
IMAGINE


THE NEXT LEVEL


LET DOWN

WHY BOTHER, AND HOW TO

MULTI-ARCH CONTAINER IMAGES

@thedoh

ABOUT ME

Lisa Seelye 🦪 thedoh

Sr. SRE at Red Hat

Cat enthusiast

Alternate arch enthusiast

Sysadmin & Sw. Eng background





DOCKER, DOCKER

DOCKER, DOCKER

THE GROUND RULES



LOW LEVEL

WHAT LEVEL ARE WE ON?

LET'S HAVE A LOOK

CONTAINER IMAGES



CONTAINER SHIPS
SHIPPING CONTAINERS

DO YOU THINK THIS?

BUT, WHAT IF IT'S ALL A LIE?



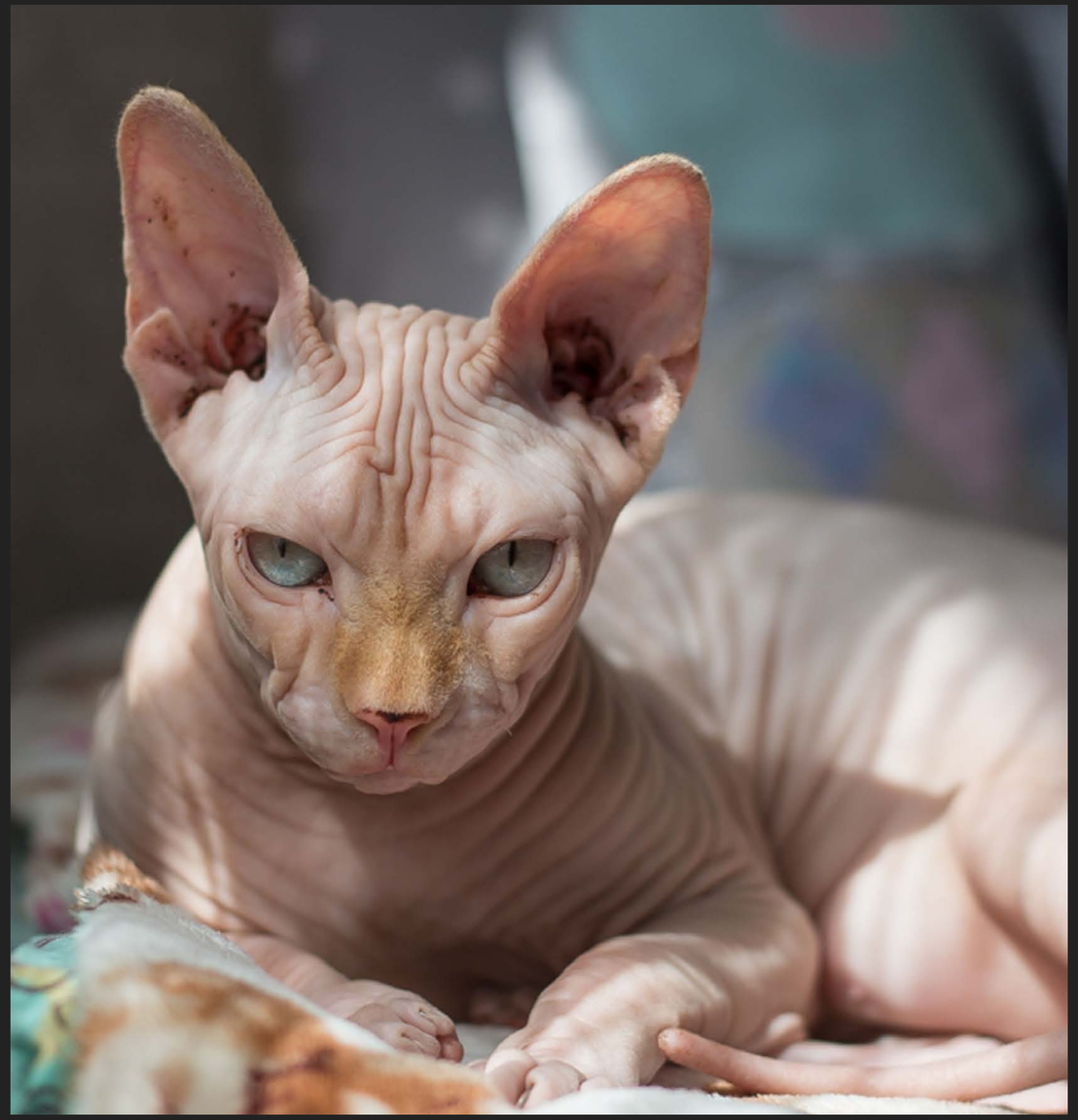
THIS IS YOUR IMAGE

```
$ tar xvf image.tar
x sha256:4fd1e92090d4d5128fac6b023f9067456ef...
x 43ef5679e440ba5060f276db0a83af9588ee9ec650....tar.gz
x 1b8349f9c8bdb67c8167d2ec1234d8dc94deee42bc....tar.gz
x manifest.json
$
```

saved with go-containerregistry pkg/v1/tarball

INSIDE THE IMAGE TARBALL

- JSON Config file (sha256:somehash)
- Layer tarballs (yaddayadda.tar.gz)
- JSON manifest file (manifest.json)



BUT LISA, HOW DO I GET AN IMAGE??

WE HAVE TO GO DEEPER

DOCKER PULL THEDOH/LISA19

- No Manifest List
  - Widely used
  - Gives you the requested image, no questions asked
- Manifest list
  - Not as widely used 😓
  - Gives you layers appropriate for your platform (sometimes)





THEDOH/LISA19:AMD64-19.08.1

RUNNING ON AMD64



THEDOH/LISA19:AMD64-19.08.1

SAD TIMES ON ARM64

@thedoh

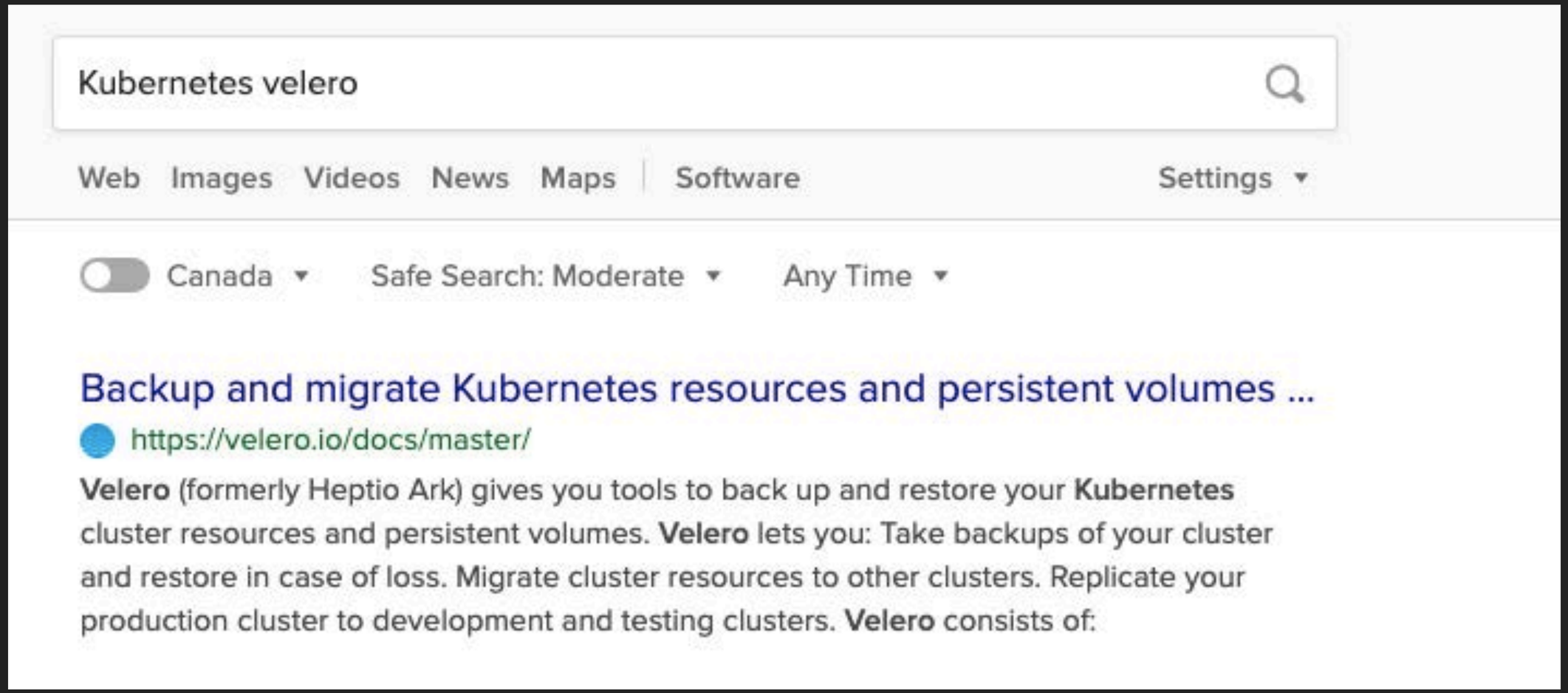
REAL WORLD EXPERIENCES

ALTERNATE ARCHITECTURE EXPERIENCE

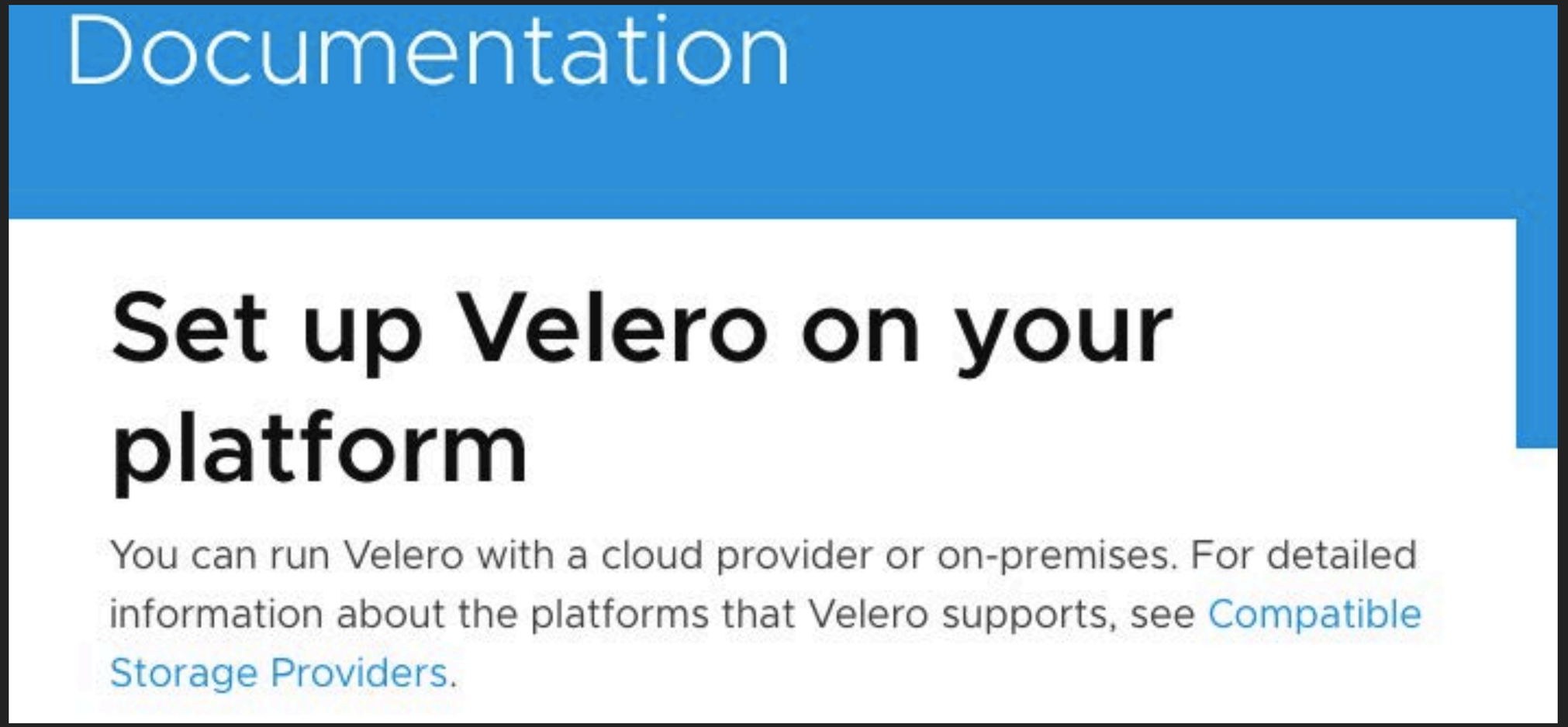
SOFTWARE ATTEMPTED

- Kubernetes Dashboard
- Velero and Helm (v2)
- Tekton Pipelines & Kaniko





Searching for Velero install docs



Found the docs!

Installation

We strongly recommend that you use an official release of Velero. The tarballs for each release contain the `velero` command-line client. Follow the installation instructions to get started.

More docs! We are close



ARM64!! We're supported!! Yay!!

```
# curl -LO https://github.com/heptio/velero/releases/download/v1.0.0/velero-v1.0.0-linux-arm
 Average Speed   Time    Time     Time  Current
 Dload  Upload   Total   Spent    Left  Speed
  2229       0 --:--:-- --:--:-- --:--:--  2229
 13.5M       0  0:00:01  0:00:01 --:--:-- 22.0M
# tar xzf velero-v1.0.0-linux-arm64.tar.gz
# cd velero-v1.0.0-linux-arm64
/velero-v1.0.0-linux-arm64# ./velero install --bucket mybucket --provider gcp --secret-file
```

Gooooooooo!!!!

```
VolumeSnapshotLocation/default: created
Deployment/velero: attempting to create resource
Deployment/velero: created
Velero is installed! ⛵ Use 'kubectl logs deployment/velero -n velero' to
root@kubemaster:~/install_velero/velero-v1.0.0-linux-arm64#
```

IT WORKED!!!!


elero/velero-v1.0.0-linux-arm64# kubectl logs deployment/velero -n
exec user process caused "exec format error"
elero/velero-v1.0.0-linux-arm64#

Not so fast 😭



WHAT THE HECK DOES THAT MEAN?

EXEC FORMAT ERROR

@thedoh

IT'S ABOUT POSITIVE USER EXPERIENCE

- Kubernetes Dashboard docs pointed me to an amd64 image
- Velero and Helm (v2) both have ARM64 installers, but give amd64 images
- Other tools and libraries make assumptions about the environment





DOCS AND CULTURE SAY ONE THING WHILE IMAGES SAY ANOTHER

GIVE ME A LIST TO PICK FROM, OK?

IMAGE MANIFEST LIST

INSIDE AN IMAGE MANIFEST

```
thedoh/lisa19:arm64-19.10.1:
  Id: sha256:a9f9ac1...
  Cmd: ["/proof"]
  Platform: arm64/linux
  RootFS:
    Layers:
      - sha256:f9d05ed...
      - sha256:2c449ec...
```

ARM64

```
thedoh/lisa19:amd64-19.10.1:
  Id: sha256:7b270b0...
  Cmd: ["/proof"]
  Platform: amd64/linux
  RootFS:
    Layers:
      - sha256:c5a3c5f...
      - sha256:3ebcc8c...
```

AMD64

MANIFEST LISTS

```
thedoh/lisa19:19.08.1:
  manifestList:
    - amd64/linux:
      digest: sha256:7b270b0...
    - arm64/linux:
      digest: sha256:a9f9ac1...
```

MANIFEST LISTS + IMAGE MANIFEST

```
thedoh/lisa19:19.08.1:
  manifestList:
    - amd64/linux:
      digest: sha256:7b270b0...
    - arm64/linux:
      digest: sha256:a9f9ac1...
```

```
thedoh/lisa19:19.08.1:
  manifestList:
    - amd64/linux:
        thedoh/lisa19:arm64-19.10.1:
          Id: sha256:a9f9ac1...
          Platform: arm64/linux
          RootFS:
            Layers:
              - sha256:f9d05ed...
              - sha256:2c449ec...
    - arm64/linux:
        thedoh/lisa19:amd64-19.10.1:
          Id: sha256:7b270b0...
          Platform: amd64/linux
          RootFS:
            Layers:
              - sha256:c5a3c5f...
              - sha256:3ebcc8c...
```

WITH MANIFEST LISTS

- Reference multiple arches with one logical image name
- Image registry gives you the right layer files based on requested platform
- Requesting an unlisted arch fails at pull-time*, not runtime



MANIFEST LISTS, COOL, COOL...

BUT HOW?



...BY HAND

BUILDING MANIFEST LISTS



WITH MAKE

BUILDING MANIFEST LISTS

WHY BOTHER?

BENEFITS OF MANIFEST LISTS

- Seamlessly support multiple architectures
  - Simplifies docs, which people don't like writing anyways
- ARM64 is coming. Amazon has EC2 instances; which vendor will make the next move?
- Maybe shift to an inclusive mindset when coding



I HAVE A CONFESSION

THIS TALK IS ABOUT SOFTWARE DEVELOPMENT PHILOSOPHY, TOO.



THE IDEA APPLIES TO SOFTWARE ENGINEERING

- Opinionated software libraries have their opinions carried forward
- Single architecture assumptions aren't portable
- Examples include:
  - go-containerregistry
  - kaniko, because it uses go-containerregistry

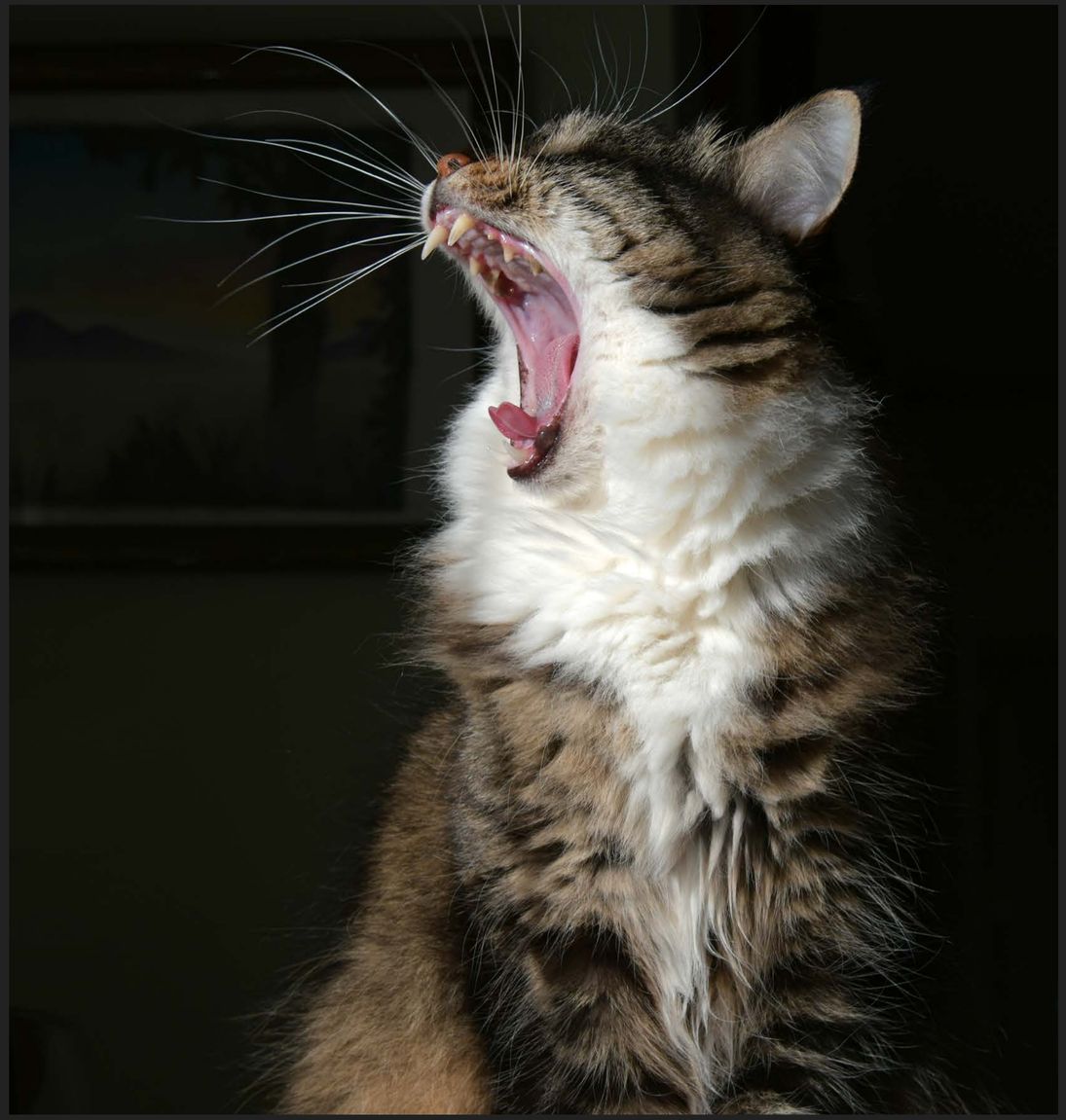


```go
// This naively matches the first manifest with matching platform attributes.
//
// We should probably use this instead:
//   github.com/containerd/containerd/platforms
//
// But first we'd need to migrate to:
//   github.com/opencontainers/image-spec/specs-go/v1
func (r *remoteIndex) childByPlatform(platform v1.Platform) (*Descriptor, error) {
  index, err := r.IndexManifest()
  if err != nil {
    return nil, err
  }
  for _, childDesc := range index.Manifests {
    // If platform is missing from child descriptor, assume it's amd64/linux.
    p := defaultPlatform
    if childDesc.Platform != nil {
      p = *childDesc.Platform
    }
```

github.com/google/go-containerregistry pkg/v1/remote/index.go (6991786)

WHAT IS KANIKO?

- Software that uses go-containerregistry to pull images to build new images
- Builds container images in Kubernetes, without Docker socket
  - Why? The Docker socket =~ root
- Scratches the "let's do everything in Kubernetes" itch
- github.com/GoogleContainerTools/kaniko



TEKTON PIPELINES

- Open source Kubernetes native CI/CD Platform
- Can use kaniko to build container images
- Also only amd64 images
- github.com/tektoncd/pipeline



Photo: Corinne Alstrom-Sonne

PORTING TEKTON PIPELINES TO ARM64

- Depends on google/ko to build
  - But google/ko depends on go-containerregistry
- Building fails in the build system, not software code
- Welcome to dependency hell :(





CREATE & DISTRIBUTE



UNDERSTAND THE CULTURE AND GOALS
TILT TOWARDS CROSS-PLATFORM

ACTIONS FOR US ALL

@thedoh

THE END

MULTI-ARCHITECTURE CONTAINER IMAGES:
WHY BOTHER, AND HOW TO

Code & Makefiles
github.com/lisa/lisa19-containers

Slides & Image Credits
lisa.dev/conferences#lisa19