Subject: [patch -mm 10/17] nsproxy: add unshare_ns and bind_ns syscalls
Posted by Cedric Le Goater on Tue, 05 Dec 2006 10:28:02 GMT
View Forum Message <> Reply to Message

From: Cedric Le Goater <clg@fr.ibm.com>

The following patch defines 2 new syscalls specific to nsproxy and
namespaces :

* unshare_ns :

 enables a process to unshare one or more namespaces. this
       duplicates the unshare syscall for the moment but we
 expect to diverge when the number of namespaces increases

* bind_ns :

 allows a process to bind
 1 - its nsproxy to some identifier
 2 - to another nsproxy using an identifier or -pid

Here's a sample user space program to use them.

```
#include <stdio.h>
#include <stdlib.h>
#include <sched.h>
#include <unistd.h>
#include <string.h>
#include <errno.h>
#include <libgen.h>

#include <linux/unistd.h>

#if __i386__
#    define __NR_unshare_ns 324
#    define __NR_bind_ns  325
#elif __ia64__
#    define __NR_unshare_ns 1303
#    define __NR_bind_ns 1304
#elif __powerpc__
#    define __NR_unshare_ns 303
#    define __NR_bind_ns 304
#elif __s390x__
#    define __NR_unshare_ns 313
#    define __NR_bind_ns 314
#elif __x86_64__
#    define __NR_unshare_ns 284
#    define __NR_bind_ns 285
```

```
#else
#    error "Architecture not supported"
#endif

static inline _syscall1 (int,  unshare_ns, unsigned long, flags)
static inline _syscall2 (int,  bind_ns, int, id, unsigned long, flags)

#define NS_MNT  0x00000001
#define NS_UTS  0x00000002
#define NS_IPC  0x00000004
#define NS_PID  0x00000008
#define NS_NET  0x00000010
#define NS_USER  0x00000020

static const char* procname;

static void usage(const char *name)
{
 printf("usage: %s [-h] [-I id] [-muiUnp] [command [arg ...]]\n", name);
 printf("\n");
 printf("  -h  this message\n");
 printf("\n");
 printf("  -I <id> bind process to nsproxy <id>\n");
 printf("  -m  mount namespace\n");
 printf("  -u  utsname namespace\n");
 printf("  -i  ipc namespace\n");
 printf("  -U  user namespace\n");
 printf("  -n  net namespace\n");
 printf("  -p  pid namespace\n");
 printf("\n");
 printf("(C) Copyright IBM Corp. 2006\n");
 printf("\n");
 exit(1);
}

int main(int argc, char *argv[])
{
 int c;
 unsigned long flags = 0;
 int id = -1;

 procname = basename(argv[0]);

 while ((c = getopt(argc, argv, "+muiUnphI:")) != EOF) {
  switch (c) {
  case 'I': if (optarg)
    id = atoi(optarg); break;
```

```
  case 'm': flags |= NS_MNT;  break;
  case 'u': flags |= NS_UTS;  break;
  case 'i': flags |= NS_IPC;  break;
  case 'U': flags |= NS_USER; break;
  case 'n': flags |= NS_NET;  break;
  case 'p': flags |= NS_PID;  break;
  case 'h':
  default:
   usage(procname);
  }
 };

 argv = &argv[optind];
 argc = argc - optind;

 if (!strcmp(procname, "unsharens")) {
  if (unshare_ns(flags) == -1) {
   perror("unshare_ns");
   return 1;
  }
 }

 if (bind_ns(id, flags) == -1) {
  perror("bind_ns");
  return 1;
 }

 if (argc) {
  execve(argv[0], argv, __environ);
  perror("execve");
  return 1;
 }

 return 0;
}
```

Signed-off-by: Cedric Le Goater <clg@fr.ibm.com>

```
 arch/i386/kernel/syscall_table.S  |    2
 arch/ia64/kernel/entry.S          |    2
 arch/s390/kernel/compat_wrapper.S |   11 +
 arch/s390/kernel/syscalls.S       |    2
 arch/x86_64/ia32/ia32entry.S      |    2
 include/asm-i386/unistd.h         |    4
 include/asm-ia64/unistd.h         |    4
 include/asm-powerpc/systbl.h      |    2
 include/asm-powerpc/unistd.h      |    4
```

```
 include/asm-s390/unistd.h         |    4
 include/asm-x86_64/unistd.h       |    6
 include/linux/syscalls.h          |    3
 kernel/nsproxy.c                  |  325 ++++++++++++++++++++++++++++++++++++++-
 kernel/sys_ni.c                   |    4
 14 files changed, 366 insertions(+), 9 deletions(-)

Index: 2.6.19-rc6-mm2/include/linux/syscalls.h
===================================================================
--- 2.6.19-rc6-mm2.orig/include/linux/syscalls.h
+++ 2.6.19-rc6-mm2/include/linux/syscalls.h
@@ -605,6 +605,9 @@ asmlinkage long sys_set_robust_list(stru
        size_t len);
 asmlinkage long sys_getcpu(unsigned __user *cpu, unsigned __user *node, struct getcpu_cache
__user *cache);

+asmlinkage long sys_unshare_ns(unsigned long unshare_flags);
+asmlinkage long sys_bind_ns(int id, unsigned long unshare_flags);
+
 int kernel_execve(const char *filename, char *const argv[], char *const envp[]);

 asmlinkage long sys_kevent_get_events(int ctl_fd, unsigned int min, unsigned int max,
Index: 2.6.19-rc6-mm2/kernel/nsproxy.c
===================================================================
--- 2.6.19-rc6-mm2.orig/kernel/nsproxy.c
+++ 2.6.19-rc6-mm2/kernel/nsproxy.c
@@ -22,7 +22,11 @@
 #include <linux/pid_namespace.h>
 #include <linux/net_namespace.h>

-#define NS_HASH_BITS   3 /* this might need some configuration */
+/*
+ * nsproxies are stored in a hash but a rbtree might be more
+ * appropriate.
+ */
+#define NS_HASH_BITS   3
 #define NS_HASH_SIZE  (1 << NS_HASH_BITS)
 #define NS_HASH_MASK  (NS_HASH_SIZE - 1)
 #define ns_hashfn(id)  (((id >> NS_HASH_BITS) + id) & NS_HASH_MASK)
@@ -193,11 +197,26 @@ static void free_nsproxy(struct nsproxy
  kfree(ns);
 }

+/*
+ * put_nsproxy() is similar to free_uid() in kernel/user.c
+ *
+ * the lock can be taken from a tasklet context (task getting freed by
+ * RCU) which requires to be irq safe.
```

```
+ */
 void put_nsproxy(struct nsproxy *ns)
 {
- if (atomic_dec_and_test(&ns->count)) {
-  free_nsproxy(ns);
- }
+  unsigned long flags;
+
+  local_irq_save(flags);
+  if (atomic_dec_and_lock(&ns->count, &ns_hash_lock)) {
+   BUG_ON(!ns->id);
+   if (ns->id != -1)
+    hlist_del(&ns->ns_hash_node);
+   spin_unlock_irqrestore(&ns_hash_lock, flags);
+   free_nsproxy(ns);
+  } else {
+   local_irq_restore(flags);
+  }
 }

 /*
@@ -218,6 +237,304 @@ static inline struct nsproxy *ns_hash_fi
  return NULL;
 }

+static int bind_ns(int id, struct nsproxy *ns)
+{
+  struct nsproxy *prev;
+ int ret = 0;
+
+  if (id < 0)
+   return -EINVAL;
+
+  spin_lock_irq(&ns_hash_lock);
+  prev = ns_hash_find(id);
+  if (!prev) {
+  ns->id = id;
+  hlist_add_head(&ns->ns_hash_node, ns_hash_head(ns->id));
+ }
+ spin_unlock_irq(&ns_hash_lock);
+
+ if (prev) {
+  ret = -EBUSY;
+  put_nsproxy(prev);
+ }
+  return ret;
+}
+
```

```
+static int switch_ns(int id, unsigned long flags)
+{
+ int err = 0;
+ struct nsproxy *ns = NULL, *old_ns = NULL, *new_ns = NULL;
+
+ if (flags & ~NS_ALL)
+  return -EINVAL;
+
+ /* Let 0 be a default value ? */
+ if (!flags)
+  flags = NS_ALL;
+
+ if (id < 0) {
+  struct task_struct *p;
+
+  err = -ESRCH;
+  read_lock(&tasklist_lock);
+  p = find_task_by_pid(-id);
+  if (p) {
+   task_lock(p);
+   get_nsproxy(p->nsproxy);
+   ns = p->nsproxy;
+   task_unlock(p);
+  }
+  read_unlock(&tasklist_lock);
+ } else {
+  err = -ENOENT;
+  spin_lock_irq(&ns_hash_lock);
+  ns = ns_hash_find(id);
+  spin_unlock_irq(&ns_hash_lock);
+ }
+
+ if (!ns)
+  goto out;
+
+ new_ns = ns;
+
+ /*
+  * clone current nsproxy and populate it with the namespaces
+  * chosen by flags.
+  */
+ if (flags != NS_ALL) {
+  new_ns = dup_namespaces(current->nsproxy);
+  if (!new_ns) {
+   err = -ENOMEM;
+   goto out_ns;
+  }
+
```

```
+  if (flags & NS_MNT) {
+   put_mnt_ns(new_ns->mnt_ns);
+   get_mnt_ns(ns->mnt_ns);
+   new_ns->mnt_ns = ns->mnt_ns;
+  }
+
+  if (flags & NS_UTS) {
+   put_uts_ns(new_ns->uts_ns);
+   get_uts_ns(ns->uts_ns);
+   new_ns->uts_ns = ns->uts_ns;
+  }
+
+  if (flags & NS_IPC) {
+   put_ipc_ns(new_ns->ipc_ns);
+   new_ns->ipc_ns = get_ipc_ns(ns->ipc_ns);
+  }
+ out_ns:
+  put_nsproxy(ns);
+ }
+
+ task_lock(current);
+ if (new_ns) {
+  old_ns = current->nsproxy;
+  current->nsproxy = new_ns;
+ }
+ task_unlock(current);
+
+ if (old_ns)
+  put_nsproxy(old_ns);
+
+ err = 0;
+out:
+ return err;
+}
+
+
+/*
+ * bind_ns - bind the nsproxy of a task to an id or bind a task to a
+ *           identified nsproxy
+ *
+ * @id: nsproxy identifier if positive or pid if negative
+ * @flags: identifies the namespaces to bind to
+ *
+ * bind_ns serves 2 purposes.
+ *
+ * The first is to bind the nsproxy of the current task to the
+ * identifier @id. If the identifier is already used, -EBUSY is
+ * returned. If the nsproxy is already bound, -EACCES is returned.
```

```
+ * flags is not used in that case.
+ *
+ * The second use is to bind the current task to a subset of
+ * namespaces of an identified nsproxy. If positive, @id is considered
+ * being an nsproxy identifier previously used to bind the nsproxy to
+ * @id. If negative, @id is the pid of a task which is another way to
+ * identify a nsproxy. Switching nsproxy is restricted to tasks within
+ * nsproxy 0, the default nsproxy. If unknown, -ENOENT is returned.
+ * @flags is used to bind the task to the selected namespaces.
+ *
+ * Both uses may return -EINVAL for invalid arguments and -EPERM for
+ * insufficient privileges.
+ *
+ * Returns 0 on success.
+ */
+asmlinkage long sys_bind_ns(int id, unsigned long flags)
+{
+ struct nsproxy *ns = current->nsproxy;
+ int ret = 0;
+
+ /*
+  * ns is being changed by switch_ns(), protect it
+  */
+ get_nsproxy(ns);
+
+ /*
+  * protect ns->id
+  */
+ spin_lock(&ns->nslock);
+ switch (ns->id) {
+ case -1:
+  /*
+   * only an unbound nsproxy can be bound to an id.
+   */
+  ret = bind_ns(id, ns);
+  break;
+
+ case 0:
+  if (!capable(CAP_SYS_ADMIN)) {
+   ret = -EPERM;
+   goto unlock;
+  }
+
+  /*
+   * only nsproxy 0 can switch nsproxy. if target id is
+   * 0, this is a nop.
+   */
+  if (id)
```

```
+   ret = switch_ns(id, flags);
+  break;
+
+ default:
+  /*
+   * current nsproxy is already bound. forbid any
+   * switch.
+   */
+  ret = -EACCES;
+ }
+unlock:
+ spin_unlock(&ns->nslock);
+ put_nsproxy(ns);
+ return ret;
+}
+
+/*
+ * sys_unshare_ns - unshare one or more of namespaces which were
+ *                  originally shared using clone.
+ *
+ * @unshare_ns_flags: identifies the namespaces to unshare
+ *
+ * this is for the moment a pale copy of unshare but we expect to
+ * diverge when the number of namespaces increases. the number of
+ * available clone flags is also very low. This is one way to get
+ * some air.
+ *
+ */
+asmlinkage long sys_unshare_ns(unsigned long unshare_ns_flags)
+{
+  int err = 0;
+  struct nsproxy *new_nsproxy = NULL, *old_nsproxy = NULL;
+  struct fs_struct *fs, *new_fs = NULL;
+  struct mnt_namespace *mnt, *new_mnt = NULL;
+  struct uts_namespace *uts, *new_uts = NULL;
+  struct ipc_namespace *ipc, *new_ipc = NULL;
+  unsigned long unshare_flags = 0;
+
+  /* Return -EINVAL for all unsupported flags */
+  err = -EINVAL;
+  if (unshare_ns_flags & ~NS_ALL)
+   goto bad_unshare_ns_out;
+
+  /*
+   * compatibility with unshare()/clone() : convert ns flags to
+   * clone flags
+   */
+  if (unshare_ns_flags & NS_MNT)
```

```
+   unshare_flags |= CLONE_NEWNS|CLONE_FS;
+  if (unshare_ns_flags & NS_UTS)
+   unshare_flags |= CLONE_NEWUTS;
+  if (unshare_ns_flags & NS_IPC)
+   unshare_flags |= CLONE_NEWIPC;
+
+  if ((err = unshare_fs(unshare_flags, &new_fs)))
+   goto bad_unshare_ns_out;
+  if ((err = unshare_mnt_ns(unshare_flags, &new_mnt, new_fs)))
+   goto bad_unshare_ns_cleanup_fs;
+  if ((err = unshare_utsname(unshare_flags, &new_uts)))
+   goto bad_unshare_ns_cleanup_mnt;
+  if ((err = unshare_ipcs(unshare_flags, &new_ipc)))
+   goto bad_unshare_ns_cleanup_uts;
+
+  if (new_mnt || new_uts || new_ipc) {
+   old_nsproxy = current->nsproxy;
+   new_nsproxy = dup_namespaces(old_nsproxy);
+   if (!new_nsproxy) {
+    err = -ENOMEM;
+    goto bad_unshare_ns_cleanup_ipc;
+   }
+  }
+
+  if (new_fs || new_mnt || new_uts || new_ipc) {
+
+   task_lock(current);
+
+   if (new_nsproxy) {
+    current->nsproxy = new_nsproxy;
+    new_nsproxy = old_nsproxy;
+   }
+
+   if (new_fs) {
+    fs = current->fs;
+    current->fs = new_fs;
+    new_fs = fs;
+   }
+
+   if (new_mnt) {
+    mnt = current->nsproxy->mnt_ns;
+    current->nsproxy->mnt_ns = new_mnt;
+    new_mnt = mnt;
+   }
+
+   if (new_uts) {
+    uts = current->nsproxy->uts_ns;
+    current->nsproxy->uts_ns = new_uts;
```

```
+    new_uts = uts;
+   }
+
+   if (new_ipc) {
+    ipc = current->nsproxy->ipc_ns;
+    current->nsproxy->ipc_ns = new_ipc;
+    new_ipc = ipc;
+   }
+
+   task_unlock(current);
+  }
+
+  if (new_nsproxy)
+   put_nsproxy(new_nsproxy);
+
+bad_unshare_ns_cleanup_ipc:
+  if (new_ipc)
+   put_ipc_ns(new_ipc);
+
+bad_unshare_ns_cleanup_uts:
+  if (new_uts)
+   put_uts_ns(new_uts);
+
+bad_unshare_ns_cleanup_mnt:
+  if (new_mnt)
+   put_mnt_ns(new_mnt);
+
+bad_unshare_ns_cleanup_fs:
+  if (new_fs)
+   put_fs_struct(new_fs);
+
+bad_unshare_ns_out:
+  return err;
+}
+
 static int __init nshash_init(void)
 {
  int i;
Index: 2.6.19-rc6-mm2/kernel/sys_ni.c
===================================================================
--- 2.6.19-rc6-mm2.orig/kernel/sys_ni.c
+++ 2.6.19-rc6-mm2/kernel/sys_ni.c
@@ -146,3 +146,7 @@ cond_syscall(compat_sys_migrate_pages);
 cond_syscall(sys_bdflush);
 cond_syscall(sys_ioprio_set);
 cond_syscall(sys_ioprio_get);
+
+/* user resources syscalls */
```

```
+cond_syscall(sys_unshare_ns);
+cond_syscall(sys_bind_ns);
Index: 2.6.19-rc6-mm2/arch/i386/kernel/syscall_table.S
===================================================================
--- 2.6.19-rc6-mm2.orig/arch/i386/kernel/syscall_table.S
+++ 2.6.19-rc6-mm2/arch/i386/kernel/syscall_table.S
@@ -323,3 +323,5 @@ ENTRY(sys_call_table)
  .long sys_kevent_ctl
  .long sys_kevent_wait
  .long sys_kevent_ring_init
+ .long sys_unshare_ns
+ .long sys_bind_ns  /* 325 */
Index: 2.6.19-rc6-mm2/arch/ia64/kernel/entry.S
===================================================================
--- 2.6.19-rc6-mm2.orig/arch/ia64/kernel/entry.S
+++ 2.6.19-rc6-mm2/arch/ia64/kernel/entry.S
@@ -1610,5 +1610,7 @@ sys_call_table:
  data8 sys_sync_file_range  // 1300
  data8 sys_tee
  data8 sys_vmsplice
+ data8 sys_unshare_ns
+ data8 sys_bind_ns

  .org sys_call_table + 8*NR_syscalls // guard against failures to increase NR_syscalls
Index: 2.6.19-rc6-mm2/arch/s390/kernel/compat_wrapper.S
===================================================================
--- 2.6.19-rc6-mm2.orig/arch/s390/kernel/compat_wrapper.S
+++ 2.6.19-rc6-mm2/arch/s390/kernel/compat_wrapper.S
@@ -1665,3 +1665,14 @@ sys_getcpu_wrapper:
  llgtr %r3,%r3   # unsigned *
  llgtr %r4,%r4   # struct getcpu_cache *
  jg sys_getcpu
+
+ .globl sys_unshare_ns_wrapper
+sys_unshare_ns_wrapper:
+ llgfr   %r2,%r2              # unsigned long
+ jg      sys_unshare_ns
+
+ .globl sys_bind_ns_wrapper
+sys_bind_ns_wrapper:
+ lgfr    %r2,%r2              # int
+ llgfr   %r3,%r3              # unsigned long
+ jg      sys_bind_ns
Index: 2.6.19-rc6-mm2/arch/s390/kernel/syscalls.S
===================================================================
--- 2.6.19-rc6-mm2.orig/arch/s390/kernel/syscalls.S
+++ 2.6.19-rc6-mm2/arch/s390/kernel/syscalls.S
@@ -321,3 +321,5 @@ SYSCALL(sys_vmsplice,sys_vmsplice,compat
```

```
 NI_SYSCALL       /* 310 sys_move_pages */
 SYSCALL(sys_getcpu,sys_getcpu,sys_getcpu_wrapper)
 SYSCALL(sys_epoll_pwait,sys_epoll_pwait,sys_ni_syscall)
+SYSCALL(sys_unshare_ns,sys_unshare_ns,sys_unshare_ns_wrapper)
+SYSCALL(sys_bind_ns,sys_bind_ns,sys_bind_ns_wrapper)
Index: 2.6.19-rc6-mm2/arch/x86_64/ia32/ia32entry.S
===================================================================
--- 2.6.19-rc6-mm2.orig/arch/x86_64/ia32/ia32entry.S
+++ 2.6.19-rc6-mm2/arch/x86_64/ia32/ia32entry.S
@@ -722,4 +722,6 @@ ia32_sys_call_table:
  .quad sys_kevent_ctl  /* 320 */
  .quad sys_kevent_wait
  .quad sys_kevent_ring_init
+ .quad sys_unshare_ns
+ .quad sys_bind_ns
 ia32_syscall_end:
Index: 2.6.19-rc6-mm2/include/asm-i386/unistd.h
===================================================================
--- 2.6.19-rc6-mm2.orig/include/asm-i386/unistd.h
+++ 2.6.19-rc6-mm2/include/asm-i386/unistd.h
@@ -329,10 +329,12 @@
 #define __NR_kevent_ctl  321
 #define __NR_kevent_wait 322
 #define __NR_kevent_ring_init 323
+#define __NR_unshare_ns  324
+#define __NR_bind_ns  325

 #ifdef __KERNEL__

-#define NR_syscalls 324
+#define NR_syscalls 326

 #define __ARCH_WANT_IPC_PARSE_VERSION
 #define __ARCH_WANT_OLD_READDIR
Index: 2.6.19-rc6-mm2/include/asm-ia64/unistd.h
===================================================================
--- 2.6.19-rc6-mm2.orig/include/asm-ia64/unistd.h
+++ 2.6.19-rc6-mm2/include/asm-ia64/unistd.h
@@ -291,11 +291,13 @@
 #define __NR_sync_file_range  1300
 #define __NR_tee   1301
 #define __NR_vmsplice   1302
+#define __NR_unshare_ns   1303
+#define __NR_bind_ns   1304

 #ifdef __KERNEL__
```

```
-#define NR_syscalls   279 /* length of syscall table */
+#define NR_syscalls   281 /* length of syscall table */

 #define __ARCH_WANT_SYS_RT_SIGACTION

Index: 2.6.19-rc6-mm2/include/asm-powerpc/systbl.h
===================================================================
--- 2.6.19-rc6-mm2.orig/include/asm-powerpc/systbl.h
+++ 2.6.19-rc6-mm2/include/asm-powerpc/systbl.h
@@ -305,3 +305,5 @@ SYSCALL_SPU(faccessat)
 COMPAT_SYS_SPU(get_robust_list)
 COMPAT_SYS_SPU(set_robust_list)
 COMPAT_SYS(move_pages)
+SYSCALL_SPU(unshare_ns)
+SYSCALL_SPU(bind_ns)
Index: 2.6.19-rc6-mm2/include/asm-powerpc/unistd.h
===================================================================
--- 2.6.19-rc6-mm2.orig/include/asm-powerpc/unistd.h
+++ 2.6.19-rc6-mm2/include/asm-powerpc/unistd.h
@@ -324,10 +324,12 @@
 #define __NR_get_robust_list 299
 #define __NR_set_robust_list 300
 #define __NR_move_pages  301
+#define __NR_unshare_ns  302
+#define __NR_bind_ns  303

 #ifdef __KERNEL__

-#define __NR_syscalls  302
+#define __NR_syscalls  304

 #define __NR__exit __NR_exit
 #define NR_syscalls __NR_syscalls
Index: 2.6.19-rc6-mm2/include/asm-s390/unistd.h
===================================================================
--- 2.6.19-rc6-mm2.orig/include/asm-s390/unistd.h
+++ 2.6.19-rc6-mm2/include/asm-s390/unistd.h
@@ -250,8 +250,10 @@
 /* Number 310 is reserved for new sys_move_pages */
 #define __NR_getcpu  311
 #define __NR_epoll_pwait 312
+#define __NR_unshare_ns  313
+#define __NR_bind_ns  314

-#define NR_syscalls 313
+#define NR_syscalls 315

 /*
```

```
  * There are some system calls that are not present on 64 bit, some
Index: 2.6.19-rc6-mm2/include/asm-x86_64/unistd.h
===================================================================
--- 2.6.19-rc6-mm2.orig/include/asm-x86_64/unistd.h
+++ 2.6.19-rc6-mm2/include/asm-x86_64/unistd.h
@@ -627,8 +627,12 @@ __SYSCALL(__NR_kevent_ctl, sys_kevent_ct
 __SYSCALL(__NR_kevent_wait, sys_kevent_wait)
 #define __NR_kevent_ring_init 283
 __SYSCALL(__NR_kevent_ring_init, sys_kevent_ring_init)
+#define __NR_unshare_ns  284
+__SYSCALL(__NR_unshare_ns, sys_unshare_ns)
+#define __NR_bind_ns  285
+__SYSCALL(__NR_bind_ns, sys_bind_ns)

-#define __NR_syscall_max __NR_kevent_ring_init
+#define __NR_syscall_max __NR_bind_ns

 #ifndef __NO_STUBS
 #define __ARCH_WANT_OLD_READDIR

--

_____
Containers mailing list
Containers@lists.osdl.org
https://lists.osdl.org/mailman/listinfo/containers
```