OpenVZ на ZFS

Установка OpenZFS

Документация по ZVOL: http://zfsonlinux.org/example-zvol.html

Снчала нужно установить OpenVZ ядро. Перезагрузиться в него и _ТОЛЬКО_ после этого установить ZFS:

```
yum remove kernel
yum install -y vzkernel-devel
rpm -ihv http://download.fedoraproject.org/pub/epel/6/i386/epel-release-6-8.noarch.rpm
yum localinstall --nogpgcheck http://archive.zfsonlinux.org/epel/zfs-release$(rpm -E
%dist).noarch.rpm
yum update -y
yum install zfs -y
```

ZFS модули подцепятся по DKMS и будут пересобираться под каждое новое ядро.

Совамещение ZFS с OpenVZ

Потом создаем тушку simfs:

```
vzctl create 101 --layout simfs --name zfs.ru  --ostemplate debian-7.0-x86_64
```

Создаем zvol на 10G:

```
zfs create -V 10G vz/ct101
```

Его можно сделать спарсом (а также задать размер блока), но стоит прочесть документацию:

```
The reservation is kept equal to the volume’s logical size to prevent unexpected behavior for
consumers. Without  the reservation, the volume could run out of space, resulting in undefined
behavior or data corruption, depending on how the volume is used. These effects can  also  occur
when  the  volume  size  is changed  while  it  is  in  use  (particularly when shrinking the size).
Extreme care should be used when adjusting the volume size. Though not recommended, a "sparse
volume" (also known as "thin provisioning") can be created by  specifying  the -s option to the zfs
create -V command, or by changing the reservation after the volume has been created. A "sparse
volume" is a volume where the reservation is less then the volume size.  Consequently, writes  to  a
sparse  volume  can  fail  with ENOSPC when the pool is low on space. For a sparse volume, changes
to volsize are not reflected in the reservation.

-b blocksize
Equivalent to -o volblocksize=blocksize. If this option is specified in conjunction with -o volblock-size,
the resulting behavior is undefined.
```

Создаем разметку GPT:

```
parted -s /dev/vz/ct101 mklabel gpt
```

Создаем 1 раздел на весь диск:

```
parted /dev/vz/ct101 "mkpart primary 1 -1"
```

Создаем FS а-ля ploop:

```
mkfs -t ext4 -j -b4096 -Elazy_itable_init,resize=4294967295 -Jsize=128 /dev/vz/ct101
```

Переносим в нее контейнер:

```
mkdir /mnt/101
mount /dev/vz/ct101 /mnt/101
cp -a /vz/private/101/* /mnt/101/
umount /mnt/101
sync
```

Теперь стираем старую тушку контейнера:

```
rm -rf /vz/private/101/
```

И подменяем ее смонтированным ZFS zvol:

```
mkdir /vz/private/101
mount /dev/vz/ct101 /vz/private/101
```

Монтируем контейнер:

```
vzctl mount 101
Container is mounted
```

Убеждаемся, что появились данные в /vz/root/101:

```
[root@fps6 ~]# ls -la /vz/root/101
итого 104K
drwxr-xr-x 23 root root 4,0K Июн 30 21:06 .
drwxr-xr-x  3 root root 4,0K Июн 30 20:50 ..
drwxr-xr-x  2 root root 4,0K Июн 27 02:38 bin
drwxr-xr-x  2 root root 4,0K Июн 27 02:37 boot
drwxr-xr-x  2 root root 4,0K Июн 27 02:36 dev
drwxr-xr-x 81 root root 4,0K Июн 30 20:50 etc
-rw-r--r--  1 root root    0 Июн 27 02:36 fastboot
drwxr-xr-x  2 root root 4,0K Июн 27 02:37 home
drwxr-xr-x 11 root root 4,0K Июн 27 02:37 lib
drwxr-xr-x  2 root root 4,0K Июн 27 02:37 lib64
drwx------  2 root root  16K Июн 27 02:36 lost+found
drwxr-xr-x  2 root root 4,0K Июн 27 02:37 media
drwxr-xr-x  2 root root 4,0K Июн 27 02:37 mnt
drwxr-xr-x  2 root root 4,0K Июн 27 02:37 opt
dr-xr-xr-x  2 root root 4,0K Июн 27 02:36 proc
drwx------  2 root root 4,0K Июн 27 02:37 root
drwxr-xr-x  2 root root 4,0K Июн 27 02:38 run
drwxr-xr-x  2 root root 4,0K Июн 27 02:37 sbin
drwxr-xr-x  2 root root 4,0K Июн 27 02:37 selinux
drwxr-xr-x  2 root root 4,0K Июн 27 02:37 srv
drwxr-xr-x  2 root root 4,0K Июн 27 02:36 sys
drwxrwxrwt  2 root root 4,0K Июн 27 02:38 tmp
drwxr-xr-x 10 root root 4,0K Июн 27 02:38 usr
drwxr-xr-x 12 root root 4,0K Июн 27 02:38 var
```

Запускаем:

```
vzctl start 101
Starting container...
Container is unmounted
Container is mounted
Setting CPU units: 1000
Container start in progress...
```

Вуаля 🙂

Внутри контейнера якобы simfs:

```
cat /proc/mounts
/dev/simfs / simfs rw,relatime 0 0
proc /proc proc rw,nosuid,nodev,noexec,relatime 0 0
sysfs /sys sysfs rw,nosuid,nodev,noexec,relatime 0 0
none /dev devtmpfs rw,nosuid,noexec,relatime,mode=755 0 0
none /dev/pts devpts rw,nosuid,noexec,relatime,gid=5,mode=620,ptmxmode=000 0 0
none /proc/sys/fs/binfmt_misc binfmt_misc rw,nosuid,nodev,noexec,relatime 0 0
none /run tmpfs rw,nosuid,noexec,relatime,size=26216k,mode=755 0 0
none /run/lock tmpfs rw,nosuid,nodev,noexec,relatime,size=5120k 0 0
none /run/shm tmpfs rw,nosuid,nodev,noexec,relatime,size=157280k 0 0
none /run/user tmpfs rw,nosuid,nodev,noexec,relatime,size=102400k,mode=755 0 0
```

ZFS Snapshots

Создаем snapshot клиента:

```
zfs snapshot vz/ct101@test
```

Смотрим имеющиеся снапшоты:

```
zfs list -t snapshot
NAME          USED  AVAIL  REFER  MOUNTPOINT
vz/ct101@test  95,9K     -   898M  -
```

ZFS клонирование

Создаем еще контейнер:

```
zfs create -V 10G vz/ct102
```

Клонируем его из существующего снапшота:

```
zfs clone vz/ct101 vz/ct102
```

Вуаля! В данный контейнер перенеслась файловая система целиком!

Недостатки решения

- Отсутствие интеграции vzctl set для изменения размера диска находу
- Отсуствие создания контейнеров без кастомизации
- Не особо корректная попытка обмана vzctl с тем, что это якобы simfs и последующее совершенно лишнее подключение vzquota.
- Отсутствие интеграции ZFS снапшотов с механизмом vzctl chkpnt, что приводит к инконсистентности бэкапа сделанного без ведом OpenVZ
- Не понятно, возникает ли ситуация двойного кэширования или нет.
- Как изолировать кэш ZFS и аккаунтить в контексте контейнера - не ясно
- Отсутствие обработки ошибок вложенной фс (ext4)
- Нужно тестировать, чтобы выяснить наиболее оптимальный размер блока на ZFS zvol устройстве
- Не понятно, как будут работать ioprio и iops лимиты и будут ли вообще

Возможные кейсы

- Rock Stable файловая система и отсутсвтие необходимости в аппаратных RAID контроллерах
- Внедрение в vzctl возможность создания одинх контейнеров из клонов других
- Использование templates уже размещенных в файловой системе для ускоренного создания новых контейнеров
- Out of box поддержка снапшотов и бэкапов на их базе
- Тотальная дедупликация и сжатие контейнерво находу -> огромная плотность размещения