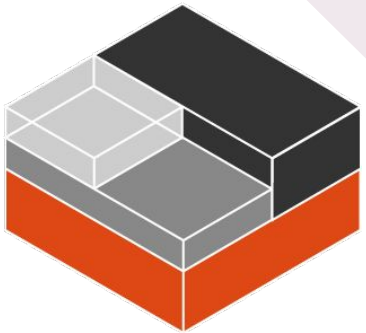
Idmapped Mounts

Flexible file ownership



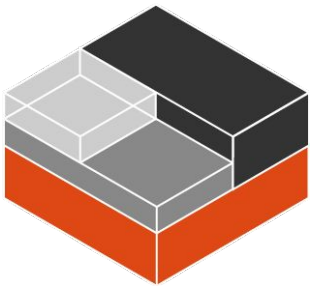
Christian Brauner
LXD maintainer & Kernel engineer

@brau_ner
https://brauner.io
christian.brauner@ubuntu.com

CANONICAL ubuntu

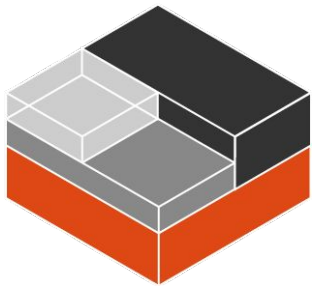
Outline

- File Ownership
- Changing File Ownership
- Inflexible Ownership Model
- File Ownership & User Namespaces
- Idmapped Mounts
  - Idea
  - Some Implementation Details
  - UAPI (Userspace API)
- Demo



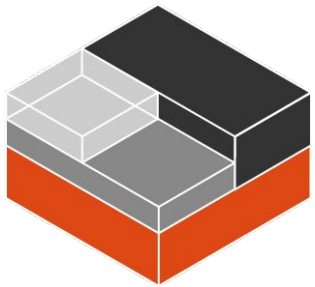
File Ownership

```
/*
 * Keep mostly read-only and often accessed (especially for
 * the RCU path lookup and 'stat' data) fields at the beginning
 * of the 'struct inode'
 */
struct inode {
      umode_t                 i_mode;
      unsigned short          i_opflags;
      kuid_t                  i_uid;
      kgid_t                  i_gid;
      unsigned int            i_flags;

#ifdef CONFIG_FS_POSIX_ACL
      struct posix_acl        *i_acl;
      struct posix_acl        *i_default_acl;
#endif

      const struct inode_operations *i_op;
      struct super_block*i_sb;
      struct address_space    *i_mapping;

#ifdef CONFIG_SECURITY
      void                    *i_security;
#endif

      /* Stat data, not accessed from path walking */
unsigned long                 i_ino;
```



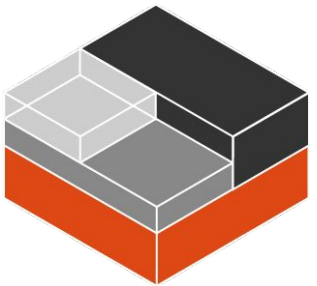
Changing File Ownership

```
SYSCALL_DEFINE5(fchownat, int, dfd, const char __user *, filename, uid_t, user,
                gid_t, group, int, flag)

SYSCALL_DEFINE3(chown, const char __user *, filename, uid_t, user, gid_t, group)


SYSCALL_DEFINE3(lchown, const char __user *, filename, uid_t, user, gid_t, group)


SYSCALL_DEFINE3(fchown, unsigned int, fd, uid_t, user, gid_t, group)
```



Inflexible File Ownership Model

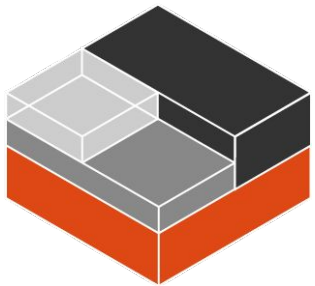
- changes to ownership through chown always global
- performance considerations when chowning large filesystems
- can't delegate ownership to multiple users with different uids and gids



File Ownership & User Namespaces

- important for building secure containers
- isolate uids and gids through idmappings
    - map uid 100000 outside of user namespace to uid 0 inside user namespace:

```
# outside user namespace
sudo ls -al /var/lib/lxd/containers/f1/rootfs/root/.bashrc
-rw-r--r-- 1 100000 100000 3106 Dec  5  2019

# inside user namespace
lxc exec f1 -- ls -al /root/.bashrc
-rw-r--r-- 1 root root 3106 Dec  5  2019 /root/.bashrc
```



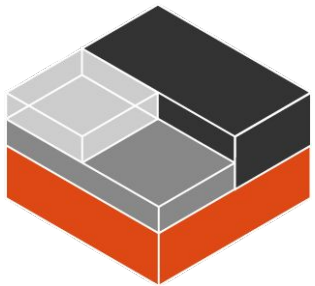
Idmapped Mounts

- `mount --bind /home /mnt1 # expose all files under /home at /mnt1`
- `mount /dev/sda /mnt2 # mount proper filesystem (create new superblock)`
  `mount --bind /mnt2 /mnt3 # expose all files under /mnt2 at /mnt3`
  `mount --bind /mnt2 /mnt4 # expose all files under /mnt3 at /mnt4`

- same ownership for all files in the mounts above

- `mount-idmapped --map-mount b:0:1000:1 /mnt2 /mnt5`

- all files owned by uid and gid 0 under /mnt2 will now be owned by uid and gid 1000 under /mnt5



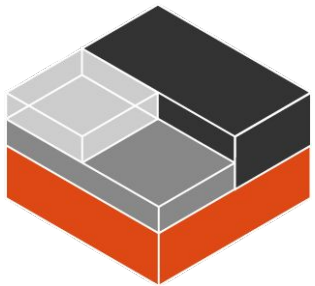
The mount-idmapped is a tool I have written. I expect the mount binary to simply gain support for this if the patchset lands.

Idmapped Mounts

- attach user namespaces to struct vfsmount in the kernel

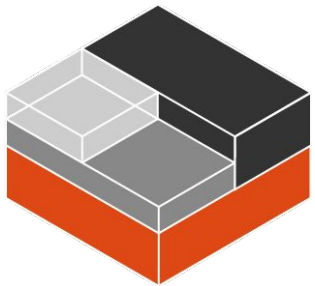
```
struct vfsmount {
        struct dentry *mnt_root; /* root of the mounted tree */
        struct super_block *mnt_sb;    /* pointer to superblock */
        int mnt_flags;
        struct user_namespace *mnt_userns;
} __randomize_layout;
```

- on-the fly efficient translation of i_uid and i_gid of the inode when needed (permission checks, allocation of new inode, ownership changes etc.)
- most changes done in the vfs
- patchsets to convert individual filesystems are small
- initial port contains ext4, xfs, and fat

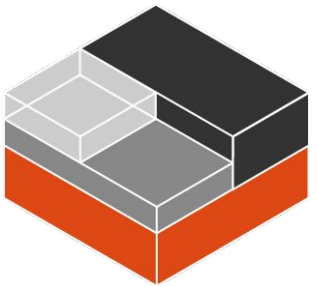


Idmapped Mounts

```
 struct mount_attr {
     __u64 attr_set;
     __u64 attr_clr;
     __u64 propagation;
     __u64 userns_fd;
};

SYSCALL_DEFINE5(mount_setattr, int, dfd, const char __user *, path,
          unsigned int, flags, struct mount_attr __user *, uattr,
          size_t, usize)

struct mount_attr attr = {
     .attr_set = MOUNT_ATTR_IDMAP,
};
attr.userns_fd = get_userns_fd(0, 10000, 10000);
mount_setattr(mount_fd, "", AT_EMPTY_PATH, &attr, sizeof(attr));
```



Demo



Questions ?

Christian Brauner
LXD maintainer & Kernel engineer

@brau_ner
https://brauner.io
christian.brauner@ubuntu.com