Databases working properly
Automation with scripts(25)	25	(+) params script with all parameters needed (+) All scripts setup and working properly
Group work details (3)	3	(+) Teamwork has been working well
Self-Evaluation (2)	2	(+) Honest self-evaluation
Report quality(3)	3	(+) Clean and readable report (+) All screenshots provided (+) All questions provided
Total score(100)	100	