LXC GPU Access

Giving a LXC guest GPU access allows you to use a GPU in a guest while it is still available for use in the host machine. This is a big advantage over virtual machines where only a single host or guest can have access to a GPU at one time. Even better, multiple LXC guests can share a GPU with the host at the same time.

> The information on this page is written for a host running Proxmox but should be easy to adapt to any machine running LXC/LXD.

> Since a device is being shared between two systems there are almost certainly some security implications and I haven't been able to determine what degree of security you're giving up to share a GPU.

Determine Device Major/Minor Numbers

To allow a container access to the device you'll have to know the devices major/minor numbers. This can be found easily enough by running `ls -l` in `/dev/`. As an example to pass through the integated UHD 630 GPU from an Core i7 8700k you would first list the devices where are created under `/dev/dri`.

```
root@blackbox:~# ls -l /dev/dri
total 0
drwxr-xr-x 2 root root        80 May 12 21:54 by-path
crw-rw---- 1 root video  226,   0 May 12 21:54 card0
crw-rw---- 1 root render 226, 128 May 12 21:54 renderD128
```

From that you can see the major device number is `226` and the minors are `0` and `128`.

Provide LXC Access

In the configuration file you'd then add lines to allow the LXC guest access to that device and then also bind mount the devices from the host into the guest. In the example above since both devices share the same major number it is possible to use a shorthand notation of `226:*` to represent all minor numbers with major number `226`.

```
# /etc/pve/lxc/*.conf
+ lxc.cgroup.devices.allow: c 226:* rwm
```

```
+ lxc.mount.entry: /dev/dri/card0 dev/dri/card0 none bind,optional,create=file,mode=0666
+ lxc.mount.entry: /dev/dri/renderD128 dev/dri/renderD128 none bind,optional,create=file
```

Allow `unprivileged` Containers Access

In the example above we saw that `card0` and `renderD128` are both owned by `root` and have their groups set to `video` and `render`. Because the "unprivilged" part of LXC unprivileged container works by mapping the UIDs (user IDs) and GIDs (group IDs) in the LXC guest namespace to an unused range of IDs on host, it is necessary to create a custom mapping for that namespace that maps those groups in the LXC guest namespace to the host groups while leaving the rest unchanged so you don't lose the added security of running an unprivilged container.

First you need to give root permission to map the group IDs. You can look in `/etc/group` to find the GIDs of those groups, but in this example `video` = `44` and `render` = `108` on our host system. You should add the following lines that allow `root` to map those groups to a new GID.

```
# /etc/subgid
+ root:44:1
+ root:108:1
```

Then you'll need to create the ID mappings. Since you're just dealing with group mappings the UID mapping can be performed in a single line as shown on the first line addition below. It can be read as "remap `65,536` of the LXC guest namespace UIDs from `0` through `65,536` to a range in the host starting at `100,000`." You can tell this relates to UIDs because of the `u` denoting users. It wasn't necessary to edit `/etc/subuid` because that file already gives root permission to perform this mapping.

You have to do the same thing for groups which is the same concept but slightly more verbose. In this example when looking at `/etc/group` in the LXC guest it shows that `video` and `render` have GIDs of `44` and `106`. Although you'll use `g` to denote GIDs everything else is the same except it is necessary to ensure the custom mappings cover the whole range of GIDs so it requires more lines. The only tricky part is the second to last line that shows mapping the LXC guest namespace GID for `render` (`106`) to the host GID for `render` (`108`) because the groups have different GIDs.

```
# /etc/pve/lxc/*.conf
  lxc.cgroup.devices.allow: c 226:* rwm
  lxc.mount.entry: /dev/dri/card0 dev/dri/card0 none bind,optional,create=file,mode=0666
  lxc.mount.entry: /dev/dri/renderD128 dev/dri/renderD128 none bind,optional,create=file
+ lxc.idmap: u 0 100000 65536
+ lxc.idmap: g 0 100000 44
+ lxc.idmap: g 44 44 1
+ lxc.idmap: g 45 100045 61
+ lxc.idmap: g 106 108 1
```

```
+ lxc.idmap: g 107 100107 65429
```

Beaues it can get confusing to read I just wanted show each line with some comments...

```
+ lxc.idmap: u 0 100000 65536    // map UIDs 0-65536 (LXC namespace) to 100000-165535 (host namespace)
+ lxc.idmap: g 0 100000 44       // map GIDs 0-43 (LXC namspace) to 100000-100043 (host namespace)
+ lxc.idmap: g 44 44 1           // map GID  44 to be the same in both namespaces
+ lxc.idmap: g 45 100045 61      // map GIDs 45-105 (LXC namspace) to 100045-100105 (host namespace)
+ lxc.idmap: g 106 108 1         // map GID  106 (LXC namspace) to 108 (host namespace)
+ lxc.idmap: g 107 100107 65429  // map GIDs 107-65536 (LXC namspace) to 100107-165536 (host namespace)
```

Add `root` to Groups

Because `root`'s UID and GID in the LXC guest's namespace isn't mapped to `root` on the host you'll have to add any users in the LXC guest to the groups `video` and `render` to have access the devices. As an example to give `root` in our LXC guest's namespace access to the devices you would simply add `root` to the `video` and `render` group.

```
usermod --append --groups video,render root
```

Potential Alernative

lxc.mount.entry - static uid/gid in LXC guest

Resources

Proxmox: Unprivileged LXC containers

Revision #15
Created 13 May 2020 11:45:33 by Dustin Sweigart
Updated 12 October 2021 00:43:10 by dustin@swigg.net