Subject: Re: OpenVZ and the Ubuntu Upstart init daemon
Posted by Daniel Pittman on Sat, 07 Apr 2007 13:20:35 GMT
View Forum Message <> Reply to Message

Daniel Pittman <daniel@rimspace.net> writes:

First, sorry for the cross-post. Since this is an issue with the
interaction between the two software packages -- and seems to be of
general interest on the OpenVZ users list where I first posted -- I felt
it was best to copy all three locations up front.

> G'day. I have been playing with the 'upstart' daemon that replaces
> the traditional /sbin/init process in Ubuntu Edgy and Feisty.
>
> I didn't have a huge degree of luck with this -- once I allocated a
> tty for the console I ended up with the system failing during the
> bootstrap process.
>
> Before investing more time in this I was wondering if anyone else had
> been working on getting upstart to cooperate in the OpenVZ boot
> process?

I spent some more time tracking this down and seem to be a conflict
between the expectations of the Upstart process and the environment
provided by OpenVZ when starting init in a new VE.

All the testing was with vzctl 3.0.16-1dso2 (from Debian/testing) and
Upstart 0.3.8 from Feisty.

At issue are the expectations about file description layout that are
made in init/main.c of upstart; on line 150 the 'control_open' method is
called.

That, eventually, binds a Unix domain socket for the Upstart control
connection, used to support 'telinit' among other options.

Later, on line 209, Upstart cleans up the environment:

```
    /* Close any file descriptors we inherited, and open the console
    * device instead.  Normally we reset the console, unless we're
    * inheriting one from another init process.
    */
    for (int i = 0; i < 3; i++)
        close (i);
```

During a normal boot sequence those three descriptors are bound to
STDIN, OUT and ERR just as expected. The result is that Upstart

correctly rebinds the console and everything "just works."

Under OpenVZ startup, however, things don't quite look the same.
OpenVZ will exec the init process with *no* open file descriptors.

This causes the control_open method to bind file description 0 to the
Unix domain socket -- and the later loop to carefully close the socket
again.

I was able to verify my belief that this was causing the problems by
replacing init with a shell script that simply did:

```
  exec /sbin/init.real "$@" </dev/null >/dev/null 2>/dev/null
```

Once that was in place Upstart kept the control socket open and
proceeded considerably further through the boot sequence.

It isn't quite clear to me who is at fault here -- I presume that in the
real world the kernel and linuxrc processes never start upstart without
the first three file descriptors open, but I don't know that is assured
anywhere.

I am happy to help with further testing or to explain how to configure
an OpenVZ environment to reproduce this, if desired.

Regards,
        Daniel
--
Digital Infrastructure Solutions -- making IT simple, stable and secure
Phone: 0401 155 707   email: contact@digital-infrastructure.com.au
                 http://digital-infrastructure.com.au/