Подготовительная часть

**Подготовка ceph:**

На узлах, где хранилище:

cd /etc/ceph/

ceph auth get-key client.libvirt | tee client.libvirt.key

Отредактируем ceph.conf (Секцию client.libvirt убрать и osd memory target = 4294967296) и scp:

scp /etc/ceph/ceph.conf server:/etc/ceph

(server имя или ip-адрес гипервизора)

На узле с СГУ:

cat > secret.xml <<EOF

<secret ephemeral='no' private='no'>

<uuid>$(uuidgen) </uuid>

<usage type='ceph'>

<name>client.libvirt secret</name>

</usage>

</secret>

EOF

Под пользователем oneadmin на все гипервизоры скопировать secret.xml и client.libvirt.key в домашний каталог пользователя oneadmin

sshpass -p "horizon" scp client.libvirt.key secret.xml oneadmin@server:

На каждом гипервизоре выполнить команду под oneadmin

su - oneadmin

virsh -c qemu:///system secret-define secret.xml

Скопировать UUID из secret.xml (cat secret.xml) и ввести команду

virsh -c qemu:///system secret-set-value --secret UUID --base64 **$(**cat client.libvirt.key**)**

rm client.libvirt.key

**Подготовка multipath device**

На всех серверах выполнить:

cat > /etc/multipath.conf <<EOF

defaults {

find\_multipaths yes

user\_friendly\_names yes

}

blacklist {

}

EOF

rc-service wdmd stop

rc-service sanlock stop

rc-service lvmlockd stop

rc-service lvmetad stop

rc-service dmeventd stop

rc-update del wdmd -a

rc-update del sanlock -a

rc-update del lvmlockd -a

rc-update del lvmetad -a

rc-update del dmeventd -a

rc-service multipath start

rc-service multipathd start

rc-update add multipath default

rc-update add multipathd default

**Подготовка LVM:**

В /etc/lvm/lvm.conf:

#Брать iscsi device через /dev/mapper/${MPATH}, а не через /dev/disk/by-id/iscsi-...

159: global\_filter = [ "a|/dev/sda|", "/dev/mapper/mpatha", "r|.\*|" ]

786: locking\_type = 1

953: use\_lvmetad = 0

967: use\_lvmlockd = 0

Выполнить

rc-service wdmd stop

rc-service sanlock stop

rc-service lvmlockd stop

rc-service lvmetad stop

rc-service dmeventd stop

rc-update del wdmd -a

rc-update del sanlock -a

rc-update del lvmlockd -a

rc-update del lvmetad -a

rc-update del dmeventd -a

Выполняем команды

vgcreate TAM-VG0 /*dev/mapper/mpatha*

*lvcreate -L 20G -n hcs-ha TAM-VG0*

## *mkfs.xfs /dev/TAM-VG0/hcs-ha*

**Подготовка docker:**

Запустить докер на всех серверах

rc-service docker start

rc-update add docker default

На каждый хост залить docker контейнер нужной версии и подписать

*for i in {30..33}; do ssh h$i docker load < /data/0/hcs-1.1.27.tar.bz2; done*

*for i in {30..33}; do ssh h$i hvs\_sign; done*

На одном из хостов запустить контейнер, чтобы он создал необходимые директории в папке /*data*/0/docker/volumes/hvol/\_data.

docker run -d --name=hcs \

--network=host \

--restart=always \

--security-opt=seccomp:unconfined \

-v /var/tmp:/var/tmp \

-v hvol:/hvol \

--stop-timeout=90 \

hcs:latest

Скопировать их на девайс на этом же хосте

*mkdir -p /data/hcs/hvol*

*mount /dev/TAM-VG0/hcs-ha /data/hcs*

*cp –a /data/0/docker/volumes/hvol/\_data/\* /data/hcs/hvol/*

*umount /data/hcs*

**Cоздать порт в ovs hvssw0 для плавающего ip**

ovs-vsctl add-port hvssw0 ha-port -- set Interface ha-port type=internal

## **Настройка кластерного ПО**

На всех гипервизорах создаем файл */etc/corosync/corosync.conf* со следующим содержимым

# Totem Protocol Configuration

totem {

version: 2

cluster\_name: TEST\_CLUSTER\_BLT

secauth: off

bindnetaddr: 10.10.101.39

transport: udpu

}

# Nodelist - Server List

nodelist {

node {

ring0\_addr: h30

nodeid: 1

}

node {

ring0\_addr: h31

nodeid: 2

}

node {

ring0\_addr: h32

nodeid: 3

}

node {

ring0\_addr: h33

nodeid:4

}

}

# Quorum configuration

quorum {

provider: corosync\_votequorum

two\_node: 0

}

# Corosync Log configuration

logging {

to\_logfile: yes

logfile: /var/log/corosync/corosync.log

to\_syslog: yes

debug: on

timestamp: on

}

service {

}

На всех гипервизорах создаем директорию для логов

mkdir -p /var/log/corosync/

Включаем и добавляем сервисы в автозагрузку на всех узлах

rc-service corosync start

rc-service pacemaker start

rc-update add pacemaker default

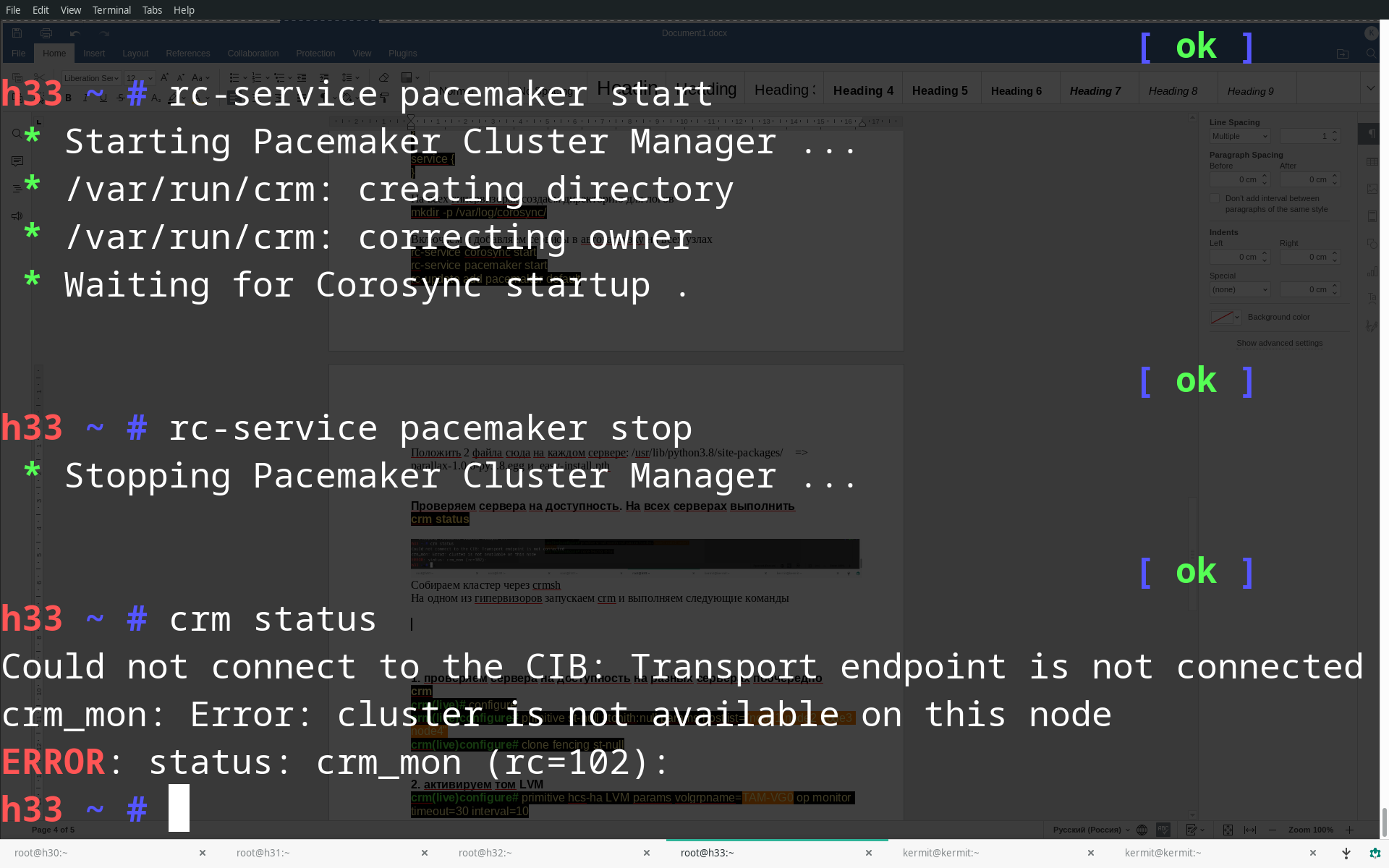
Положить 2 файла сюда на каждом сервере: /usr/lib/python3.8/site-packages/ =>

parallax-1.0.6-py3.8.egg и easy-install.pth

**Проверяем сервера на доступность. На всех серверах выполнить**

**crm status**

Если выдаёт такую ошибку, значит pacemaker/corosync не был запущен



**Собираем кластер через crmsh**

На одном из гипервизоров запускаем crm и выполняем следующие команды

**1. Добавляем fencing**

**crm**

**crm(live)#** configure

**crm(live)configure#** primitive st-null stonith:null params hostlist="h30 h31 h32 h33"

**crm(live)configure#** clone fencing st-null

**2. активируем том LVM**

**crm(live)configure#** primitive hcs-ha LVM params volgrpname=TAM-VG0 op monitor timeout=30 interval=10

**3. примонтируем файловую систему**

**crm(live)configure#** primitive fs\_work Filesystem params device="/dev/TAM-VG0/hcs-ha" directory="/data/hcs" fstype=xfs op monitor interval=10s

**4. запускаем контейнер**

**crm(live)configure#** primitive docker\_start docker \

params image="hcs:latest" name=hcs run\_opts="--network=host --restart=always --security-opt=seccomp:unconfined -v /var/tmp:/var/tmp -v /data/hcs/hvol:/hvol --stop-timeout=90" \

meta target-role=Started

**5. назначаем адрес на ранее созданый порт в hvssw0**

**crm(live)configure#** primitive vip\_sgu IPaddr2 \

params ip=10.10.101.39 cidr\_netmask=24 nic=ha-port op monitor interval=10 timeout=20

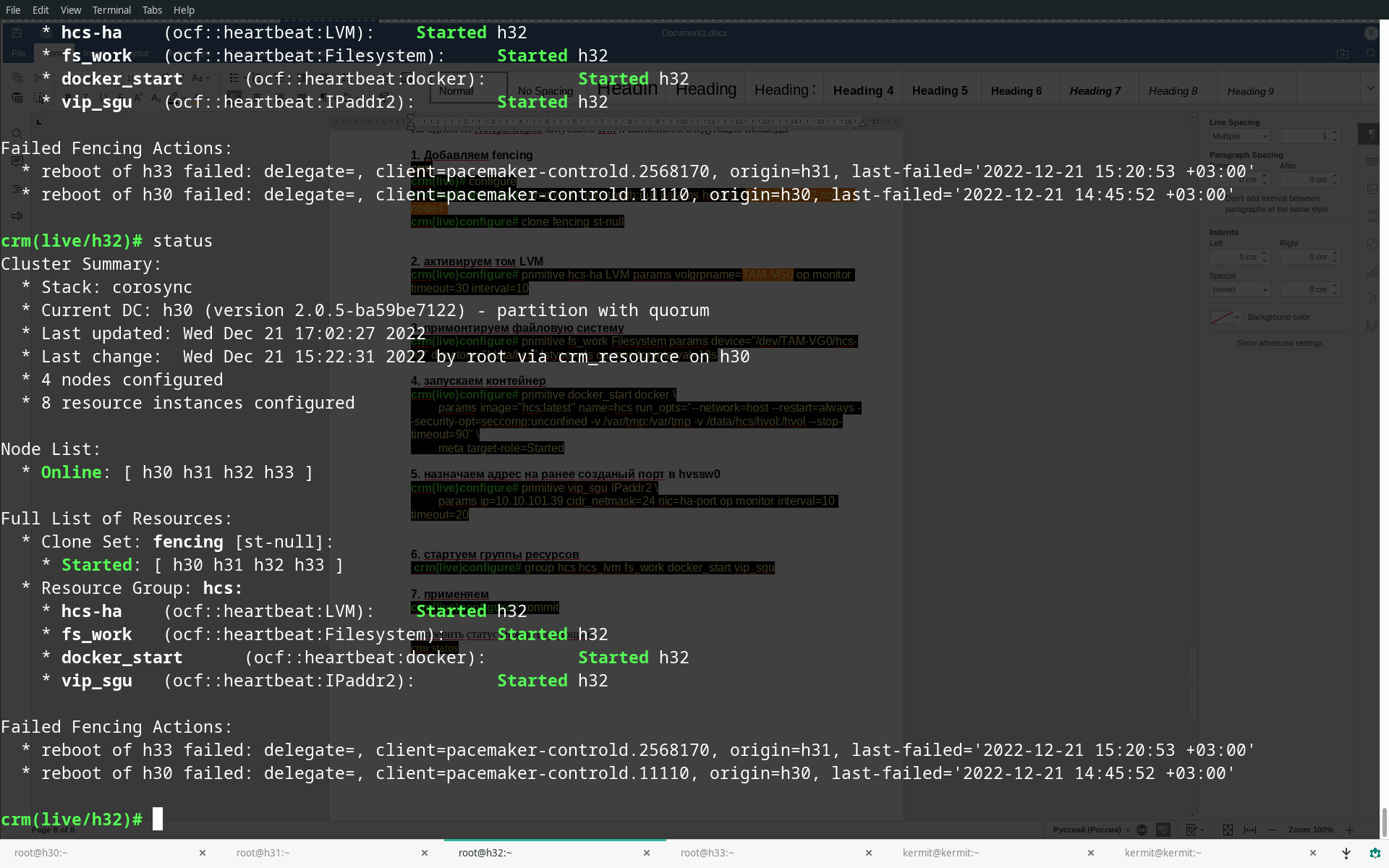
**6. стартуем группы ресурсов**

**crm(live)configure#** group hcs hcs\_lvm fs\_work docker\_start vip\_sgu

**7. применяем**

**crm(live)configure#** commit

Проверить статус можно с помощью crm status



Дополнительно

Чтобы поменять образ для СГУ в докере, в командной строке:

EDITOR=vim crm configure edit

Исправить в строке **primitive docker\_start** параметр

*image="hcs:latest"* на *image="****YOUR\_IMAGE****"*