Subject: Beginners question: NFS client in VPS
Posted by noe2 on Fri, 03 Mar 2006 10:52:58 GMT
View Forum Message <> Reply to Message

Hi,

I'm just making my first steps with openVZ and I'm wondering
if it's possible to have an NFS client running inside a VPS ?

If yes : Can anyone please give me some hints on how to enable
it ? Currently my VPS do not list nfs
in /proc/filesystems.

If not : Why not ?

I'm currently running 2.6.8-022stab070-smp on x86_64 with
NFS enabled in the kernel (as module). The module is loaded
on the hardware host.

Regards,
Norbert

Subject: Re: Beginners question: NFS client in VPS
Posted by dev on Fri, 03 Mar 2006 15:38:09 GMT
View Forum Message <> Reply to Message

There are 2 possible ways to achieve this:
1. mount NFS in host system, and then bind mount it to required VPS. This is the prefferable way of doing things. And this is why NFS is disabled in VPS by default.

2. you can mark nfs_fs_type as FS_VIRTUALIZED, and NFS will be available in VPSs. But it is not a recommended way since NFS is not virtualized fully yet.

Subject: Re: Beginners question: NFS client in VPS
Posted by youp on Fri, 03 Mar 2006 22:14:53 GMT
View Forum Message <> Reply to Message

how to change nfs_fs_type as FS_VIRTUALIZED ?

thanks

Subject: Re: Beginners question: NFS client in VPS
Posted by dev on Sat, 04 Mar 2006 09:12:52 GMT
View Forum Message <> Reply to Message

apply this patch. it should help.

```
--- linux-2.6.8-022stab064/fs/nfs/inode.c.nonfs 2006-02-10 15:21:25.822784424 +0300
+++ linux-2.6.8-022stab064/fs/nfs/inode.c 2006-02-10 15:25:29.587726520 +0300
@@ -1399,7 +1399,8 @@ static struct file_system_type nfs_fs_ty
  .name  = "nfs",
  .get_sb  = nfs_get_sb,
  .kill_sb = nfs_kill_super,
- .fs_flags = FS_ODD_RENAME|FS_REVAL_DOT|FS_BINARY_MOUNTDATA,
+ .fs_flags = FS_ODD_RENAME|FS_REVAL_DOT|
+     FS_BINARY_MOUNTDATA|FS_VIRTUALIZED,
 };

 #ifdef CONFIG_NFS_V4
--- linux-2.6.8-022stab064/net/sunrpc/rpc_pipe.c.nonfs 2006-02-10 15:20:28.040568656 +0300
+++ linux-2.6.8-022stab064/net/sunrpc/rpc_pipe.c 2006-02-10 14:25:48.000000000 +0300
@@ -804,6 +804,7 @@ static struct file_system_type rpc_pipe_
  .name  = "rpc_pipefs",
  .get_sb  = rpc_get_sb,
  .kill_sb = kill_litter_super,
+ .fs_flags = FS_VIRTUALIZED,
 };

 static void
--- linux-2.6.8-022stab064/kernel/exit.c.nonfs 2006-02-10 15:21:55.098333864 +0300
+++ linux-2.6.8-022stab064/kernel/exit.c 2006-02-10 15:19:53.045888656 +0300
@@ -271,9 +271,6 @@ void __set_special_pids(pid_t session, p
 {
  struct task_struct *curr = current;

- WARN_ON(is_virtual_pid(pgrp));
- WARN_ON(is_virtual_pid(session));
-
  if (curr->signal->session != session) {
   detach_pid(curr, PIDTYPE_SID);
   curr->signal->session = session;
--- linux-2.6.8-022stab064/arch/i386/kernel/process.c.nonfs 2006-02-10 15:23:36.306947816
+0300
+++ linux-2.6.8-022stab064/arch/i386/kernel/process.c 2006-02-10 15:29:36.513188144 +0300
@@ -276,13 +276,6 @@ int kernel_thread(int (*fn)(void *), voi
 {
  struct pt_regs regs;

- /* Don't allow kernel_thread() inside VE */
- if (!ve_is_super(get_exec_env())) {
```

```
-  printk("kernel_thread call inside VE\n");
-  dump_stack();
-  return -EPERM;
- }
-
  memset(&regs, 0, sizeof(regs));

  regs.ebx = (unsigned long) fn;
--- ./arch/x86_64/kernel/process.c.xxxx 2006-01-23 20:25:04.000000000 +0300
+++ ./arch/x86_64/kernel/process.c 2006-02-10 15:30:13.000000000 +0300
@@ -719,12 +719,6 @@ long do_fork_kthread(unsigned long clone
        int __user *parent_tidptr,
        int __user *child_tidptr)
 {
- if (ve_is_super(get_exec_env()))
-  return do_fork(clone_flags, stack_start, regs, stack_size,
-    parent_tidptr, child_tidptr);
-
- /* Don't allow kernel_thread() inside VE */
- printk("kernel_thread call inside VE\n");
- dump_stack();
- return -EPERM;
+ return do_fork(clone_flags, stack_start, regs, stack_size,
+   parent_tidptr, child_tidptr);
 }
```
