Agenda

Maintaining a Local Registry Mirror

Brandon Mitchell
Twitter: @sudo_bmitch
GitHub: sudo-bmitch

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

Why

Why

- Outages happen
- Bandwidth is expensive and slow
- Rate limits

Why

- Outages happen
- Bandwidth is expensive and slow
- Rate limits
- Pull through cache is unpredictable
- Ability to revert an upstream change
- Control over approved images

Original Solution

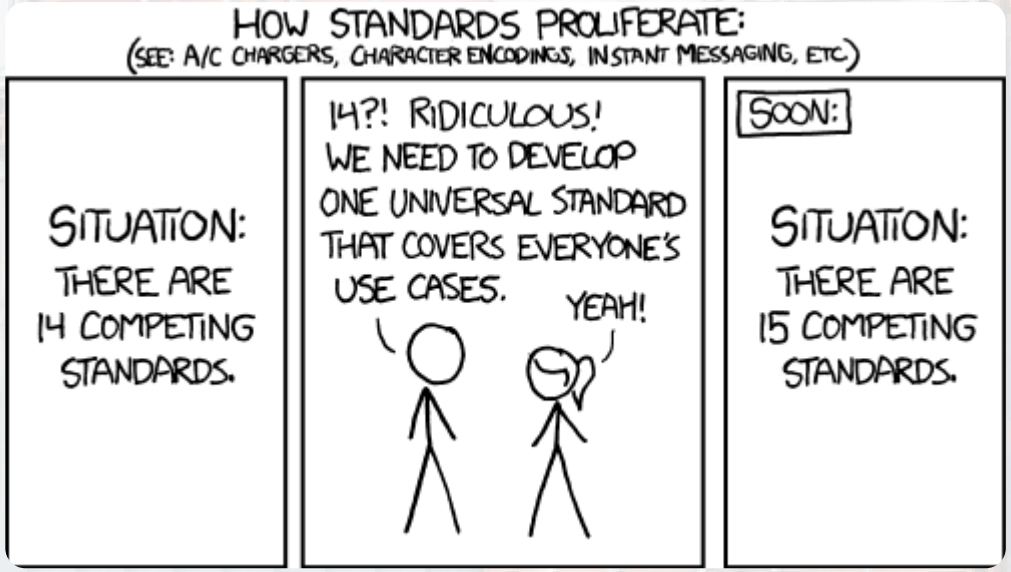
Original Solution

```
docker image pull alpine:latest
docker image tag alpine:latest localhost:5000/alpine:latest
docker image push localhost:5000/alpine:latest
```

Original Solution Issues

- Missing the `library`
  - `alpine:latest` is really `docker.io/library/alpine:latest`
  - So we should copy to `localhost:5000/library/alpine:latest`
- `docker pull` dereferences an image to a single platform
- Every layer is pulled
- Requires a docker engine

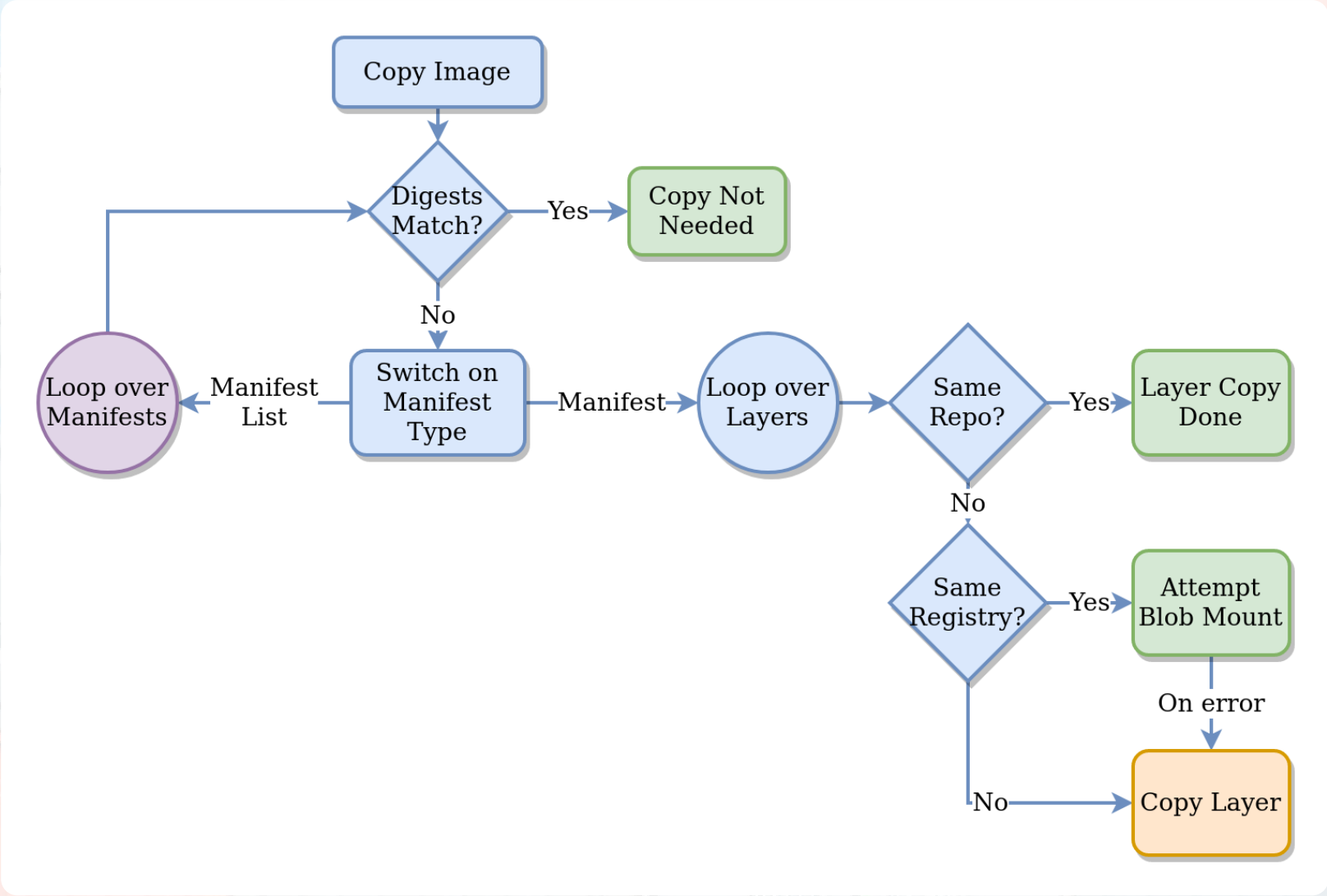
New Solution


HOW STANDARDS PROLIFERATE:
(SEE: A/C CHARGERS, CHARACTER ENCODINGS, INSTANT MESSAGING, ETC)
SITUATION: THERE ARE 14 COMPETING STANDARDS.
14?! RIDICULOUS! WE NEED TO DEVELOP ONE UNIVERSAL STANDARD THAT COVERS EVERYONE'S USE CASES.
YEAH!
SOON:
SITUATION: THERE ARE 15 COMPETING STANDARDS.

xkcd.com/927/

Registry Tooling

- skopeo
- crane
- regclient


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

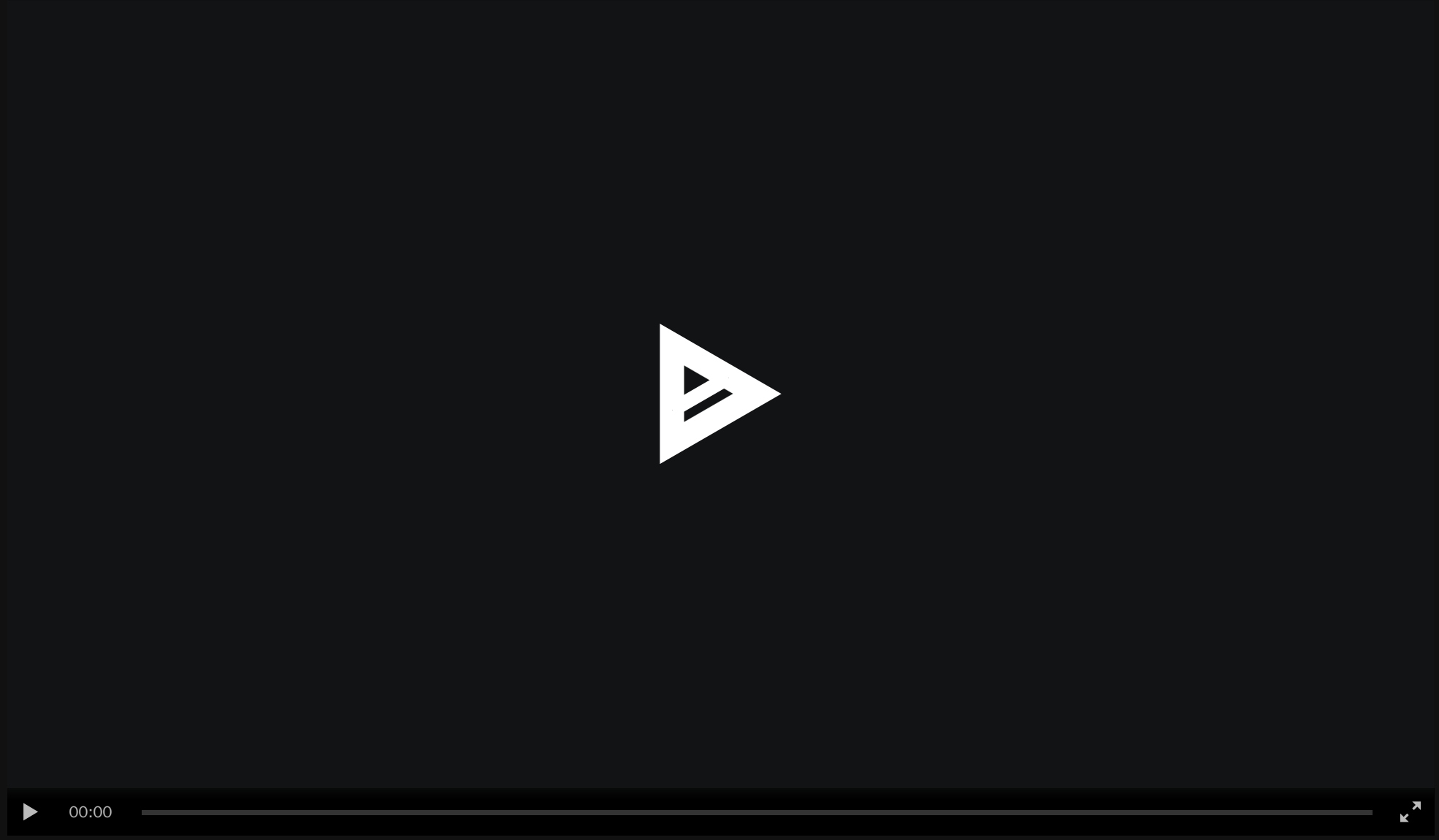
Workshop

https://github.com/sudo-bmitch/presentations/tree/main/reg-mirror/workshop/

Step 1: Deploy a Registry

Step 1: Deploy a Registry

```
docker container run -d --restart=unless-stopped --name registry \
  -e "REGISTRY_STORAGE_FILESYSTEM_ROOTDIRECTORY=/var/lib/registry" \
  -e "REGISTRY_STORAGE_DELETE_ENABLED=true" \
  -e "REGISTRY_VALIDATION_DISABLED=true" \
  -v "registry-data:/var/lib/registry" \
  -p "127.0.0.1:5000:5000" \
  registry:2
```


00:00

Step 2: Configure a Policy

Step 2: Configure a Policy

```
creds:
  - registry: localhost:5000
    tls: disabled
  - registry: docker.io
    user: "{{env \"HUB_USER\"}}"
    pass: "{{file \"/home/appuser/.docker/hub_token\"}}"
defaults:
  ratelimit:
    min: 60
    retry: 15m
  parallel: 2
  interval: 60m
  backup: "{{ $t := time.Now }}{{
            printf \"bkup-%s-%d%d%d\" .Ref.Tag $t.Year $t.Month $t.Day}}"
sync:
  ...
```

Step 2: Configure a Policy (cont.)

```
...
sync:
  - source: busybox:latest
    target: localhost:5000/library/busybox:latest
    type: image
  - source: alpine
    target: localhost:5000/library/alpine
    type: repository
    tags:
      allow:
      - "latest"
      - "3"
      - "3.13"
      - "3.14"
```

Step 3: Test the Configuration

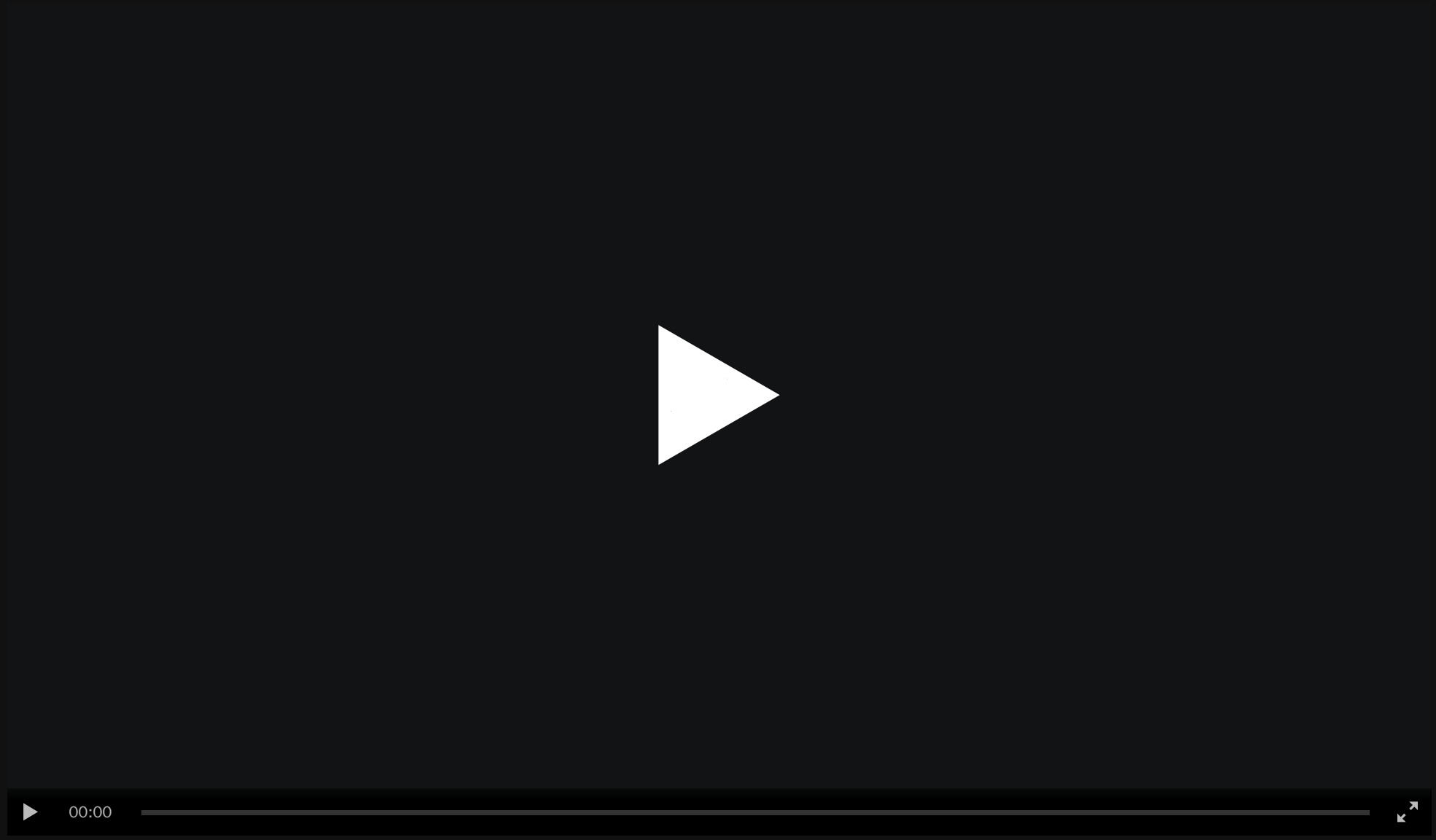
Step 3: Test the Configuration

```
export HUB_USER=your_username

mkdir -p ${HOME}/.docker
echo "your_hub_password" >${HOME}/.docker/hub_token

docker container run -it --rm --net host \
  -v "$(pwd)/regsync.yml:/home/appuser/regsync.yml:ro" \
  -v "${HOME}/.docker/hub_token:/home/appuser/.docker/hub_token:ro" \
  -e "HUB_USER" \
  regclient/regsync:latest -c /home/appuser/regsync.yml once
```

Step 4: Automate

Step 4: Automate

```
docker container run -d --net host \
  --restart=unless-stopped --name regsync \
  -v "$(pwd)/regsync.yml:/home/appuser/regsync.yml:ro" \
  -v "$(pwd)/hub_token:/var/run/secrets/hub_token:ro" \
  -e "HUB_USER" \
  regclient/regsync:latest -c /home/appuser/regsync.yml server
```

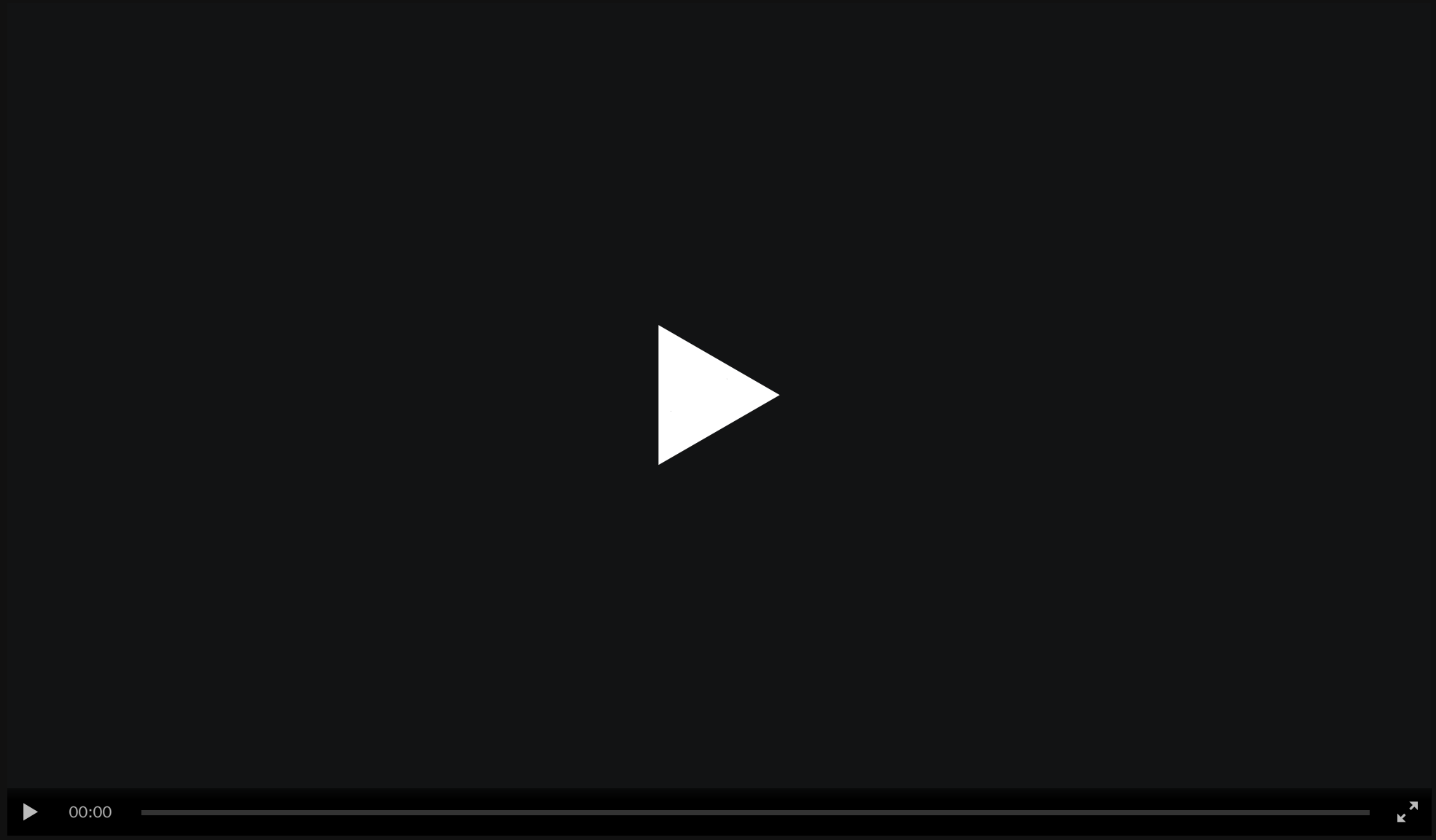

00:00

Using the Mirror

Using the Mirror

```
ARG REGISTRY=docker.io
FROM ${REGISTRY}/library/alpine:3.14

RUN apk add curl
ENTRYPOINT [ "curl" ]
```

```
docker image build --build-arg REGISTRY=localhost:5000 \
  -t localhost:5000/${HUB_USER}/curl .

docker image push localhost:5000/${HUB_USER}/curl

docker container run -it --rm \
  localhost:5000/${HUB_USER}/curl https://google.com/
```


00:00

Garbage Collection

Garbage Collection - regctl

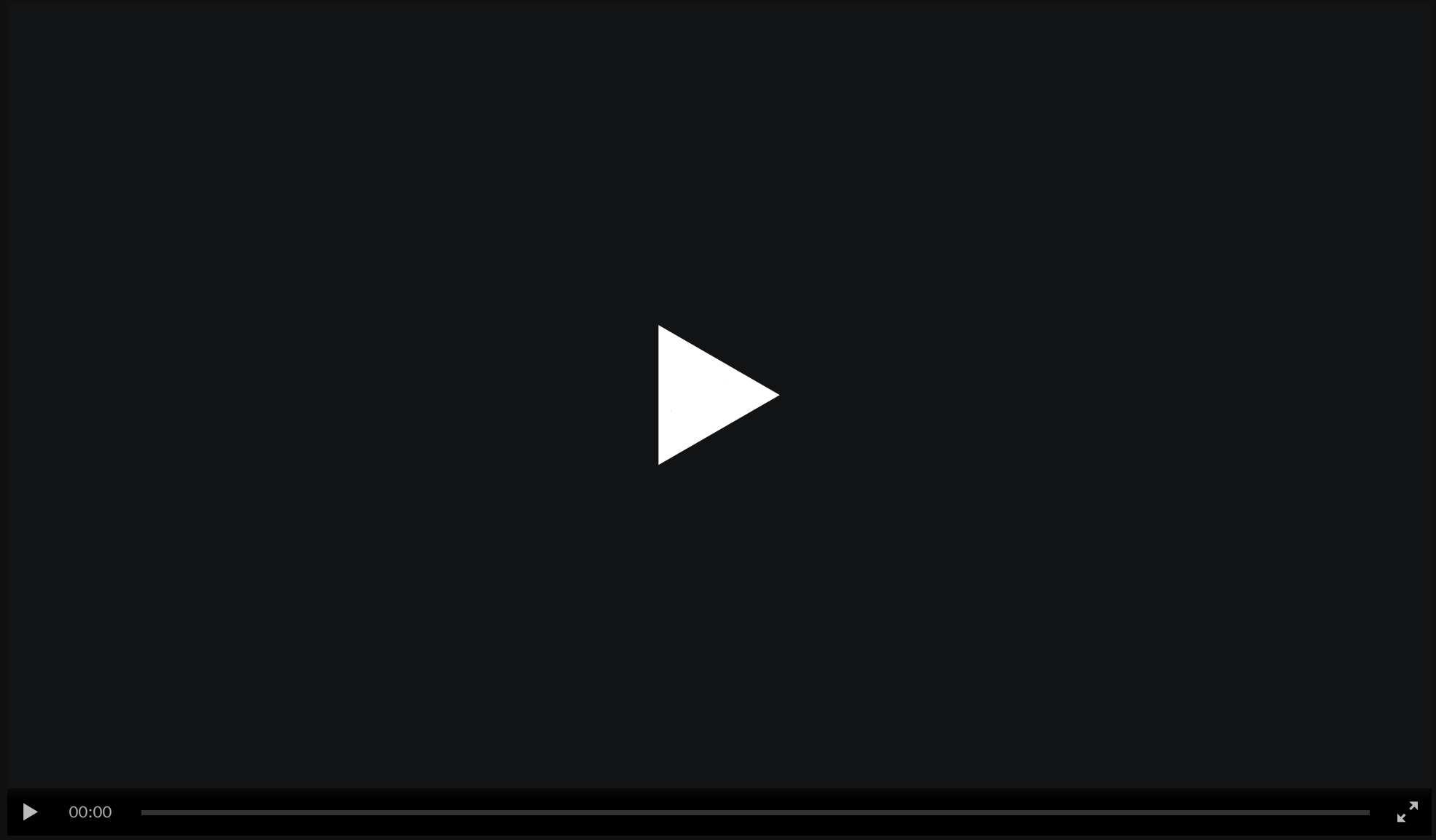
```
alias regctl='docker container run --rm --net host \
  -u "$(id -u):$(id -g)" -e HOME -v "$HOME:$HOME" -w "$(pwd)" \
  regclient/regctl:latest'

regctl registry set localhost:5000 --tls disabled
```

Garbage Collection - Simulate Old Backups

```
for day in 20210831 20210901 20210902 20210903 20210904; do
  regctl image copy localhost:5000/library/alpine:3 \
                    localhost:5000/library/alpine:bkup-3-${day}
  regctl image copy localhost:5000/library/alpine:latest \
                    localhost:5000/library/alpine:bkup-latest-${day}
  regctl image copy localhost:5000/library/busybox:latest \
                    localhost:5000/library/busybox:bkup-latest-${day}
done
```

Garbage Collection - List Tags

```
regctl tag ls localhost:5000/library/alpine
regctl tag ls localhost:5000/library/busybox
```


00:00

Garbage Collection - regbot.yml

```
version: 1
creds:
  - registry: localhost:5000
    tls: disabled
defaults:
  parallel: 1
  interval: 60m
  timeout: 600s
scripts:
...
```

Garbage Collection - regbot.yml (cont)

```
...
scripts:
  - name: delete old backups
    script: |
      reg = "localhost:5000"
      backupExpr = "^bkup%-(.+)%-(%d+)$"
      backupLimit = 3
      -- list all repos, could replace this with a fixed list
      repos = repo.ls(reg)
      table.sort(repos)
      -- loop over each repo
      for k, r in pairs(repos) do
        -- list all tags in the repo
        tags = tag.ls(reg .. "/" .. r)
        table.sort(tags)
...
```

Garbage Collection - regbot.yml (cont)

```
...
        backupTags = {}
        for k, t in pairs(tags) do
          -- search for tags matching backup expression (e.g. bkup-latest-20210102)
          if string.match(t, backupExpr) then
            tOrig, tVer = string.match(t, backupExpr)
            -- backupTags is a nested table, e.g. backupTags[latest]={.. backup tags for latest ..}
            if not backupTags[tOrig] then
              backupTags[tOrig] = {}
            end
            table.insert(backupTags[tOrig], t)
          end
        end
...
```

Garbage Collection - regbot.yml (cont)

```
...
        for tOrig, tVers in pairs(backupTags) do
          -- if any original tag has too many backups
          if #tVers > backupLimit then
            -- delete the first n tags to get back to the limit, sorted to delete oldest backups fir
            table.sort(tVers)
            delVers = {unpack(tVers, 1, #tVers - backupLimit)}
            for k, t in pairs(delVers) do
              -- log("Deleting old backup: " .. reg .. "/" .. r .. ":" .. t)
              tag.delete(reg .. "/" .. r .. ":" .. t)
            end
          end
        end
      end
```

Garbage Collection - regbot dry-run

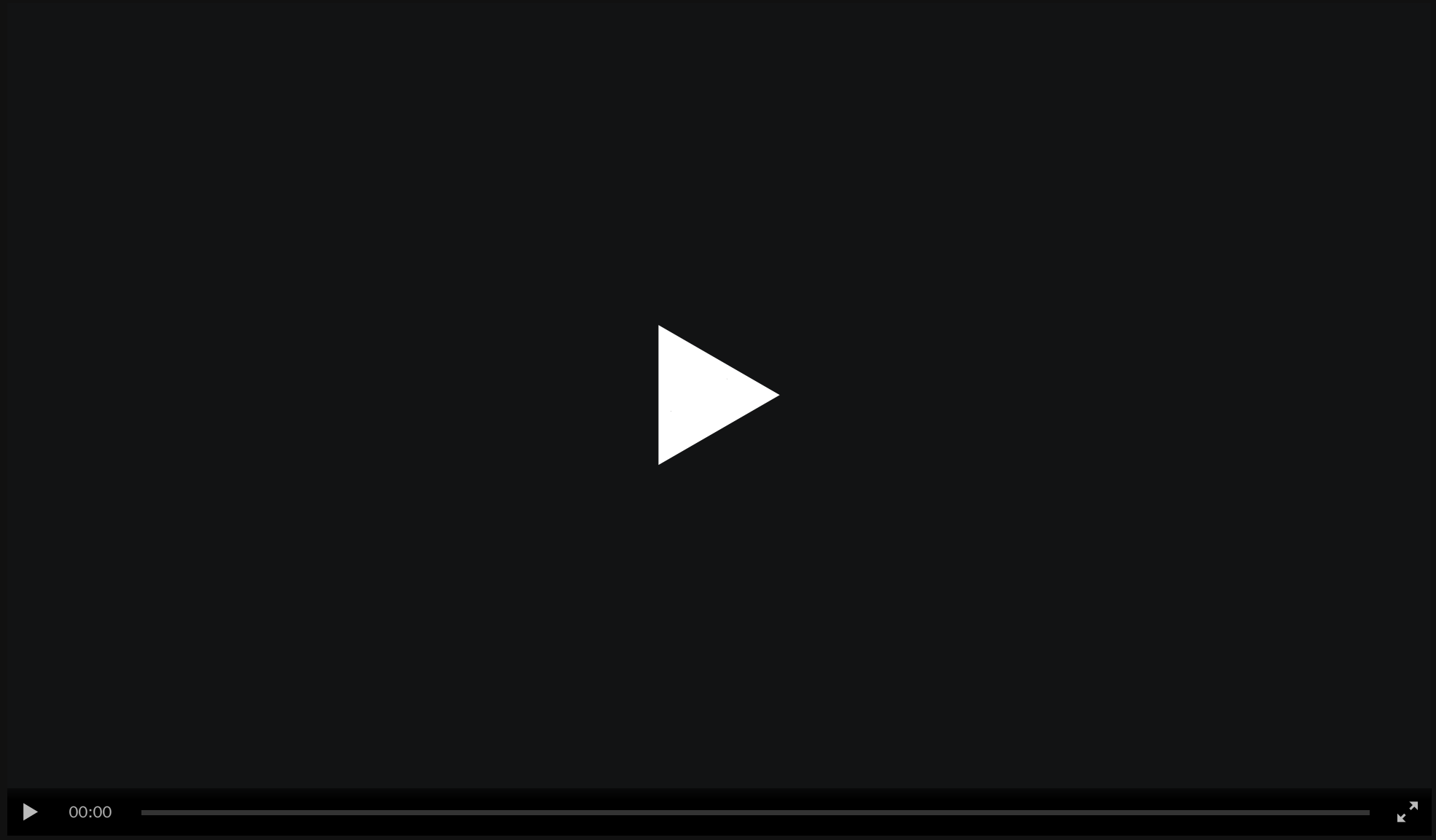
```
docker container run -it --rm --net host \
  -v "$(pwd)/regbot.yml:/home/appuser/regbot.yml" \
  regclient/regbot:latest -c /home/appuser/regbot.yml once --dry-run
```

Garbage Collection - regbot once

```
docker container run -it --rm --net host \
  -v "$(pwd)/regbot.yml:/home/appuser/regbot.yml" \
  regclient/regbot:latest -c /home/appuser/regbot.yml once
```

Garbage Collection - Automate

```
docker container run -d --restart=unless-stopped --name regbot --net host \
  -v "$(pwd)/regbot.yml:/home/appuser/regbot.yml" \
  regclient/regbot:latest -c /home/appuser/regbot.yml server
```

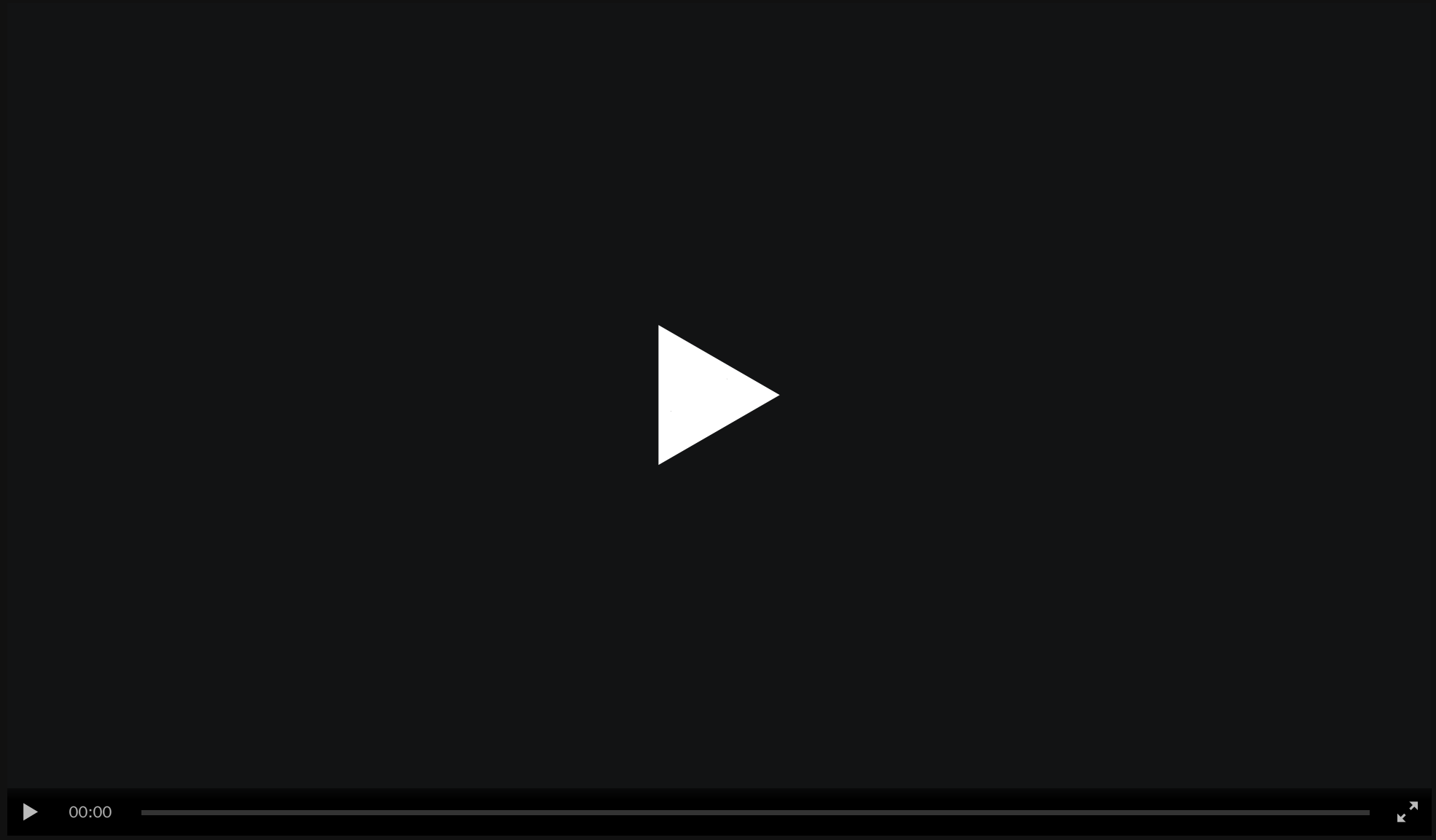

00:00

Garbage Collection - Registry GC

```
docker exec registry /bin/registry garbage-collect \
  /etc/docker/registry/config.yml --delete-untagged
```

Garbage Collection - Verify

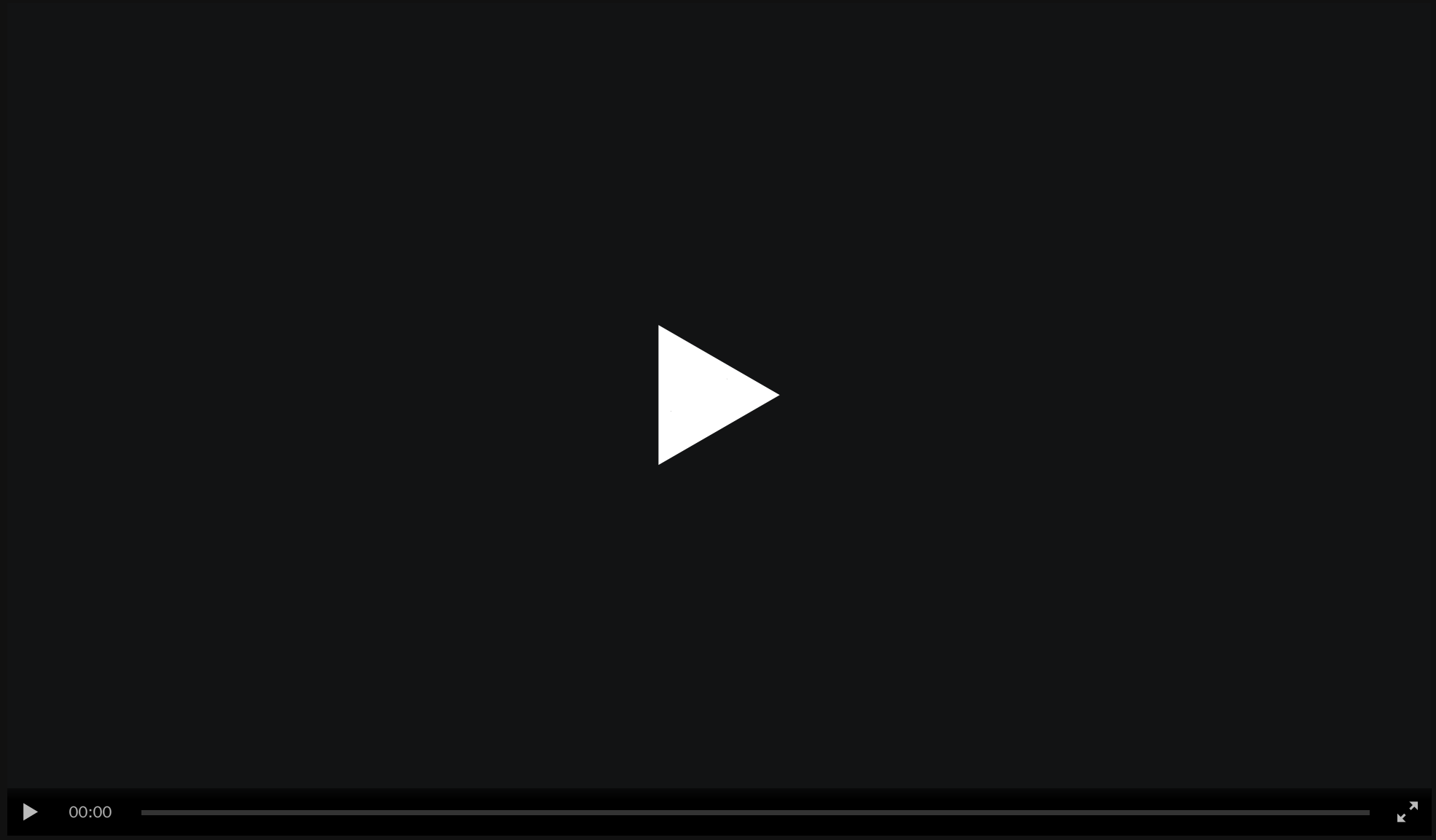
```
regctl tag ls localhost:5000/library/alpine
regctl tag ls localhost:5000/library/busybox
```


00:00

Cleanup

Cleanup

```
docker container stop registry regsync regbot
docker container rm registry regsync regbot
docker volume rm registry-data
```


00:00

Wrapping Up

Wrapping Up

- Started a local registry
- Copied images matching upstream names
- Build using this registry via ARG
- Push images to the local registry under our namespace
- Delete old unneeded images and garbage collect the registry

Thank You

github.com/regclient/regclient
github.com/sudo-bmitch/presentations

Brandon Mitchell
Twitter: @sudo_bmitch
GitHub: sudo-bmitch