systemd in Containers

Tokyo, Japan

June 2015

Containers

Containers

Docker, Rocket, LXC, libvirt-lxc, OpenVZ, ...

Containers

Docker, Rocket, LXC, libvirt-lxc, OpenVZ, ...

systemd-nspawn +

Containers

Docker, Rocket, LXC, libvirt-lxc, OpenVZ, ...

systemd-nspawn + systemd-machined +

Containers

Docker, Rocket, LXC, libvirt-lxc, OpenVZ, ...

systemd-nspawn + systemd-machined + systemd-importd

Containers as a part of the OS concept itself

Containers as a part of the OS concept itself

Inspiration: Solaris Zones

Containers as a part of the OS concept itself

Inspiration: Solaris Zones

OS running inside the container similar to OS outside of the container

Containers as a part of the OS concept itself

Inspiration: Solaris Zones

OS running inside the container similar to OS outside of the container

“Integrated Isolation”

Containers as a part of the OS concept itself

Inspiration: Solaris Zones

OS running inside the container similar to OS outside of the container

“Integrated Isolation”

Features that container systems provide, should also be available on the host system

Minimal

Minimal

Focus is on getting the low-level parts right

Minimal

Focus is on getting the low-level parts right

No hacks! Clean, integrated implementation matters for us

Minimal

Focus is on getting the low-level parts right

No hacks! Clean, integrated implementation matters for us

btrfs, no LVM

Minimal

Focus is on getting the low-level parts right

No hacks! Clean, integrated implementation matters for us

btrfs, no LVM

Defined execution environment that is close to what we expect for the host OS

Minimal

Focus is on getting the low-level parts right

No hacks! Clean, integrated implementation matters for us

btrfs, no LVM

Defined execution environment that is close to what we expect for the host OS

Open doors for alternatives

Avoid defining new standards where there already are standards

Avoid defining new standards where there already are standards

We test systemd itself daily in a container. In fact, we test it more often in a container than on bare metal

Avoid defining new standards where there already are standards

We test systemd itself daily in a container. In fact, we test it more often in a container than on bare metal

Cluster-wide orchestration is not the focus

Avoid defining new standards where there already are standards

We test systemd itself daily in a container. In fact, we test it more often in a container than on bare metal

Cluster-wide orchestration is not the focus – But it makes sense to built it on top

Avoid defining new standards where there already are standards

We test systemd itself daily in a container. In fact, we test it more often in a container than on bare metal

Cluster-wide orchestration is not the focus – But it makes sense to built it on top

Rocket makes use of nspawn for the actual containerization

machinectl + systemd-machined

machinectl + systemd-machined

Any container or VM manager can register its machines with it

machinectl + systemd-machined

Any container or VM manager can register its machines with it

Integration with systemctl -M, systemctl -r, systemctl list-machines, loginctl -M, journalctl -M, journalctl -m, ...

machinectl + systemd-machined

Any container or VM manager can register its machines with it

Integration with systemctl -M, systemctl -r, systemctl list-machines, loginctl -M, journalctl -M, journalctl -m, ...

Integration with ps, gnome-system-monitor, ...

machinectl + systemd-machined

Any container or VM manager can register its machines with it

Integration with systemctl -M, systemctl -r, systemctl list-machines, loginctl -M, journalctl -M, journalctl -m, ...

Integration with ps, gnome-system-monitor, ...

systemd-run -M, machinectl login, machinectl stop, ...

machinectl + systemd-machined

Any container or VM manager can register its machines with it

Integration with systemctl -M, systemctl -r, systemctl list-machines, loginctl -M, journalctl -M, journalctl -m, ...

Integration with ps, gnome-system-monitor, ...

systemd-run -M, machinectl login, machinectl stop, ...

Automatic host name resolution (using nss-mycontainers)

machinectl + systemd-machined

Any container or VM manager can register its machines with it

Integration with systemctl -M, systemctl -r, systemctl list-machines, loginctl -M, journalctl -M, journalctl -m, ...

Integration with ps, gnome-system-monitor, ...

systemd-run -M, machinectl login, machinectl stop, ...

Automatic host name resolution (using nss-mycontainers)

sd-bus D-Bus API is container-aware

systemd-nspawn

systemd-nspawn

Minimal container manager, integrates with systemd-machined

systemd-nspawn

Minimal container manager, integrates with systemd-machined

You pass it a directory, and it will boot it

systemd-nspawn

Minimal container manager, integrates with systemd-machined

You pass it a directory, and it will boot it

Preferred container directory is /var/lib/machines

systemd-nspawn

Minimal container manager, integrates with systemd-machined

You pass it a directory, and it will boot it

Preferred container directory is /var/lib/machines

Also, disassembles GPT partition tables, and boots raw disks

systemd-nspawn

Minimal container manager, integrates with systemd-machined

You pass it a directory, and it will boot it

Preferred container directory is /var/lib/machines

Also, disassembles GPT partition tables, and boots raw disks

Container-as-a-service

systemd-nspawn

Minimal container manager, integrates with systemd-machined

You pass it a directory, and it will boot it

Preferred container directory is /var/lib/machines

Also, disassembles GPT partition tables, and boots raw disks

Container-as-a-service, literally: systemd-nspawn@.service

systemd-nspawn

Minimal container manager, integrates with systemd-machined

You pass it a directory, and it will boot it

Preferred container directory is /var/lib/machines

Also, disassembles GPT partition tables, and boots raw disks

Container-as-a-service, literally: systemd-nspawn@.service

Resource Management like for normal services: systemctl set-propery systemd-nspawn@foobar.service CPUShares=100

systemd-networkd

systemd-networkd

Container support by default:

systemd-networkd

Container support by default:

When run on host, when a new veth tunnel to a container appears, automatically picks an unused IP range for it, and runs a DHCP server on it, as well as IPv4LL, configure IP Masquerading.

systemd-networkd

Container support by default:

When run on host, when a new veth tunnel to a container appears, automatically picks an unused IP range for it, and runs a DHCP server on it, as well as IPv4LL, configure IP Masquerading.

When run in container, and it sees a tunnel to the host, automatically runs a DHCP client on it, as well as IPv4LL.

systemd-resolved

systemd-resolved

Register host name by default via LLMNR, regardless if run in container or host.

systemd-resolved

Register host name by default via LLMNR, regardless if run in container or host.

LLMNR name lookups by default

systemd-resolved

Register host name by default via LLMNR, regardless if run in container or host.

LLMNR name lookups by default

systemd-networkd + systemd-resolved in container and on host: connectivity just works, with name resolution both ways.

systemd-importd

systemd-importd

Import container images from the Internet or locally, export them locally.

systemd-importd

Import container images from the Internet or locally, export them locally.

Formats: .tar or .raw (also, import-only: dkr)

systemd-importd

Import container images from the Internet or locally, export them locally.

Formats: .tar or .raw (also, import-only: dkr)

machinectl pull-raw –verify=no http://ftp.halifax.rwth-aachen.de/fedora/linux/releases/21/Cloud/Images/x86_64/Fedora-Cloud-Base-20141203-21.x86_64.raw.xz

systemd-importd

Import container images from the Internet or locally, export them locally.

Formats: .tar or .raw (also, import-only: dkr)

machinectl pull-raw –verify=no http://ftp.halifax.rwth-aachen.de/fedora/linux/releases/21/Cloud/Images/x86_64/Fedora-Cloud-Base-20141203-21.x86_64.raw.xz

systemd-nspawn -M Fedora-Cloud-Base-20141203-21

Stateless Systems:

Stateless Systems:

Make /usr sufficient to boot up system

Stateless Systems:

Make /usr sufficient to boot up system

Boot once with empty /etc and/or /var for factory reset

Stateless Systems:

Make /usr sufficient to boot up system

Boot once with empty /etc and/or /var for factory reset

Boot each time with empty /etc and/or /var for stateless systems

Stateless Systems:

Make /usr sufficient to boot up system

Boot once with empty /etc and/or /var for factory reset

Boot each time with empty /etc and/or /var for stateless systems

Mount the same /usr into many systems, for golden master systems, with central updating

Stateless Systems:

Make /usr sufficient to boot up system

Boot once with empty /etc and/or /var for factory reset

Boot each time with empty /etc and/or /var for stateless systems

Mount the same /usr into many systems, for golden master systems, with central updating

systemd-nspawn –volatile=

Assorted features:

Assorted features:

machinectl clone

Assorted features:

machinectl clone

machinectl set-limit

Assorted features:

machinectl clone

machinectl set-limit

machinectl copy-from

Assorted features:

machinectl clone

machinectl set-limit

machinectl copy-from

machinectl copy-to

Assorted features:

machinectl clone

machinectl set-limit

machinectl copy-from

machinectl copy-to

machinectl bind

Assorted features II:

Assorted features II:

systemd-nspawn –ephemeral

Assorted features II:

systemd-nspawn –ephemeral

systemd-nspawn –port=

That's all, folks!