Linux Containers

"Virtual machines" on OS level

Petr Baudiš ⟨pasky@ucw.cz⟩

Linux Containers

- OS level virtualization
- Multiple “hosts” with in-kernel separation
- The same kernel maintains multiple “tables” of processes, mount points, network interfaces etc.
- Processes in two containers are equal in kernel POV, but they can have different root filesystem, the same PID, see a different hostname, etc.
- Of course somewhat reduced security (kernel bugs break separation)

Namespaces

- Many kind of *names* in UNIX — **filesystem** names, **process** names, **network interface** names, **IPC** names, **utsname** hostnames
- Modern Linux system: namespace is (inherited) attribute of process, different namespaces may exist
- `chroot` — primite form of filesystem namespace separation
- Better form are mountpoint namespaces — each process can see their own `/proc/mounts` (incl. filesystem root)
- Lowlevel syscalls: **clone**, **unshare**

Control Groups

- A way to group processes on large scale
- Resources can be allocated per cgroup
- The whole control group can be frozen, checkpointed, resumed atomically

Containers in Practice: The "lxc" tool

- Let's start a fresh process (e.g. init) with new namespaces set up according to namespaces, in a new cgroup
- Isolation: Namespaces
- Control, resources allocation: cgroups
- Network: virtual ethernet interface + bridge
- **lxc** tool: Convenience wrapper for using containers; maintaining configuration, list, accounting of containers

“lxc” example

```
lxc.utsname = test
lxc.network.type = veth
lxc.network. ... (snip)
lxc.mount = /etc/lxc/lxc-test.fstab
lxc.rootfs = /srv/lxc/test
```

- debootstrap wheezy /srv/lxc/test
- lxc-create -f /etc/lxc/lxc-test.conf -n test
- lxc-start -n test

Docker: Containers Packaged

- Platform-as-a-Service backend by dotCloud
- Public registry of "images" (differential filesystem tarball + metadata)
- Easily download an image and start it in a container, with no special configuration
- Aimed to allow easy distribution of ready-to-run software running in a container
- Containers can be built by shell commands in existing container, or according to Dockerfile build recipe

- Still deep in development and very feature-incomplete, but usable
- Implemented in Go; server components only partially opensource yet

Docker Example

- `lxc-docker pull tianon/debian`
- `lxc-docker run -d tianon/debian python -m SimpleHTTPServer`
- `lxc-docker run -t -i tianon/debian ls -l /`
- `lxc-docker run -t -i tianon/debian ps axu`
- `lxc-docker run -t -i tianon/debian /bin/bash`
- `lxc-docker ls`
- `lxc-docker commit 12345 pasky/debian`
- `lxc-docker push pasky/debian`

That's all, folks!

happy hacking