Werkzeuge zum Bauen von Container-Images

Tools to Build Container Images

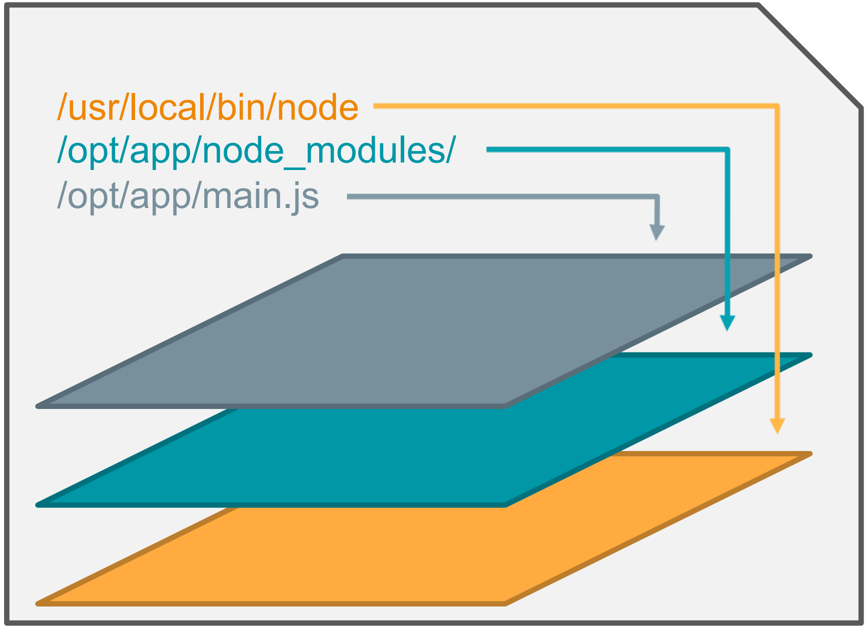
Patrick Harböck / Martin Höfling

TNG TECHNOLOGY CONSULTING

November 13th, 2019
(ContainerConf)

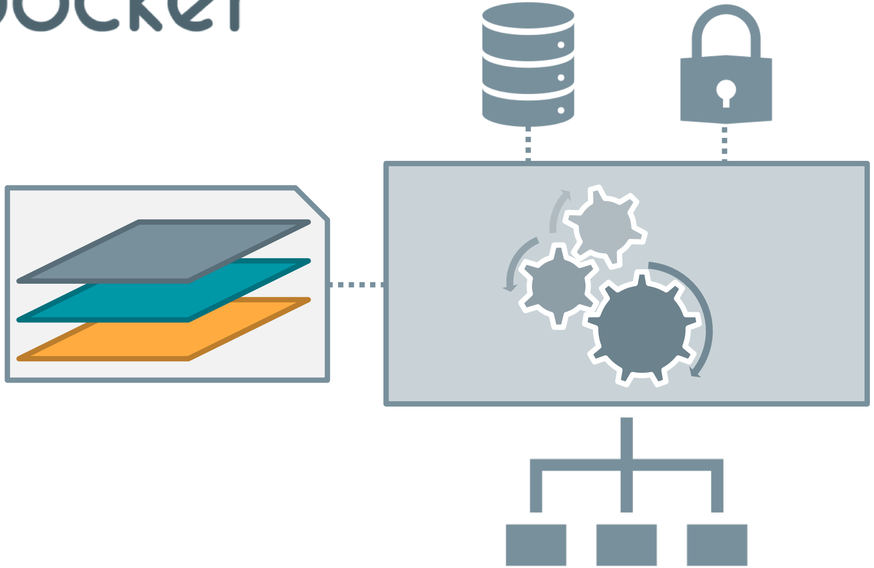
What is Docker?

Container Image Builder

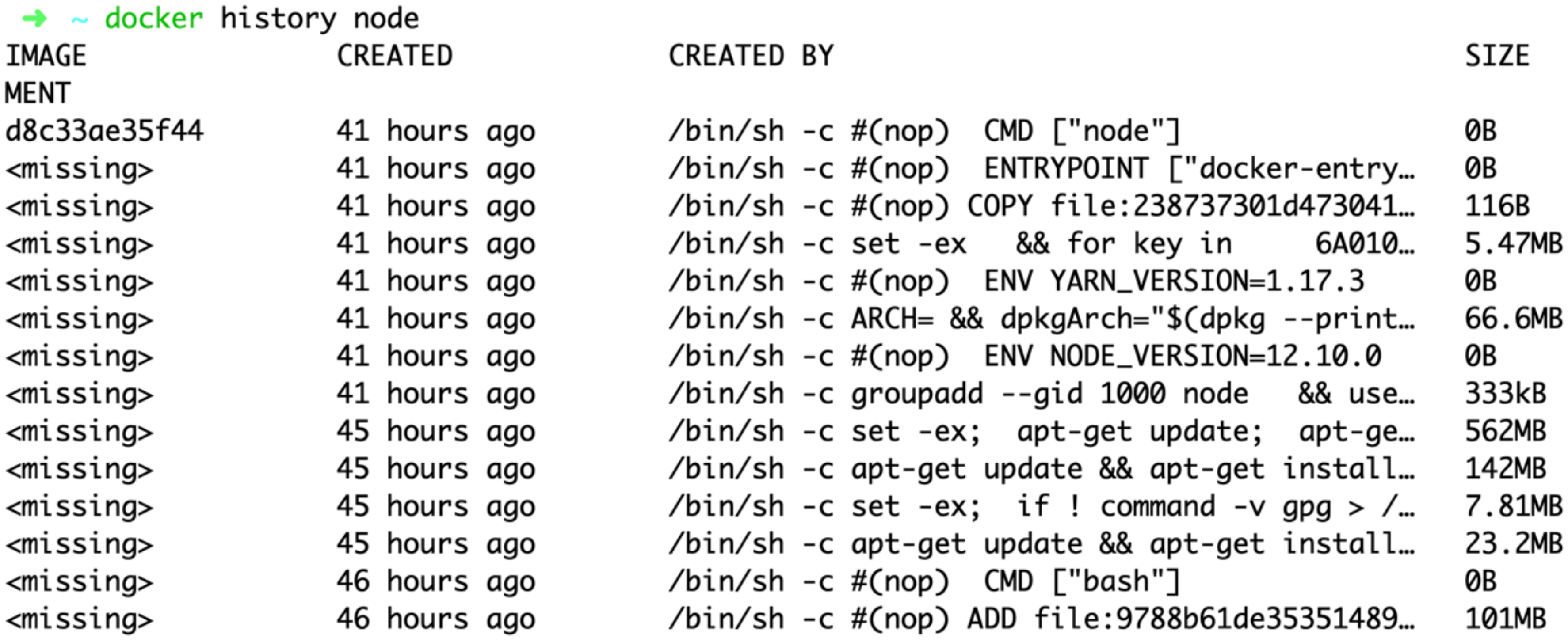


/usr/local/bin/node
/opt/app/node_modules/
/opt/app/main.js

Container Runtime



Docker Images



```
➜  ~ docker history node
IMAGE               CREATED             CREATED BY                                      SIZE
MENT
d8c33ae35f44        41 hours ago        /bin/sh -c #(nop)  CMD ["node"]                 0B
<missing>           41 hours ago        /bin/sh -c #(nop)  ENTRYPOINT ["docker-entry…   0B
<missing>           41 hours ago        /bin/sh -c #(nop) COPY file:238737301d473041…   116B
<missing>           41 hours ago        /bin/sh -c set -ex   && for key in     6A010…   5.47MB
<missing>           41 hours ago        /bin/sh -c #(nop)  ENV YARN_VERSION=1.17.3      0B
<missing>           41 hours ago        /bin/sh -c ARCH= && dpkgArch="$(dpkg --print…   66.6MB
<missing>           41 hours ago        /bin/sh -c #(nop)  ENV NODE_VERSION=12.10.0     0B
<missing>           41 hours ago        /bin/sh -c groupadd --gid 1000 node   && use…   333kB
<missing>           45 hours ago        /bin/sh -c set -ex;  apt-get update;  apt-ge…   562MB
<missing>           45 hours ago        /bin/sh -c apt-get update && apt-get install…   142MB
<missing>           45 hours ago        /bin/sh -c set -ex;  if ! command -v gpg > /…   7.81MB
<missing>           45 hours ago        /bin/sh -c apt-get update && apt-get install…   23.2MB
<missing>           46 hours ago        /bin/sh -c #(nop)  CMD ["bash"]                 0B
<missing>           46 hours ago        /bin/sh -c #(nop) ADD file:9788b61de35351489…   101MB
```

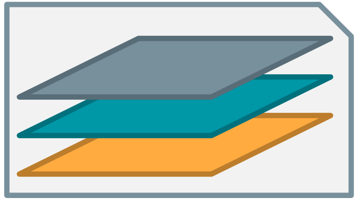
Container Images

- Manifest / Metadata
  - Default configuration for creating containers
  - Content hashes of layers to ensure integrity

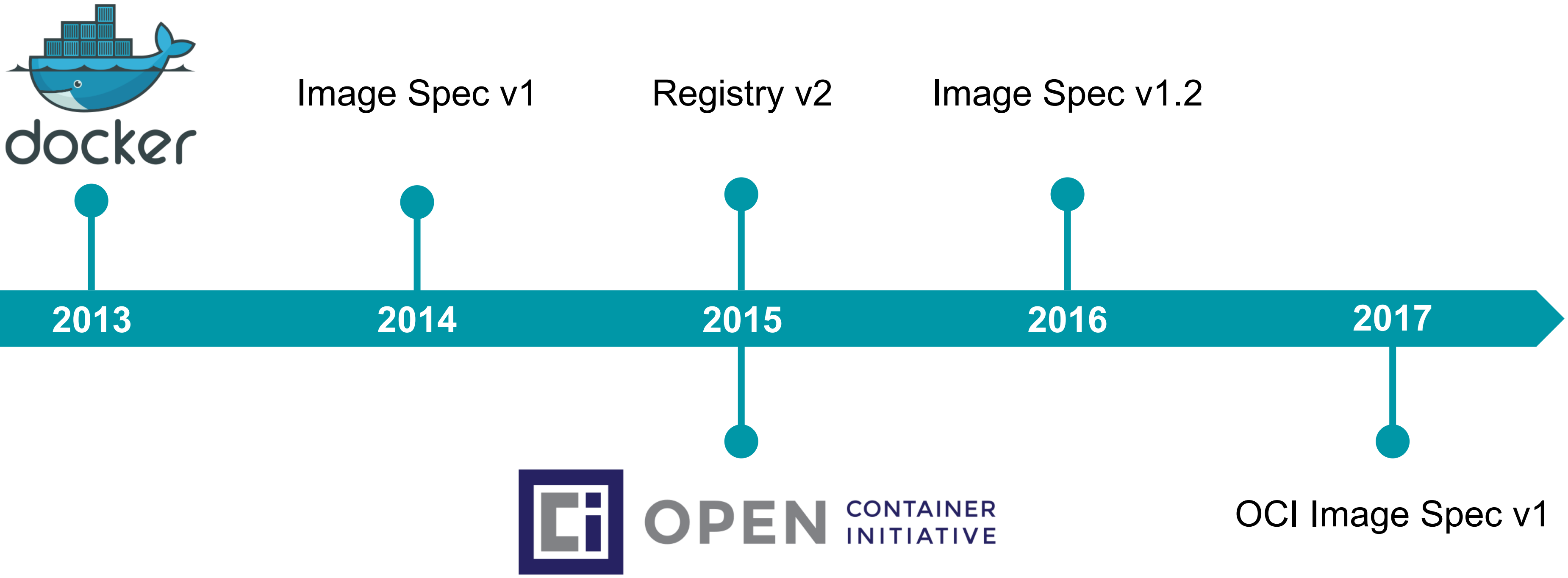
```
ENV | WORKDIR | USER | CMD
```

```
Layer1: 7d97e98f8af71
Layer2: e703abc8f639e
```

- Layers
  - File system packed with *tar*
  - Multiple layers → root file system for containers



Container Image Format Evolution



Open Container Initiative

九州云 Cloud
Alibaba Cloud
amazon web services™
CISCO
D2 IQ
DELL EMC

docker
EasyStack open cloud computing
facebook.
FUJITSU
Goldman Sachs
Google

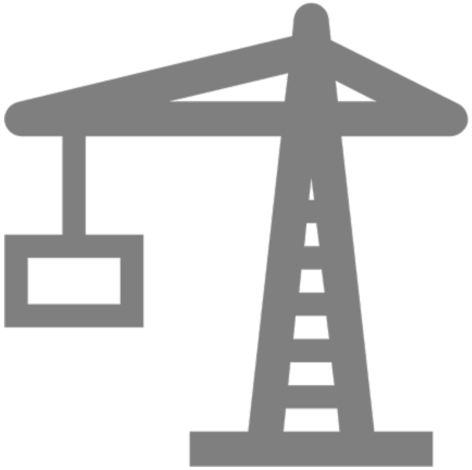
HUAWEI
IBM
InfoSiftr
intel
Joyent
Microsoft

openstack.
ORACLE®
Pivotal.
portworx
Red Hat
REPLICATED

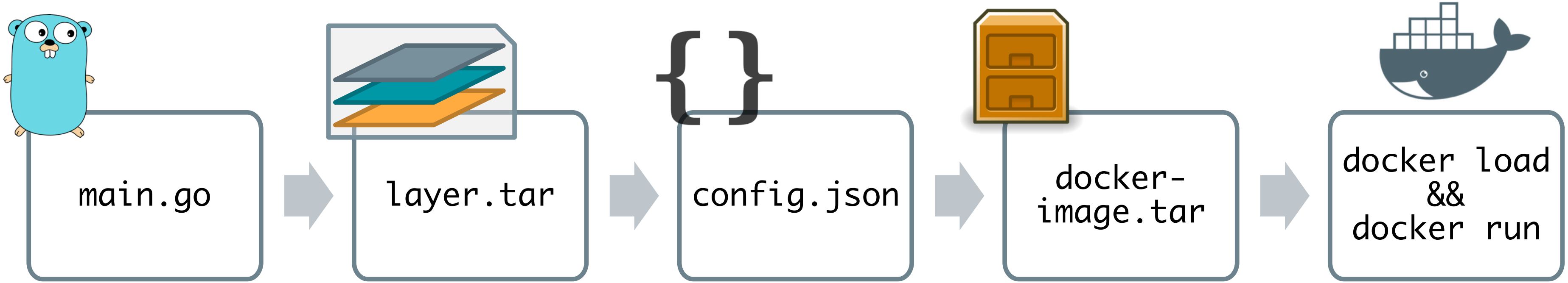
SUSE.
sysdig
Twistlock
vmware®
weaveworks

https://www.opencontainers.org/about/members

DEMO: Build a Container Image from Scratch



DEMO: Build a Container Image from Scratch



main.go

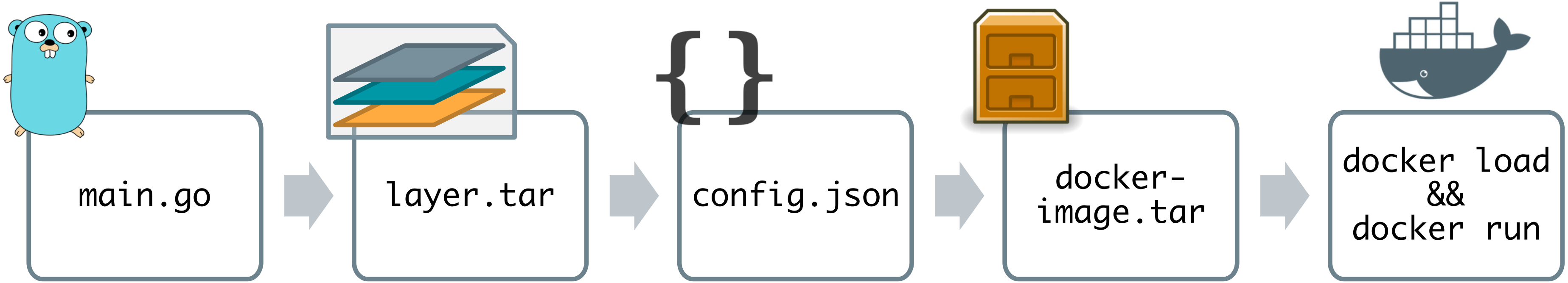
layer.tar

config.json

docker-image.tar

docker load
&&
docker run

DEMO: Build a Container Image from Scratch



`main.go` → `layer.tar` → `config.json` → `docker-image.tar` → `docker load && docker run`

- No elevated privileges required
- No Dockerfile
- No *docker build*

What’s wrong with building images via Docker?

➔ **Security**

➔ **Scalability**

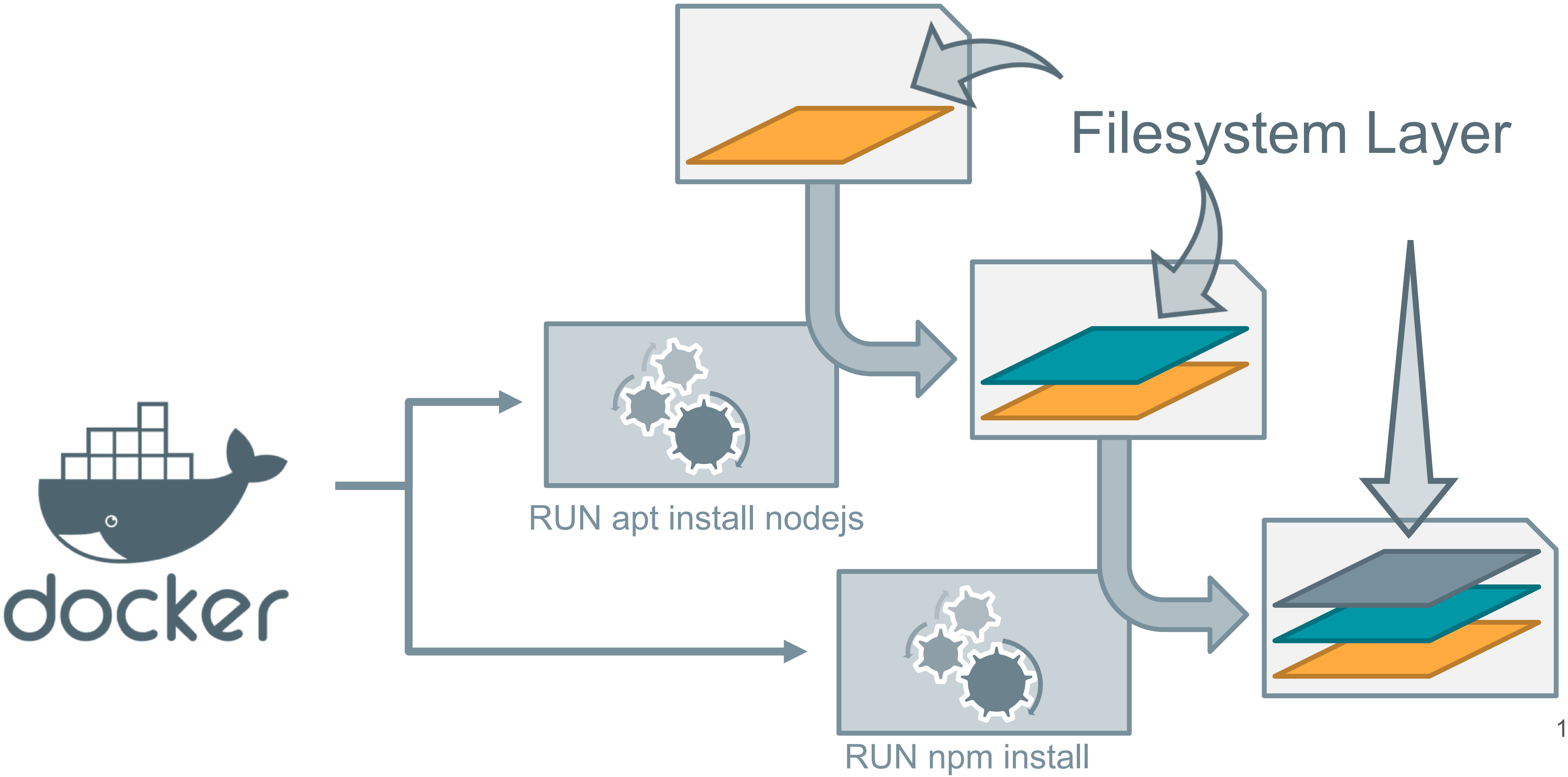
➔ **Flexibility**

➔ **Security**

➔ Scalability

➔ Flexibility

How Docker builds container images



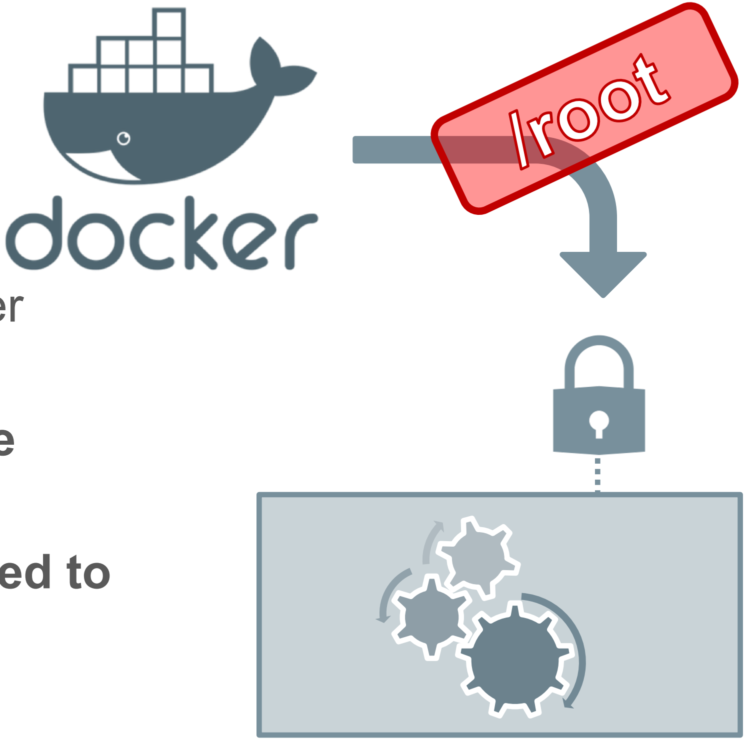
How Docker builds container images

- docker build uses Docker containers
- Docker containers require isolation
- Docker requires elevated privileges
- Build pipelines / developers can access Docker

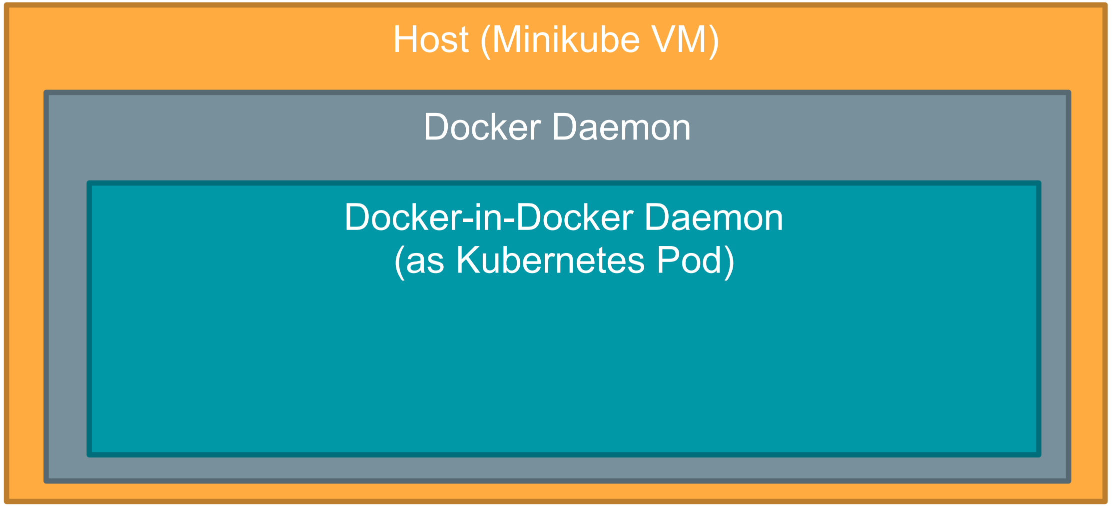
➔ **Security nightmare on shared infrastructure**

“First of all, **only trusted users should be allowed to control your Docker daemon.**”

https://docs.docker.com/engine/security/security/#docker-daemon-attack-surface



DEMO: Host Access via privileged container



Host (Minikube VM)

Docker Daemon

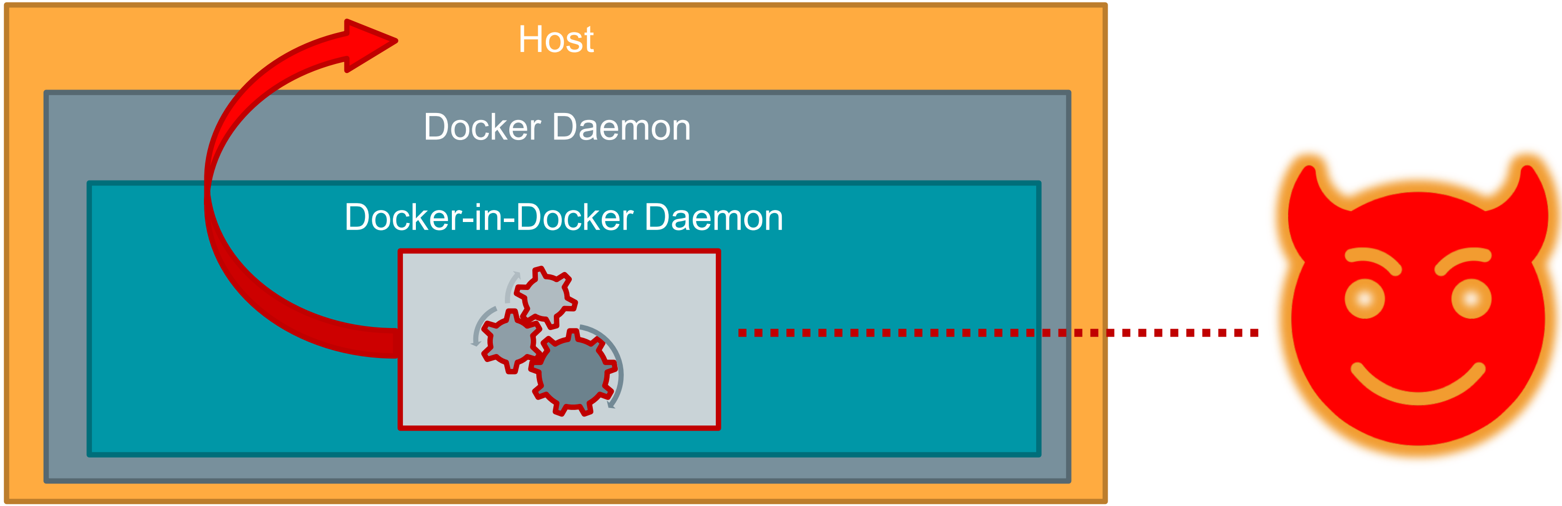
Docker-in-Docker Daemon
(as Kubernetes Pod)

Docker in Docker Kubernetes Pod Spec

```
apiVersion: v1
kind: Pod
metadata:
  name: dind
spec:
  hostname: dind-pod
  containers:
    - name: dind
      image: docker:dind
      securityContext:
        privileged: True
      ports:
        - containerPort: 2375
```

/root

DEMO: Host Access via privileged container



Security Risks?

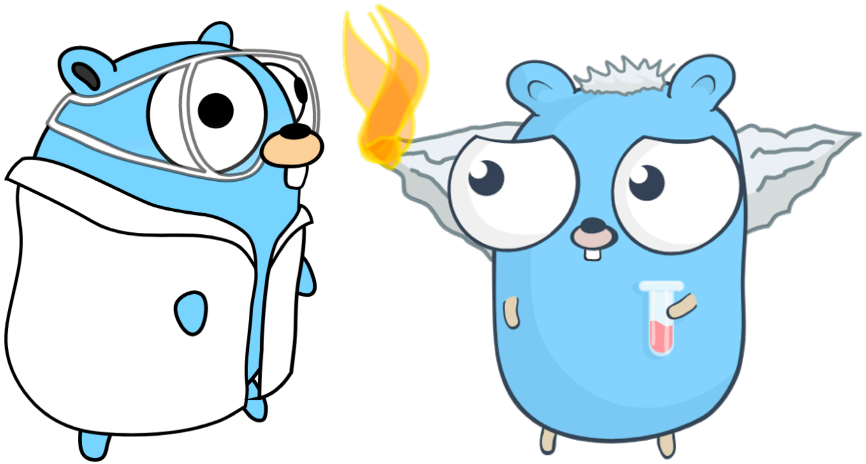
- Privileged Docker-in-Docker → **full host access**
- Mounting or exposing Docker socket → **full host access**
- Base image runs as container *root* → larger vulnerability surface

→ Easy to break and lose container isolation

Remark: Hermetic Builds and Reproducibility

→ Hermetic: sandboxed build process

→ Reproducible builds result in verifiable artifacts



➔ Security

➔ Scalability

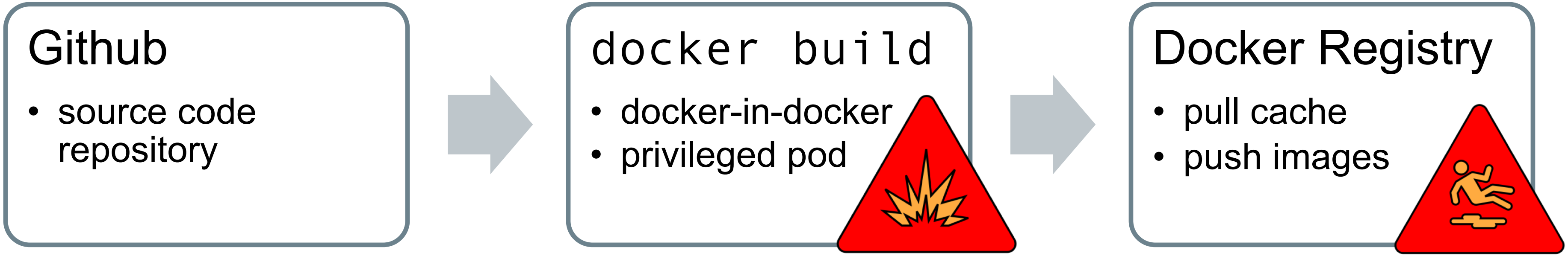
➔ Flexibility

Caching

- Allows scaling up CI/CD pipelines
- Reuse base layers across different branches and builds
- Reproducible builds improve caching

Build Pipeline

kubernetes + **Jenkins**



Scalability Issues

- One Docker daemon does not scale for parallel builds
- No distributed caching support

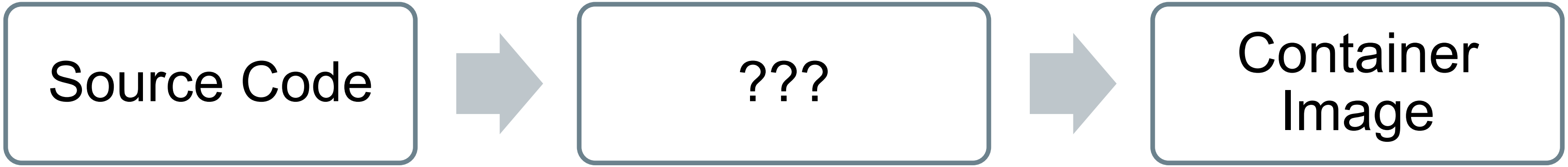
➔ Security

➔ Scalability

➔ **Flexibility**

Flexibility

- How restricted is the build process and image definition?
    - Can developers use any tools and languages they want?
    - How well does it integrate into an existing development pipeline?


Source Code

???

Container Image

Dockerfile based Tools

- Extract base layer(s)
- Run a command in sub container or directly
- Snapshot Filesystem

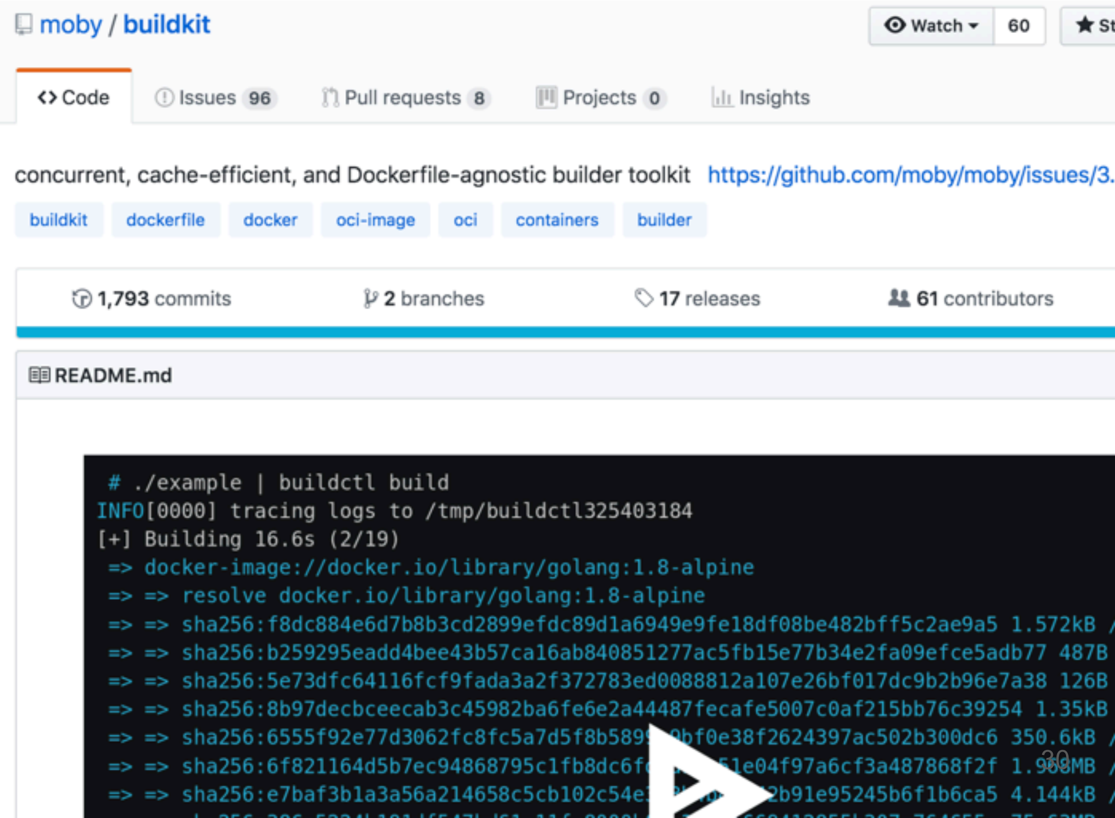
✓ Generic

– Problem: Supported Dockerfile Features?
  - USER – run commands as specific user
  - Multistage builds
  - Root vs. non-root

```
FROM ubuntu:18.04

RUN apt-get update
RUN apt-get install –y nginx

COPY nginx.conf /etc/nginx/

EXPOSE 8080

ENTRYPOINT [‘/usr/bin/nginx’]
```

Tool	Primary Maintainer	Security	Scalability	Flexibility
BuildKit	Docker	😐	🙂	Dockerfile

- Focus on scalability, performance, extensibility
- Experimental support in newer Docker versions and with *docker buildx*
- Optional rootless mode

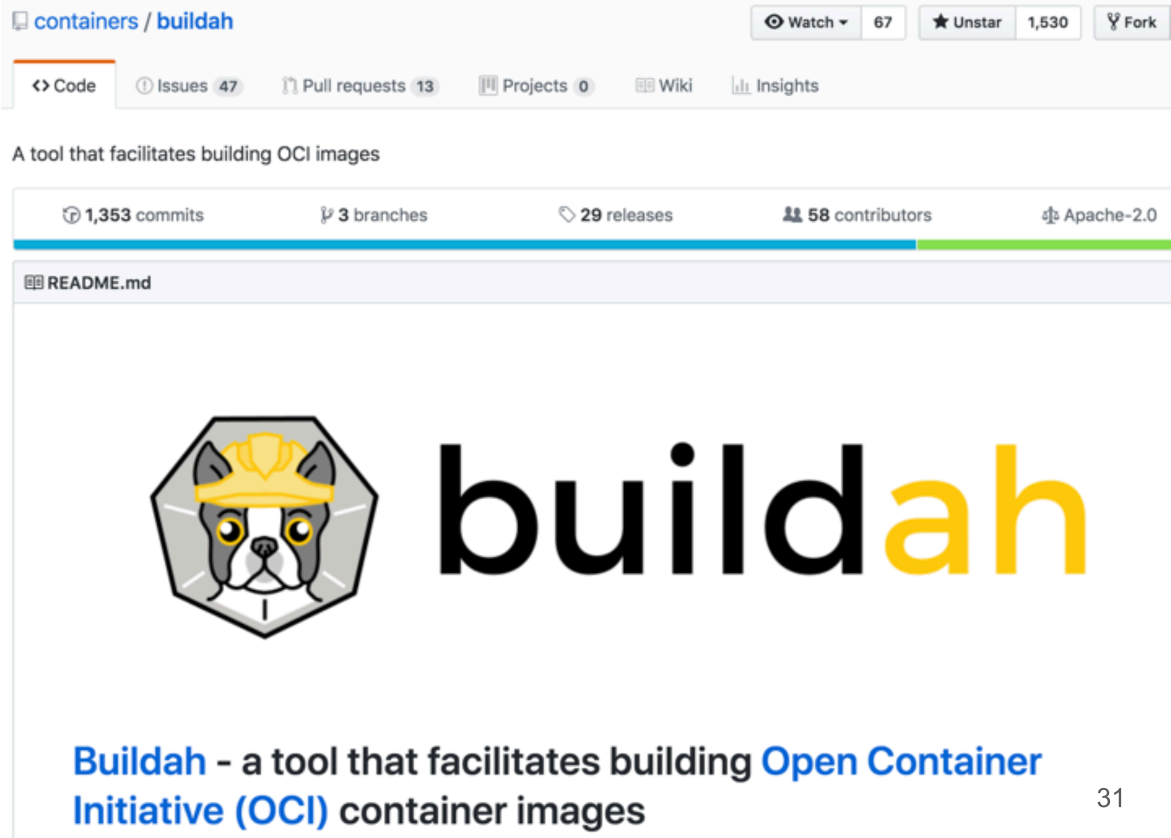


concurrent, cache-efficient, and Dockerfile-agnostic builder toolkit https://github.com/moby/moby/issues/3

```
 # ./example | buildctl build
INFO[0000] tracing logs to /tmp/buildctl325403184
[+] Building 16.6s (2/19)
 => docker-image://docker.io/library/golang:1.8-alpine
 => => resolve docker.io/library/golang:1.8-alpine
 => => sha256:f8dc884e6d7b8b3cd2899efdc89d1a6949e9fe18df08be482bff5c2ae9a5 1.572kB
 => => sha256:b259295eadd4bee43b57ca16ab840851277ac5fb15e77b34e2fa09efce5adb77 487B
 => => sha256:5e73dfc64116fcf9fada3a2f372783ed0088812a107e26bf017dc9b2b96e7a38 126B
 => => sha256:8b97decbceecab3c45982ba6fe6e2a44487fecafe5007c0af215bb76c39254 1.35kB
 => => sha256:6555f92e77d3062fc8fc5a7d5f8b589[illegible]0bf0e38f2624397ac502b300dc6 350.6kB
 => => sha256:6f821164d5b7ec94868795c1fb8dc6f[illegible]1e04f97a6cf3a487868f2f 1.9[illegible]MB
 => => sha256:e7baf3b1a3a56a214658c5cb102c54e[illegible]2b91e95245b6f1b6ca5 4.144kB
```

Tool	Primary Maintainer	Security	Scalability	Flexibility
Buildah	Red Hat	☺	☺	Dockerfile

- Secure and flexible builds of OCI images
- Intended as a Docker replacement together with Podman

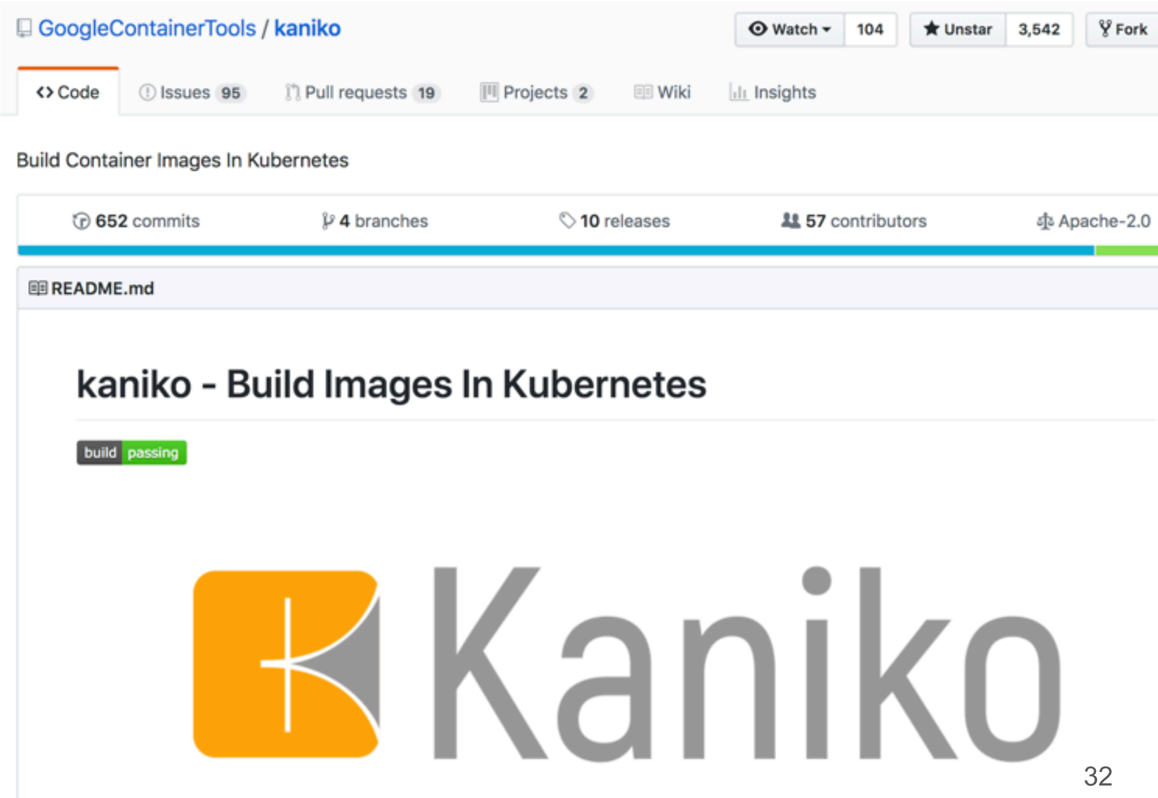


A tool that facilitates building OCI images

Buildah - a tool that facilitates building Open Container Initiative (OCI) container images

Tool	Primary Maintainer	Security	Scalability	Flexibility
Kaniko	Google	🙂	🙂	Dockerfile

- Designed for Kubernetes
- Compatible with
    - AppArmor / SELinux
    - gVisor
- Focus on security and performance
- Reproducible builds



GoogleContainerTools / kaniko

Build Container Images In Kubernetes

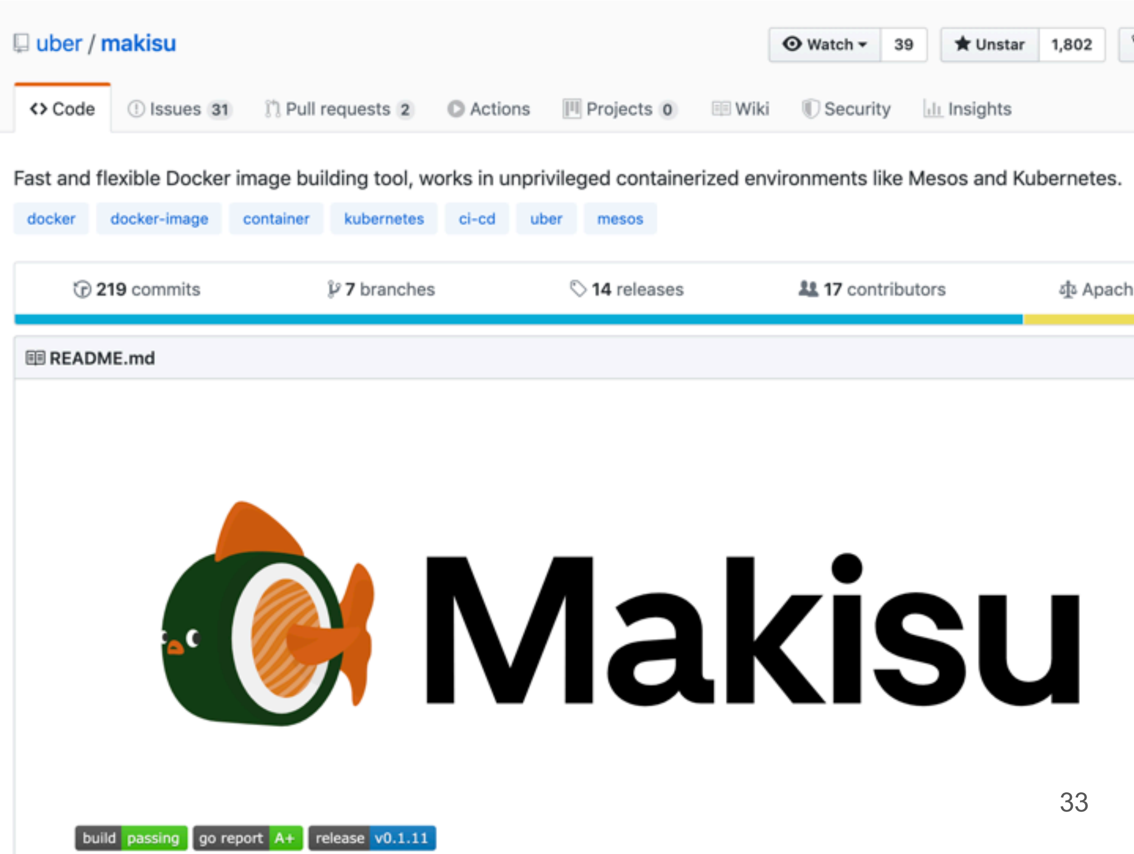
652 commits · 4 branches · 10 releases · 57 contributors · Apache-2.0

README.md

kaniko - Build Images In Kubernetes

Tool	Primary Maintainer	Security	Scalability	Flexibility
Makisu	Uber	☺	☺	Dockerfile

- Focus on security and performance
- Dockerfile support with opinionated modifications
- Distributed caching of layers



Fast and flexible Docker image building tool, works in unprivileged containerized environments like Mesos and Kubernetes.

Tailored image construction

- Tailored for a distinct language and build-system
- The actual **build is not performed in a (child) container**
- The build result is often combined with a base image
    - e.g. Node.js runtime + node_modules + application

No arbitrary command execution required

Limited flexibility

Tool	Primary Maintainer	Security	Scalability	Flexibility
Jib	Google	☺	☺	Java only

- Maven / Gradle plugin
- Distroless Java base image

Builds are

- ✓ Minimal
- ✓ Reproducible
- ✓ Fast (caching)

GoogleContainerTools / jib

Build container images for your Java applications.

936 commits | 18 branches | 63 releases | 37 contributors

Jib

Containerize your Java application.

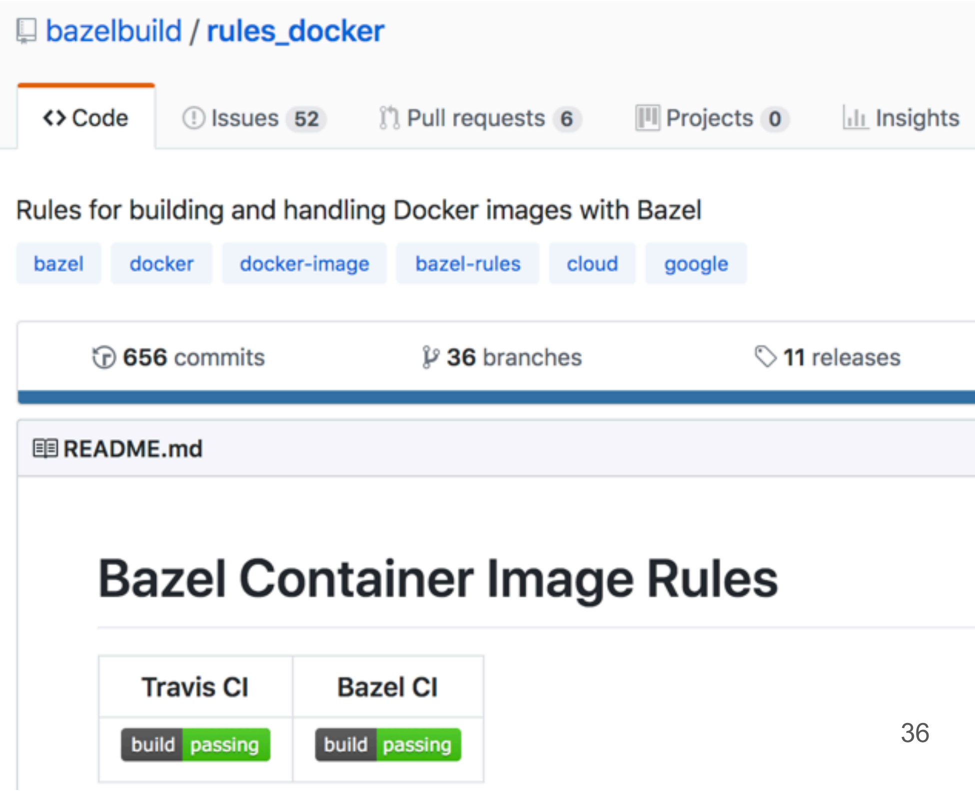
Tool	Primary Maintainer	Security	Scalability	Flexibility
Bazel	Google	☺	☺	Starlark rules

- Supports Python, Node.js, Java, C/C++, Go, Rust, …

Builds are

- ✓ Minimal
- ✓ Reproducible
- ✓ Fast (caching)

– Complex rules written in Starlark



bazelbuild / **rules_docker**

Rules for building and handling Docker images with Bazel

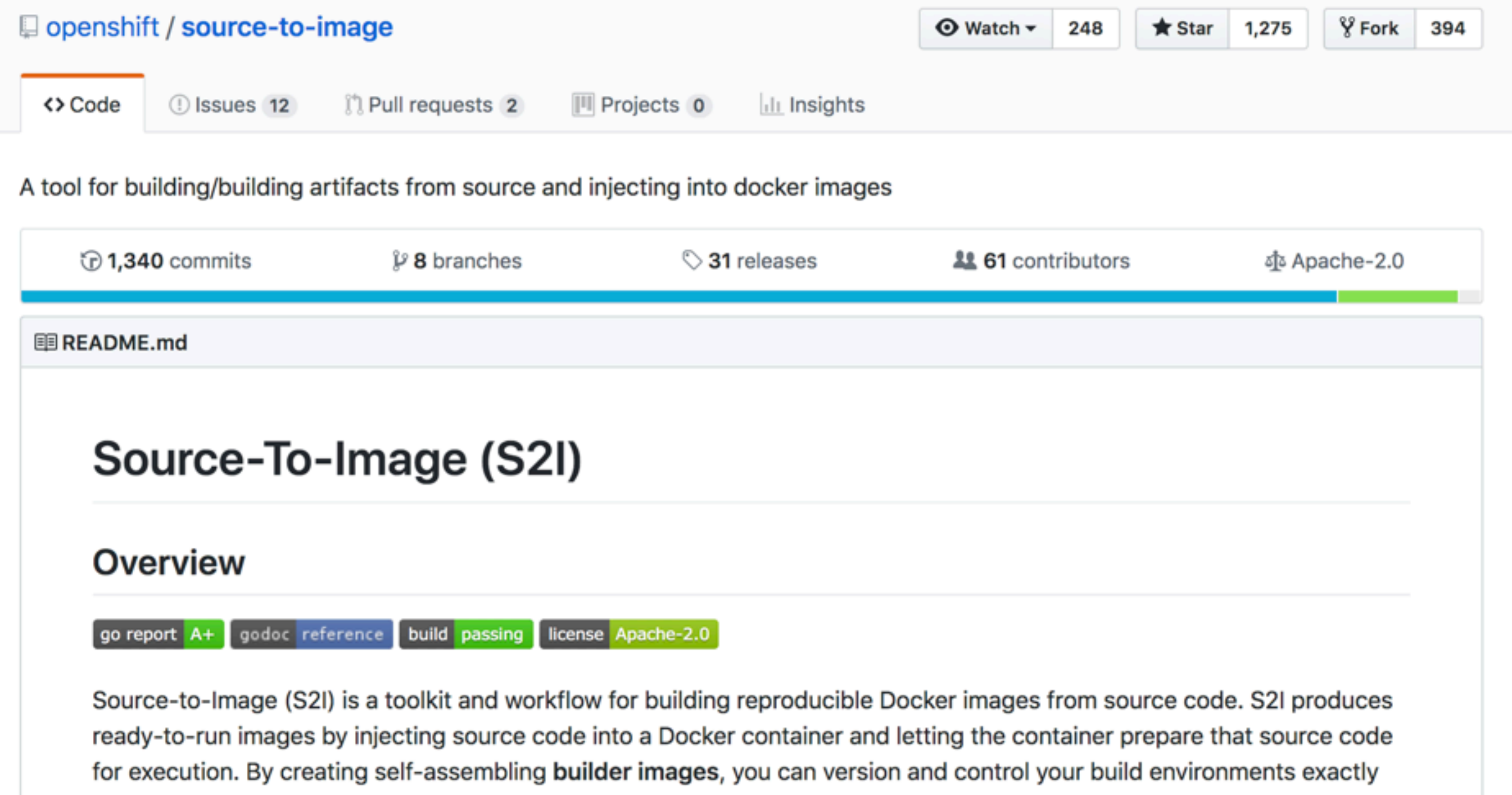
656 commits **36** branches **11** releases

README.md

Bazel Container Image Rules

Travis CI	Bazel CI
build passing	build passing

Tool	Primary Maintainer	Security	Scalability	Flexibility
OpenShift Source-to-Image	Red Hat	☺	☺	Common stacks



openshift / **source-to-image**

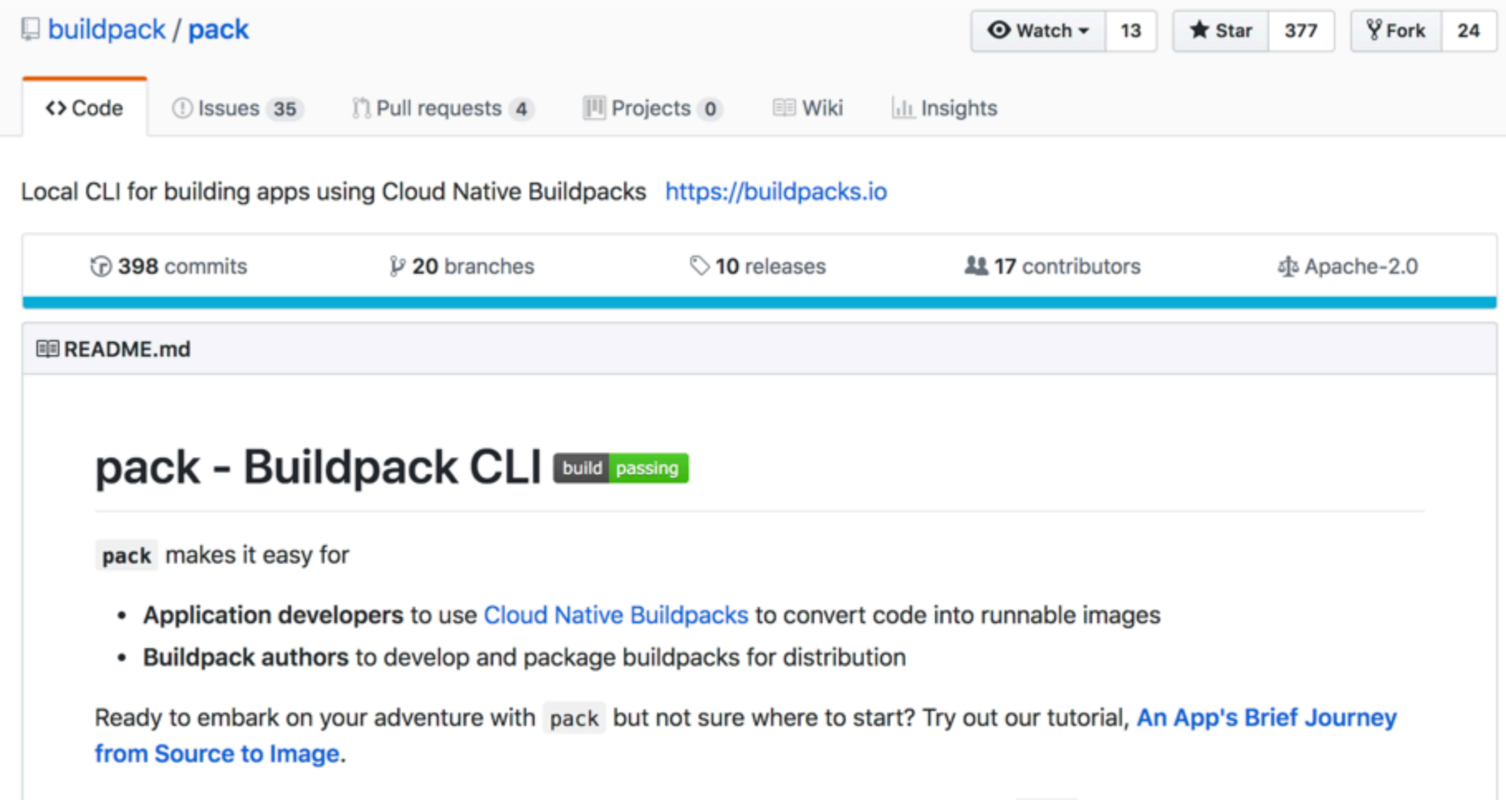
A tool for building/building artifacts from source and injecting into docker images

Source-To-Image (S2I)

Overview

Source-to-Image (S2I) is a toolkit and workflow for building reproducible Docker images from source code. S2I produces ready-to-run images by injecting source code into a Docker container and letting the container prepare that source code for execution. By creating self-assembling **builder images**, you can version and control your build environments exactly

Tool	Primary Maintainer	Security	Scalability	Flexibility
Cloud Native Buildpacks	Heroku / Pivotal / CNCF	🙂	🙂	Common stacks



Local CLI for building apps using Cloud Native Buildpacks https://buildpacks.io

pack - Buildpack CLI

`pack` makes it easy for

- **Application developers** to use Cloud Native Buildpacks to convert code into runnable images
- **Buildpack authors** to develop and package buildpacks for distribution

Ready to embark on your adventure with `pack` but not sure where to start? Try out our tutorial, **An App's Brief Journey from Source to Image**.

Tool	Primary Maintainer	Security	Scalability	Flexibility
docker build	Docker	🙁	🙁	Dockerfile
docker buildx (BuildKit)	Docker	😐	🙂	Dockerfile
Buildah	Red Hat	🙂	🙂	Dockerfile
Kaniko	Google	🙂	🙂	Dockerfile
Makisu	Uber	🙂	🙂	Dockerfile
Jib	Google	🙂	🙂	Java only
Bazel	Google	🙂	🙂	Starlark rules
OpenShift Source-to-Image	Red Hat	🙂	🙂	Common stacks
Cloud Native Buildpacks	Heroku / Pivotal / CNCF	🙂	🙂	Common stacks

What should I use now?

Use case: Small Team

- No strict security requirements for team isolation
- Teams have full access to CI infrastructure
  - ➔ **Docker**
Still a valid choice
  - ➔ **Buildah**
Flexible, only parts of Dockerfile syntax supported securely
  - ➔ **BuildKit**
Are you feeling adventurous? Potential transition path with *docker buildx*

Use case: Multiple teams, Provided K8s infrastructure

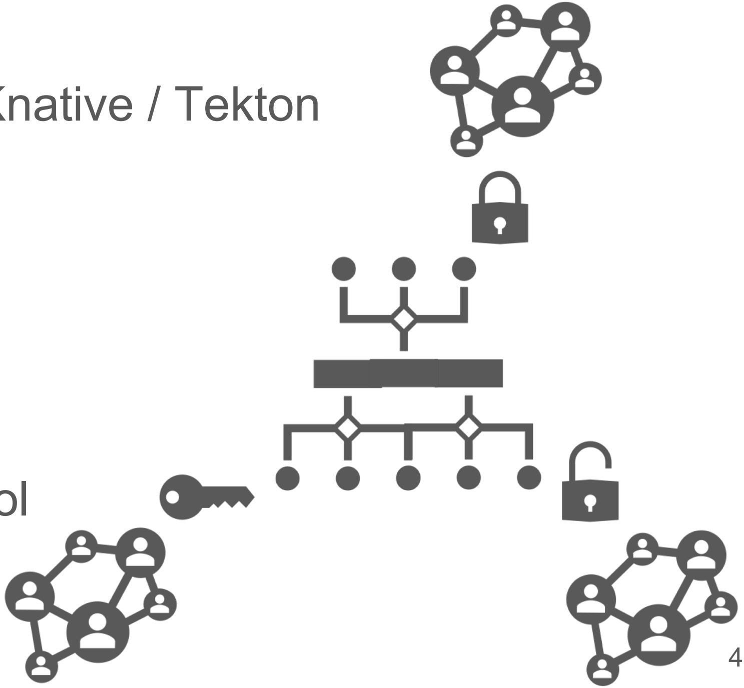
- Cannot modify K8s infrastructure, no privileged containers, no container nesting
- Teams are isolated, e.g. on namespace level

➔ **Kaniko**, e.g. combined with Skaffold or Knative / Tekton

  - Shared volume caching
    (e.g. on Google Cloud Platform)

➔ Take a look at: **Makisu**

  - Fine grained distributed cache control



Use case: No Dockerfile required

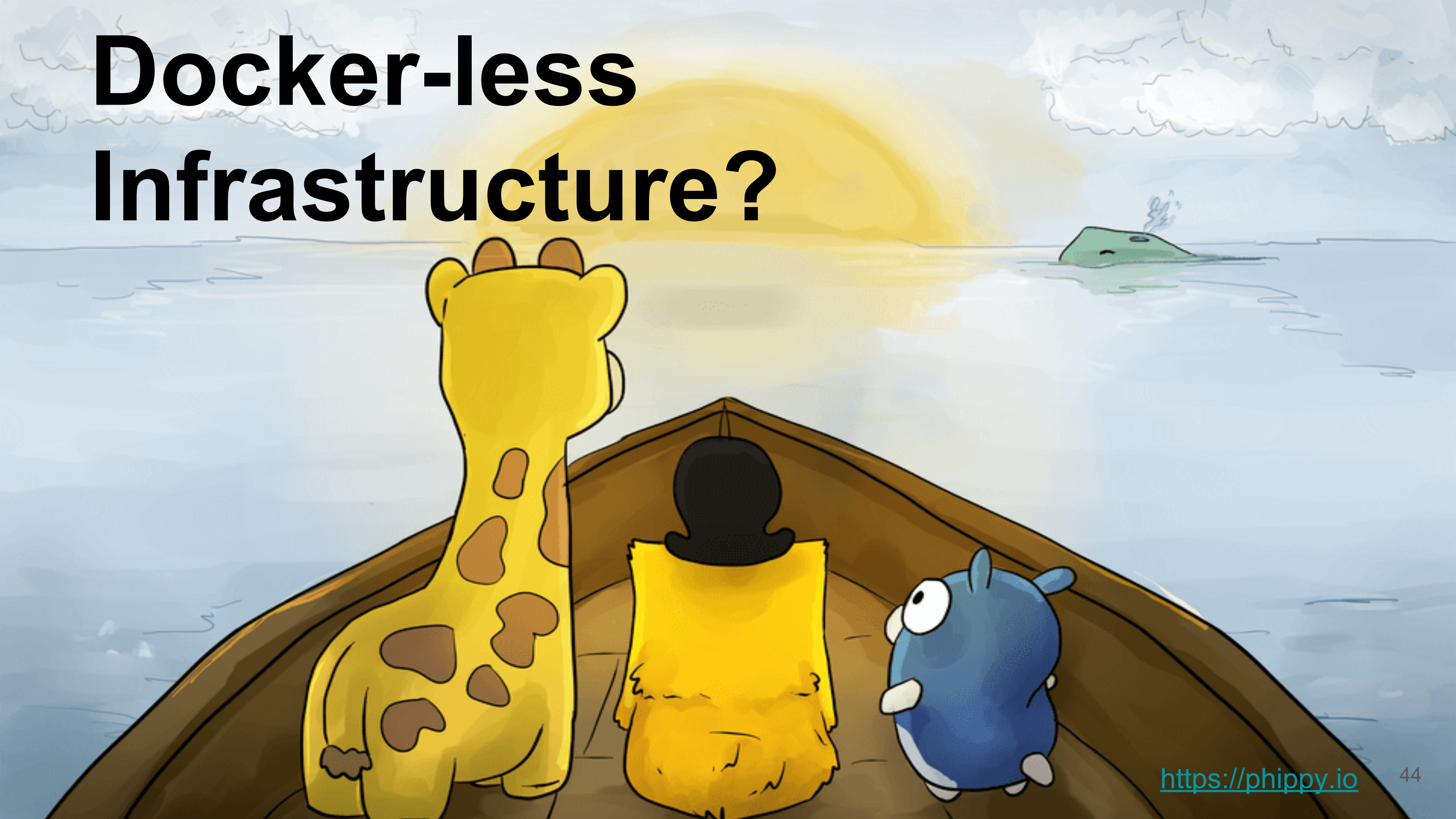
e.g. modern dev stack, Container Native Team

- ➔ **Bazel**
- ➔ **Jib**
- ➔ **Cloud Native Buildpacks**

~~`apt-get install python-dev`~~

No classical ops pattern!

Docker-less Infrastructure?



https://phippy.io

Re-evaluate your container build process!

- [/martinhoefling]()
- [/Pharb]()
- [/TNG]()

TNG TECHNOLOGY CONSULTING