



Rootless Containers with `runc`

Aleksa Sarai
Software Engineer
SUSE

asarai@suse.de



“Required Reading”

- “Kubernetes meets Linux” — **Vishnu Kannan** — 1030 yesterday.
- “Containers from scratch - the sequel!” — **Liz Rice** — 1235 yesterday.
- “OCI and Open Container Standards” — **Jonathan Boule** — 1645 yesterday.
- “Mixing cgroupfs v1 and cgroupfs v2” — **Christian Brauner** — T-plus 35 minutes.

- *It is about pianos? pianolas?*

Revision: Open Container Initiative

- *It is about pianos? pianolas?* No.

Revision: Open Container Initiative

- *It is about pianos? pianolas?* No.
- Standards body created in 2015 to standardise container formats and runtimes.
- Two main components:
 - Runtime configuration.
 - Image format.
- **runc** is the de-facto implementation of the runtime specification.
 - It just needs a root filesystem and configuration file.
- ... and it's the runtime used by Docker and containerd.

A Hypothetical Researcher ...

- Researcher wants to run Python 3 code on a Python 2 cluster.

A Hypothetical Researcher ...

- Researcher wants to run Python 3 code on a Python 2 cluster.
- So they use a container to package Python 3 — right?
 - But the administrator (quite rightly) doesn't want to install “new-fangled software”.

A Hypothetical Researcher ...

- Researcher wants to run Python 3 code on a Python 2 cluster.
- So they use a container to package Python 3 — right?
 - But the administrator (quite rightly) doesn't want to install “new-fangled software”.
- Okay, so they compile all of the dependencies from scratch — right?
 - *Ha, ha.*

A Hypothetical Researcher ...

- Researcher wants to run Python 3 code on a Python 2 cluster.
- So they use a container to package Python 3 — right?
 - But the administrator (quite rightly) doesn't want to install “new-fangled software”.
- Okay, so they compile all of the dependencies from scratch — right?
 - *Ha, ha.*
- What should our researcher do?
 - What if we could create and run containers without privileges?

A Hypothetical Researcher ...

- Researcher wants to run Python 3 code on a Python 2 cluster.
- So they use a container to package Python 3 — right?
 - But the administrator (quite rightly) doesn't want to install “new-fangled software”.
- Okay, so they compile all of the dependencies from scratch — right?
 - *Ha, ha.*
- What should our researcher do?
 - What if we could create and run containers without privileges?
- Not actually a hypothetical — this was me in early 2016.
- ... and many other researchers have this problem.

Revision: The stuff that containers are made of

- Containers are mostly made of Linux kernel namespaces.
 - **cgroups** are not actually required.
- We just want isolation — and we want it without privileges.
- The key kernel feature is **USER** namespaces.
 - You can “pretend” that an unprivileged user is root.
 - I’ve already given a talk about how it works.
 - New kernels (3.8) let you create this namespace as an unprivileged user.

- LXC — still requires privileged binaries and processes to run things.
- `bubblewrap` — not a full container runtime.

- LXC — still requires privileged binaries and processes to run things.
- **bubblewrap** — not a full container runtime.
- Or you could do it by hand (Liz-style) ...

- LXC — still requires privileged binaries and processes to run things.
- **bubblewrap** — not a full container runtime.
- Or you could do it by hand (Liz-style) ...
 - `unshare -UrmunipCf bash`
 - `mount --make-rprivate / && mount --rbind rootfs/ rootfs/`
 - `mount -t proc proc rootfs/proc`
 - `mount -t tmpfs tmpfs rootfs/dev`
 - `mount -t devpts -o newinstance devpts rootfs/dev/pts`
 - `# ... skipping over a lot more mounting ...`
 - `pivot_root rootfs/ rootfs/.pivot_root && cd /`
 - `mount --make-rprivate /.pivot_root && umount -l /.pivot_root`
 - `exec bash # finally`

What works?

- **runc** recently got support for rootless containers.
- Due to kernel limitations and our requirements, some things are not possible.

Works	Broken
<code>run</code>	<code>checkpoint [criu]</code>
<code>exec</code>	<code>restore [criu]</code>
<code>kill</code>	<code>pause [cgroups]</code>
<code>delete</code>	<code>resume [cgroups]</code>
<code>list</code>	<code>events [cgroups]</code>
<code>state</code>	<code>ps [cgroups]</code>
<code>spec</code>	
<code>create</code>	
<code>start</code>	
<code>--detach</code>	

What doesn't?

- Checkpoint and restore isn't well-tested and still needs kernel work.
 - Unprivileged live migration of any process!
- **cgroups** — Let's not go there. 30 minutes is too short for the full rant, and we're all far too sober.
 - *Not every hierarchy should be a VFS.*
- Network namespaces aren't really useful (so we don't use them).

What about images?

- Runtime is only half of the story — what about images?
- Recently, two tools have been created to make this very easy.
 - skopeo Download and convert images from various sources and registries.
 - umoci Unpack, repack and otherwise modify local OCI images.
- Now you can use images as an unprivileged user too!
- You can even build **Dockerfile** images with orca-build.



Live Demo!

May the demo gods have mercy.

[Rootless Containers] # _

Rootless Containers

Overview

Rootless containers refers to the ability for an unprivileged user to create, run and otherwise manage

rootlesscontaine.rs

- Currently only **btrfs** supports unprivileged “subvolume” operations.
 - Ubuntu ships **overlayfs** allowing unprivileged mounting.
- The demo didn’t create a new **NET** namespace — it used the hosts’s.
 - ...but what if we want to use cool networking and still have network access?
 - I have some horrific ideas with implementing **veth** in user-space.

Unprivileged clusters!

- I have access to a bunch of nodes and want to manage jobs across those, what do I do?
- Biggest blocker looks to be networking (see **kube-proxy**).
- Most orchestrators assume root and it's hard to work around those assumptions upstream.

Unprivileged clusters!

The screenshot shows the GitHub interface for the repository `cloudfoundry/garden-runc-release`. At the top, there are buttons for `Watch` (64), `Star` (24), and `Fork` (25). Below these are tabs for `Code`, `Issues` (2), `Pull requests` (0), `Projects` (0), `Wiki`, and `Insights`. The file path `garden-runc-release / docs / rootless-containers.md` is displayed, along with a `Branch: develop` dropdown and buttons for `Find file` and `Copy path`. A commit by `williammartin` with the message `wip` is shown, dated 14 days ago. Below this, it says `5 contributors` with their avatars. The file statistics are `195 lines (151 sloc) | 6.97 KB`. At the bottom of the file view, there are buttons for `Raw`, `Blame`, and `History`, along with edit and delete icons.

cloudfoundry / garden-runc-release

Watch 64 Star 24 Fork 25

<> Code Issues 2 Pull requests 0 Projects 0 Wiki Insights

Branch: develop garden-runc-release / docs / rootless-containers.md Find file Copy path

williammartin wip 02dcaf0 14 days ago

5 contributors

195 lines (151 sloc) | 6.97 KB Raw Blame History

(Experimental) Rootless containers in Garden

With the latest releases of Garden it is now possible to create and run processes in containers without requiring root privileges! This document details the various components that enable this to work, as well as providing a step-by-step guide for installing and configuring the latest Garden dev builds to run as a non-root user.

HUGE DISCLAIMER: Garden's support for rootless containers is still very much a work-in-progress at the moment, and as such is subject to a number of known limitations (see the end of this doc for details).

<https://github.com/cloudfoundry/garden-runc-release>

Contain absolutely everything!!

- Why stop at servers, why can't we run all of our things inside containers.
- Flatpak is working on this, but is not going all the way with containers.
- If you can run it as your user, it should be able to work in rootless containers.
 - *Unprivileged overlay might have some interesting properties here ...*
- *Ultimate convergenceTM: What if you could use container features in desktop applications?*

Contain absolutely everything!!

Ultimate Linux on the Desktop

Monday, January 16, 2017

<https://blog.jessfraz.com/post/ultimate-linux-on-the-desktop/>

Acknowledgements

- **Jessie Frazelle** — started working on this first and inspired me with her PoC, [binctr](#).
- **Eric Biederman** and **Serge Hallyn** — working tirelessly to get **USER** namespaces and countless other unprivileged kernel features working.
- **James Bottomley** — helped me with kernel patches trying to fix the **cgroup** issue and also has done a lot of kernel “container” work.

Questions?

<https://www.cyphar.com/src/talks>