* to deploy the LVS system, we start by creating and configuring LXC (Linux Containers) containers for the main components of the system, including the LVS Master, LVS Standby, and two Real Servers. These containers will be used to simulate the servers in the LVS system
* Use the lxc-create command to create four containers: one for lvs-master, one for lvs-standby, and the remaining two for real-servers. We then start these containers with the lxc-start command and configure a static IP address for each container, ensuring that each container can communicate with each other over the network.

*hanbiro@hanbiro:~$* ***sudo lxc-create -n lvs-master -t download -- --dist ubuntu --release***

*hanbiro@hanbiro:~$* ***sudo lxc-create -n lvs-standby -t download -- --dist ubuntu --release***

*hanbiro@hanbiro:~$* ***sudo lxc-create -n real-server-1 -t download -- --dist ubuntu --release***

*hanbiro@hanbiro:~$* ***sudo lxc-create -n real-server-2 -t download -- --dist ubuntu --release***

* start 4 containers:

sudo lxc-start -n lvs-master

sudo lxc-start -n lvs-standby

sudo lxc-start -n real-server-1

sudo lxc-start -n real-server-2

* set static IP for lvs-master (open 4 terminals for each container):

access: **sudo lxc-attach -n lvs-master**

**vi /etc/netplan/10-lxc.yaml**

network:

version: 2

ethernets:

eth0:

addresses:

**- 10.0.3.107/24**

gateway4: 10.0.3.1

nameservers:

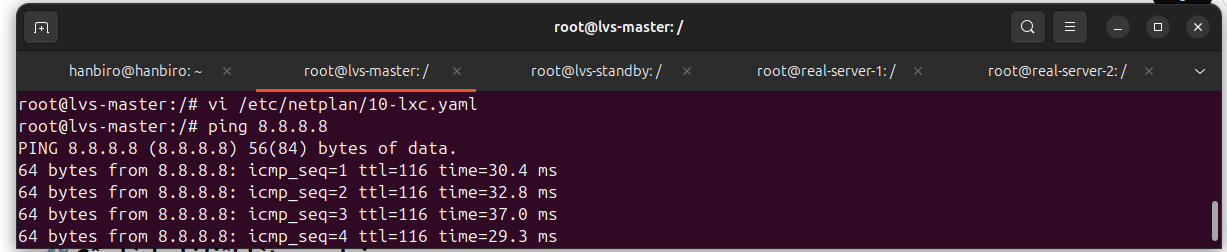
addresses: [8.8.8.8, 8.8.4.4]

netplan apply

ip a

ping -c 4 8.8.8.8





* set static IP for lvs-standby:

access: **sudo lxc-attach -n lvs-standby**

**vi /etc/netplan/10-lxc.yaml**

network:

version: 2

ethernets:

eth0:

addresses:

**- 10.0.3.108/24**

gateway4: 10.0.3.1

nameservers:

addresses: [8.8.8.8, 8.8.4.4]

netplan apply

ip a

ping -c 4 8.8.8.8

* set static IP for real-server1:

access: **sudo lxc-attach -n real-server-1**

**vi /etc/netplan/10-lxc.yaml**

network:

version: 2

ethernets:

eth0:

addresses:

**- 10.0.3.109/24**

gateway4: 10.0.3.1

nameservers:

addresses: [8.8.8.8, 8.8.4.4]

netplan apply

ip a

ping -c 4 8.8.8.8

* set static IP for real-server-2:

access: **sudo lxc-attach -n real-server-2**

**vi /etc/netplan/10-lxc.yaml**

network:

version: 2

ethernets:

eth0:

addresses:

**- 10.0.3.108/24**

gateway4: 10.0.3.1

nameservers:

addresses: [8.8.8.8, 8.8.4.4]

netplan apply

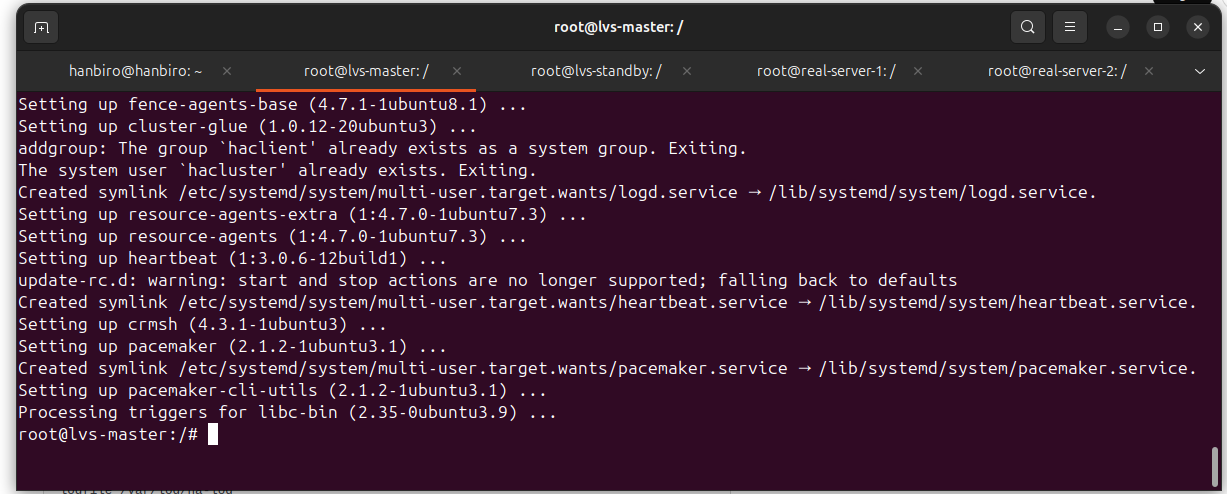
ip a

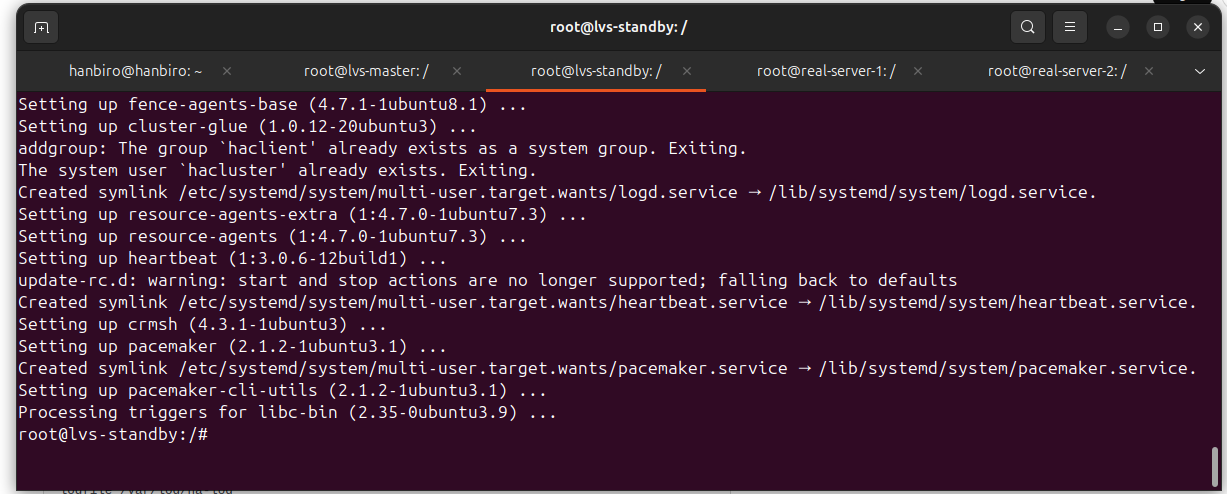
ping -c 4 8.8.8.8

* install the tools needed to deploy LVS and HA (High Availability). The two main software that need to be installed are **ipvsadm** (LVS management and configuration tool) and **Heartbeat** (tool that helps manage VIP handover between LVS Master and Standby). On both lvs-master and lvs-standby

**apt update**

**apt install -y ipvsadm heartbeat**



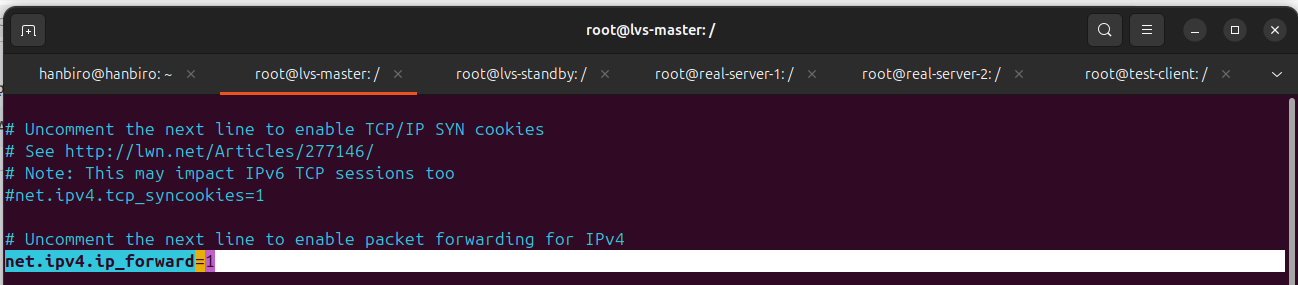


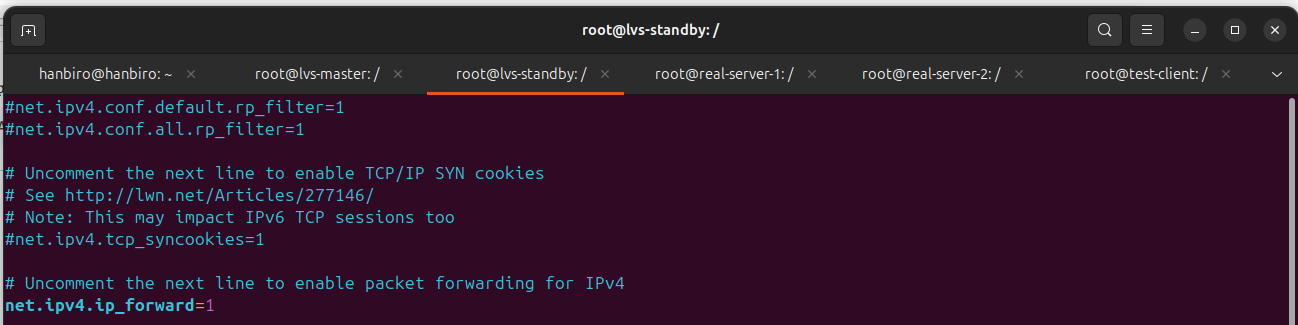
* adding **net.ipv4.ip\_forward=1** to the **/etc/sysctl.conf** file is required to enable IP forwarding – a mandatory condition when using LVS with NAT type (masq), because LVS will need to forward packets from client to real server and vice versa

**vi /etc/sysctl.conf**

*uncommand:* **net.ipv4.ip\_forward=1**

*apply:* **sysctl -p**

****



**Configure Heartbeat to ensure failover between servers**

* create and configure **vi /etc/ha.d/ha.cf (**both on **lvs-master and lvs-standby)**

*logfile /var/log/ha-log*

*logfile /var/log/heartbeat.log*

*keepalive 2*

*deadtime 10*

*warntime 5*

*initdead 20*

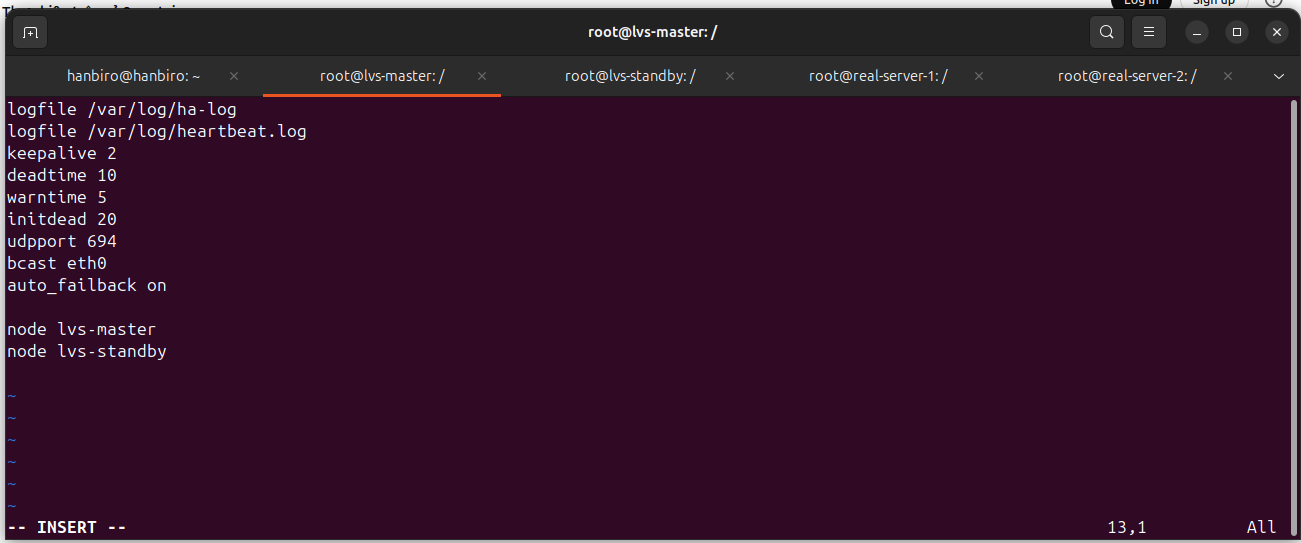
*udpport 694*

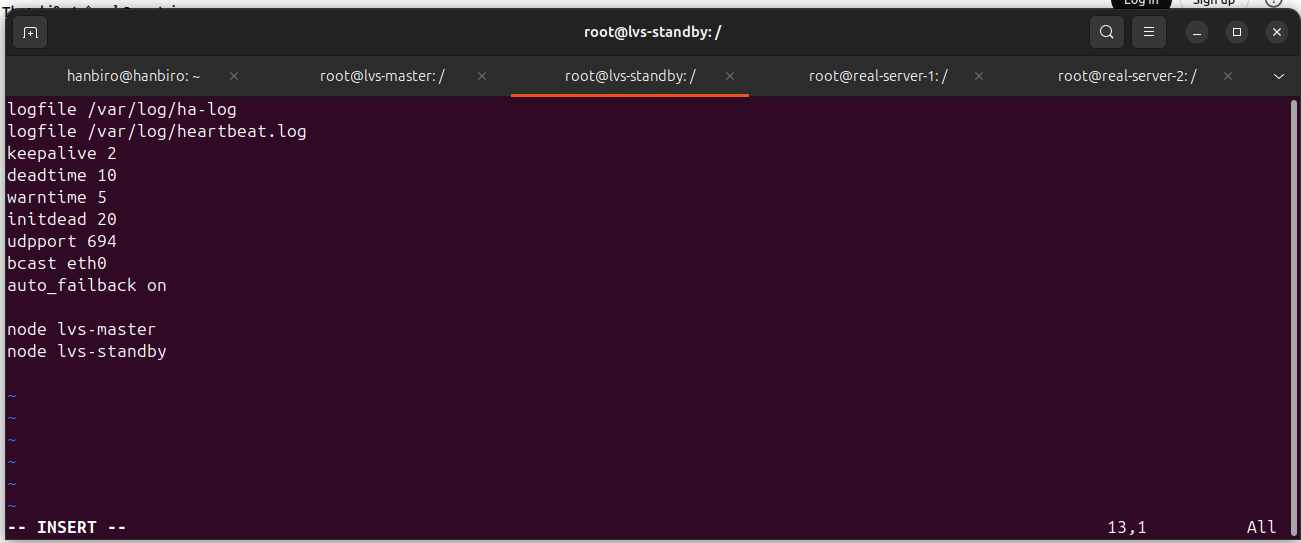
*bcast eth0*

*auto\_failback on*

*node lvs-master*

*node lvs-standby*

**

**

**logfile:** Defines the log files to store Heartbeat status and event information.

**keepalive:** Sets the interval (2 seconds) between "keep-alive" signals to check the status of the nodes.

**deadtime:** The interval (10 seconds) Heartbeat will wait before determining a node is dead.

**warntime:** The interval (5 seconds) before Heartbeat reports a failure.

**initdead:** The interval (20 seconds) before starting to monitor the status of the nodes.

**udpport:** The UDP port (694) for communication between nodes in the cluster.

**bcast:** Configures the broadcast protocol over the eth0 network interface.

**auto\_failback:** Enables automatic failover to the master node (if the slave node becomes active again).

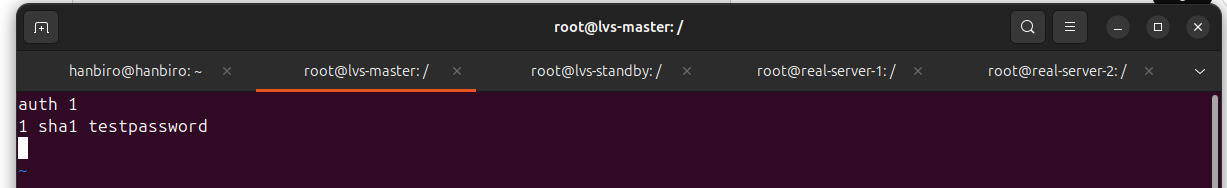
**node:** Defines the nodes in the cluster, here there are two nodes lvs-master and lvs-standby.

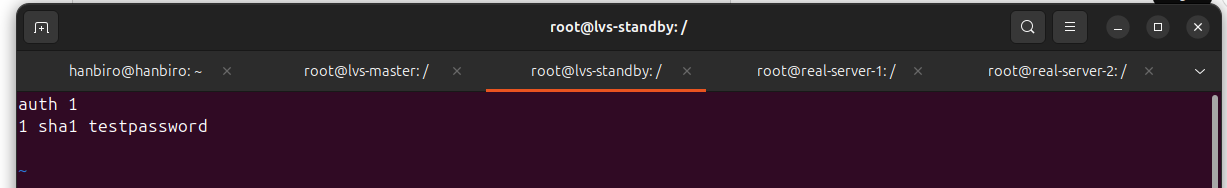
* *configure* ***vi /etc/ha.d/authkeys*** *(security configuration for Heartbeat) -* ***(****both on* ***lvs-master and lvs-standby)***

*auth 1*

*1 sha1 testpassword*

*sudo chmod 600 /etc/ha.d/authkeys*

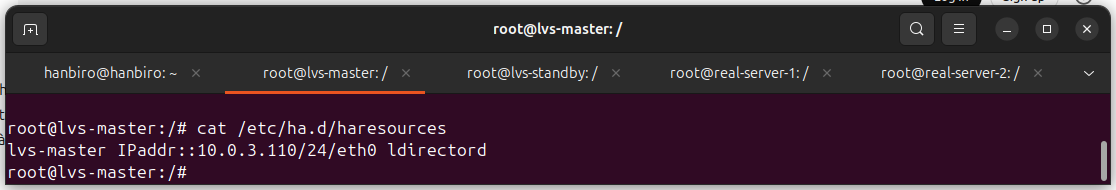
**

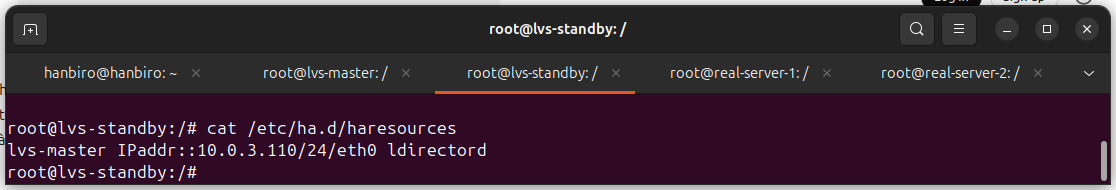
**

***Configure Virtual IP (VIP) is the IP that LVS will transfer when there is failover***

* *create and configure the file* ***vi /etc/ha.d/haresources (****both on* ***lvs-master and lvs-standby)***

*lvs-master IPaddr::10.0.3.110/24/eth0 ldirectord*

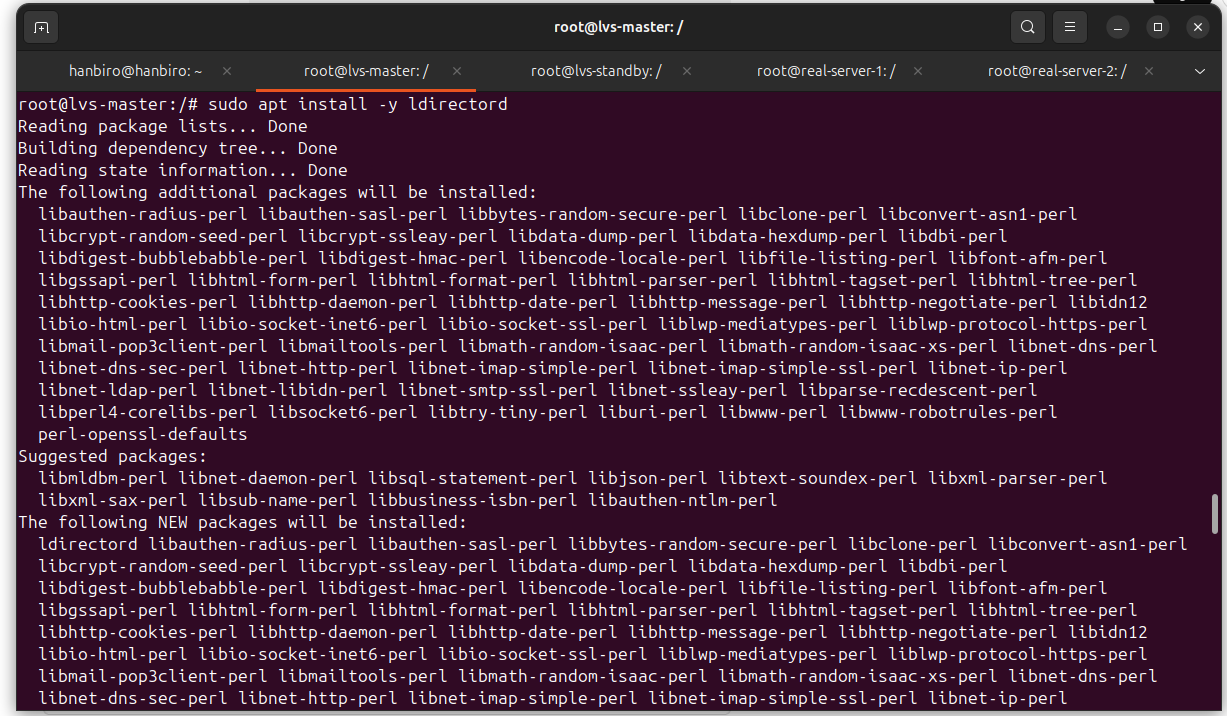
**

**

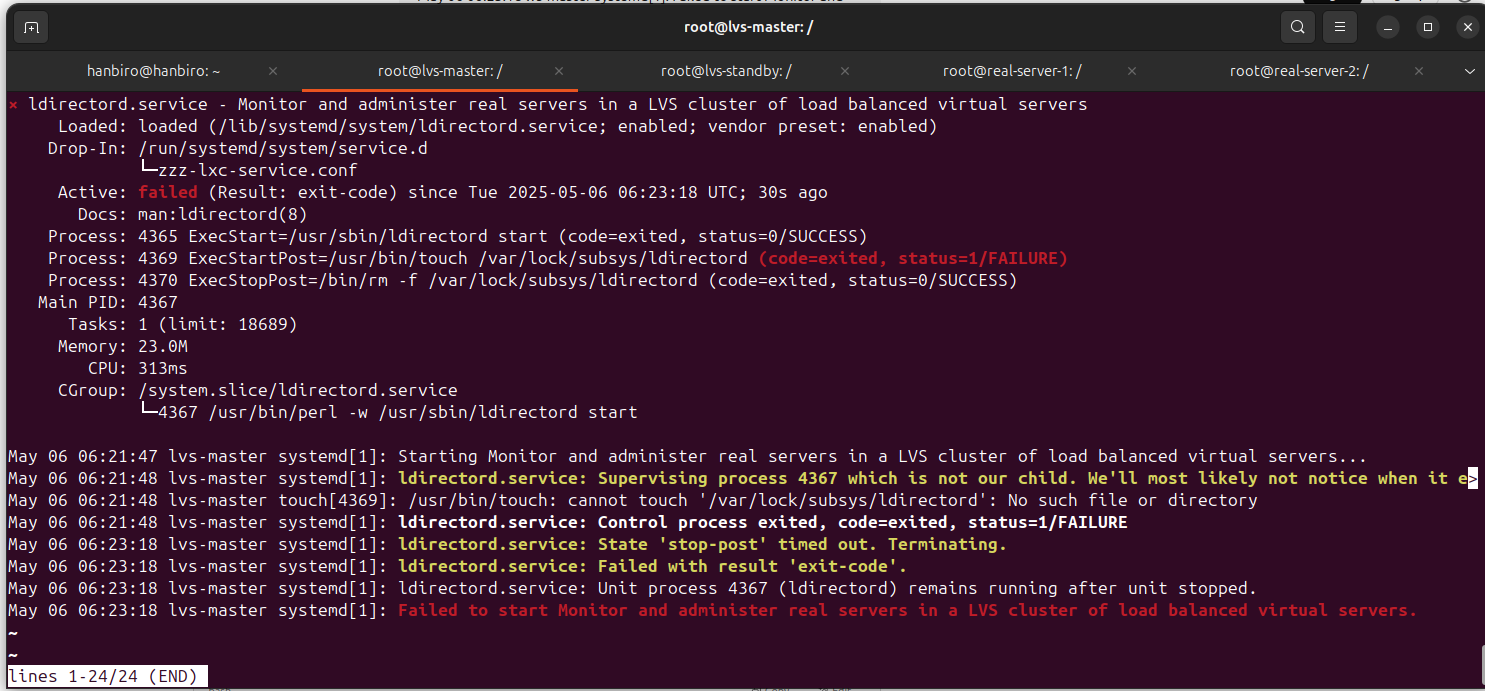
*=> This configuration means: if lvs-master fails, VIP will switch to lvs-standby*

***Configure ldirectord on LVS master-standby*** *(ldirectord is a tool for monitoring and distributing traffic to real servers)*

* *install ldirectord:* ***sudo apt install -y ldirectord***

**

***issue****: touch: cannot touch '/var/lock/subsys/ldirectord': No such file or directory*

**

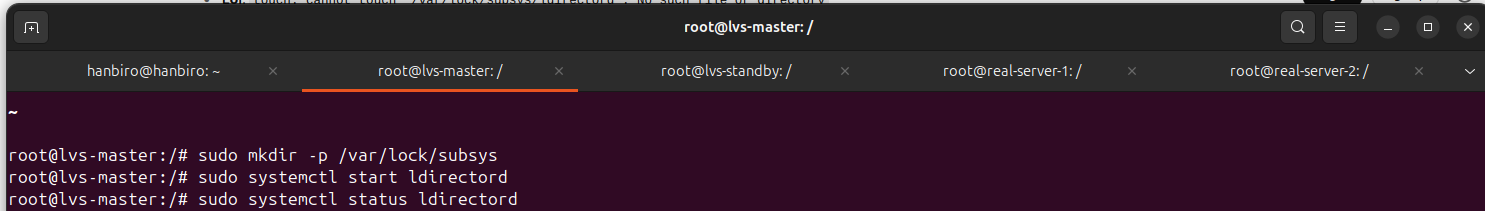
***resolve****:*

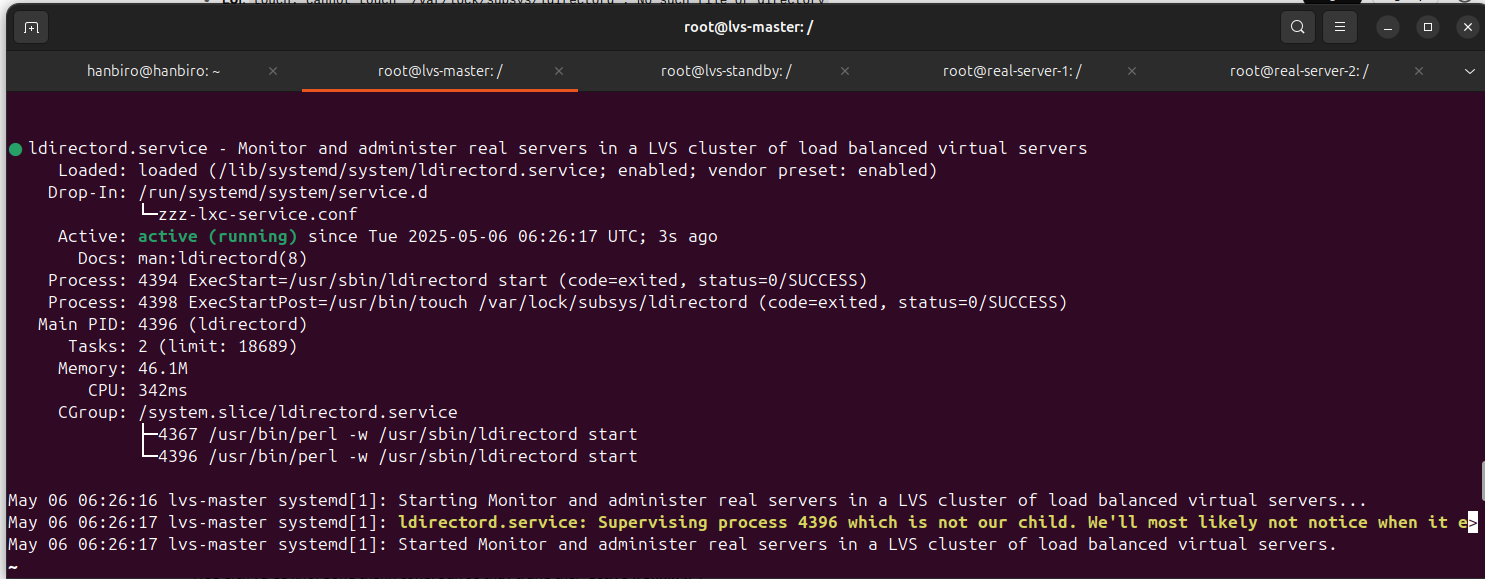
*sudo mkdir -p /var/lock/subsys*

*sudo systemctl enable ldirectord*

*sudo systemctl start ldirectord*

*sudo systemctl status ldirectord*

**

**

* *continue on LVS master, Create and edit configuration file* ***vi /etc/ha.d/ldirectord.cf****:*

*checktimeout=10*

*checkinterval=2*

*autoreload=no*

*logfile="/var/log/ldirectord.log"*

*quiescent=no*

*virtual=10.0.3.110:80*

*fallback=127.0.0.1:0*

*real=10.0.3.109:80 masq 100*

*service=http*

*request="hanbiro.lvs"*

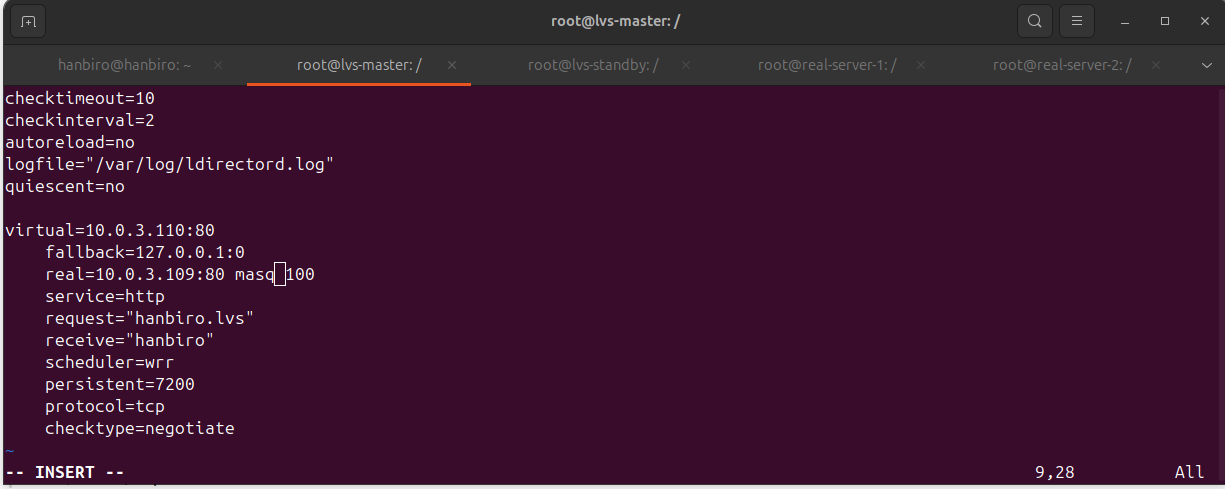
*receive="hanbiro"*

*scheduler=wrr*

*persistent=7200*

*protocol=tcp*

*checktype=negotiate*

**

* *continue on LVS standby, Create and edit configuration file* ***vi /etc/ha.d/ldirectord.cf****:*

*checktimeout=10*

*checkinterval=2*

*autoreload=no*

*logfile="/var/log/ldirectord.log"*

*quiescent=no*

*virtual=10.0.3.110:80*

*fallback=127.0.0.1:0*

*real=10.0.3.109:80 masq 100*

*service=http*

*request="hanbiro.lvs"*

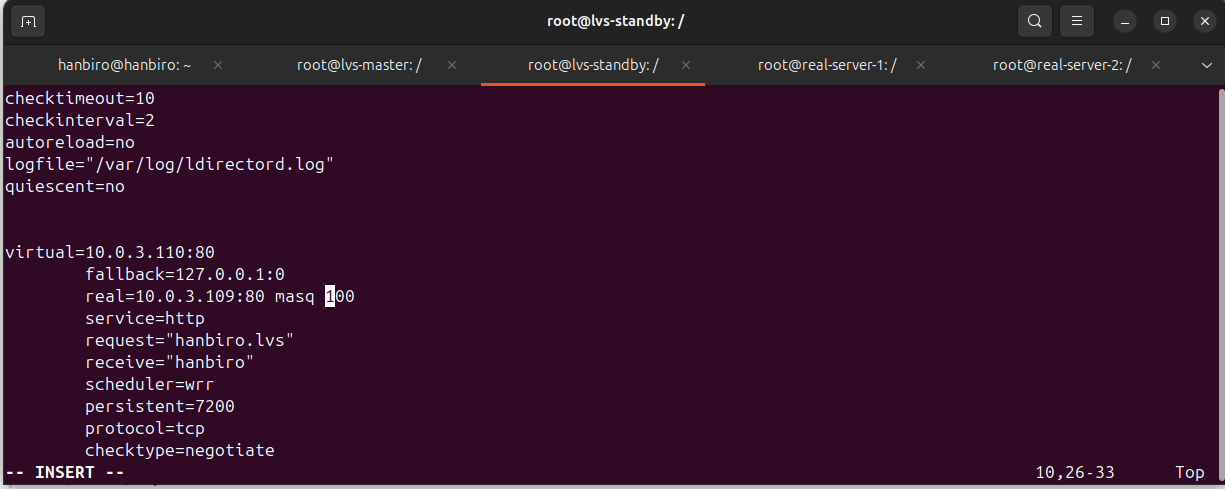
*receive="hanbiro"*

*scheduler=wrr*

*persistent=7200*

*protocol=tcp*

*checktype=negotiate*

**

***What does each option mean on ldirectord.cf ?***

**checktimeout=10:** Maximum time (seconds) that ldirectord will wait for a response from the real servers. If this time is exceeded, the server will be considered "down"

**checkinterval=2:** Interval (seconds) between consecutive checks for the real server

**autoreload=no:** If set to "no", ldirectord will not automatically reload the configuration when there are changes in the configuration files

**logfile="/var/log/ldirectord.log":** Path to the log file where ldirectord status information is stored

**quiescent=no:** If set to "yes", ldirectord will not forward new connections to the real server when there are changes in the configuration

**virtual=10.0.3.110:80:** Virtual IP address (VIP) and port that LVS will listen on

**fallback=127.0.0.1:0:** Fallback address and port. This is the server that will be used if no real server is available

**real=10.0.3.109:80 masq 100:** the IP address of the real server and the service port. The "masq" parameter indicates that NAT will be used to hide the real server's real IP addresses when responding to clients

**service=http:** The HTTP service that LVS is serving

**request="hanbiro.lvs":** the request data that will be sent to the real server to check for availability

**receive="hanbiro":** the data that ldirectord expects to receive from the real server during the check

**scheduler=wrr:** select the scheduler (Weighted Round Robin) to distribute traffic to the real servers

**persistent=7200:** the time (seconds) that the HTTP connection will be kept alive if the same client continues to send requests to LVS

**protocol=tcp:** the protocol used is TCP

**checktype=negotiate:** the type of check. When "negotiate", ldirectord will perform the check by opening a connection and waiting for a response from the real server (which can be a "hello" or a handshake)

***Start Heartbeat and ldirectord****: after configuring the files,* ***start Heartbeat and ldirectord on lvs-master and lvs-standby***

* *both on**lvs-master and lvs-standby****:***

**sudo systemctl enable heartbeat**

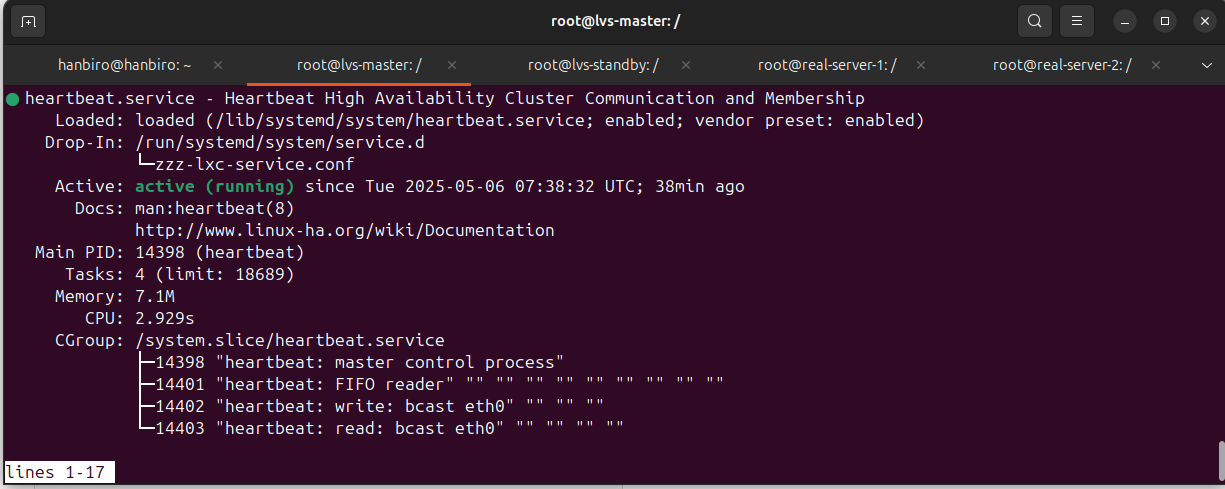
**sudo systemctl start heartbeat**

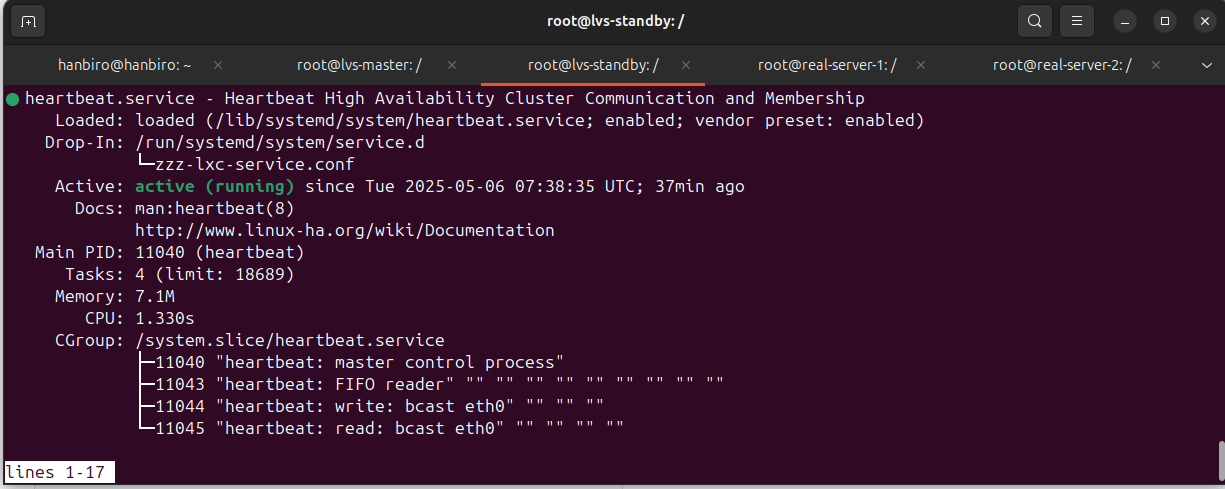
**sudo systemctl status heartbeat**

**sudo systemctl enable ldirectord**

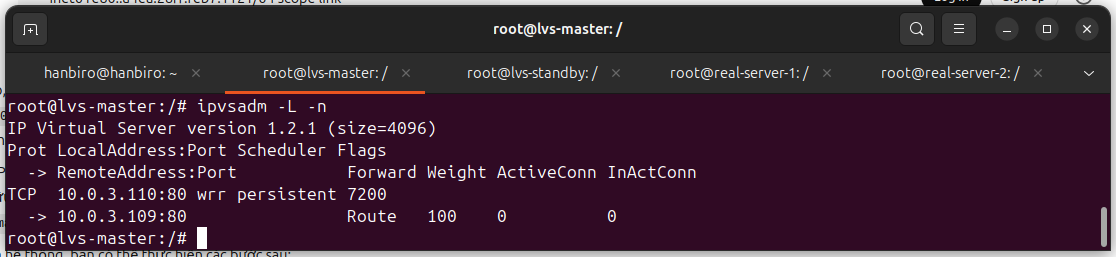
**sudo systemctl start ldirectord**

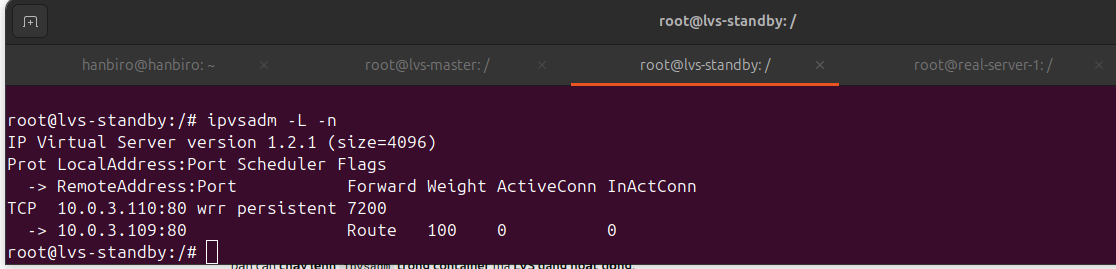
**sudo systemctl status ldirectord**





* Check LVS status on LVS: **ipvsadm -L -n**

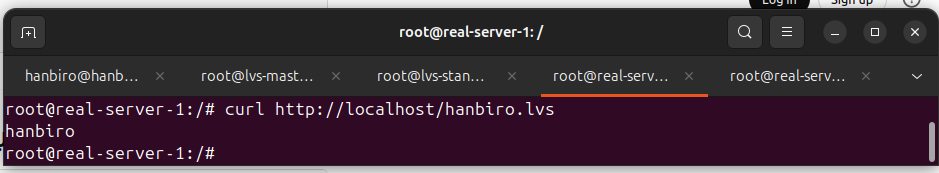
****

****

* Install Apache and create test file — TESTED ON: **real-server-1**

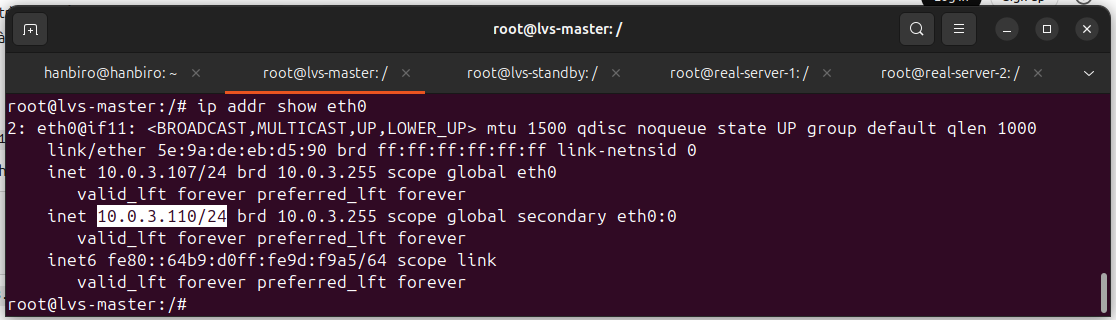
echo "hanbiro" > /var/www/html/hanbiro.lvs

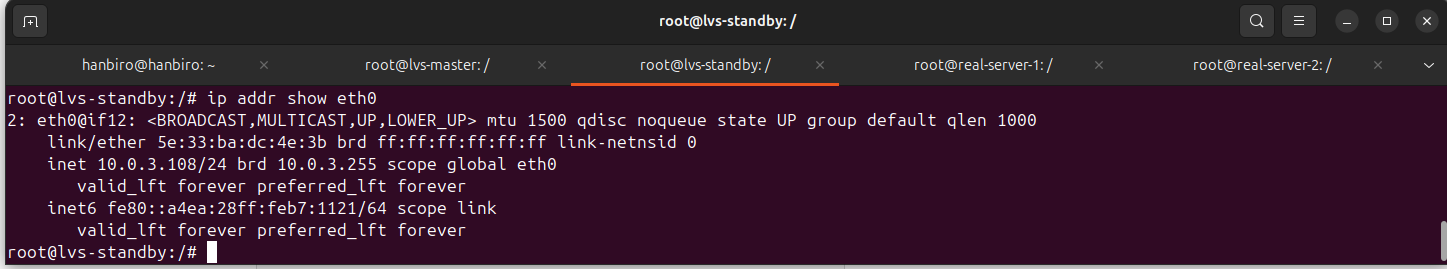
curl http://localhost/hanbiro.lvs

****

* check the VIP configuration (10.0.3.110) on lvs-master and **lvs-standby(No VIP 10.0.3.110)**

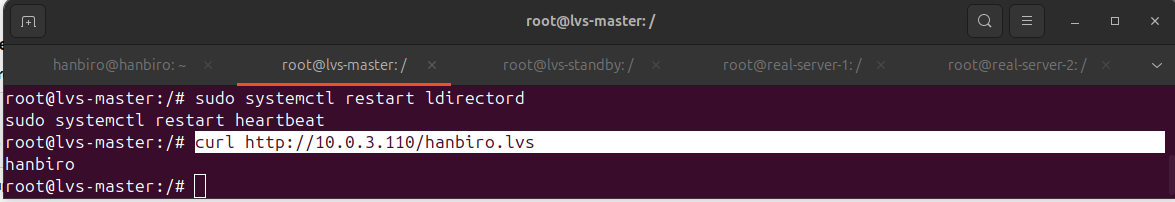
ip addr show eth0





* check HTTP access by VIP on lvs-master

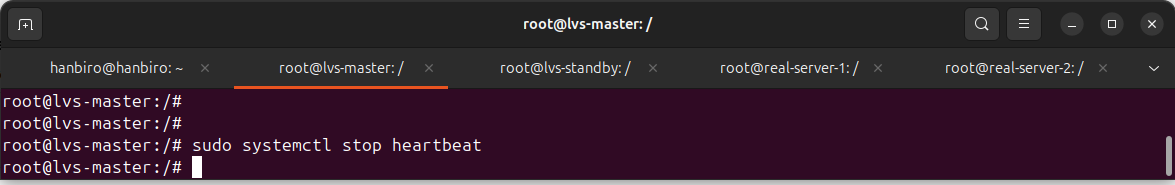
**curl http://10.0.3.110/hanbiro.lvs**



**[ failover test with heartbeat ]**

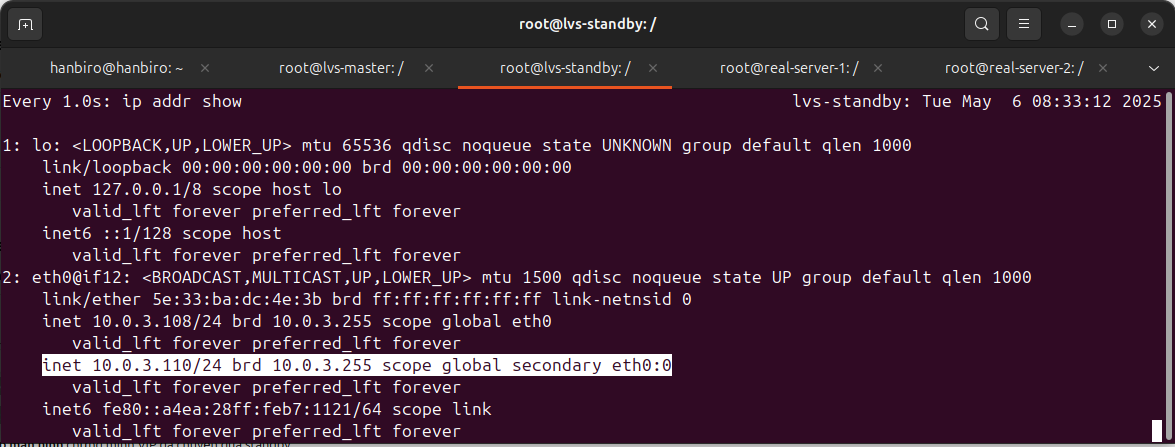
* stop Heartbeat on LVS Master (emulate error)

sudo systemctl stop heartbeat



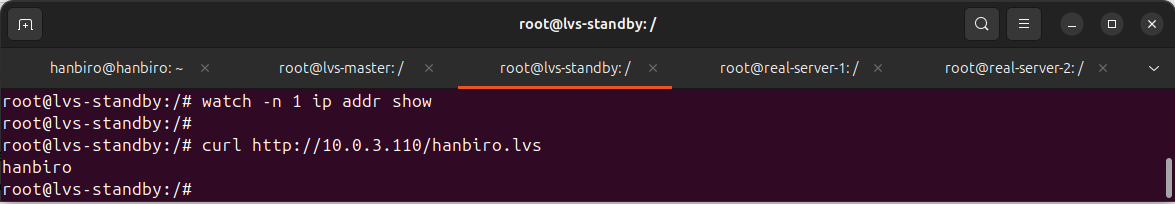
* in **lvs-standby** container, run **watch -n 1 ip addr show** and wait about 10–20 seconds to see VIP 10.0.3.110 appear on eth0 of LVS Standby

*\*see: inet 10.0.3.110/24 brd 10.0.3.255 scope global secondary eth0:0*



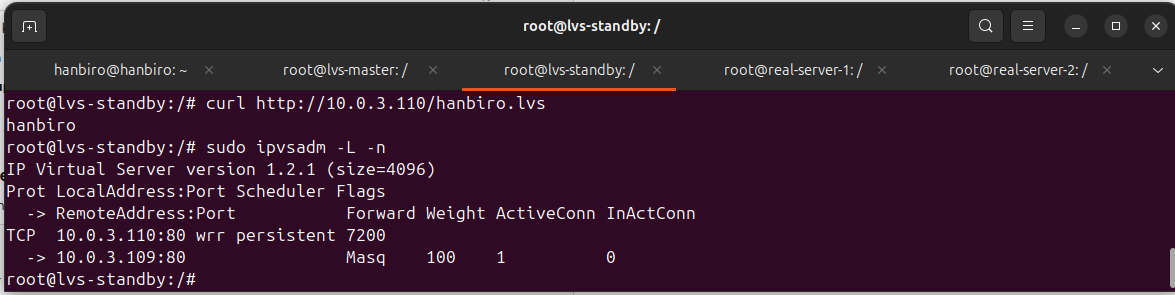
* Check access via VIP on LVS Standby

**curl http://10.0.3.110/hanbiro.lvs**

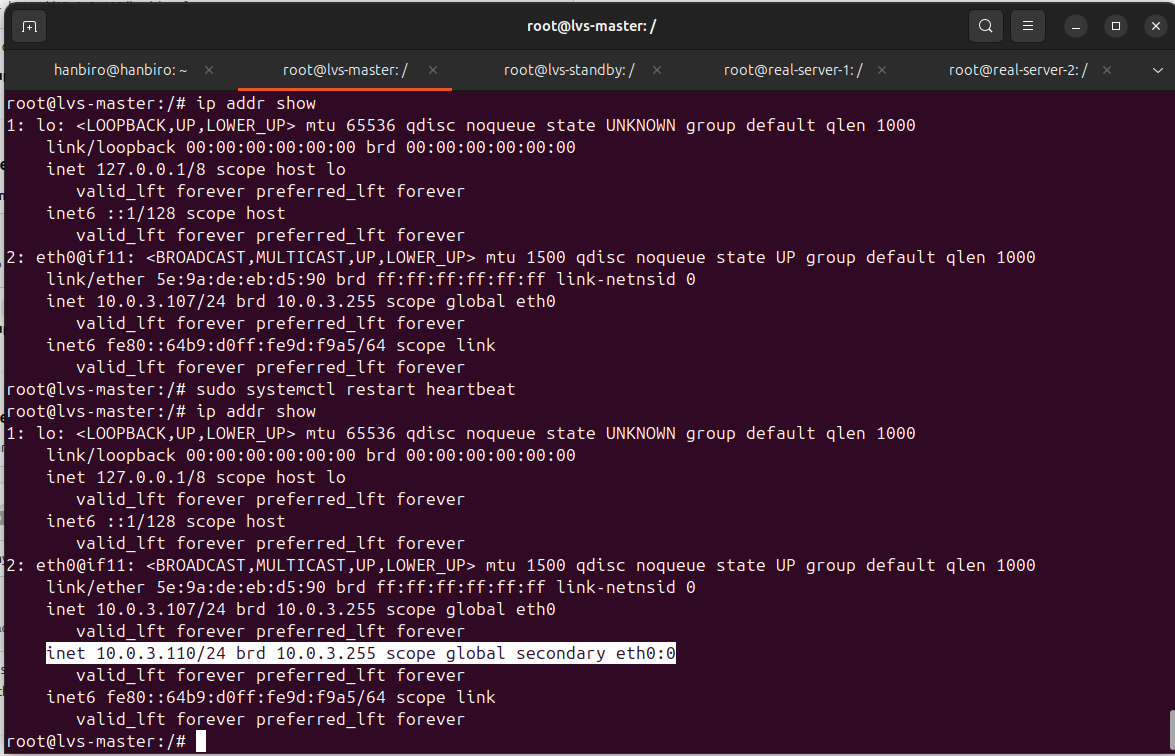


* Check LVS table on LVS Standby

**sudo ipvsadm -L -n**



* Turn LVS Master back on to test auto-failback (**auto\_failback on**). Check with: **ip addr show**



**SERVER COMPONENTS 2 CONFIGURATION**

**lvs-master**

**lvs-standby + Real Server 1**

**real-server-2**

**Configure lvs-standby to become Real Server**

* access container **lvs-standby**

sudo lxc-attach -n lvs-standby

* install apache:

apt update

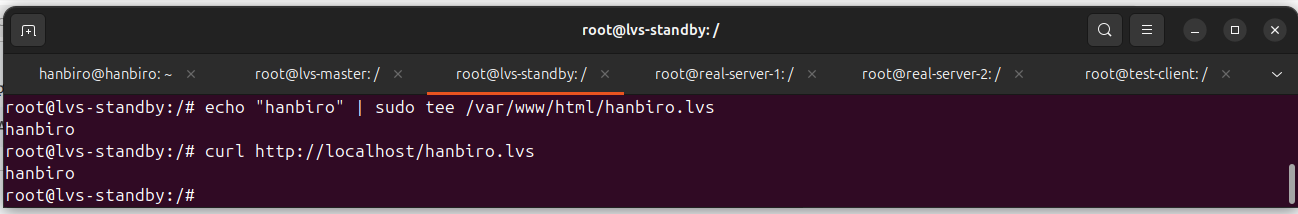
apt install -y apache2

* create test file:

echo "hanbiro" | sudo tee /var/www/html/hanbiro.lvs

* check:

curl http://localhost/hanbiro.lvs



* on **real-server-2**

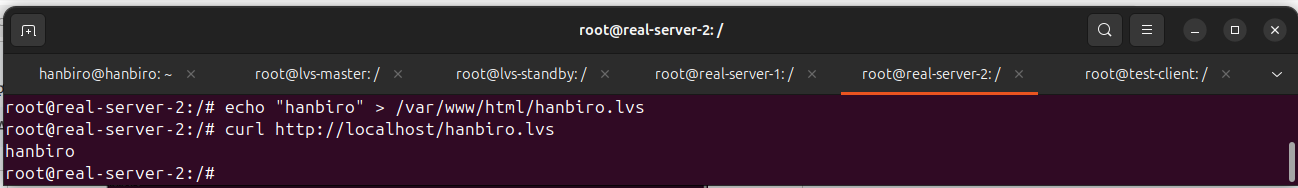
install apache:

apt update

apt install -y apache2

echo "hanbiro" > /var/www/html/hanbiro.lvs

curl http://localhost/hanbiro.lvs



* Update ldirectord.cf (on both lvs-master and lvs-standby)

**vi /etc/ha.d/ldirectord.cf**

checktimeout=10

checkinterval=2

autoreload=no

logfile="/var/log/ldirectord.log"

quiescent=no

virtual=10.0.3.110:80

fallback=127.0.0.1:0

**real=10.0.3.108:80 masq 100**

real=10.0.3.109:80 masq 100

service=http

request="hanbiro.lvs"

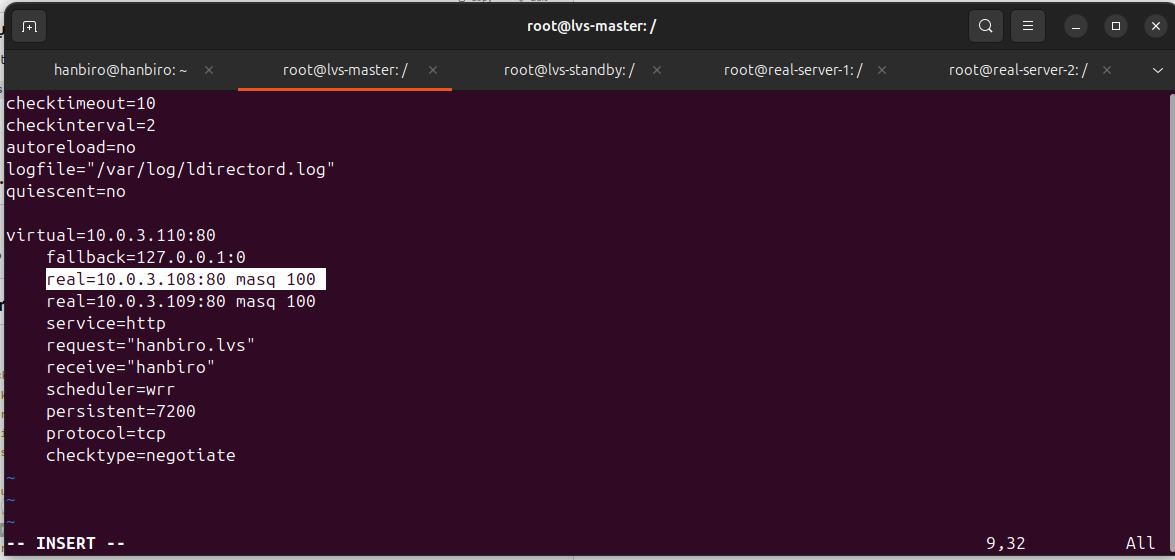
receive="hanbiro"

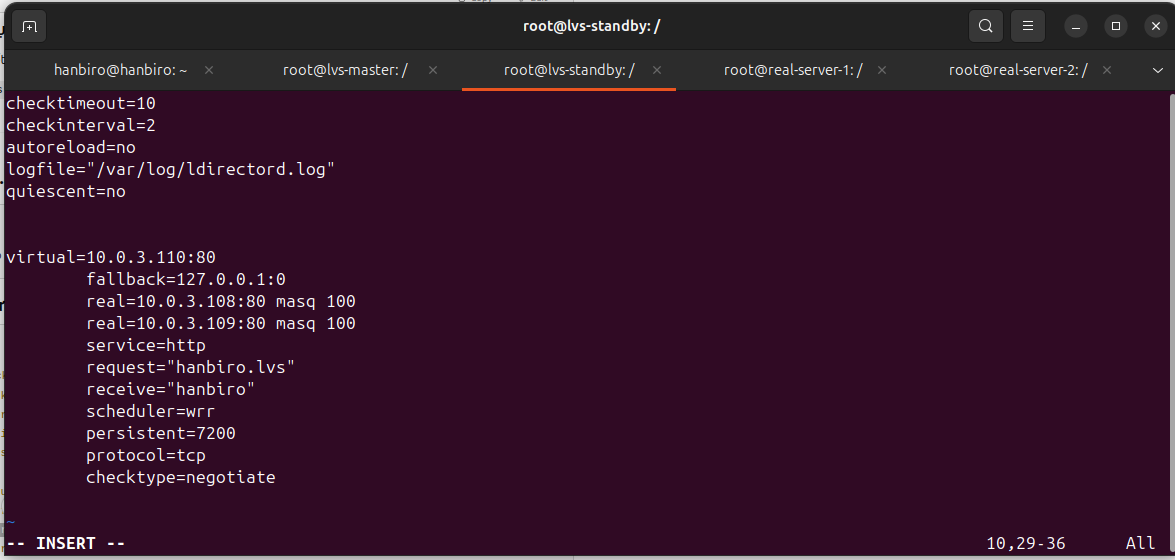
scheduler=wrr

persistent=7200

protocol=tcp

checktype=negotiate





\*with:

**virtual=10.0.3.110:80**: VIP address that will serve HTTP requests

**real=10.0.3.108:80 masq 100**: This is Real Server 2 (real-server-2 | 10.0.3.108, configured as lvs-standby)

**real=10.0.3.109:80 masq 100**: This is Real Server 1 (real-server-1 | 10.0.3.109)

**checktimeout, checkinterval**: Time and frequency of checking the status of the real servers

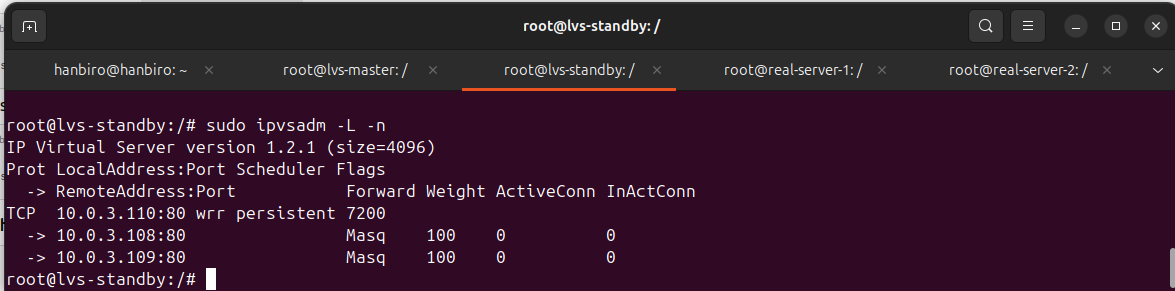
* run (on both lvs-master and lvs-standby):

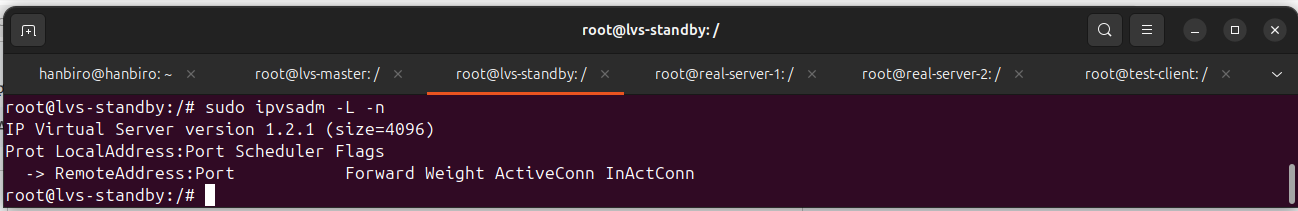
**sudo systemctl restart ldirectord**

**sudo systemctl restart heartbeat**

* Check LVS status with ipvsadm on LVS master/standby

sudo ipvsadm -L -n





***one of the important features of LVS system is failover – when one server has problem, VIP will be transferred to standby server (lvs-standby).***

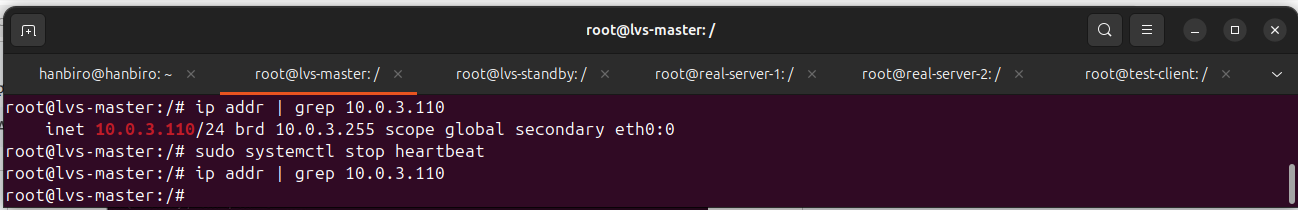
***this helps to ensure that service is not interrupted. We will perform failover and auto-failback test to make sure that the system will automatically transfer VIP to lvs-standby when needed and back to lvs-master when system recovers.***

[ **failover testing** ]

* On **LVS Master**, pause Heartbeat to simulate the problem

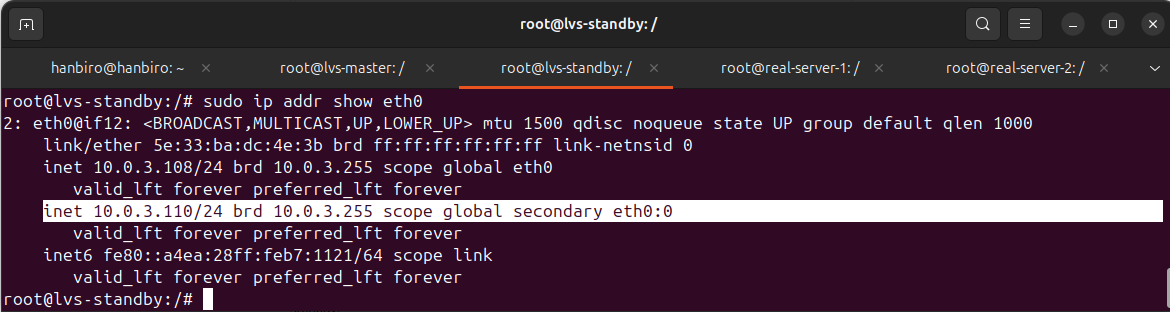
ip addr | grep 10.0.3.110

sudo systemctl stop heartbeat



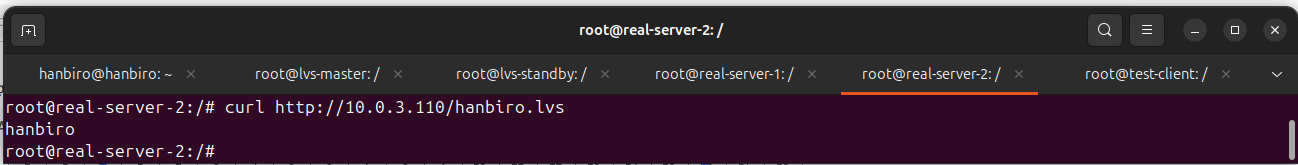
* Check if VIP 10.0.3.110 has switched to LVS Standby by checking the IP on **LVS Standby**

sudo ip addr show eth0



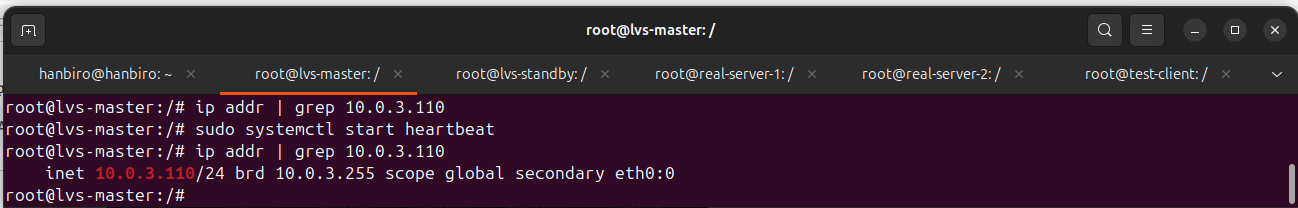
* Test VIP access on real-server-2 with VIP

curl http://10.0.3.110/hanbiro.lvs



* Restore Master and check VIP

sudo systemctl start heartbeat



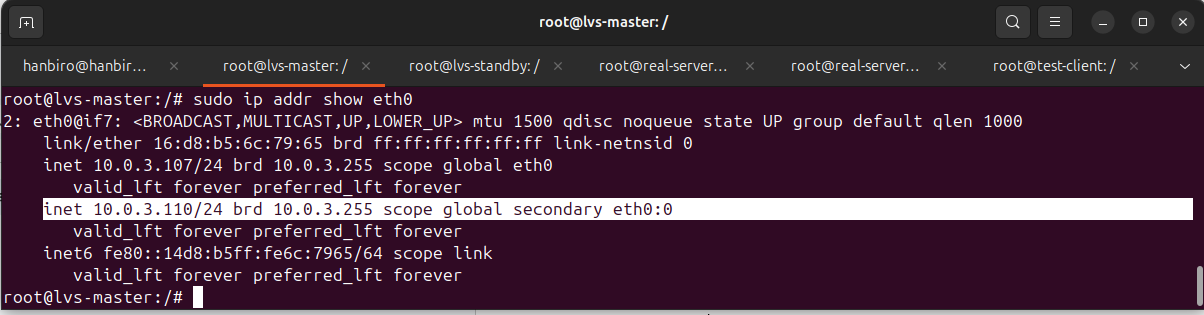
[ **Auto-failback testing** ]

* Restart Heartbeat on LVS Master:

sudo systemctl restart heartbeat

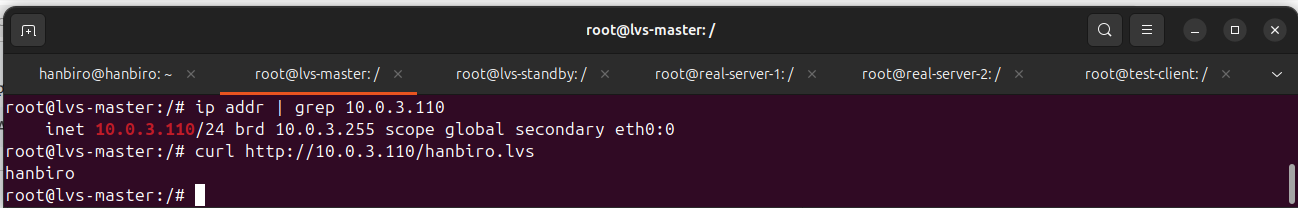
* Check if the VIP has been transferred back to LVS Master by checking the IP on LVS Master

sudo ip addr show eth0



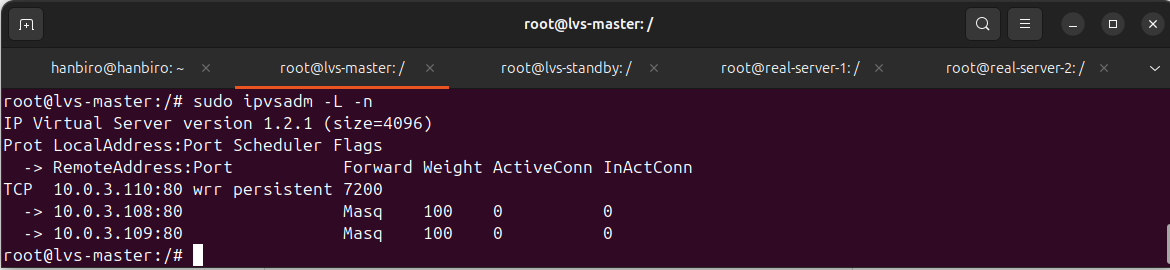
* Test HTTP connection over VIP on LVS Master

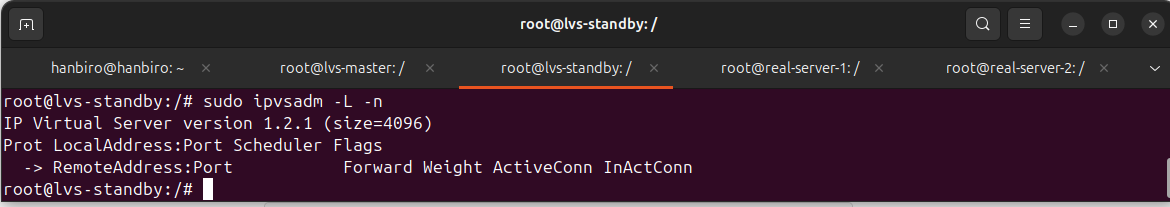
curl http://10.0.3.110/hanbiro.lvs



* Check LVS Table on LVS Master and LVS Standby:

sudo ipvsadm -L -n





***What is the difference negotiate and connect of ‘checktype’ options on ldirectord.cf?***

In the ldirectord.cf file, the checktype option determines how ldirectord checks the status of the real server

* **checktype=connect**: ldirectord only opens a TCP connection to the real server (ex: 10.0.3.109:80). If the connection is successful, the server is considered "UP". The return content is not checked

→ Suitable for services that only need an active port, such as MySQL

* **checktype=negotiate**: ldirectord sends an HTTP request (specified by request=...) and checks the return content (by receive=...). If the content does not match, the server is considered "DOWN"

→ Suitable for checking webservers (Apache, Nginx)

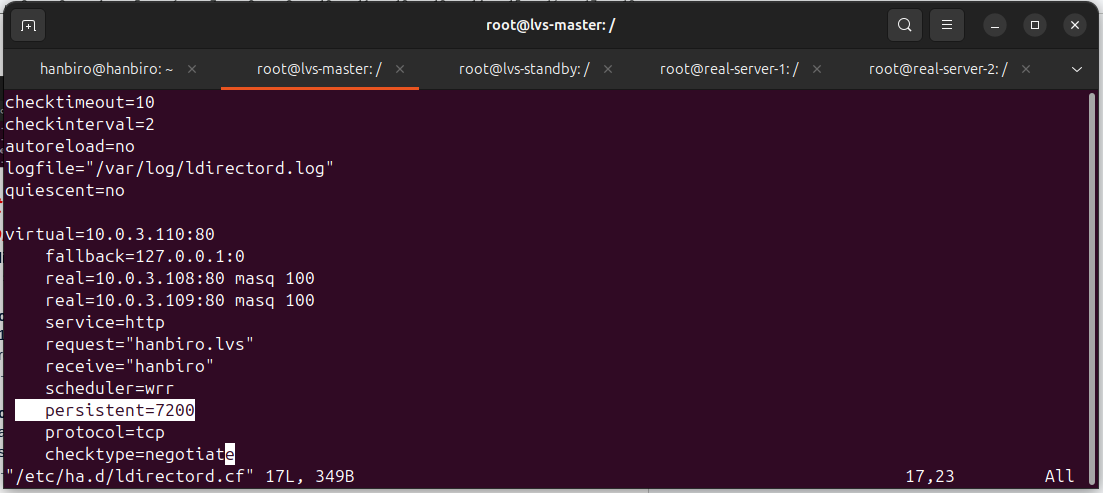
In the configuration, I used negotiate to check the "**hanbiro**" content returned from **/var/www/html/hanbiro.lvs** → This ensures that the web server is working properly, not simply "opening the port"

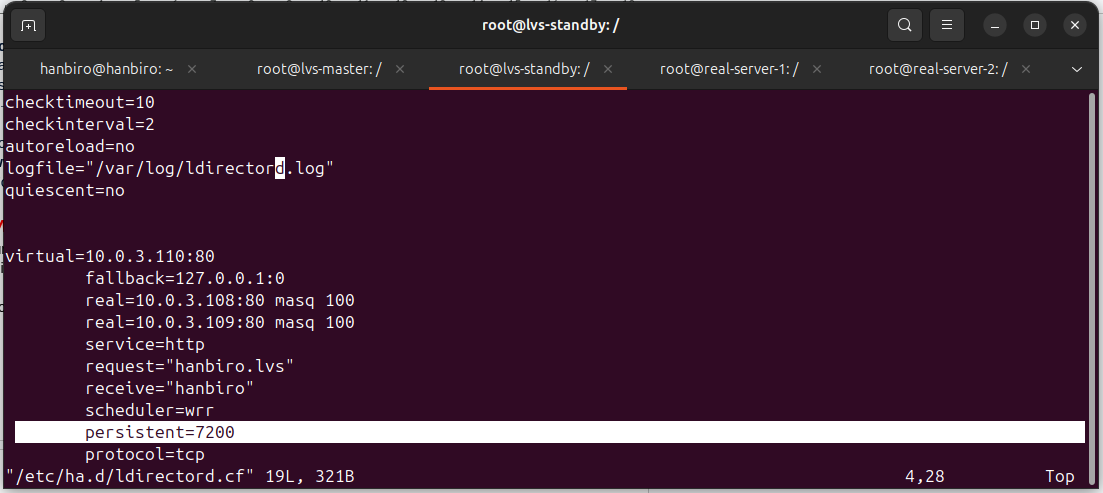
***The way of keep sessions alive ?***

To ensure that user sessions are persistent and not interrupted when accessing via LVS, the system is configured to use the session persistence feature of ldirectord.

In the configuration file **/etc/ha.d/ldirectord.cf**, the following option has been used:

**persistent = 7200**





=> this option allows LVS to remember the client's IP address and ensures that all connections from that client within 7200 seconds (2 hours) will always be routed to the same real server. This is especially necessary for web services with logins, shopping carts, or persistent sessions where state is required to not change between requests

***When is using fwm ?***

Firewall Mark (FWM) in LVS is a mechanism that allows marking packets and distributing them to real servers based on marks instead of specifying them by IP address or port

In LVS, using FWM, it is easy to distinguish traffic to different service groups. This is very useful when handling services with different requests such as HTTP, HTTPS, MySQL on the same Virtual IP (VIP).

Use FWM in the following cases:

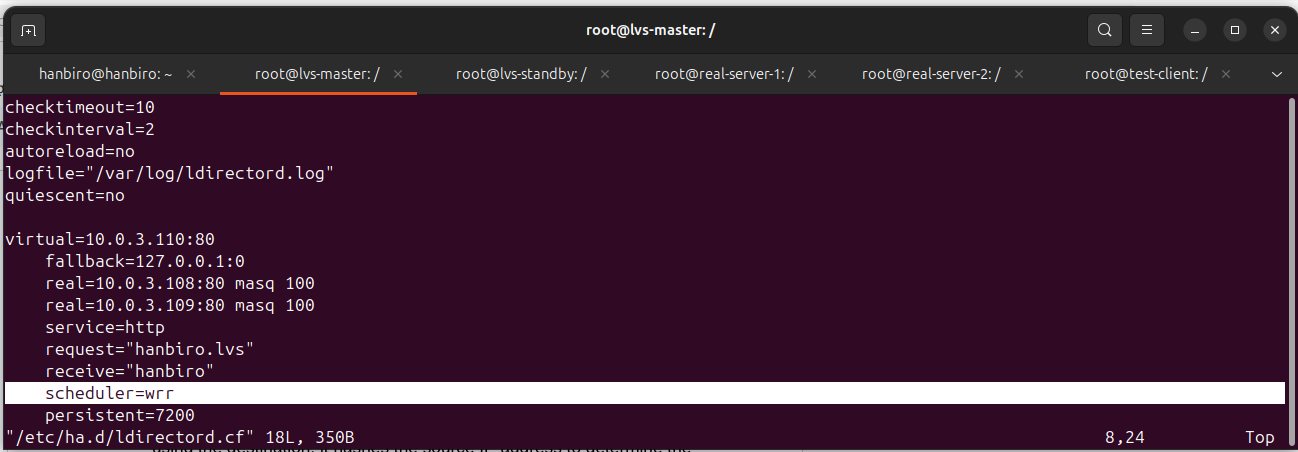
* **Traffic Classification:** When classifying network traffic to apply policies or route traffic through different paths
* **Load Balancing:** Use FWM to distribute load between servers based on specific criteria, such as distinguishing between HTTP and MySQL or different types of services
* **Mark Routing:** FWM can be used to assign different actions to packets based on a mark. For example, marked packets will be routed through a different gateway, or different QoS policies will be applied
* **Traffic Shaping:** When controlling bandwidth or prioritizing certain types of traffic (such as HTTP or DNS), use FWM in conjunction with tc or iptables to create traffic shaping rules
* **Fault Detection and Failover:** Use FWM to mark the traffic of servers and monitor them, switching traffic to other servers when there is a failure

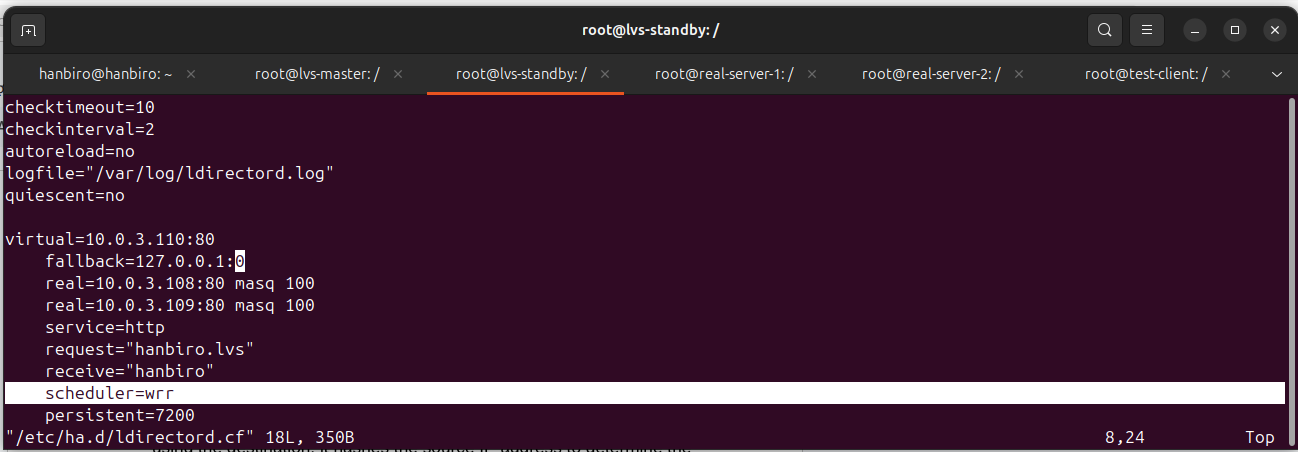
***Find the types of scheduler (such as lc, wlc, rr, wrr, etc..)***

LVS (Linux Virtual Server) supports various scheduling algorithms to distribute incoming traffic across real servers. Each algorithm has a different method for selecting which server should handle a given request.

* Round Robin (rr): this is the simplest method. It sends each new request to the next server in the list, looping back to the first server once the end of the list is reached. It works well when all servers have similar capabilities
* Weighted Round Robin (wrr): similar to Round Robin but with weights assigned to each server. Servers with higher weights will receive more traffic. This is useful when servers have different capacities, ensuring that stronger servers handle more requests
* Least Connections (lc): this scheduler sends requests to the server with the least number of active connections. It is suitable for services where connections may last a long time, such as database connections
* weighted Least Connections (wlc): combines Least Connections with weighted servers. Servers with fewer connections and higher weights will receive more requests. This helps balance the load while considering both the number of connections and the server’s capability
* Destination Hashing (dh): this method uses a hash function based on the destination address (ex: IP, …) to decide which server should handle the request. this ensures that requests to the same destination are consistently routed to the same server
* Source Hashing (sh): similar to Destination Hashing, but instead of using the destination, it hashes the source IP address to determine the server. This ensures that requests from the same client are routed to the same server
* Shortest Time (st): this scheduler sends requests to the server that takes the least amount of time to handle the request. It is ideal when response time is critical and ensures efficient use of server resources

I USED: **scheduler=wrr**





***set with port 80 and 3306***

**Install Apache and MySQL on Real Servers (real-server-1 and real-server-2)**

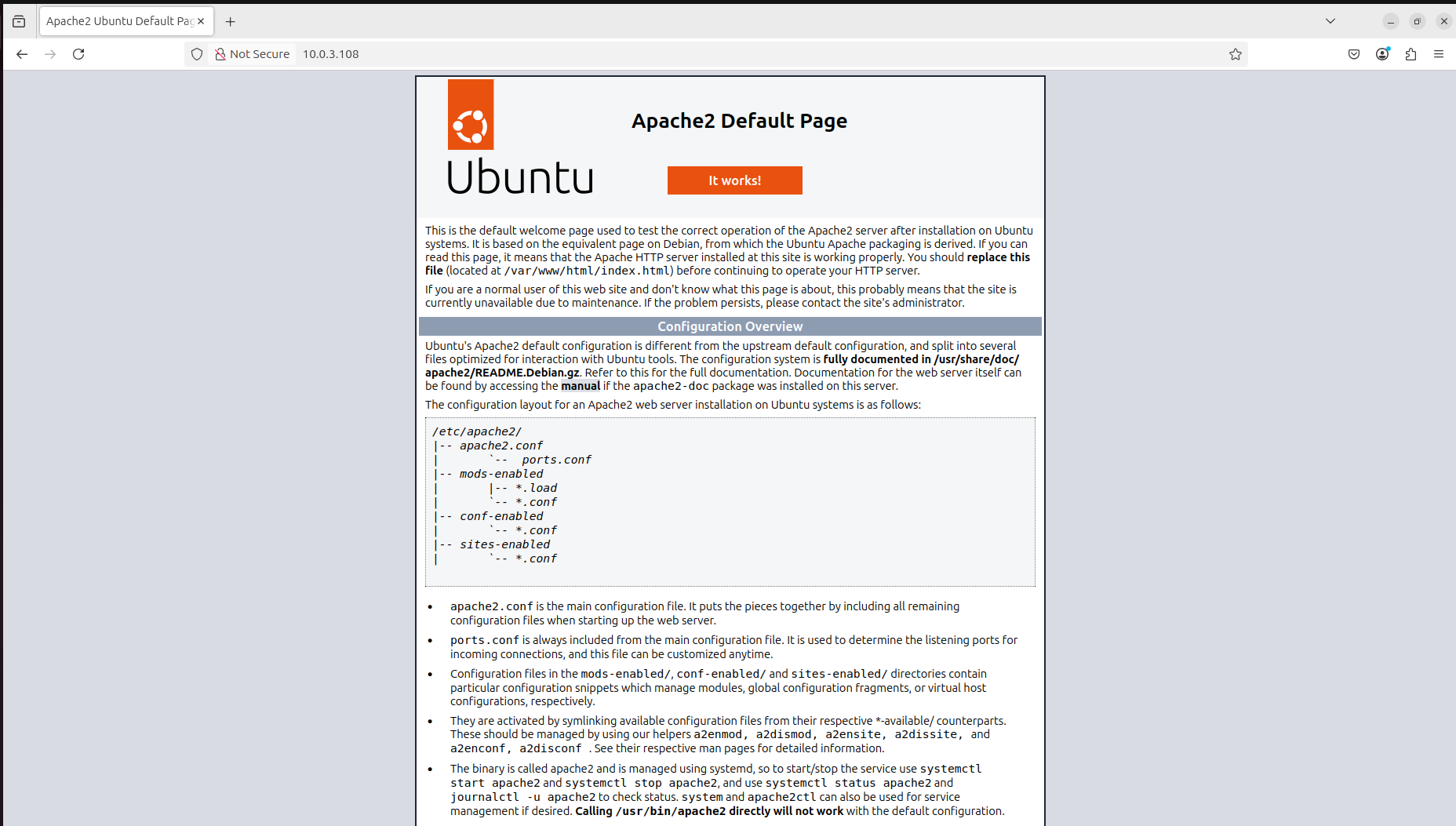
sudo apt update

sudo apt install -y apache2

sudo apt install -y mysql-server

* check Apache:

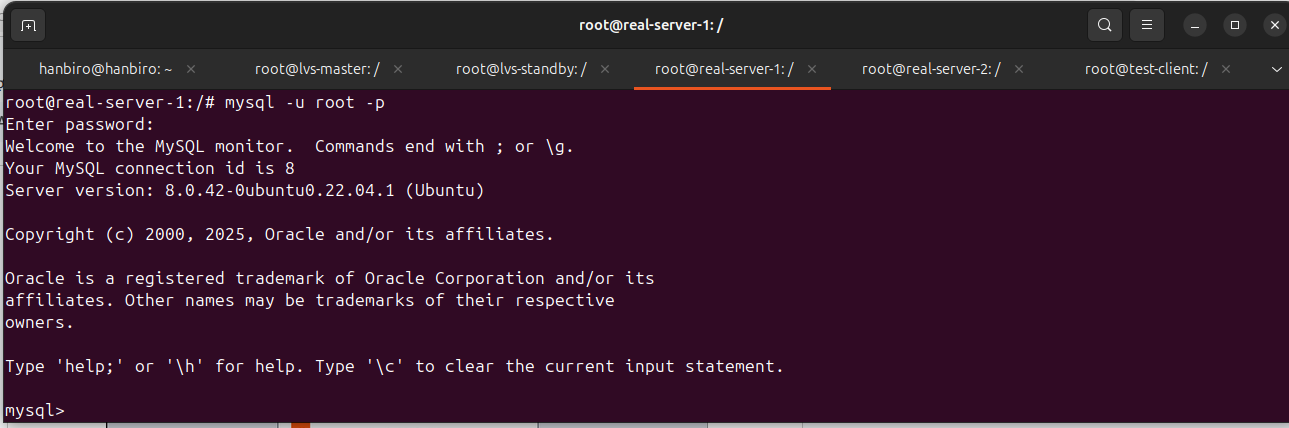
access **http://10.0.3.108/** and **http://10.0.3.109/** on browser

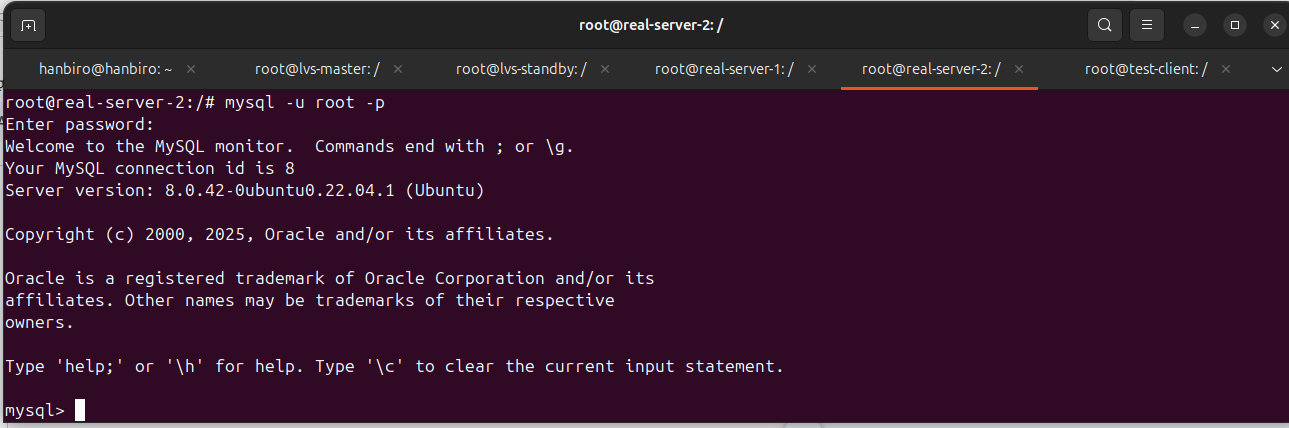




* check MySQL (root - **root**) on both real-server:

mysql -u root -p



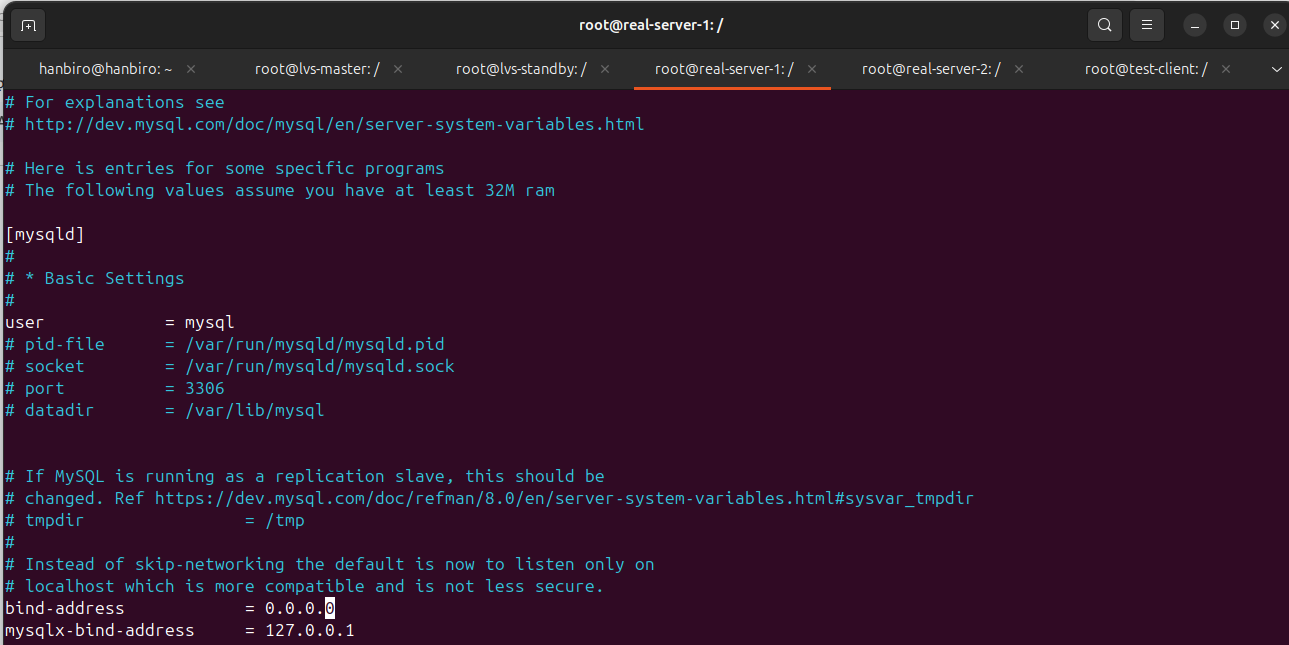


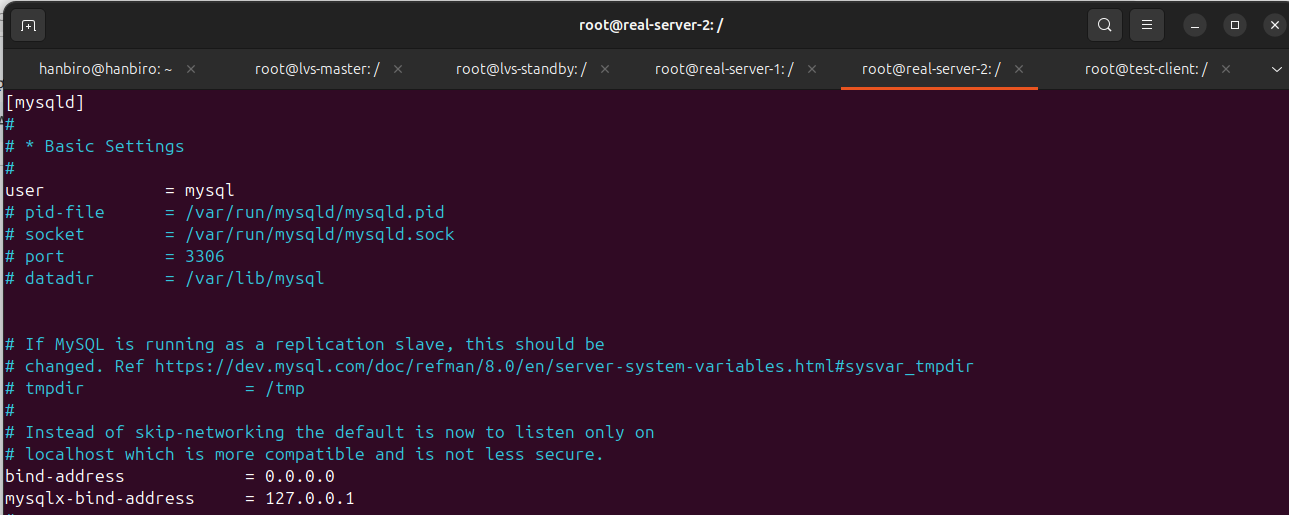
sudo vi /etc/mysql/mysql.conf.d/mysqld.cnf

**bind-address = 0.0.0.0**

\*accept connections from any IP

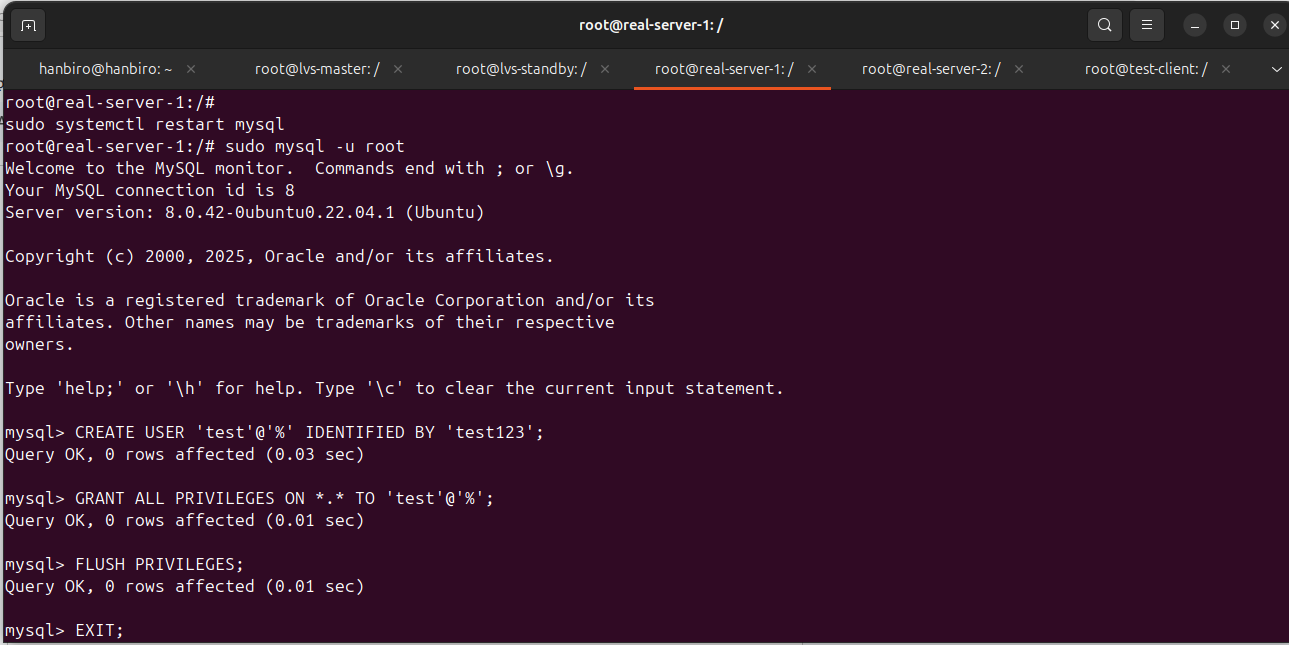
sudo systemctl restart mysql

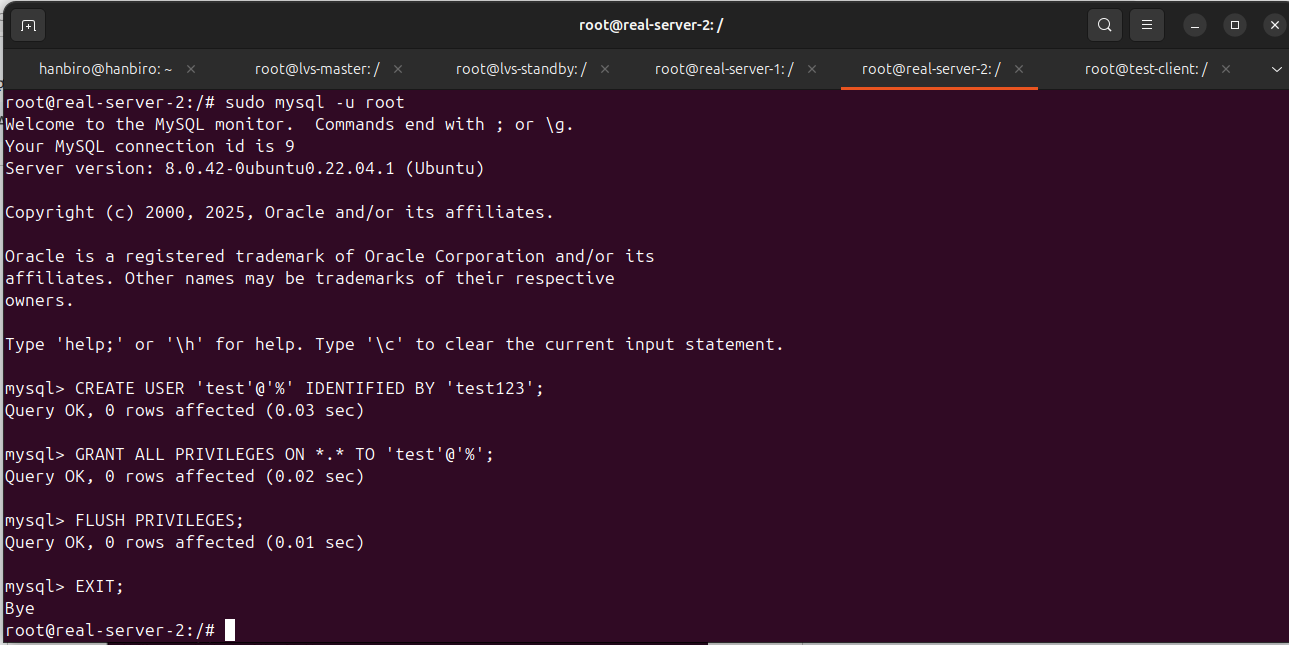




* access to mysql in both real-server:

sudo mysql -u root

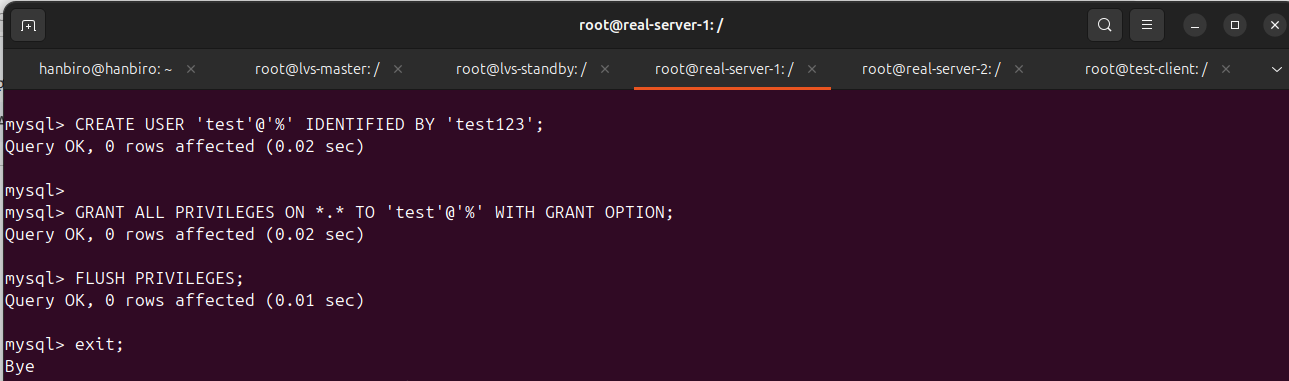


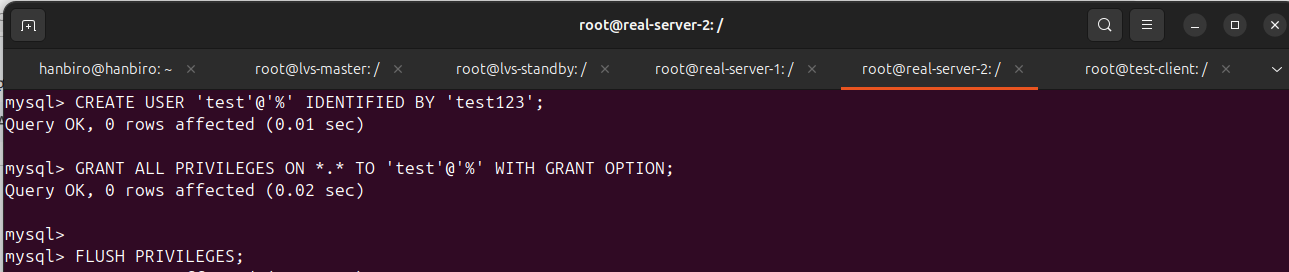


CREATE USER 'test'@'%' IDENTIFIED BY 'test123';

GRANT ALL PRIVILEGES ON \*.\* TO 'test'@'%' WITH GRANT OPTION;

FLUSH PRIVILEGES;





**Configure LVS on lvs-master and lvs-standby**

* access container lvs-master

sudo lxc-attach -n lvs-master

* Edit the ldirectord.cf configuration file

**sudo vi /etc/ha.d/ldirectord.cf**

checktimeout=10

checkinterval=2

autoreload=no

logfile="/var/log/ldirectord.log"

quiescent=no

virtual=10.0.3.110:80

real=10.0.3.108:80 masq 100

real=10.0.3.109:80 masq 100

service=http

request="hanbiro.lvs"

receive="hanbiro"

scheduler=wrr

persistent=7200

protocol=tcp

checktype=negotiate

**virtual=10.0.3.110:3306**

**real=10.0.3.108:3306 masq 100**

**real=10.0.3.109:3306 masq 100**

**service=mysql**

**request="SELECT 1"**

**receive="1"**

**scheduler=wrr**

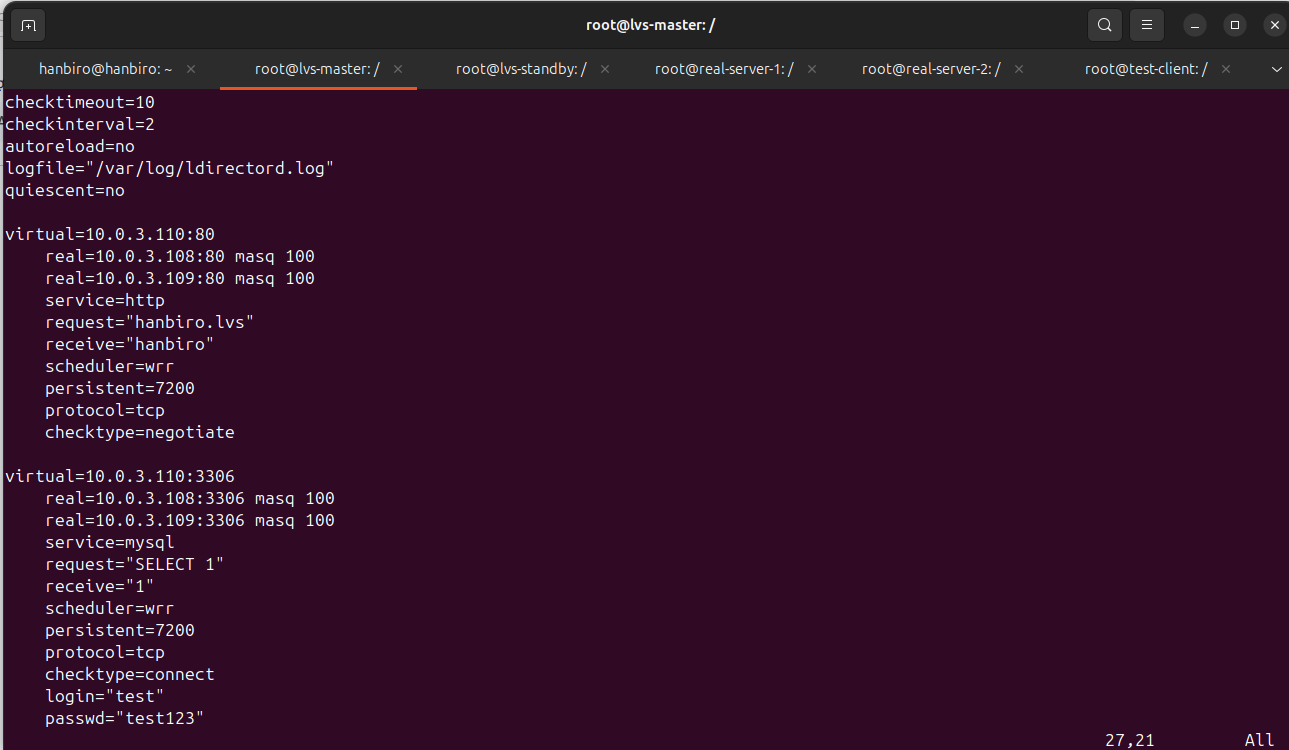
**persistent=7200**

**protocol=tcp**

**checktype=connect**

**login="test"**

**passwd="test123"**



\*with:

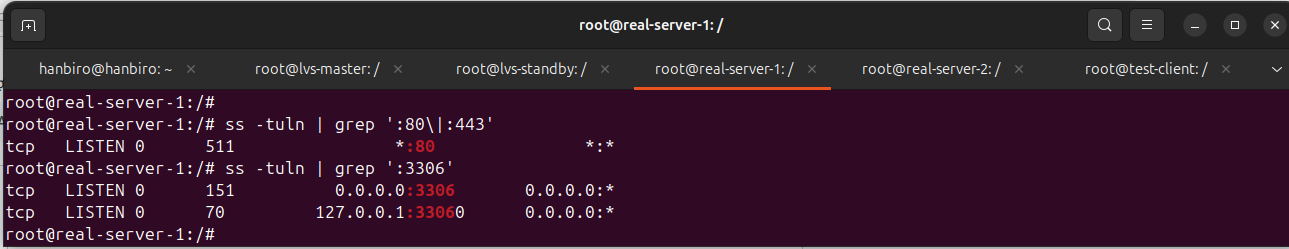
* **virtual=10.0.3.110:3306**: Virtual IP address for MySQL port
* **real=10.0.3.108:3306** and **real=10.0.3.109:3306**: Real servers providing MySQL service
* **scheduler=wrr**: Use Weighted Round Robin to distribute traffic
* **checktype=connect:** How to check the real server by trying to connect (instead of checking the protocol or content)
* **request="SELECT 1"**: Send a simple SQL statement SELECT 1 to check the MySQL connection
* **receive="1"**: The expected response from the MySQL server is 1, i.e. the result of the SELECT 1 statement
* **login="test"**: MySQL login name
* **passwd="test123"**: Password for the MySQL account
* Restart the ldirectord service:

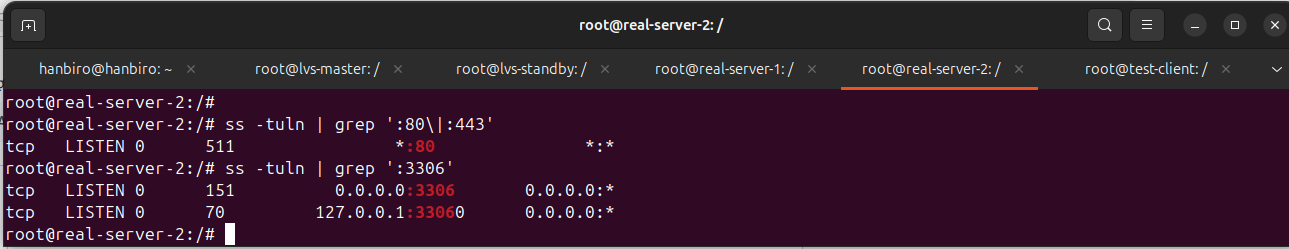
sudo systemctl restart ldirectord

* check port 80 and 3306:

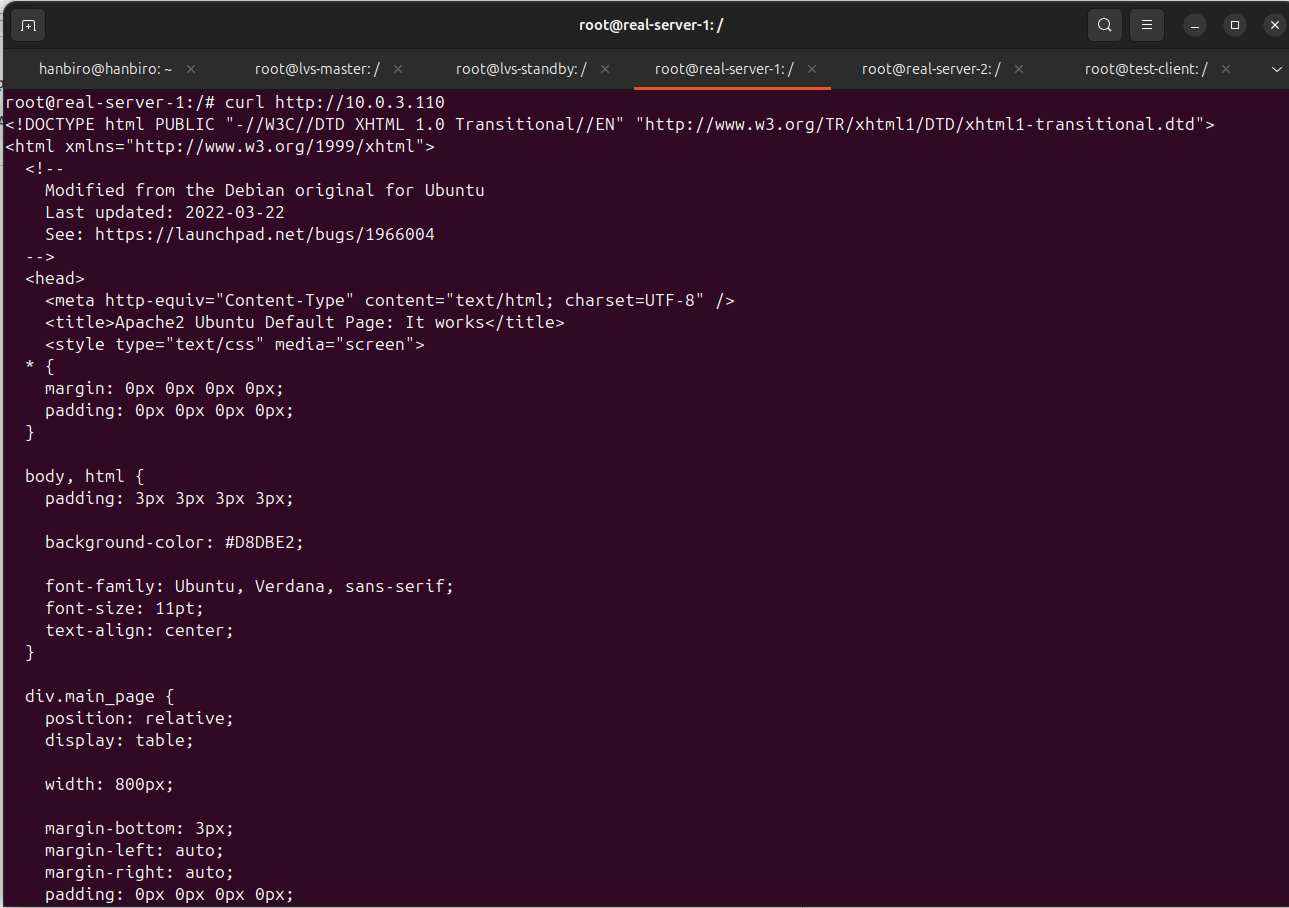
ss -tuln | grep ':80\|:443'

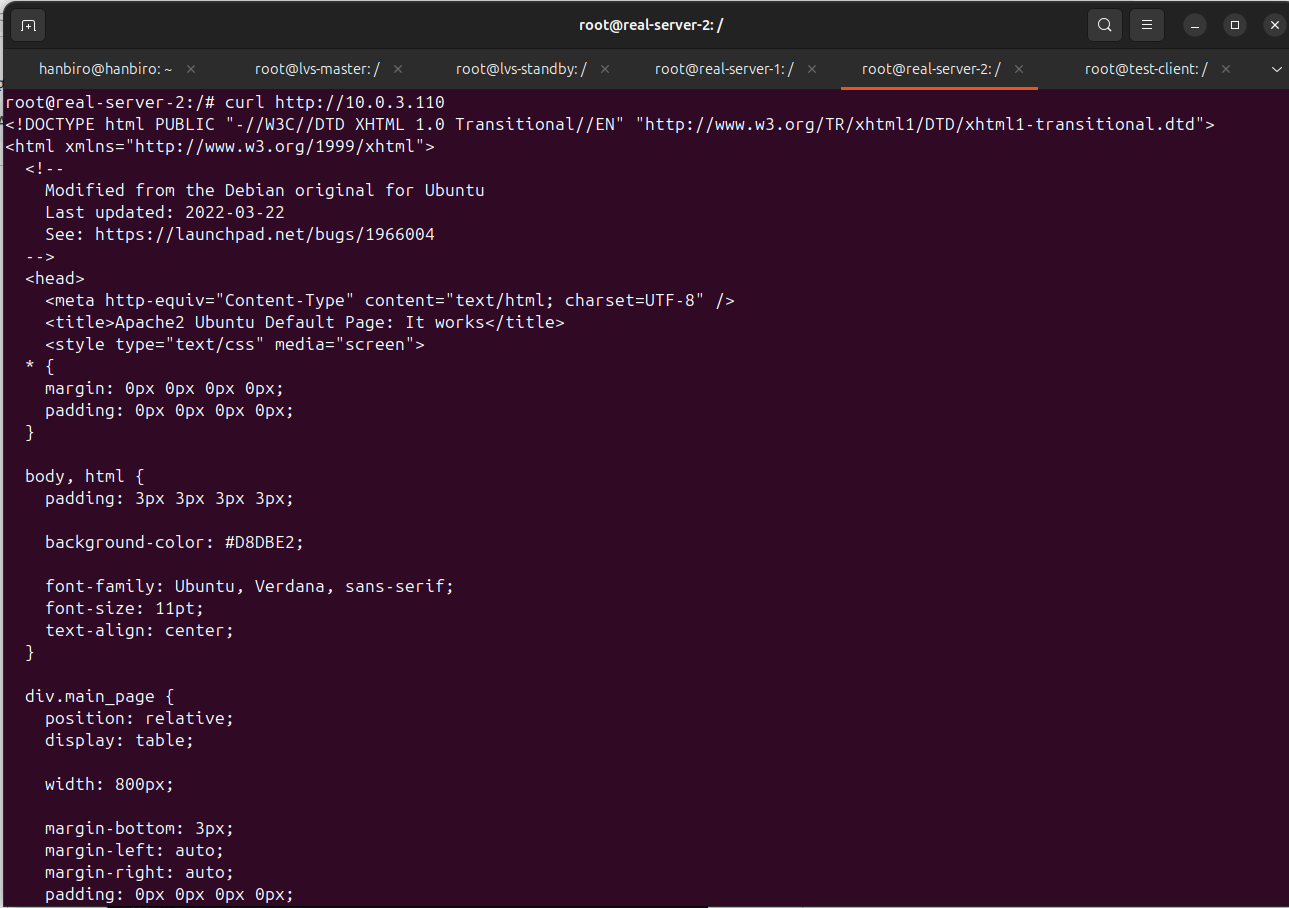
ss -tuln | grep ':3306'





* check connect port 80: **curl http://10.0.3.110** (VIP)





* check connect port 3306:

mysql -u test -p -h 10.0.3.110 -P 3306

