Subject: Proposal for directory based VE containers
Posted by augustz on Thu, 21 Sep 2006 20:43:55 GMT
View Forum Message <> Reply to Message

Proposal for directory based VE containers

A VE is defined by a directory on the system. This directory might have the following structure

/veid
./dump
./private
./lock
./conf

All VE specifc items are located in this directory.

Reasoning
This creates a super elegant setup, which makes the VE itself very portable and easy to extend.

I'd leave vz.conf and some other hn specific items in the usual areas, but for a VE to spread files all over the host nodes configuration area is I think messier then it needs to be. OpenVZ is already pretty close to this, this I think just polishes it off a bit and makes it even sexier and cleaner.

There are some immediate wins for some common use cases under this proposal:

Backing up the VE:
Backup becomes very easy. You simply tar up the /veid directory. No longer do you have to grab /etc/sysconfig/vz-scripts/veid.conf etc and mangle that into a backup location when backing up a ve. VE data stays within the /veid directory container.

Migration:
VE's can move from one machine to another. With all ve specific data in the /veid container, migration is simplified. VE configuration data is not host node data, and so would move with the VE rather then needed to be setup and cleaned up on each host node the VE moves to.

Managability
Managment apps can inspect a /veid and see how type of resources it's expecting to get access to. Ve can easily be reduced to a single file. If needed for bookkeeping or efficiancy symbolic links could tie new locations for things to their existing spots.

Expandability:
Each subdirectory of the /veid is in some ways a resource fork, which makes future expandability easier as well. If I created a managment app that added some data about a ve I can simply add a directory to the /veid directory, which will survive even if the ve moves to another machine, and can track VE specific info

Future:
I'd go even so far as to define a veid by the full path at which the ve can be accessed, with the last

part of the path used to start it as an alias. This leverages the filesystem and removes bookkeeping requirements. eg /vz/ve/104 is the full name of the ve, with 104 being the alias.

With the lock file in this /veid container we even have an easy way to insure that multiple ways of accessing the same root path don't result in duplicate instances of the same vm. Duplicate veid's are similarly prevented, as they result in file conflicts on the file system. Under this to delete a ve you can simply stop a ve and delete the /veid container, no need to chase down old .conf files etc.

Thoughts and/or comments on this proposal? I think it has the potential to make openvz cleaner and even easier to understand and use.
