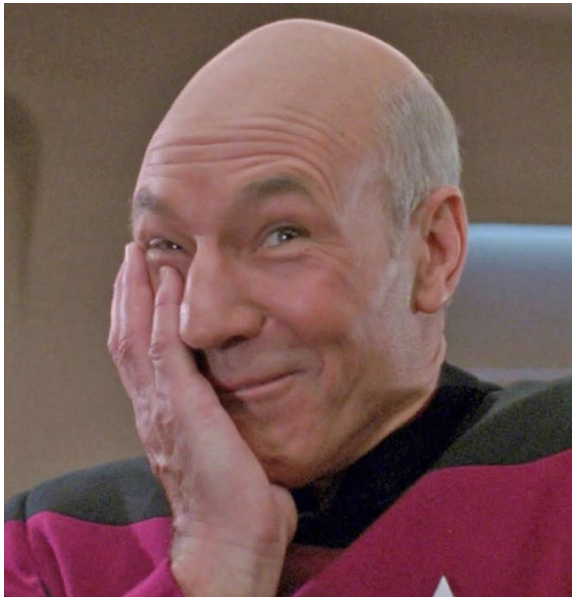
Container Tools

The Next Generation

Presenters

Laine Minor



Joshua Smith

Agenda

1. What are containers?
2. Where do containers run?
3. The Container Lifecycle and Related Tools
4. Running Securely
5. Demo!
6. Your Favorites

What are containers?

Thanks, Google...

Refine by material

Plastic

Glass

Stainless Steel

Ceramic

A brief history...

A “container” is actually made up of several of the features that are part of the Linux kernel, such as:

- `chroot`
- `cgroups`
- `namespaces`
- `SELinux profiles`

A brief history...

They're also made up of lots of open source projects like:

- `Docker` (the packaging format)
- `Docker` (the container management runtime)

The list of open source projects included *in* a "container," or used to *manage* containers, changes all the time as our collective maturity with them increases.

...that's actually how this talk came to be!

A brief history to 2015... (not to scale)

1979

chroot added to Unix - the ability to change the root of the filesystem for a process

2000

jail introduced to BSD, to enforce the concept of chroot for files and resources

Also, **SELinux** created to restrict how groups of processes can interact with other processes, files and devices

2001

Linux-VServer project introduced, a similar jail concept to Linux

2005

Seccomp - limit system calls a task can use (useful later)

2006

cpusets adapted to create “generic process containers” which were renamed control groups, or **cgroups**

2008

user namespaces - processes can have their own users, privileged users in a container

Also, **LXC from IBM** - tooling for containers, but relies on security incomplete Discretionary Access Control

2013

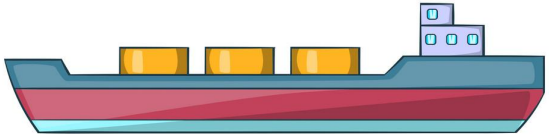
Docker is open sourced

2014

First release of **Kubernetes**

Okay but *why* containers?

"The modern shipping industry only works as well as it does because we have **standardized on a small set of shipping container sizes**.



...Instead of ships that specialize in bringing smartphones from Asia, we can just put them all into containers and know that those will fit on *every container ship*."

- WTF is a container?

An application, decomposed

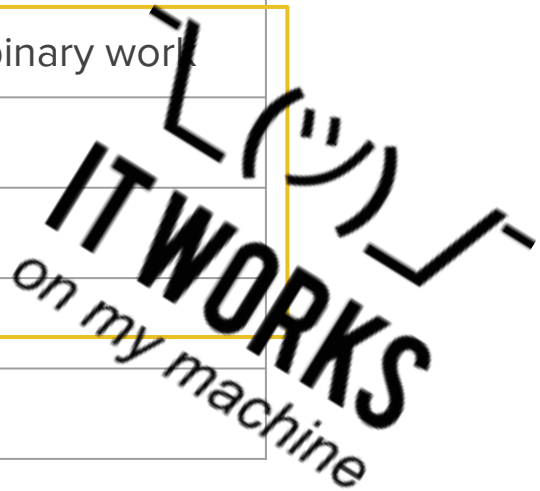
For an application to run, it needs…

Compiled and built application code binary
Dependencies to make the application code binary work
Dependencies to make the middleware work
Middleware
Operating System
Hardware

The Problem...

On a traditional VM or mainframe, this middle area causes...*uhh*, complication.

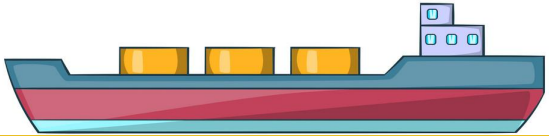
Compiled and built application code binary
Dependencies to make the application code binary work
Dependencies to make the middleware work
Middleware
Operating System
Hardware



The Solution! (or part of it, anyway)

Containers are amazing because they bundle all of *this* into **one standardized deliverable**:

Compiled and built application code binary
Dependencies to make the application code binary work
Dependencies to make the middleware work
Middleware
Operating System
Hardware

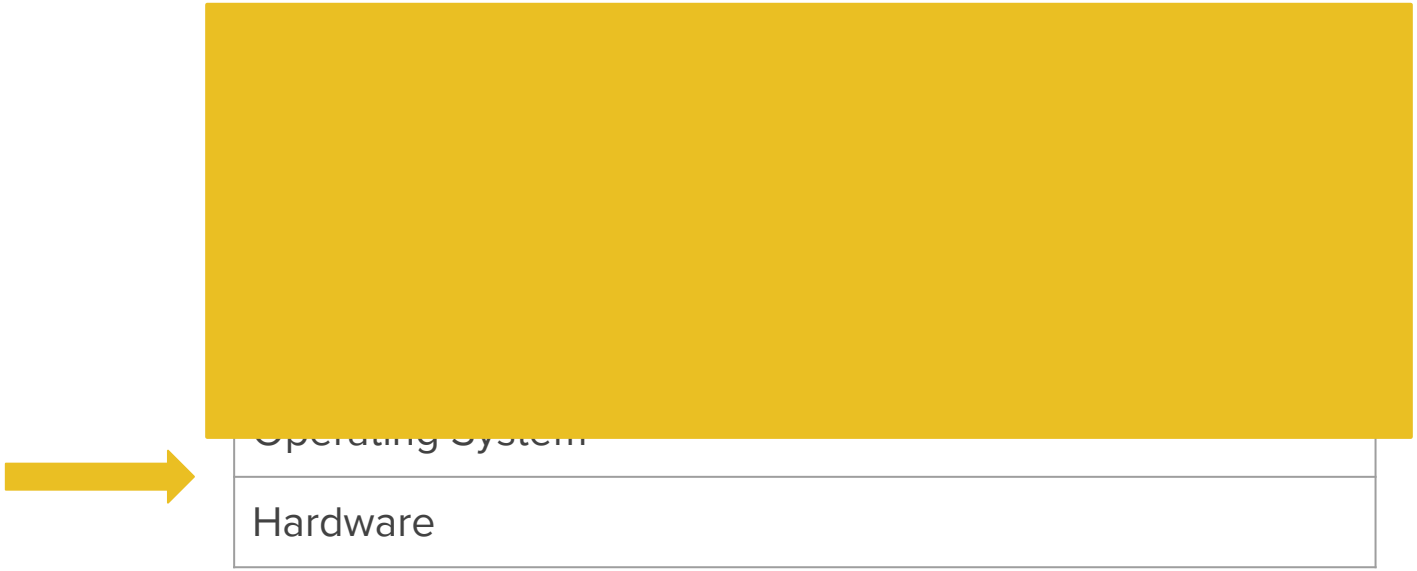


Where do containers run?

Where do containers run?

Compiled and built application code binary
Dependencies to make the application code binary work
Dependencies to make the middleware work
Middleware
Operating System
Hardware

Where do containers run?



Where do containers run?

There are two common options:

1. Linux
    a. *various…*
    b. *…flavors…*
    c. *…of…*
    d. *…Linux*
2. Windows
    a. *except this is super complicated.*

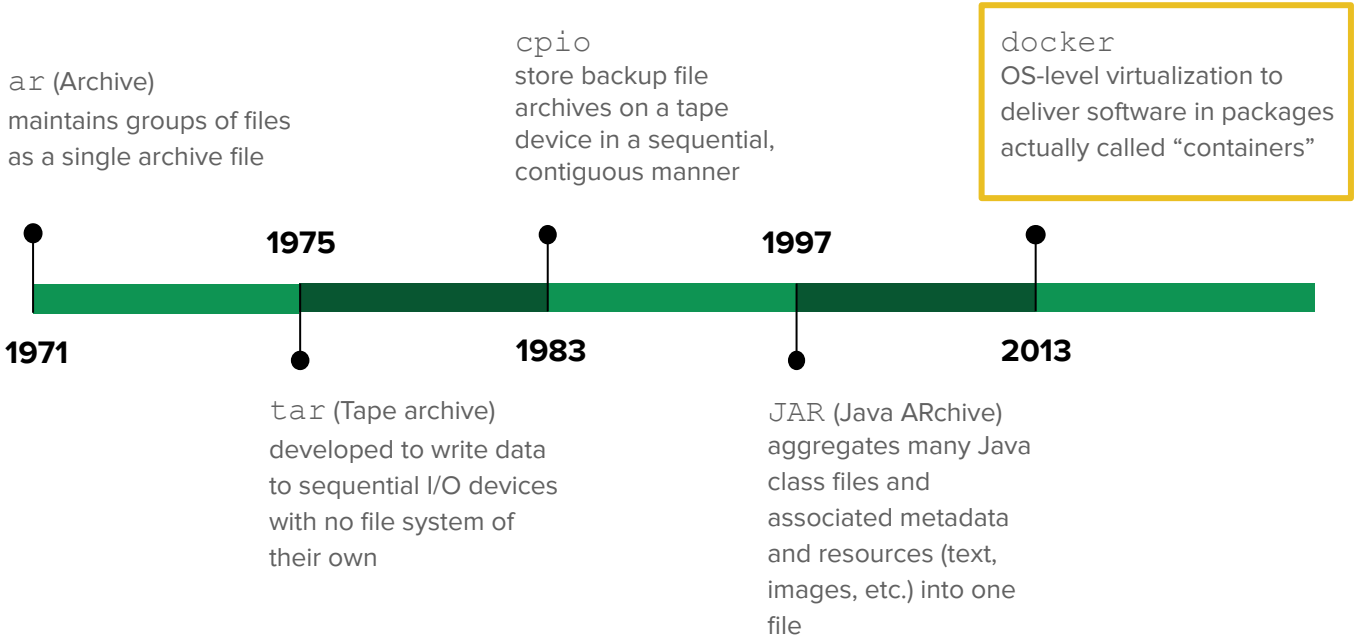
The Container Lifecycle and Related Tools

Another brief history...

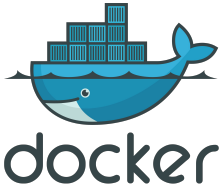
Using containers (or things like containers) to deliver business priorities in the form of running code isn't a new plan.

Linux archives/containers/etc have been doing this in a number of ways over the years.

Another brief history... (still not to scale)



Why Docker?



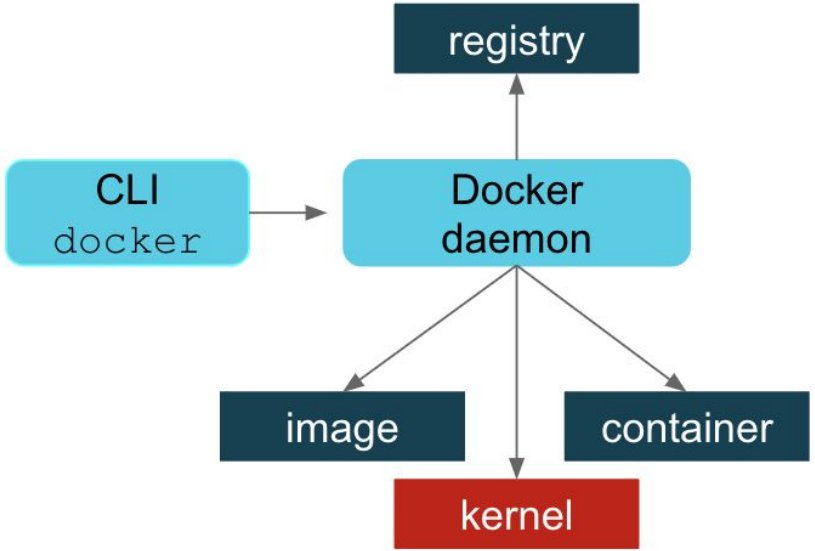
Standard, consistent format, consistent runtime, open source.

Brought together several pieces needed to create and manipulate containers in one complete package:

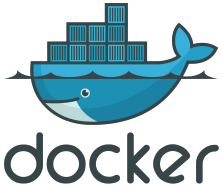
- Pull and push to/from an image registry
- Make local copies of images and add layers
- Commit image changes
- Image management/clean up
- Ask the kernel to run a container with the right namespace and cgroup
- Manage resources

Article: Podman and Buildah for Docker users

How Docker Works

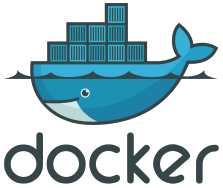


Why *Not* Docker (Technical Reasons)



1. Complicated multi-level builds
   - required root access for all stages
   - build had to be done in sequence, on only one host
     - parent process owned all child processes - if a failure occurred, there were orphaned processes
   - required a Daemon
2. Security concerns
   - root access -> increased security risk
   - insecure registry - provenance and security cycles

Why *Not* Docker (Business Reasons)



Corporately - struggled to make money

This led to some wavering regarding their stance on keeping their tools open source.

(Anyone use Docker Desktop? Because... that costs money now.)

https://www.zdnet.com/article/docker-is-in-deep-trouble/
https://opensource.stackexchange.com/questions/5436/is-docker-still-free-and-open-source

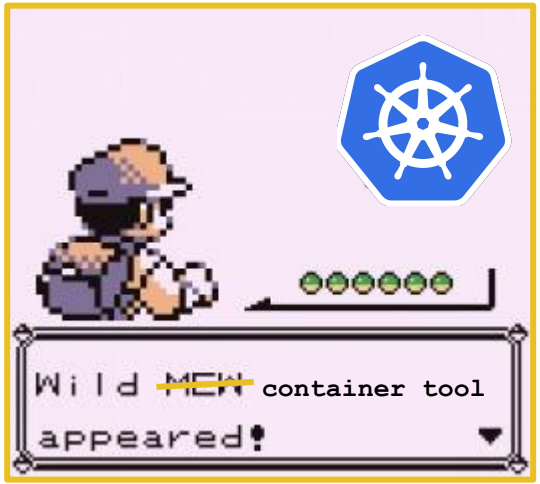
So… New Tools Needed!

Ideally, new container tools would be:

- Easier to use
- Reflective of how enterprises use containers at scale:
  - Building
  - Deploying and distributing
  - Running

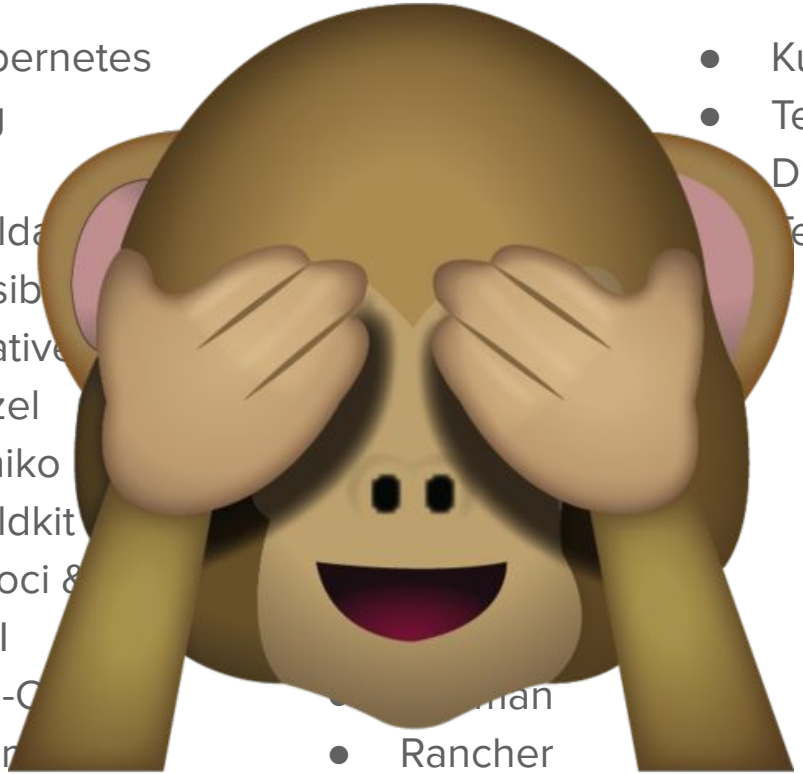
So... New Tools Needed!

- Kubernetes
- img
- S2I
- Buildah
- Ansible Container
- Knative
- Bazel
- kaniko
- Buildkit
- umoci & orca
- OCI
- CRI-O
- Helm
- Open Policy Agent
- ArgoCD
- AgnosticD
- Moby
- k3s
- containerd
- Istio
- Mesos
- Twistlock
- RancherOS
- CoreOS
- rkt
- Podman
- Rancher
- KubeLinter
- KubeBench
- Tekton
- Distroless Containers
- TerraScan

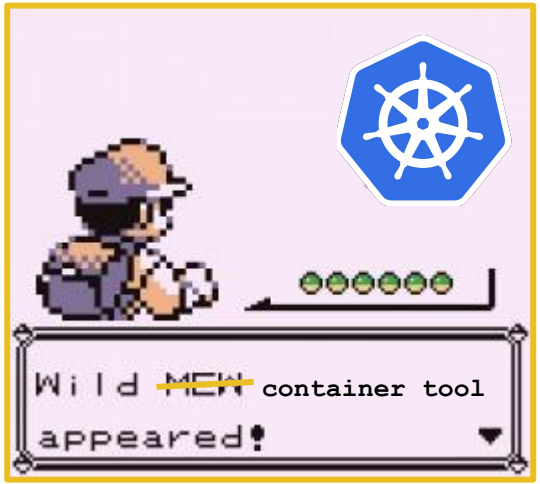


So... New Tools Needed!

- Kubernetes
- img
- S2I
- Builda
- Ansib
- Knative
- Bazel
- kaniko
- Buildkit
- umoci &
- OCI
- CRI-O
- Helm
- Open Policy Agent



- Rancher
- KubeLinter
- KubeBench
- Tekton
- Distroless Containers
- TerraScan



So… New Tools Needed!

Some of these tools have emerged as the leaders in their respective categories, but the landscape continues to evolve.

Let’s take a step back for a minute…

What do we actually want to **do** with containers?

What do we want to **do** with containers?

1. Develop
2. Build (Create *or* Update)
3. Store/Archive/Version
4. Deploy
5. Run (Securely!)
6. Move

The Software Delivery Lifecycle 🎉

1. Develop
2. Build (Create *or* Update)
3. Store/Archive/Version
4. Deploy
5. Run (Securely!)
6. Move

The *Container* Lifecycle

1. Develop
2. Build (Create *or* Update)
3. Store/Archive/Version
4. Deploy
5. Run (Securely!)
6. Move

Develop

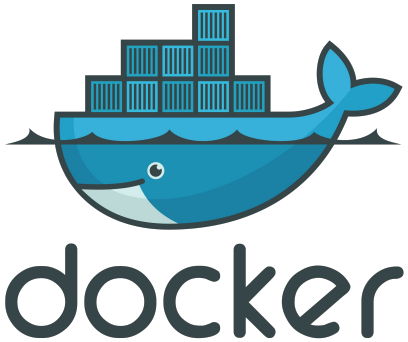
- Eclipse Che: Kubernetes-native IDE
- VSCode + Marketplace
- Skaffold

What happened with Docker?

Docker, in a nutshell, provided the ability to build, run, and move images/containers.

- Buildah: “efficiently and quickly **building** OCI-compliant images and containers”
- Podman: “daemonless container engine for developing, managing, and **running** OCI containers”
- Skopeo: **moves**, inspects, signs images

What happened with Docker?

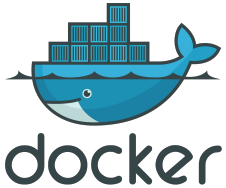


buildah

podman

skopeo

Build



S2I - self-assembling builder images

Knative - serverless workloads

CI/CD Tools

Store/Archive/Version

Skopeo - move between registries

Quay - registry

Docker Hub - registry

Deploy

- Docker
- Podman
- Buildah
- Kubernetes Platforms
  - OpenShift, GKE, AKS, EKS
- Knative

CI/CD Tools

Run

- Kubernetes Platforms
- Knative
- CRI-O - lightweight container runtime
- Podman (Edge)
- Docker (Edge)
- Cloud Container Services

Move


Skopeo

Docker


Podman


Buildah

Running *Securely*

Security Comes in Layers

Compiled and built application code binary
Dependencies to make the application code binary work
Dependencies to make the middleware work
Middleware
Operating System
Hardware

Inspection and Signing

Skopeo - inspection, signing

Dive - inspection, also some build, also some optimization

Security Tools

- SonarQube - code scanning
- Clair - container dependency scanning
- KubeLinter - config checking
- Open Policy Agent - security policy, see also Falco
- Kube-Hunter - vulnerability detection/exploit

Demo!

Demo!

https://github.com/lainie-ftw/demos/blob/master/container-tools-tng/container-tools-demo.sh (if you want to try it yourself!)

Your Favorites

Some of Your Favorites

Snyk

Trivy

Kind

VSCode DevContainers - https://github.com/Microsoft/vscode-dev-containers

More questions?

Thank you!

Josh:
josh@soul-repairs.com
@architect_josh

Laine:
laine@soul-repairs.com
@lainieftw

soul-repairs.com