[← RSS](https://s3hh.wordpress.com/2013/07/02/rss-2/)

[Line buffering (and talking computers and irc meetings) →](https://s3hh.wordpress.com/2013/07/23/line-buffering-and-talking-computers-and-irc-meetings/)

[**Creating and using containers – without privilege**](https://s3hh.wordpress.com/2013/07/19/creating-and-using-containers-without-privilege/)

Posted on [July 19, 2013](https://s3hh.wordpress.com/2013/07/19/creating-and-using-containers-without-privilege/) by [s3hh](https://s3hh.wordpress.com/author/s3hh/)

Today I posted a (working but mainly POC) patchset against lxc which allows me to create and start ubuntu-cloud containers – completely as an unprivileged user. For more details see the introductory email to the patchset at <http://sourceforge.net/mailarchive/forum.php?thread_name=1374246151-7069-9-git-send-email-serge.hallyn%40ubuntu.com&forum_name=lxc-devel>

Glossing over prerequisites (which you can see in the email), the actual commands I used were:

lxc-create -t /home/serge/lxc-ubuntu-cloud -P /home/serge/lxcbase -n x3 -f default.conf -- -T precise.tar.gz

lxc-start -P /home/serge/lxcbase -n x3

There’s more work to be done:

* unprivileged containers cannot (yet) be networked
* something needs to set up per-user cgroups at boot or login
* something needs to create a per-user lockdir under /run
* user namespaces need to be enabled in the default kernels
* template handling of caching and locking needs to be made saner with respect to configurable lxcpaths

These are pretty minor, though, compared to what we’ve already achieved:

* User namespace support (minus XFS support) is in the kernel – thanks to a heroic effort by Eric Biederman (and sponsored by Canonical).
* The work needed enable subuids – also written by Eric – was accepted into our shadow package in saucy.
* The basic patchset to enable use of user namespaces by privileged users has been in lxc for some time now.

Background on user namespaces:

When you create a new user namespace, it initially is unmapped. Your task has uid and gid -1. You can then map userids from the parent namespace onto userids in the new namespace by ranges. For instance if you are userid 1000, then you can map uid 1000 in the parent to uid 0 in the namespace. From the kernel’s point of view, you can only map uids which you have privilege over – either by being that uid, or having CAP\_SYS\_ADMIN in the parent.

This is where subuids come in. /etc/subuids and /etc/subgids list range of uids which users are allowed to map. The newuidmap and newgidmap are setuid-root programs which will respect those subuids to allow unprivileged users to map their allotted subuids.

Lxc uses these programs (indirectly through the ‘usernsexec’ program) to allow unprivileged users to map their allotted subuids to containers.

Of note is that regular DAC and MAC remain unchanged. Therefore although I as user serge/uid 1000 may have 100000-199999 as my subuids, I do not own files owned by those subuids! To work around this, map the uids together into a namespace. For instance, if you are uid 1000 and want to create a file owned by uid 100000, you can

touch foo

usernsexec -m b:0:100000:1 -m b:1000:1000 -- /bin/chown 0 foo

This maps 100000 on the host to root in the new namespace, and 1000 on the host to 1000 in the namespace. So host uid 100000 actually has privilege over the namespace, including over host uid 1000. It is therefore allowed to chown a file owned by uid 1000 to uid 0 (which is host uid 100000). You end up with foo owned by uid 100000. You can do the same sort of games to clean up containers (and lxc-destroy will do so).