Subject: Re: [RFD] reboot / shutdown of a container
Posted by Bruno Pr on Fri, 14 Jan 2011 23:11:09 GMT
View Forum Message <> Reply to Message

On Thu, 13 January 2011 Daniel Lezcano <daniel.lezcano@free.fr> wrote:
> On 01/13/2011 10:50 PM, Bruno Prémont wrote:
> > On Thu, 13 January 2011 Daniel Lezcano<daniel.lezcano@free.fr> wrote:
> >> On 01/13/2011 09:09 PM, Bruno Prémont wrote:
> >>> On Thu, 13 January 2011 Daniel Lezcano<daniel.lezcano@free.fr> wrote:
> >>>> in the container implementation, we are facing the problem of a process
> >>>> calling the sys_reboot syscall which of course makes the host to
> >>>> poweroff/reboot.
> >>>>
> >>>> If we drop the cap_sys_reboot capability, sys_reboot fails and the
> >>>> container reach a shutdown state but the init process stay there, hence
> >>>> the container becomes stuck waiting indefinitely the process '1' to exit.
> >>>>
> >>>> The current implementation to make the shutdown / reboot of the
> >>>> container to work is we watch, from a process outside of the container,
> >>>> the<rootfs>/var/run/utmp file and check the runlevel each time the file
> >>>> changes. When the 'reboot' or 'shutdown' level is detected, we wait for
> >>>> a single remaining in the container and then we kill it.
> >>>>
> >>>> That works but this is not efficient in case of a large number of
> >>>> containers as we will have to watch a lot of utmp files. In addition,
> >>>> the /var/run directory must *not* mounted as tmpfs in the distro.
> >>>> Unfortunately, it is the default setup on most of the distros and tends
> >>>> to generalize. That implies, the rootfs init's scripts must be modified
> >>>> for the container when we put in place its rootfs and as /var/run is
> >>>> supposed to be a tmpfs, most of the applications do not cleanup the
> >>>> directory, so we need to add extra services to wipeout the files.
> >>>>
> >>>> More problems arise when we do an upgrade of the distro inside the
> >>>> container, because all the setup we made at creation time will be lost.
> >>>> The upgrade overwrite the scripts, the fstab and so on.
> >>>>
> >>>> We did what was possible to solve the problem from userspace but we
> >>>> reach always a limit because there are different implementations of the
> >>>> 'init' process and the init's scripts differ from a distro to another
> >>>> and the same with the versions.
> >>>>
> >>>> We think this problem can only be solved from the kernel.
> >>>>
> >>>> The idea was to send a signal SIGPWR to the parent of the pid '1' of the
> >>>> pid namespace when the sys_reboot is called. Of course that won't occur
> >>>> for the init pid namespace.
> >>> Wouldn't sending SIGKILL to the pid '1' process of the originating PID
> >>> namespace be sufficient (that would trigger a SIGCHLD for the parent

> >>> process in the outer PID namespace.
> >> This is already the case. The question is : when do we send this signal ?
> >> We have to wait for the container system shutdown before killing it.
> > I meant that sys_reboot() would kill the namespace's init if it's not
> > called from boot namespace.
> >
> > See below
> >
> >>> (as far as I remember the PID namespace is killed when its 'init' exits,
> >>> if this is not the case all other processes in the given namespace would
> >>> have to be killed as well)
> >> Yes, absolutely but this is not the point, reaping the container is not
> >> a problem.
> >>
> >> What we are trying to achieve is to shutdown properly the container from
> >> inside (from outside will be possible too with the setns syscall).
> >>
> >> Assuming the process '1234' creates a new process in a new namespace set
> >> and wait for it.
> >>
> >> The new process '1' will exec /sbin/init and the system will boot up.
> >> But, when the system is shutdown or rebooted, after the down scripts are
> >> executed the kill -15 -1 will be invoked, killing all the processes
> >> expect the process '1' and the caller. This one will then call
> >> 'sys_reboot' and exit. Hence we still have the init process idle and its
> >> parent '1234' waiting for it to die.
> > This call to sys_reboot() would kill "new process '1'" instead of trying to
> > operate on the HW box.
> > This also has the advantage that a container would not require an informed
> > parent "monitoring" it from outside (though it would not be restarted even if
> > requested without such informed outside parent).
>
> Oh, ok. Sorry I misunderstood.
>
> Yes, that could be better than crossing the namespace boundaries.
>
> >> If we are able to receive the information in the process '1234' : "the
> >> sys_reboot was called in the child pid namespace", we can take then kill
> >> our child pid. If this information is raised via a signal sent by the
> >> kernel with the proper information in the siginfo_t (eg. si_code
> >> contains "LINUX_REBOOT_CMD_RESTART", "LINUX_REBOOT_CMD_HALT", ...), the
> >> solution will be generic for all the shutdown/reboot of any kind of
> >> container and init version.
> > Could this be passed for a SIGCHLD? (when namespace is reaped, and received
> > by 1234 from above example assuming sys_reboot() kills the "new process '1'")
>
> Yes, that sounds a good idea.
>

> > Looks like yes, but with the need to define new values for si_code (reusing
> > LINUX_REBOOT_CMD_* would certainly clash, no matter which signal is choosen).
>
> CLD_REBOOT_CMD_RESTART

> CLD_REBOOT_CMD_HALT
> CLD_REBOOT_CMD_POWER_OFF

I would just map both to the same thing...

> CLD_REBOOT_CMD_RESTART2 (what about the cmd buffer, shall we ignore it ?)

The cmd buffer could be passed via si_ptr if we want it, otherwise it would
be the same as for CLD_REBOOT_CMD_RESTART (which would have si_ptr set to NULL
in case no si_code differentiation is needed)

> CLD_REBOOT_CMD_KEXEC (?)

I don't think kexec makes any sense inside a container, such a sys_reboot()
call should probably fail or fallback to _RESTART

> CLD_REBOOT_CMD_SW_SUSPEND (useful for the future checkpoint/restart)

Looks reasonable

> LINUX_REBOOT_CMD_CAD_ON and LINUX_REBOOT_CMD_CAD_OFF could be disabled
> for a non-init pid namespace, no ?

I haven't looked at how/when the state set by these is checked, but it could
keep its meaning and a CAD shortcut would act on the container to which the
active task on the given tty belongs. (so as if the process which would have
gotten SIGINT had issued sys_reboot(LINUX_REBOOT_CMD_RESTART), permissions
set aside)

Bruno

____
Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containe rs