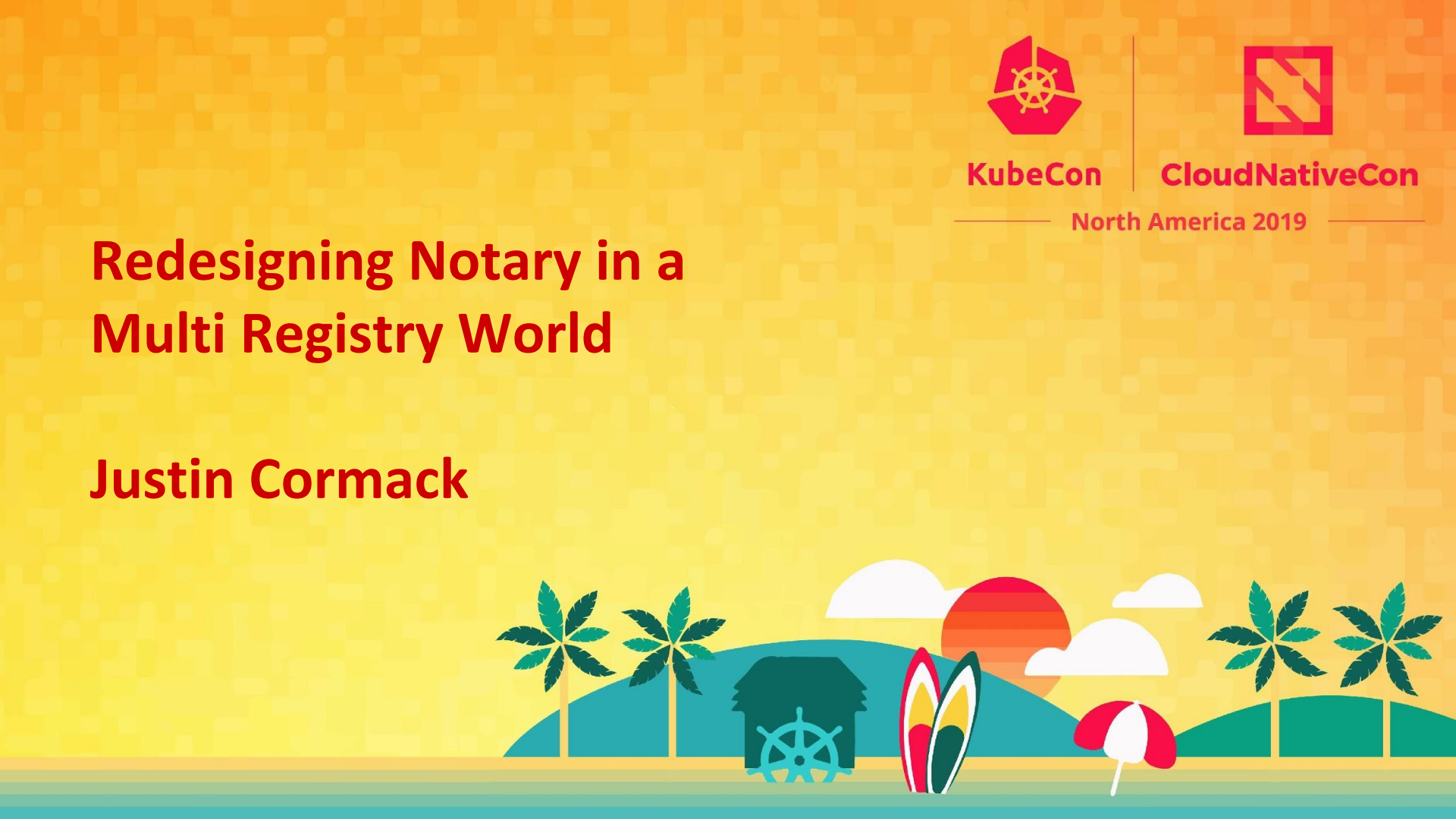

KubeCon
CloudNativeCon
North America 2019




# Redesigning Notary in a Multi Registry World

Justin Cormack

# Who am I?



- Security Lead at Docker
- Maintainer of Notary
- Work with CNCF SIG Security
- Based in Cambridge, UK
- Had a weird week last week

# Helpful discussions with...



- Steve Lasker, Microsoft
- Amazon EC2 Registry team
- Justin Cappos, NYU, TUF
- also some early conversations with IBM, plan to speak with others
- this talk is about suggestions, not a committed roadmap


KubeCon
CloudNativeCon
North America 2019
Supply Chain Security

# Supply chain security



- Back in the pre-cloud native days we had hardware firewalls and fixed infrastructure
- If all your infrastructure is code, then anything can be changed with code
- If an attacker can change your code, she can change *anything*
- The "supply chain" of how your code gets to production becomes incredibly important to secure
- Many supply chain attacks, growing recently
  - NotPetya caused billions in damage in 2018
- We want protections in container ecosystem!


KubeCon
CloudNativeCon
North America 2019
A very short introduction to TUF (and Notary)

# The original TUF problem



- Linux package repositories turn out to have a bunch of security issues
  - serving up fake packages
  - replay attacks – say an old vulnerable package is a new one
  - freeze attack – say there are no updates when package is vulnerable
  - change dependencies, so extra vulnerable packages installed
  - mix and match dependencies from different dates
  - and more ...
- Just having a package *signature* does not stop most of these
  - need to sign a *collection* of items
  - need to re-sign regularly to allow *freshness* checks

# TUF basics



- TUF has a "pull" view of what is going on
- Imagine you are looking at a Linux package repo trying to decide if you trust the updates
- You have a public key, as you already installed the distribution which included it
- So you can check signatures to validate content
- Potential attacks still include hiding all the new updates, labelling an old signed package as new, renaming files
- So TUF adds additional signed metadata, such as current list of files, and regularly re-signs it with delegated keys

# What about in-toto?



- new CNCF Sandbox project
- answers different questions
  - how was this code built?
  - who built it?
  - how did we get from source to artifact?

# *What do I want to fix?*

# How did we get here?



- Notary was originally a Docker project implementing TUF for registries
- Saw that making better security guarantees with TUF was great opportunity
- Launched 2015
- Donated to CNCF along with the TUF specification in 2017
- However, back when Notary was designed
  - containers in production were kind of new
  - we didn't have a good feel for how exactly to use them
  - a bunch of design mistakes were made I think
  - but the basic idea was the right one

# Current issues



- Usability and usage
- Registry native
- Design and usage differences with the TUF formulation
- Observability, understandability and debugging
- Modular approach, use general specifications to build more complex tools

# *Registry native?*

# What is a registry?



- Content addressed storage, with a naming layer. Like git.
- Content looked up by sha256 hash, as layers or other content
- Historically just container images but more general now, see https://github.com/opencontainers/artifacts for ongoing work
- Every leaf object has a *manifest* JSON object pointing at it
- May be other levels of metadata objects, eg *index* pointing at *manifests*
- Mutable naming layer, much like git tags
- Some folk ways to use it, for example we don't usually delete content
- Some implementation choices, like limited hierarchy of namespace
- Client issues with mutating content, eg pull and push changing hashes

# Signatures?



- Unlike git, registries don't have a "native" signing model at all
- Notary is the only available model, and in effect is a sidecar, a database that runs beside a registry but not native to it
- git developed a native signing model, so signatures are attached to commits and tags
- git signing still not used effectively by most users

# What is native signing?



- Signatures inside the registry, can pull and push with rest of content
- Can validate whichever registry it is in
- For signing an index or a manifest there are two options
  - inline signing
  - add a signature manifest pointing at the object to be signed
- For tag signing, there is currently no way to add this
  - tags do not have metadata in a registry AFAIK
  - never design things without metadata support!
  - Notary avoids this by providing its own tag resolution
  - this adds client complexity


KubeCon
CloudNativeCon
North America 2019
Diversion: signing
JSON

# How many ways to sign JSON?



- Registry documents other than leaf (layer) blobs need to be JSON objects
  - this is so data can be parsed for UI and for garbage collection
- Five or so ways to sign JSON
  - detached signature, that is another blob pointing at signed object
  - attach signature blob to JSON, but then cannot be parsed as JSON
  - canonicalize JSON to sign it, with signature removed. TUF does this. Most canonical JSON libraries have bugs.
  - include copy of JSON object in exact serialization
  - swap signature in pre-added space
- See Latacora, How (not) to sign a JSON object

# *Detached signatures*

# The cleanest solution is detached signatures



- Maps best to content addressed storage
- A signature is just some metadata pointing at what it signs
- Do not change any existing documents (much)
  - for most compatibility, you might have an index (manifest list) pointing at all the OS/architectures as now, and in addition pointing at a signature for clients that understand it
  - the signature would point at another manifest list
  - alternatively, you could tag twice, once for legacy clients, eg `latest` and `latest.signed` which points at signature for same content as `latest`

# What do detached signatures look like?



```
{
"signatures" : [
   { "keyid" : "6f4e69a5ff18d92e7315e3ee31c62165ebf25bfa05cad05c0d09d8f412dae401",
     "sig" : "ab56a675d0e47a29b8584830f371da1b058eb17c3d96b6b88defb3c5aa52bc" }
   , ... ]
"signed": {
     "mediaType": "application/vnd.oci.image.index.v1+json",
     "size": 7682,
     "digest": "sha256:5b0bcabd1ed22e9fb1310cf6c2dec7cdef19f0ad69efa1f392e94a4333501270"
   }
}
```

- multiple signatures potentially, only one object. Just the hash is signed.
- no metadata, as it could be modified. Keys known out of band.

# Signatures continued



- Note no expiry data, as it could be tampered with, must be in object that we sign
- We need to specify some additional standard annotations for objects
  - expiry, so we can re-sign regularly, version numbers
  - TUF already defines these for its metadata
- Multiple signatures
  - sign at once multiple times
  - add signatures in same document (changes hash, but old signatures remain valid)
  - point different signatures at document or chain signatures

# Backwards compatibility


KubeCon | CloudNativeCon

North America 2019

- We can create a manifest list wit the images and a pointer to a signature for another manifest list
- clients that do not understand signing must ignore unknown media types
- some more indirection but over time can remove

# Signatures and OCI



- Existing issue in image-spec, Define structure of signatures in OCI #400
- From 2016, nothing happened in three years
- OCI is not very good at defining new things
  - it was created to standardize existing usage
  - similar issues with image encryption
- Has been some movement recently by just doing stuff and not trying to change the specification yet
- Need to move to "*rough consensus and running code*" instead


# *Signing Collections*

# Signing a collection of items



- TUF signs a collection of items
  - this comes from the model of signing a software repository, the current set of packages
- Registries are designed on the distributed system model, consistency models like S3
  - listings of the tag KV store may be eventually consistent
  - no metadata in the KV store, just name to hash association

# TUF solution is a signed targets manifest




```
"targets": {
 "file1.txt": {
  "hashes": {
   "sha256": "65b8c67f51c993d898250f40aa57a317d854900b3a04895464313e48785440da"
  },
  "length": 31
 },
 "dir/file2.txt": {
  "hashes": {
   "sha256": "452ce8308500d83ef44248d8e6062359211992fd837ea9e370e561efb1a4ca99"
  },
  "length": 39
 }
}, ...
```

# Model mismatch



- The TUF manifest is to make sure that we consider a set of packages that go together, eg to prevent *metadata inconsistency attacks*.
- This is not the same issue we have with image repositories?
  - in current usage a repository only contains different versions of the same image; microservices decouple dependencies
  - where we have a set of tightly related images, eg CNAB, they get created as a single content addressed tree of items (same as layers)

# What is wrong here?



- We seem to have a model divergence from the TUF model for package repositories
- This is because we designed the mapping to TUF with registries wrong!
- In Notary each registry repository is a TUF repository, eg `docker.io/library/debian` is a TUF repository with its own root key
- The vast majority of registries do not allow a deeper hierarchy
- *This was the wrong design!*
- Huge proliferation of root keys makes usability really annoying
- Makes delegations largely useless
- Instead, `docker.io/library` should have been the TUF repository

# Repository structure



- We have done some weird things with image repositories
  - for many repositories it is not clear what you should update to, for example the `docker.io/debian` repository has all current versions of Debian in it
  - client has to work out what is the policy
  - so we have to specify that in the config
  - and then there is `latest` the hot garbage of the container world
- Deeper hierarchy would allow `docker.io/library/debian/stretch/debian:9.9`
  - Perhaps allow redirects? Need to work this out...

# Moving root key up



- Having a single root key for everything you own is way simpler
- Currently, to trust all of docker.io/library you need

```
var officialImagesRootKeys = []string{
        "487731f3a9ef2a094d456b37ea7880def4ced3de6f7876ed6258527ac35bcca2",
        "c6faa6d9c53edc26a372c20875f24407584674b834c719aca8c2a781a3fab49d",
        "35ecc5282ec9e9571910bd6fddd7e3b676ef3a480011b4d6e2e7a9ba9d1af1c7",
        "8530a0b0ff0d2c4db50c990faa9ba15d3593292f66312df1cb510a769388ccfa",
        "178c56227f727505bc3181c7fd33af583a6bd5dabdec6c9b3b2faa104386168a",
        "5a740b368b3b71088472e55d17e3a9e7e4bc9ef338bbf3fff3b8475911d803f7",
        "b4147b452cbb17b56eeaca3600244082bb73b2a82f986c8d549eee92baa360a1",
        "26488584f9b21efa90d15e8865c6fa22876e1b3bfc87ec79ba0d7e8aad720a03",
        "6e6beebc7346e7e7320166407b940f4283c8acb0c4940fac8c6f5c0b8dbcf3bb",
        "993134909608be52c5d833a9c6fd4626dc5b78502b4bf74bfc9bc62d290f18a0",
        "93a7bb6c0b1e5c74a141f6e4612e6f2cca1c5e29ce86fd70c61ab2b302ac5275",
        "ee3512f3b31a4a2ac723f31acf1850c1e4a0da124766a3e89c474af6b5c828eb",
        "2f1e6afeea89d56c92ed39ce661f6b29d93b20a1f1b082bc18ff0913b6bb73b5",
        "ac05a3f109ffedf8460a16f287747b04a3db877603a259a31b6ac55cd2b4ff50",
        "99185eb2ab3ccb10f8fa9f354c8c6e7bae8101c4a01dc4021e7a0446e17a50ae",
        "ee3aafa9fec9d3e458dcf56e025e5ebdc99183bd82cbe812d0d55853adc85ff1",
        "00981ccd5e43972e8885e6363a0318d9a02cab371311cd23ffd459c4731b9530",
        "c15164663f80d58bf3d8d5a0cd08b8086cba02065feb00421fc2a5436c563210",
}
```

- There should be a single root key!

# We should only sign current images



- Keeping old images for archival purposes is great
- Often I wish Linux package repos did this (Debian has an archive)
- However we should not automatically re-sign the entire registry repository contents
- We should set policies about expiry, so that the snapshot does not include all the repository contents as it does now
  - for example only sign most recent few images, by number or date
- With the TUF data stored in the registry, we will need a service that will do the snapshot and timestamp signatures, so let us make these more configurable than the current ones.


KubeCon
CloudNativeCon
North America 2019
Cloud Native Supply
Chain Security Model?

# Supply chain model



- We start by committing code changes
- Some images get built and pushed to the registry, usually by tag
- Images get tested
- We update the tag in a production repo with Kubernetes yaml or Helm or whatever (usually)
- This gets deployed

This is clearly a terrible design...

# Problems



- There is no secure link between what we code and the image we run
  - trusting our CI
  - assuming no mistakes were made and the pipeline pushed right image
  - tags are mutable, so something else could have got pushed later
- It would make way more sense if the Kubernetes config contained the hash of the image not the tag in flow as is (but tooling does not make this easy)
- Makes sense to use in-toto to validate flow from source to image, which would also give as a content hash to expect.

# Summary

# Summary



- Notary was an ambitious project to get TUF working in container space
- We made some design mistakes
- These are fixable
- We need community to work together on a very new design
- Have had a lot of interest, no one wants fragmentation
- This talk only really covers some parts of what we need to do
  - more of the technical pieces
  - usability is vital and haven't talked about this today, lots to do

# Q&A



- #notary-v2 in CNCF Slack with links to docs, meetings
- planning some more meetings soon (Seattle, around 12 December)
- github.com/theupdateframework/notary
- @justincormack


KubeCon
CloudNativeCon
North America 2019
The Config/Image
Split of
Responsibilities

# Split responsibilities



- A container image is not a self contained artifact. It does not include how to run it
  - `docker run nginx`
  - `docker run --entrypoint sh --privileged -v /:/root nginx -c rm -rf /root`
- So signing images alone is not enough.
- There are two solutions
  - combine the config into the image that gets signed
  - apply supply chain security to the config as well

# Include config in bundle



- This is the model that CNAB has, and Docker App that is based on it
- The "invocation image" is called to create or update the application
- All the images and config bundled together in one artifact that can be pushed to a registry

# Supply chain for config updates

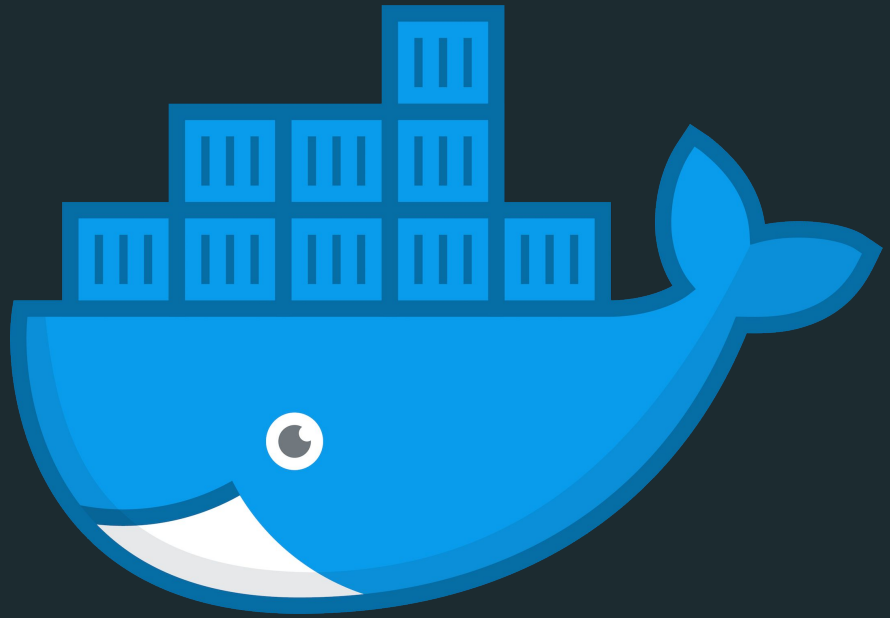


- Gitops to the rescue?
- We could use git signature checks for authenticity
- Use structure of repo to see that commit is an update, ie latest tag or commit
- We still need to validate that images are built from the same commit
  - use in-toto for this?
- Obviously this requires gitops, other forms of delivery would need different tooling
- Requires some key management that git does not provide
  - who is allowed to sign commits when?

# Without gitops



- Slightly lighter weigh as do not need whole git repo
- Need signatures and update guarantees on configs
- Config then becomes an artifact, why not push it to a repo?
- In which case, if it has hashes of images in, it looks much like the CNAB situation, but with a declarative config
- See also https://github.com/deislabs/cnab-spec/issues/285



# THANK YOU