Portability System Services

Linux Plumbers 2016, Santa Fe

November 2016
Containers?
Containers?

Resource Bundles + Isolation + Delivery
Portable Services:
Portable Services:
Resource Bundles + Integration + Sandboxing
Portable Services:
Resource Bundles + Integration + Sandboxing
Modular
Consider Range:  *Integrated*  $\rightarrow$  *Isolated*:  

**Portable System Services**
Consider Range: *Integrated* → *Isolated*:

- Classic System Services → *Portable System Services*
- Docker-style micro services → Full OS containers á la LXC
- VMs á la KVM
Consider Range: *Integrated* $\rightarrow$ *Isolated*:

Classic System Services $\rightarrow$ Portable System Services $\rightarrow$

Docker-style micro services $\rightarrow$ Full OS containers á la LXC $\rightarrow$ VMs á la KVM

Consider what’s shared, not shared: Networking, File System, PID Namespace, Init System, Device Access, Logging
Why?
Why?

Next step for service management
Why?

Next step for service management

Everything already has a systemd service file
Why?

Next step for service management

Everything already has a systemd service file

Admins are used to services already, let’s just make them more powerful
Why?

Next step for service management

Everything already has a systemd service file

Admins are used to services already, let’s just make them more powerful

“Superprivileged Containers”
Why?

Next step for service management

Everything already has a systemd service file

Admins are used to services already, let’s just make them more powerful

“Superprivileged Containers”

Integration is good, not bad (frequently at least)
Three supported service formats: SysV, Native, Portable
Let's avoid defining something new (instead: simple directory tree/subvolume, or GPT containing squashfs)

Services run directly from it (think: RootImage=, similar to RootDirectory=)

Let's fix chroot()!
Disk Images

Let's avoid defining something new
Disk Images

Let's avoid defining something new (instead: simple directory tree/subvolume, or GPT containing squashfs)
Disk Images

Let's avoid defining something new (instead: simple directory tree/subvolume, or GPT containing squashfs)

Services run directly from it (think: RootImage=, similar to RootDirectory=)
Disk Images

Let’s avoid defining something new (instead: simple directory tree/subvolume, or GPT containing squashfs)

Services run directly from it (think: RootImage=, similar to RootDirectory=)

Let’s fix chroot()!
Sandboxing: PrivateDevices=, PrivateNetwork=, DynamicUser=, RemoveIPC=, PrivateTmp=, PrivateUsers=, ProtectSystem=, ProtectHome=, SystemCallFilter=, RestrictAddressFamilies=, RuntimeDirectory=, RestrictRealtime=, ProtectKernelModules=, ProtectKernelTunables=, ProtectControlGroups=, ...
Sandboxing: PrivateDevices=, PrivateNetwork=, DynamicUser=, RemoveIPC=, PrivateTmp=, PrivateUsers=, ProtectSystem=, ProtectHome=, SystemCallFilter=, RestrictAddressFamilies=, RuntimeDirectory=, RestrictRealtime=, ProtectKernelModules=, ProtectKernelTunables=, ProtectControlGroups=, ...

More to come: ProtectKernelLogs=, ProtectClock=, ProtectTracing=, ProtectMount=, ProtectKeyRing=, DataDirectory=, CacheDirectory=, LogDirectory=, RestrictNamespaces=, ...
Sandboxing: PrivateDevices=, PrivateNetwork=, DynamicUser=, RemoveIPC=, PrivateTmp=, PrivateUsers=, ProtectSystem=, ProtectHome=, SystemCallFilter=, RestrictAddressFamilies=, RuntimeDirectory=, RestrictRealtime=, ProtectKernelModules=, ProtectKernelTunables=, ProtectControlGroups=, ...

More to come: ProtectKernelLogs=, ProtectClock=, ProtectTracing=, ProtectMount=, ProtectKeyRing=, DataDirectory=, CacheDirectory=, LogDirectory=, RestrictNamespaces=, ...

Per-Service Firewalling and Accounting
Sandboxing: PrivateDevices=, PrivateNetwork=, DynamicUser=, RemoveIPC=, PrivateTmp=, PrivateUsers=, ProtectSystem=, ProtectHome=, SystemCallFilter=, RestrictAddressFamilies=, RuntimeDirectory=, RestrictRealtime=, ProtectKernelModules=, ProtectKernelTunables=, ProtectControlGroups=, ...

More to come: ProtectKernelLogs=, ProtectClock=, ProtectTracing=, ProtectMount=, ProtectKeyRing=, DataDirectory=, CacheDirectory=, LogDirectory=, RestrictNamespaces=, ...

Per-Service Firewalling and Accounting

For portable services (unlike for native and SysV): Sandboxing is opt-out, not opt-in!
Hard problems:
Hard problems:
Dynamic Users
Hard problems:
Dynamic Users
User Database mismatch
Hard problems:
Dynamic Users
User Database mismatch
D-Bus, . . .
In scope: Simple delivery, Verification, Simple building, Versioning, Socket activation, ...
In scope: Simple delivery, Verification, Simple building, Versioning, Socket activation, . . .

Out of Scope: Load distribution/migration à la fleetd, Cluster deployment, claim we’d define a universal API, server side functionality, desktop stuff
Example:
Example:

systemctl start https://myservicerepo.org/foobar.psi
Example:

```bash
systemctl start https://myservicerepo.org/foobar.psi
systemctl status foobar
```

Portable System Services
Example:

```sh
systemctl start https://myservicerepo.org/foobar.psi
systemctl status foobar
systemctl stop foobar
```
Example:

```bash
systemctl start https://myservicerepo.org/foobar.psi
systemctl status foobar
systemctl stop foobar
systemctl purge foobar
```
Example:

```bash
systemctl start https://myservicerepo.org/foobar.psi
systemctl status foobar
systemctl stop foobar
systemctl purge foobar
systemctl start -H otherhost
https://myservicerepo.org/foobar.psi
```
Example:
```
systemctl start https://myservicerepo.org/foobar.psi
systemctl status foobar
systemctl stop foobar
systemctl purge foobar
systemctl start -H otherhost https://myservicerepo.org/foobar.psi
systemctl start -H otherhost ./workproject.psi
```
That’s all, folks!