Agenda

- Challenges
- Other Options
- regctl - demo
- regsync - demo
- Using Mirrors - demo
- regbot - demo
- Summary
- Fin

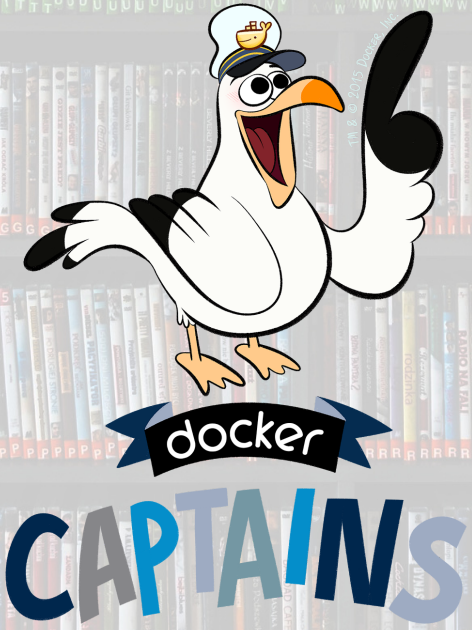
regclient

Tooling to Work with Registries

Brandon Mitchell

Twitter: @sudo_bmitch
GitHub: sudo-bmitch

```
$ whoami
- Solutions Architect @ BoxBoat
- Docker Captain
- Frequenter of StackOverflow
```



Challenges

Retagging Images

```
docker image pull $repo:ci-1234
docker image tag $repo:ci-1234 $repo:stable
docker image push $repo:stable
```

Mirroring Images

```
docker image pull $upstream:latest
docker image tag $upstream:latest $local:latest
docker image push $local:latest
```

Deleting Images

```
docker ??? #@$!@#
```

These Solutions Have Issues

- Require a docker engine = privileged
- Ephemeral nodes pull everything, even if nothing has changed
- Docker only pulls a single platform
- Curl commands need to implement authentication

Search for Better Options

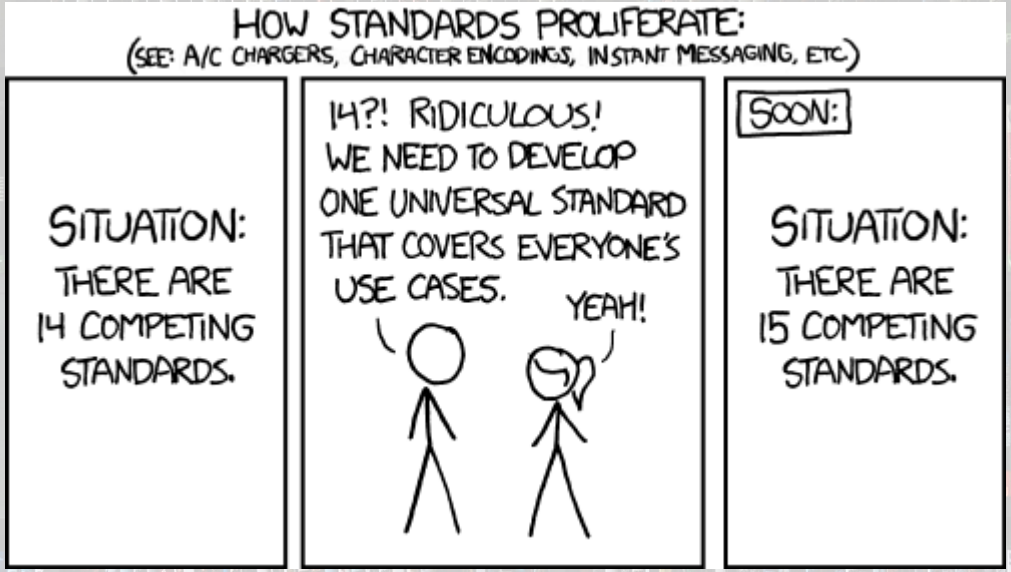
Search for Better Options

- Hostile competitors
- Unmaintained
- Part of a larger project

Search for a Library

- `docker/distribution` appears designed for internal usage
- containerd focuses on pulling images
- Everyone seems to be creating their own partial library

So I Made My Own



https://xkcd.com/927/

Merchandising?

Spaceballs

Developers?

Tools

regctl

regctl

- Provides CLI access to registries using the registry API
- Packaged as a binary you can download or image to run
- Includes an Alpine based image for CI pipelines

regctl Commands

```
$ regctl --help
...
  image       manage images
  layer       manage image layers/blobs
  registry    manage registries
  repo        manage repositories
  tag         manage tags
```

regctl Commands

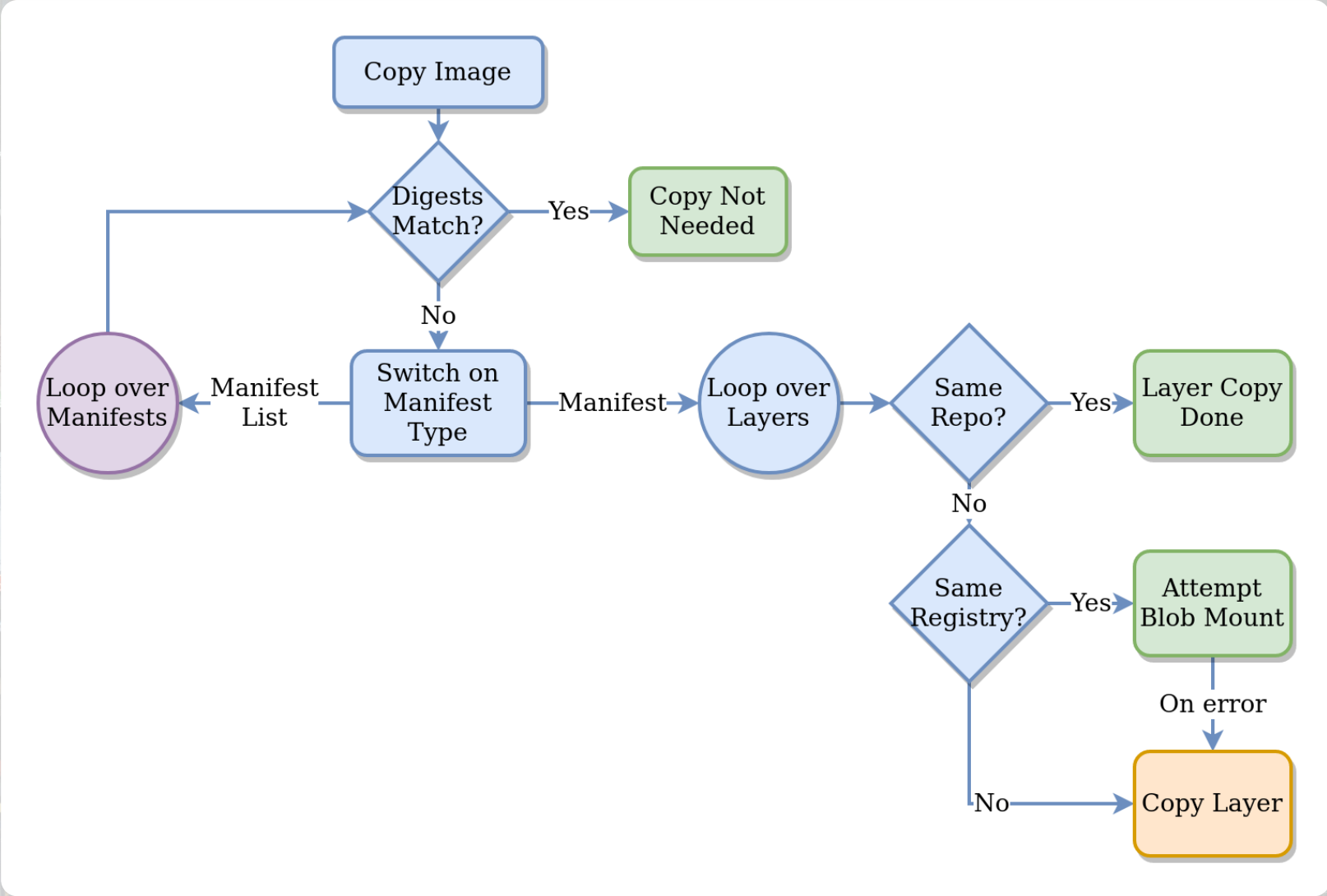
```
$ regctl registry --help
...
  config      show registry config
  login       login to a registry
  logout      logout of a registry
  set         set options on a registry
```

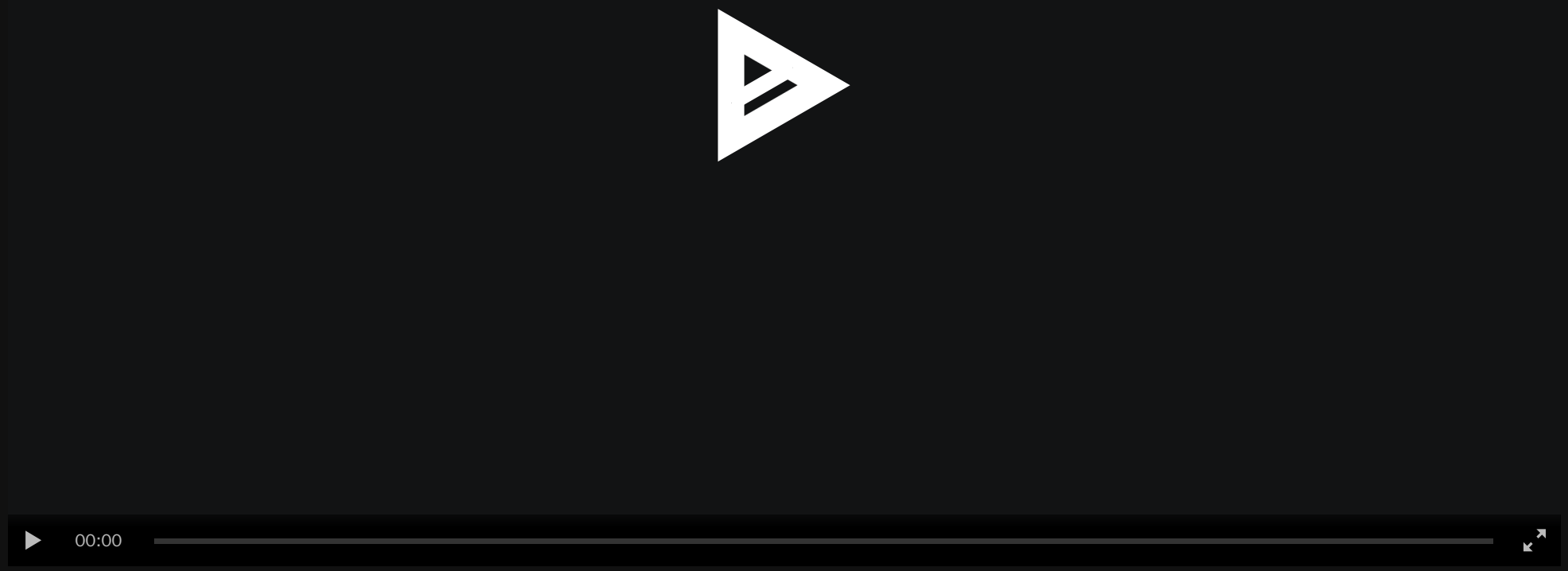
regctl Commands

```
$ regctl repo --help
...
  ls          list repositories in a registry

$ regctl tag --help
...
  delete      delete a tag in a repo
  ls          list tags in a repo

$ regctl layer --help
...
  pull        download a layer/blob
```

regctl Commands

```
$ regctl image --help
...
  copy         copy or retag image
  delete       delete image
  digest       show digest for pinning
  export       export image
  inspect      inspect image
  manifest     show manifest or manifest list
  ratelimit    show the current rate limit
```


Copy Image
Digests Match?
Yes
Copy Not Needed
No
Loop over Manifests
Manifest List
Switch on Manifest Type
Manifest
Loop over Layers
Same Repo?
Yes
Layer Copy Done
No
Same Registry?
Yes
Attempt Blob Mount
On error
No
Copy Layer


00:00

regsync

regsync

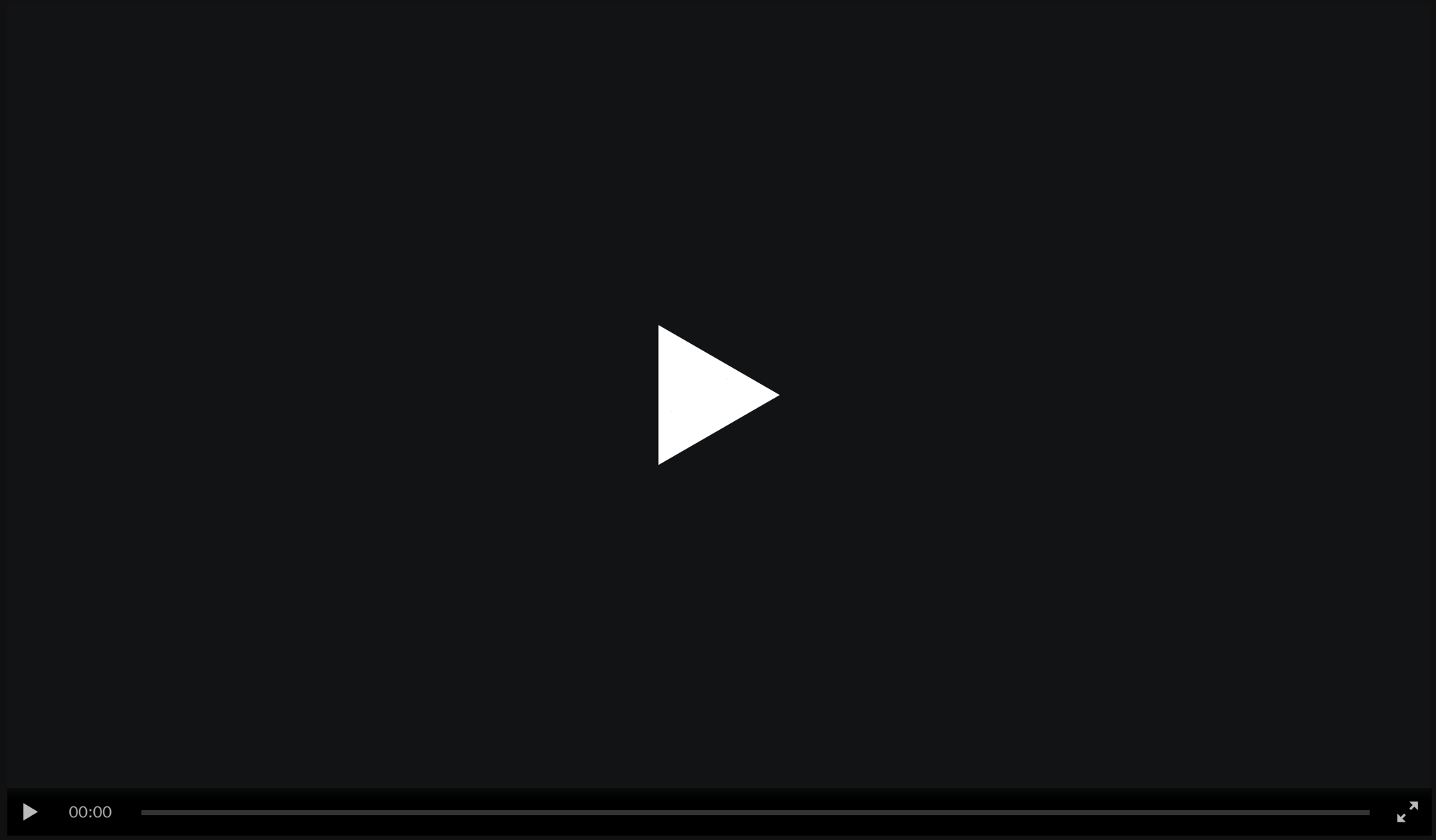
- Why create a mirror?
- Upstream outages happen
- Upstream images can change unexpectedly
- Bandwidth cost time and money
- Hub rate limits

regsync

- Automates the copying of images
- Runs as a standalone binary or in a container
- Uses a yaml configuration file with templating support
- Run sync steps immediately or on a schedule
- Can copy one or more tags
- Can delay for rate limits
- Can backup old image before overwriting
- Structured logging integrates with aggregators

regsync Config

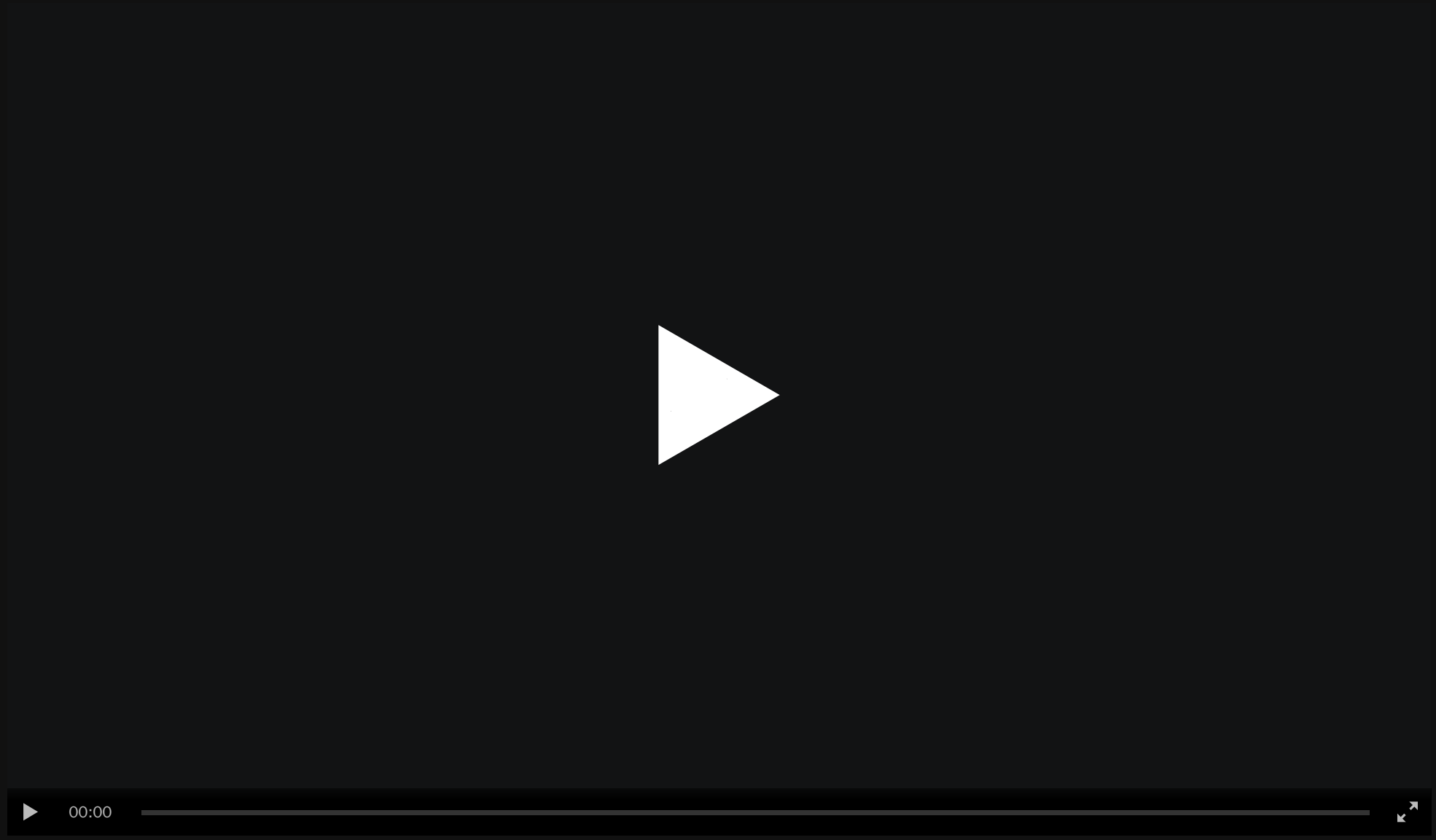
```
creds:
  - registry: docker.io
    user: "{{env \"HUB_USER\"}}"
    pass: "{{file \"/var/run/secrets/hub_token\"}}"
defaults:
  ratelimit: { min: 100, retry: 15m }
  interval: 60m
  backup: "bkup-{{.Ref.Tag}}"
sync:
  - source: alpine
    target: registry:5000/library/alpine
    type: repository
    tags:
      allow:
      - "3"
      - "3.\\d+"
```


00:00

Using a Mirror

```
ARG registry=docker.io
ARG tag=3
FROM ${registry}/library/alpine:${tag}
```

```
docker image build \
  --build-arg registry=localhost:5000 \
  -t localhost:5000/example/app:${ver} .
```


00:00


regbot

regbot

- How do you define a configuration file for registry image pruning?
- Prune all CI builds before a certain date
- Prune everything matching an expression, but keeping the most recent 10 images
- Check a label to see if the image is from an old sprint
- Delete local images when upstream mirror deletes the original

regbot

- Use the much of the config syntax from regsync
- Take the flexibility of scripting regctl commands
- Add Lua for a scripting language

Lua

- Go native, not even CGo
- Variables
- Tables(arrays, key/value)
- Conditionals
- Loops
- Math
- String parsing and matching
- Date and time

- Extendable with our own API calls
- Repository list
- Tag list
- Tag delete
- Rate limit
- Image manifest
- Image config
- Image copy
- Image delete

regbot Config

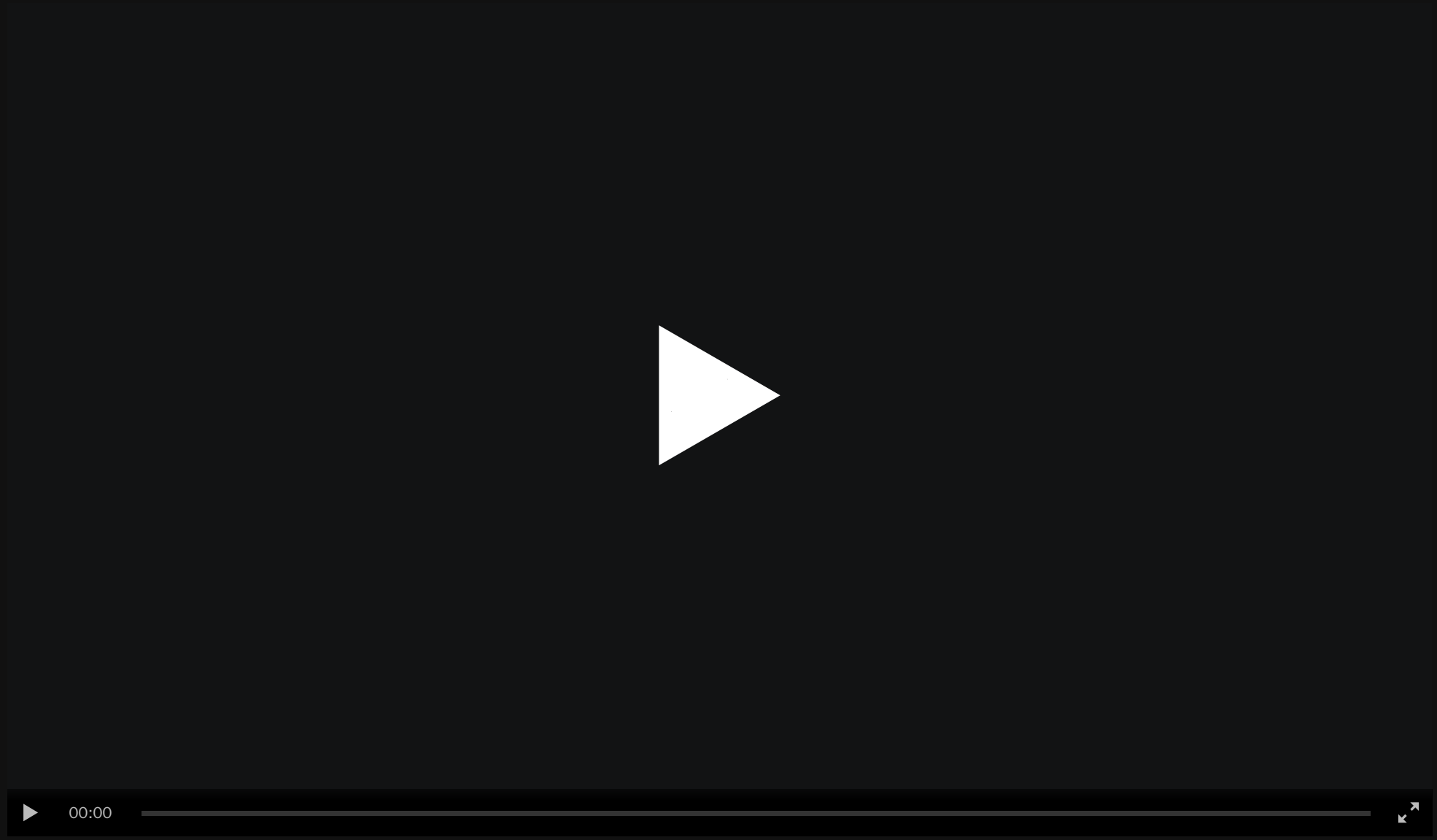
```
creds:
  - registry: docker.io
    user: "{{env \"HUB_USER\"}}"
    pass: "{{file \"/var/run/secrets/hub_token\"}}"
defaults:
  interval: 60m
scripts:
  - name: mirror minor versions
    timeout: 59m
    script: |
      imageList = {"library/alpine", "library/debian"}
      localReg = "registry:5000"
      tagExp = "^%d+%.%d+$"
      minRateLimit = 100
      maxKeep = 3
      -- ...
```

regbot Config

```
    script: |
      for k, imageName in ipairs(imageList) do
        tags = tag.ls(upstreamRef)
        for k, t in pairs(tags) do
          if string.match(t, tagExp) then
            table.insert(matchTags, t)
          end
        end
        table.sort(matchTags, orderSemVer)
        for k, t in ipairs(matchTags) do
          upstreamRef:tag(t)
          localRef:tag(t)
          if not image.ratelimitWait(upstreamRef, minRateLimit) then
            error "Timed out waiting on rate limit"
          end
          image.copy(upstreamRef, localRef)
        end
      end
```

regbot Config

```
- name: delete old builds
  script: |
    -- ... variables set above ...
    timeRef = os.time() - (86400*maxDays)
    cutoff = os.date(dateFmt, timeRef)
    ref = reference.new(imageName)
    tags = tag.ls(ref)
    for k, t in pairs(tags) do
      if string.match(t, tagExp) then
        ref:tag(t)
        ic = image.config(ref)
        if ic.Config.Labels[imageLabel] < cutoff then
          tag.delete(ref)
        end
      end
    end
```


00:00

Wrapping Up

Future Plans

- Improve handling of other registries
- Handle more media types
- Add OCI Distribution support
- Helm charts, CNAB, Notary v2
- Import/export workflows to support offline registries
- Create a pull through cache

Try It Out

- github.com/regclient/regclient
- hub.docker.com/u/regclient
- Quick start guides for regsync and regbot

Thank You

github.com/sudo-bmitch/presentations

Brandon Mitchell
Twitter: @sudo_bmitch
GitHub: sudo-bmitch