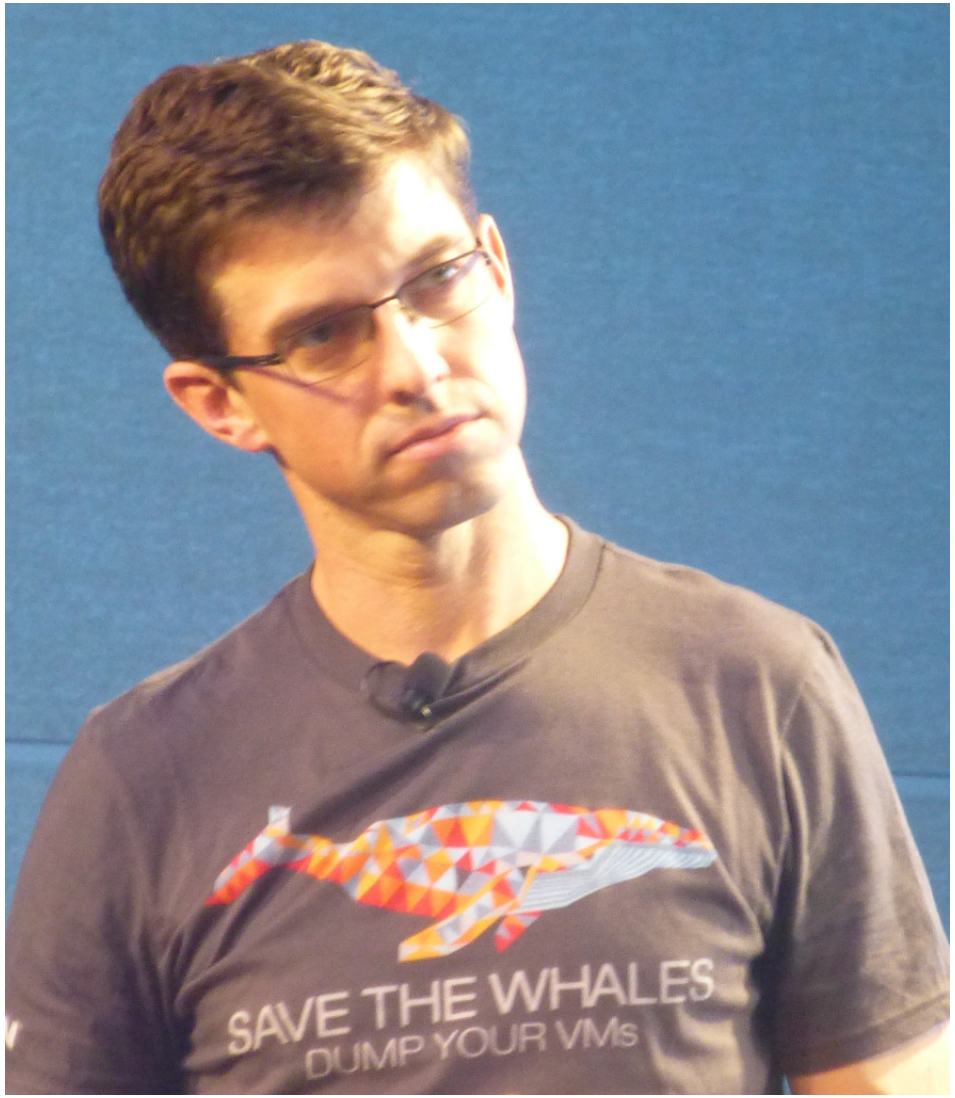
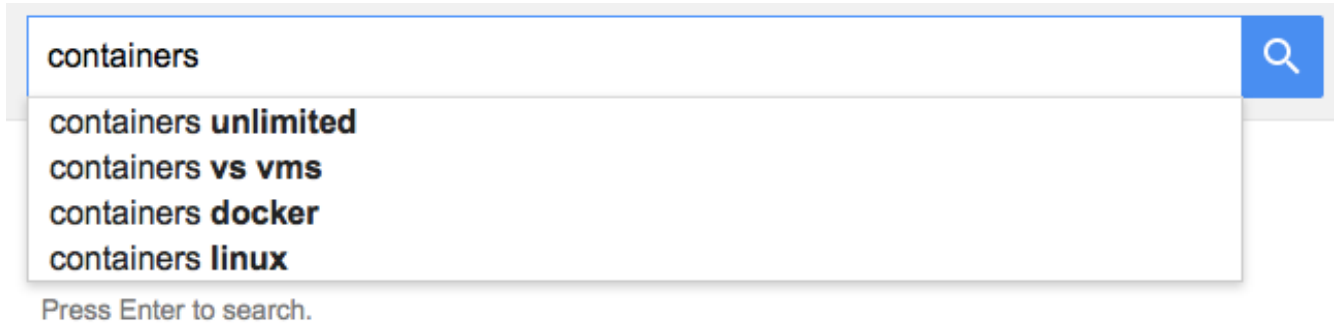
Exploding the Linux Container Host

Presenter: Ben Corrie (@bensdoings)

vmware

Containers vs VMs



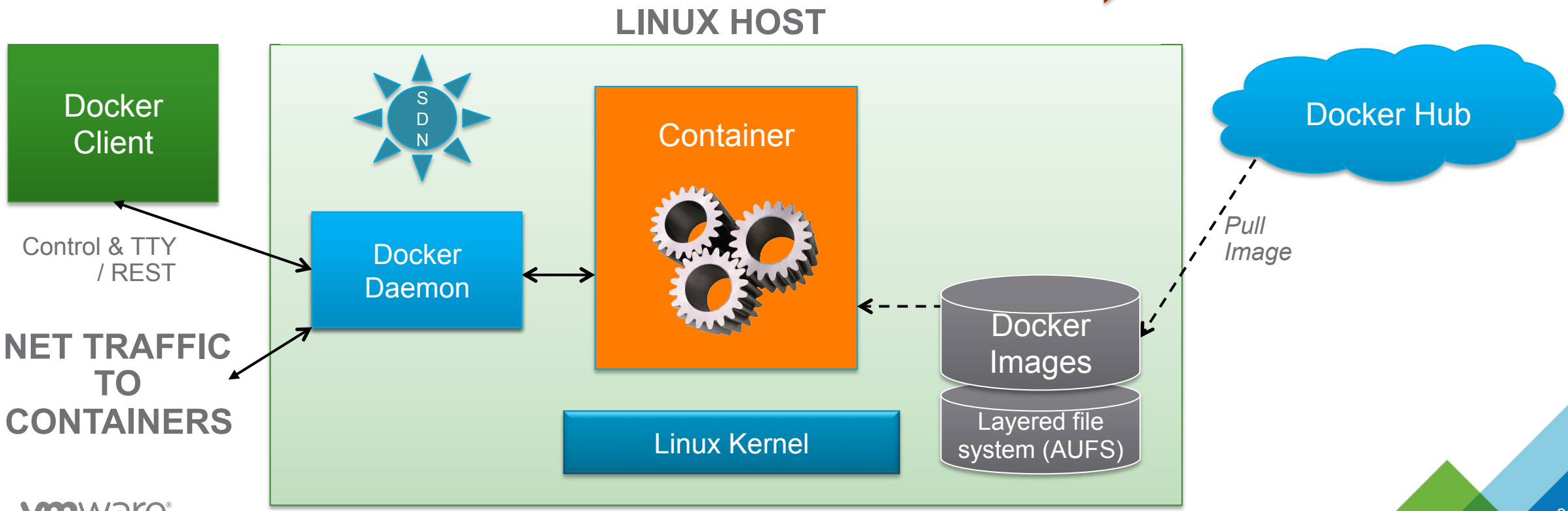


- Google Wisdom:
  - VMs and Containers are similar but different
  - Try running containers in VMs for security
  - Containers are best for scale-out density
  - VMs are better for legacy apps

What is a Container?

1. A executable process
2. Resource constraints / private namespace
3. Binary dependencies: Application, runtime, OS
4. A shared Linux kernel for running the executable

STATELESS
PORTABLE
FAST
SECURE

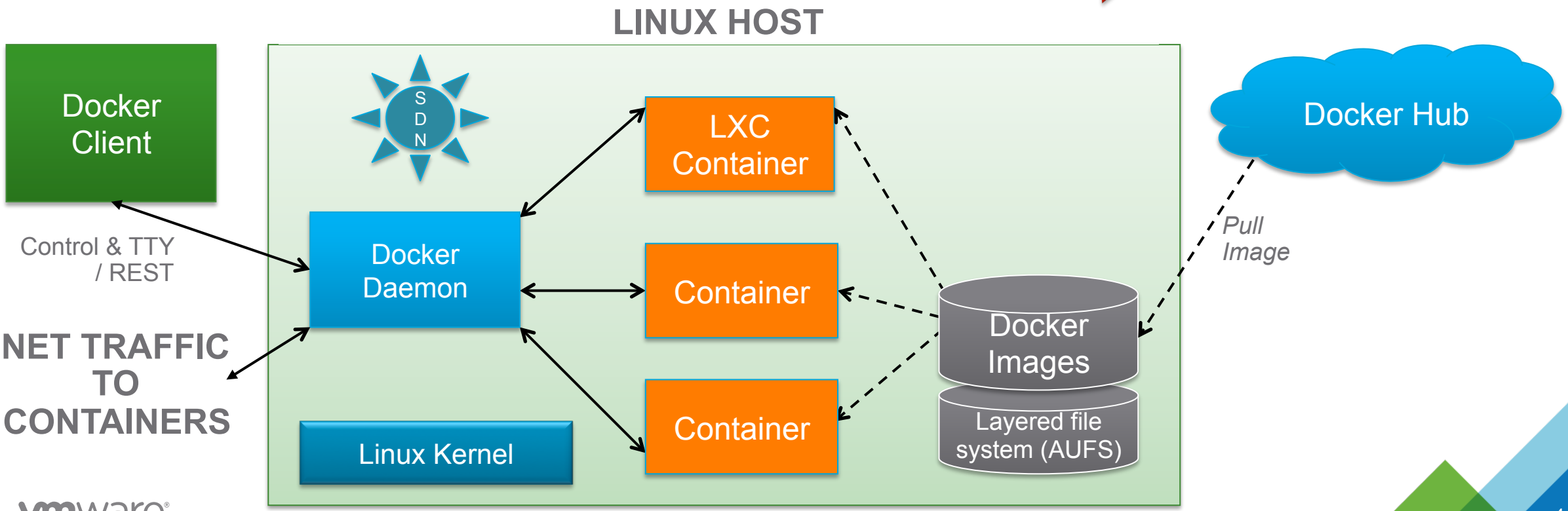


vmware

What is a Container Host?

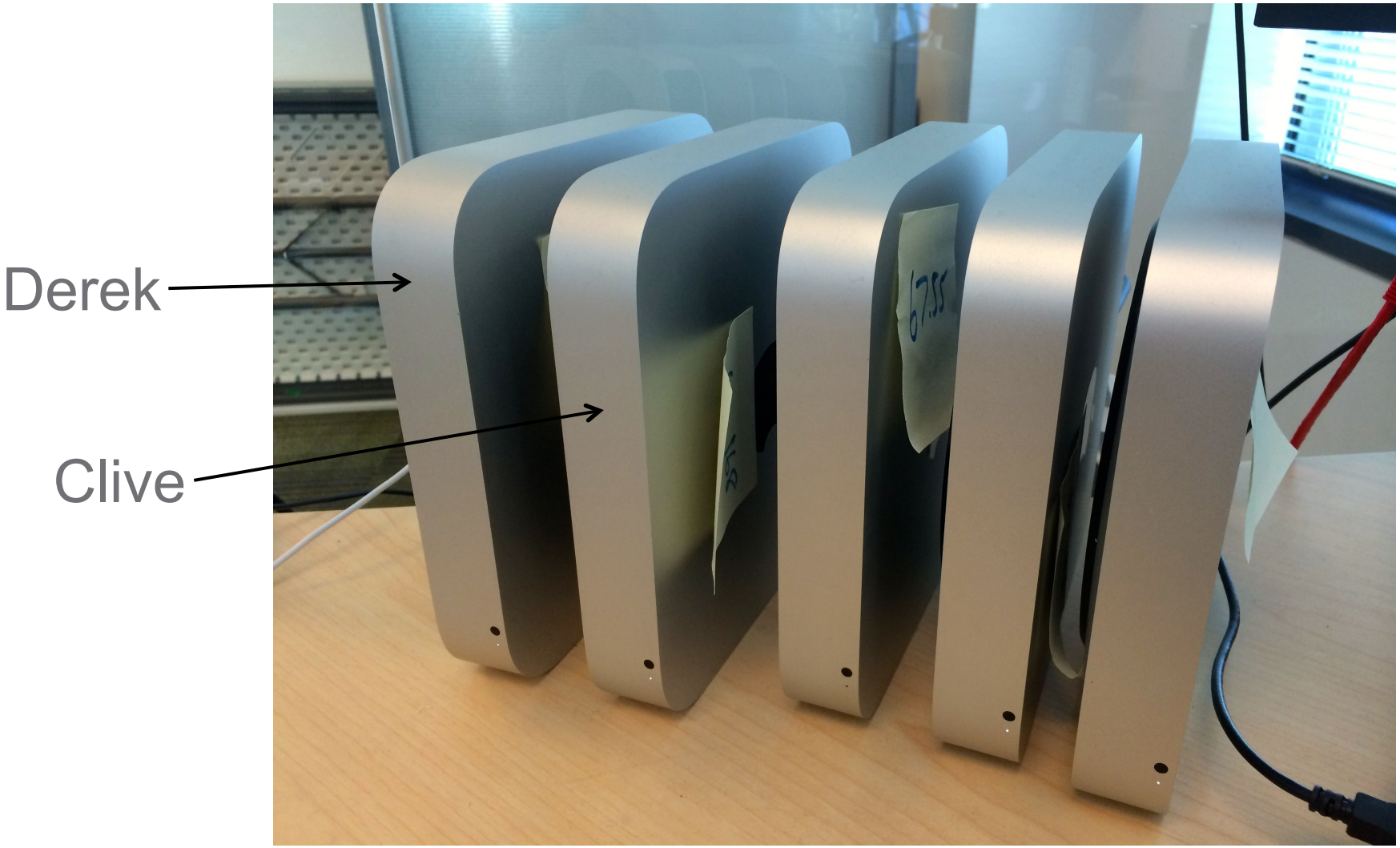
1. Control plane & lifecycle management for containers
2. Resource scheduling and a container abstraction
3. Infrastructure abstractions: Storage, networking etc
4. A Linux kernel

STATEFUL
LONG-RUNNING
SINGLE USER
SINGLE USE



vmware

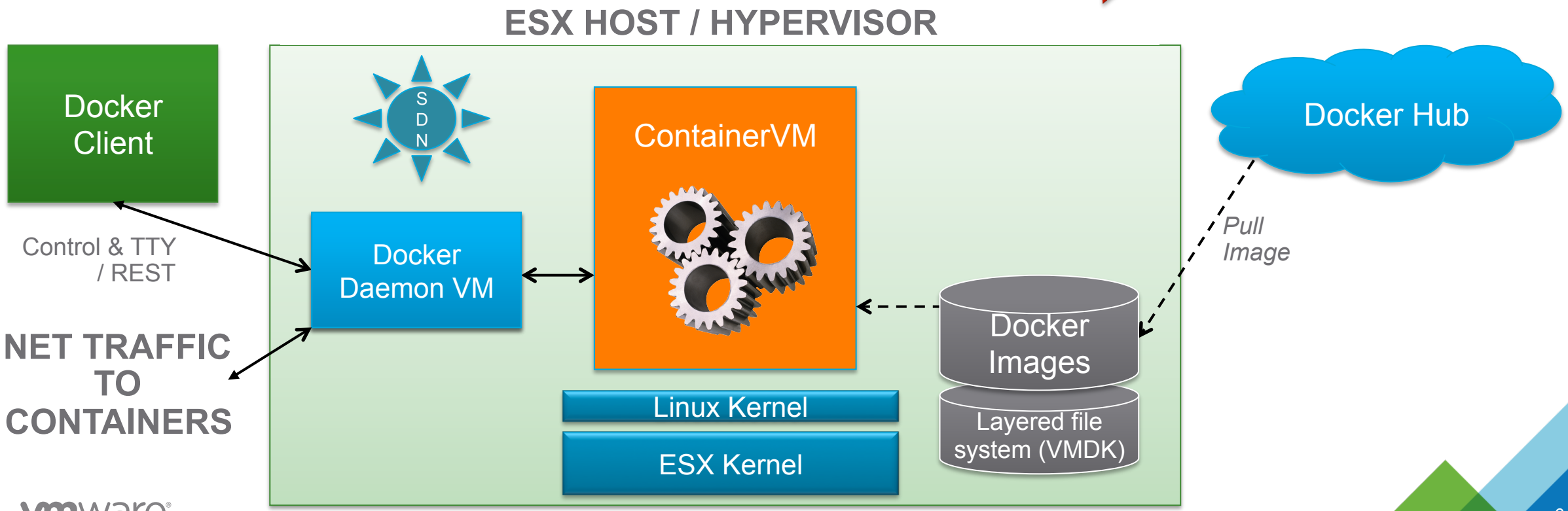
My Demo Container Hosts



What is a ContainerVM?

1. A executable process
2. Resource constraints / private namespace
3. Binary dependencies: Application, runtime, OS
4. A “shared” Linux kernel for running the executable

STATELESS
PORTABLE
FAST
SECURE



Why?????

- Simple answer: The Container Host
- Linux container host limitations
  - Single Docker daemon = single user
  - Long running – slow and disruptive to refresh
  - Stateful – images, volumes, containers, patch levels
  - Static size – only resource efficient if well-packed
  - Kernel is a single point of failure
- When virtualized
  - Limited access to virtual infrastructure
  - Limited monitoring of containers without 3rd party agents
  - Duplicated infrastructure layer

Differences between Derek & Clive

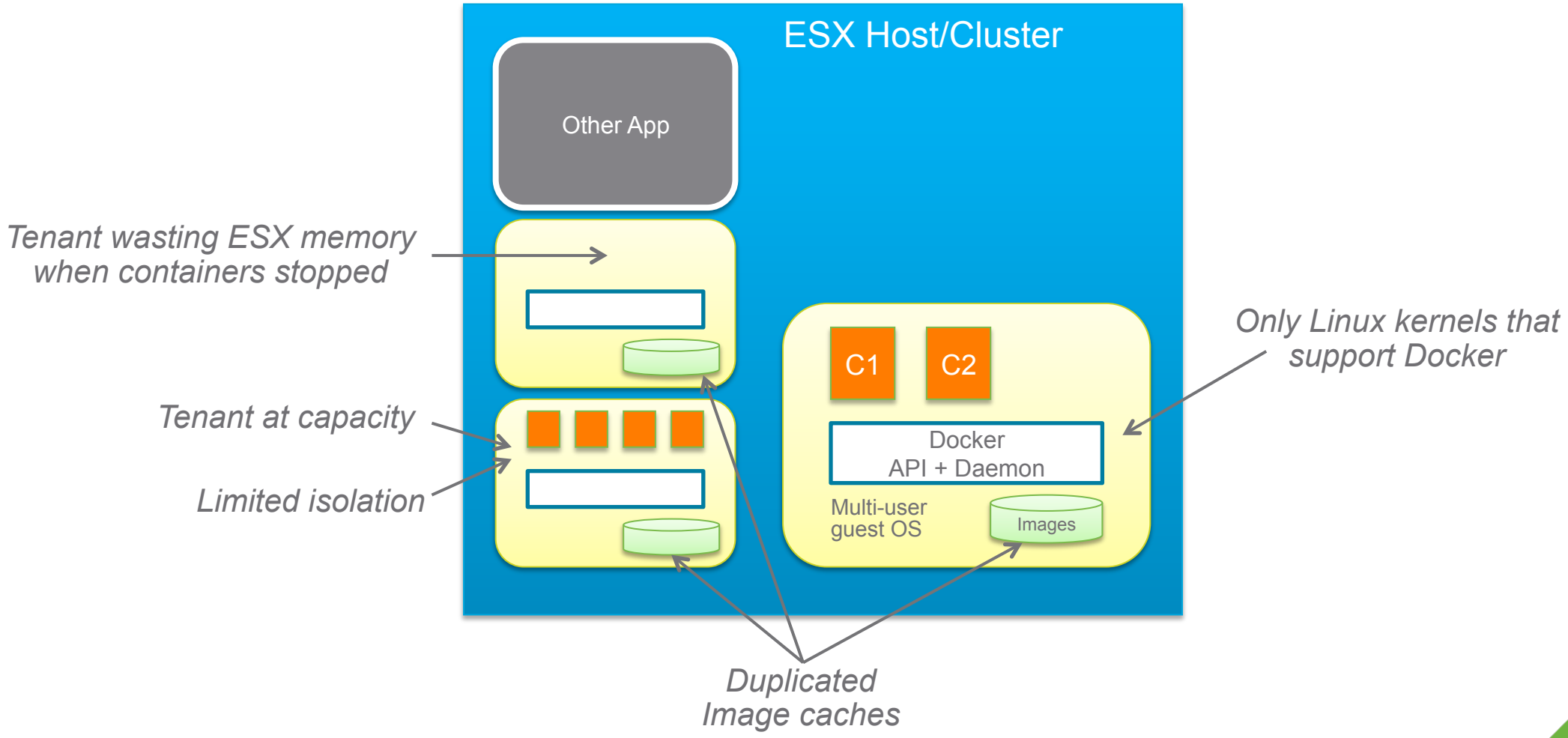
1. Multi-tenancy
2. Dynamic resource boundaries
3. Disposable nested container hosts
   - Control plane performance
   - Statelessness – container hosts as cattle!
   - Eg. Docker in Jenkins Slaves
     - Dependencies on slaves are contained
     - Slaves themselves need to be “garbage collected”
   - Eg. Pre-populated container cache for Docker build -> push -> dispose
   - Eg. Save /var/lib/docker in a volume – state persists, host does not
4. Multi-OS support

What is Bonneville?

The Docker ecosystem you love on the Hypervisor you trust

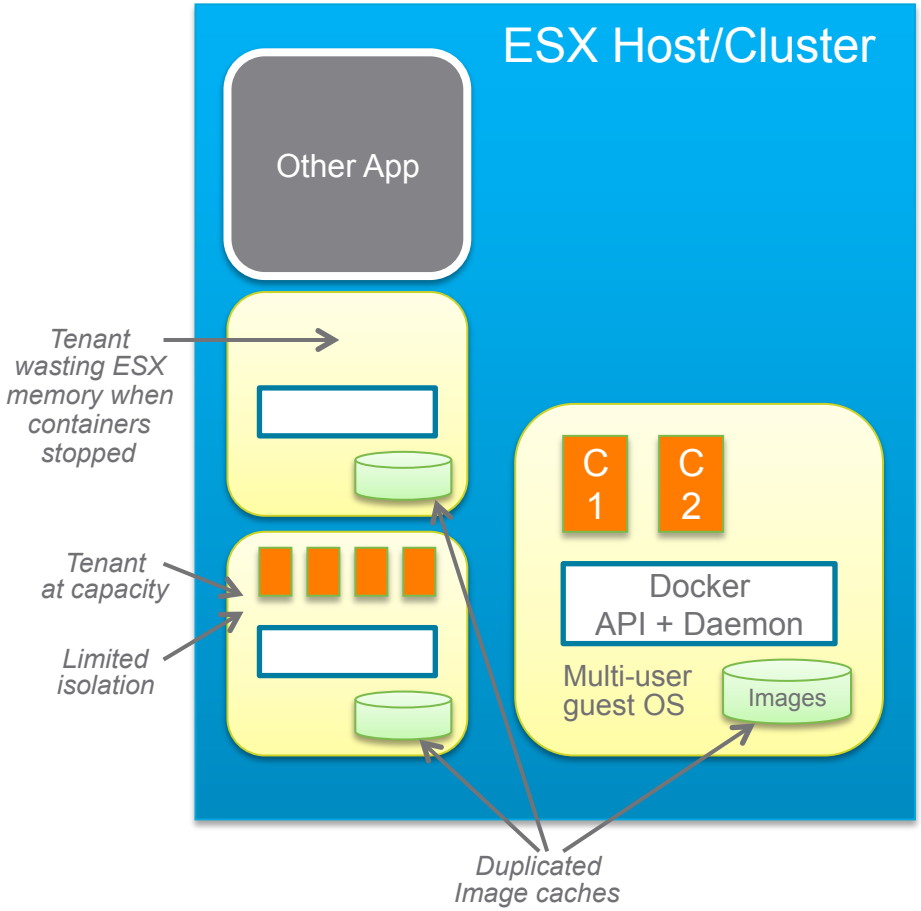
- **Provision Docker containers direct to vSphere**
  - No need for a Linux container host
  - Vanilla Docker client connects to Docker Daemon appliance
- **Hardware-virtualized "containerVM" abstraction**
  - Containers are provisioned *as* VMs, not *in* VMs
  - Hardware virtualization provides unprecedented security and isolation
  - x86 abstraction allows for more than just Linux
- **"Instant Clone" delivers container speed and efficiency**
  - Container start in 2 seconds with a "shared" Linux Kernel

Limitations Virtualizing Docker As-Is

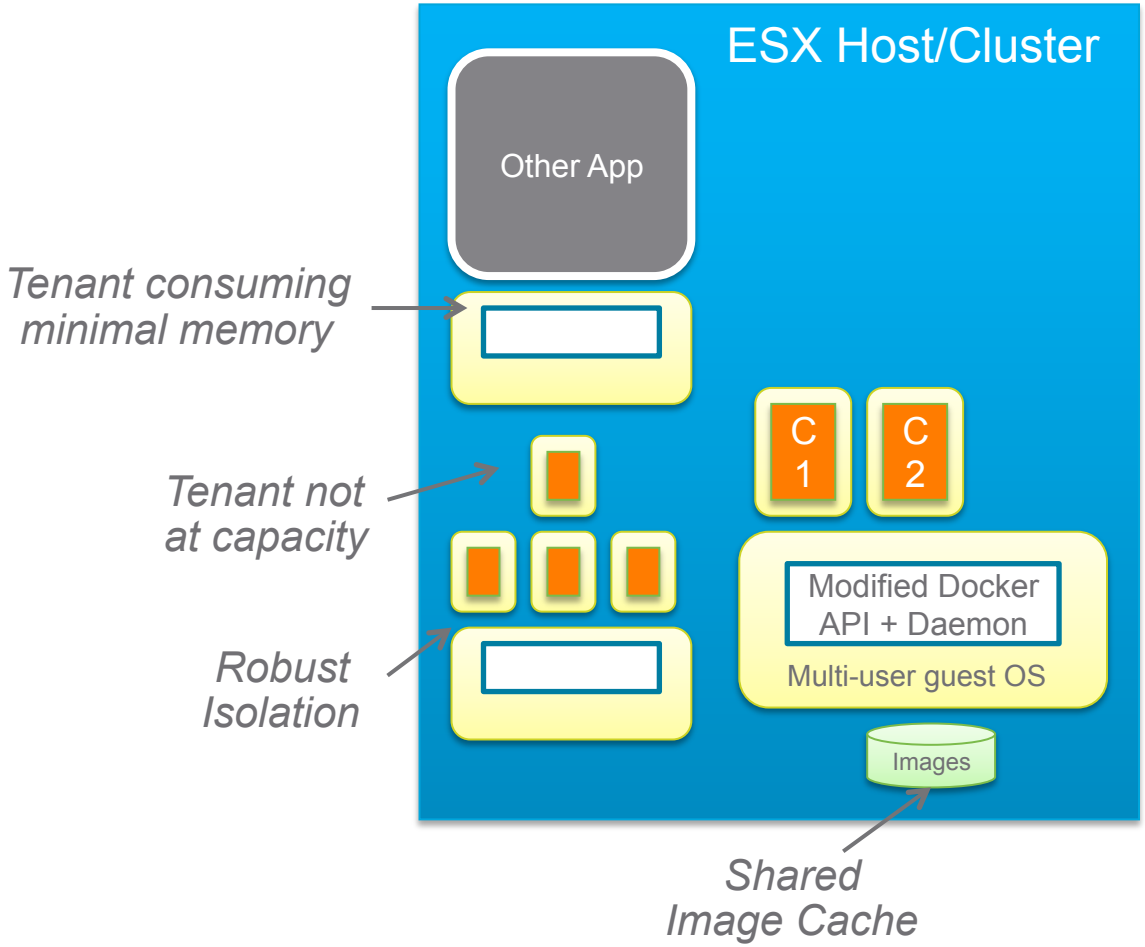


Exploding the Linux Container Host – in detail

From earlier…

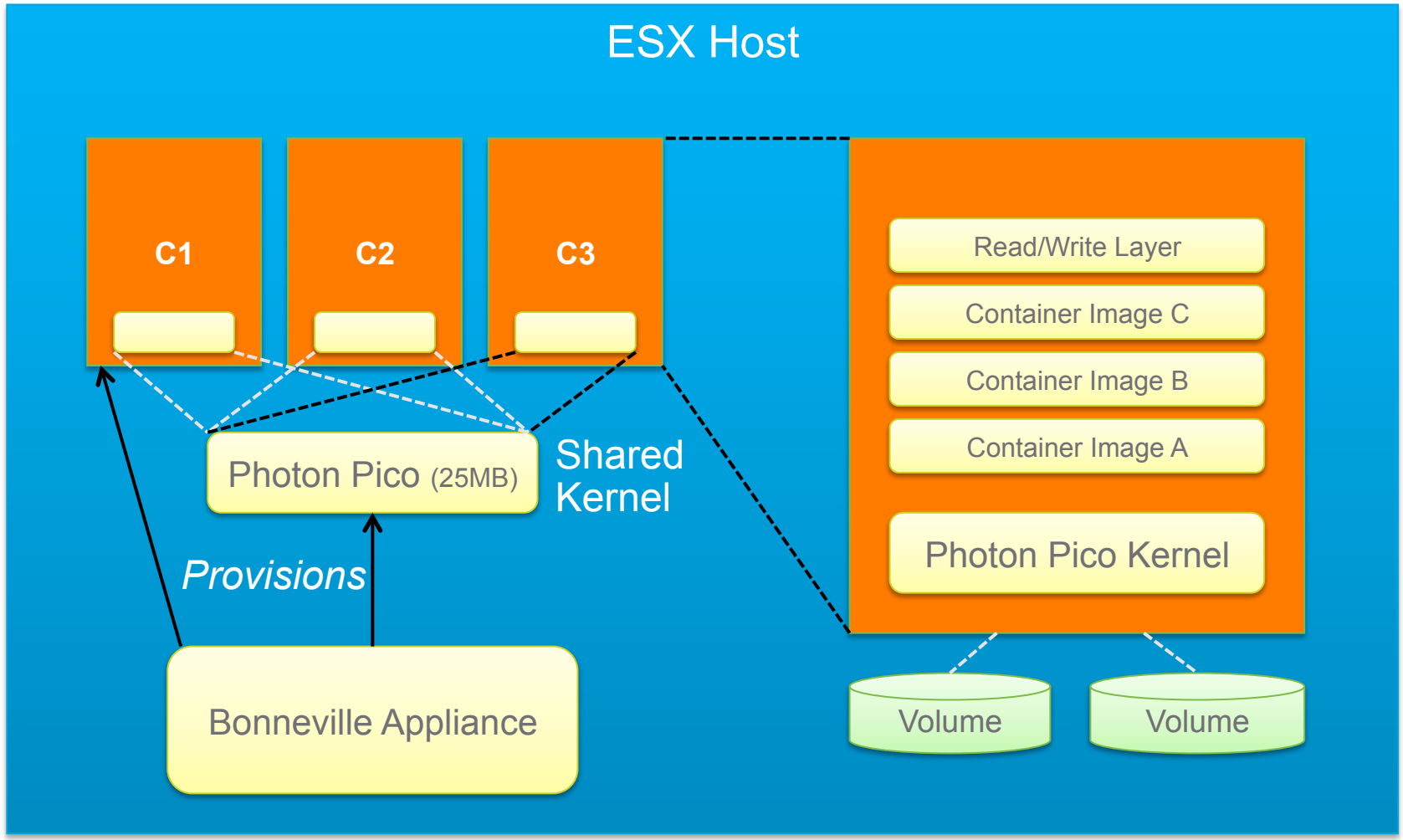


To this…



vmware

What's inside? Instant Clone and the "shared" Linux Kernel



Bonneville Efficiency

- Early concerns about efficiency of 1:1 container / VM mapping
- Container efficiency typically measured in terms of start time and memory consumption
- **Start Time**
  - Start time not inherent limitation of VMs, simply the need to boot an OS
  - Instant Clone removes the need for OS boot
  - Docker appeal more than just container start time – pull image, run image, delete image flow
  - Developers want instant container start, less critical when provisioning apps
- **Memory consumption**
  - Misleading “Hello World” comparisons often made. Real apps use memory regardless
  - Bonneville memory efficiencies achieved through Instant Clone + Photon Pico
  - Instant Clone raises the potential for sharing much more than just the base OS

Docker Feature Parity: Can you even tell?

- **Goal for Bonneville is complete transparency to the client / user**
- **Some concepts have to be a little different**
- **Container privileged access**
  - In Docker, flag gives a container privileged access to both the host kernel and the host itself
  - In Bonneville, privileged access is the default with zero access to the host
- **Host mounted volumes**
  - In Docker, you can mount a volume on the host into a container
    - Useful for certain things, but means that the container is not idempotent
  - In Bonneville, the host and container don't share a filesystem
- **Default container size**
  - In Docker if no constraints are specified, container has access to all the hosts resources
  - In Bonneville this wouldn't make sense, so a default size is used

vSphere Integrated Containers: The Virtual Container Host

- **What is a "Container Host"?**
  - A finite amount of compute resource with the necessary capability to host containers
- **A container host does not have to be bound to an OS or physical machine**

Concept	Linux	ESX	VCH
Container host boundaries	A VM or physical box	An ESX server	A vSphere resource pool
Grow container host	Shut down VM / N/A	N/A	Reconfigure the pool
Clustering	Docker Swarm	Docker Swarm	vSphere cluster
Nested hosts	Docker-in-Docker	Resource pool / Photon	Resource pool / Photon

Isolation and Security

- **Various takes on the "containerVM" concept have recently emerged**
  - **"Clear Containers" from Intel**
    - Similar to Bonneville in concept, but different in execution – more of an OSS POC
    - KVM without x86 QEMU layer or BIOS initializes Intel "Clear Linux" very fast
  - **"Hyper"**
    - Startup based in China with a very similar concept to Bonneville
    - Supports KVM and Xen with a custom Linux kernel. Intended as Container-as-a-Service infrastructure
- **Security and Isolation at the heart of these solutions**
  - Hypervisor hardware isolation is well proven and battle-hardened. Linux kernel exploits keep emerging
  - Need to be able to secure and verify provenance of container images
- **Bonneville delivers best of all worlds**
  - Robust security and isolation of a VM
  - Full privileged access to a kernel – load kernel modules, loopback mount etc.

Summary

- Docker is a platform
- Bonneville is the Docker platform for vSphere
- Bonneville gives you best of both worlds
  - Speed, efficiency and workflow of containers
  - Security, isolation and flexibility of VMs
- Don’t let your container hosts become pets!

@bensdoings