From docker To podman In 5 Minutes

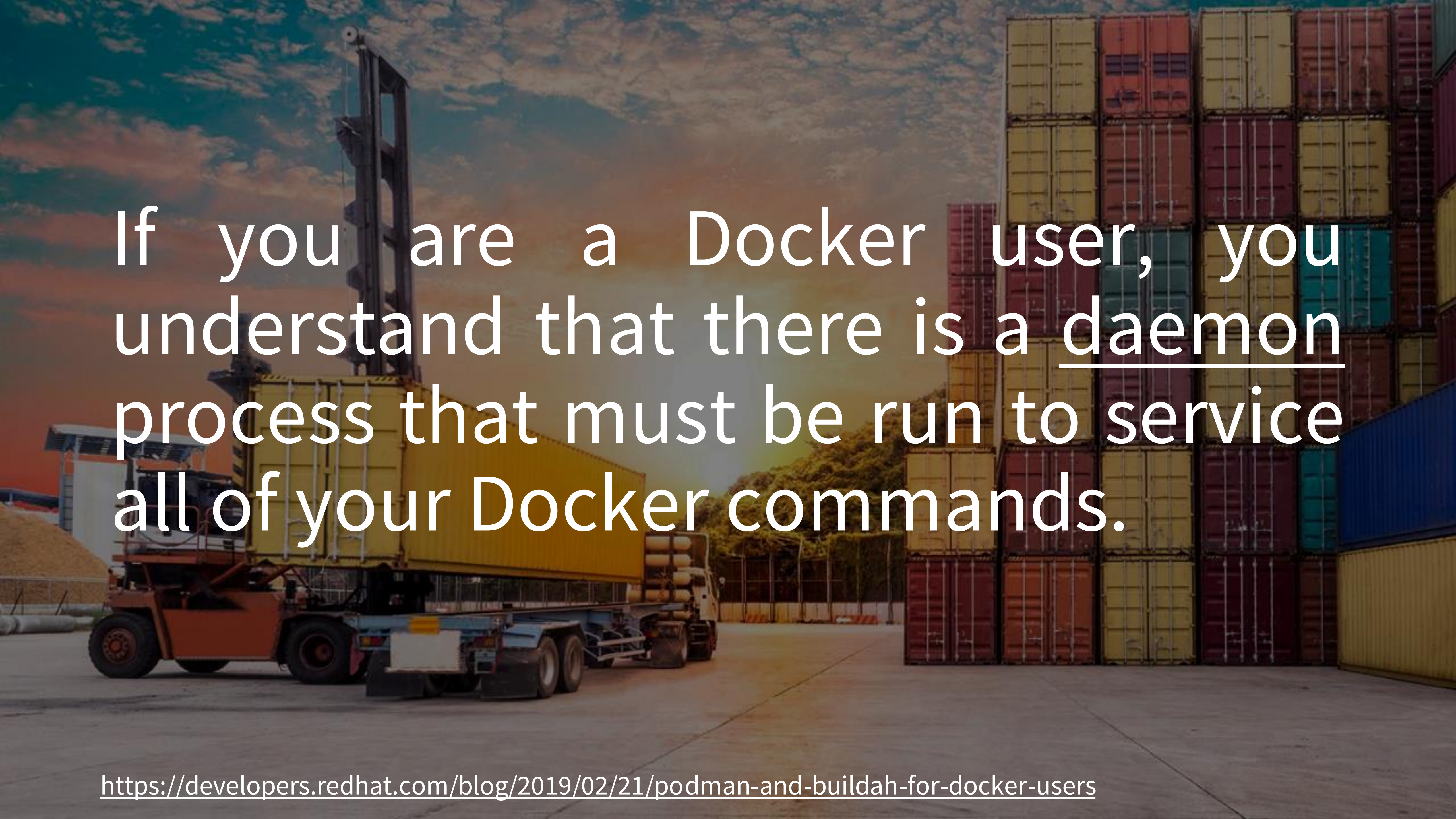
Kukuh Syafaat

kukuh.syafaat@citra.digital

kukuh@syafaat.id

FOSS ASIA 2021

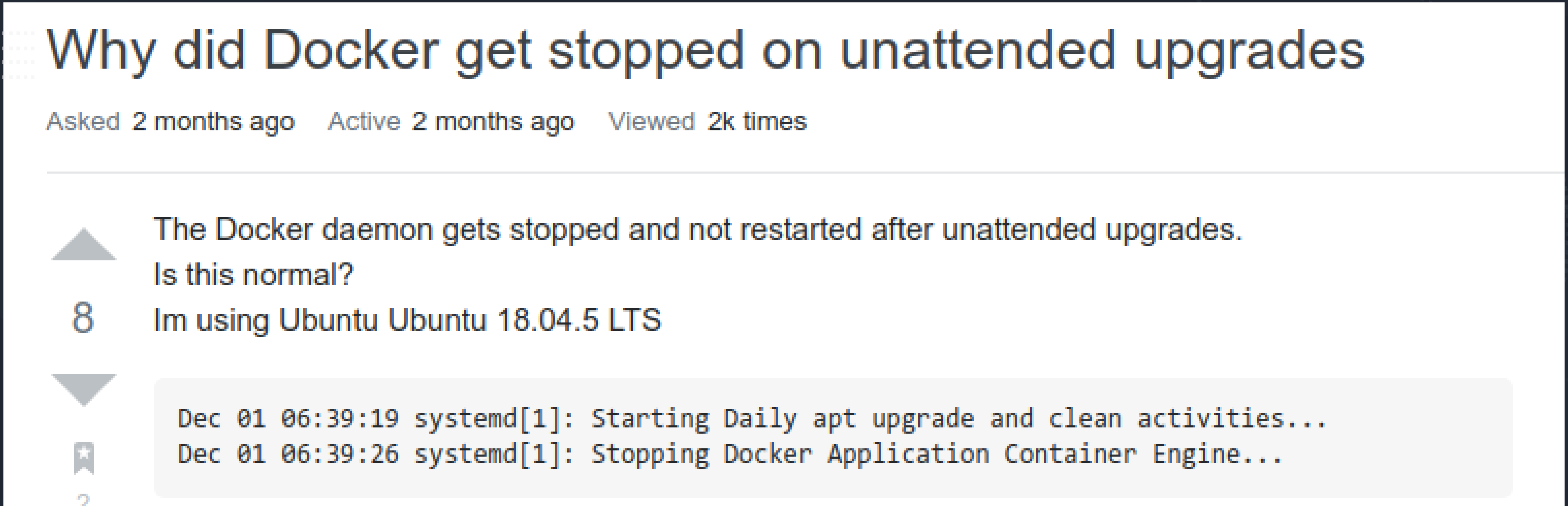
foss asia



If you are a Docker user, you understand that there is a daemon process that must be run to service all of your Docker commands.

https://developers.redhat.com/blog/2019/02/21/podman-and-buildah-for-docker-users

Daemon Problem



Why did Docker get stopped on unattended upgrades

Asked 2 months ago Active 2 months ago Viewed 2k times

The Docker daemon gets stopped and not restarted after unattended upgrades.

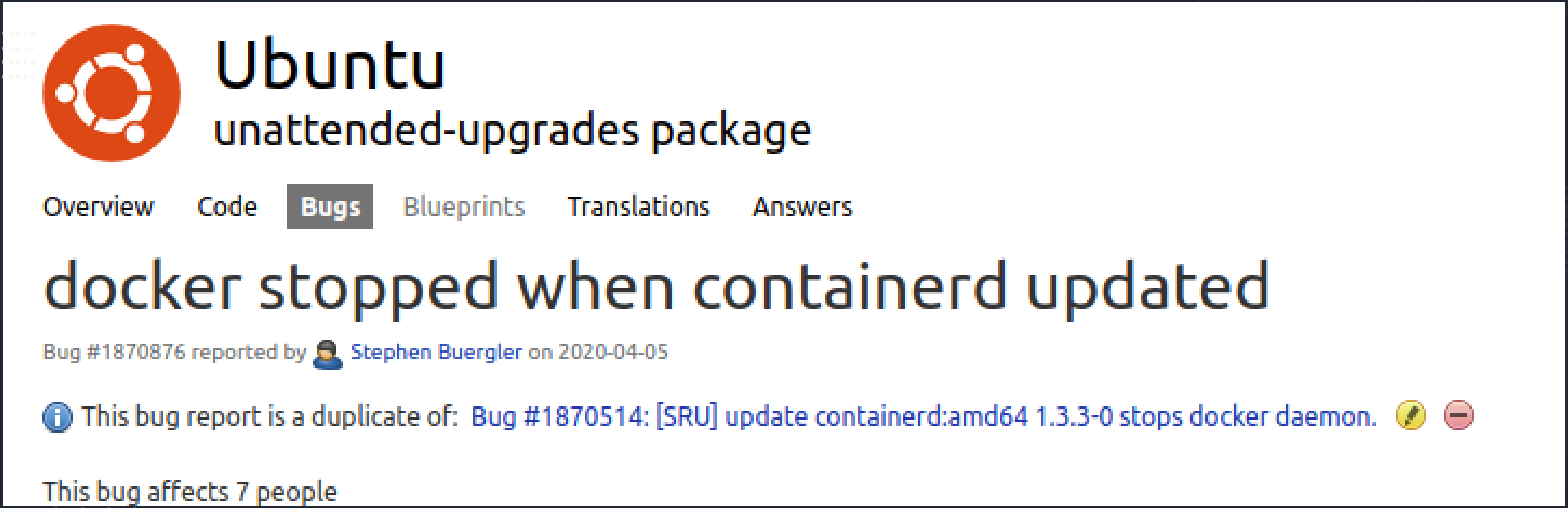
Is this normal?

Im using Ubuntu Ubuntu 18.04.5 LTS

```
Dec 01 06:39:19 systemd[1]: Starting Daily apt upgrade and clean activities...
Dec 01 06:39:26 systemd[1]: Stopping Docker Application Container Engine...
```

https://askubuntu.com/questions/1296583/why-did-docker-get-stopped-on-unattended-upgrades

Daemon Problem



Ubuntu

unattended-upgrades package

docker stopped when containerd updated

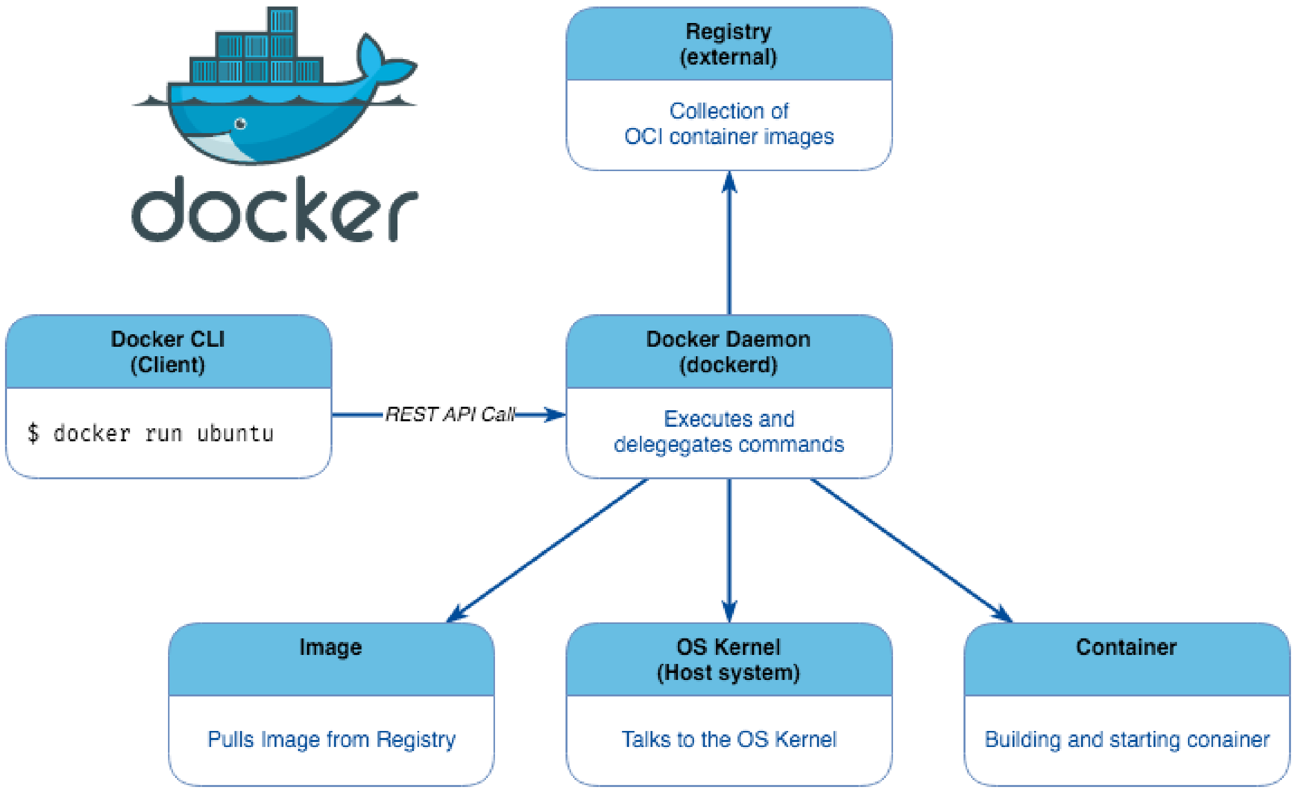
Bug #1870876 reported by Stephen Buergler on 2020-04-05

This bug report is a duplicate of: Bug #1870514: [SRU] update containerd:amd64 1.3.3-0 stops docker daemon.

This bug affects 7 people

https://bugs.launchpad.net/ubuntu/+source/unattended-upgrades/+bug/1870876 docker

podman



Registry
(external)

Collection of
OCI container images

Docker CLI
(Client)

```
$ docker run ubuntu
```

REST API Call

Docker Daemon
(dockerd)

Executes and
delegegates commands

Image

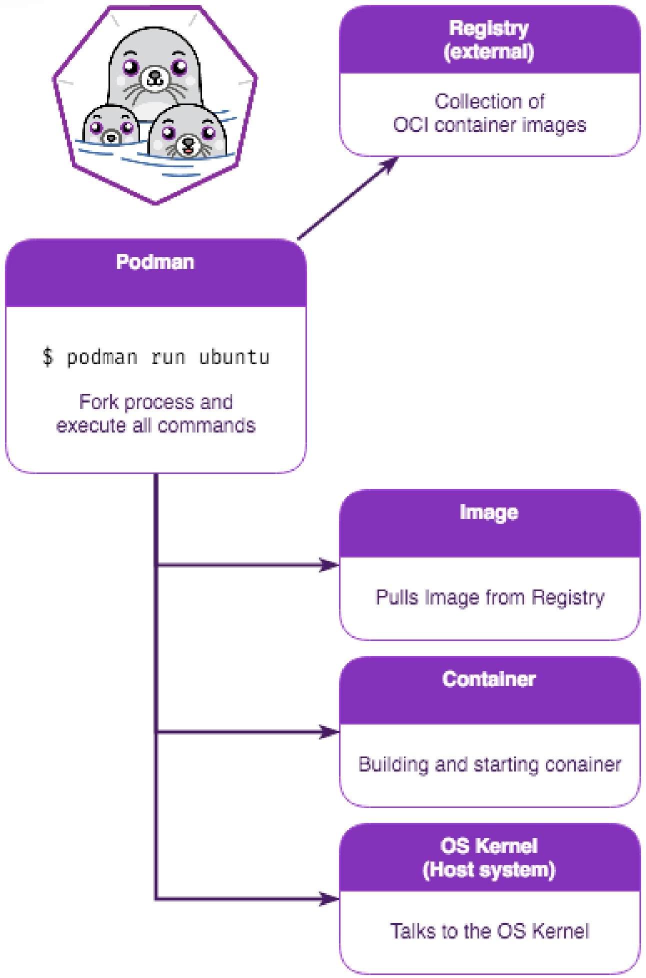
Pulls Image from Registry

OS Kernel
(Host system)

Talks to the OS Kernel

Container

Building and starting conainer



Registry
(external)

Collection of
OCI container images

Podman

```
$ podman run ubuntu
```

Fork process and
execute all commands

Image

Pulls Image from Registry

Container

Building and starting conainer

OS Kernel
(Host system)

Talks to the OS Kernel

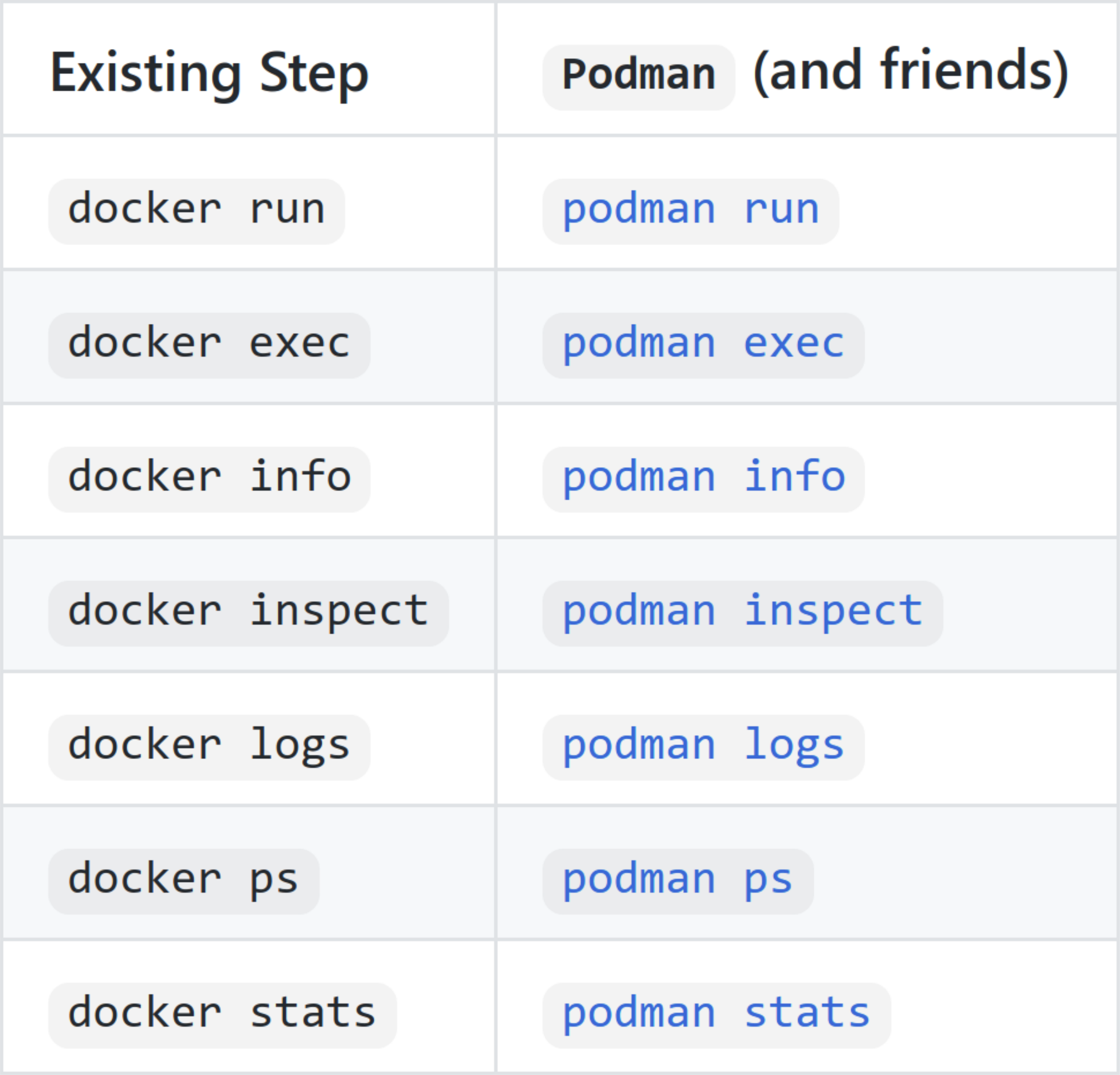
https://www.nine.ch/en/blog/lets-have-some-fun-with-containers-on-nine-managed-servers

DAEMONLESS

Podman is a daemonless container engine for developing, managing, and running OCI Containers on your Linux System. Containers can either be run as root or in rootless mode.

https://podman.io

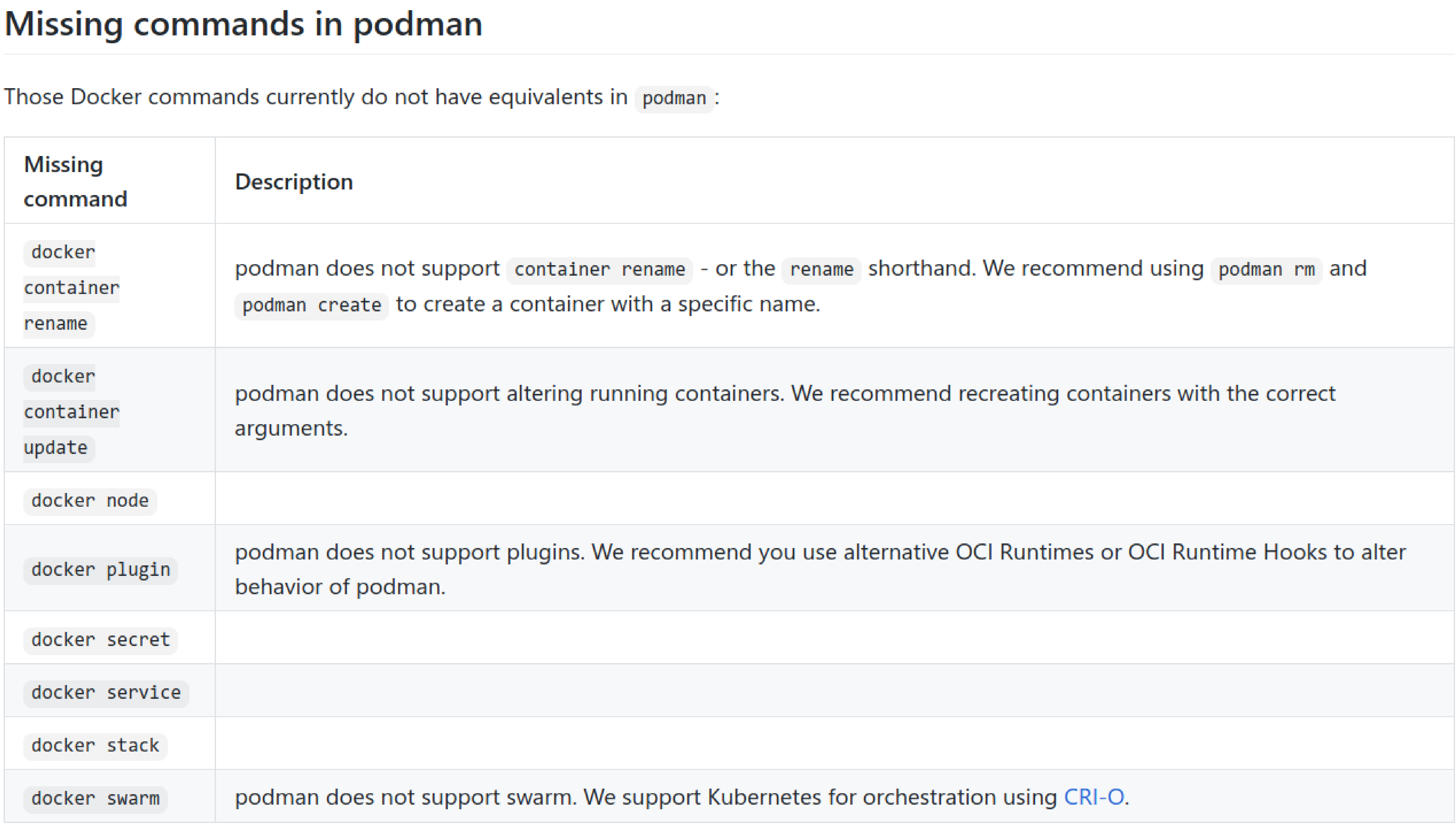
alias docker=podman



Existing Step	`Podman` (and friends)
`docker run`	`podman run`
`docker exec`	`podman exec`
`docker info`	`podman info`
`docker inspect`	`podman inspect`
`docker logs`	`podman logs`
`docker ps`	`podman ps`
`docker stats`	`podman stats`

https://github.com/containers/podman/blob/master/transfer.md

...but...

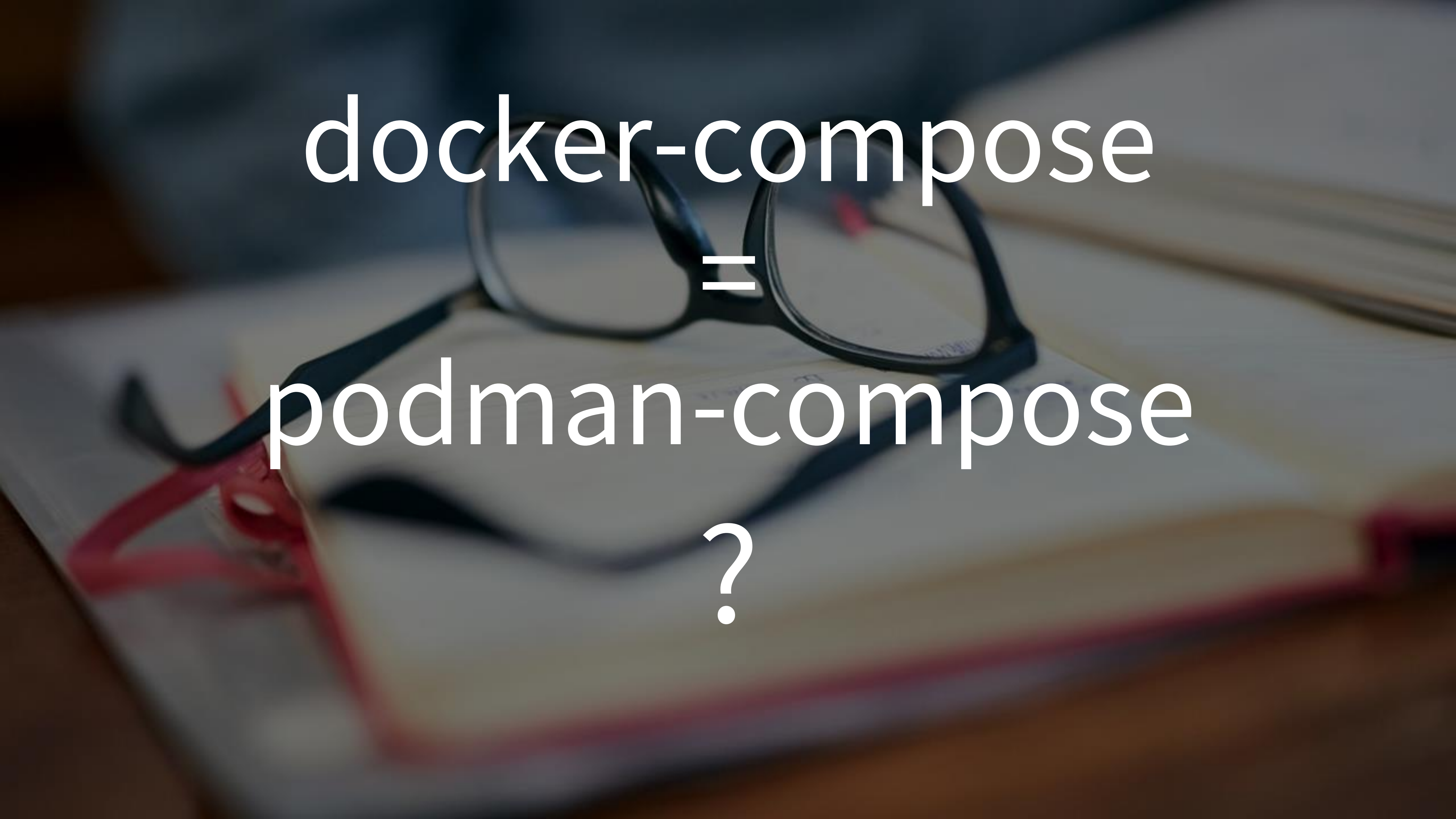


Missing commands in podman

Those Docker commands currently do not have equivalents in `podman`:

Missing command	Description
`docker container rename`	podman does not support `container rename` - or the `rename` shorthand. We recommend using `podman rm` and `podman create` to create a container with a specific name.
`docker container update`	podman does not support altering running containers. We recommend recreating containers with the correct arguments.
`docker node`	
`docker plugin`	podman does not support plugins. We recommend you use alternative OCI Runtimes or OCI Runtime Hooks to alter behavior of podman.
`docker secret`	
`docker service`	
`docker stack`	
`docker swarm`	podman does not support swarm. We support Kubernetes for orchestration using CRI-O.

https://github.com/containers/podman/blob/master/transfer.md

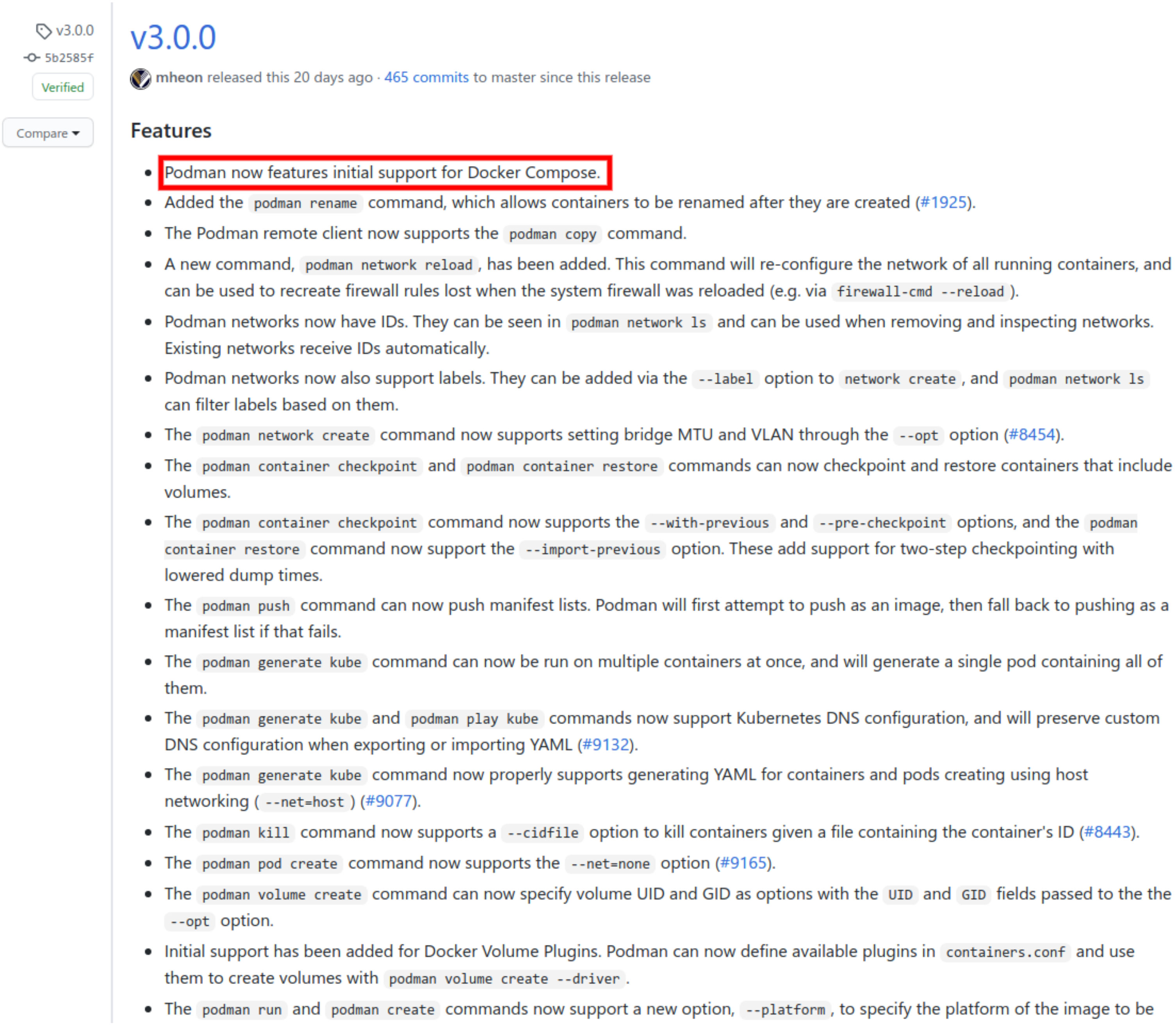


docker-compose

=

podman-compose

?



v3.0.0

mheon released this 20 days ago · 465 commits to master since this release

Features

- Podman now features initial support for Docker Compose.
- Added the `podman rename` command, which allows containers to be renamed after they are created (#1925).
- The Podman remote client now supports the `podman copy` command.
- A new command, `podman network reload`, has been added. This command will re-configure the network of all running containers, and can be used to recreate firewall rules lost when the system firewall was reloaded (e.g. via `firewall-cmd --reload`).
- Podman networks now have IDs. They can be seen in `podman network ls` and can be used when removing and inspecting networks. Existing networks receive IDs automatically.
- Podman networks now also support labels. They can be added via the `--label` option to `network create`, and `podman network ls` can filter labels based on them.
- The `podman network create` command now supports setting bridge MTU and VLAN through the `--opt` option (#8454).
- The `podman container checkpoint` and `podman container restore` commands can now checkpoint and restore containers that include volumes.
- The `podman container checkpoint` command now supports the `--with-previous` and `--pre-checkpoint` options, and the `podman container restore` command now support the `--import-previous` option. These add support for two-step checkpointing with lowered dump times.
- The `podman push` command can now push manifest lists. Podman will first attempt to push as an image, then fall back to pushing as a manifest list if that fails.
- The `podman generate kube` command can now be run on multiple containers at once, and will generate a single pod containing all of them.
- The `podman generate kube` and `podman play kube` commands now support Kubernetes DNS configuration, and will preserve custom DNS configuration when exporting or importing YAML (#9132).
- The `podman generate kube` command now properly supports generating YAML for containers and pods creating using host networking (`--net=host`) (#9077).
- The `podman kill` command now supports a `--cidfile` option to kill containers given a file containing the container's ID (#8443).
- The `podman pod create` command now supports the `--net=none` option (#9165).
- The `podman volume create` command can now specify volume UID and GID as options with the `UID` and `GID` fields passed to the the `--opt` option.
- Initial support has been added for Docker Volume Plugins. Podman can now define available plugins in `containers.conf` and use them to create volumes with `podman volume create --driver`.
- The `podman run` and `podman create` commands now support a new option, `--platform`, to specify the platform of the image to be

https://github.com/containers/podman/releases/tag/v3.0.0

Podman
and
Docker Compose

Start the Podman System Service

- Be sure the `podman-docker` and `docker-compose` packages are installed
- Enable and start the Podman `systemd` socket-activated service

```
$ sudo systemctl enable --now start podman.socket
```

- Verify the service

```
$ sudo curl -H "Content-Type: application/json" --unix-socket
/var/run/docker.sock http://localhost/_ping
OK
```

https://www.redhat.com/sysadmin/podman-docker-compose
https://www.redhat.com/sysadmin/compose-kubernetes-podman

Run Compose

- For example, using compose from https://github.com/docker/awesome-compose/tree/master/wordpress-mysql
- Bring up the containers

```
$ sudo docker-compose up -d
```

- Verify the containers running

```
$ sudo podman ps
```

- Verify WordPress is running using the localhost and port 80

https://www.redhat.com/sysadmin/podman-docker-compose
https://www.redhat.com/sysadmin/compose-kubernetes-podman

Acknowledgement

- Rifki Affandi (Awan Sakti, openSUSE-ID) "*Daemonless with Podman*" on Btech Bicara #20 https://www.youtube.com/watch?v=TXFLDQ3SS0I

Thank You

We are hiring

joinus@citra.digital