SNIA ESF | ETHERNET STORAGE

Current State of Storage in the Container World

Chad Hintz, Cisco
Eric Forgette, Nimble Storage

November 17, 2016

SNIA Legal Notice

SNIA ESF | ETHERNET STORAGE

- The material contained in this presentation is copyrighted by the SNIA unless otherwise noted.
- Member companies and individual members may use this material in presentations and literature under the following conditions:
  - Any slide or slides used must be reproduced in their entirety without modification
  - The SNIA must be acknowledged as the source of any material used in the body of any document containing material from these presentations.
- This presentation is a project of the SNIA.
- Neither the author nor the presenter is an attorney and nothing in this presentation is intended to be, or should be construed as legal advice or an opinion of counsel. If you need legal advice or a legal opinion please contact your attorney.
- The information presented herein represents the author's personal opinion and current understanding of the relevant issues involved. The author, the presenter, and the SNIA do not assume any responsibility or liability for damages arising out of any reliance on or use of this information.

NO WARRANTIES, EXPRESS OR IMPLIED. USE AT YOUR OWN RISK.

About SNIA

SNIA ESF | ETHERNET STORAGE

SNIA-At-A-Glance

160
unique member companies

3,500
active contributing members

50,000
IT end users & storage pros worldwide

Learn more: **snia.org/technical** **@SNIA**

Today's Presenters

Eric Forgette
Technical Director
Nimble Storage
@eric4jet

Eric Hintz
Principal Systems Engineer Cisco,
SNIA-ESF Board
@chadh0517

Agenda

- Overview of Containers
  - Virtual machines vs. Containers
  - Quick history, where we are now
  - How Docker containers work
  - Why containers are compelling
- Storage for Containers
  - Persistent and non-persistent
  - Options for Containers
  - NAS vs. SAN
- Future Considerations

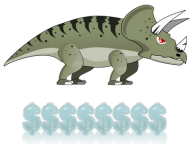
Virtual Machine vs. Container

A Brief History of On-premises Technologies

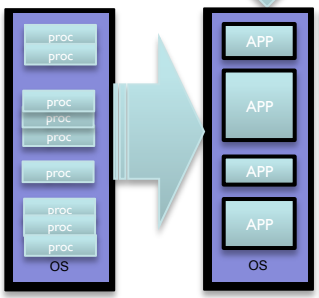
SNIA ESF | ETHERNET STORAGE

DEC

IBM



Mainframe



Zones

Projects

Fair Share

Timesharing

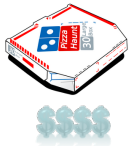
Unix ®

"App per Machine"

$O_{(1)}$

2x2x2

2x4x2



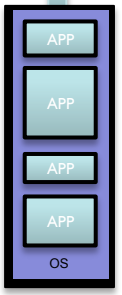
Linux / Windows

"App per Virtual Machine"

KVM Xen

ESX Hyper-V

Hypervisor



namespaces

cgroups

Completely Fair Share

Container

Multi-core

Mass-produced

Multi-processor

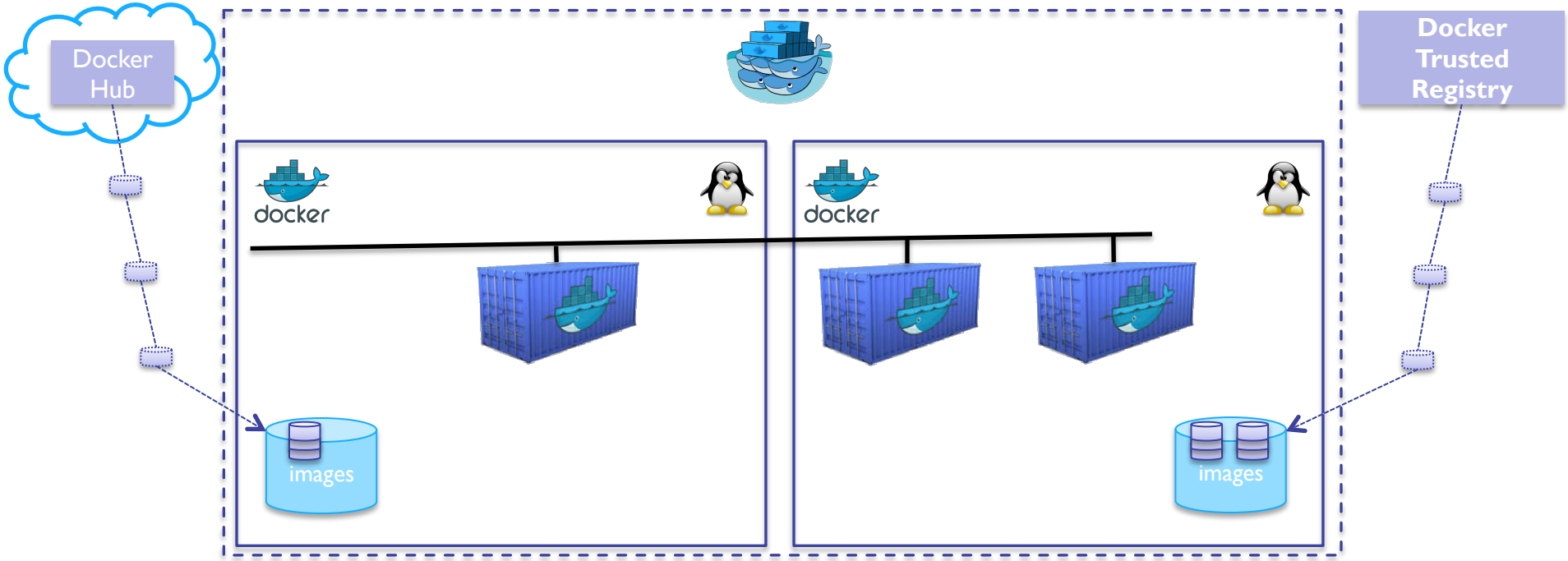
Why Containers are Compelling

- Operating System level isolation
  - Uses **cgroups and namespaces** in the Linux Kernel
  - Native Windows Containers in Windows Server 2016
    - Two types: Windows Server Container & Hyper-V Container
- Containers are about **applications**
  - **Define application needs** – the infrastructure will build it
  - Agility and consistency **in the software supply chain**
- Gives **Developers and Operations Teams** common interface
  - **Developers** care about **software** dependencies for **their app**
  - Operations care about **reliability, availability and performance**
- Docker **builds, ships and runs** applications everywhere

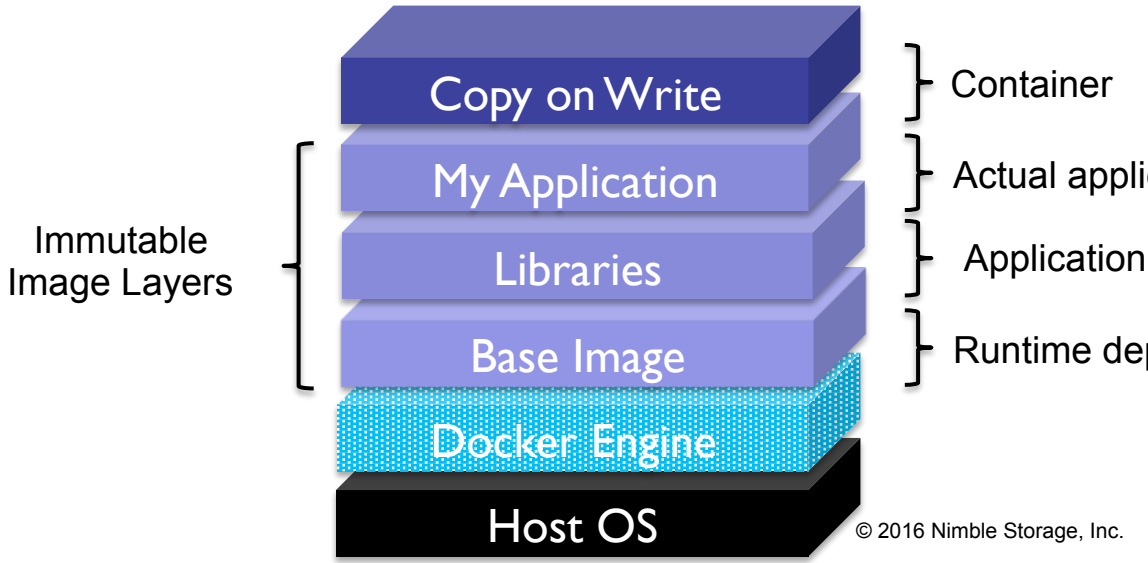
Orchestration

- Distributed Cluster
- Container Scheduler
- Examples
  - Docker Swarm
  - Kubernetes
  - Apache Mesos
  - Nomad

Docker Environment Overview

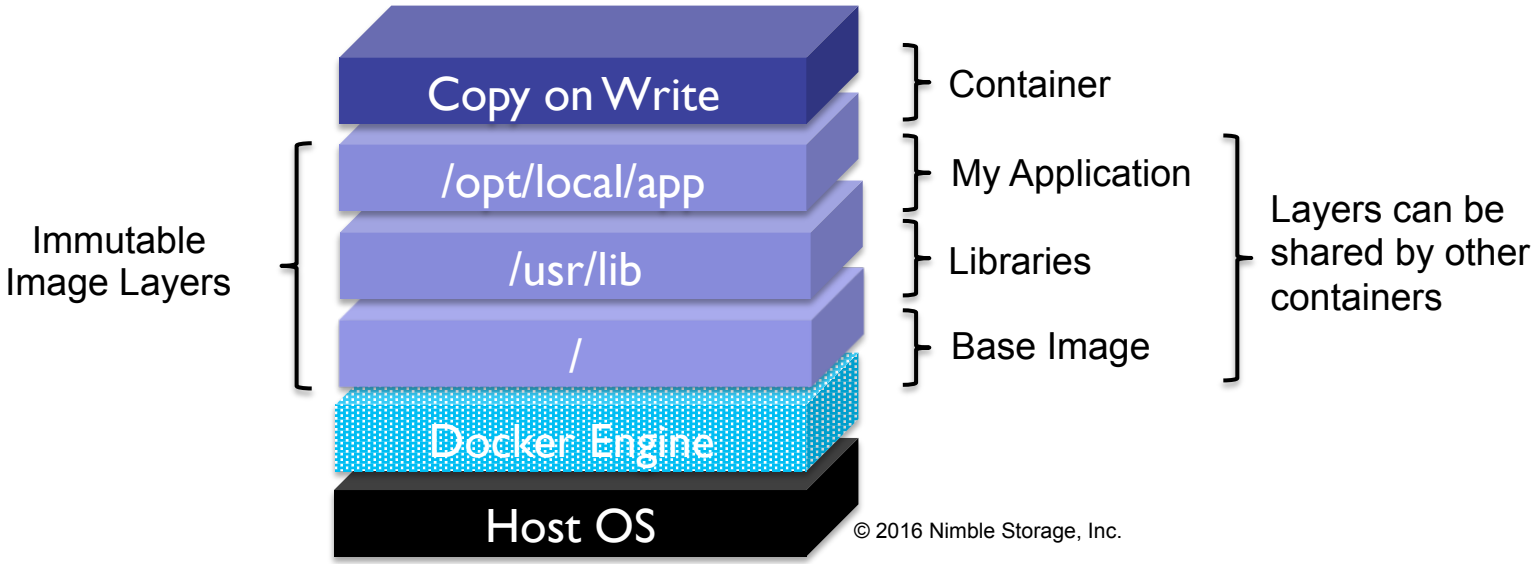


Docker Image File System Model



- Layers are composed by a union file system
- Changes are stored with the particular container instance image (COW)
- Data stored in the container post-creation is only suitable for transient content

Docker Image File System Model



- Layers are identified using cryptographic hashes of the layer's content
- Graph driver stacks the layers which provides the unified view from the container
- When the container is deleted, so are the changes in the Copy on Write layer

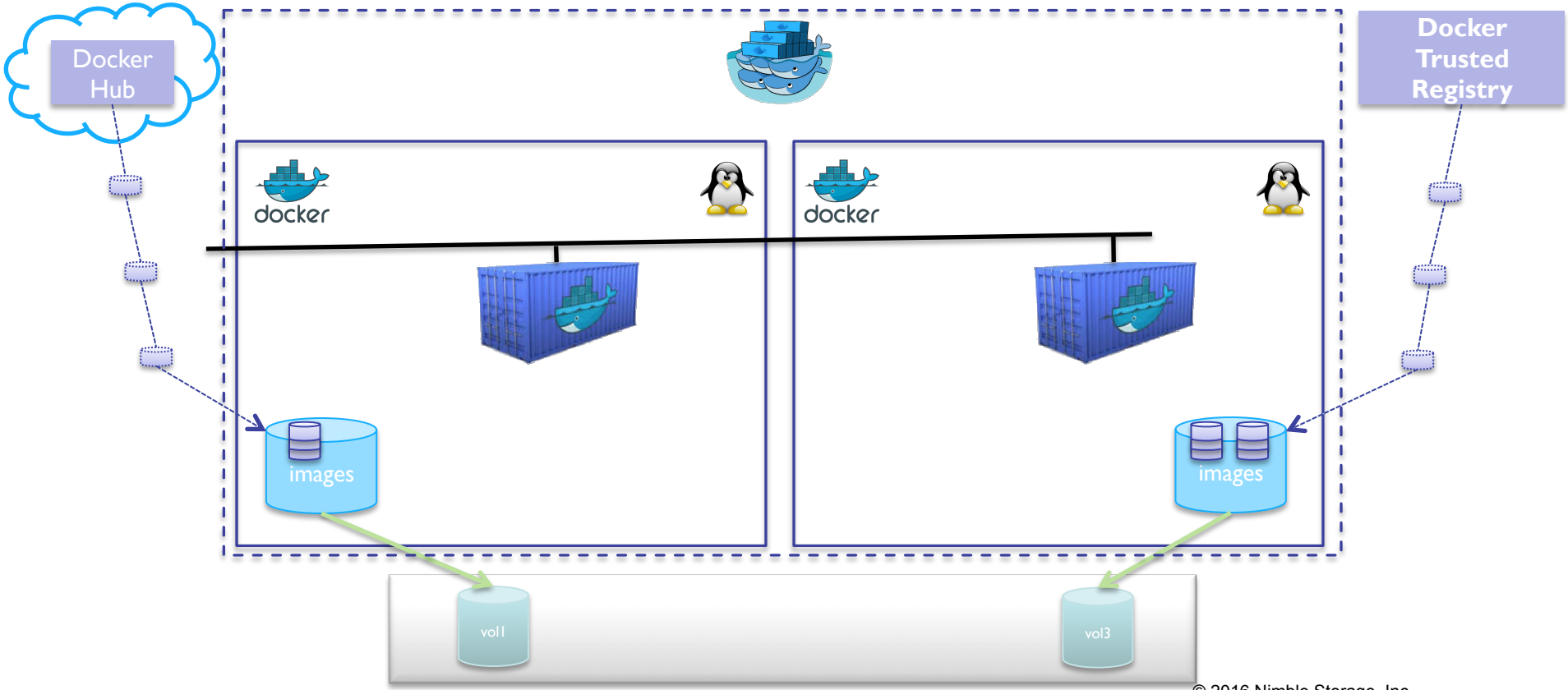
Graph Drivers

- Provides a local registry of images and layers
- Provides the Copy-On-Write functionality
- Allows for Layer creation
- Selection recommendations
  1. Use the default driver for your distribution*
  2. If implementation is within the limitations of Overlay2, use it
  3. If using the Commercially Supported Docker Engine, check Docker's compatibility matrix

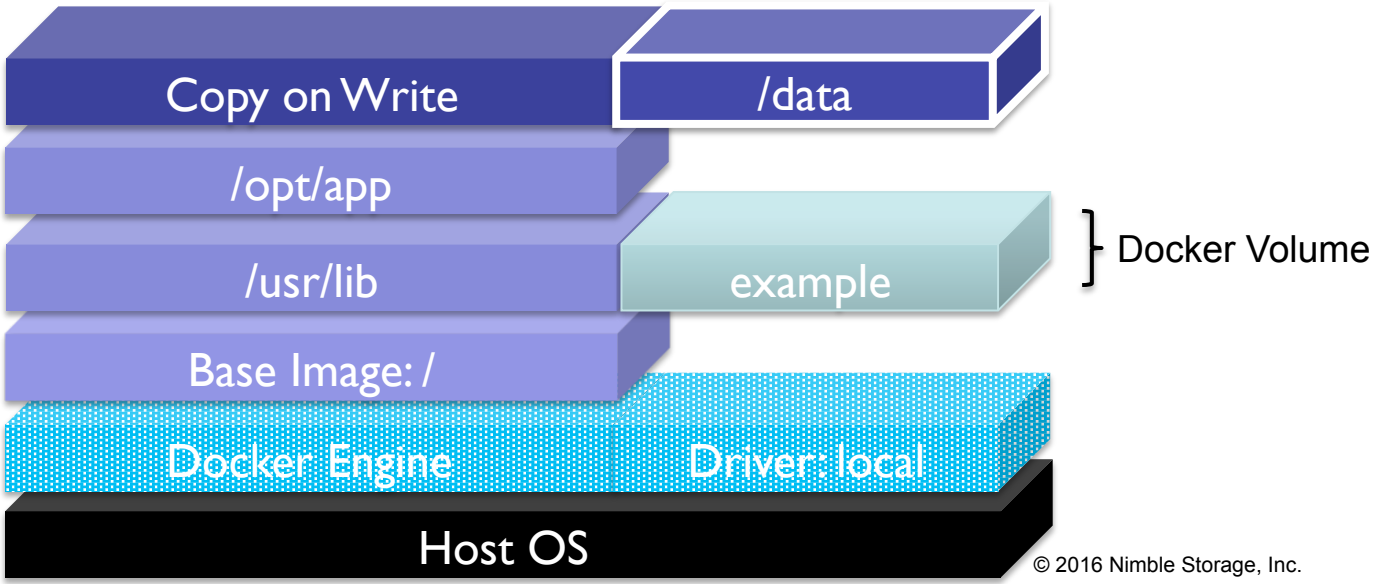
- OverlayFS
- AUFS
- Btrfs
- Device Mapper
- VFS
- ZFS

*Do not use the Device Mapper driver in loopback mode in production

Locate Node Registry on External Storage



Docker Volume



- Provides local persistence
- Bypass Copy-On-Write (COW) layer
- Presents a directory inside the container
- Persist after container is destroyed
- Docker on Linux supports mount options
- Docker on Windows (currently) supports no options
- Docker Volumes are separate, named, and reusable entities

Local Docker Volume


```
$ docker volume create --name example
example
$ docker volume inspect example
    {
        "Name": "example",
        "Driver": "local",
        "Mountpoint": "/var/lib/docker/volumes/example/_data",
        "Labels": {},
        "Scope": "local"
    }
```

```
$ docker run --name myContainer -v example:/data alpine date
$ docker inspect myContainer
<snip>
 "Mounts": [
            {
                "Name": "example",
                "Source": "/var/lib/docker/volumes/example/_data",
                "Destination": "/data",
                "Driver": "local",
                "Mode": "z",
                "RW": true,
</snip>
```

Docker Volume Behaviors

- When the image has data in the directory where the volume is to be mounted, and the volume is empty, the content of the directory is copied to the volume
- A Docker Volume cannot be removed if it is referenced by a container

Create a volume and show it is empty:

```
$ docker volume create --name example
example
$ ls -l /var/lib/docker/volumes/example/_data
total 0
```

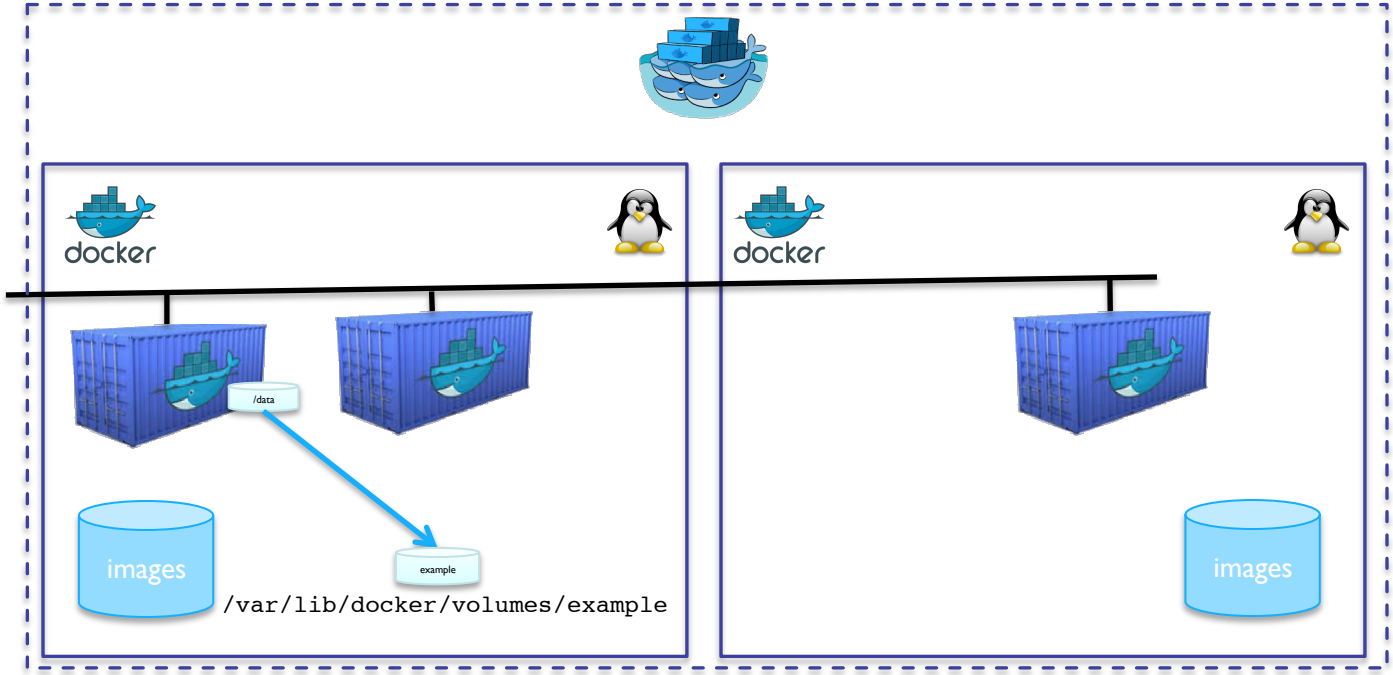
Run a container and show that the directories from the image have been copied to the volume:

```
$ docker run -it -v example:/var/lib alpine date
$ ls -l /var/lib/docker/volumes/example/_data
total 0
drwxr-xr-x 2 root root 6 Oct 18 11:58 apk
drwxr-xr-x 2 root root 6 Oct 18 11:58 misc
drwxr-xr-x 2 root root 6 Oct 18 11:58 udhcpd
```

Unable to remove a volume that is referenced by a container:

```
$ docker volume rm example
Error response from daemon: Unable to remove volume, volume still in use:
remove example: volume is in use - [a41a33563358b75ae483a595659fa433d34ea7761f05bd09f3151e43fd28870f]
```

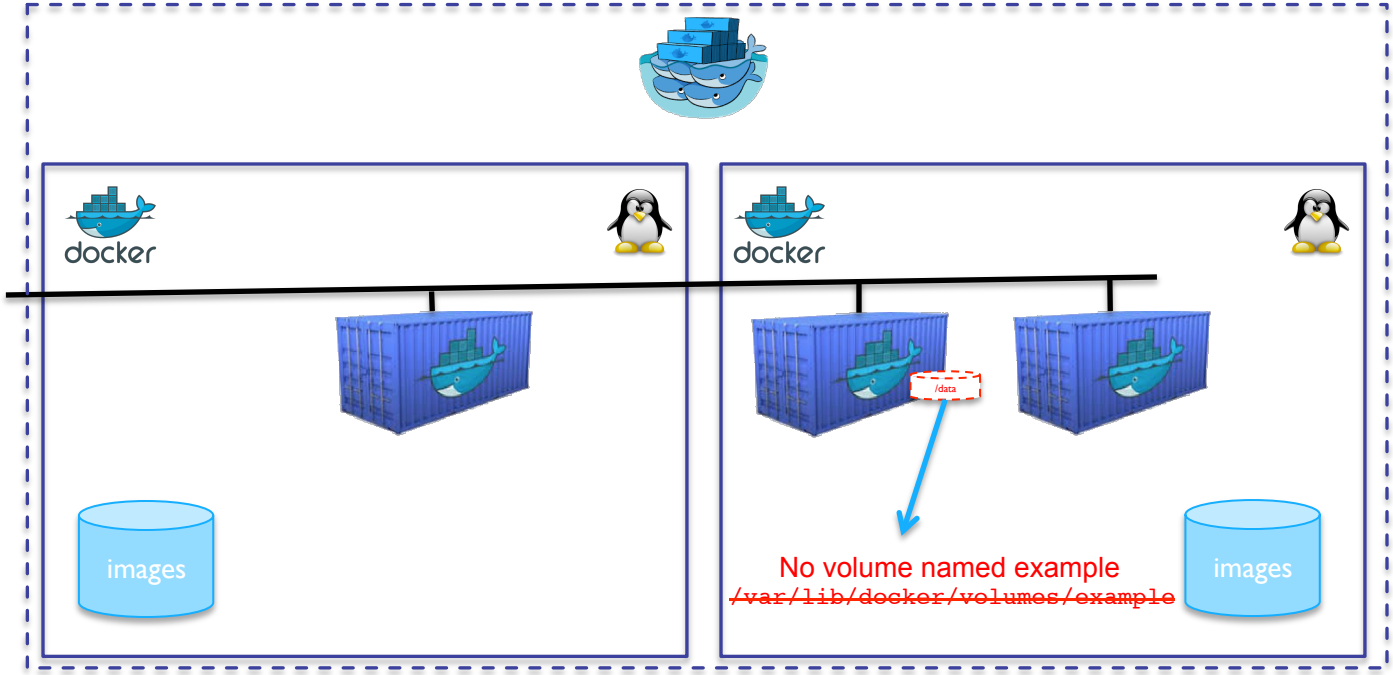
Local Persistent Storage




```
$ docker volume ls
DRIVER              VOLUME NAME
local               example
```

```
$ docker volume ls
DRIVER              VOLUME NAME
```

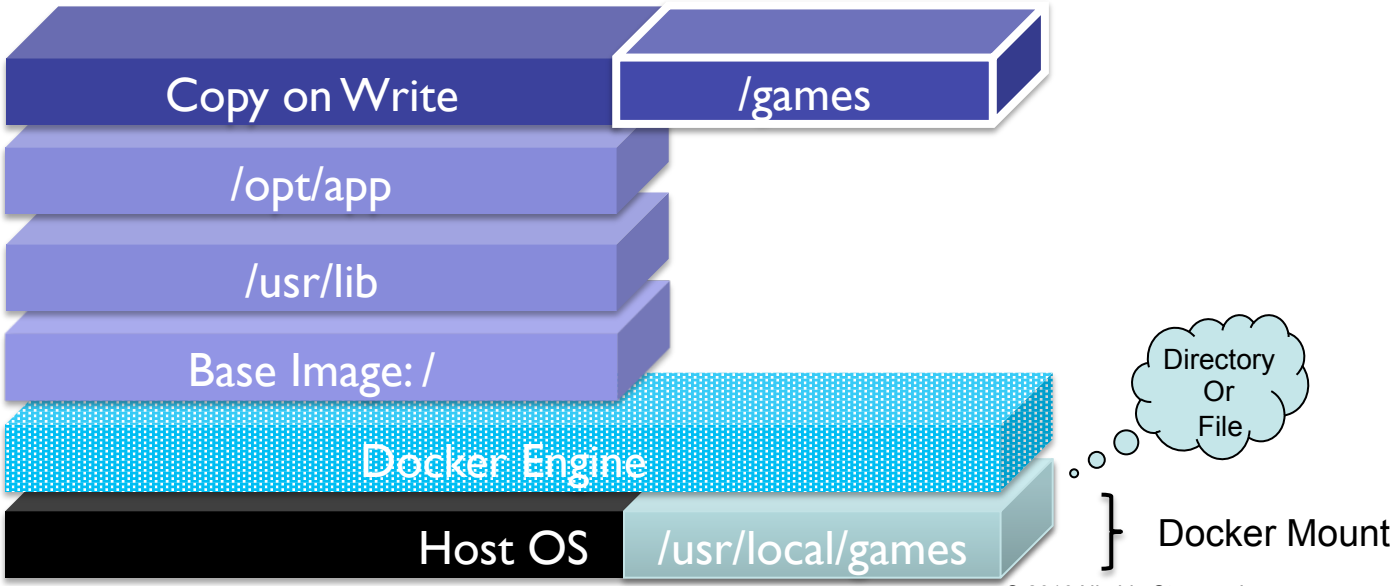
Local Persistent Storage




```
$ docker volume ls
DRIVER              VOLUME NAME
local               example
```

```
$ docker volume ls
DRIVER              VOLUME NAME
```

Host Directory/File Exposed to Container



- Provides local persistence
- Presents a directory from the host into the container
- Bypass Copy-On-Write (COW) layer
- **Not a Docker Volume**: Docker Volume Behaviors Don't Apply

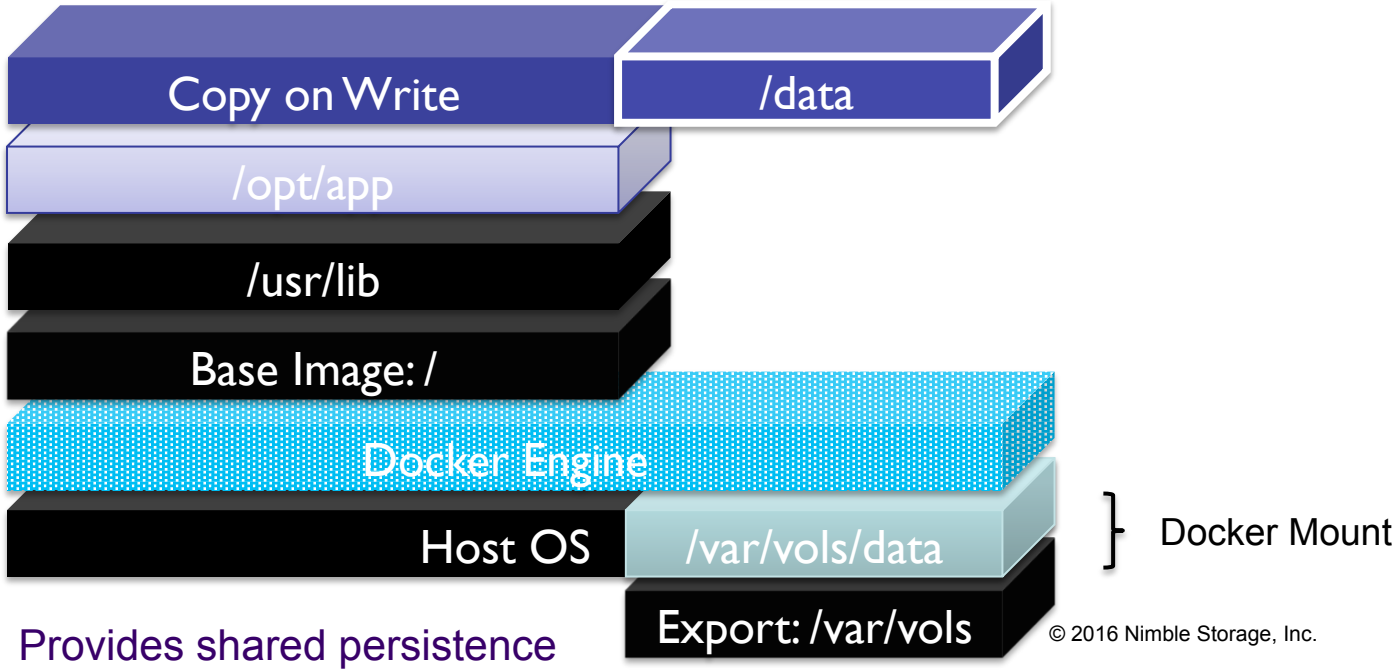
Host Directory/File Exposed to Container

- Directories need not be created manually
  - Docker engine silently creates the directory if missing
  - Docker engine bind mounts the directory into the container (hiding existing content)
  - Often used to expose read-only access

```
$ docker run --name myContainer -v /usr/local/games:/games:ro alpine date

$ docker inspect myContainer
<snip>
        "Mounts": [
            {
                "Source": "/usr/local/games",
                "Destination": "/games",
                "Mode": ”ro",
                "RW": false,
</snip>
```

Directory from NFS Mounted Filesystem



- Provides shared persistence
- Presents a directory from an NFS mounted export into the container
- Bypass Copy-On-Write (COW) layer
- **Not a Docker Volume**: Docker Volume Behaviors Don't Apply

Directory from NFS Mounted Filesystem

- Export should allow root access (no_root_squash)
- Mount should be present in /etc/fstab
- Directories need not be created manually
- Directory is bind mounted into the container (hiding existing content)
- No protection from accidental deletion of directory
- Little or no isolation between containers leads to noisy neighbor
  - Single “bucket” of capacity
  - Single filesystem/device providing IO

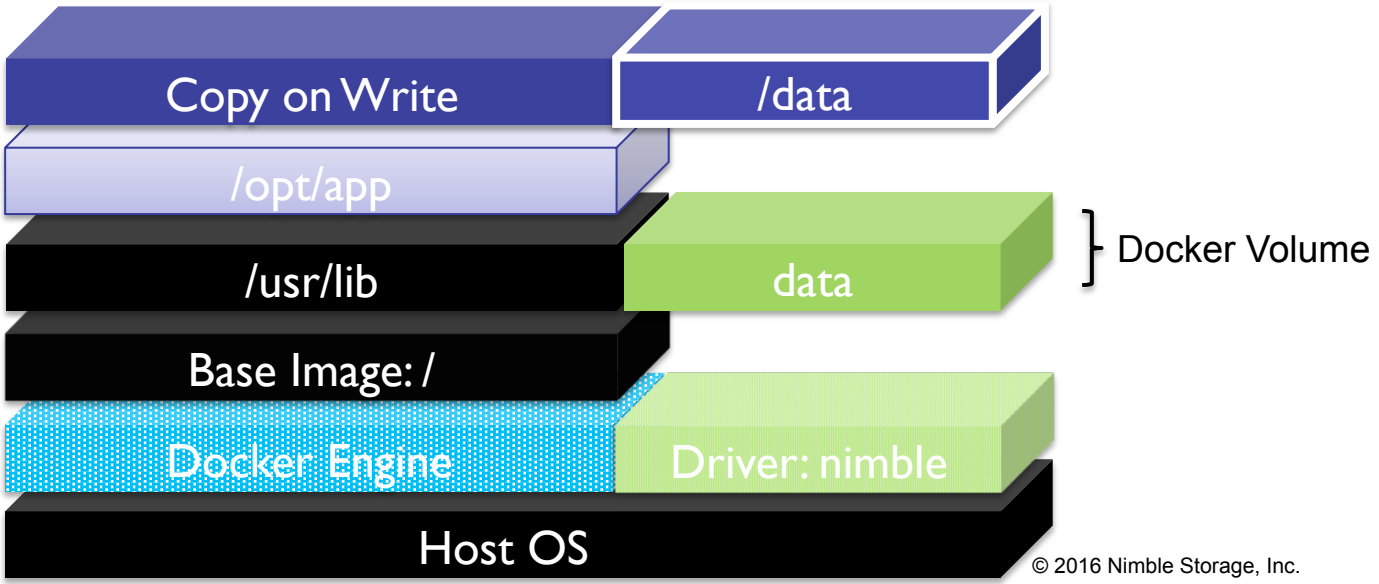
Anti-pattern

```
$ docker run --name myContainer -v /var/vols/data:/data alpine date
$ docker inspect myContainer
<snip>
            "Mounts": [
            {
                "Source": "/var/vols/data",
                "Destination": "/data",
                "Mode": "",
                "RW": true,
</snip>
```

“Batteries Included But Swappable”

- Docker plug-in framework announced @ DockerCon 2015
  - Network plug-ins
  - Volume plug-ins
- Plug-ins allow 3rd parties to extend the capabilities of Docker
- Volume Plug-ins exist for both SAN and NAS solutions
- Volume Plug-ins allow
  - Local and global scope
  - Storage system capabilities to be exposed
  - High performance storage options for Docker Containers

Docker Volume Created by Plug-in



- Can provides locally or globally scoped persistence
- Docker Volumes are separate, named, and reusable entities
- Bypass Copy-On-Write (COW) layer
- Presents a directory or filesystem inside the container

Docker Volume Created by Plug-in

SNIA ESF | ETHERNET STORAGE

```
$ docker volume create --name example --driver=nimble
example
$ docker volume inspect example
    {   "Name": ”example",
        "Driver": "nimble",
        "Mountpoint": "",
        "Status": {
            "Blocksize": 4096,
            "DedupeEnabled": true,
            "Description": "Docker knows this volume as example.",
            "EncryptionCipher": ”AES-256",
<lines removed />
            "PerfPolicy": "DockerDefault",
            "ThinlyProvisioned": true,
            "VolSizeMiB": 10240,
            "VolumeName": ”example.docker”
        },
        "Labels": {},
        "Scope": "global”
    }
```

Plug-in Driven Docker Volume Creation

- Using the vanilla Docker client:

```
$ docker volume create --driver=nimble –o sizeInGiB=50 --name myvol1
$ docker run -it -v myvol1:/data alpine /bin/sh
```

- Using Docker service with Docker SwarmKit:

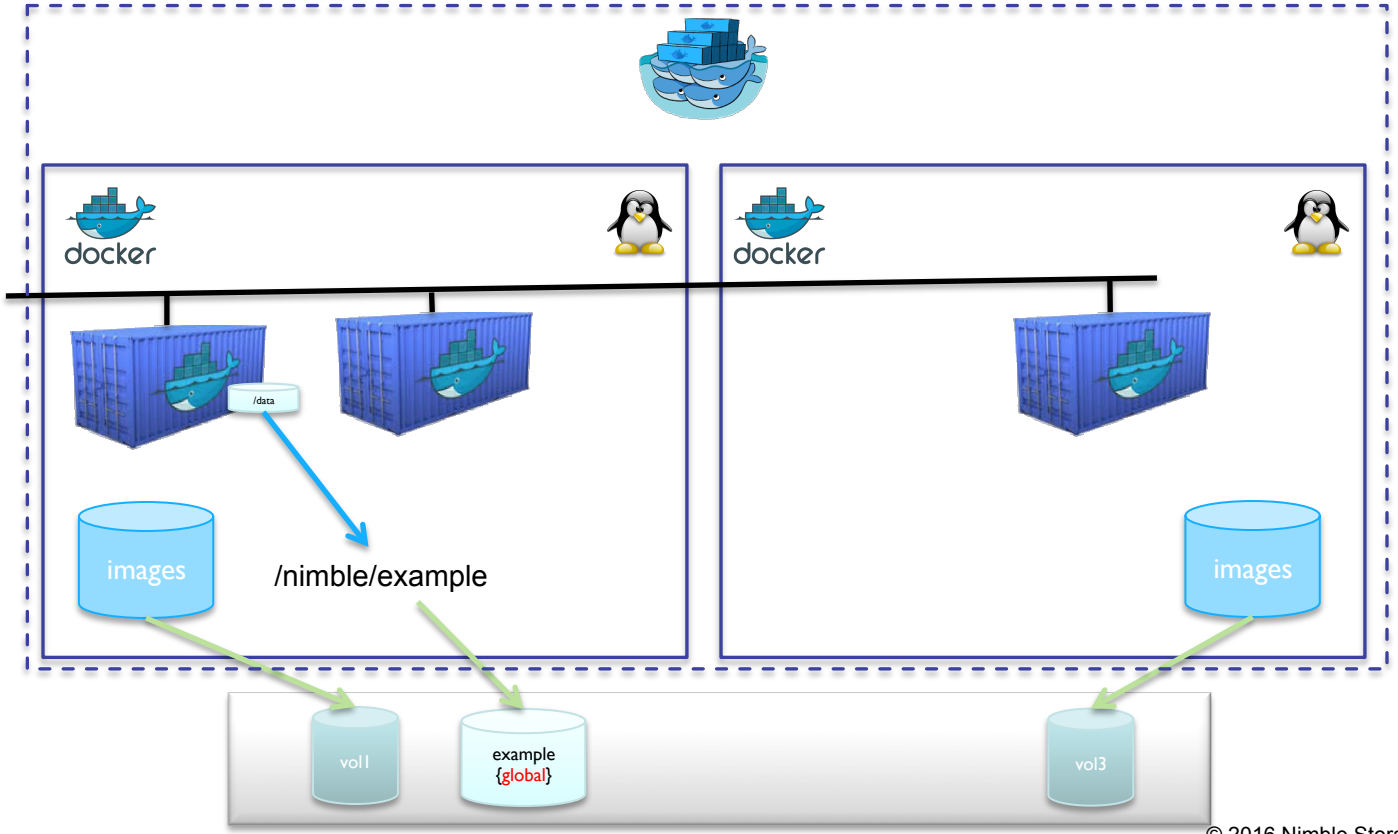
```
$ docker service create –mount type=volume,target=/usr/share/nginx/html,\
        source=myvol1,volume-driver=nimble,volume-opt=sizeInGiB=50 nginx
```

- Using Docker Compose:

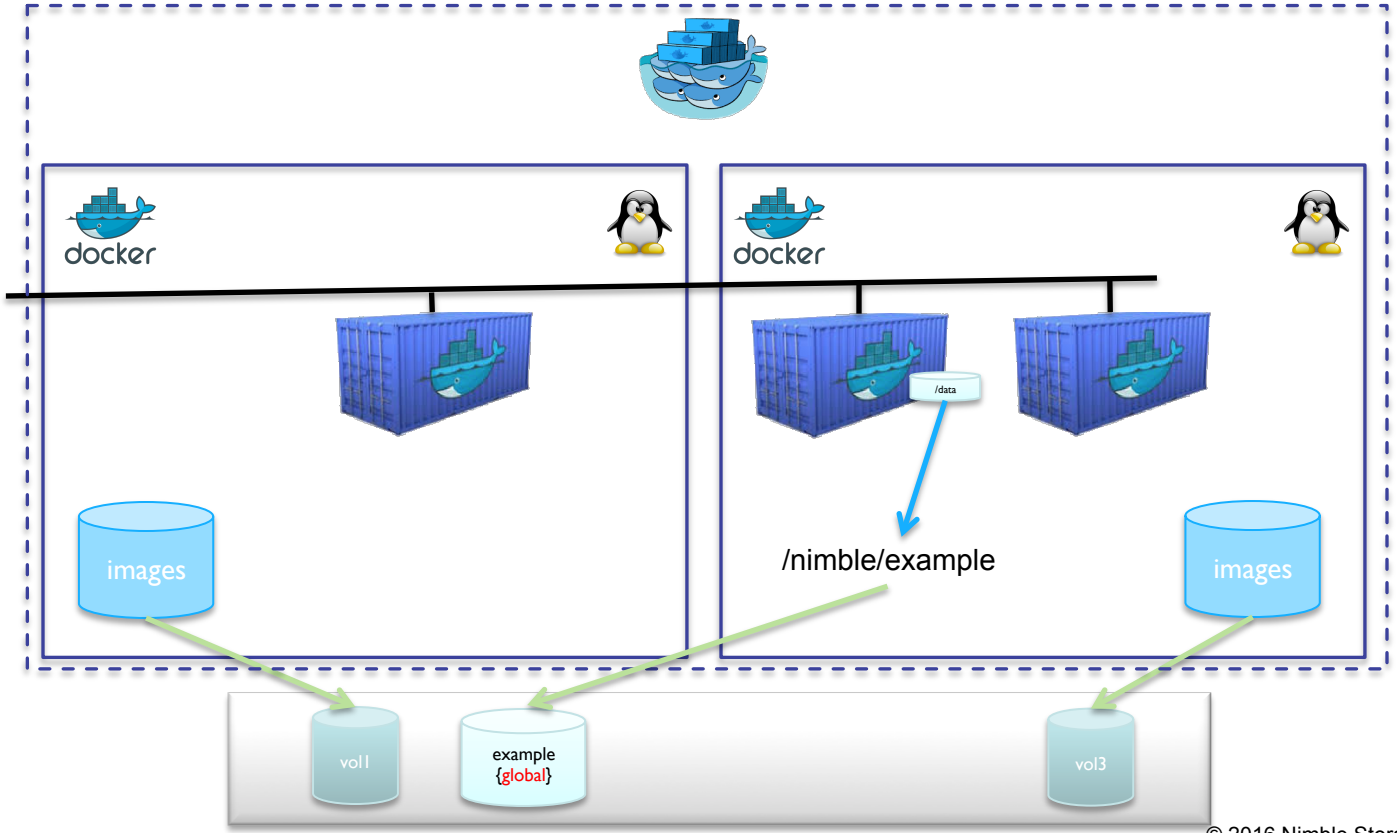
```
$ docker-compose -f web.yml –p web up –d
```

```
# web.yml
version: "2"
services:
  web:
    image: nginx:latest
    ports:
    - "8080:80"
    volumes:
    - myvol1:/usr/share/nginx/html
volumes:
  myvol1:
    driver: nimble
    driver_opts:
      sizeInGiB: 50
```

Persistent Shared Storage via Plug-in



Persistent Shared Storage via Plug-in



Future Considerations

- Storage for Central Image Registry
  - Performance
  - Protection
  - Scalability
  - Availability
- Orchestration
  - Plugging into other layers

Resources

- Docker Community
  - https://forums.docker.com/c/open-source-projectsCommunity
- Cloud Native Computing Foundataion
  - https://www.cncf.io/
- “Windows Containers” post by Taylor Brown
  - https://msdn.microsoft.com/en-us/virtualization/windowscontainers/about/about_overview
- Docker Documentation
  - https://docs.docker.com/
- Michael Mattsson’s blog
  - https://connect.nimblestorage.com/people/mmattsson/activity

More SNIA Webcasts on Containers

SNIA ESF | ETHERNET STORAGE

- On-Demand: **Intro to Containers, Container Storage and Docker**
  https://www.brighttalk.com/webcast/663/217971

- Live December 7, 2016: **Containers: Best Practices and Data Management Services**
  https://www.brighttalk.com/webcast/663/227349

- Stay updated! Join our Containers opt-in email List
  http://eepurl.com/ciMk0P

After This Webcast

SNIA ESF | ETHERNET STORAGE

- Please rate this webcast and provide us with feedback
- This Webcast and a PDF of the slides will be posted to the SNIA Ethernet Storage Forum (ESF) website and available on-demand
- www.snia.org/forums/esf/knowledge/webcasts
- A full Q&A from this webcast, including answers to questions we couldn't get to today, will be posted to the SNIA-ESF blog: sniaesfblog.org
- Follow us on Twitter @SNIAESF

Thank You