Cluster

- HaProxy
- DRBD + Pacemaker & Corosync NFS Cluster Centos7
- Keepalived LoadBalacing
- Distributed memcached on 2 Webserver [CentOS7]
- GlusterFS + Heketi [Ubuntu 18.04]

HaProxy

> This is not a tutorial of how haproxy works, this is just some notes on a config i did, and some of the options i used that made it stable for what i needed.

In the example bellow you will find a acceptable cipher, how to add a cookie sessions on HA, SSL offloading, xforward's, ha stats, good timeout vaules, and a httpchk.

```
global
        log 127.0.0.1 local0 warning
        maxconn 10000
        user haproxy
        group haproxy
        daemon
        spread-checks 5
        tune.ssl.default-dh-param 2048
        ssl-default-bind-ciphers ECDHE-RSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-RSA-AES256-GCM-SHA384:ECDHE-ECDSA-AES256-GCM-SHA384:DHE-RSA-AES128-GCM-SHA256:DHE-DSS-AES128-GCM-SHA256:kEDH+AESGCM:ECDHE-RSA-AES128-SHA256:ECDHE-ECDSA-AES128-SHA256:ECDHE-RSA-AES128-SHA:ECDHE-ECDSA-AES128-SHA:ECDHE-RSA-AES256-SHA384:ECDHE-ECDSA-AES256-SHA384:ECDHE-RSA-AES256-SHA:ECDHE-ECDSA-AES256-SHA:DHE-RSA-AES128-SHA256:DHE-RSA-AES128-SHA:DHE-DSS-AES128-SHA256:DHE-RSA-AES256-SHA256:DHE-DSS-AES256-SHA:DHE-RSA-AES256-SHA:AES128-GCM-SHA256:AES256-GCM-SHA384:AES128-SHA256:AES256-SHA256:AES128-SHA:AES256-SHA:AES:CAMELLIA:DES-CBC3-SHA:!aNULL:!eNULL:!EXPORT:!DES:!RC4:!MD5:!PSK:!aECDH:!EDH-DSS-DES-CBC3-SHA:!EDH-RSA-DES-CBC3-SHA:!KRB5-DES-CBC3-SHA

defaults
        log     global
        option  dontlognull
        retries 3
        option redispatch
        maxconn 10000
        mode http
        option dontlognull
        option httpclose
        option httpchk
        timeout connect 5000ms
```

```
        timeout client 150000ms
        timeout server 30000ms
        timeout check 1000

listen  lb_stats
        bind    {PUBLIC IP}:80
        balance roundrobin
        server  lb1 127.0.0.1:80
        stats   uri /
        stats   realm "HAProxy Stats"
        stats   auth admin:FsoqyNpJAYuD

frontend frontend_{PUBLIC IP}_https
       mode □□  tcp
       bind            {PUBLIC IP}:443 ssl crt /etc/haproxy/ssl/domain.com.pem no-sslv3
       reqadd X-Forwarded-Proto:\ https
       http-request add-header X-CLIENT-IP %[src]
       option          forwardfor
       default_backend backend_cluster_http_web1_web2

frontend frontend_{PUBLIC IP}_http
       bind            {PUBLIC IP}:80
       reqadd X-Forwarded-Proto:\ https
       http-request add-header X-CLIENT-IP %[src]
       option          forwardfor
       default_backend backend_cluster_http_web1_web2

frontend frontend_www_custom
       bind            {PUBLIC IP}:666
       option          forwardfor
       default_backend backend_cluster_http_web1_web2

 backend backend_cluster_http_web1_web2
        option httpchk HEAD /
        server  web1 10.1.2.100:80 weight 1 check cookie web1 inter 1000 rise 5 fall 1
        server  web2 10.1.2.101:80 weight 1 check cookie web2 inter 1000 rise 5 fall 1
```

Enable xforward on httpd.conf on the web servers

```
LogFormat "%{X-Forwarded-For}i %h %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-Agent}i\ " combine
LogFormat "%{X-Forwarded-For}i %h %l %u %t \"%r\" %s %b \"%{Referer}i\" \"%{User-agent}i\"" combined-forwarded
```

Cookie

It is also possible to use the session cookie provided by the backend server.

```
backend www
        balance roundrobin
        mode http
        cookie PHPSESSID prefix indirect nocache
        server web1 10.1.2.100:80 check cookie web1
        server web2 10.1.2.101:80 check cookie web2
```

In this example we will intercept the PHP session cookie and add / remove the reference of the backend server.

The prefix keyword allows you to reuse an application cookie and prefix the server identifier, then delete it in the following queries.

Default name of cookies by type of feeder backend:
Java : JSESSIONID
ASP.Net : ASP.NET_SessionId
ASP : ASPSESSIONID
PHP : PHPSESSID

Active/Passive config

```
backend backend_web1_primary
        option httpchk HEAD /
        server  web1 10.1.2.100:80 check
        server  web2 10.1.2.101:80 check backup

backend backend_web2_primary
        option httpchk HEAD /
        server  web2 10.1.2.100:80 check
        server  web1 10.1.2.101:80 check backup
```

Test config file:

```
haproxy -c -V -f /etc/haproxy/haproxy.cfg
```

Hapee Check syntax

```
/opt/hapee-1.7/sbin/hapee-lb -c
```

Hapee VRRP

```
# /etc/hapee-1.7/hapee-vrrp.cfg

vrrp_script chk_hapee {
    script "pidof hapee-lb"
    interval 2
}

vrrp_instance vrrp_1 {
  interface eth0
  virtual_router_id 51
  priority 101
  virtual_ipaddress_excluded {
        eth0
        eth1
  }
  track_interface {
        eth0 weight -2
        eth1 weight -2
  }
  track_script {
        chk_hapee
  }
}

vrrp_instance vrrp_2 {
  interface eth1
  virtual_router_id 51
  priority 101
```

```
  virtual_ipaddress_excluded {
        X.X.X.X
  }
  track_interface {
        eth0 weight -2
        eth1 weight -2
  }
  track_script {
        chk_hapee
  }
}
```

Doc

https://cbonte.github.io/haproxy-dconv/

DRBD + Pacemaker & Corosync NFS Cluster Centos7

On Both Nodes

Host file

```
vim /etc/hosts
```

> 10.1.2.114 nfs1 nfs1.localdomain.com
> 10.1.2.115 nfs2 nfs2.localdomain.com

Corosync will not work if you add something like this: ***127.0.0.1 nfs1 nfs2.localdomain.com*** - however you do not need to delete 127.0.0.1 localhost

Firewall

Option 1 ***Firewalld***

```
systemctl start firewalld
systemctl enable firewalld
firewall-cmd --permanent --add-service=nfs
firewall-cmd --permanent --add-service=rpc-bind
firewall-cmd --permanent --add-service=mountd
firewall-cmd --permanent --add-service=high-availability
```

On ***NFS1***

```
firewall-cmd --permanent --add-rich-rule='rule family="ipv4" source address="10.1.2.115" port port="7789" protocol="tcp" accept'
```

```
firewall-cmd --reloadfirewall-cmd --reload
```

*On **NFS2***

```
firewall-cmd --permanent --add-rich-rule='rule family="ipv4" source address="10.1.2.114" port port="7789" protocol="tcp" accept'
firewall-cmd --reloadfirewall-cmd --reload
```

Disable SELINUX

```
vim /etc/sysconfig/selinux
```

> SELINUX=disabled

Pacemaker Install

Install PaceMaker and Corosync

```
yum install -y pacemaker pcs
```

Authenticate as the hacluster user

```
echo "H@xorP@assWD" | passwd hacluster --stdin
```

Start and enable the service

```
systemctl start pcsd
systemctl enable pcsd
```

ON NFS1

Test and generate the Corosync configuration

```
pcs cluster auth nfs1 nfs2 -u hacluster -p H@xorP@assWD
```

```
pcs cluster setup --start --name mycluster nfs1 nfs2
```

ON BOTH NODES

Start the cluster

```
systemctl start corosync
systemctl enable corosync
pcs cluster start --all
pcs cluster enable --all
```

Verify Corosync installation

> Master should have ID 1 and slave ID 2

```
corosync-cfgtool -s
```

> **ON NFS1**

Create a new cluster configuration file

```
pcs cluster cib mycluster
```

Disable the Quorum & STONITH policies in your cluster configuration file

```
pcs -f /root/mycluster property set no-quorum-policy=ignore
pcs -f /root/mycluster property set stonith-enabled=false
```

Prevent the resource from failing back after recovery as it might increases downtime

```
pcs -f /root/mycluster resource defaults resource-stickiness=300
```

LVM partition setup

> **Both Nodes**

Create a empty partition

```
fdisk /dev/sdb
```

> Welcome to fdisk (util-linux 2.23.2).

> Command (m for help): **n**
> Partition type:
> p primary (0 primary, 0 extended, 4 free)
> e extended
> Select (default p):**(ENTER)**
> Partition number (1-4, default 1): **(ENTER)**
> First sector (2048-16777215, default 2048): **(ENTER)**
> Using default value 2048
> Last sector, +sectors or +size{K,M,G} (2048-16777215, default 16777215):
> **(ENTER)**
> Using default value 16777215
> Partition 1 of type Linux and of size 8 GiB is set
>
> Command (m for help): **w**
> The partition table has been altered!

Create LVM partition

```
pvcreate /dev/sdb1
vgcreate vg00 /dev/sdb1
lvcreate -l 95%FREE -n drbd-r0 vg00
```

View LVM partition after creation

```
pvdisplay
```

Look in "/dev/mapper/" find the name of your LVM disk

```
ls /dev/mapper/
```

OUTPUT:

```
control vg00-drbd--r0
```

> **You will use "vg00-drbd--r0" in the "drbd.conf" file in the below steps

DRBD Installation

Install the DRBD package

```
rpm --import https://www.elrepo.org/RPM-GPG-KEY-elrepo.org
rpm -Uvh http://www.elrepo.org/elrepo-release-7.0-3.el7.elrepo.noarch.rpm
```

```
yum install -y kmod-drbd84 drbd84-utils
modprobe drbd
echo drbd > /etc/modules-load.d/drbd.conf
```

Edit the DRBD config and add the to hosts it will be connecting to (NFS1 and NFS2)

```
vim /etc/drbd.conf
```

> Delete all and replace for the following

```
include "drbd.d/global_common.conf";
include "drbd.d/*.res";

global {
usage-count no;
}
resource r0 {
protocol C;
startup {
degr-wfc-timeout 60;
outdated-wfc-timeout 30;
wfc-timeout 20;
}
disk {
on-io-error detach;
}
net {
cram-hmac-alg sha1;
shared-secret "Daveisc00l123313";
}
on nfs1.localdomain.com {
device /dev/drbd0;
disk /dev/mapper/vg00-drbd--r0;
address 10.1.2.114:7789;
meta-disk internal;
}
on nfs2.localdomain.com {
device /dev/drbd0;
disk /dev/mapper/vg00-drbd--r0;
address 10.1.2.115:7789;
meta-disk internal;
}
```

```
}
```

```
vim /etc/drbd.d/global_common.conf
```

Delete all and replace for the following

```
common {
        handlers {
        }
        startup {
        }
        options {
        }
        disk {
        }
        net {
                after-sb-0pri discard-zero-changes;
                after-sb-1pri discard-secondary;
                after-sb-2pri disconnect;
         }
}
```

On NFS1

Create the DRBD partition and assign it primary on NFS1

```
drbdadm create-md r0
drbdadm up r0
drbdadm primary r0 --force
drbdadm -- --overwrite-data-of-peer primary all
drbdadm outdate r0
mkfs.ext4 /dev/drbd0
```

On NFS2

Configure r0 and start DRBD on NFS2

```
drbdadm create-md r0
drbdadm up r0
drbdadm secondary all
```

Pacemaker cluster resources

> **On NFS1**

Add resource r0 to the cluster resource

```
pcs -f /root/mycluster resource create r0 ocf:linbit:drbd drbd_resource=r0 op monitor interval=10s
```

Create an additional clone resource r0-clone to allow the resource to run on both nodes at the same time

```
pcs -f /root/mycluster resource master r0-clone r0 master-max=1 master-node-max=1 clone-max=2 clone-node-max=1 notify=true
```

Add DRBD filesystem resource

```
pcs -f /root/mycluster resource create drbd-fs Filesystem device="/dev/drbd0" directory="/data" fstype="ext4"
```

Filesystem resource will need to run on the same node as the r0-clone resource, since the pacemaker cluster services that runs on the same node depend on each other we need to assign an infinity score to the constraint:

```
pcs -f /root/mycluster constraint colocation add drbd-fs with r0-clone INFINITY with-rsc-role=Master
```

Add the Virtual IP resource

```
pcs -f /root/mycluster resource create vip1 ocf:heartbeat:IPaddr2 ip=10.1.2.116 cidr_netmask=24 op monitor interval=10s
```

The VIP needs an active filesystem to be running, so we need to make sure the DRBD resource starts before the VIP

```
pcs -f /root/mycluster constraint colocation add vip1 with drbd-fs INFINITY
pcs -f /root/mycluster constraint order drbd-fs then vip1
```

Verify that the created resources are all there

```
pcs -f /root/mycluster resource show
pcs -f /root/mycluster constraint
```

And finally commit the changes

```
pcs cluster cib-push mycluster
```

> **On Both Nodes**

Installing NFS

Install nfs-utils

```
yum install nfs-utils -y
```

Stop all services

```
systemctl stop nfs-lock &&  systemctl disable nfs-lock
```

Setup service

```
pcs -f /root/mycluster resource create nfsd nfsserver nfs_shared_infodir=/data/nfsinfo
pcs -f /root/mycluster resource create nfsroot exportfs clientspec="10.1.2.0/24"
options=rw,sync,no_root_squash directory=/data fsid=0
pcs -f /root/mycluster constraint colocation add nfsd with vip1 INFINITY
pcs -f /root/mycluster constraint colocation add vip1 with nfsroot INFINITY
pcs -f /root/mycluster constraint order vip1 then nfsd
pcs -f /root/mycluster constraint order nfsd then nfsroot
pcs -f /root/mycluster constraint order promote r0-clone then start drbd-fs
pcs resource cleanup
pcs cluster cib-push mycluster
```

Test failover

```
pcs resource move drbd-fs nfs2
```

Other notes on DRBD

To update a resource after a commit

```
cibadmin --query > tmp.xml
```

> Edit with vi tmp.xml or do a pcs -f tmp.xml %do your thing%

```
cibadmin --replace --xml-file tmp.xml
```

Delete a resource

```
pcs -f /root/mycluster resource delete db
```

Delete cluster

```
pcs cluster destroy
```

Recover a split brain

Secondary node
drbdadm secondary all
drbdadm disconnect all
drbdadm -- --discard-my-data connect all

Primary node
drbdadm primary all
drbdadm disconnect all
drbdadm connect all

On both
drbdadm status
cat /proc/drbd

Keepalived LoadBalacing

LVS Config

```
## Pool ID
virtual_server <WAN "frontend" IP> 80 {
        delay_loop 6
        lb_algo sh      # source hash
        lb_kind NAT
        protocol TCP

        real_server <LAN "backend" IP Server 1> 80 {
                weight 1
                TCP_CHECK {
                        connect_timeout 3
                }
        }
        real_server <LAN "backend" IP Server 2> 80 {
                weight 1
                TCP_CHECK {
                        connect_timeout 3
                }
        }
}

virtual_server <WAN "frontend" IP> 443 {
        delay_loop 6
        lb_algo sh      # source hash
        lb_kind NAT
        protocol TCP

        real_server <LAN "backend" IP Server 1> 443 {
                weight 1
                TCP_CHECK {
                        connect_timeout 3
                }
        }
```

```
        real_server <LAN "backend" IP Server 2> 443 {
                weight 1
                TCP_CHECK {
                        connect_timeout 3
                }
        }
}
```

VRRP

```
vrrp_instance VI_LOCAL {
        state MASTER
        interface eth1
        virtual_router_id 51
        priority 101
        virtual_ipaddress {
                10.X.X.X
        }

        track_interface {
                eth0
                eth1
        }

}

vrrp_instance VI_PUB {
        state MASTER
        interface eth0
        virtual_router_id 52
        priority 101
        virtual_ipaddress {
                X.X.X.X
        }
        track_interface {
                eth0
                eth1
        }
}
```

```
vrrp_instance VI_PUB2 {
        state MASTER
        interface eth0
        virtual_router_id 53
        priority 101
        virtual_ipaddress {
                X.X.X.X
        }

        track_interface {
                eth0
                eth1
}
}
```

sysctl

```
# Use ip that are not configured locally (HAProxy + KeepAlived requirements)
net.ipv4.ip_nonlocal_bind = 1

# Enable packet forwarding
net.ipv4.ip_forward=1

# Disables IP source routing
net.ipv4.conf.all.accept_source_route = 0

# Enable IP spoofing protection, turn on source route verification
net.ipv4.conf.all.rp_filter = 1

# Disable ICMP Redirect Acceptance
net.ipv4.conf.all.accept_redirects = 0

# Decrease the time default value for tcp_fin_timeout connection
net.ipv4.tcp_fin_timeout = 15

# Decrease the time default value for tcp_keepalive_time connection
net.ipv4.tcp_keepalive_time = 1800

# Enable TCP SYN Cookie Protection
```

```
net.ipv4.tcp_syncookies = 1

# Enable ignoring broadcasts request
net.ipv4.icmp_echo_ignore_broadcasts = 1

# Enable bad error message Protection
net.ipv4.icmp_ignore_bogus_error_responses = 1

# Log Spoofed Packets, Source Routed Packets, Redirect Packets
net.ipv4.conf.all.log_martians = 1

# Increases the size of the socket queue
net.ipv4.tcp_max_syn_backlog = 1024

# Increase the tcp-time-wait buckets pool size
net.ipv4.tcp_max_tw_buckets = 1440000

# Arp
net.ipv4.conf.lo.rp_filter = 0
net.ipv4.conf.all.arp_announce = 2
net.ipv4.conf.all.arp_ignore = 1
```

DR

```
vim /etc/modules
```

> iptable_mangle
> xt_multiport
> xt_MARK
> ip_vs
> ip_vs_rr
> ip_vs_nq
> ip_vs_wlc

```
${IPTABLES} -t mangle -A PREROUTING -p tcp -d <VIP-WAN>/32 -j MARK --set-mark 0x1
```

Keepalived

```
virtual_server fwmark 1 {
        delay_loop 10
        lb_algo lc
        lb_kind DR
        protocol TCP
persistence_timeout 28800

        real_server <WAN-WEB1> 0 {
                weight 1
                TCP_CHECK {
connect_port 443
                        connect_timeout 3
                }
        }
        real_server <WAN-WEB2> 0 {
                weight 2
                TCP_CHECK {
connect_port 443
                        connect_timeout 3
                }
        }
}
```

Distributed memcached on 2 Webserver [CentOS7]

Install memcached

```
yum install memcached libmemcached -y
```

```
vi /etc/sysconfig/memcached
```

Change options to listen to the private IP on both web's:

```
OPTIONS="-l 10.1.1.X -U 0"
```

Restart memcached

```
systemctl restart memcached
systemctl enable memcached
```

Edit php ini

```
vi /etc/php.ini
session.save_handler = memcache
session.save_path = "tcp://10.1.1.100:11211, tcp://10.1.1.101:11211"
```

Install php-pecl-memcache

```
yum -y install php-pecl-memcache
echo "extension=memcache.so" >> /etc/php.d/memcache.ini
systemctl restart httpd
```

Allow in FW

```
firewall-cmd --zone=public --permanent --add-port=11211/tcp
firewall-cmd --reload
```

Check if memcached is running

```
watch memcached-tool X.X.X.X stats
```

Create test page:

```
vim /home/www/domain.com/session.php
```

Test Page:

```
<?php
header('Content-Type: text/plain');
session_start();
if(!isset($_SESSION['visit']))
{
echo "Page to test memcache.\n";
$_SESSION['visit'] = 0;
}
else
echo "You have visited this server ".$_SESSION['visit'] . " times. \n";
$_SESSION['visit']++;
echo "Server IP: ".$_SERVER['SERVER_ADDR'] . "\n";
echo "Client IP: ".$_SERVER['REMOTE_ADDR'] . "\n";
print_r($_COOKIE);
?>
```

GlusterFS + Heketi [Ubuntu 18.04]

> Requirement to this guide : Having an empty / unused partition available for configuration on all bricks. Size does not really matter, but it needs to be the same on all nodes.

Configuring your nodes

Configuring your **/etc/hosts** file :

```
## on gluster00 :
127.0.0.1 localhost localhost.localdomain glusterfs00
10.1.1.3 gluster01
10.1.1.4 gluster02

## on gluster01
127.0.0.1 localhost localhost.localdomain glusterfs01
10.1.1.2 gluster00
10.1.1.4 gluster02

## on gluster02
127.0.0.1 localhost localhost.localdomain glusterfs02
10.1.1.2 gluster00
10.1.1.3 gluster01
```

Installing glusterfs-server on your bricks (data nodes). In this example, on gluster00 and gluster01 :

```
apt update
apt upgrade
apt-get install software-properties-common
add-apt-repository ppa:gluster/glusterfs-7
apt-get install glusterfs-server
```

Enable/Start GLuster

```
systemctl enable glusterd
systemctl start glusterd
```

Connect on either node peer with the second host. In this example I'm connected on gluster00 and allow peer on the other hosts using the hostname :

```
gluster peer probe gluster01
```

Should give you something like this :

```
Number of Peers: 1

Hostname: gluster01
Uuid: 6474c4e6-2957-4de7-ac88-d670d4eb1320
State: Peer in Cluster (Connected)
```

> If you are going to use Heketi skip the volume creation steps

Creating your storage volume

Now that you have both of your nodes created and in sync, you will need to create a volume that your clients will be able to use.

Syntax :

```
gluster volume create $VOL_NAME replica $NUMBER_OF_NODES transport tcp
$DOMAIN_NAME1:/path/to/directory $DOMAIN_NAME2.com:/path/to/directory force

## actual syntax in for our example

gluster volume create testvolume replica 2 transport tcp glusterfs00:/gluster-volume glusterfs01:/gluster-volume
force
```

Start the volume you have created :

```
gluster volume start testvolume
```

Configuring your client(s)

```
apt-get install software-properties-common
add-apt-repository ppa:gluster/glusterfs-7
```

```
apt install glusterfs-client
```

Once completed, you will need to mount the storage that you previously created. First, make sure you have your mount point created :

```
mkdir /gluster-data
```

Mount your volume to your newly created mount point :

```
mount -t glusterfs gluster00:testvolume /gluster-data
```

Adding / Removing a brick from production

Once your node is ready with the proper packages and updates...
Make sure to edit its /etc/hosts and update every other nodes as well with your new entry :

```
echo "10.1.1.5 gluster03" >> /etc/hosts
```

Adding a new brick

Once you've completed the above points, simply connect on a node already part of the cluster :

```
gluster peer probe gluster03
```

And connect it to the volumes you want the new node to be connected to :

```
gluster volume add-brick testvolume replica 3 gluster03:/gluster-volum
```

Removing a clustered brick
Re-adding a node that has been previously removed

Install Heketi on **one** of the nodes

> Requirement : Already existing GlusterFS install

Download Heketi bin

```
wget https://github.com/heketi/heketi/releases/download/v9.0.0/heketi-v9.0.0.linux.amd64.tar.gz
tar -zxvf heketi-v9.0.0.linux.amd64.tar.gz
```

Copy bin

```
chmod +x heketi/{heketi,heketi-cli}
cp heketi/{heketi,heketi-cli} /usr/local/bin
```

Check heketi is working

```
heketi --version
heketi-cli --version
```

Add a user/group for heketi

```
groupadd --system heketi
useradd -s /sbin/nologin --system -g heketi heketi
```

Create dir for heketi

```
mkdir -p /var/lib/heketi /etc/heketi /var/log/heketi
```

```
vim /etc/heketi/heketi.json
```

> Make sure you replace the "key" values with proper passwords

```
{
  "_port_comment": "Heketi Server Port Number",
  "port": "8080",

  "_enable_tls_comment": "Enable TLS in Heketi Server",
  "enable_tls": false,

  "_cert_file_comment": "Path to a valid certificate file",
  "cert_file": "",

  "_key_file_comment": "Path to a valid private key file",
  "key_file": "",


  "_use_auth": "Enable JWT authorization. Please enable for deployment",
  "use_auth": false,

  "_jwt": "Private keys for access",
  "jwt": {
```

```
    "_admin": "Admin has access to all APIs",
    "admin": {
      "key": "KEY_HERE"
    },
    "_user": "User only has access to /volumes endpoint",
    "user": {
      "key": "KEY_HERE"
    }
  },

  "_backup_db_to_kube_secret": "Backup the heketi database to a Kubernetes secret when running in
Kubernetes. Default is off.",
  "backup_db_to_kube_secret": false,

  "_profiling": "Enable go/pprof profiling on the /debug/pprof endpoints.",
  "profiling": false,

  "_glusterfs_comment": "GlusterFS Configuration",
  "glusterfs": {
    "_executor_comment": [
      "Execute plugin. Possible choices: mock, ssh",
      "mock: This setting is used for testing and development.",
      "      It will not send commands to any node.",
      "ssh:  This setting will notify Heketi to ssh to the nodes.",
      "      It will need the values in sshexec to be configured.",
      "kubernetes: Communicate with GlusterFS containers over",
      "            Kubernetes exec api."
    ],
    "executor": "ssh",

    "_sshexec_comment": "SSH username and private key file information",
    "sshexec": {
      "keyfile": "/etc/heketi/heketi_key",
      "user": "root",
      "port": "22",
      "fstab": "/etc/fstab"
    },

    "_db_comment": "Database file name",
    "db": "/var/lib/heketi/heketi.db",
```

```
    "_refresh_time_monitor_gluster_nodes": "Refresh time in seconds to monitor Gluster nodes",
    "refresh_time_monitor_gluster_nodes": 120,

    "_start_time_monitor_gluster_nodes": "Start time in seconds to monitor Gluster nodes when the heketi comes up",
    "start_time_monitor_gluster_nodes": 10,

    "_loglevel_comment": [
      "Set log level. Choices are:",
      "  none, critical, error, warning, info, debug",
      "Default is warning"
    ],
    "loglevel" : "debug",

    "_auto_create_block_hosting_volume": "Creates Block Hosting volumes automatically if not found or exsisting volume exhausted",
    "auto_create_block_hosting_volume": true,

    "_block_hosting_volume_size": "New block hosting volume will be created in size mentioned, This is considered only if auto-create is enabled.",
    "block_hosting_volume_size": 500,

    "_block_hosting_volume_options": "New block hosting volume will be created with the following set of options. Removing the group gluster-block option is NOT recommended. Additional options can be added next to it separated by a comma.",
    "block_hosting_volume_options": "group gluster-block",

    "_pre_request_volume_options": "Volume options that will be applied for all volumes created. Can be overridden by volume options in volume create request.",
    "pre_request_volume_options": "",

    "_post_request_volume_options": "Volume options that will be applied for all volumes created. To be used to override volume options in volume create request.",
    "post_request_volume_options": ""
  }
}
```


Load all Kernel modules that will be required by Heketi.

```
for i in dm_snapshot dm_mirror dm_thin_pool; do
  sudo modprobe $i
done
```

Create ssh key for the API to connect to the other hosts

```
ssh-keygen -f /etc/heketi/heketi_key -t rsa -N ''
chown heketi:heketi /etc/heketi/heketi_key*
```

Send key to all hosts

```
for i in gluster00 gluster01 gluster02; do
  ssh-copy-id -i /etc/heketi/heketi_key.pub root@$i
done
```

Create a systemd file

```
vim /etc/systemd/system/heketi.service
```

```
[Unit]
Description=Heketi Server

[Service]
Type=simple
WorkingDirectory=/var/lib/heketi
EnvironmentFile=-/etc/heketi/heketi.env
User=heketi
ExecStart=/usr/local/bin/heketi --config=/etc/heketi/heketi.json
Restart=on-failure
StandardOutput=syslog
StandardError=syslog

[Install]
WantedBy=multi-user.target
```

Reload systemd and enable new heketi service

```
systemctl daemon-reload
systemctl enable --now heketi
```

Allow heketi user perms on folders

```
chown -R heketi:heketi /var/lib/heketi /var/log/heketi /etc/heketi
```

Create topology

```
vim /etc/heketi/topology.json
```

```
{
  "clusters": [
    {
      "nodes": [
                {
          "node": {
            "hostnames": {
              "manage": [
                "gluster00"
              ],
              "storage": [
                "10.1.1.2"
              ]
            },
            "zone": 1
          },
          "devices": [
            "/dev/vdc","/dev/vdd","/dev/vde"
          ]
        },           {
          "node": {
            "hostnames": {
              "manage": [
                "gluster01"
              ],
              "storage": [
                "10.1.1.3"
              ]
            },
            "zone": 1
          },
          "devices": [
            "/dev/vdc","/dev/vdd","/dev/vde"
          ]
```

```
      },          {
        "node": {
          "hostnames": {
            "manage": [
              "gluster02"
            ],
            "storage": [
              "10.1.1.4"
            ]
          },
          "zone": 1
        },
        "devices": [
          "/dev/vdc","/dev/vdd","/dev/vde"
        ]
      }
    ]
  }
 ]
}
```

Load topology

(note you can make changes and the load it again in the future if you want to add more drives)

```
heketi-cli topology load --json=/etc/heketi/topology.json
```

Check connection to other devices work

```
heketi-cli cluster list
```

Notes

Mount all volumes

```
for i in `gluster volume list`
do mkdir -p /etc/borg/gluster_backup/$i && \
mount -t glusterfs 127.0.0.1:$i /mnt/$i
```

done