Establishing Image Provenance and Security in Kubernetes

Adrian Mouat

Container Solutions

info@container-solutions.com
www.container-solutions.com

Photo by Eddie Howell


ASDA
GROWER'S
SELECTION
RIPE & READY
AVOCADOS
WASH BEFORE USE • CONTAINS STONES
PERU
HASS
SITE: 253443
BEST BEFORE:
10 OCT
X2
175-2176
1 of 5 a day per 1/2 avocado
Packed for Asda Stores Ltd, Leeds LS11 5AD. Phone us on: 0800 952 0101

KWTFIGOIYC

@adrianmouat
Container Solutions

KWTFIGOIYC

Know What The F*** Is Going On In Your Cluster

@adrianmouat
Container Solutions

For every image in our cluster, we should be able to answer:

- What is it?
- Where did it come from?
- How can I rebuild it?
- Does it have any known vulnerabilities?
- Is it up-to-date?

Can we ***prove*** the answers?

- **What is it?**
  - Where did it come from?
  - How can I rebuild it?
  - Does it have any known vulnerabilities?
  - Is it up-to-date?

“**Docker** will do to **apt** what **apt** did to **tar**”

Bryan Cantrill

Joyent

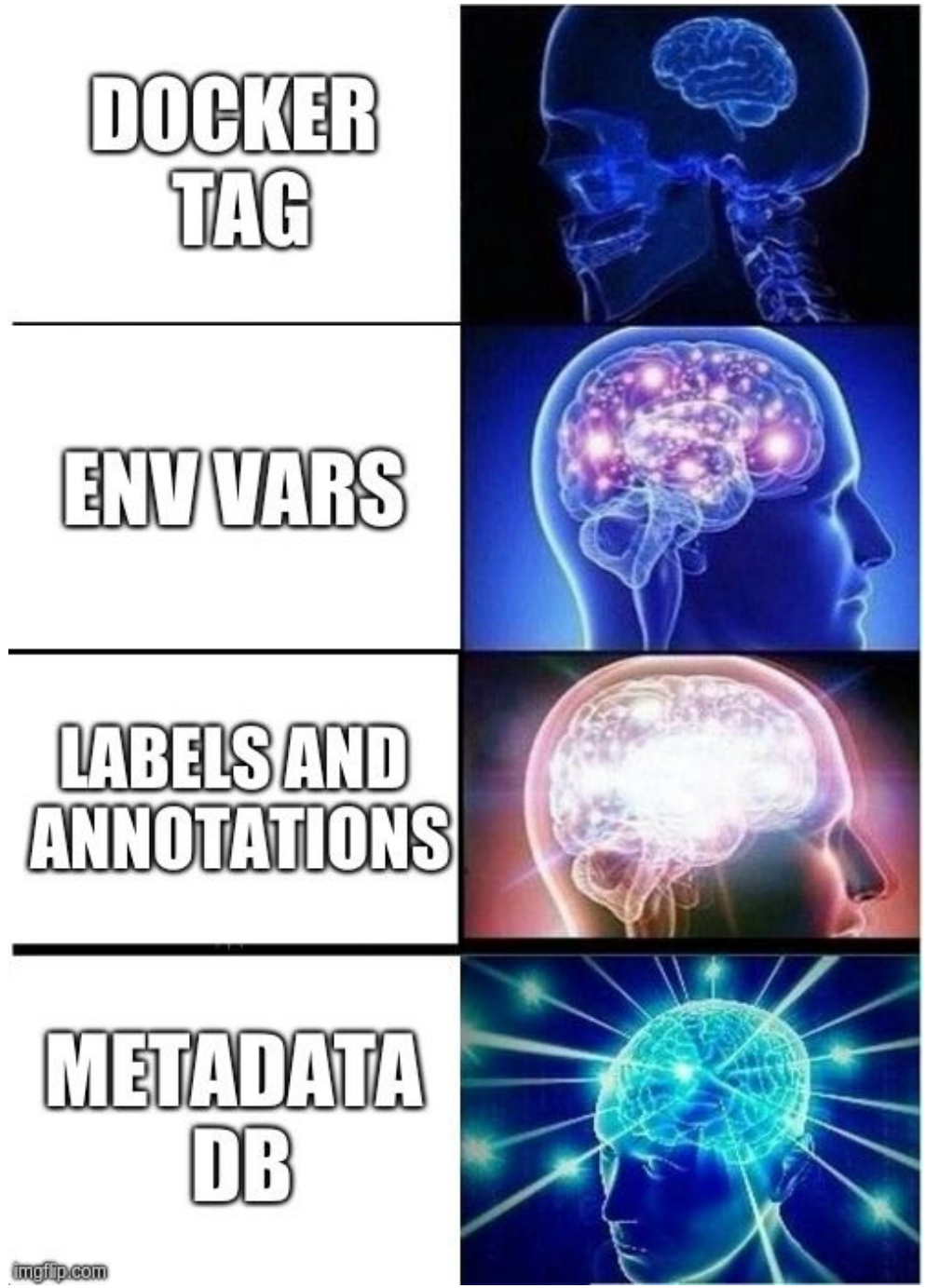
@bcantrill

Kubectl Output

```
$ kubectl get pods --all-namespaces
NAMESPACE     NAME                                                ...
default       blog-7886fbf79b-mvndx
default       db-75d77f7c88-tpkwr
default       proxy-c65d78cbc-b5lq2
kube-system   event-exporter-v0.2.1-5f5b89fcc8-65dxs
kube-system   fluentd-gcp-scaler-7c5db745fc-rjfwf
...
```

```
$ kubectl describe pod blog-7886fbf79b-mvndx
...
Containers:
  blog:
    Container ID:   docker://9e9b48b11fb0e53a8dcec5989d942...
    Image:          wordpress:4.9-php7.0-apache
    Image ID:       docker-pullable://wordpress@sha256:3d7b4...
...
```


MOAR!
MOAR!
MOAR!


DOCKER TAG
ENV VARS
LABELS AND ANNOTATIONS
METADATA DB
imgflip.com

Kubernetes views tags as **immutable**

Docker views tags as **mutable**

Both are useful.

@adrianmouat

Tagging Images

- Treat production images as immutable
  - Git Hash
  - Full version number
  - Digest

Environment Variables

```
$ kubectl exec proxy-c65d78cbc-b5lq2 env
...
NGINX_VERSION=1.15.5-1~stretch
NJS_VERSION=1.15.5.0.2.4-1~stretch
...
```

Environment Variables

- Limited
- Not structured/standardised
- Mixes config and metadata
- Labels were meant to fix this!
  - (And annotations)

Labels

```
$ cat Dockerfile
...
ARG VCS_REF
LABEL org.opencontainers.image.revision=$VCS_REF \
      org.opencontainers.image.source= \
         "https://github.com/ContainerSolutions/trow"
...

$ docker build -t amouat/trow \
     --build-arg VCS_REF=$(git rev-parse --short HEAD) .
```

Labels

```
$ docker inspect -f "{{json .ContainerConfig.Labels}}" \
    amouat/trow | jq .
{
  "org.opencontainers.image.revision": "fef36bd",
  "org.opencontainers.image.source":
"https://github.com/ContainerSolutions/trow"
}
```

Annotations

- Defined in OCI Image Spec
- Technically different to Labels
- “Pre-Defined Annotation Keys”

Pre-Defined Annotation Keys

This specification defines the following annotation keys, intended for but not limited to image index and image manifest authors:

- **org.opencontainers.image.created** date and time on which the image was built (string, date-time as defined by RFC 3339).
- **org.opencontainers.image.authors** contact details of the people or organization responsible for the image (freeform string)
- **org.opencontainers.image.url** URL to find more information on the image (string)
- **org.opencontainers.image.documentation** URL to get documentation on the image (string)
- **org.opencontainers.image.source** URL to get source code for building the image (string)
- **org.opencontainers.image.version** version of the packaged software
  - The version MAY match a label or tag in the source code repository
  - version MAY be Semantic versioning-compatible
- **org.opencontainers.image.revision** Source control revision identifier for the packaged software.
- **org.opencontainers.image.vendor** Name of the distributing entity, organization or individual.
- **org.opencontainers.image.licenses** License(s) under which contained software is distributed as an SPDX License Expression.
- **org.opencontainers.image.ref.name** Name of the reference for a target (string).
  - SHOULD only be considered valid when on descriptors on `index.json` within image layout.
  - Character set of the value SHOULD conform to alphanum of `A-Za-z0-9` and separator set of `-._:@/+`
  - The reference must match the following grammar:

```
ref       ::= component ("/" component)*
component ::= alphanum (separator alphanum)*
alphanum  ::= [A-Za-z0-9]+
separator ::= [-._:@+] | "--"
```

- **org.opencontainers.image.title** Human-readable title of the image (string)
- **org.opencontainers.image.description** Human-readable description of the software packaged in the image (string)

Annotations

- Currently unsupported by build tools
- Just use Labels
  - And predefined keys

Annotations

Hopes for the future:

- Better support in Kubernetes
- Better support in build tooling
- Greater awareness and use

Metadata DB

- Store information on images
- Keyed by digest
- Can be updated with events
- Build data, contents and versions, known vulns

Grafeas

What about the Registry?

- Would like to:
  - Search for all tags for digest
  - Have audit information
  - Plus other metadata

- What is it?
- Where did it come from?
- **How can I rebuild it?**
- Does it have any known vulnerabilities?
- Is it up-to-date?

*“**Reproducibility** is a **virtue**”*

Dinah McNutt
Google Release Engineer
@dinahSBR

Reproducible Docker Builds

- Use tagged base images
  - or digests
- Version package installed software
  - run a mirror for total control

Downloading Software

- Be careful when using curl/wget
  - Use GPG to verify signatures
  - Checksums

Binary Reproducibility

- File timestamps
- Other metadata
  - Build container IDs
  - Created timestamp

Bazel

@adrianmouat
Container Solutions

DEMO TIME!

Distroless

- Base Images from Google
- Only contain runtime dependencies
- No package manager or shell
- Great for vulnerability scans
- And reducing image size

So we should all use Bazel?

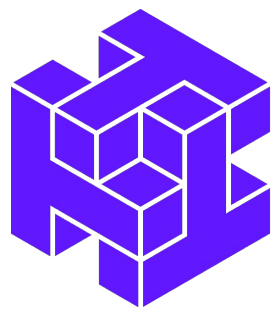
- Err, probably not:
  - It’s big and complicated
  - Wants to build all your stuff
  - Large learning curve
  - Docs need work

- What is it?
- Where did it come from?
- How can I rebuild it?
- **Does it have any known vulnerabilities?**
- **Is it up-to-date?**

clair

aqua

NeuVector



Twistlock

Photo by Aqua Mechanical

Up-to-date vs Stable

- Tension
  - Don’t want breaking changes
  - Do want bug-fixes!
- Good test suite
- Semantic versioning
  - Pin to minor version (4.1.x)

Library Dependencies

- Generally tooling available
  - Maven display-plugin-updates
  - NPM updtr

Base Images

- Easy to use out-of-date base images
- Constant rebuilds?
- Hooks?

PROVE IT!

- Digests are great
  - Content hashes
  - Unwieldy
- GPG signing useful


notary

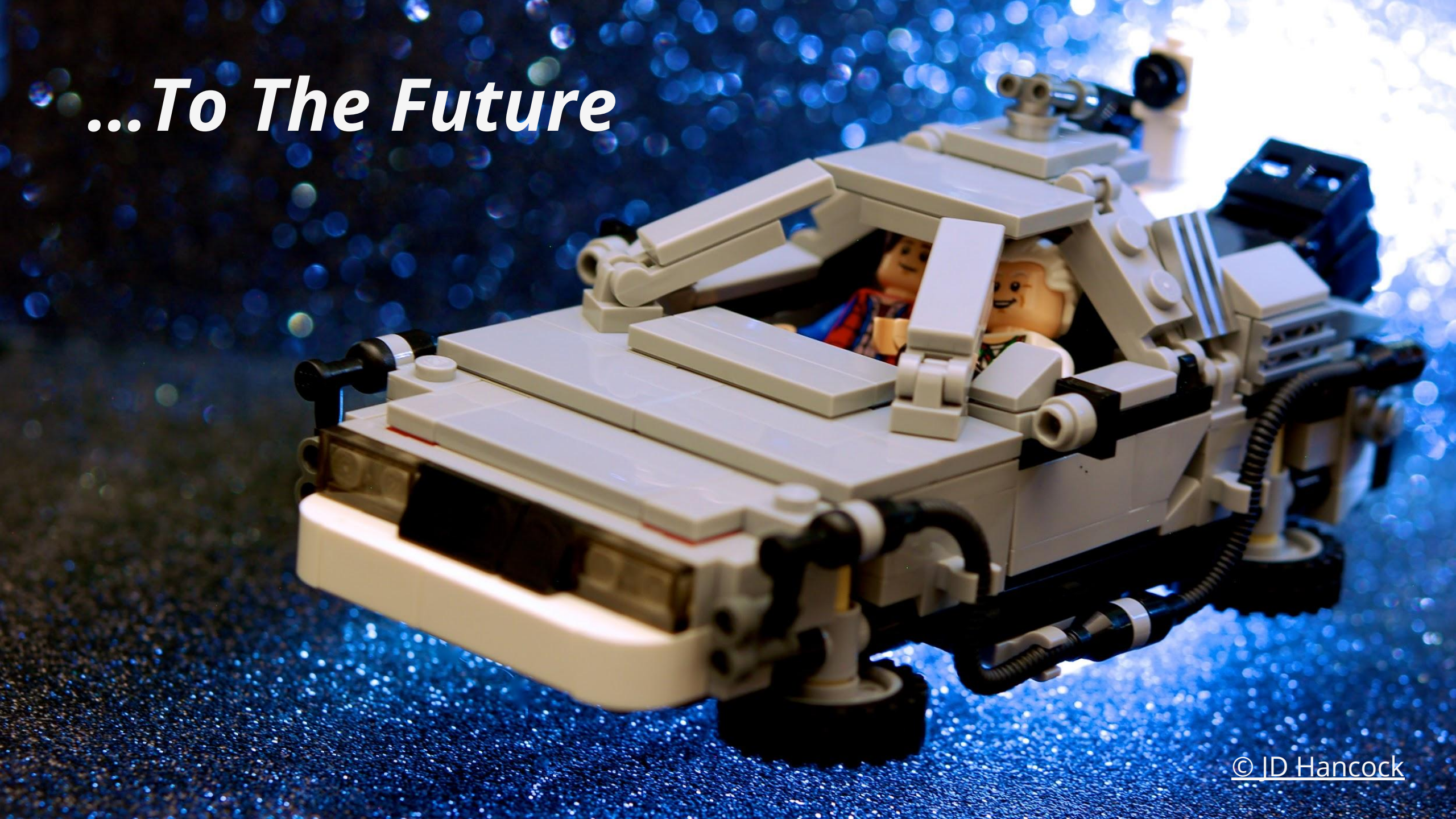
Notary

- Complete signing solution
- TOFU
- Implements TUF
- Protects against range of attacks
  - Including *replay attacks*

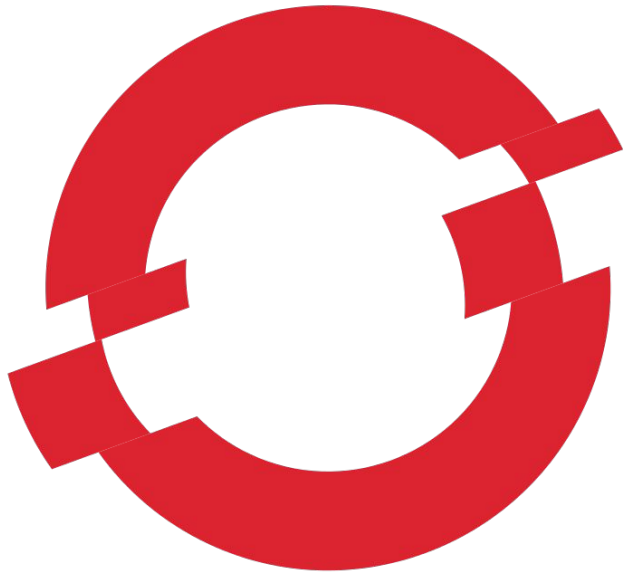
Only run images from a ***controlled*** registry

- Not easily possible
- Should be

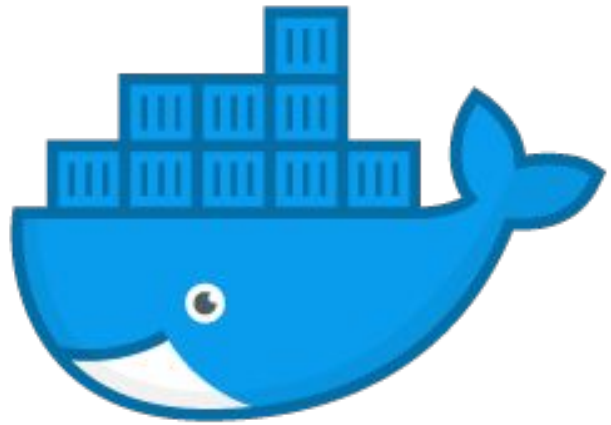
...To The Future



More ***holistic*** solutions



OPENSHIFT



docker

More ***tooling***

Grafeas


HARBOR™

Trow.io

Image Management for Kubernetes Clusters

- **KWTFIGOIYC**
- Use immutable tags
- Use Labels
- Use Tools
  - Notary, registries, scanners

References

- Trow https://trow.io
- Grafeas https://grafeas.io/
- OCI Annotations https://github.com/opencontainers/image-spec/blob/master/annotations.md
- Release Engineering (from Google SRE Book) https://landing.google.com/sre/book/chapters/release-engineering.html
- AlwaysPullImages Admission Controller https://kubernetes.io/docs/admin/admission-controllers/#alwayspullimages
- ImageStreams in OpenShift https://blog.openshift.com/image-streams-faq/
- Docker EE https://www.docker.com/enterprise-edition
- Notary https://github.com/theupdateframework/notary
- Weave Flux https://www.weave.works/oss/flux/
- Clair https://github.com/coreos/clair
- Aqua https://www.aquasec.com/
- NeuVector https://neuvector.com/
- Twistlock https://www.twistlock.com/
- Bazel https://bazel.build/
- Kaniko https://github.com/GoogleContainerTools/kaniko

Workspace File

```
load("@bazel_tools//tools/build_defs/repo:http.bzl", "http_archive")

http_archive(
    name = "io_bazel_rules_docker",
    sha256 = "29d109605e0d6f9c892584f07275b8c9260803bf0c6fcb7de2623b2bedc910bd",
    strip_prefix = "rules_docker-0.5.1",
    urls = ["https://github.com/bazelbuild/rules_docker/archive/v0.5.1.tar.gz"],)

load(
    "@io_bazel_rules_docker//container:container.bzl",
    "container_pull", "container_image",
    container_repositories = "repositories",)
...
```

Build File Pt 1

```
load("@io_bazel_rules_docker//go:image.bzl",
     "go_image")

go_image(
  name = "foo",
  srcs = ["code/main.go"],
  goarch = "amd64",
  goos = "linux",
  pure = "on",
)
```

Build File Pt 2

```
load("@io_bazel_rules_docker//container:container
.bzl", "container_push")

container_push(
  name = "publish",
  image = ":foo",
  format = "Docker",
  registry = "index.docker.io",
  repository = "amouat/go-example",
  tag = "test",)
```

Bazel Run

```
$ bazel run //:publish
INFO: Analysed target //:publish (1 packages loaded).
INFO: Found 1 target...
Target //:publish up-to-date:
  bazel-bin/publish
INFO: Elapsed time: 0.430s, Critical Path: 0.02s
INFO: 0 processes.
INFO: Build completed successfully, 1 total action
INFO: Build completed successfully, 1 total action
index.docker.io/amouat/go-example:test was published with
digest:
sha256:0f2c5d8cdefc0b74eafce7fc65064a734c16770f7401331043f68d10
893f9bc6
```

Bazel Run

```
$ bazel clean
INFO: Starting clean (this may take a while). Consider using
--async if the clean takes more than several minutes.

$ bazel run //:publish
...
index.docker.io/amouat/go-example:test was published with
digest:
sha256:0f2c5d8cdefc0b74eafce7fc65064a734c16770f7401331043f68d10
893f9bc6
```

Bazel Output

```
$ docker save amouat/go-example:test -o test.tar
$ tar tvf test.tar
-rw-r--r-- 0/0                 710 1970-01-01 01:00
5d629c1a7df55c2c46...688a29340.json
drwxr-xr-x 0/0                   0 1970-01-01 01:00
b8e07a381fbd8ca7c0...3eda96f8d3/
-rw-r--r-- 0/0                   3 1970-01-01 01:00
b8e07a381fbd8ca7c0...96f8d3/VERSION
...
$ docker inspect -f "{{.Created}}" amouat/go-example:test
1970-01-01T00:00:00Z
```