Harbor

James Zabala
Maintainer

Harbor Focus

Harbor is a trusted cloud native registry that stores, signs, and scans content. The mission is to provide cloud native environments the ability to confidently manage and serve container images.

Agenda

1. Containers 101
2. Introduction to **Harbor**
3. Image **Consistency**
4. Image **Security**
5. Image **Distribution**
6. Registry Robustness / **High Availability**

Agenda

1 **Containers 101**

2 Introduction to **Harbor**

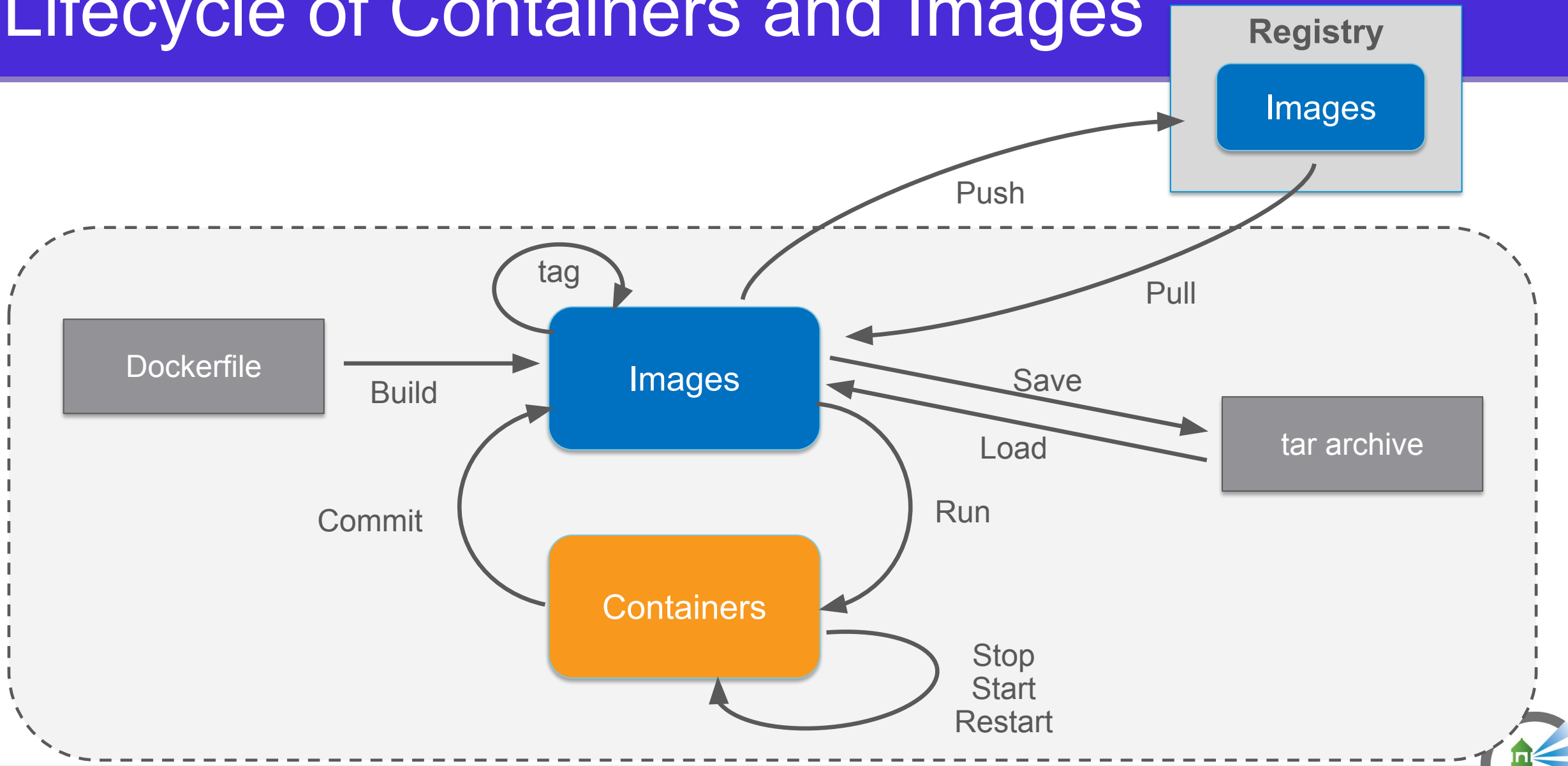
3 Image **Consistency**

4 Image **Security**

5 Image **Distribution**

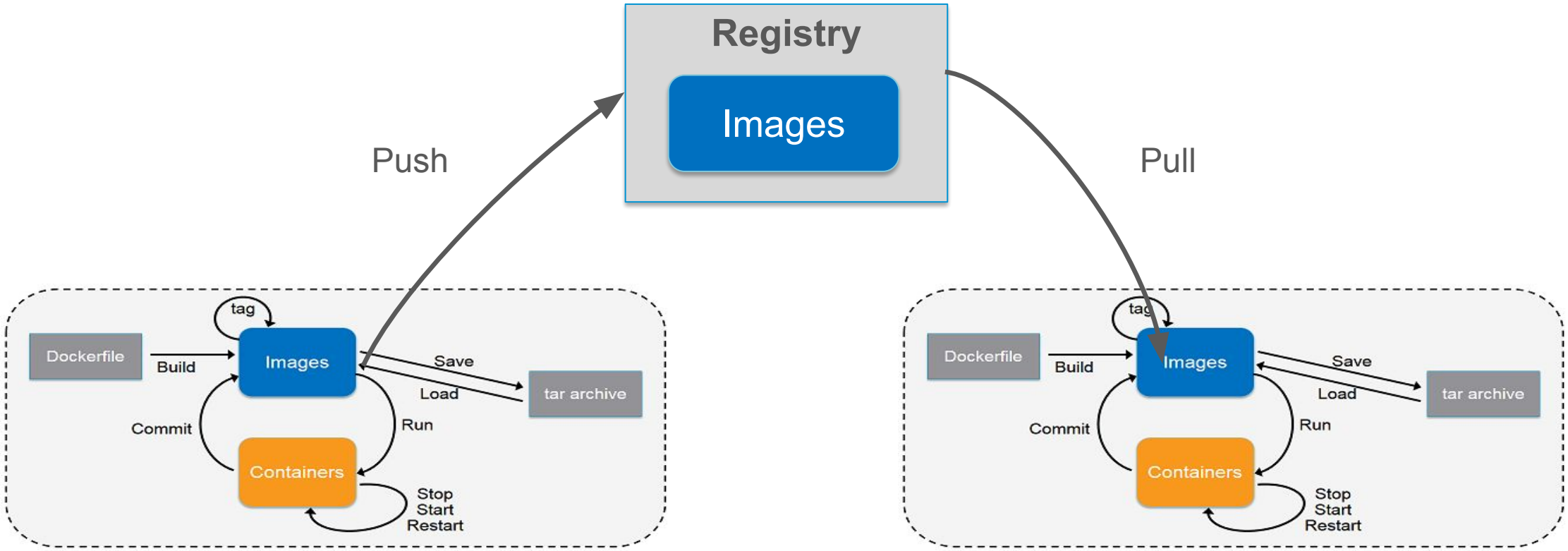
6 Registry Robustness / **High Availability**

Lifecycle of Containers and Images



Lifecycle of Containers and Images

- Repository for storing images
- Intermediary for shipping and distributing images and applying RBAC



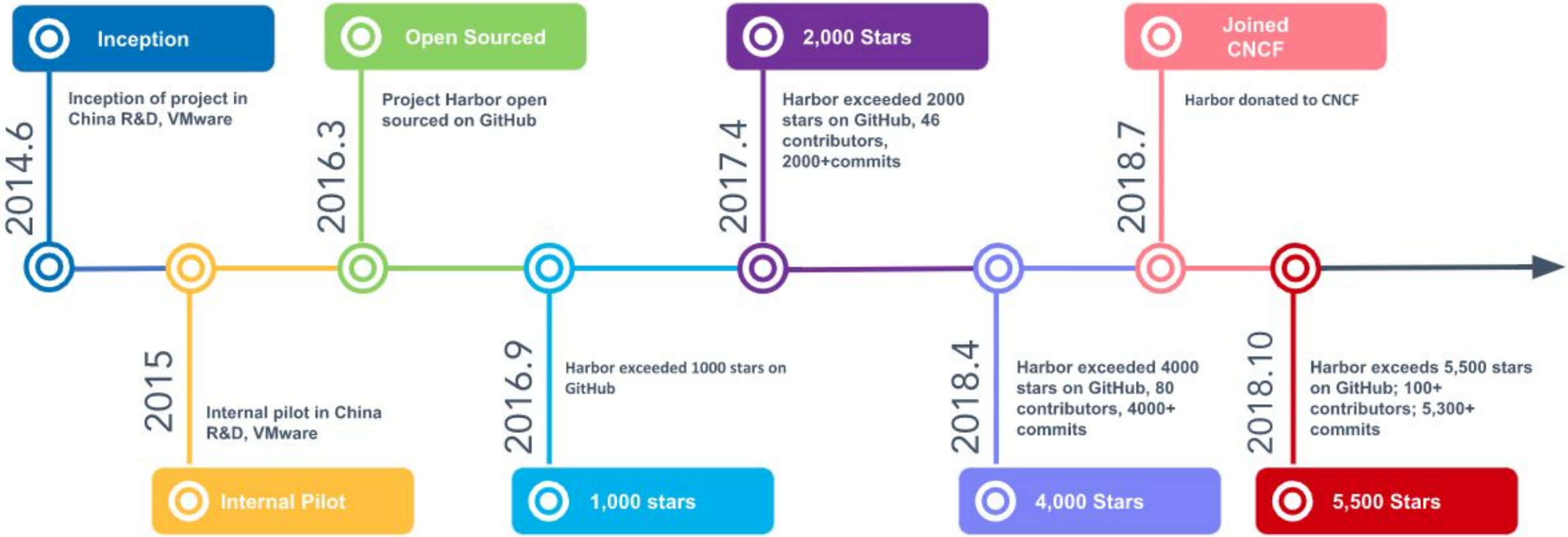
Agenda

1. Containers 101
2. **Introduction to Harbor**
3. Image **Consistency**
4. Image **Security**
5. Image **Distribution**
6. Registry Robustness / **High Availability**

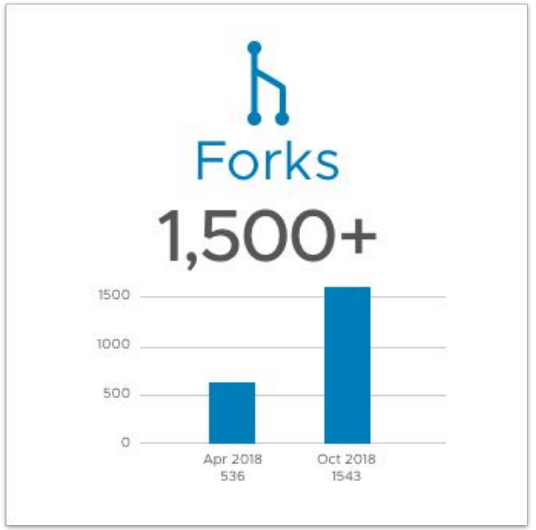
Project Harbor

- Created by VMware in 2014, adopted by users worldwide
- Registry for containers and Helm charts
- Focus: stores, signs and scans content
  - Provides consistent experience on- and off-prem
- Open Source (Apache 2.0)
- Accepted into sandbox stage in July 2018 as first container registry

Project History



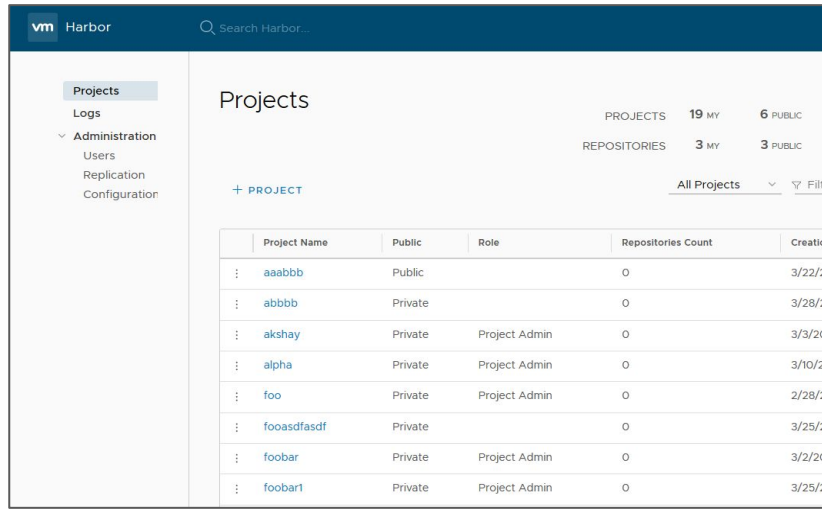
Open Source Stats



Key Features

Registry features include

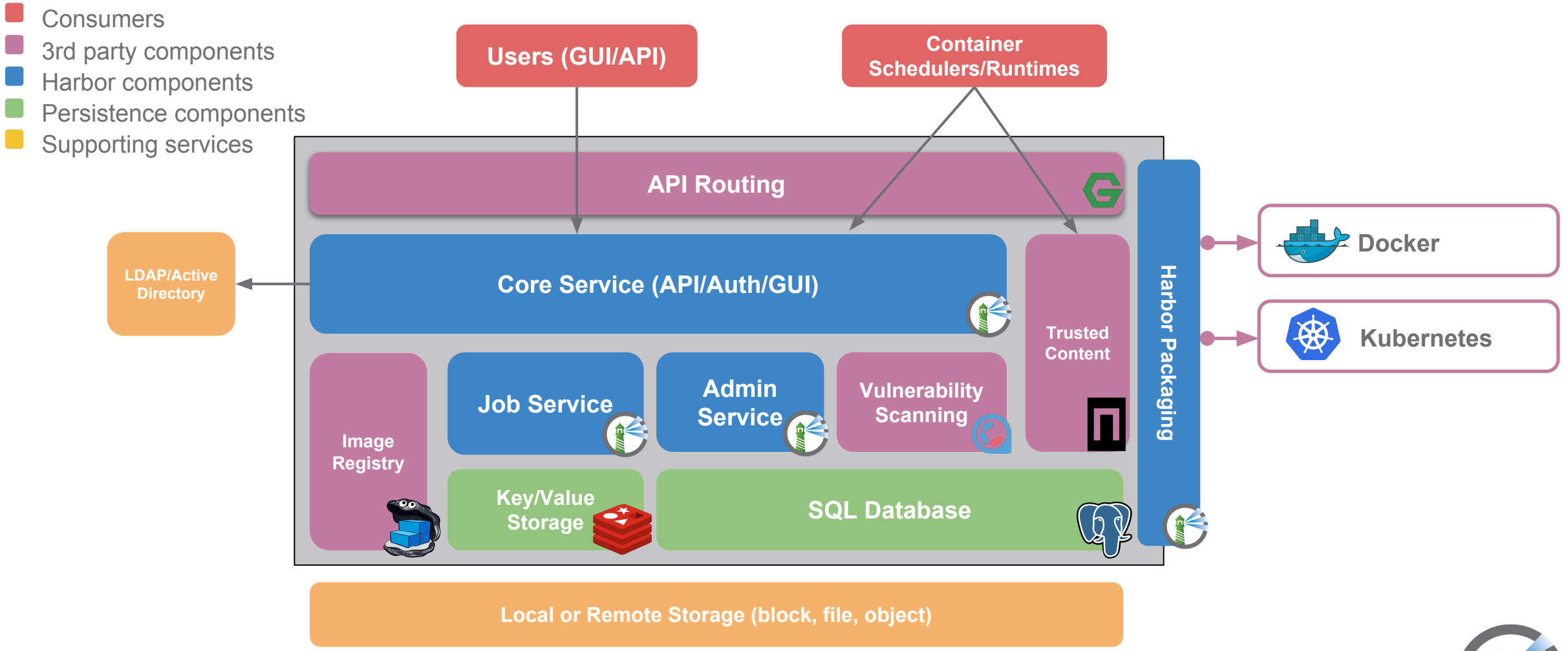
- **Multi-tenant content signing and validation**
- **Identity integration and role-based access control**
- **Security and vulnerability analysis**
- Image replication between instances
- Internationalization (currently English and Chinese)



Operational experience

- Deployed in containers
- Extends, manages, and integrates proven open source components

Architecture



Publicly Referenceable Customers

Agenda

1. Containers 101
2. Introduction to **Harbor**
3. **Image Consistency**
4. Image **Security**
5. Image **Distribution**
6. Registry Robustness / **High Availability**

Deterministic Images?

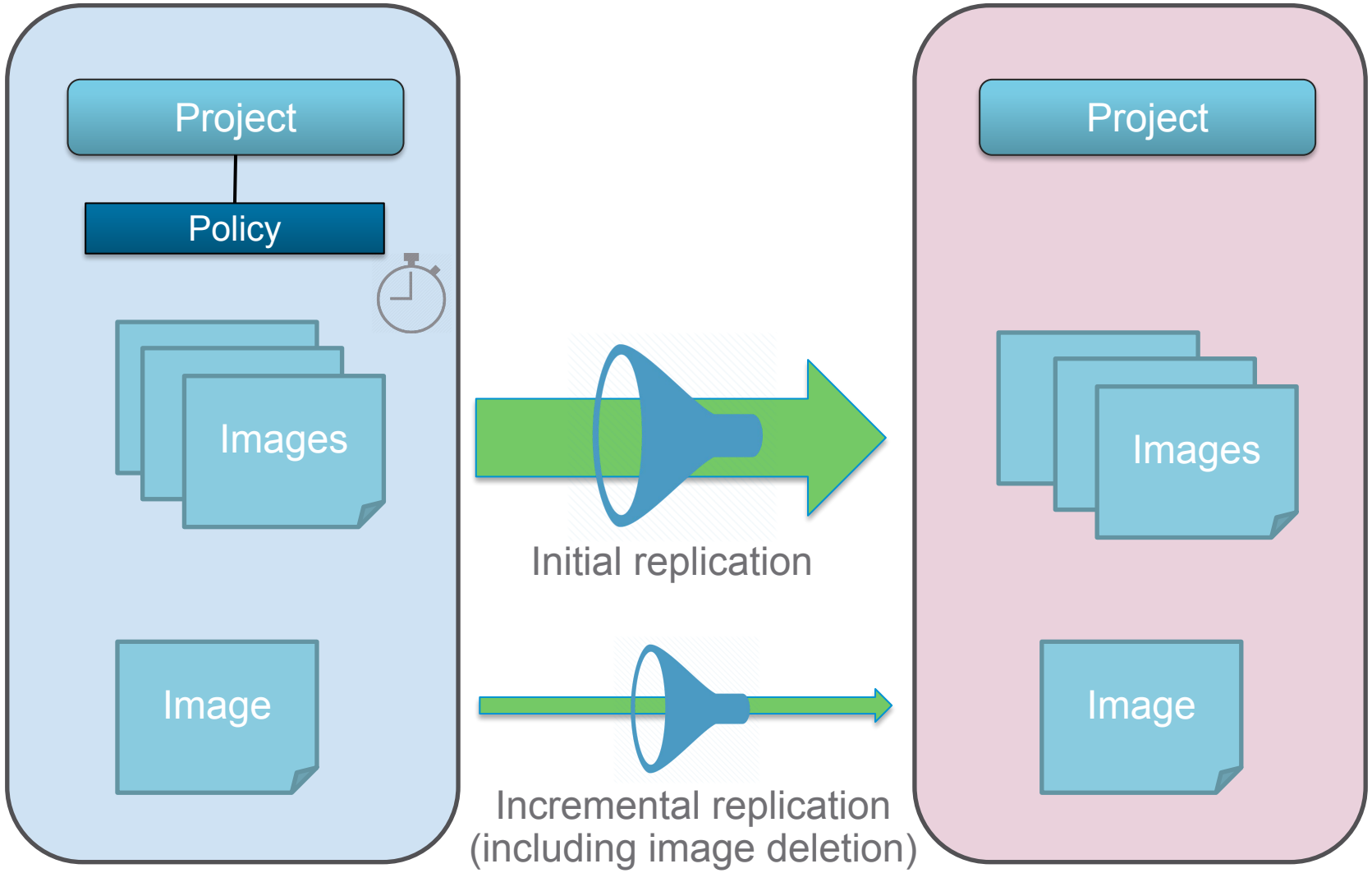
Dockerfile

```
FROM ubuntu
RUN apt-get install -y python
ADD app.jar /myapp/app.jar
```

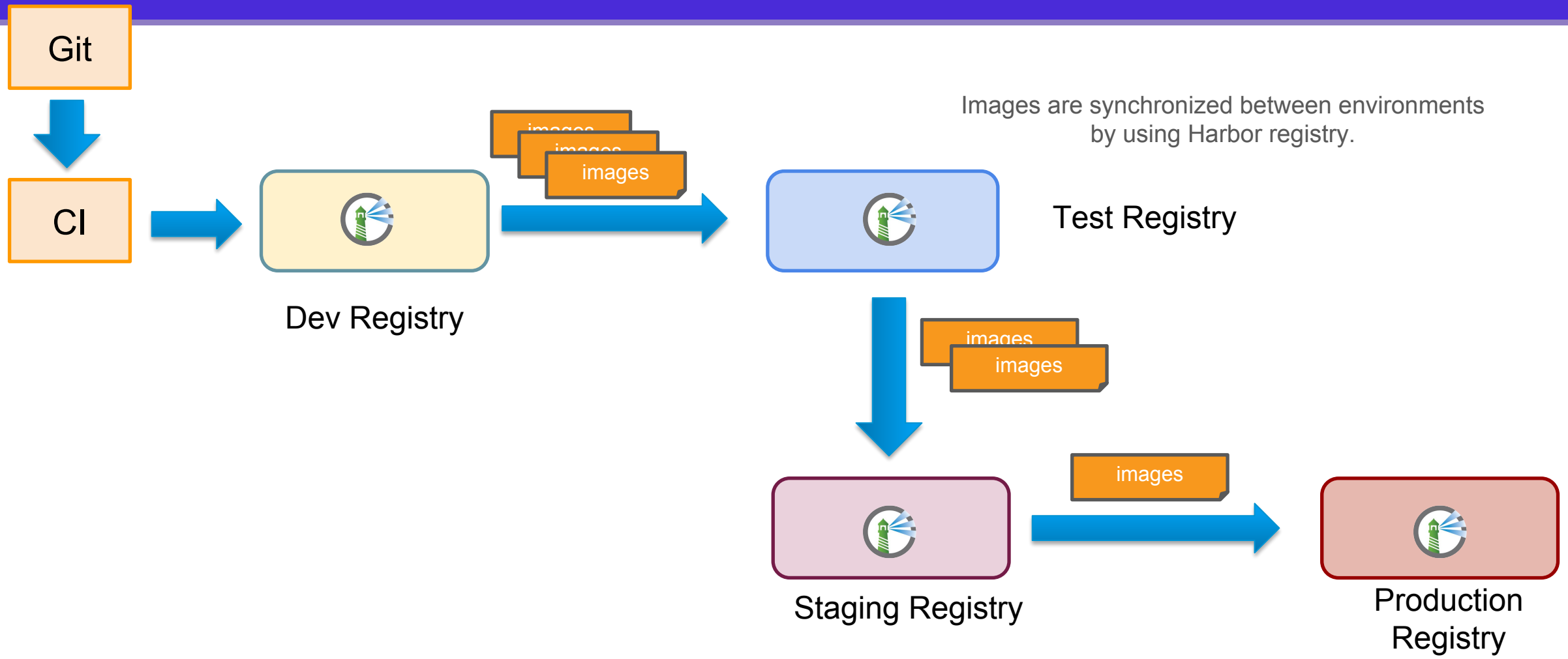
Challenges

- Base image `ubuntu:latest` could be changed between builds
- `ubuntu:14.04` could also be changed due to patching
- `apt-get` (`curl`, `wget`..) does not guarantee *identical* packages
- `ADD` depends on the build time environment to add files

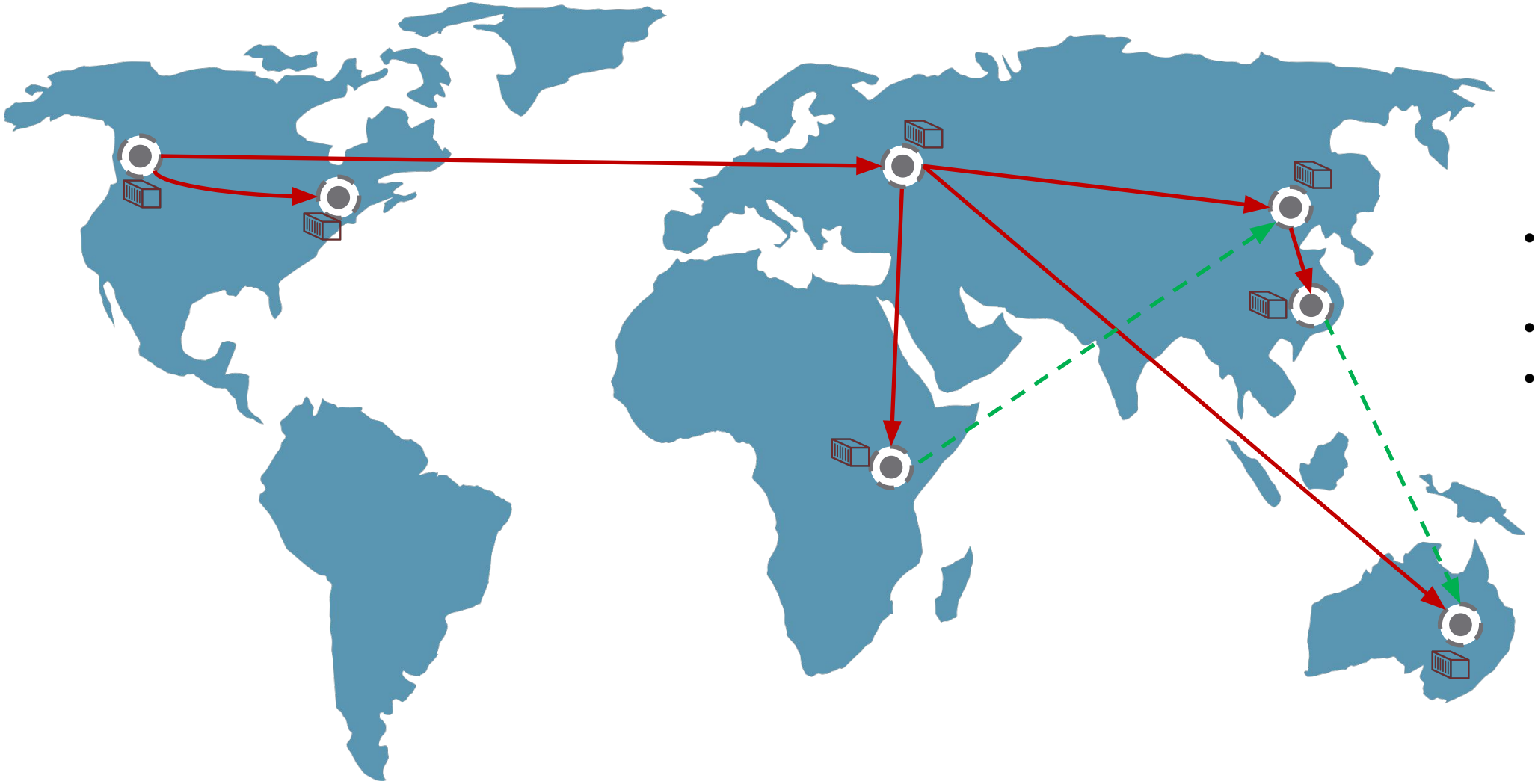
Image replication (synchronization)



Shipping image in “binary format”



Global Image Replication



- **Identical images across multiple sites**
- **Image backup**
- **Local access**

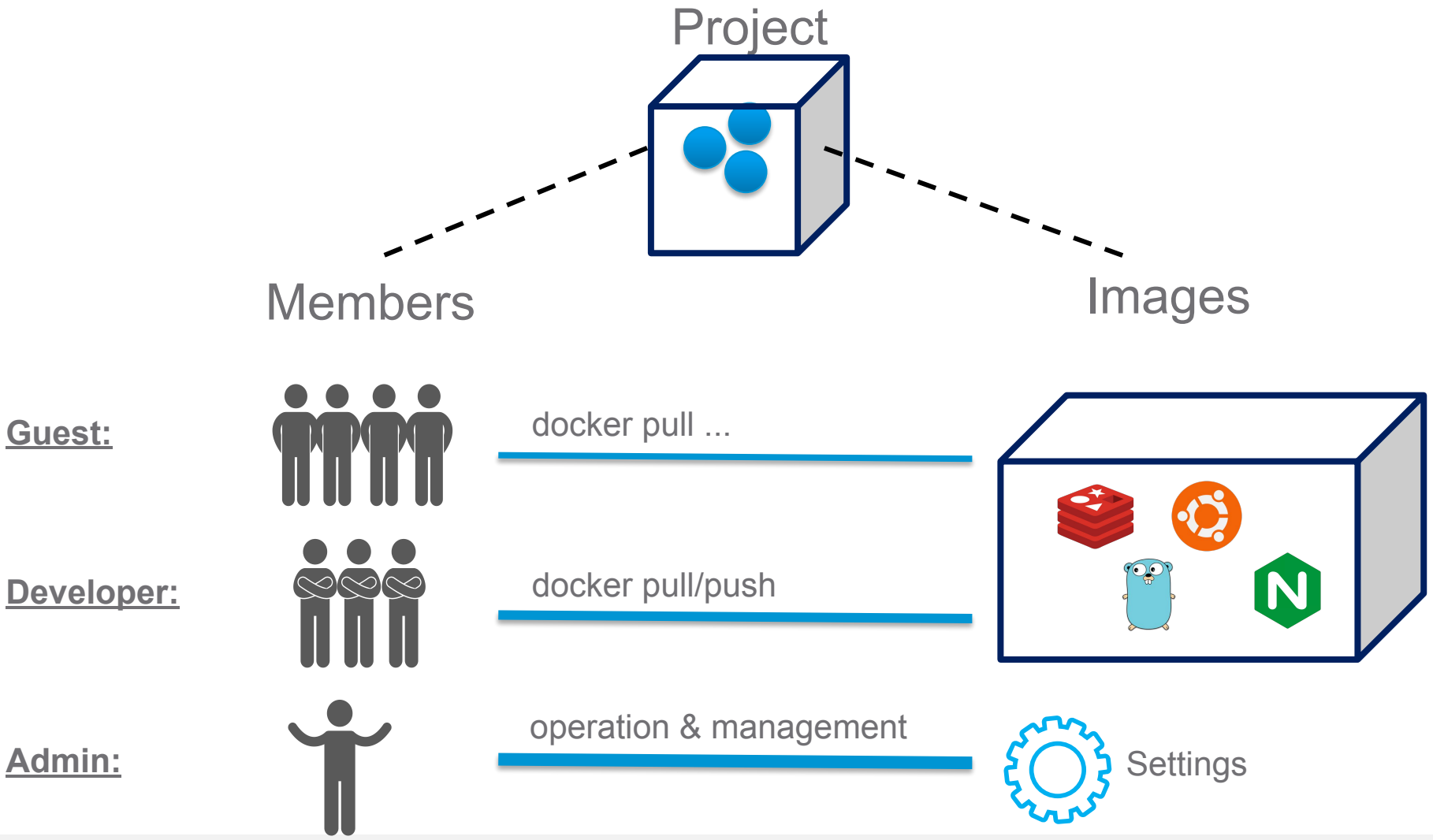
Agenda

1. Containers 101
2. Introduction to **Harbor**
3. Image **Consistency**
4. **Image Security**
5. Image **Distribution**
6. Registry Robustness / **High Availability**

Access Control to Images

- Organizations often keep images within their own organizations
  - Intellectual property stays in organization
- People with different roles should have different access
  - Developer – Read/Write
  - QA / QE – Read Only
- Different rules should be enforced in different environments
  - Dev/Test Environment – many people can access
  - Production – a limited number of people can access
- Can be integrated with internal user management system
  - LDAP/Active Directory

Role-Based Access Control (RBAC)



Other security considerations

- Enable content trust by installing Notary service
  - Image is signed by publisher's private key during pushing
  - Image is pulled using digest
- Perform vulnerability scanning
  - Prevent images with vulnerabilities from being pulled
  - Regular scanning based on updated vulnerability database

Content trust for image provenance

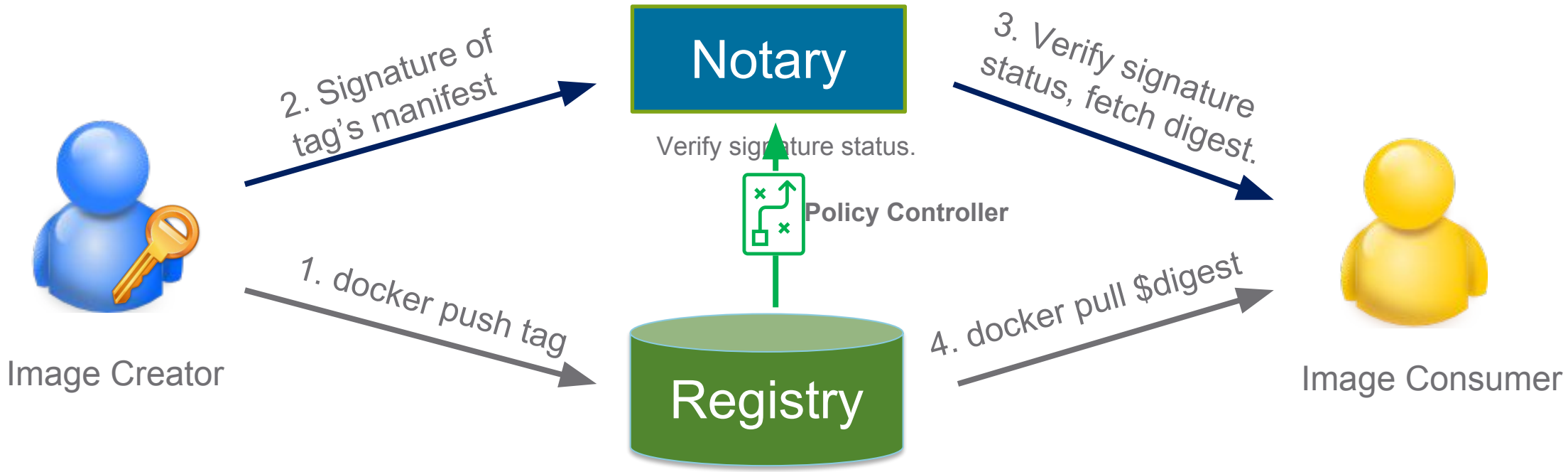


Image Vulnerability Scanning

Vulnerability scanning

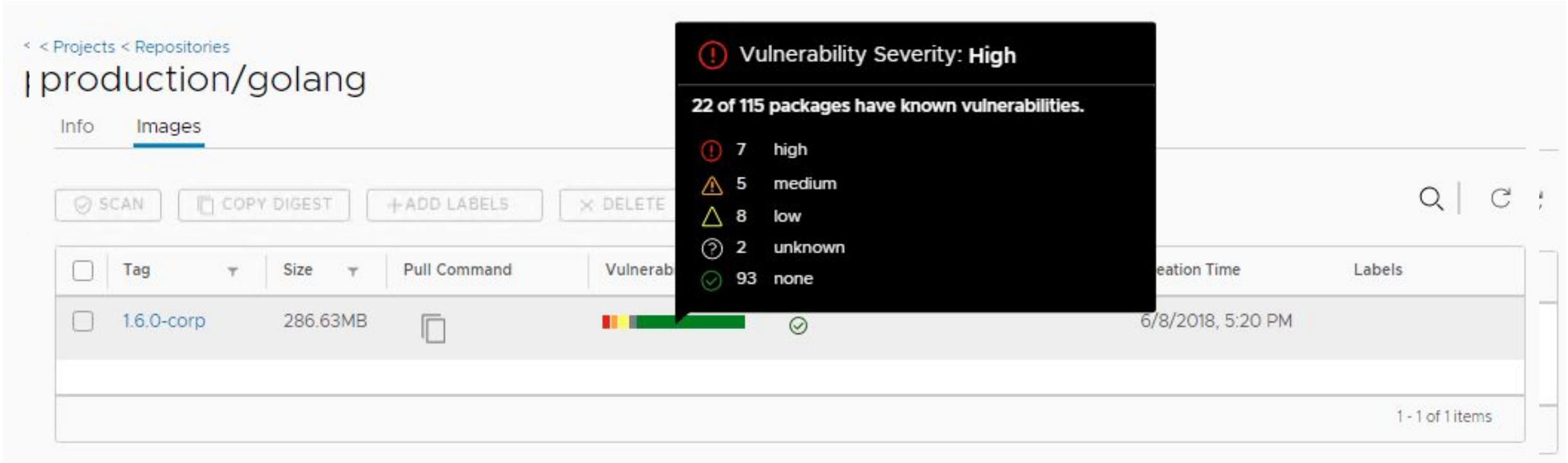
- Set vulnerability **threshold**
- **Static analysis** of vulnerability by inspecting filesystem of container image and indexing features in database
- Prevent images from being pulled if they exceed threshold
- Periodic scanning based on updated vulnerability database

Image Vulnerability Scanning

Update vulnerability data regularly from various sources:

- Debian security Bug Tracker
- Ubuntu CVE Tracker
- Red Hat Security Data
- Oracle Linux Security Data
- Alpine SecDB
- NIST

Web interface and vulnerability scanning



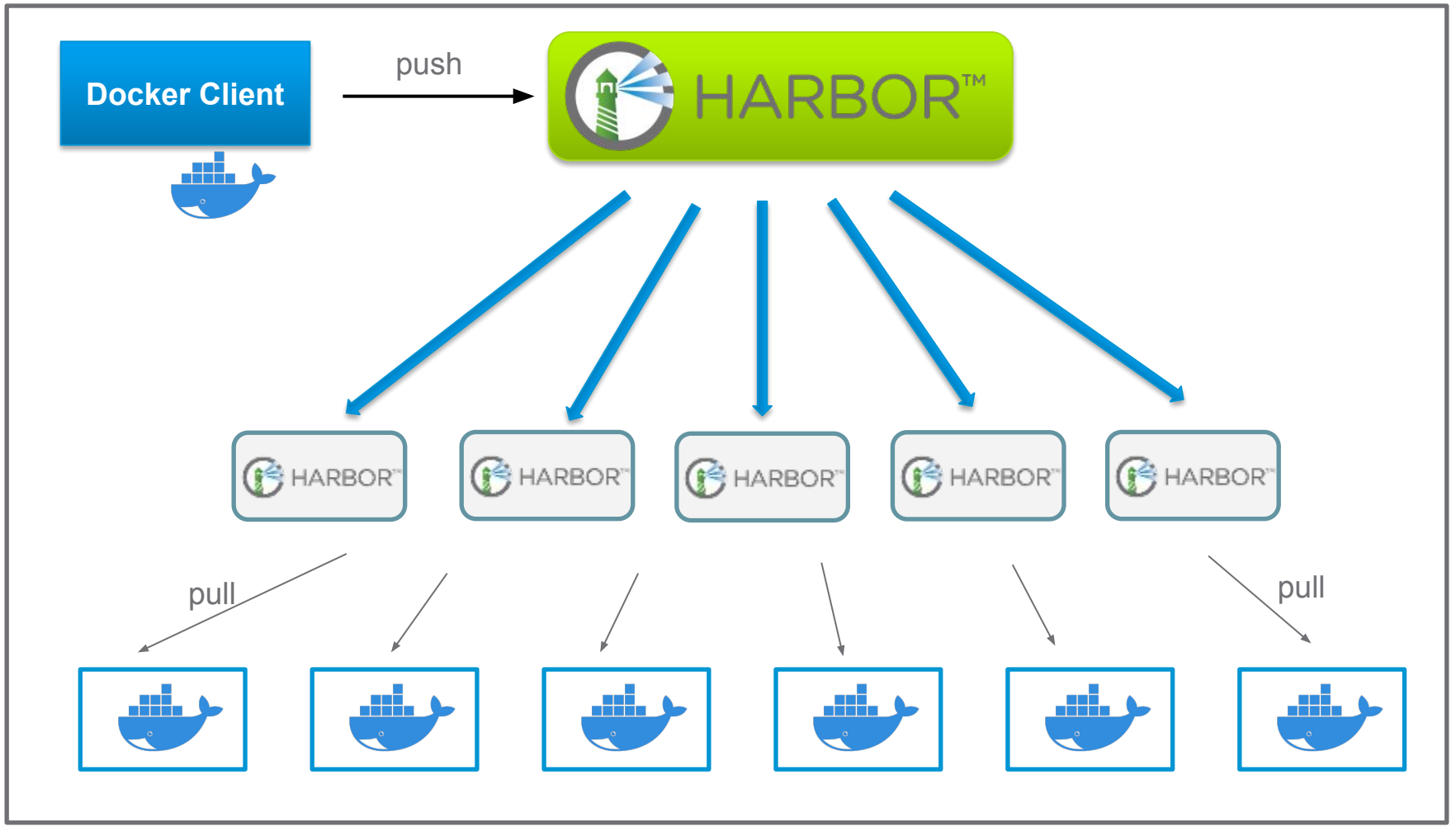
Agenda

1. Containers 101
2. Introduction to **Harbor**
3. Image **Consistency**
4. Image **Security**
5. **Image Distribution**
6. Registry Robustness / **High Availability**

Image distribution

- Container images are usually distributed from a registry
- Registry becomes the bottleneck for a large cluster of nodes
  - I/O
  - Network
- Scaling out an registry server
  - Multiple instances of registry sharing same storage (such as S3 or local Ceph cluster)
  - Multiple instances of independent registry sharing no storage

Replication



- Load balancing
- Works well with geographically distributed clients

Agenda

1. Containers 101
2. Introduction to **Harbor**
3. Image **Consistency**
4. Image **Security**
5. Image **Distribution**
6. **Registry Robustness / High Availability**

High Availability of Registry

- Goal: remove single point of failure on registry
- Three models to achieve HA
  - Shared storage
  - Replication (no shared storage)
  - Using other HA platform
- Current focus: easy HA deployment via Helm chart
- Evaluating cluster-like functionality to automagically share data between nodes

Demo

Roadmap

Roadmap

- v1.7 release features being worked on
- v1.8 roadmap: open for feedback :)
  - (see next slide)
  - Ping us on GitHub

Roadmap

- Quotas
- Lifecycle support of Harbor on K8s
- Image proxy'ing + cache
  - Update / rollback of upstream cache
  - Caching of upstream repos (e.g., DockerHub)
- Token-based auth
- Image scanning improvements
- Clustering – local and remote
- Increase scalability
- Improved RBAC
- Improved multi-tenancy
- `harborctl` CLI client
- Tag lifecycle management

Contributing to Future of Harbor

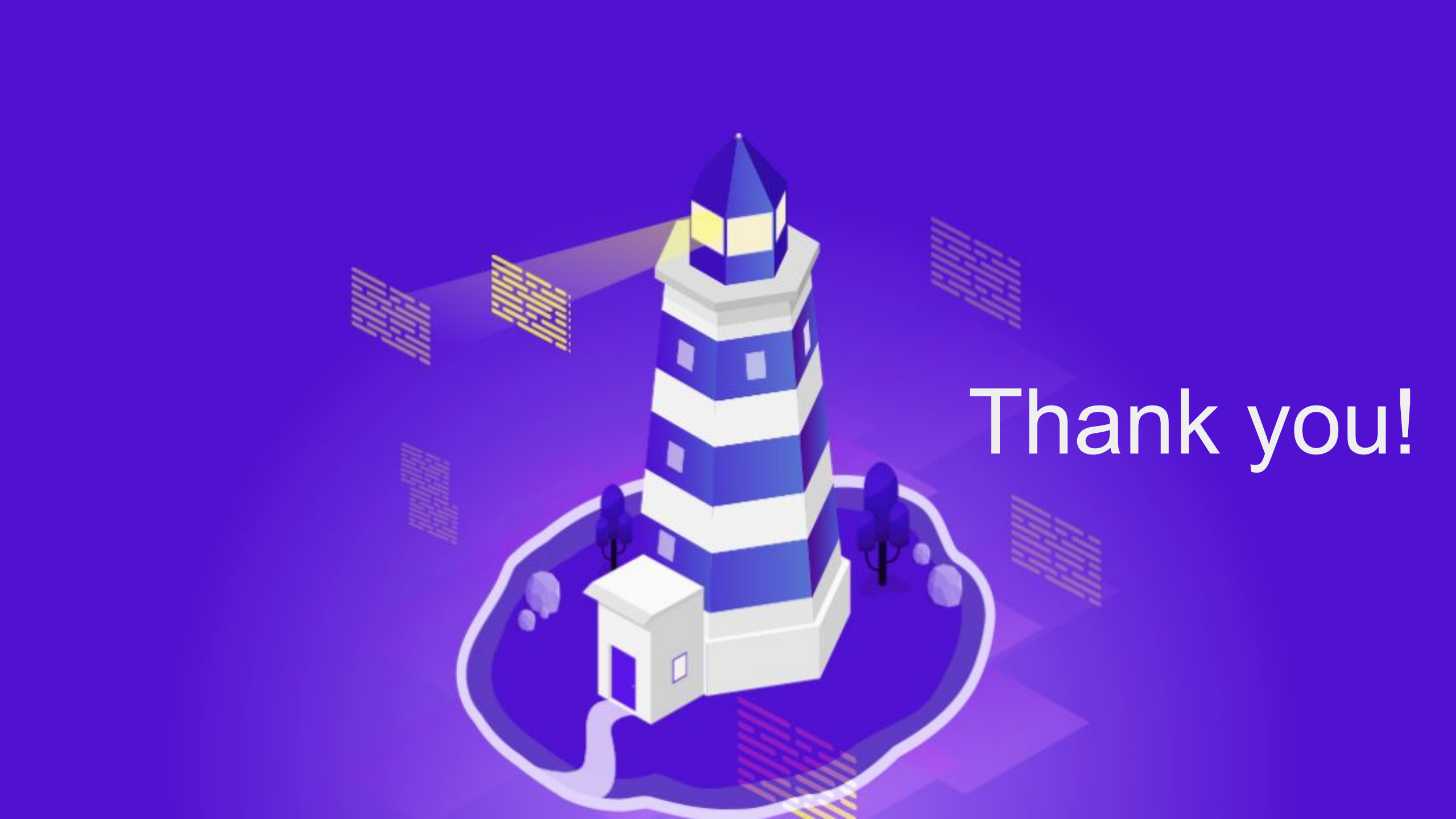
- Contributions of all kinds are welcome
  - Documentation
  - Issues
    - Finding and opening issues
    - Wrangling issues
  - Code (and reviews!)
  - Testing builds
  - Etcetera

Ping us!

web: https://www.goharbor.io

gh: https://github.com/goharbor

slack: slack.cncf.io (#habor and #harbor-dev)


Thank you!