

# 采用Harbor开源企业级Registry实现高效安全的镜像运维

张海宁

VMware中国研发中心技术总监

vmware®

© 2017 VMware Inc. All rights reserved.

# 自我介绍

- VMware中国研发先进技术中心首席架构师、技术总监
- Harbor开源企业级容器Registry项目创始人
- Cloud Foundry中国社区最早技术布道师之一
- 多年全栈工程师
- 《区块链技术指南》、《软件定义存储》作者之一



亨利笔记



《区块链技术指南》



《软件定义存储》

# 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

---

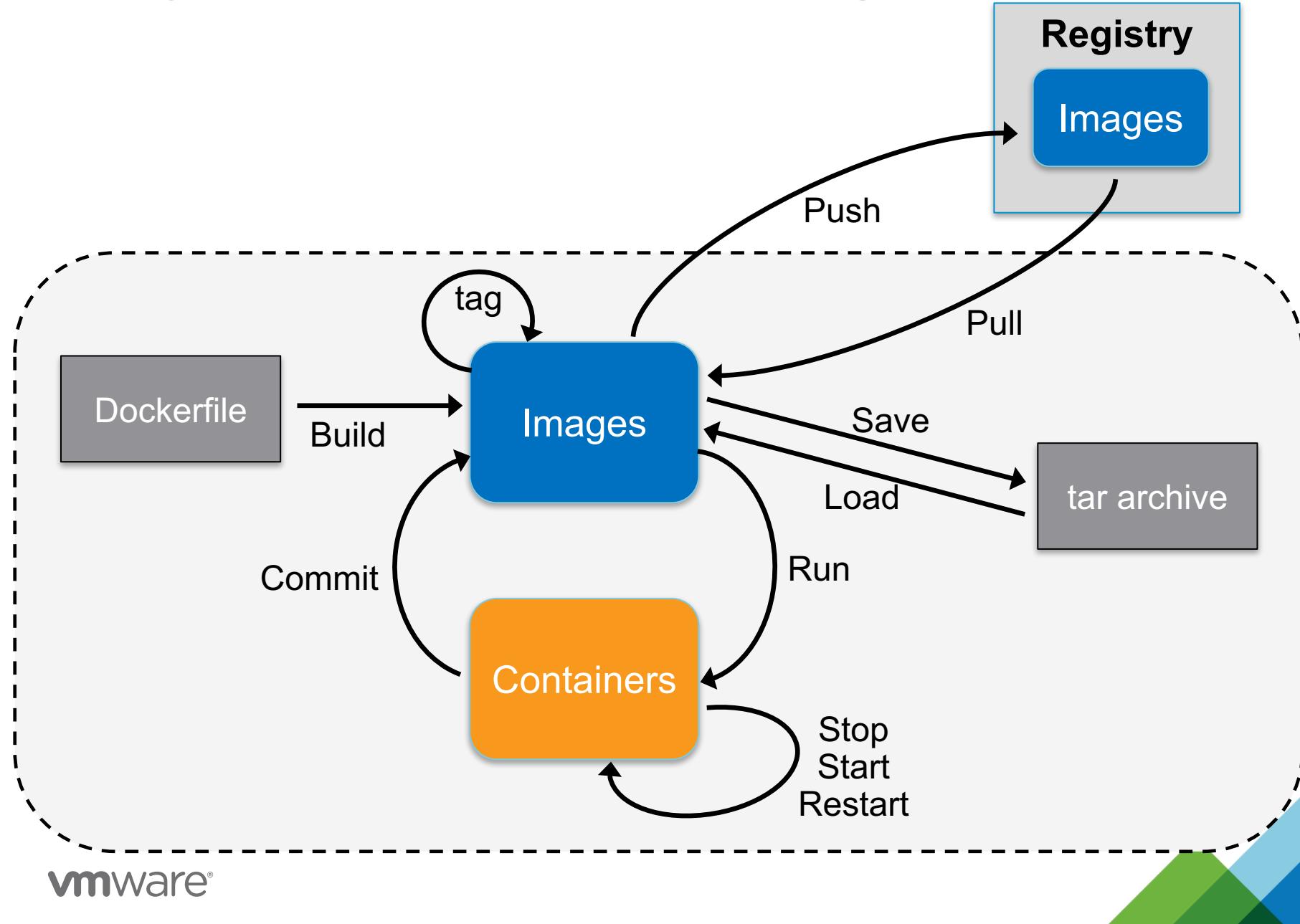
5 Image Distribution

---

6 High Availability of Registry

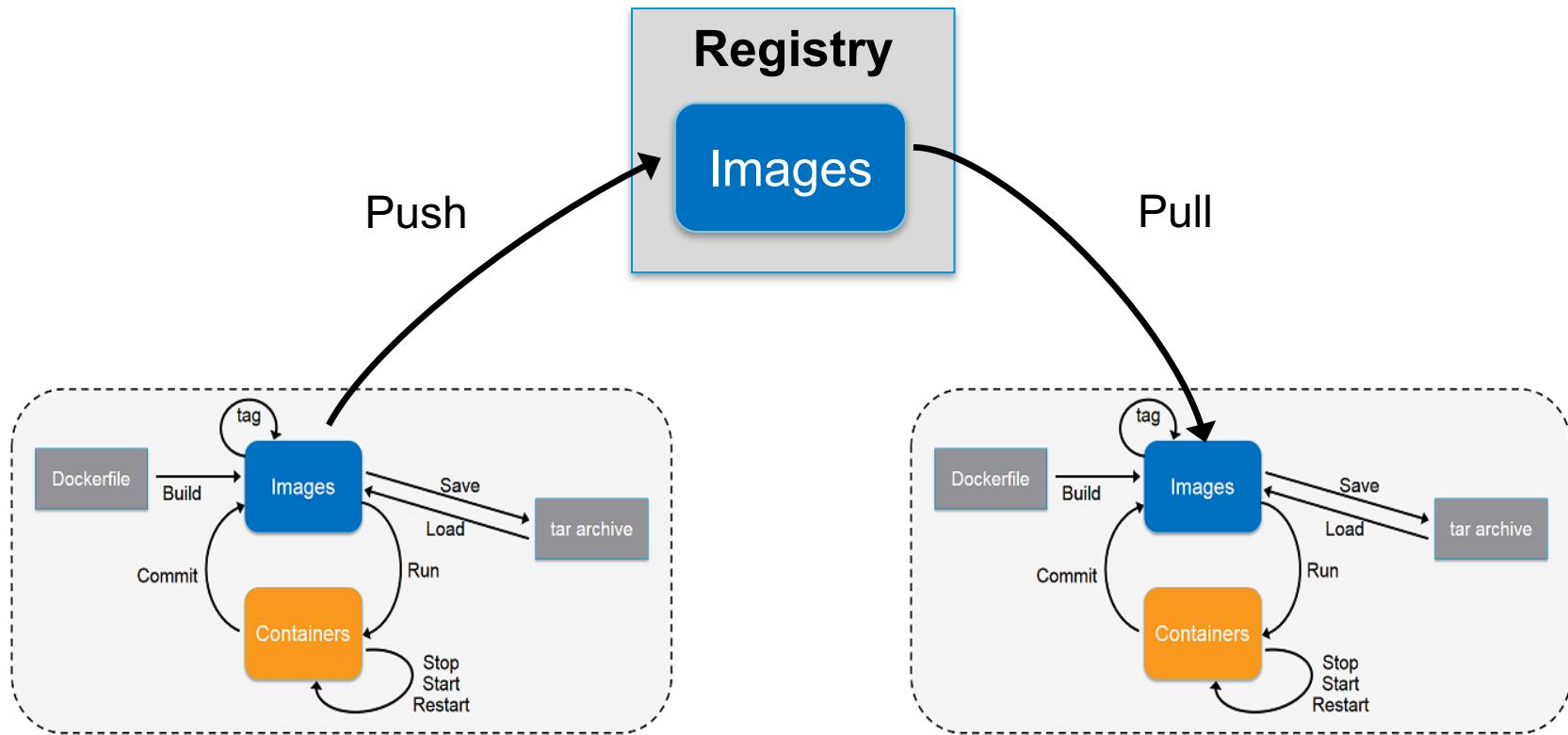


# Lifecycle of Containers and Images



# 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.
- Integrated into vSphere Integrated Containers.
- 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
- Vulnerability Scanning
- Notary
- Web UI
- Audit and logs
- Restful API for integration
- Lightweight and easy deployment

A screenshot of the Harbor web interface. The top navigation bar shows "vm Harbor" and a search bar. On the left, a sidebar menu includes "Projects" (which is selected), "Logs", and "Administration" (which is expanded, showing "Users", "Replication", and "Configuration"). The main content area is titled "Projects" and displays a table of projects. The table has columns for "Project Name", "Public", "Role", "Repositories Count", and "Creation Time". There are 19 total projects listed, with 3 public and 16 private. The table includes rows for "aaabbb", "abbbb", "akshay", "alpha", "foo", "fooasdfasdf", "foobar", and "foobar1".

Project Name	Public	Role	Repositories Count	Creation Time
aaabbb	Public		0	3/22/2017, 3:59 PM
abbbb	Private		0	3/28/2017, 10:24 AM
akshay	Private	Project Admin	0	3/3/2017, 2:51 AM
alpha	Private	Project Admin	0	3/10/2017, 1:32 AM
foo	Private	Project Admin	0	2/28/2017, 8:17 AM
fooasdfasdf	Private		0	3/25/2017, 2:07 AM
foobar	Private	Project Admin	0	3/2/2017, 3:52 AM
foobar1	Private	Project Admin	0	3/25/2017, 1:45 AM

# Users and Developers

- **Users**



Downloads



Stars



Users

- **Developers**



Forks

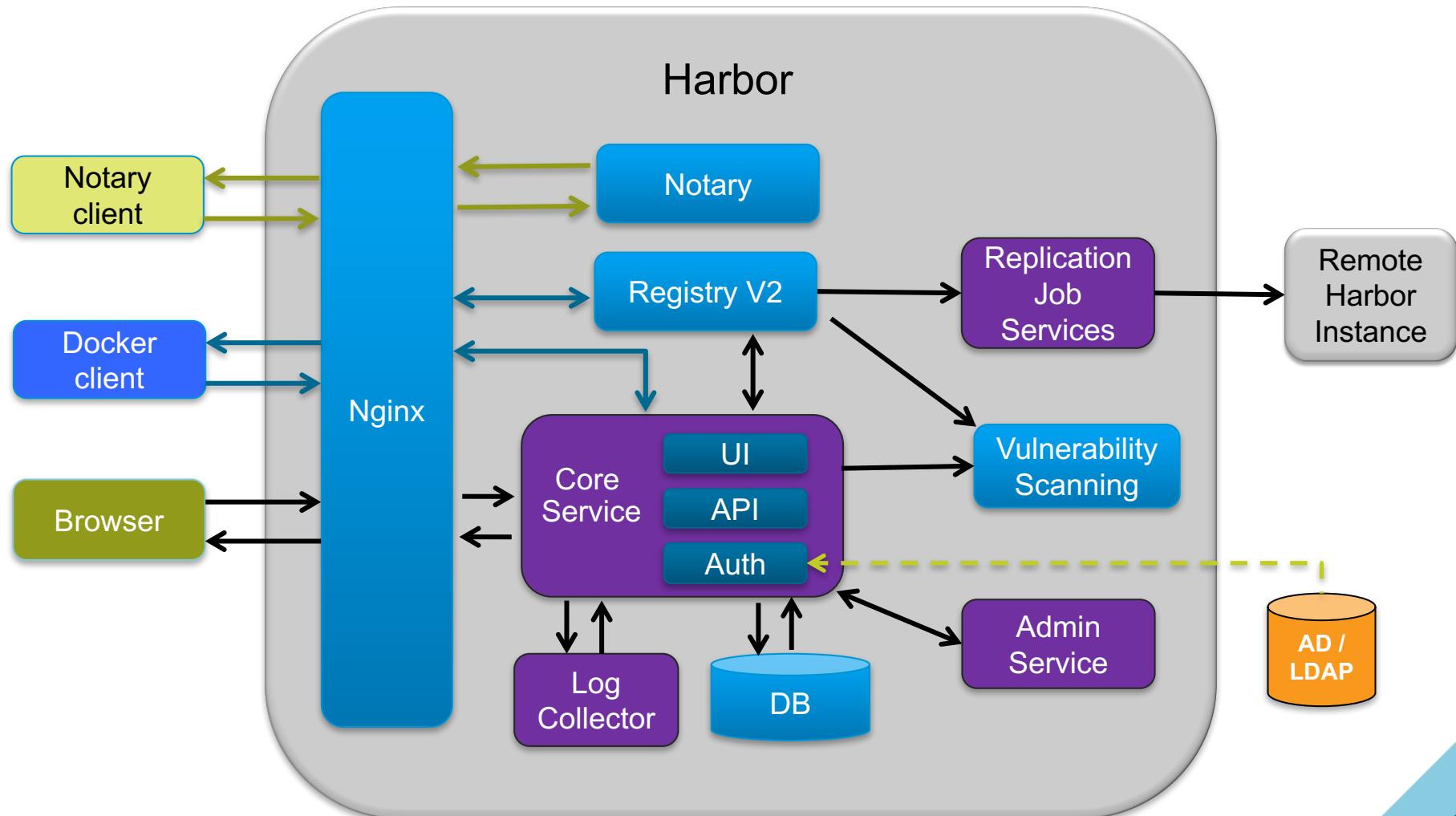


Contributors



Partners

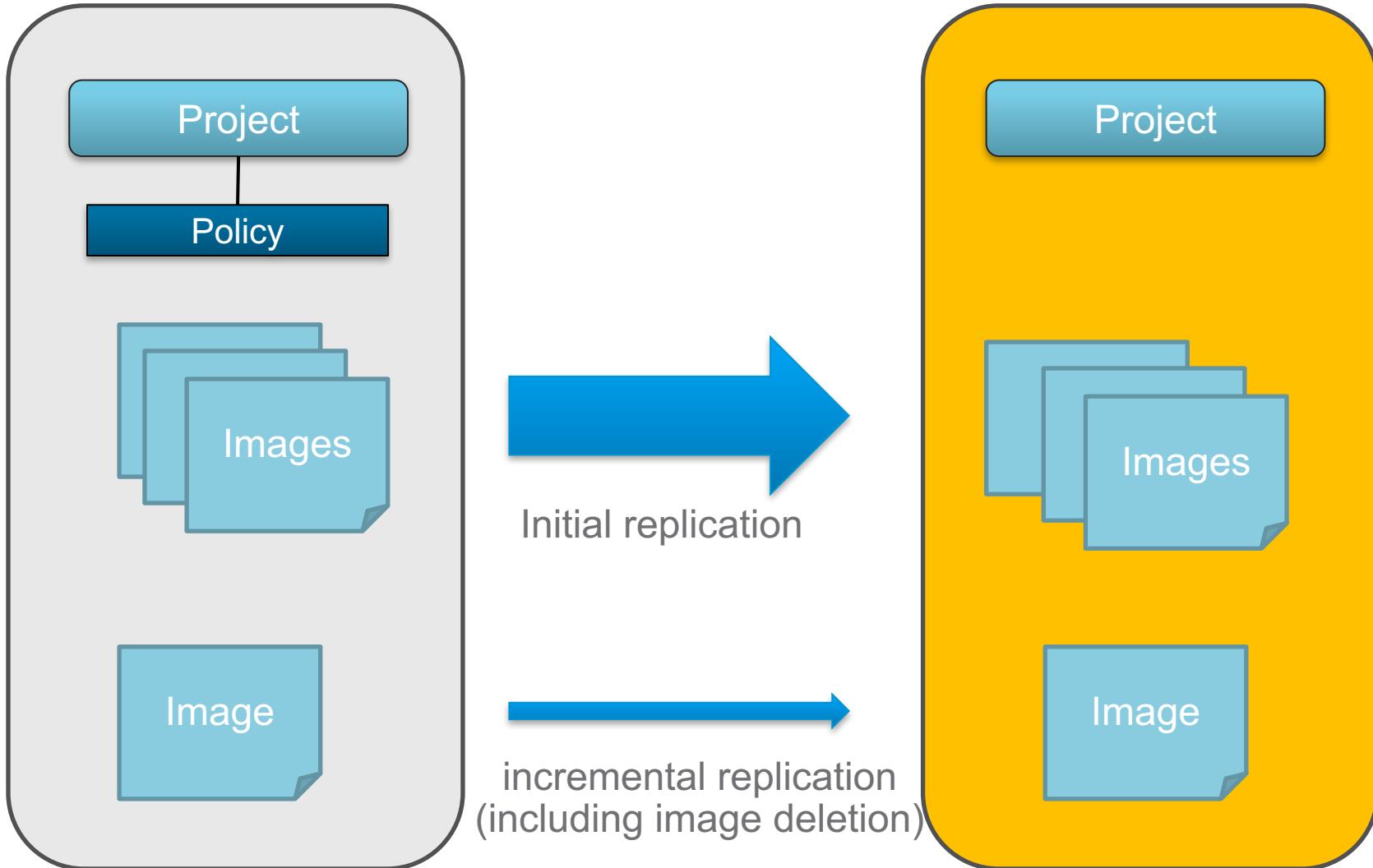
# Harbor Architecture



# Harbor users and partners (selected)



# 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:

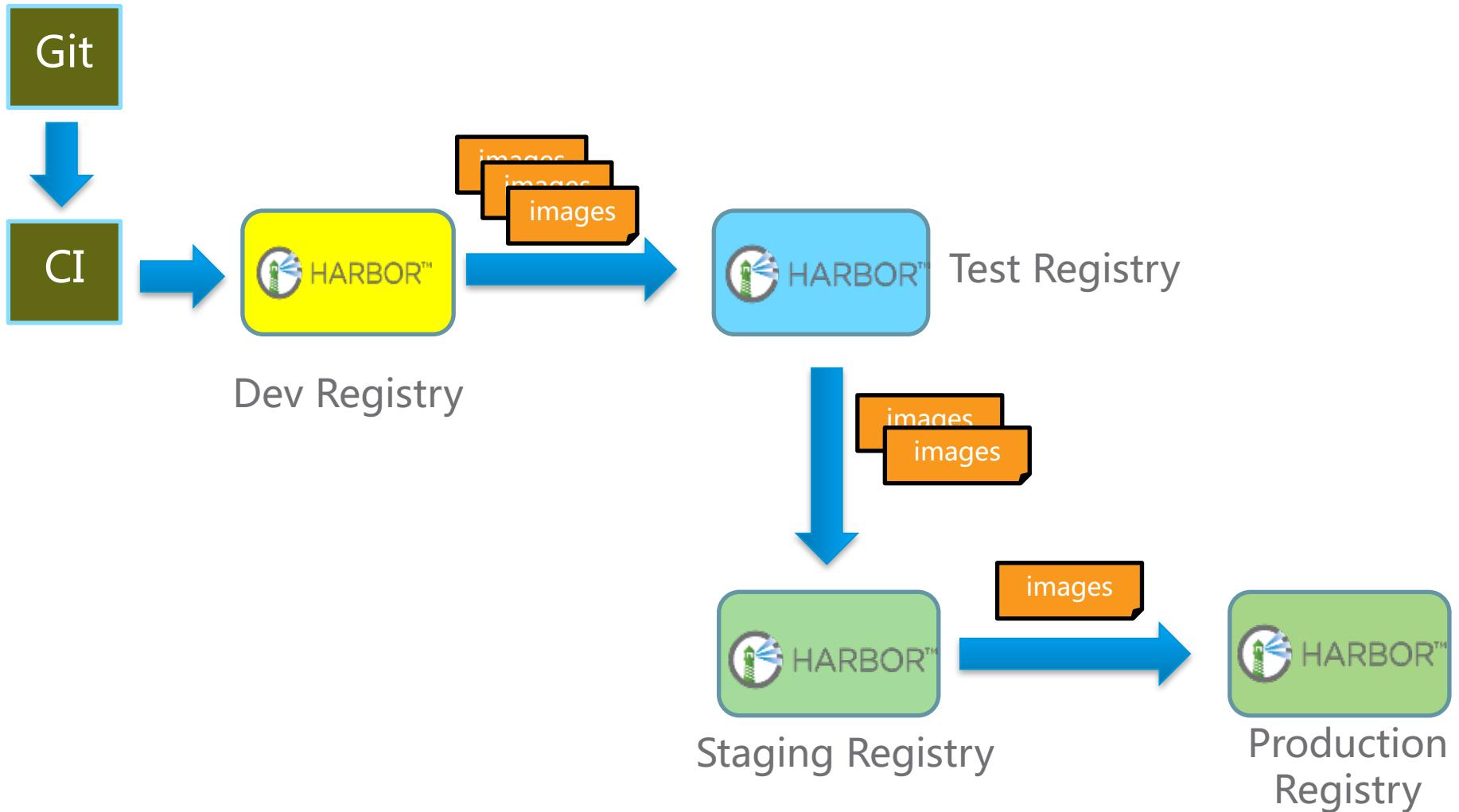
```
FROM ubuntu

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



# Agenda

---

1 Container Image Basics

---

2 Project Harbor Introduction

---

3 Consistency of Images

---

4 Security

---

5 Image Distribution

---

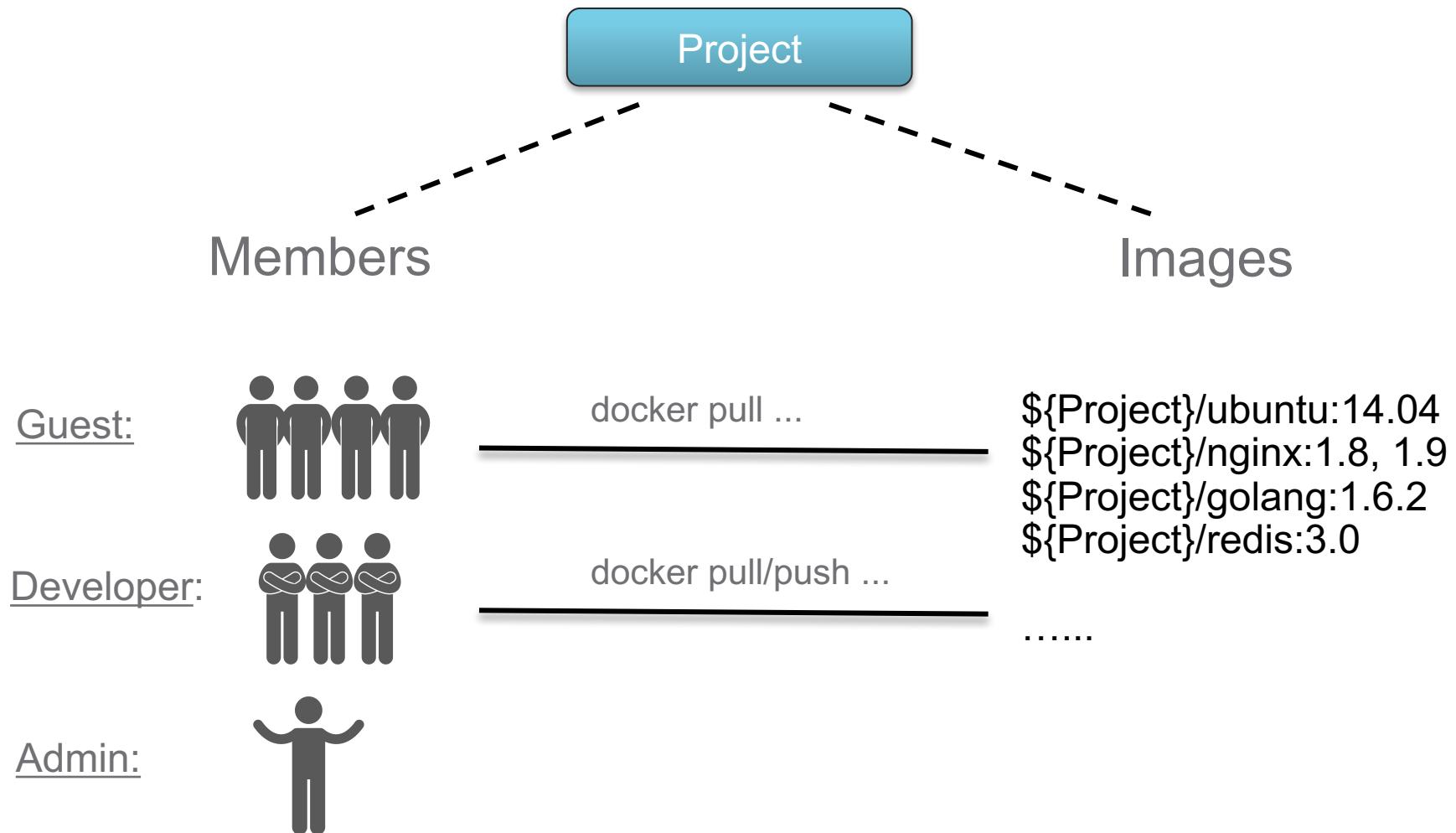
6 High Availability of Registry



# 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 rules should be enforced in different environments
  - 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

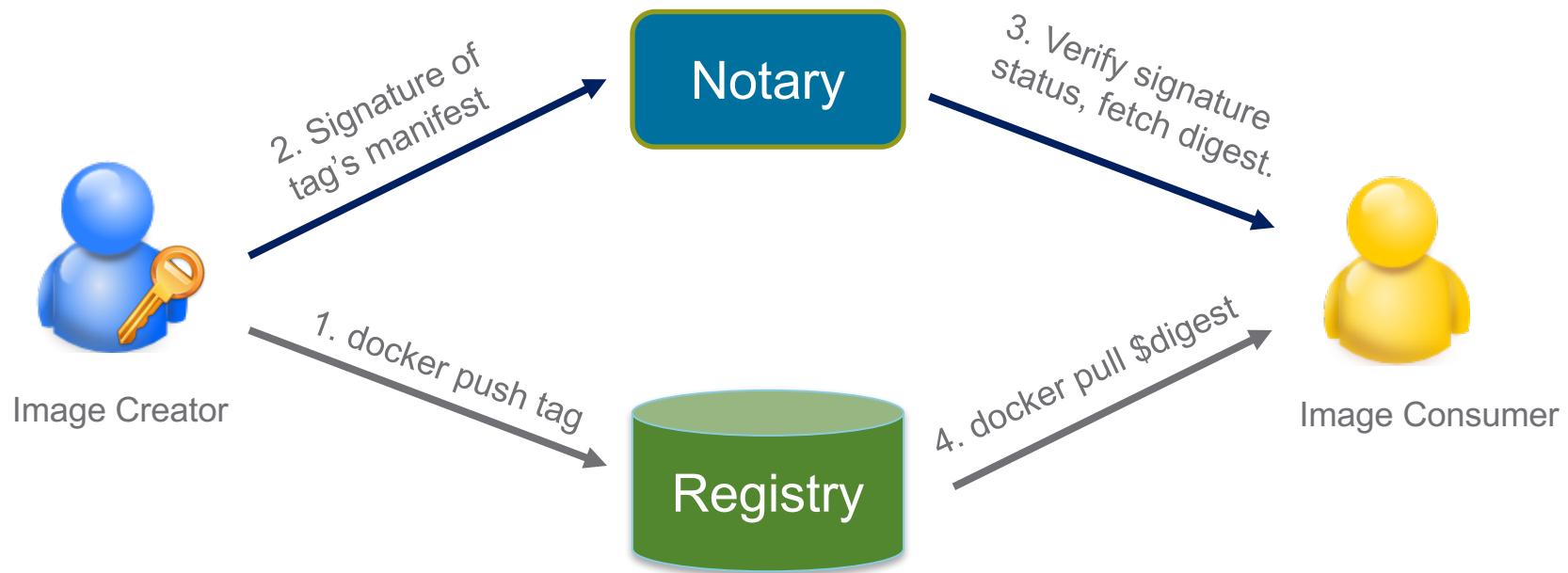
# Example: Role Based Access Control in Harbor



# Other security considerations

- Enable content trust by installing Notary service
  - Image is signed by publisher's private key during pushing
  - Image is 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



# Vulnerability Scanning

- **Static analysis** of vulnerability by inspecting filesystem of container image and indexing features in database.
- **Rescanning** is needed only and only if new detectors are added.
- Update vulnerability data regularly
  - Debian Security Bug Tracker
  - Ubuntu CVE Tracker
  - Red Hat Security Data
  - Oracle Linux Security Data
  - Alpine SecDB

# Registry – Image Vulnerability Scanning

Vulnerability scanning

**Set vulnerability threshold**

**Prevent images from being pulled** if they exceed threshold

**Periodic scanning** based on updated vulnerability database

Project Repositories

The screenshot shows a table of project repositories with columns for Name, Tags, and vulnerability status. A tooltip over the 'vulnerability' column for the 'default-project/ubuntu' repository provides detailed information about the scan results.

	Name	Tags	vulnerability
:	> default-project/redis	1	
:	default-project/ubuntu	1	
⋮	14.04	docker pull 10.160.247.138/default-project/ubuntu:14.04	 ✓ 1/29/2015, 2:37 AM
⋮	> default-project/demo-busybox	2	15 1 - 3 of 3 items

**36 of 126 packages have known vulnerabilities.**

- 6 high
- 22 medium
- 8 low
- 90 none

Scan completed time: 08/09/2017 23:03:19

# Agenda

---

1 Container Image Basics

---

2 Project Harbor Introduction

---

3 Consistency of Images

---

4 Security

---

5 Image Distribution

---

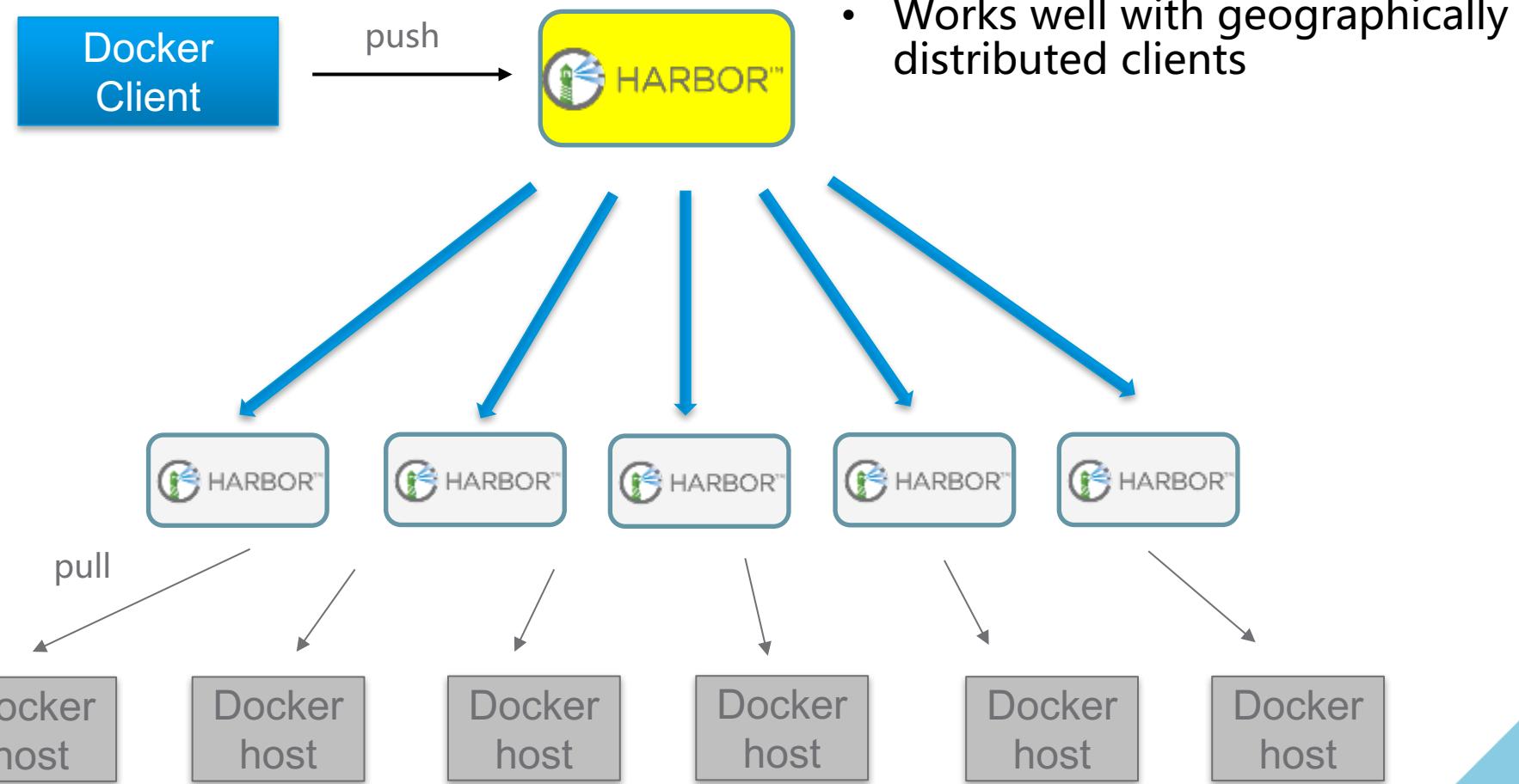
6 High Availability of Registry



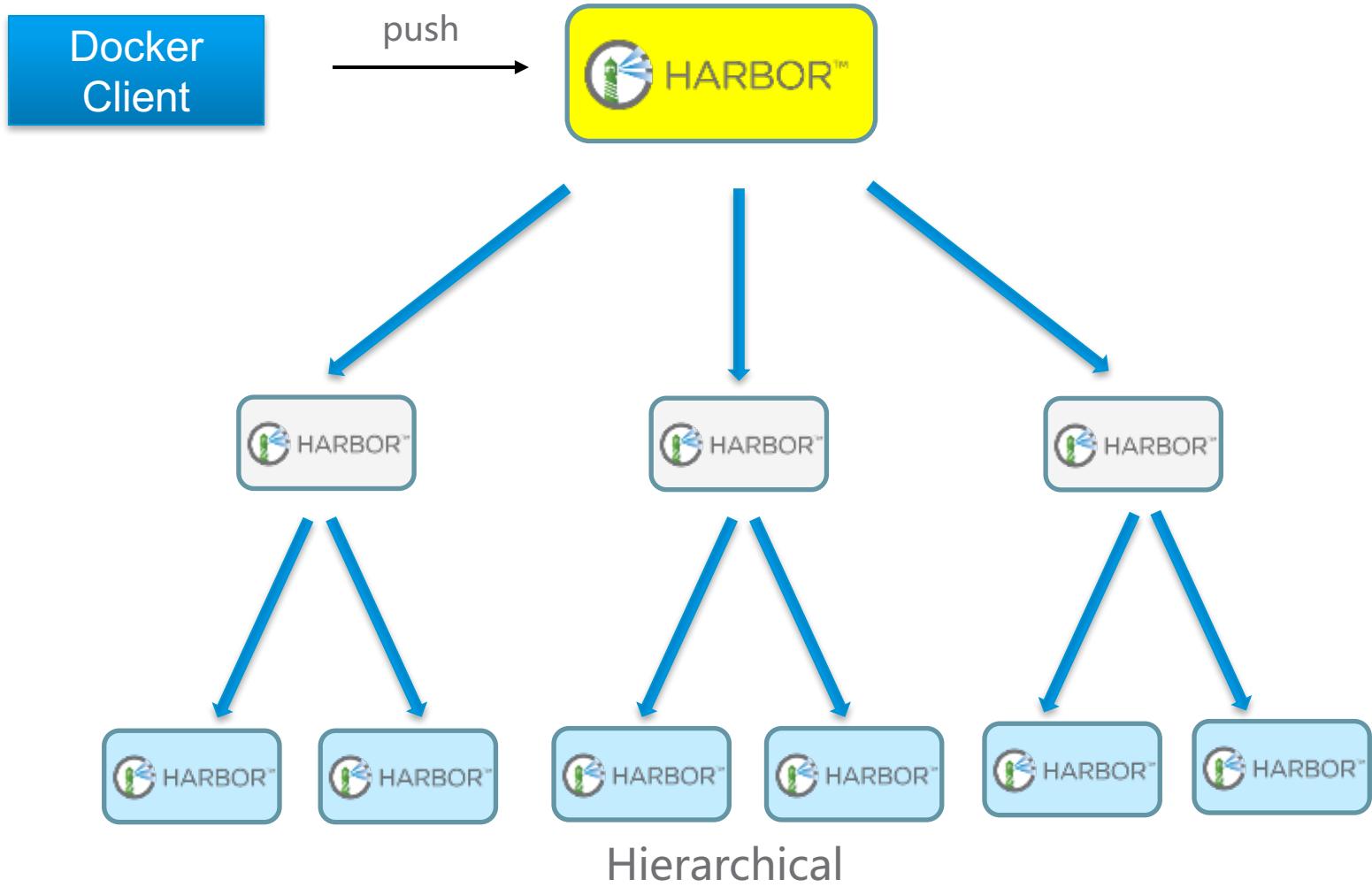
# Image Distribution

- 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



# Hierarchical Image Distribution



# Agenda

---

1 Container Image Basics

---

2 Project Harbor Introduction

---

3 Consistency of Images

---

4 Security

---

5 Image Distribution

---

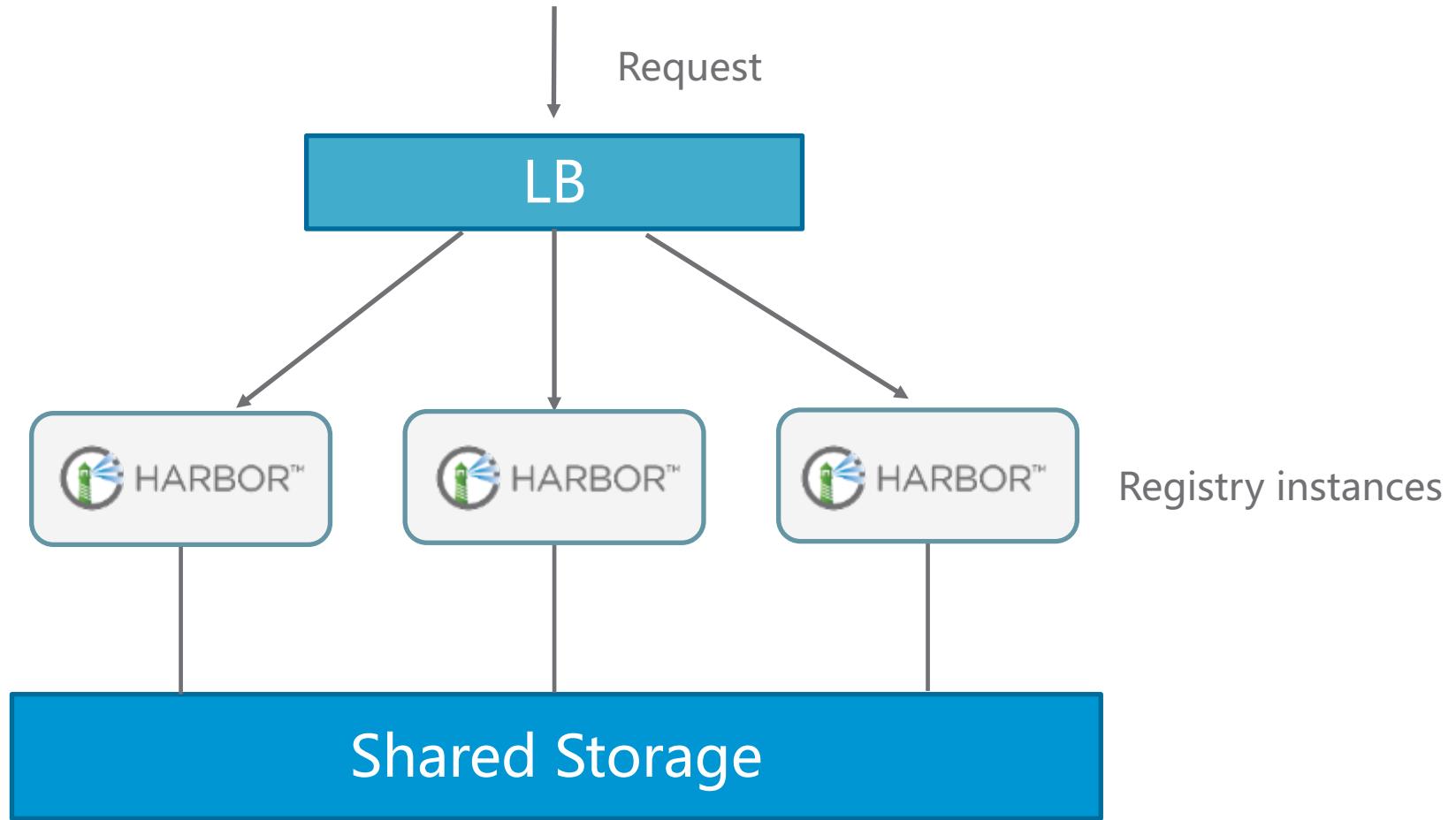
6 High Availability of Registry



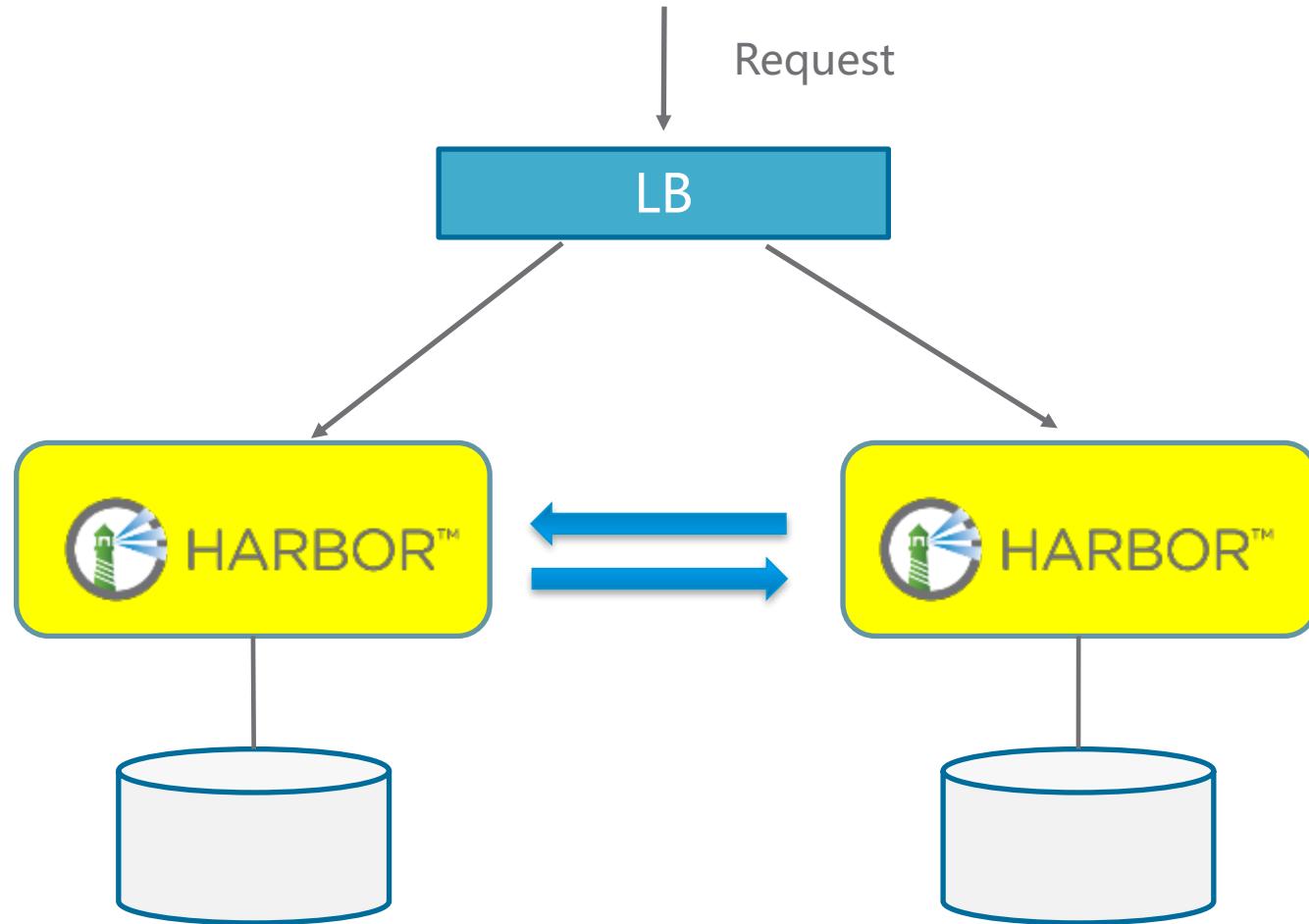
# High Availability of Registry

- 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

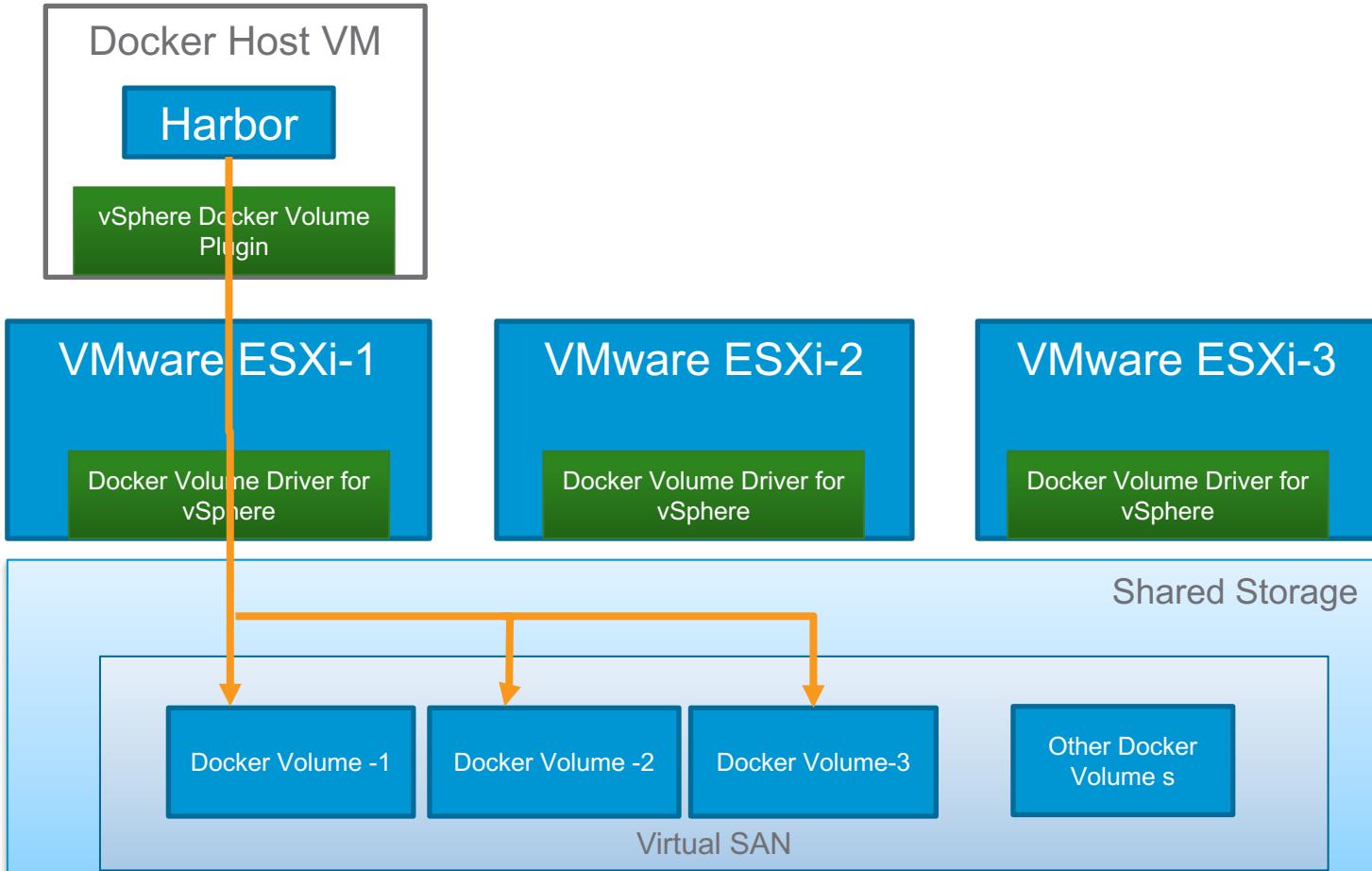


# Image replication between registries



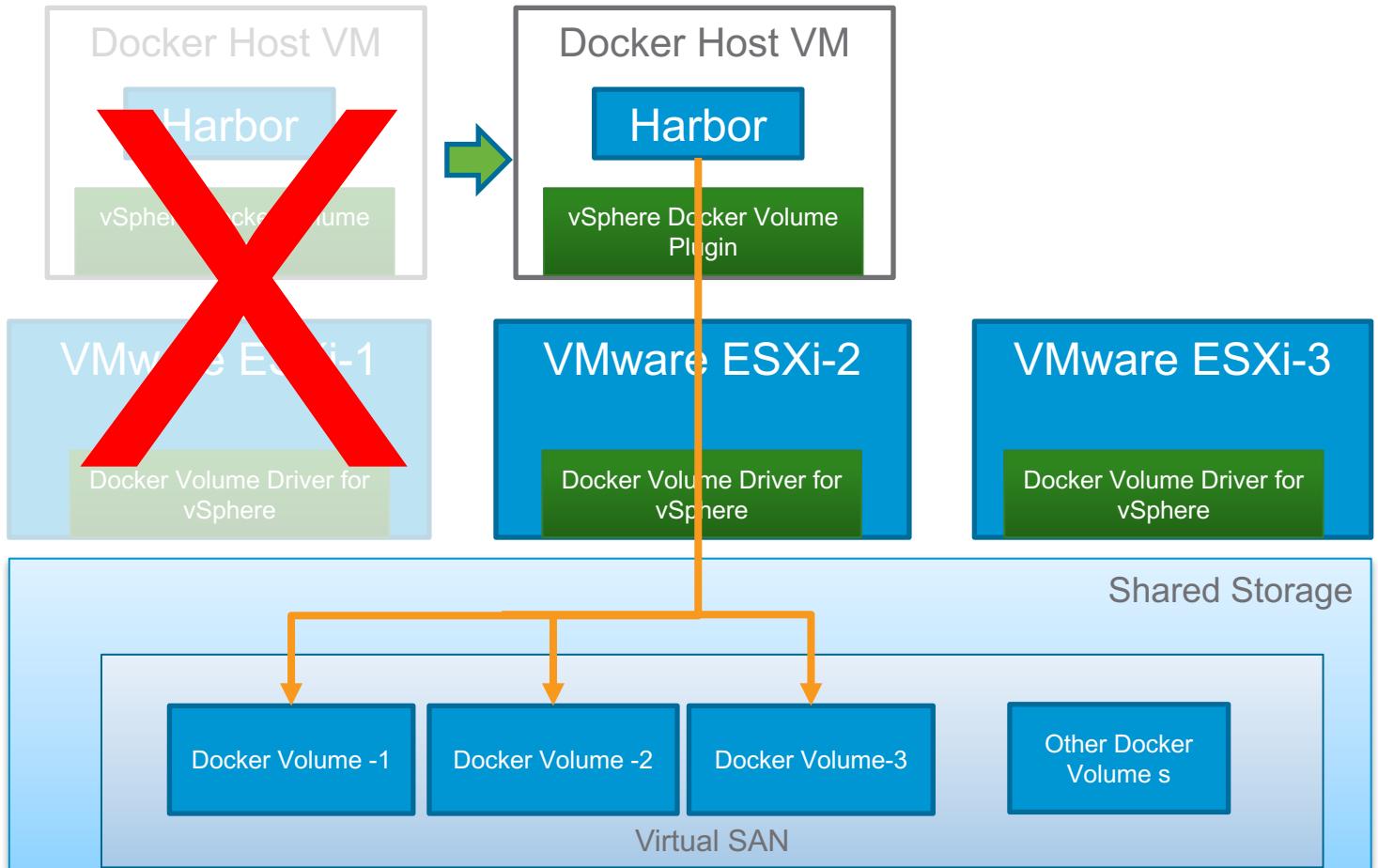
# Registry HA on vSphere

- Registry in a VM protected by vSphere
- Image storage by VSAN Docker Volume



# Registry HA on vSphere

- VM failed over to a healthy host
- Image storage still connected by VSAN



# 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

# Harbor开源项目有奖征文活动

- 您的公司或单位必须是Harbor开源项目v1.1+的真实用户
- 文章应为Harbor镜像仓库的使用案例、经验分享、功能介绍等方面的中文文章，1000字以上。
- 文章需要在2017年3月1日之后在网上公开发表，例如技术论坛、个人博客、微信公众号等平台。
- 文章必须内容真实，且是参与者原创，严禁抄袭。
- 立刻扫码参与



# 提问



Harbor开源项目群



# Thank you!

<https://github.com/vmware/harbor>