Containers without a Container Manager, with systemd

Linux Piter 2017, St. Petersburg, Russia

November 2017

Containers?

Containers?

. . . Resource Bundling,

Containers?

. . . Resource Bundling, Sandboxing,

Containers?

. . . Resource Bundling, Sandboxing, Delivery

Containers?

. . . Resource Bundling, Sandboxing, Delivery

Let's focus on the first two!

Resource Bundling

```
RootDirectory=
```

```
RootDirectory=

RootImage=
```

Images for `RootImage=`?

Images for `RootImage=`?

Discoverable GPT,

Images for `RootImage=`?

Discoverable GPT, unambiguous GPT or MBR,

Images for `RootImage=`?

Discoverable GPT, unambiguous GPT or MBR, raw file system

Images for `RootImage=`?

Discoverable GPT, unambiguous GPT or MBR, raw file system

For example, create it with `mkosi`

Images for `RootImage=`?

Discoverable GPT, unambiguous GPT or MBR, raw file system

For example, create it with `mkosi`

`dm-verity`, LUKS, ...

`RootImage=/RootDirectory=`

`RootImage=/RootDirectory=`

Fancy `chroot()`

`RootImage=/RootDirectory=`

Fancy `chroot()`

Inherits the same problems

MountAPIVFS=

How to share data?

How to share data?

`BindPaths=`, `BindReadOnlyPaths=`

How to share data?

`BindPaths=`, `BindReadOnlyPaths=`

`RuntimeDirectory=`, `StateDirectory=`, `CacheDirectory=`, `LogsDirectory`, `ConfigurationDirectory=`

How to share data?

`BindPaths=`, `BindReadOnlyPaths=`

`RuntimeDirectory=`, `StateDirectory=`, `CacheDirectory=`, `LogsDirectory`, `ConfigurationDirectory=`

Also nice to keep bundles self-contained, no need for `tmpfiles.d/`

How to share user table?

How to share user table?

Not at all: use `PrivateUsers=`, to disconnect the tables

How to share user table?

Not at all: use `PrivateUsers=`, to disconnect the tables

`nss-systemd` synthesizes user entries for both `root` and `nobody`, the user IDs always needed, and always defined the same way.

How to share user table?

Not at all: use `PrivateUsers=`, to disconnect the tables

`nss-systemd` synthesizes user entries for both `root` and `nobody`, the user IDs always needed, and always defined the same way.

Missing: how to make sure the host and the bundle environment can share the service's own user identity?

Sandboxing

Established way to sandbox UNIX services:

Established way to sandbox UNIX services: UNIX users

Established way to sandbox UNIX services: UNIX users

`DynamicUser=`

Established way to sandbox UNIX services: UNIX users

`DynamicUser=`

Also nice to keep bundles self-contained, no need for `sysusers.d/`

Established way to sandbox UNIX services: UNIX users

`DynamicUser=`

Also nice to keep bundles self-contained, no need for `sysusers.d/`

No artifacts

```
RemoveIPC=
```

```
RemoveIPC=

PrivateTmp=
```

`RemoveIPC=`

`PrivateTmp=`

No artifacts

PrivateDevices=

```
PrivateDevices=

PrivateNetwork=
```

```
PrivateDevices=

PrivateNetwork=

IPAddressAllow=/IPAddressDeny=
```

ProtectKernelTunables=

```
ProtectKernelTunables=

ProtectKernelModules=
```

```
ProtectKernelTunables=
ProtectKernelModules=
ProtectControlGroups=
```

```
SystemCallFilter=
```

`SystemCallFilter=`

Now with system call groups!

```
RestrictAddressFamilies=
```

```
RestrictAddressFamilies=

SystemCallArchitectures=
```

```
RestrictAddressFamilies=

SystemCallArchitectures=

RestrictNamespaces=
```

```
RestrictAddressFamilies=

SystemCallArchitectures=

RestrictNamespaces=

LockPersonality=
```

```
RestrictAddressFamilies=

SystemCallArchitectures=

RestrictNamespaces=

LockPersonality=

MemoryDenyWriteExecute=
```

```
RestrictAddressFamilies=
SystemCallArchitectures=
RestrictNamespaces=
LockPersonality=
MemoryDenyWriteExecute=
RestrictRealtime=
```

```
RestrictAddressFamilies=
SystemCallArchitectures=
RestrictNamespaces=
LockPersonality=
MemoryDenyWriteExecute=
RestrictRealtime=
KeyringMode=
```

Outlook: portable service generator

Outlook: portable service generator

Iterate through bundle images/dirs in `/var/lib/portables`, extract relevant unit files, make them available on the host

Outlook: portable service generator

Iterate through bundle images/dirs in `/var/lib/portables`, extract relevant unit files, make them available on the host

Extend them with `RootImage=`/`RootDirectory=`, plus sandboxing options, via `.service.d/` drop-ins

Outlook: portable service generator

Iterate through bundle images/dirs in `/var/lib/portables`, extract relevant unit files, make them available on the host

Extend them with `RootImage=`/`RootDirectory=`, plus sandboxing options, via `.service.d/` drop-ins

Result: bundled images, containing service code, made available locally easily like native services

Missing: `pidns`? `hidepid`?

Missing: `pidns`? `hidepid`?

`systemctl purge`?

Open for everybody

Open for everybody

Use externally managed network namespace

Open for everybody

Use externally managed network namespace

Use externally managed eBPF programs

Open for everybody

Use externally managed network namespace

Use externally managed eBPF programs

Use externally managed console TTY

That's all, folks!