Secure and Efficient Container Image Management in Enterprise

Henry Zhang
Chief Architect, Cloud Native Apps
R&D, VMware China

vmware

About Me

- Chief Architect, VMware China R&D
  - Current focus: Cloud Native Apps, Blockchain, IoT
- Creator and Architect of Project Harbor, an open source enterprise class container registry
- Full stack engineer
- Coauthor of two books (in Chinese):
  - Blockchain Technical Guides
  - Software Defined Storage: Principle, Practice and Ecosystem

vmware

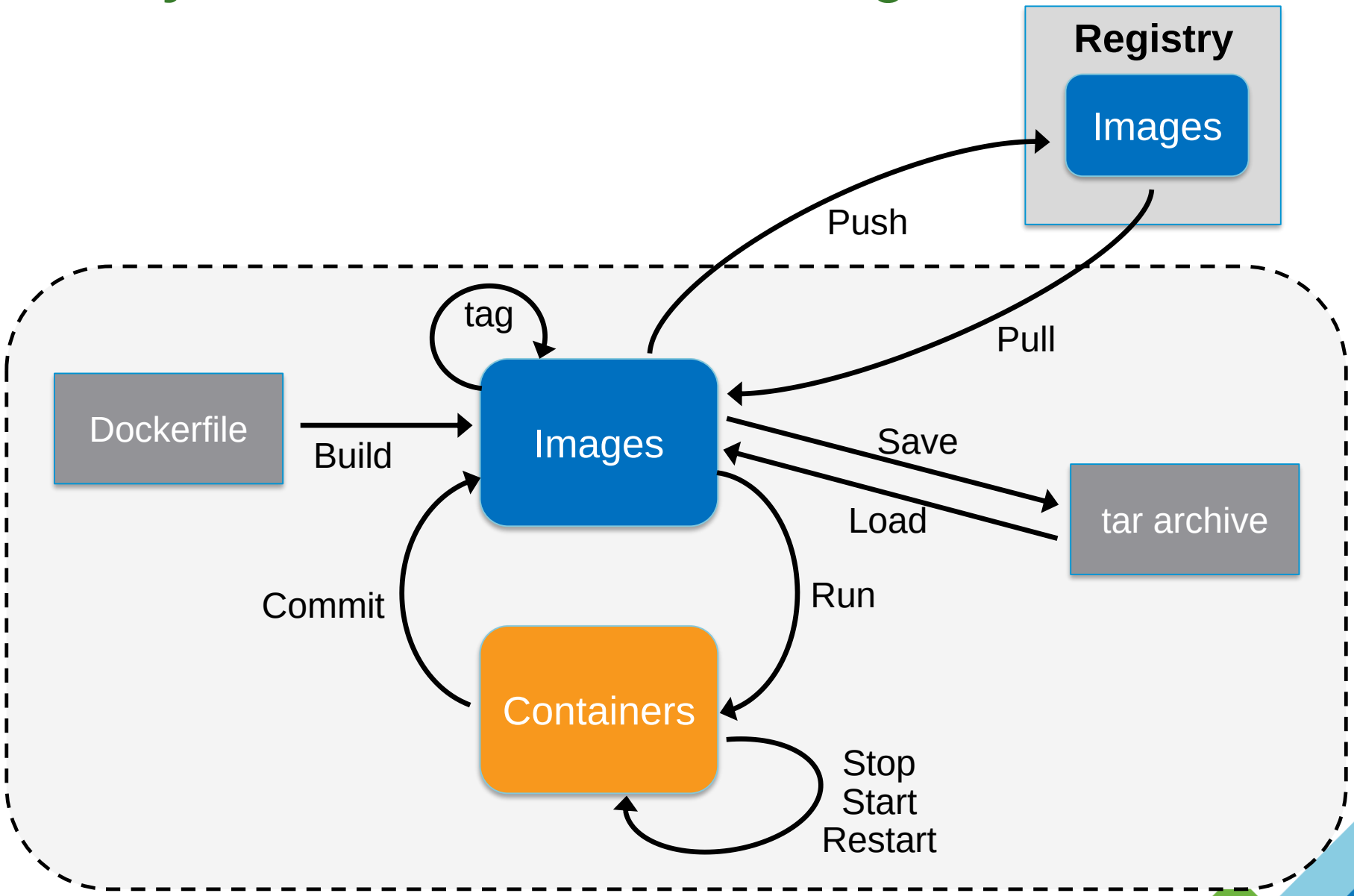
Agenda

1. Container Image Basics
2. Project Harbor Introduction
3. Consistency of Images
4. Security
5. Image Distribution
6. High Availability of Registry

Agenda

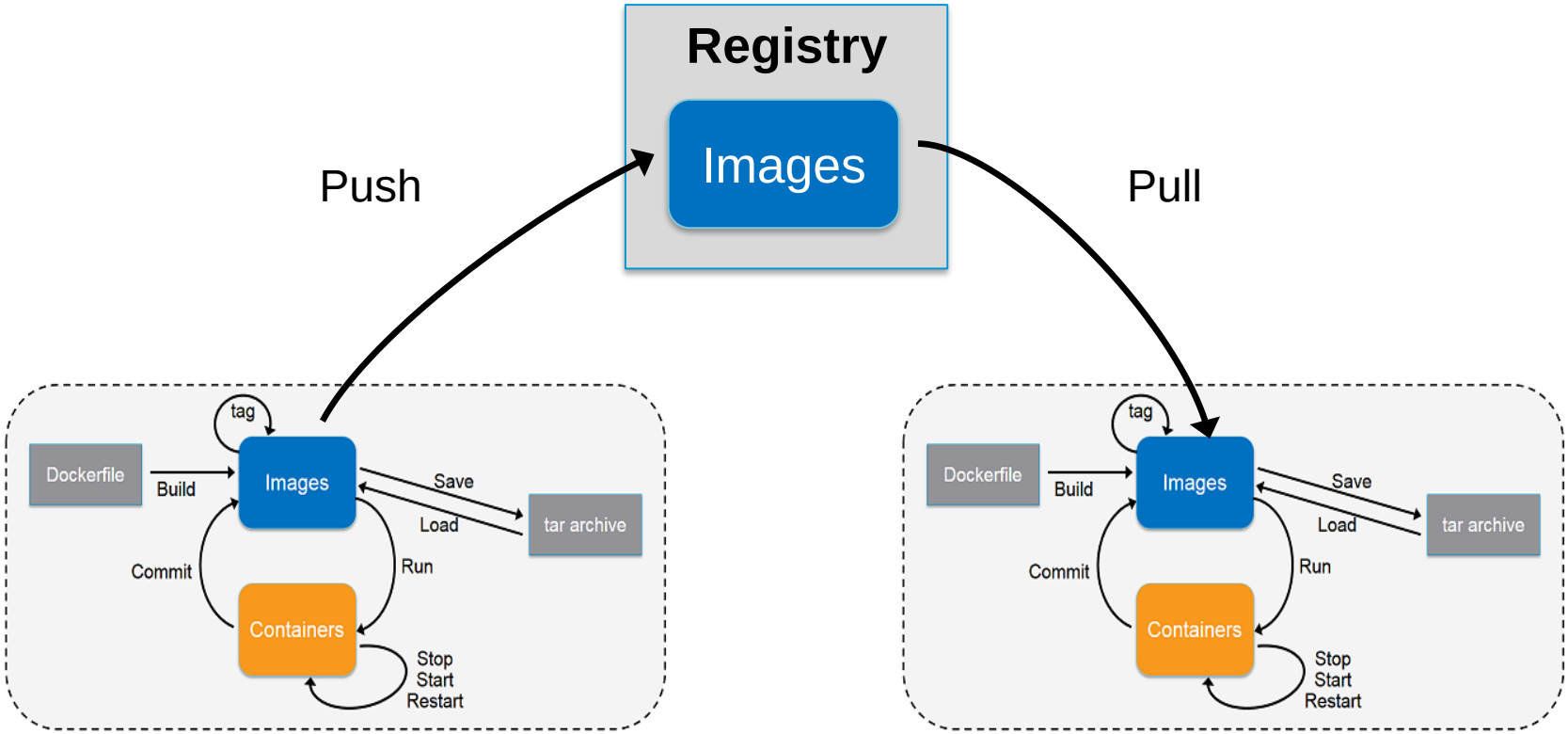
1. Container Image Basics
2. Project Harbor Introduction
3. Consistency of Images
4. Security
5. Image Distribution
6. High Availability of Registry

Lifecycle of Containers and Images



Registry - Key Component to Manage Images

- Repository for storing images
- Intermediary for shipping and distributing images
- Ideal for access control and other image management



Agenda

1. Container Image Basics
2. Project Harbor Introduction
3. Consistency of Images
4. Security
5. Image Distribution
6. High Availability of Registry

Project Harbor

- An open source enterprise-class registry server.
- Initiated by VMware China, adopted by users worldwide.
- Apache 2 license.
- https://github.com/vmware/harbor/

Key Features

- User management & access control
  - RBAC: admin, developer, guest
  - AD/LDAP integration
- Policy based image replication
- Web UI
- Audit and logs
- Restful API for integration
- Lightweight and easy deployment

Users, Partners and Developers

HARBOR™

- **Users**

Downloads

Stars

Users

- **Developers**

Forks

Contributors

Partners

vmware®

Harbor Architecture

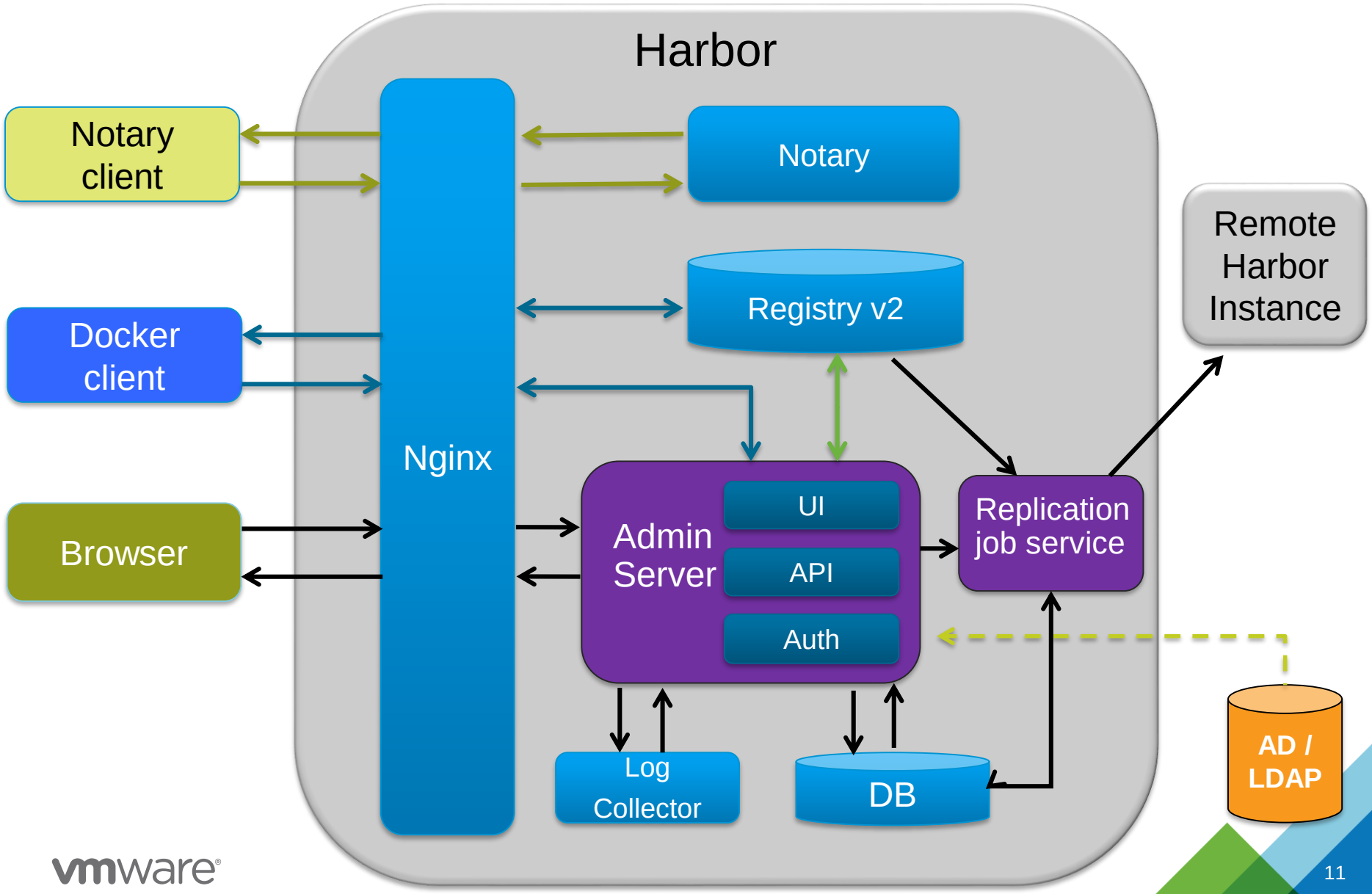
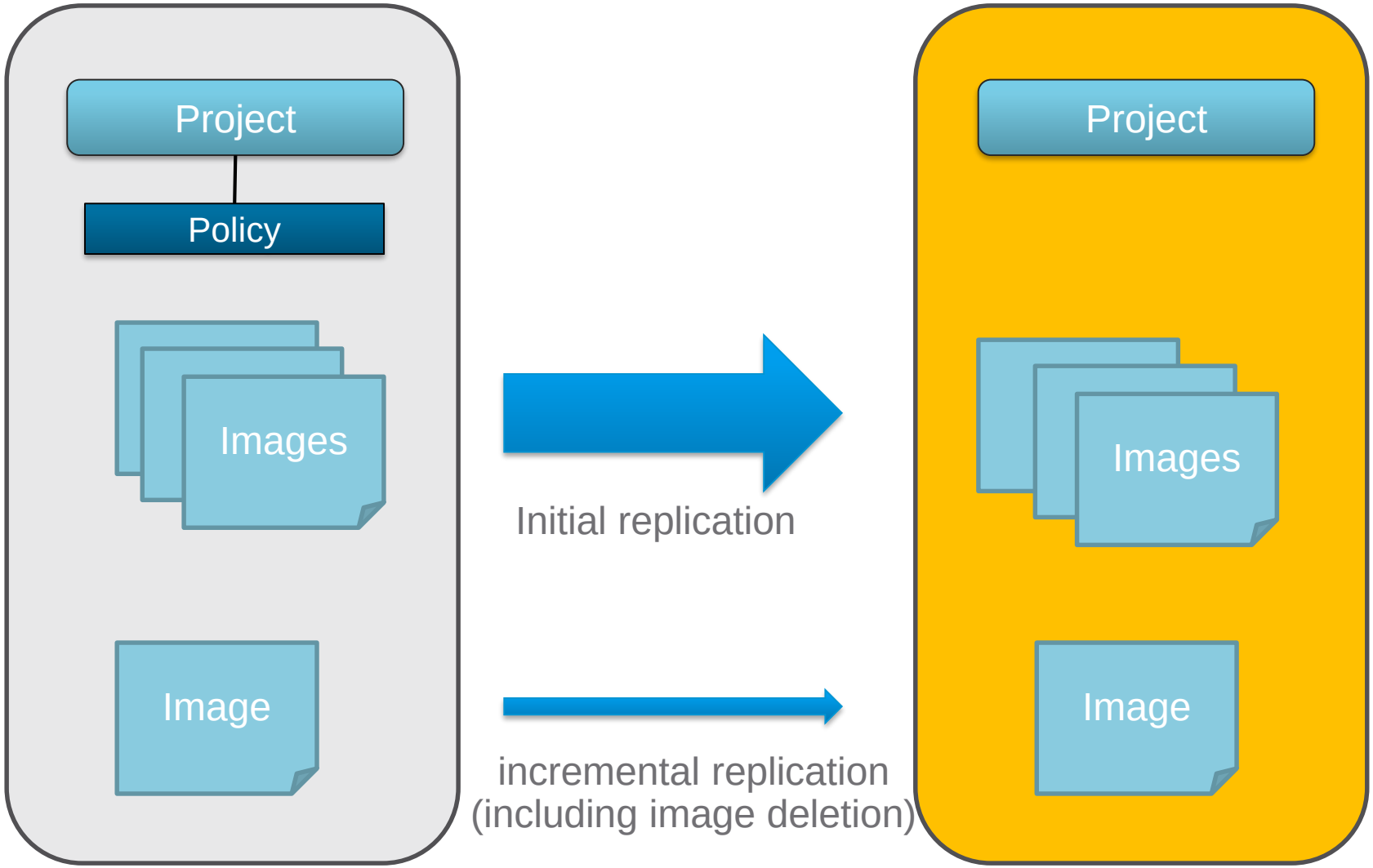


Image replication (synchronization)



Agenda

1. Container Image Basics
2. Project Harbor Introduction
3. Consistency of Images
4. Security
5. Image Distribution
6. High Availability of Registry

Consistency of Container Images

- Container images are used throughout the life cycle of software development
  - Dev
  - Test
  - Staging
  - Production
- Consistency must be maintained
  - Version control
  - Issue tracking
  - Troubleshooting
  - Auditing

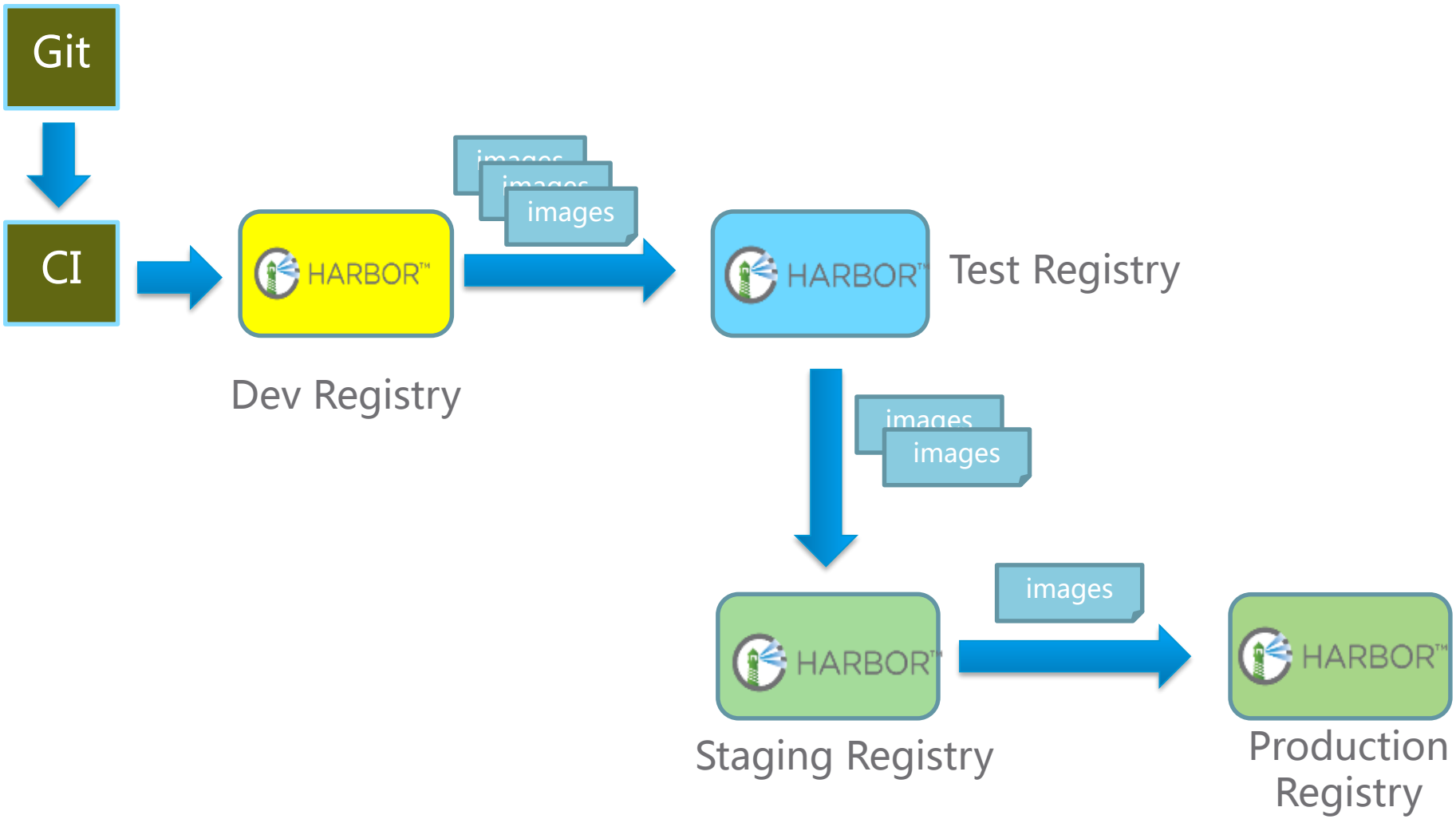
Same Dockerfile Always Builds Same Image?

Example:

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

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

Shipping Images in Binary Format for Consistency



Images are synchronized between environments by using Harbor registry.

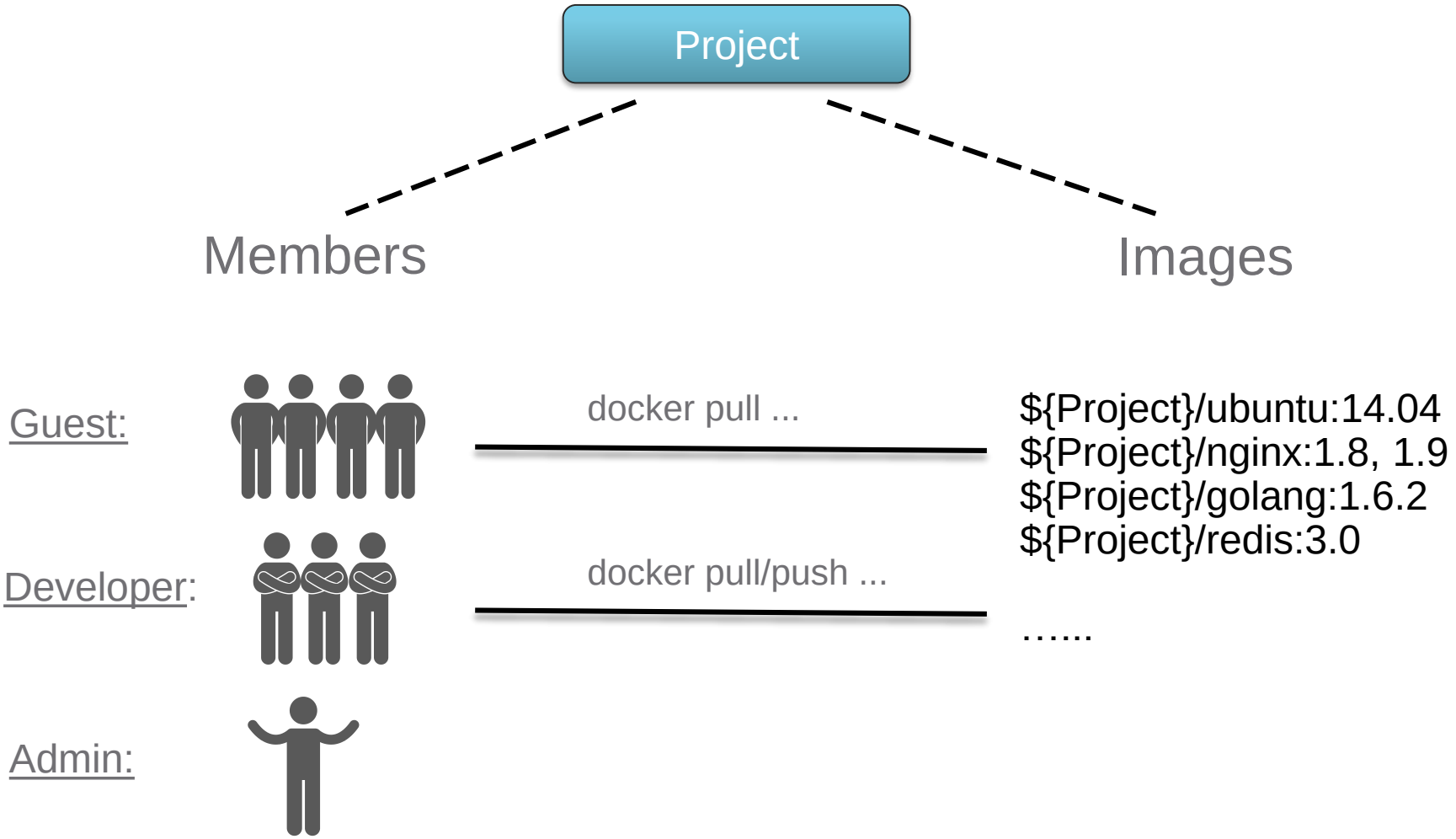
Agenda

1. Container Image Basics
2. Project Harbor Introduction
3. Consistency of Images
4. Security
5. Image Distribution
6. High Availability of Registry

Access Control to Images

- Organizations often keep images within their own organizations
  - Intellectual property stays in organization
  - Efficiency: LAN vs WAN
- People with different roles should have different access
  - Developer – Read/Write
  - Tester – Read Only
- Different environments should enforce different rules
  - Dev/test env – many people can access
  - Production – a limited number of people can access
- Can be integrated with internal user management system
  - LDAP/Active Directory

Example: Role Based Access Control in Harbor



Other security considerations

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

Agenda

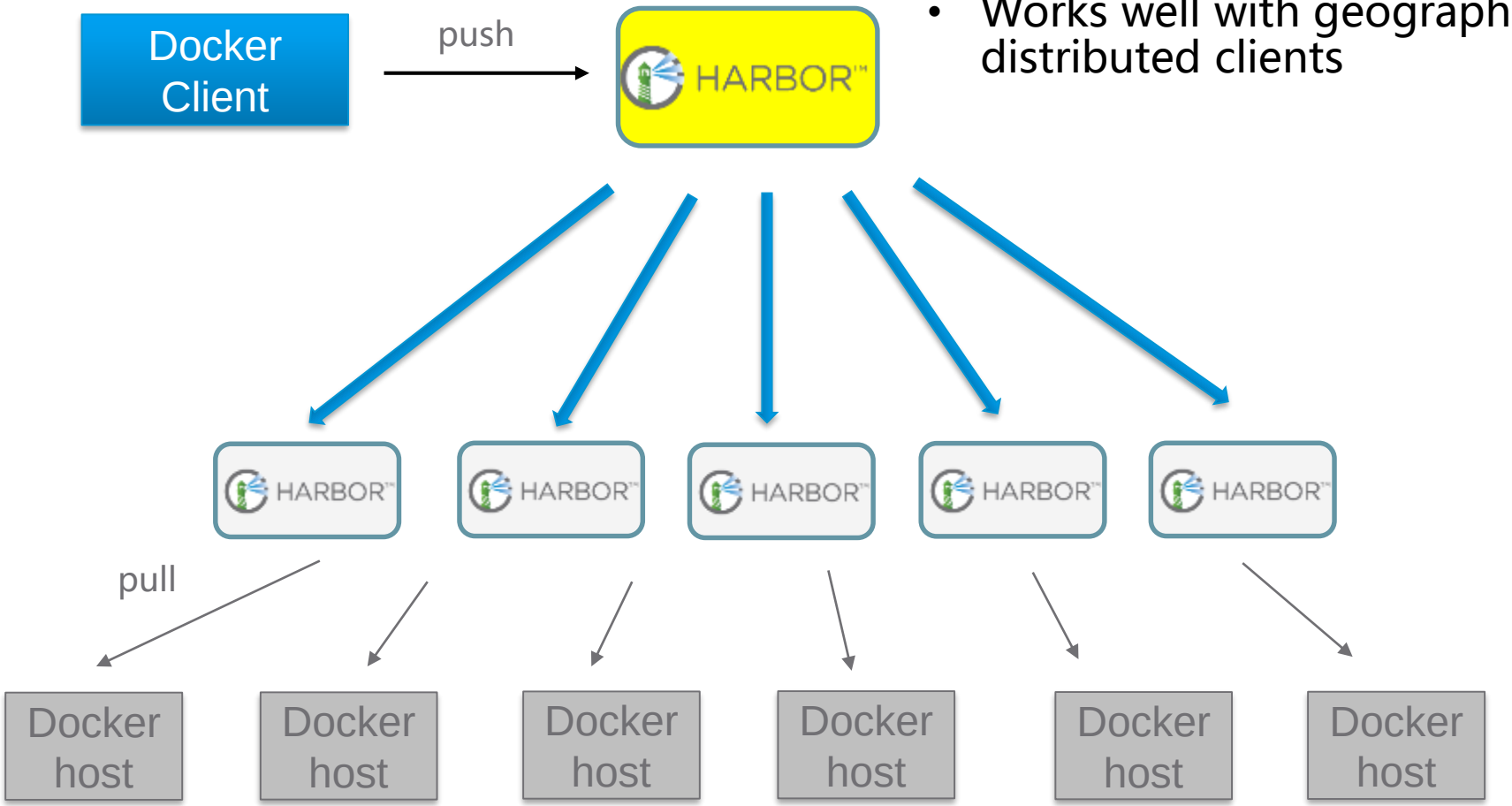
1. Container Image Basics
2. Project Harbor Introduction
3. Consistency of Images
4. Security
5. Image Distribution
6. High Availability of Registry

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
  - Multiple instances of independent registry sharing no storage

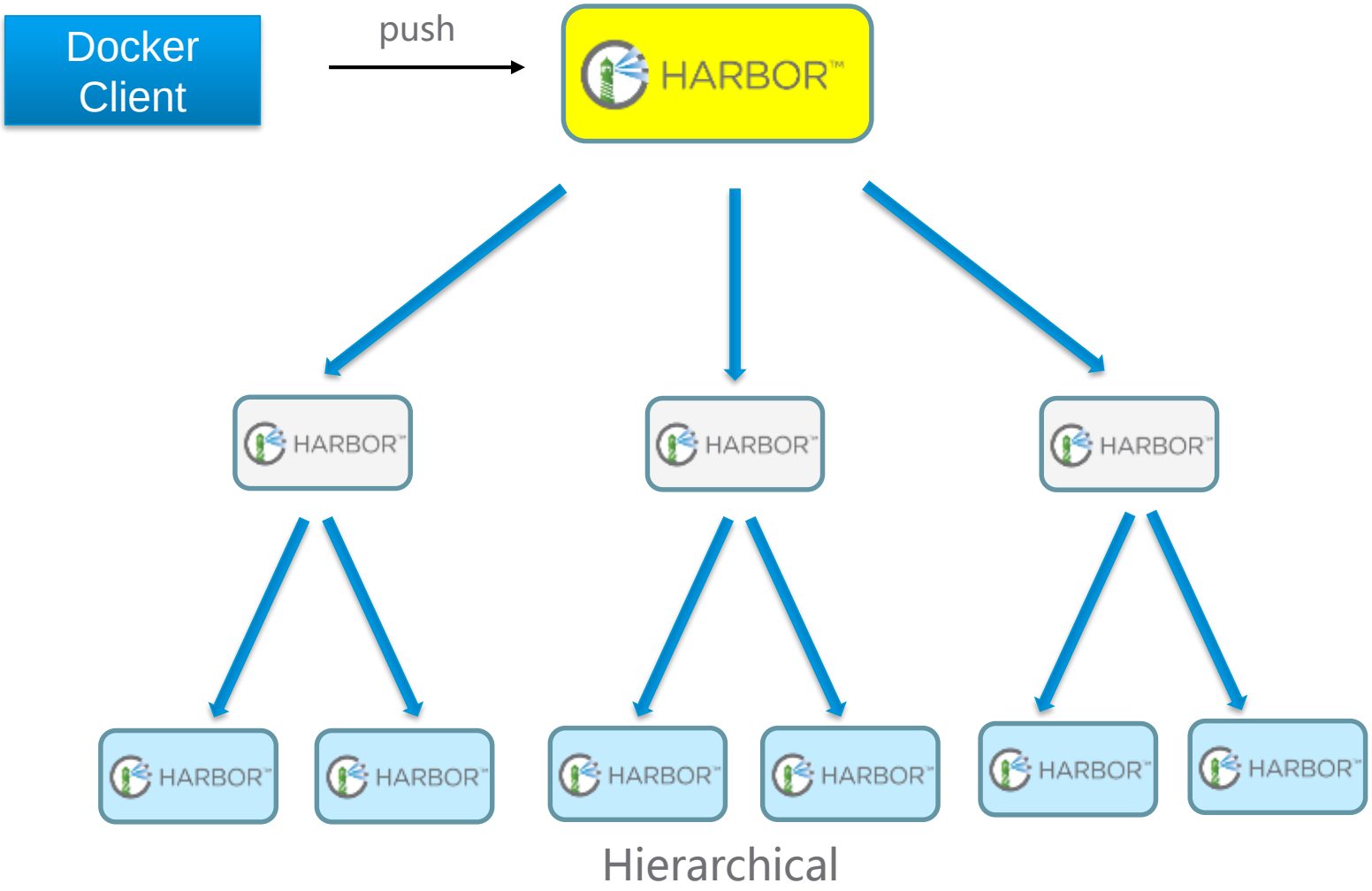
Image Distribution via Master-Slave Replication

- Load balancing
- Works well with geographically distributed clients



Master – Slave model

Hierarchical Image Distribution



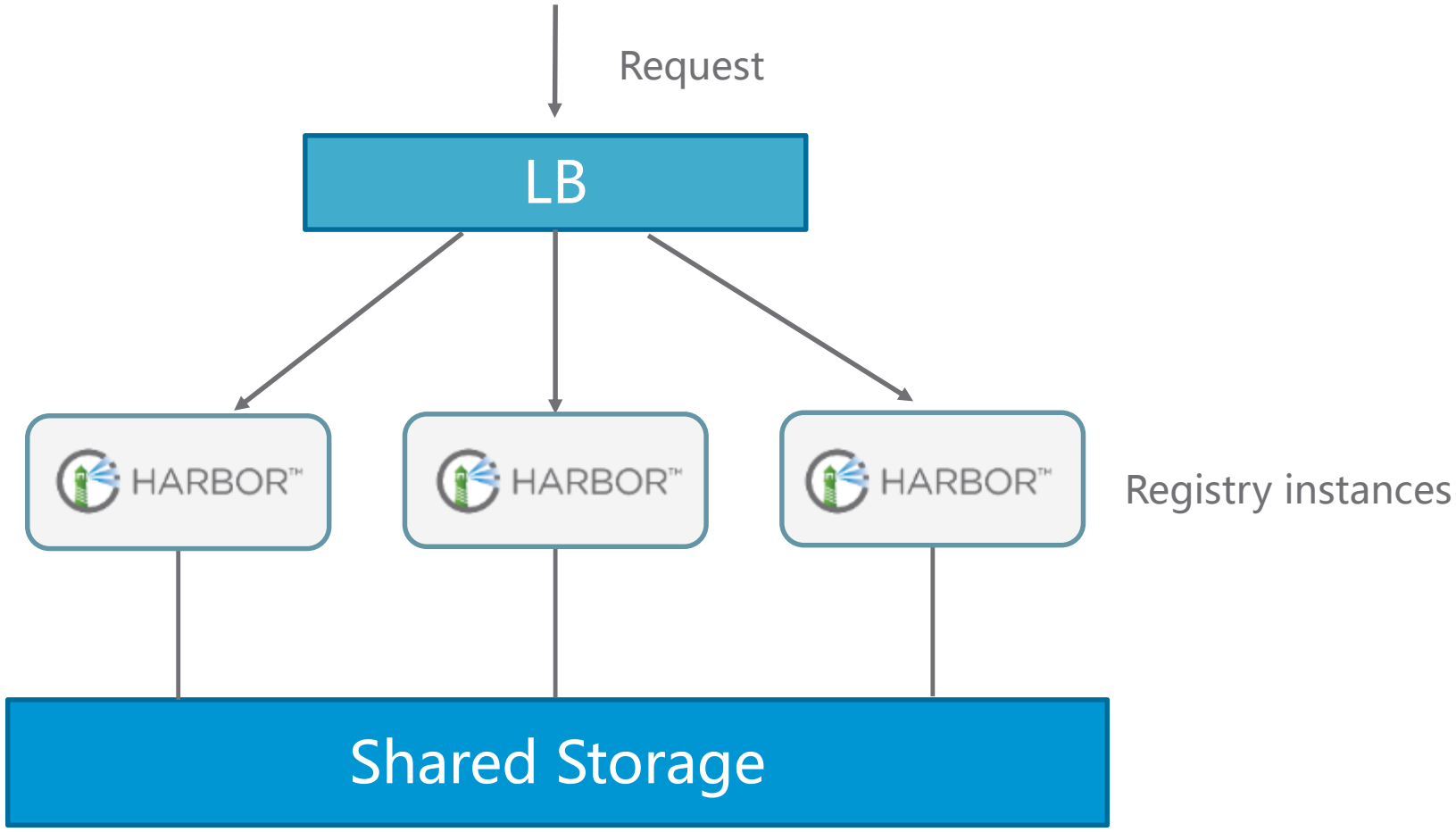
Agenda

1. Container Image Basics
2. Project Harbor Introduction
3. Consistency of Images
4. Security
5. Image Distribution
6. High Availability of Registry

High Availability of Registry

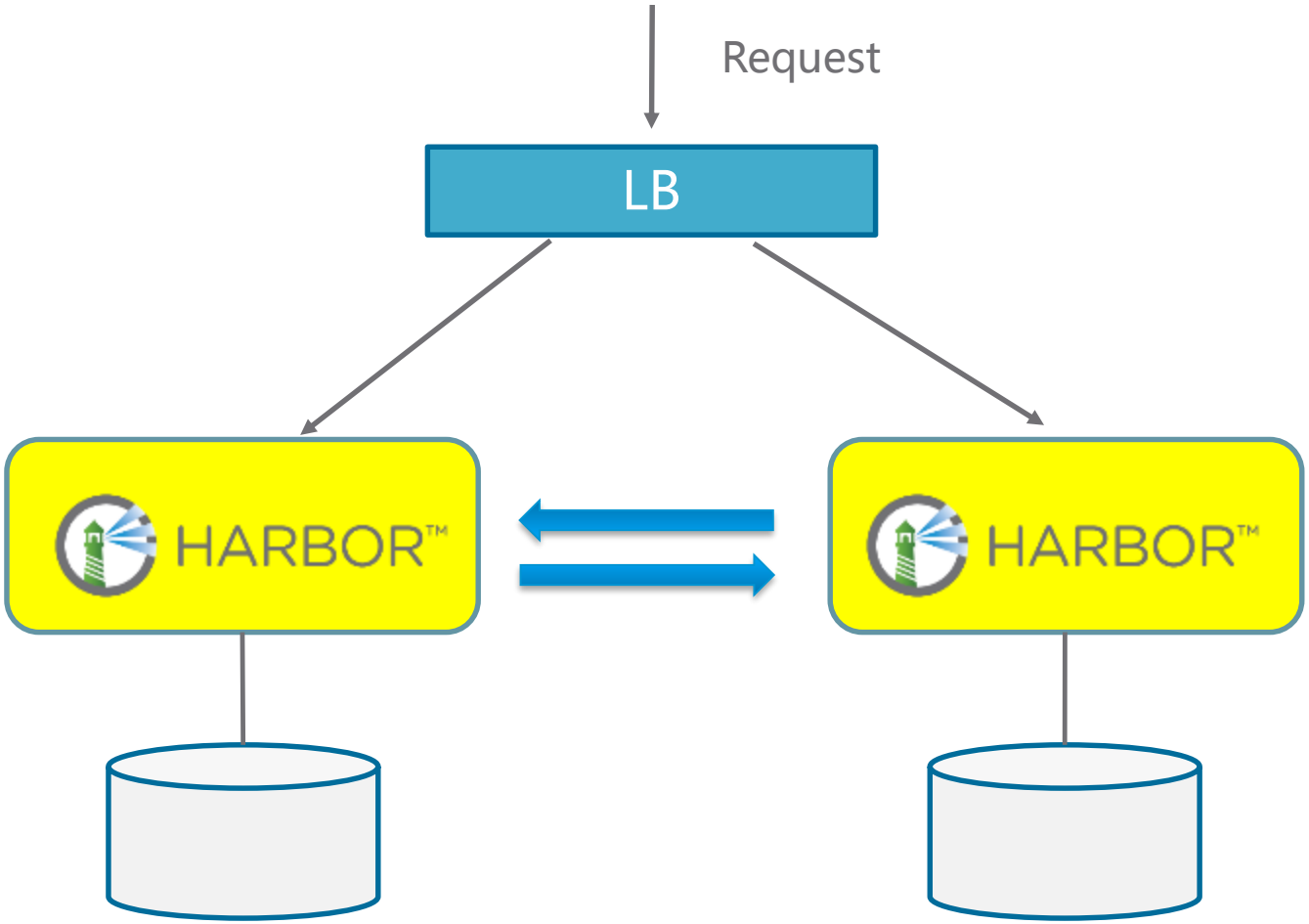
- To remove single point of failure on registry
- Three models to achieve HA
  - Shared storage
  - Replication (no shared storage)
  - Using other HA platform

Registries using Shared Storage

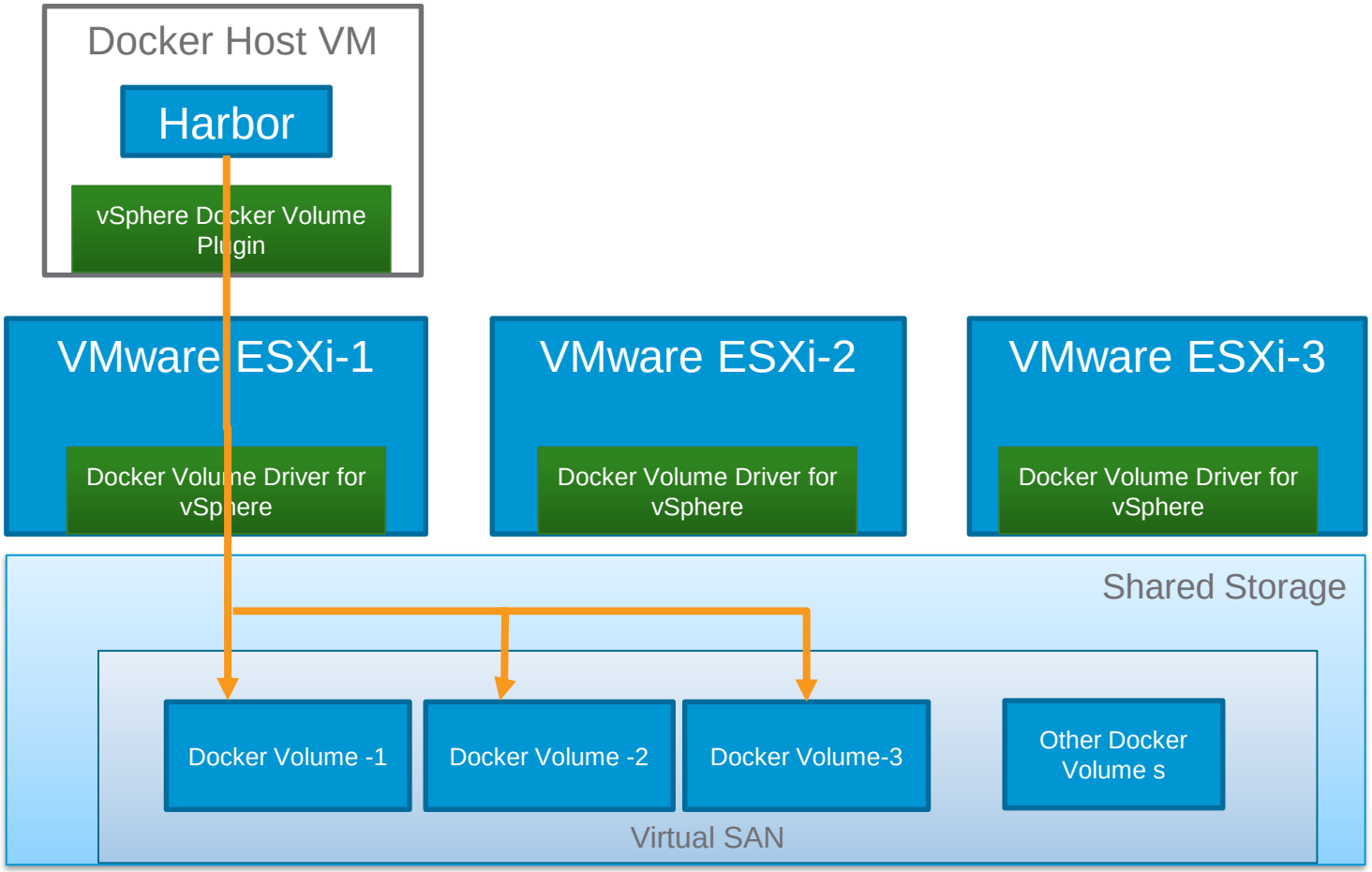


vmware

Image replication between registries

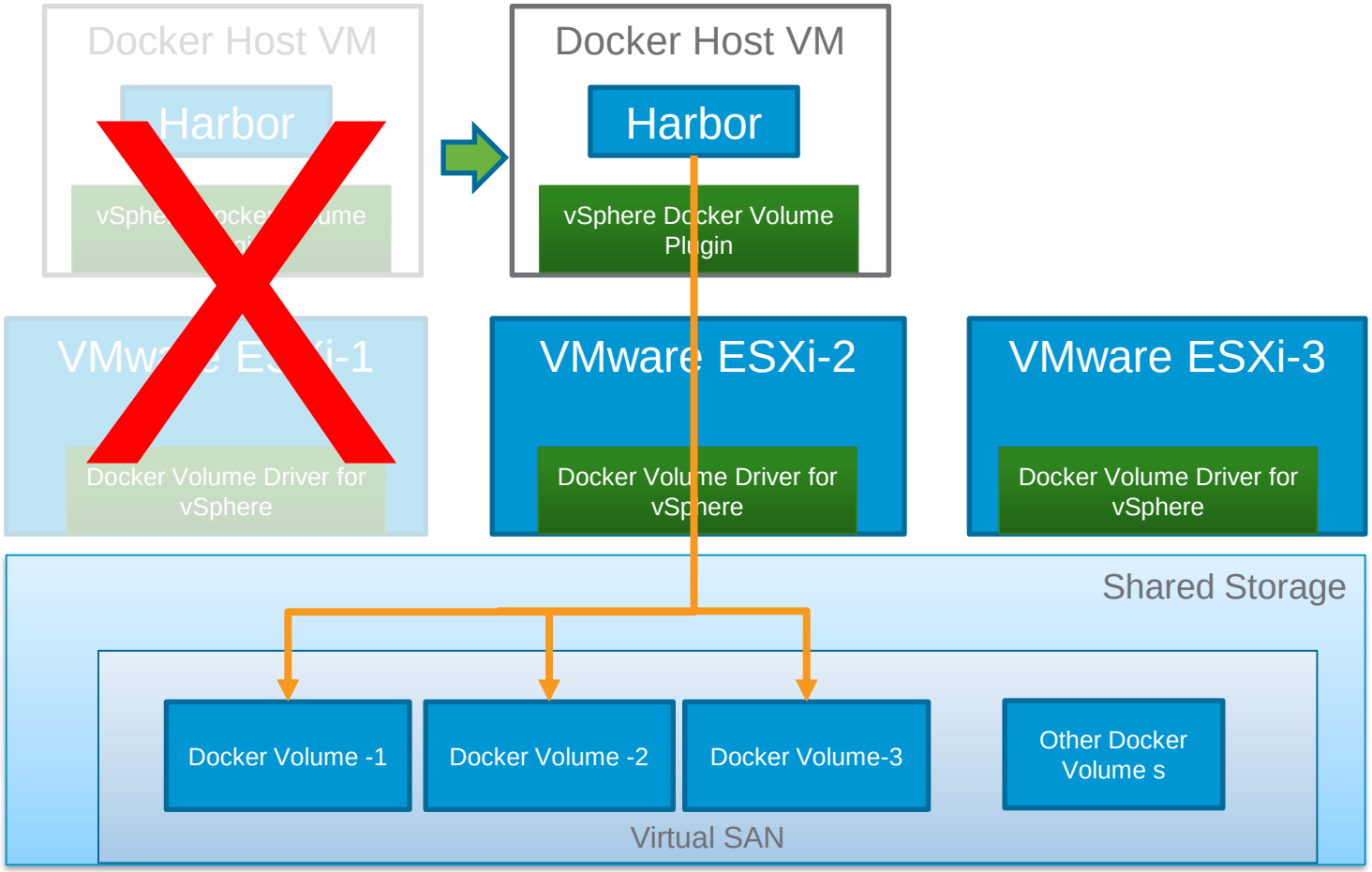


Registry HA on vSphere



vmware

Registry HA on vSphere



Summary

- Container image is the static part of container lifecycle
- Registry is the key component to manage images
- Organizations usually need a private registry
  - Security
  - Efficiency

Thank you!

https://github.com/vmware/harbor

vmware®