Agenda

- Caching
- Caching Demos
- Caching Limitations
- Caching Other Registries
- Caching Gitlab Demo
- Mirroring
- Manual Mirroring
- Summary

How to Use Mirroring and Caching to Optimize Your Image Registry

Brandon Mitchell

Twitter: @sudo_bmitch
GitHub: sudo-bmitch

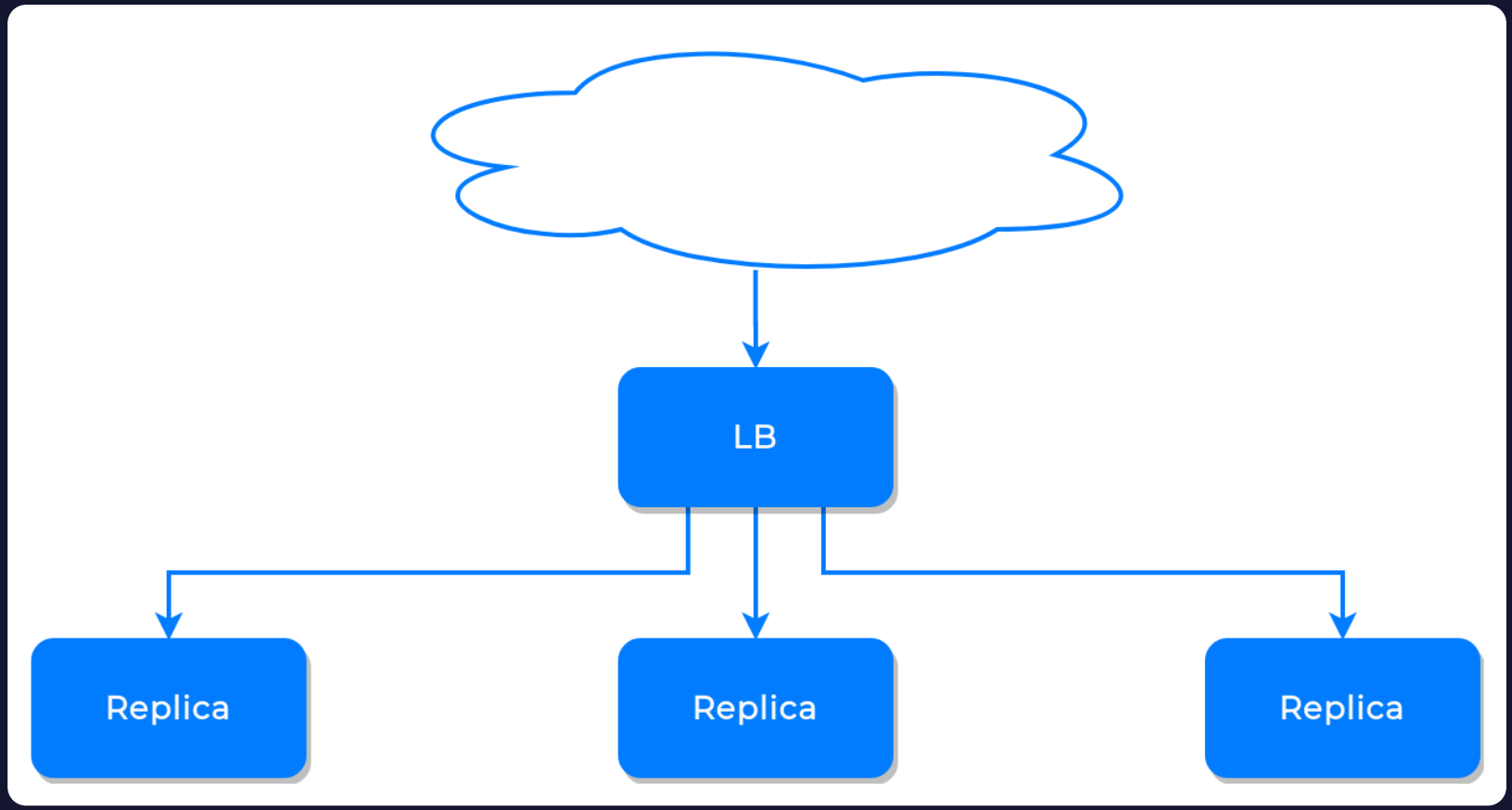
Ephemeral Build Server?

Cluster Pulling Remote Images?

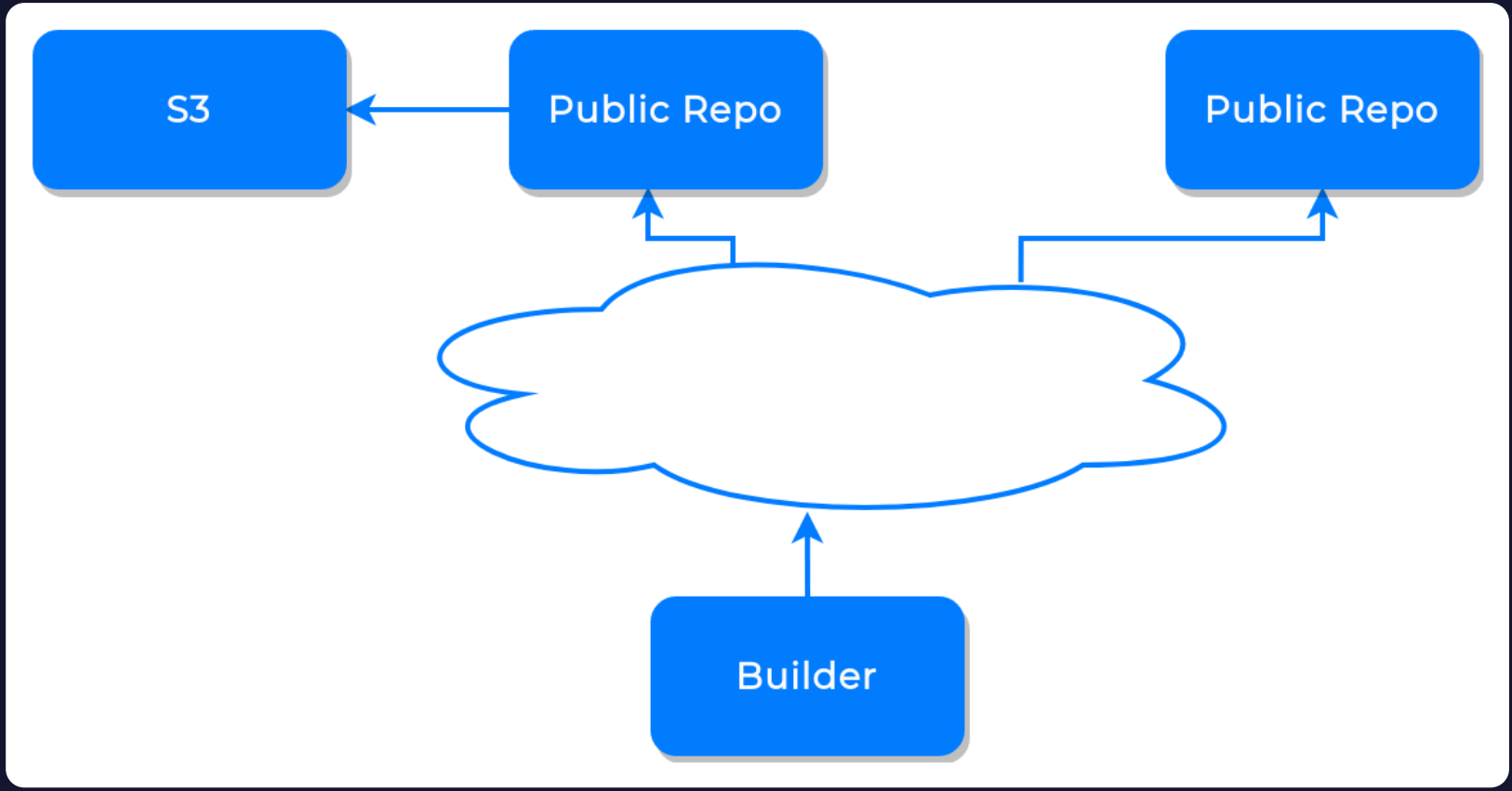
Worry About Upstream Image Changes?

Build and Deploy Infrastructure Tolerant of Upstream Outages?

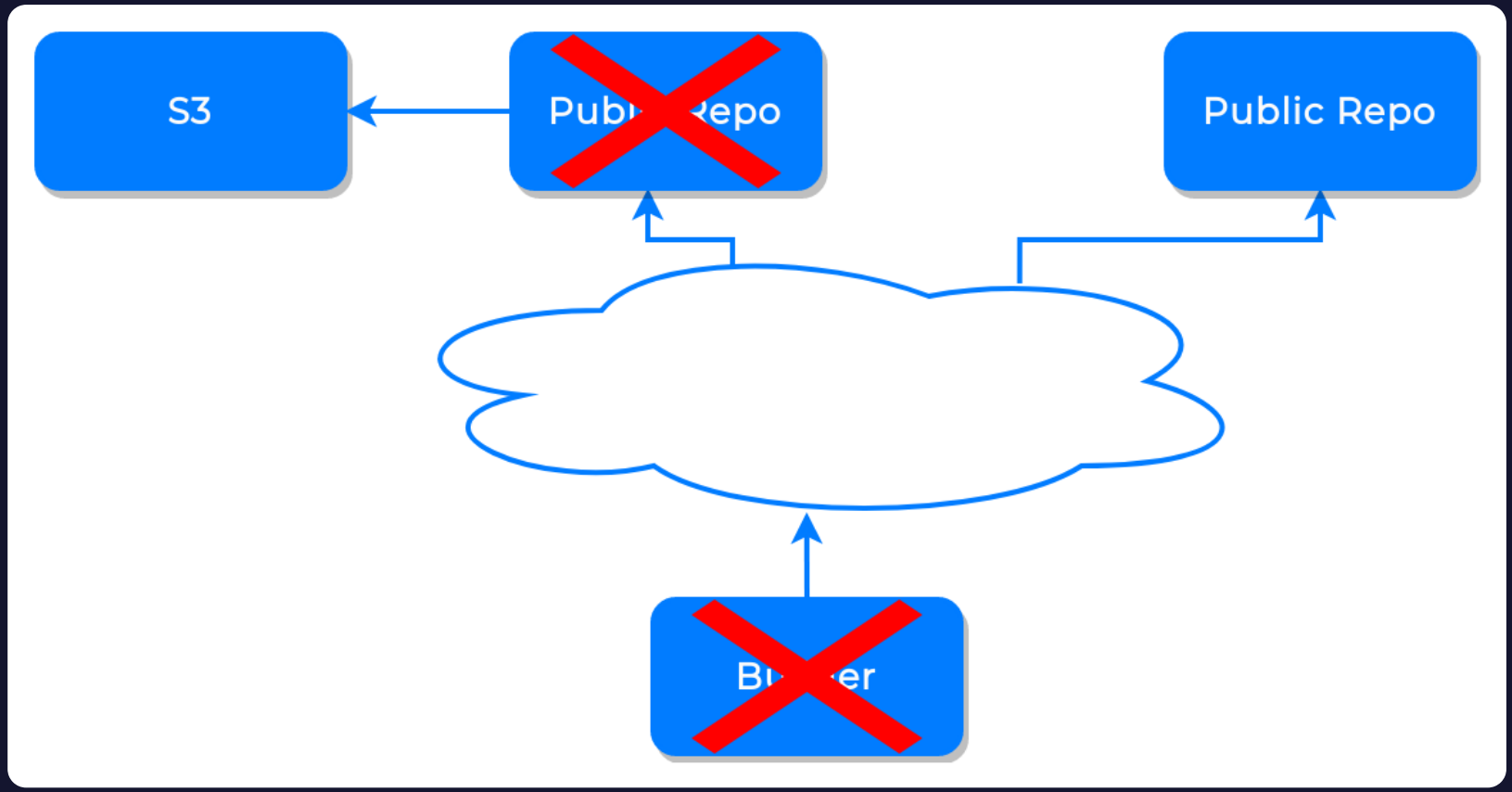
Production Resilience



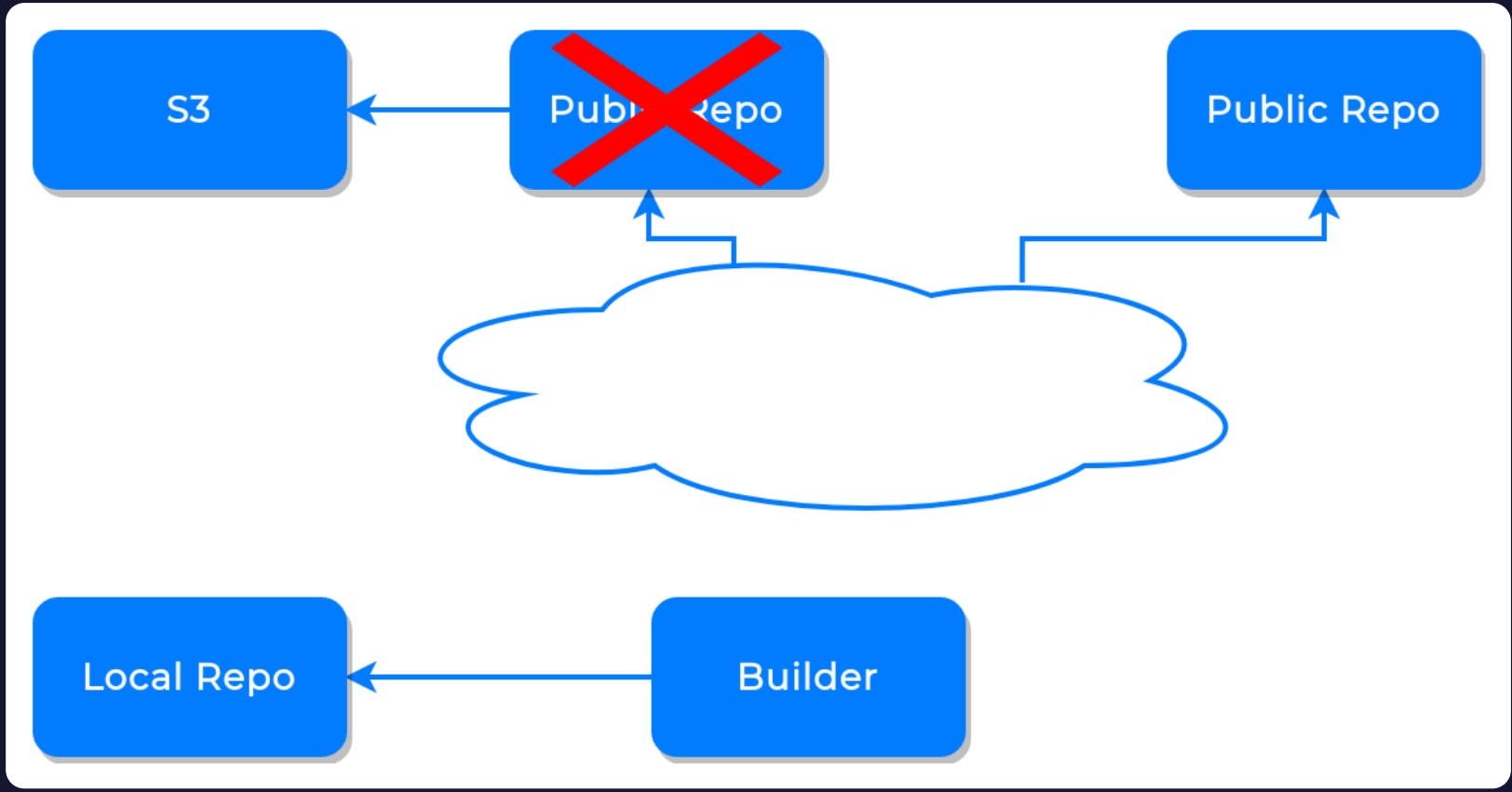
Build Infrastructure



Build Outage



Build Resilience

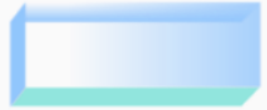


Faster Builds and Less Bandwidth

Registry Mirroring and Caching

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

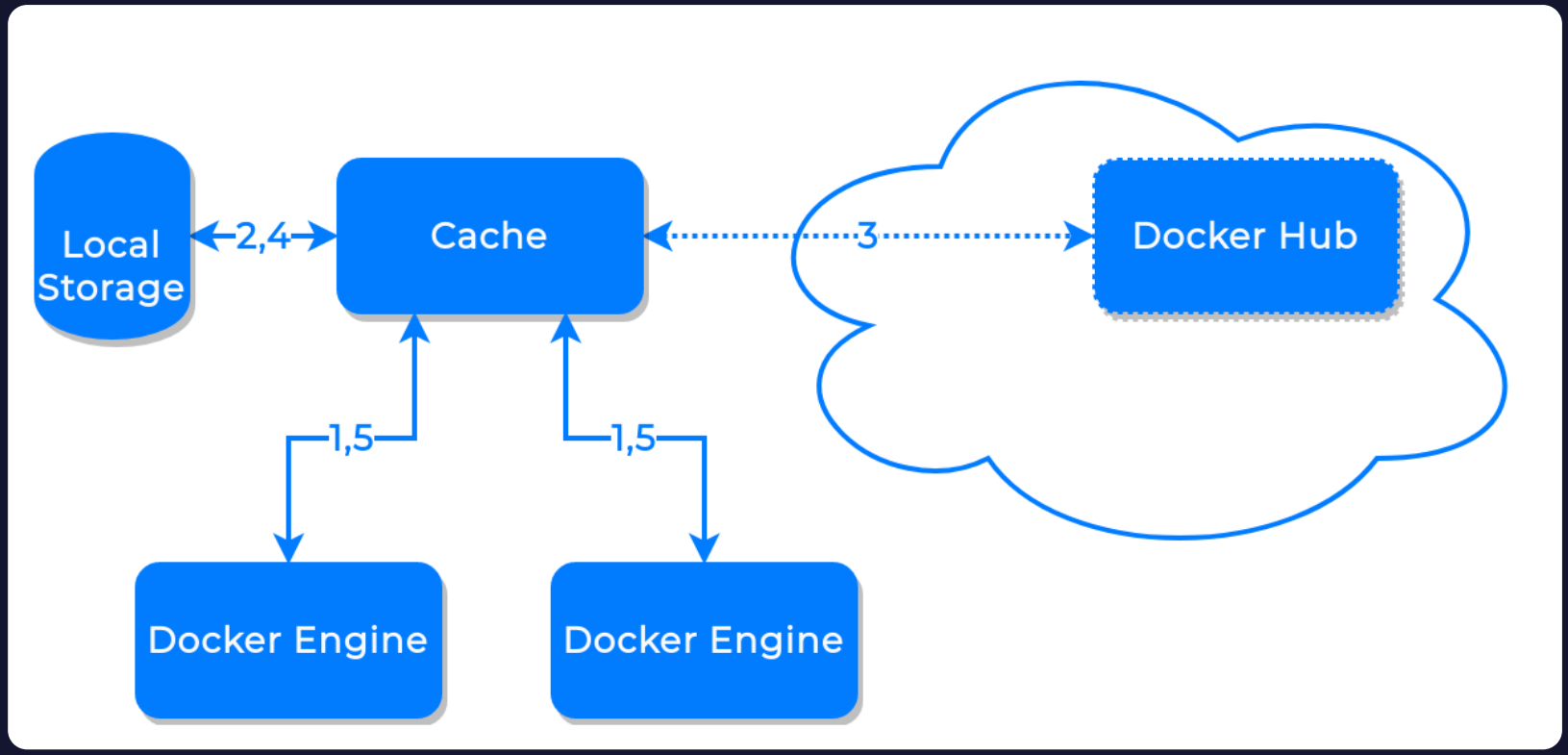
Caching



Caching is the Easy Button

https://commons.wikimedia.org/wiki/File:Easy_button.JPG

Cache Architecture



Cache Implementation

Either the dockerd CLI:

```
dockerd --registry-mirror <cache-url>
```

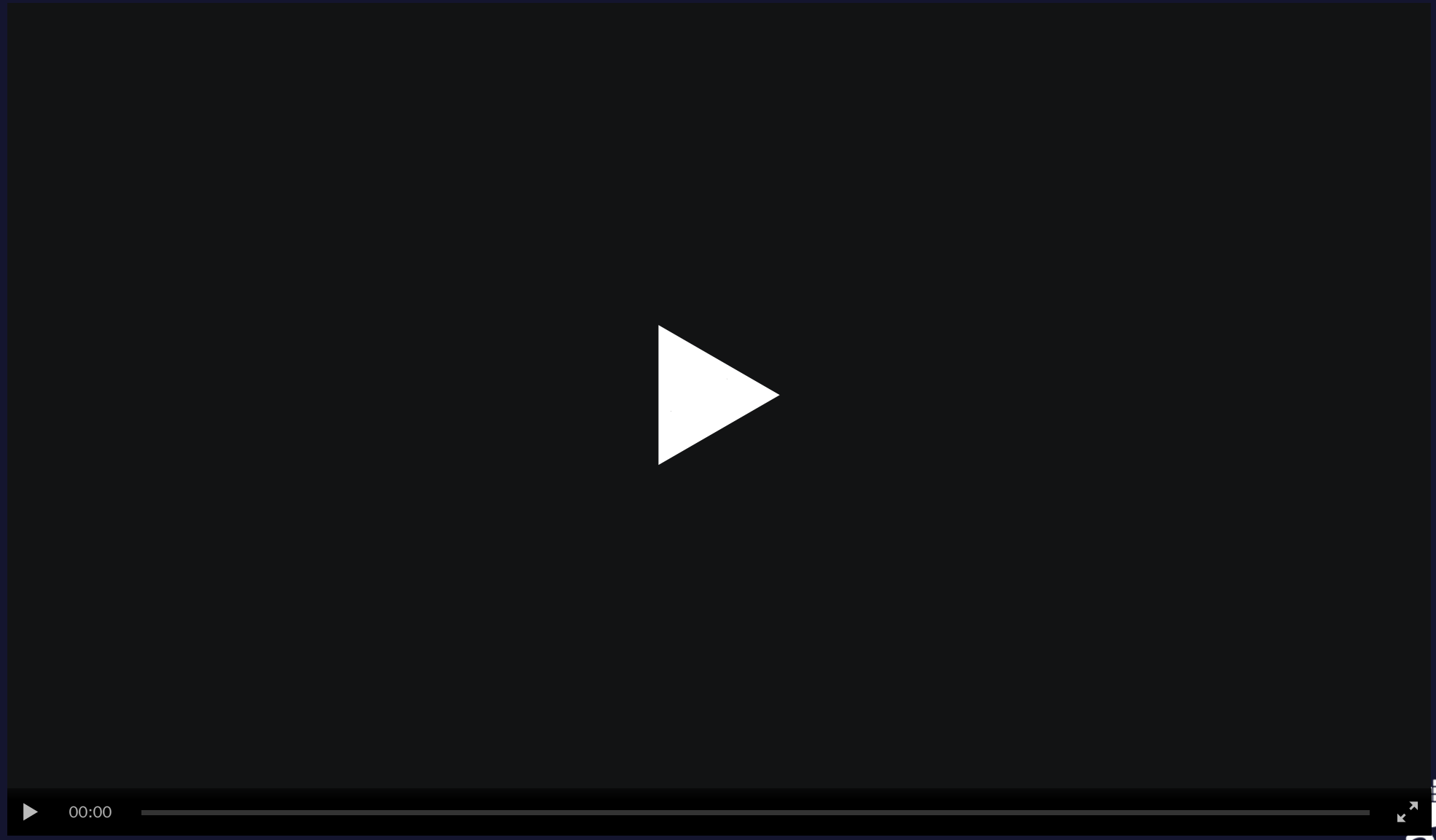
Or /etc/docker/daemon.json

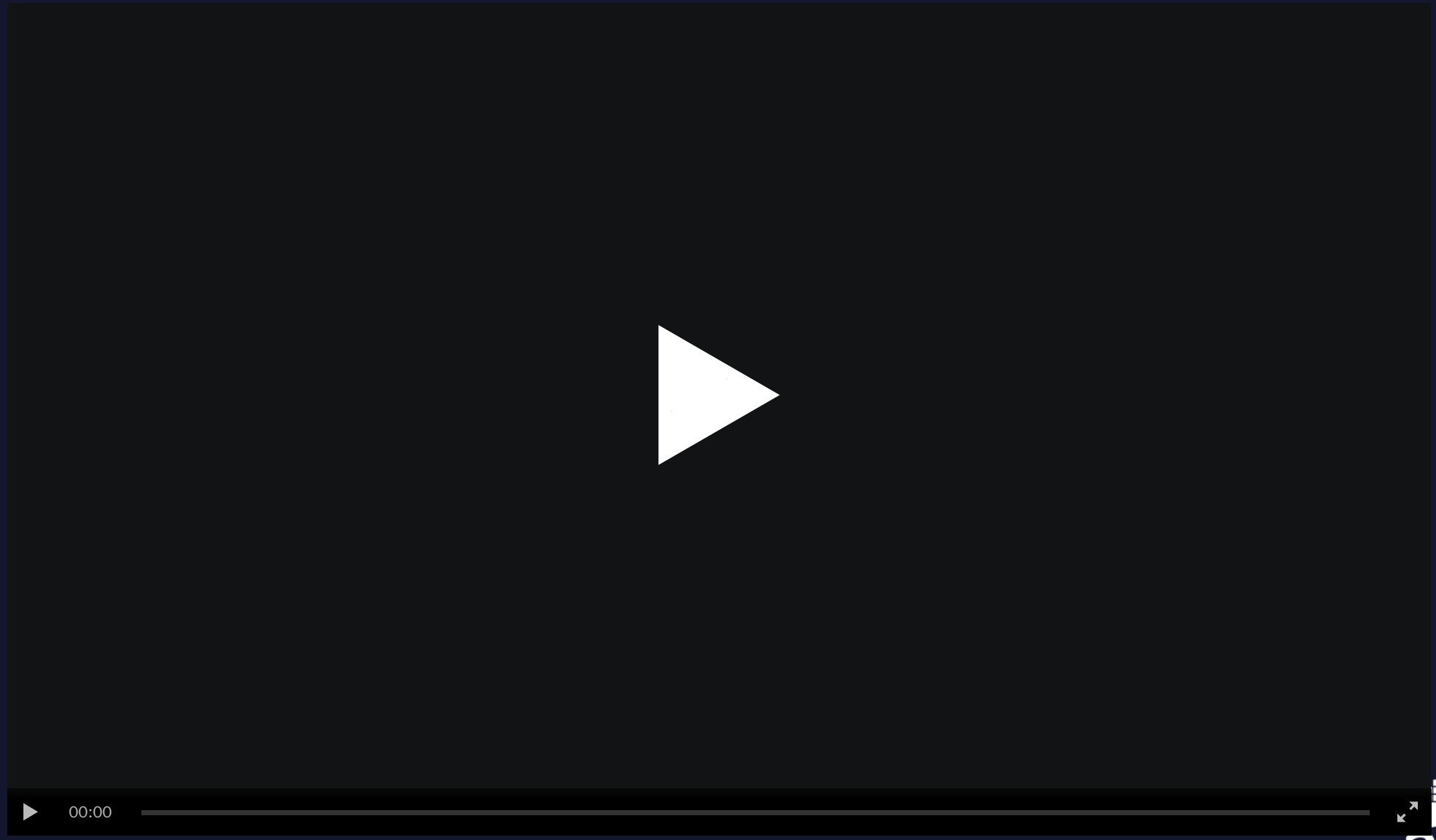
```
{ "registry-mirrors": [ "<cache-url>" ] }
```

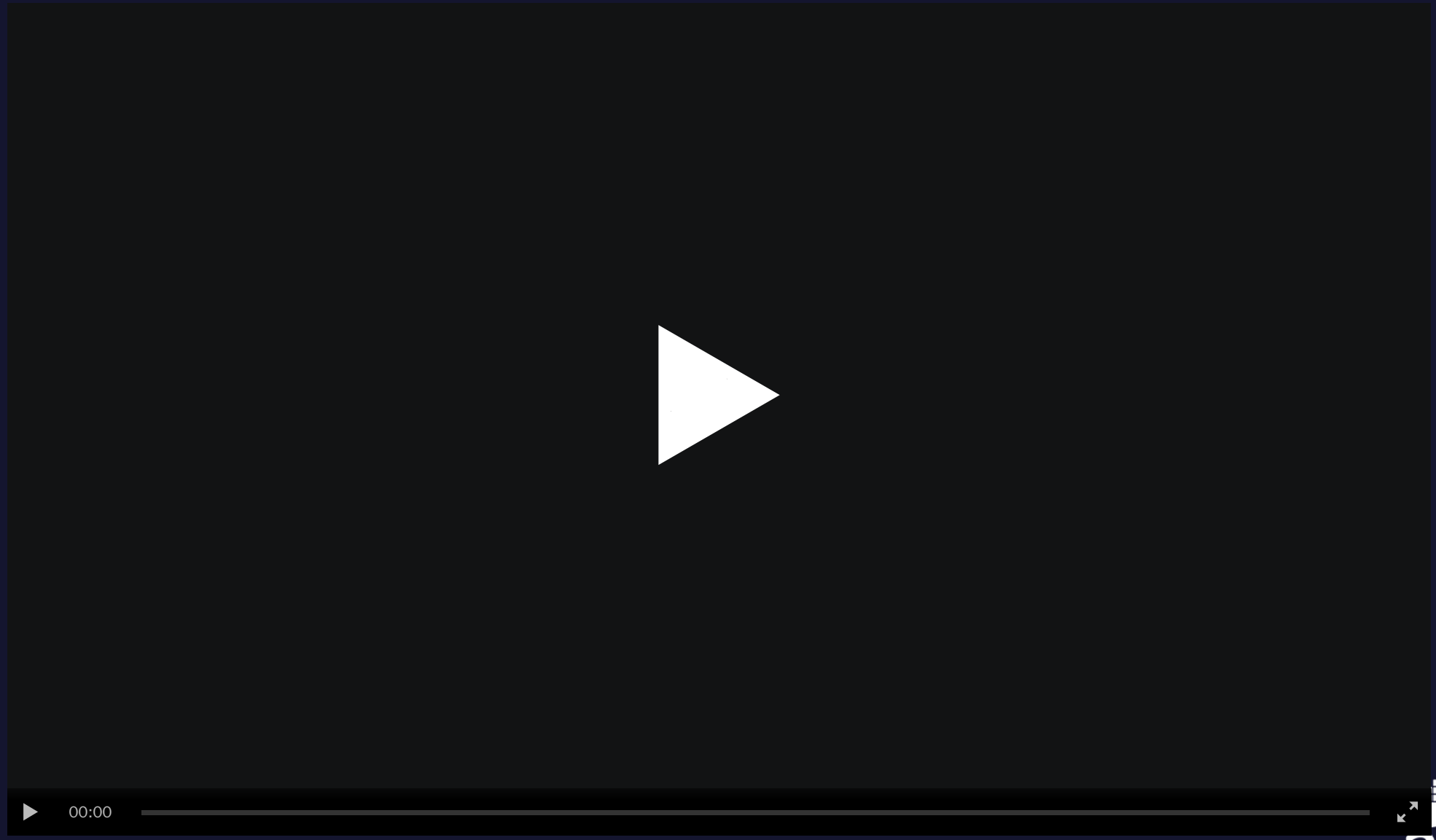
Plus a registry:

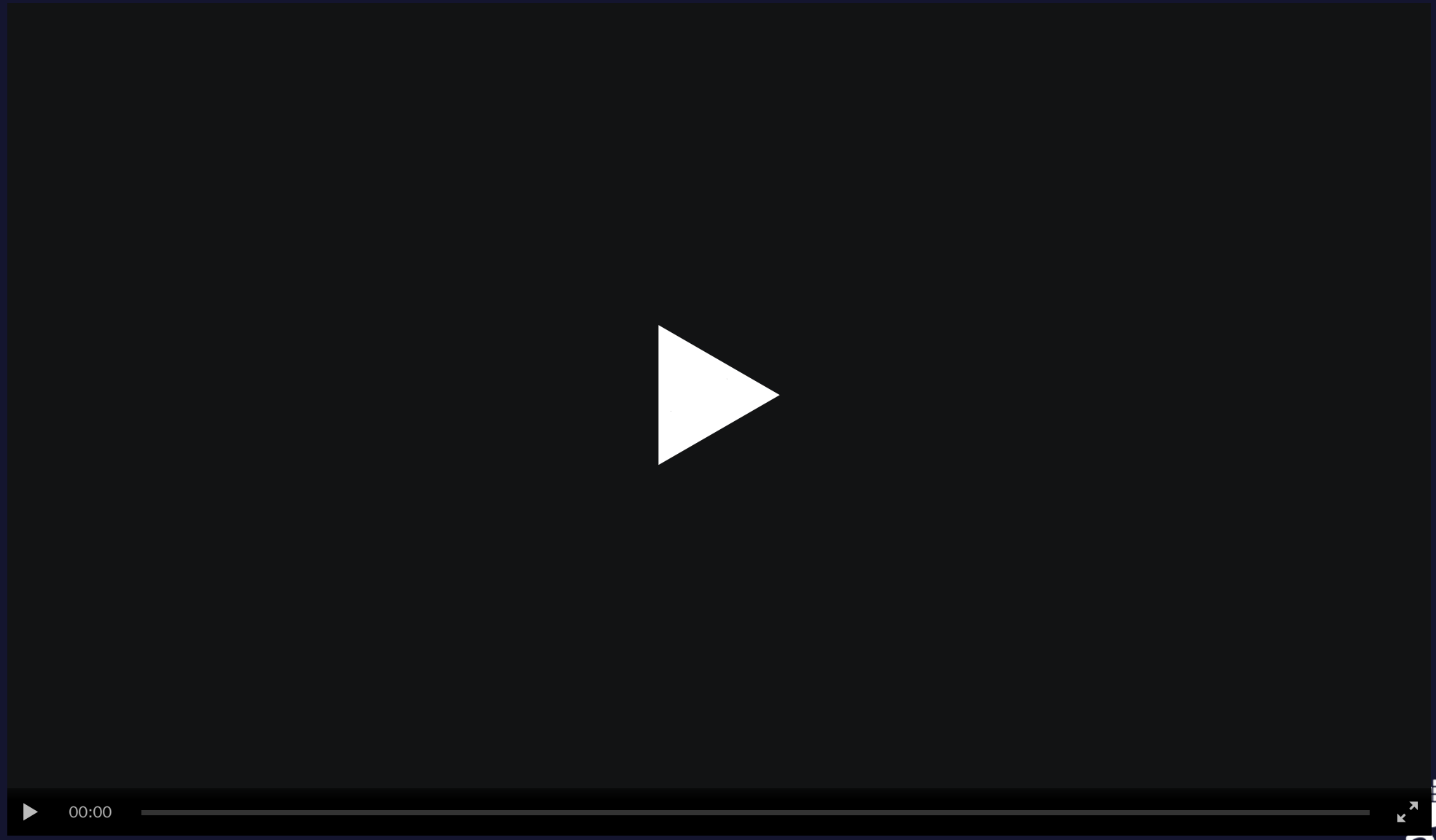
```
docker run -e REGISTRY_PROXY_REMOTEURL=<upstream-url> registry:2
```

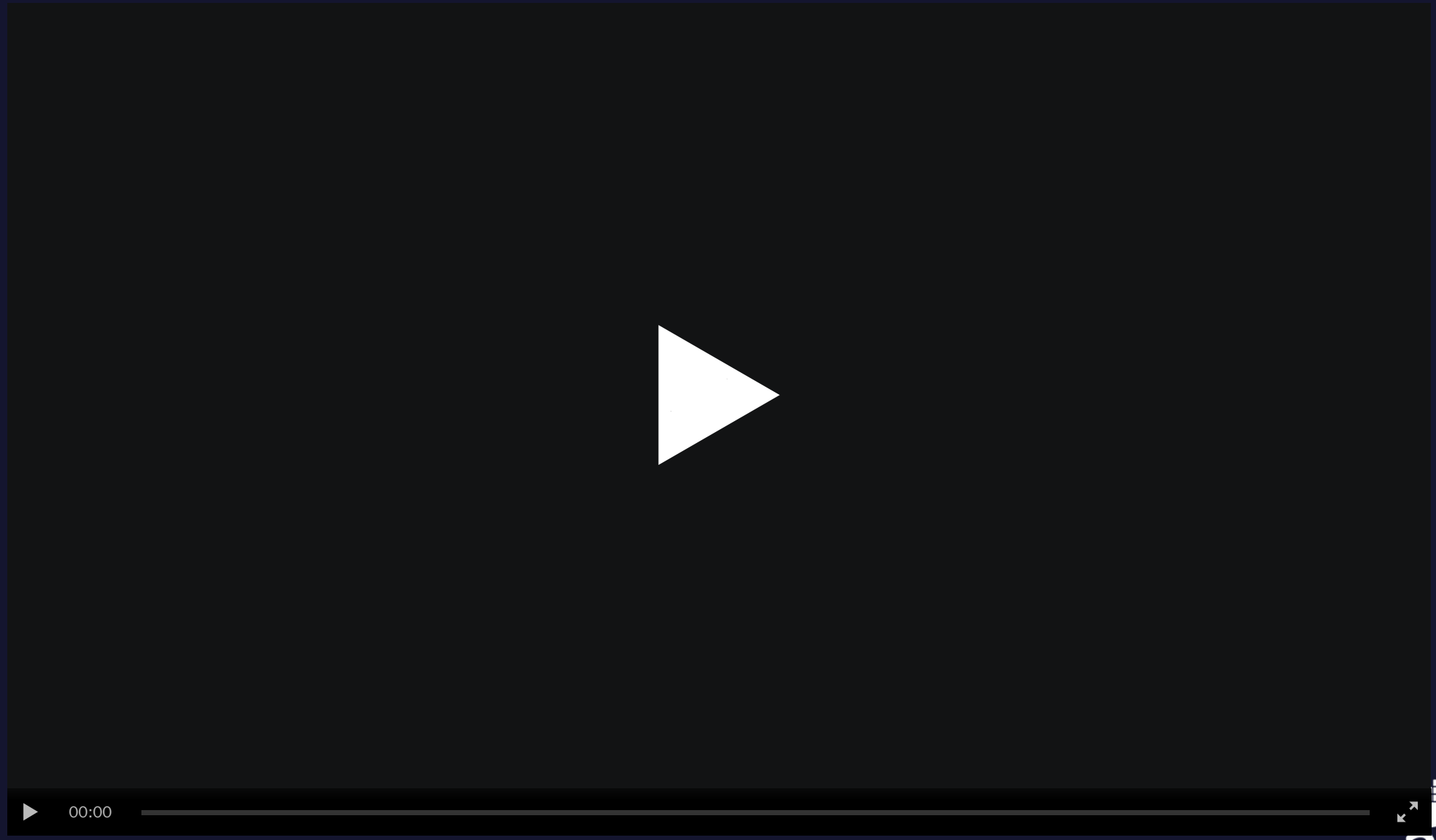

00:00


00:00

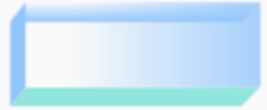

00:00


00:00


00:00


00:00

So What's the Catch?

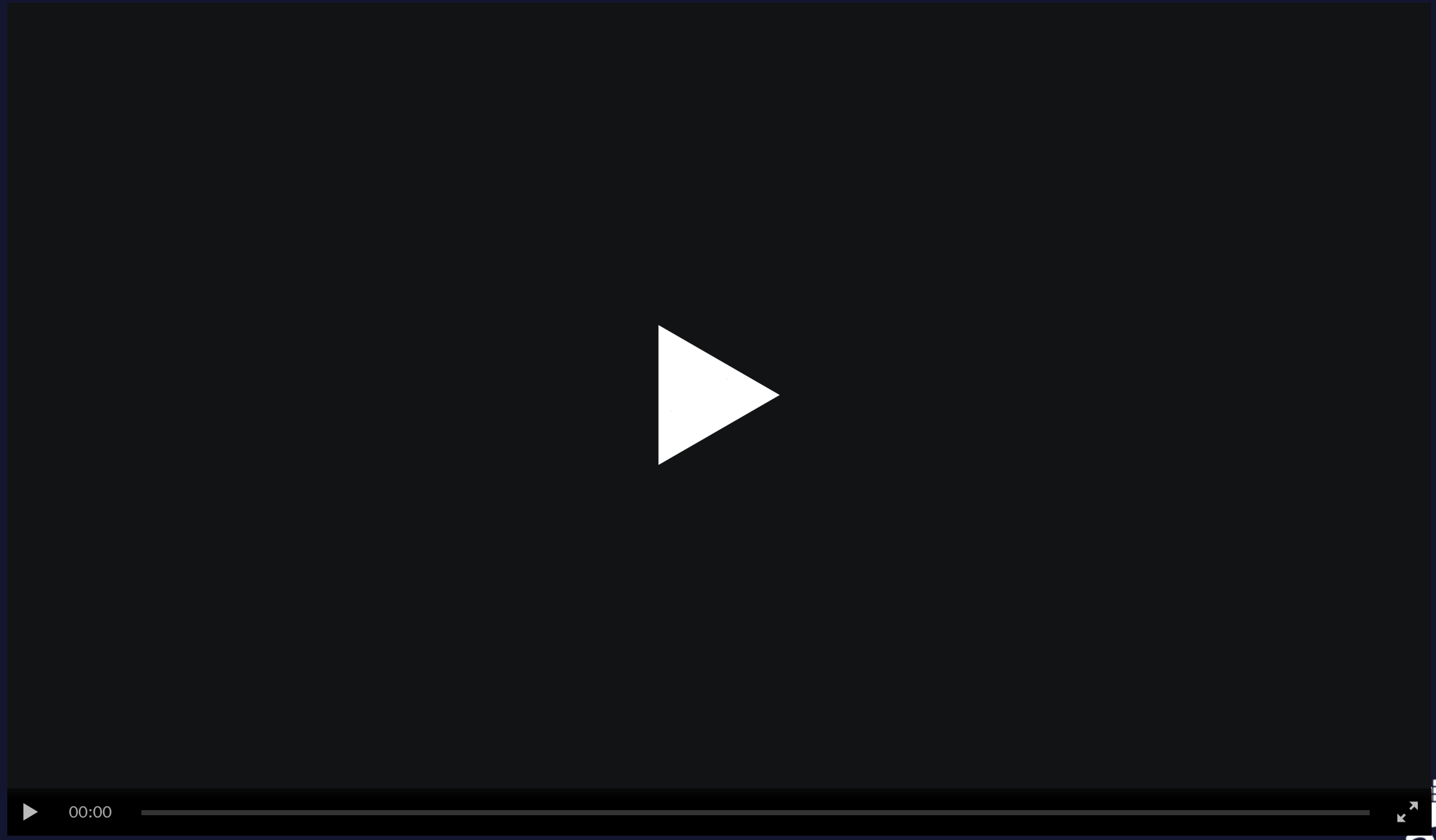


Cache Limitations

- The "registry-mirror" setting only applies to Docker Hub
- Only caches pulls not pushes
- Pulls still check the image manifest on Hub
- Credentials are in the cache server
- Docker implementation only supports one authentication method

Options to Cache Other Registries

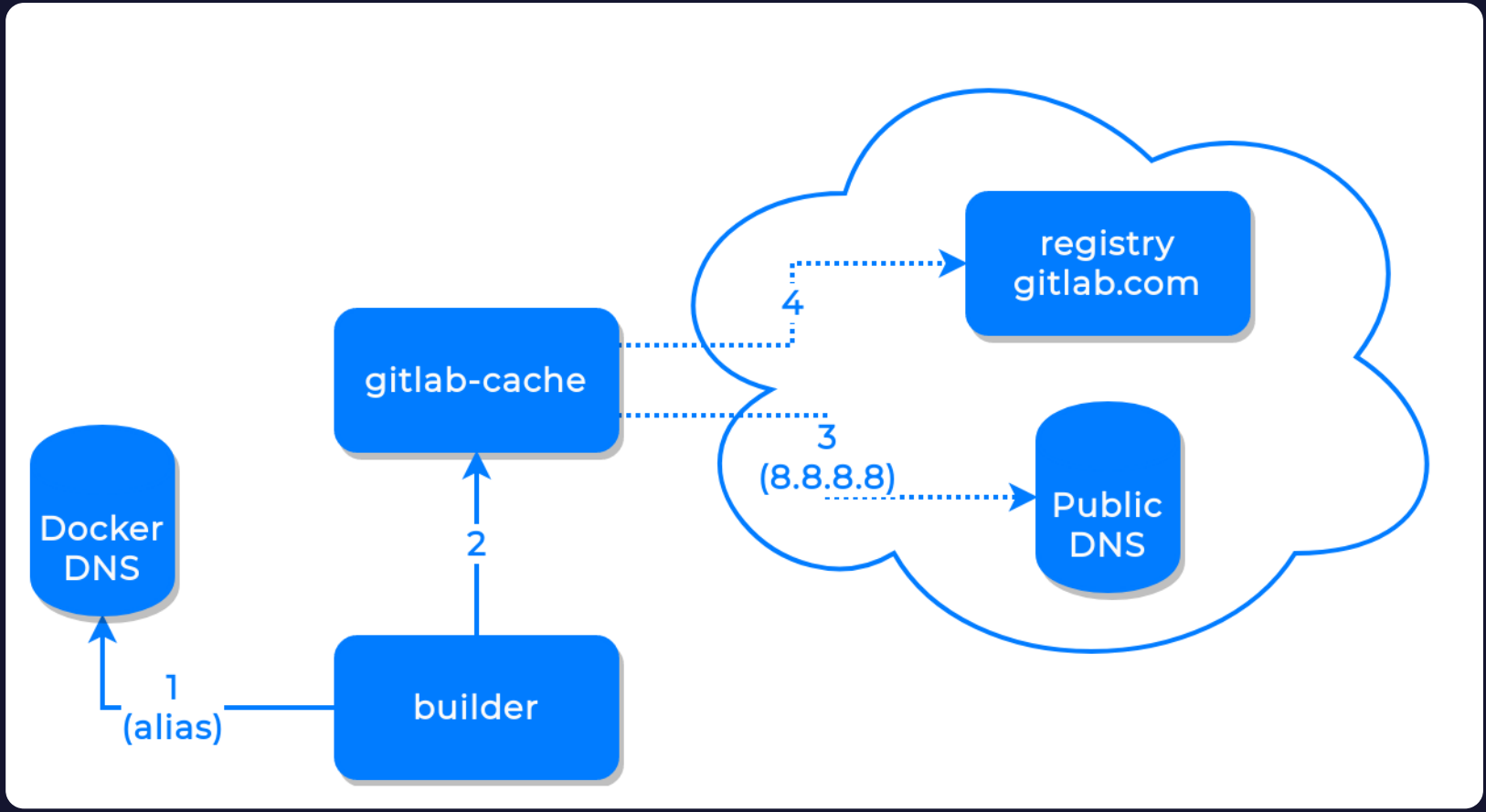
- Configure a squid HTTP caching proxy
- Pull directly from the cache
- Use DNS and TLS certs to send pulls to the proxy

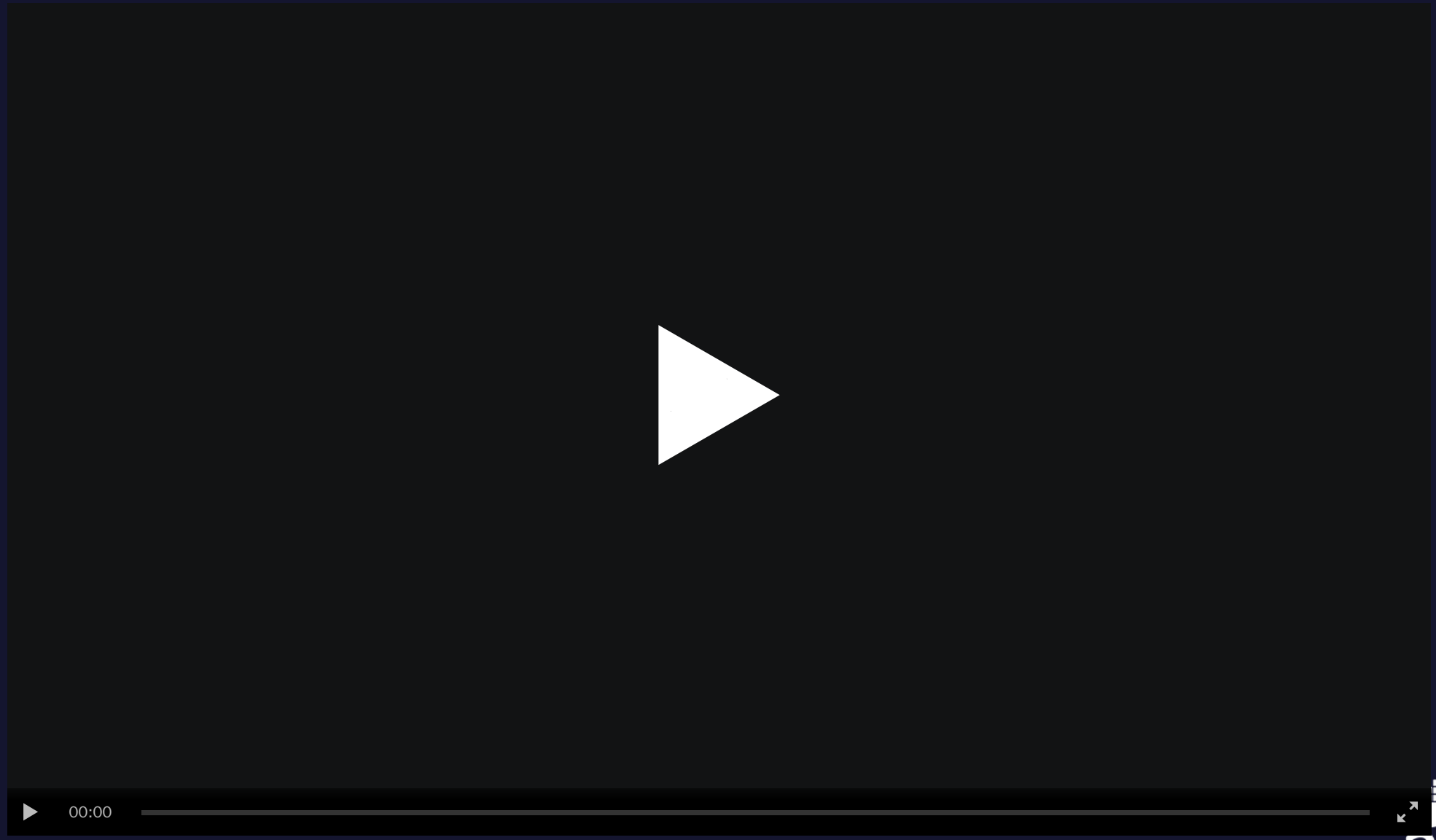

00:00

Intercepting DNS

```
version: '3.7'
services:
  gitlab-cache:
    image: registry:2
    networks:
      cache:
        aliases:
        - registry.gitlab.com
    dns:
    - 8.8.8.8
    - 8.8.4.4
```

Intercepting DNS

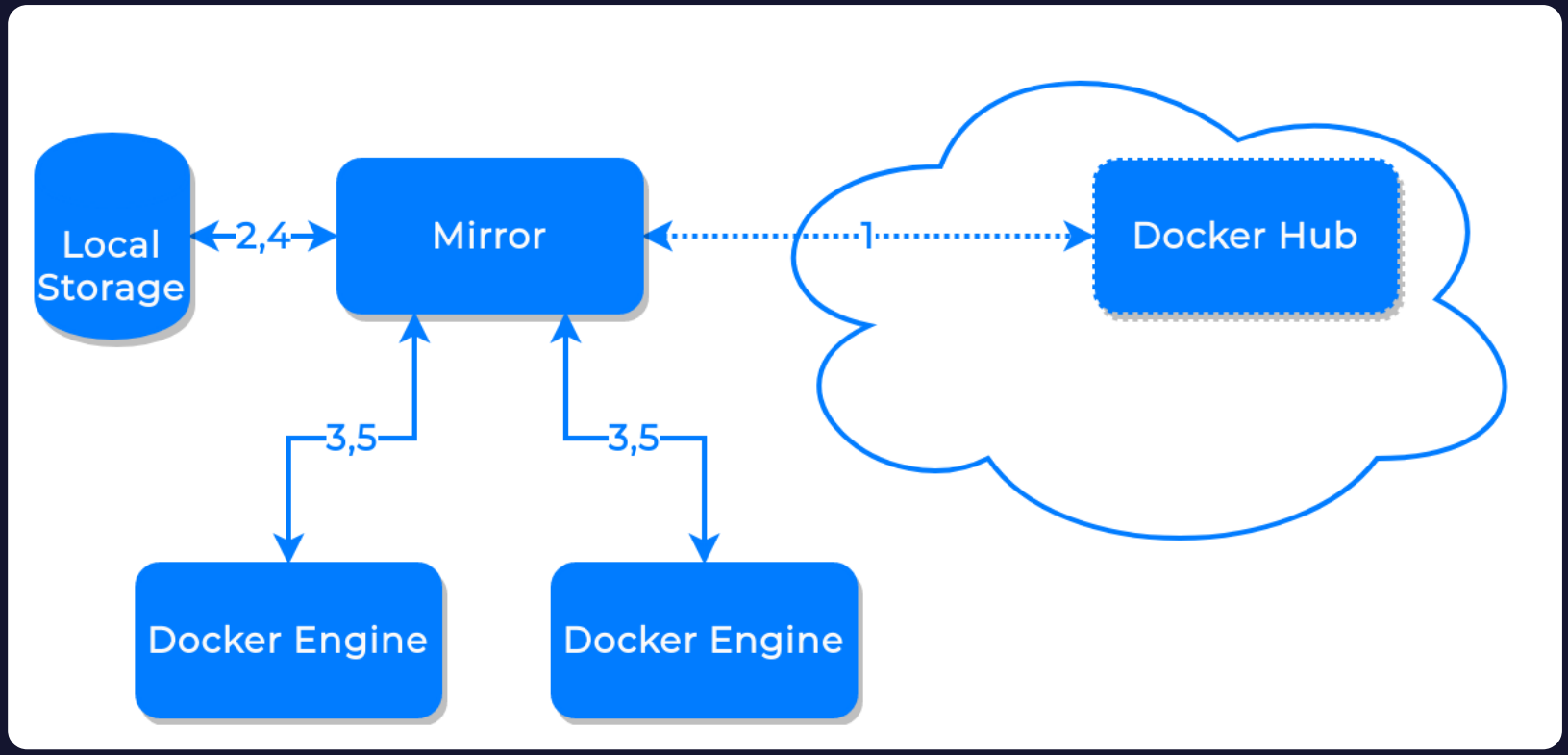



00:00

I Want More

Mirroring

Mirror Architecture



Running a Registry

- Docker image

```
docker container run -p 5000:5000 registry:2
```

- Harbor
- Many Artifact Repositories

Manually Mirroring

```
docker image pull ${image}
docker image tag ${image} local-mirror:5000/${image}
docker image push local-mirror:5000/${image}
```

Manual Mirror Script

```
docker image pull "$localimg"
docker image pull "$remoteimg"

remoteid=$(docker image inspect "$remoteimg" --format '{.Id}')
localid=$(docker image inspect "$localimg" --format '{.Id}')

if [ "$remoteid" != "$localid" ]; then
  docker image tag "$localimg" "$localimg.$datestamp"
  docker image tag "$remoteimg" "$localimg"
  docker image push "$localimg.$datestamp"
  docker image push "$localimg"
fi
```

Why All the Complication?

Advantages of Manually Mirroring

- Over Automatically Syncing Repos:
  - Changes to images happen on your schedule
  - Backout option exists with breaking changes
- Over Pull Through Cache
  - Those reasons plus...
  - Pushing locally built images to the registry
  - Upstream outage doesn't stop local builds/deploys

Risks of Manually Mirroring

- Images go stale if you do not automate the script
- Adding new images is an added process
- Recovering from a mirror outage requires populating images
- FROM line in images needs to point to mirror

```
ARG REGISTRY=docker.io
FROM ${REGISTRY}/alpine:3.9
...
```

```
docker build --build-arg REGISTRY=local-mirror:5000 .
```

Summary

Both

- Saves bandwidth
- Faster builds

Pull Through Cache

- Easy to create
- Little maintenance

Managed Mirror

- Control changes
- Tolerate upstream outages

Thank You

github.com/sudo-bmitch/presentations

Brandon Mitchell
Twitter: @sudo_bmitch
GitHub: sudo-bmitch