how Docker works

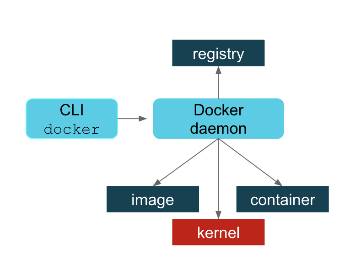
docker daemon

a daemon process that must be run to service all of your Docker commands providing functionalities: Docker daemon does all the work with registries, images, containers, and the kernel.

* Pull and push images from an image registry
* Make copies of images in a local container storage and to add layers to those containers
* Commit containers and remove local container images from the host repository
* Ask the kernel to run a container with the right namespace and cgroup, etc.

Docker client

The Docker command-line interface (CLI) asks the daemon to do this on your behalf.



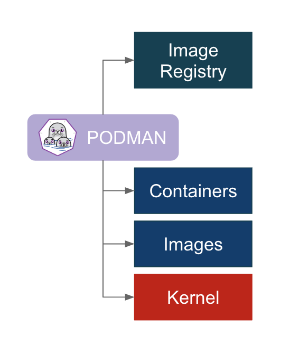
Concerns

* A single process could be a single point of failure.
* This process owned all the child processes (the running containers).
* If a failure occurred, then there were orphaned processes.
* Building containers led to security vulnerabilities.
* All Docker operations had to be conducted by a user (or users) with the same full root authority.

how Podman works Podman.io

No daemon

directly interact with the image registry, with the container and image storage, and with the Linux kernel through the **runC container runtime process** (not a daemon).



Improvements

(1)You install Podman instead of Docker. You do not need to start or manage a daemon process like the Docker daemon.

(2)The commands you are familiar with in Docker work the same for Podman.

(3)Podman stores its containers and images in a different place than Docker.

(4)Podman and Docker images are compatible.

(5)Podman does more than Docker for [Kubernetes](https://developers.redhat.com/topics/kubernetes/) environments.

Consuming Podman

(1)**Installing Podman**You install Podman instead of Docker. You do not need to start or manage a daemon process like the Docker daemon.

1) keep Docker around while you try out Podman. There are some useful [tutorials](https://github.com/containers/libpod/blob/master/docs/tutorials/podman_tutorial.md) and an awesome [demonstration](https://github.com/containers/Demos/tree/master/building/buildah_intro)

2) on [Red Hat Enterprise Linux](https://developers.redhat.com/products/rhel/overview/) 7.6 or later

**# yum -y install podman**

(2)The commands you are familiar with in Docker work the same for Podman.

Same

alias docker=podman/ install podman-docker package that does this for conversion(drops a script at /usr/bin/docker that executes Podman with the same arguments.)

different

Podman has added some convenience flags to some commands. For example, Podman has added --all (-a) flags for podman rm and podman rmi.

(3)Podman stores its containers and images in a different place than Docker.

places

Podman’s local repository is in /var/lib/containers instead of /var/lib/docker.  This isn’t an arbitrary change; this new storage structure is based on the Open Containers Initiative (OCI) standards.

OCI

In 2015, Docker, Red Hat, CoreOS, SUSE, Google, and other leaders in the Linux containers industry created the Open Container Initiative in order to provide an independent body to manage the standard specifications for defining container images and the runtime. In order to maintain that independence, the [containers/image](https://github.com/containers/image) and [containers/storage](https://github.com/containers/storage) projects were created on [GitHub](https://github.com/containers).

user’s home directory

run Podman from your normal non-root user in Podman 1.0 on Fedora. RHEL support is aimed for version 7.7 and 8.1 onwards.

store images and containers in the user’s home directory.  ~/.local/share/containers. This avoids making /var/lib/containers world-writeable or other practices that might lead to potential security problems

how

[How does rootless Podman work?](https://opensource.com/article/19/2/how-does-rootless-podman-work)

(4)Podman and Docker images are compatible.

Despite the new locations for the local repositories, the images created by Docker or Podman are compatible with the OCI standard.

Registries

Docker hub

**$ podman pull fedora:latest**

**$ podman run -it fedora bash**

[Quay.io](https://quay.io/)

**$ podman pull quay.io/ipbabble/myfedora:latest**

**$ podman push myfedora quay.io/myquayid/myfedora:latest**

move images from /var/lib/docker to /var/lib/containers

**# podman pull docker-daemon:fedora:latest**

**# podman push myfedora docker-daemon:myfedora:latest**

Default

default to searching through registries listed in the registries.conf file

(5)Podman does more than Docker for [Kubernetes](https://developers.redhat.com/topics/kubernetes/) environments.

Commands

podman generate kube. Podman can generate a Kubernetes YAML file based on a running container

podman pod can be used to help debug running Kubernetes pods along with the standard container commands.

How

[Podman can now ease the transition to Kubernetes and CRI-O](https://developers.redhat.com/blog/2019/01/29/podman-kubernetes-yaml/).

Terminologies

Open Containers Initiative (OCI) standards

[containers/image](https://github.com/containers/image) and [containers/storage](https://github.com/containers/storage)

runC container runtime process

CRI-O Open Container Initiative-based implementation of Kubernetes Container Runtime Interface

Buildah [Buildah.io](https://buildah.io/)

Why we have podman

Podman does do builds and for those familiar with Docker, the build process is the same. You can either build using a Dockerfile using podman build or you can run a container and make lots of changes and then commit those changes to a new image tag.

Buildah can be described as a superset of commands related to creating and managing container images and, therefore, it has much finer-grained control over images.

Best Practice

write Bash scripts for creating your images—in a similar way that you would write a Dockerfile.

Run K8S not on docker but CRI-O

When Kubernetes moved to [CRI-O](https://developers.redhat.com/blog/2019/01/29/podman-kubernetes-yaml/) based on the OCI runtime specification, there was no need to run a Docker daemon and, therefore, no need to install Docker on any host in the Kubernetes cluster for running pods and containers. Kubernetes could call CRI-O and it could call runC directly.

Points

1. It allows for finer control of creating image layers. Committing many changes to a single layer is desirable.
2. Buildah’s run command is not the same as Podman’s run command.  Because Buildah is for building images, the run command is essentially the same as the Dockerfile RUN command.

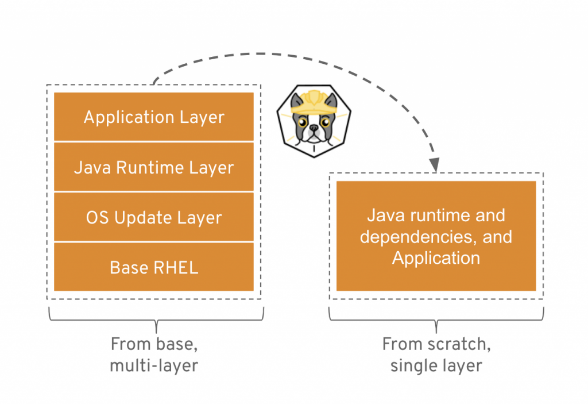
No port mapping. No volume mounting. Those flags were removed.  Instead buildah run is for running specific commands in order to help build a container image, for example, buildah run dnf -y install nginx.

1. Buildah can build images from scratch, that is, images with nothing in them at all. Nothing. In fact, looking at the container storage created as a result of a buildah from scratch command yields an **empty directory**. This is useful for creating very lightweight images that contain only the packages needed in order to run your application

use case for a scratch build

During development, a Java application container image may require the Java compiler and Maven and other tools.

But in production, you may only require the Java runtime and your packages. And, by the way, you also do not require a package manager such as DNF/YUM or even Bash.



* [github.com/containers](https://github.com/containers)projects (get involved, get the source, see what’s being developed):
  + [libpod](https://github.com/containers/libpod) (Podman)
  + [buildah](https://github.com/containers/buildah)
  + [image](https://github.com/containers/image) (code for working with OCI container images)
  + [storage](https://github.com/containers/storage) (code for local image and container storage)