Let's decompose storage (again)

Why? How? Huh?

MSST May 17 2017

Evan Powell

blog.openebs.io

https://github.com/openebs

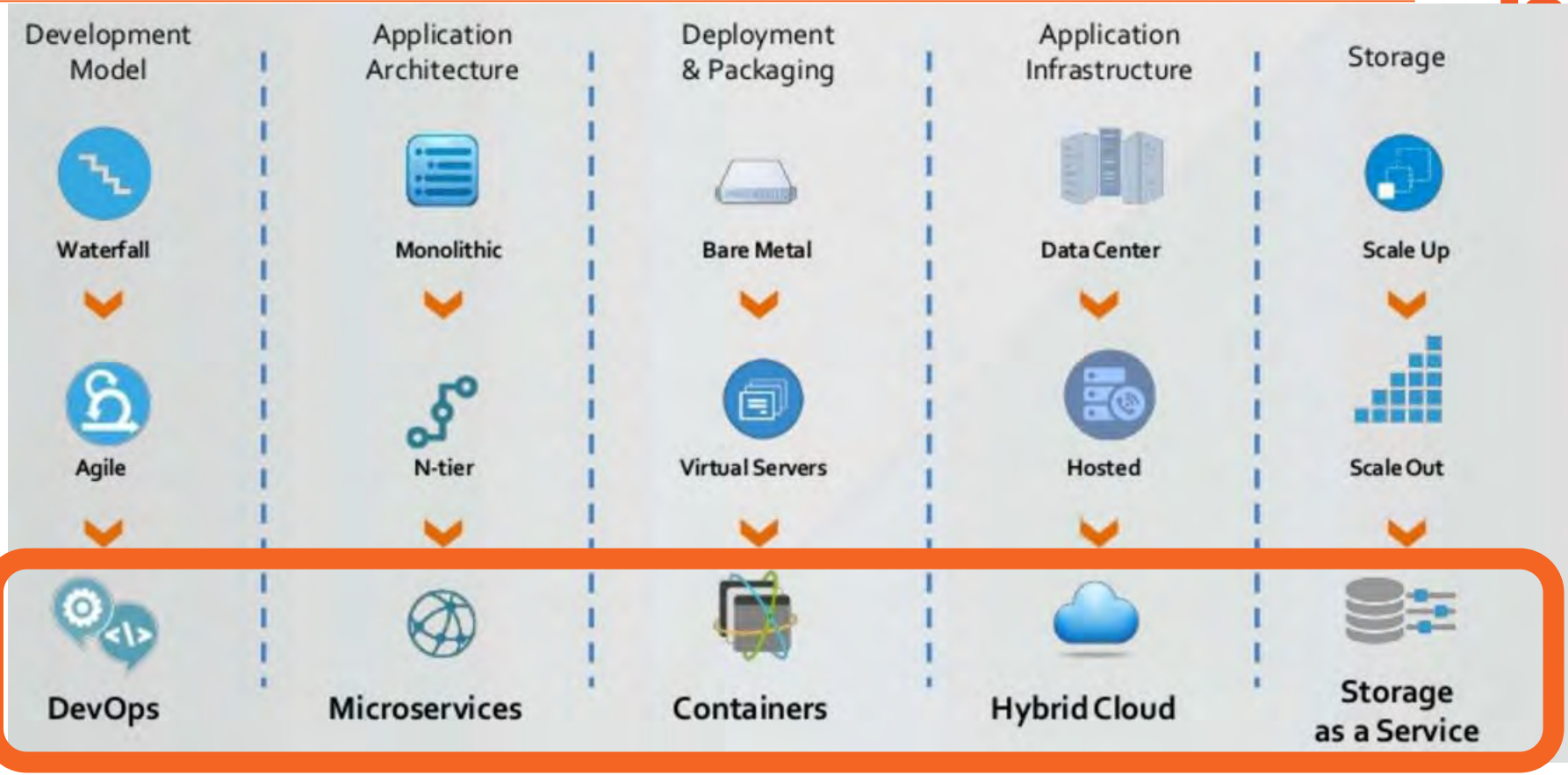
Join the
community
#slack
slack.openebs.io

@openebs

What's new?

http://www.slideshare.net/ColleenCorrice/persistent-storage-for-containerized-applications

What's new?



http://www.slideshare.net/ColleenCorrice/persistent-storage-for-containerized-applications

Layering

(3-TIER APPs)

CERTIFIED SYSADMIN(S)

Supervised Provisioning
Manage Storage Upgrades!
Manage 100s of Volumes
Managed Upgrades

NFS,iSCSI

Enterprise Storage
(SAN, NAS, ScaleOut,
DP, DR, Backup,
Compression, Dedup)

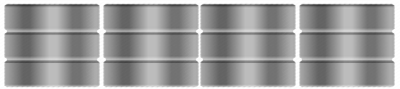
(LOCAL APPs)

GEEK

Format Disks and Use.
Manage Few Disks

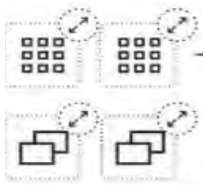
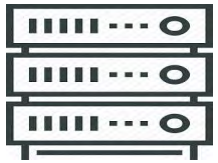
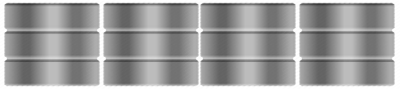
XFS, ZFS, LVM, EXT4

POSIX
(IO, Checksum, Snapshots)

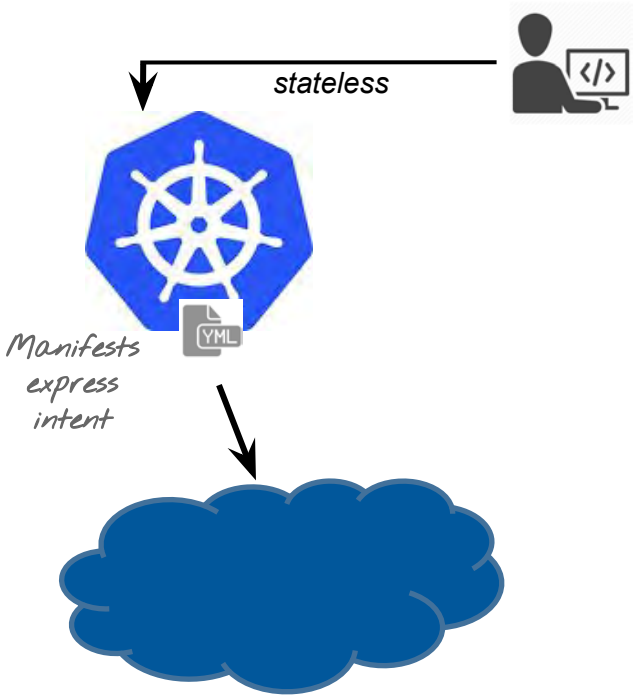


Storage Medium
(Disk, Flash, RAM)

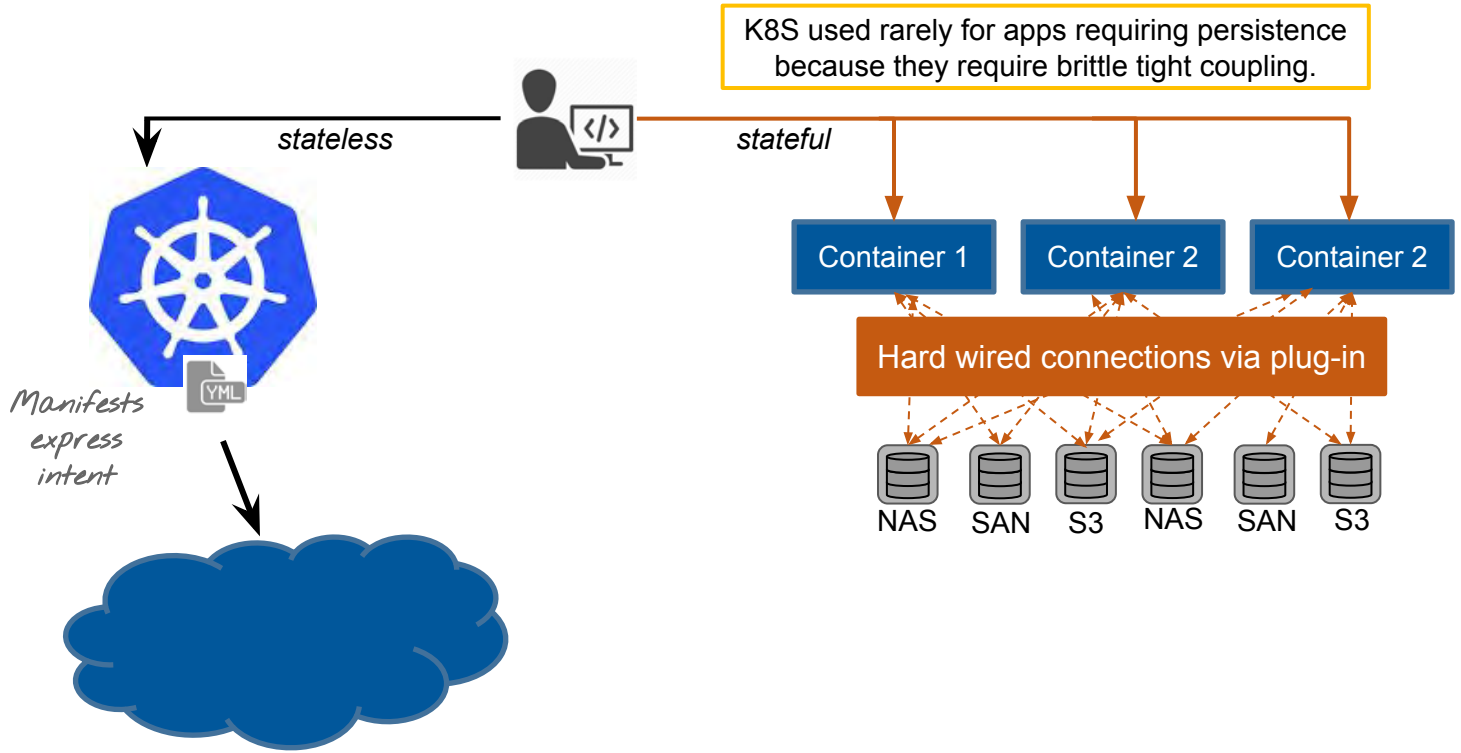
Layering

(MICRO APPS) 	DEVOPS AT SPEED 	Auto-magically Provisioned Auto-managed Storage(ML) Data Mobility Manage Millions of Volumes Seamless Upgrades	OpenEBS	Containerized Storage (Hyper-Converged, Auto-Scaling, Auto-Upgradable, Yaml Driven)
(CLOUD SERVICE) 	PAAS ADMINs 	Auto-Provisioned Delegated Administration Manage Shared Infra Manage 1000s of Volumes Scheduled Upgrades	OpenStack Cinder, VASA	Cloud Storage - SDS (Public, Private, Hybrid)
(3-TIER APPs) 	CERTIFIED SYSADMIN(S) 	Supervised Provisioning Manage Storage Upgrades! Manage 100s of Volumes Managed Upgrades	NFS,iSCSI	Enterprise Storage (SAN, NAS, ScaleOut, DP, DR, Backup, Compression, Dedup)
(LOCAL APPs) 	GEEK 	Format Disks and Use. Manage Few Disks	XFS, ZFS, LVM, EXT4 	POSIX (IO, Checksum, Snapshots) Storage Medium (Disk, Flash, RAM)

Happy days!

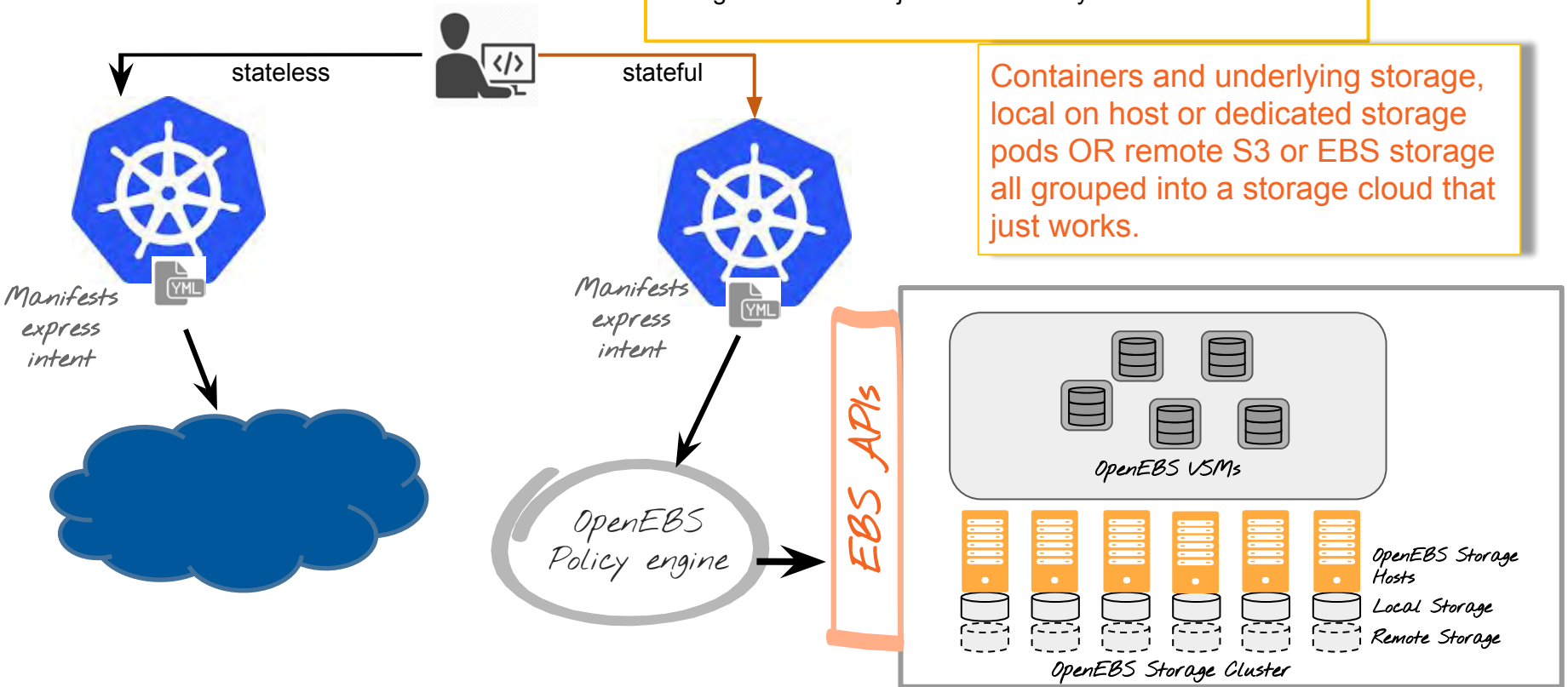


Painful persistence



Desired state of state

No changes to DevOps workflow even for containers requiring persistence. Users manifest their intent and the storage and storage controllers adjust automatically as needed.



Containers and underlying storage, local on host or dedicated storage pods OR remote S3 or EBS storage all grouped into a storage cloud that just works.

Architecture and Design

- Powered by Linux, Go and OpenSource
- Built and Delivered as Containers / Micro-services
- Longhorn, Gotgt, Kubernetes, Consul

Design Goals and Constraints

Fault tolerant and secure by default

Low entry barrier, easy to setup

Storage optimized for Containerized Applications

Horizontally scalable to millions of Containers

Seamless integration into existing private and public cloud environments

Non-disruptive upgrades

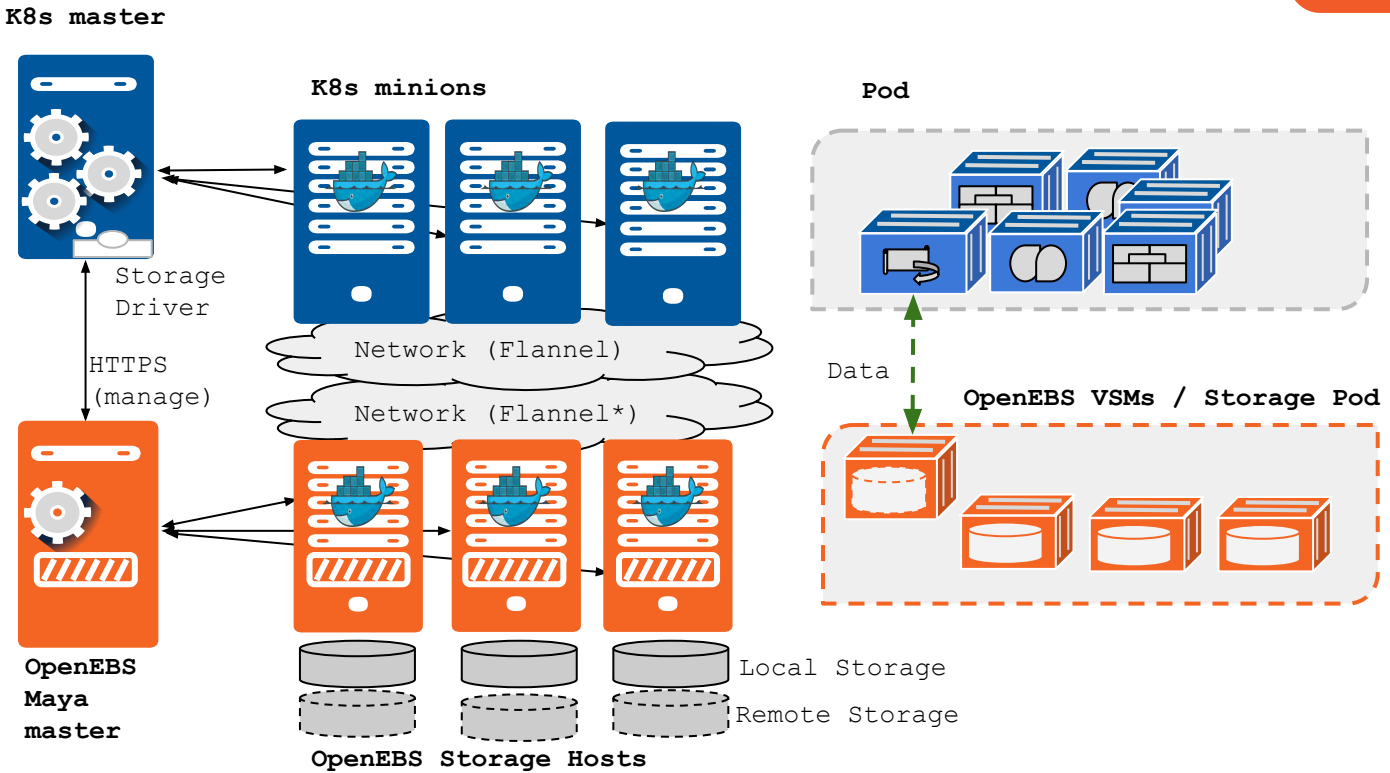
Developer and Operators Friendly

Completely OpenSource (Apache license)

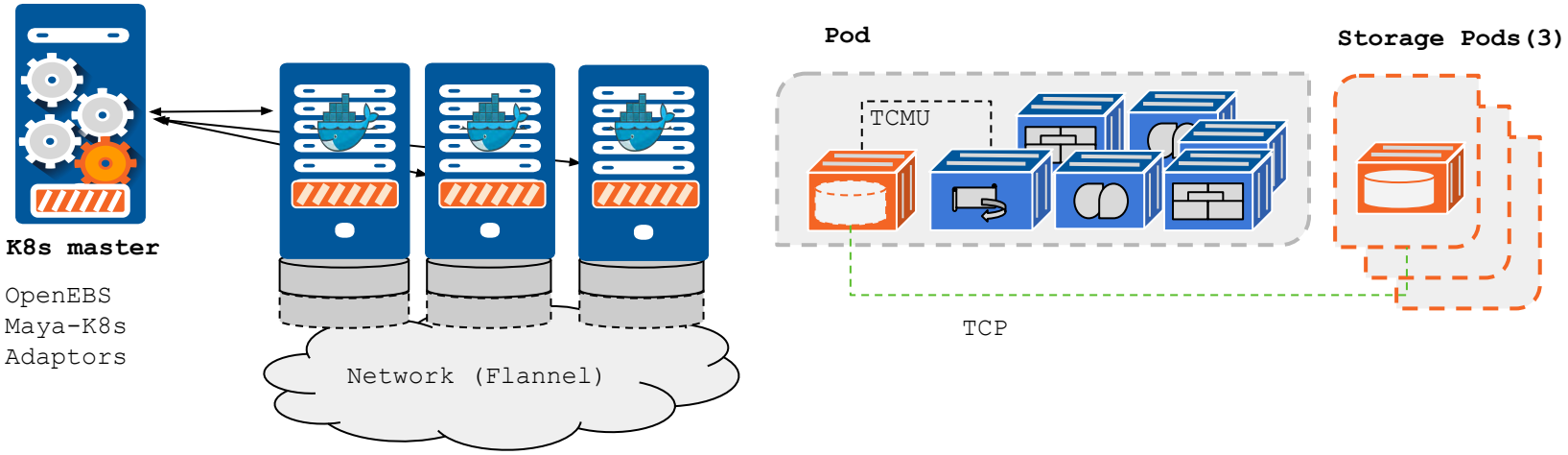
Microservices based

DevOps architecture

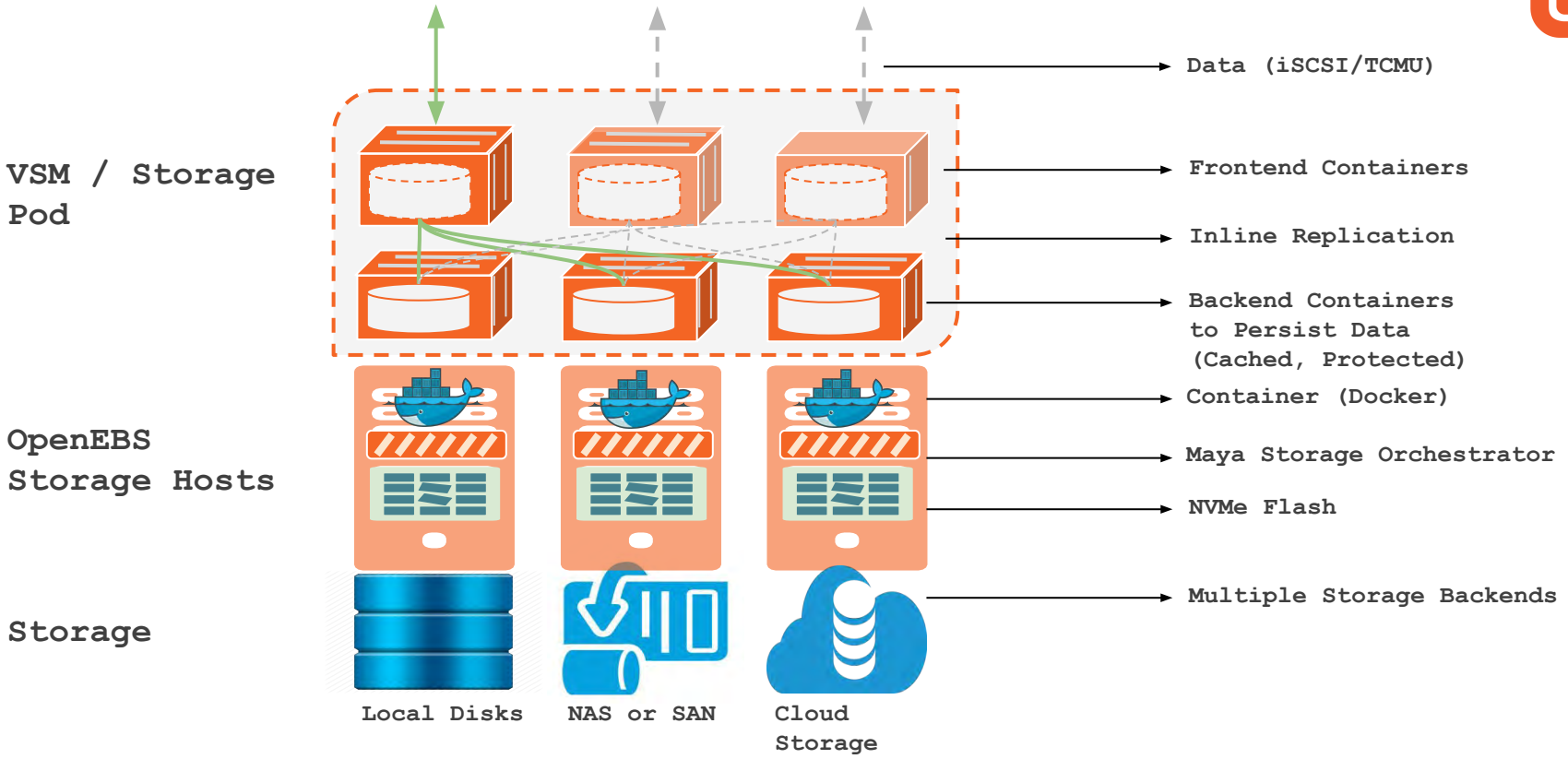
Overview & Terminology



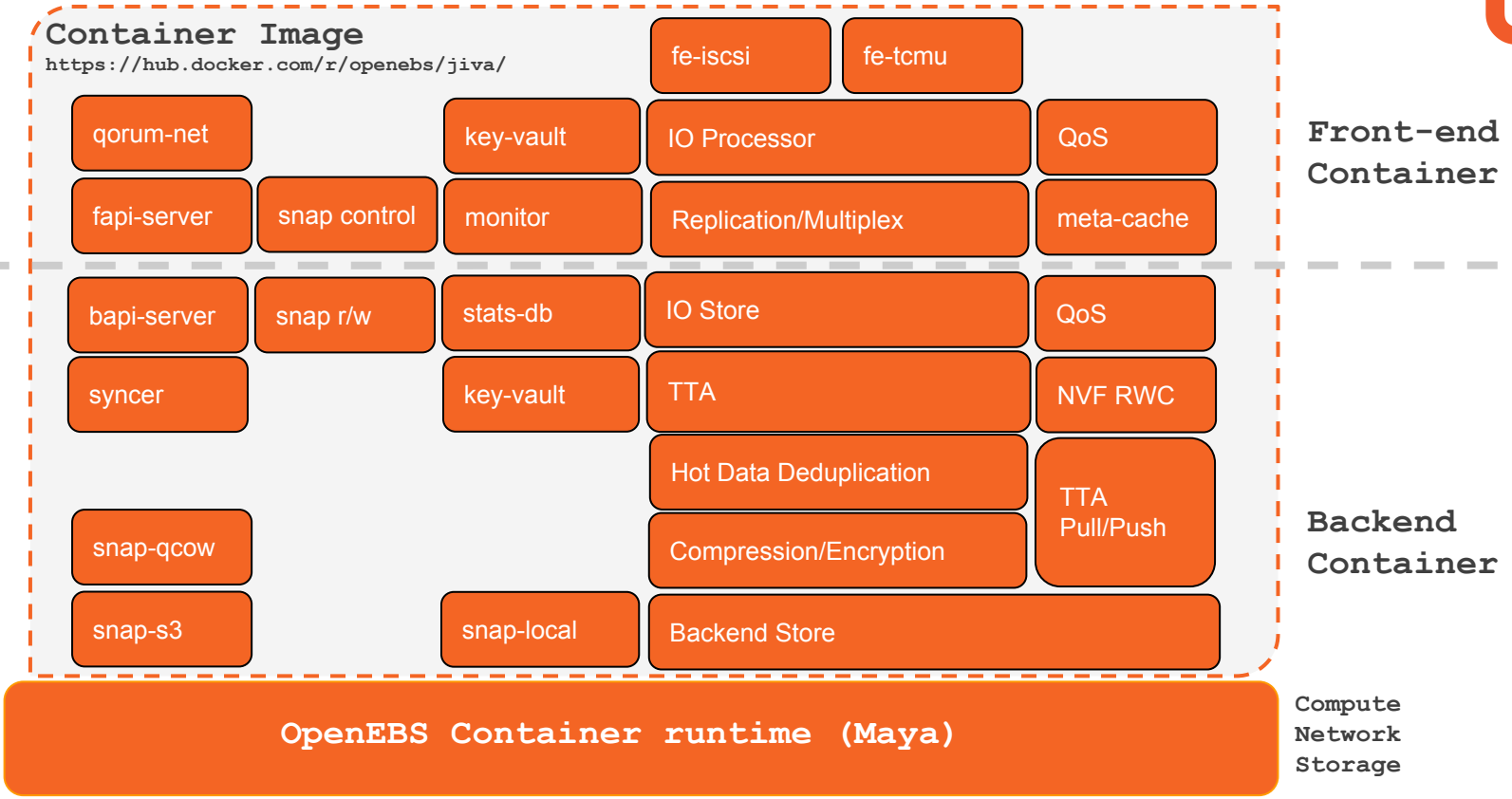
Deployment - Hyper-Converged



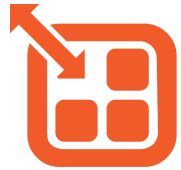
VSM - Storage in Containers



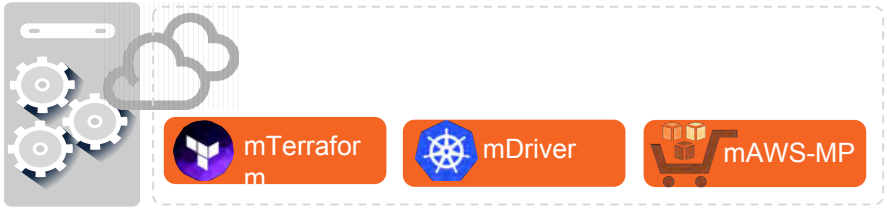
Jiva - Containerized Storage Image



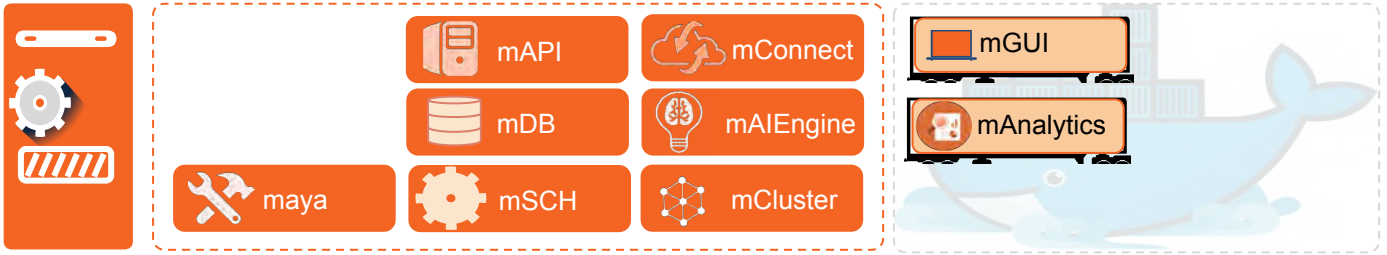
Maya - Container Storage Orchestration



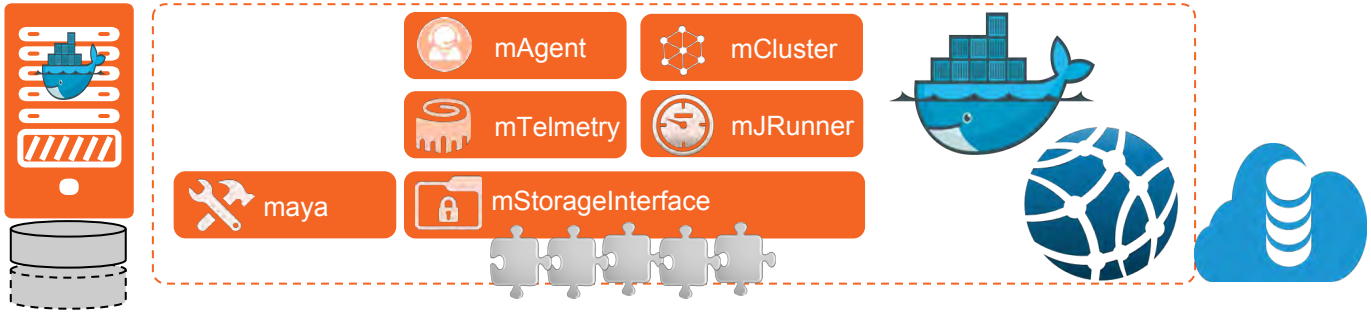
Integrations



OpenEBS Maya Master



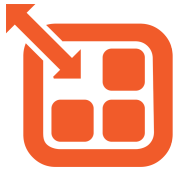
OpenEBS Storage Host



Storage Internals

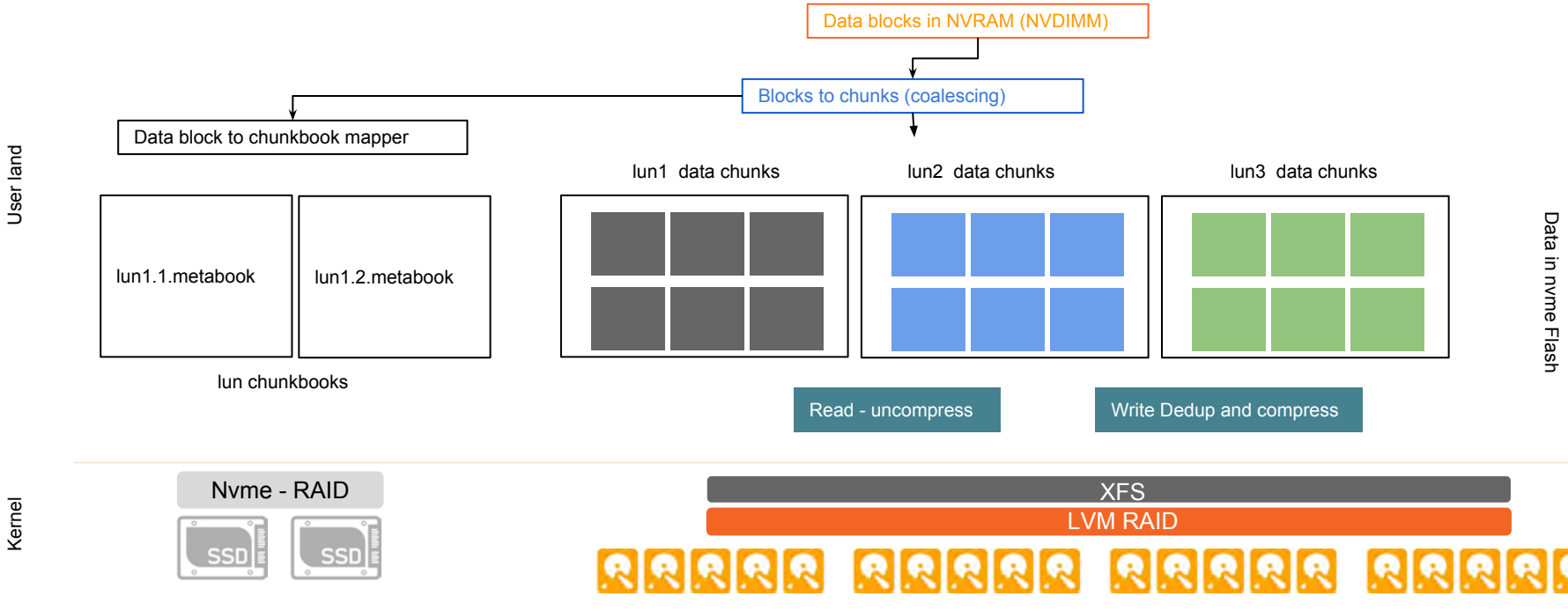
- Capacity Management
- QoS
- Access - iSCSI, TCMU
- Snapshot / Restore (S3)
- Backup / Migration
- Caching/Tiering
- Replication / Rebuild

OpenEBS - Core differentiations

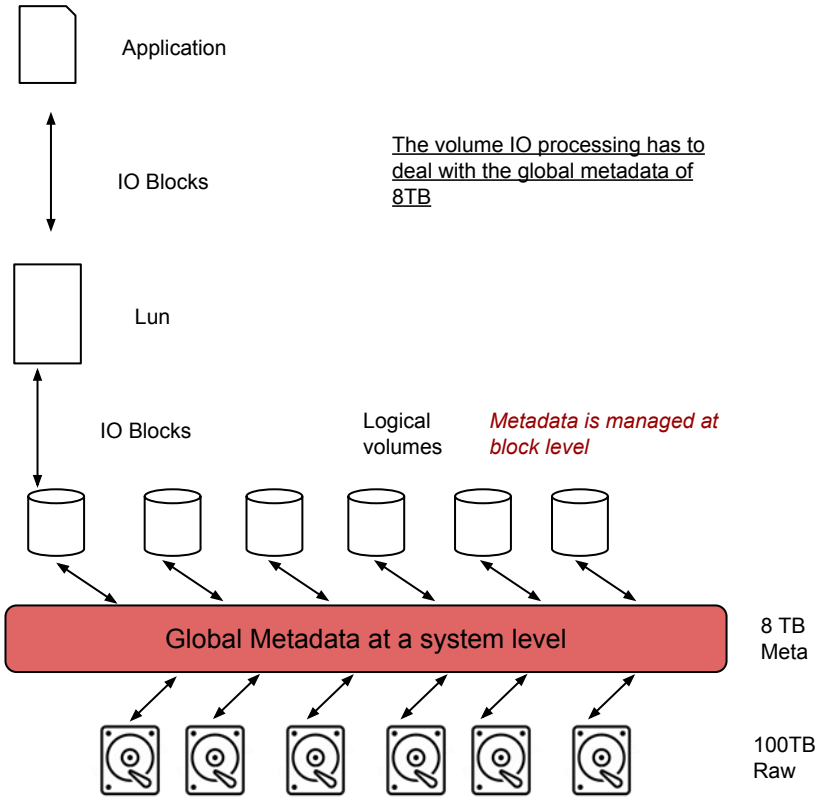


- The block storage software is made into a micro service
- The ‘micro service’ has its own block protocol stack, tiering engine, QoS engine and ML prediction capability
- The block storage knowledge is maintained on a per-volume basis. The data of each volume is divided into cold-data and hot-data. Cold-data resides on NVMe-Flash or on 3DX-Memory. Hot-data resides in slower disks / SAN/ Cloud-Storage/S3
- The metadata knowledge also is maintained at a volume level (not the entire storage). This saves us from the issue of huge-metadata-sifting at scale. The traversal through meta-data depends on the “size of the volume” and not on the “number of volumes”.
- Within the volume the meta data is not managed at “block-level” but at “chunk-level”. Typical block-size is 4KB and Typical chunk-size is 4 MB. This results in the huge reduction of metadata size of the block-volume that needs to be maintained.
- Checksum - One of the important metadata is checksum. OpenEBS guarantees bit-rot protection through the use of checksums. The checksums are managed at a chunk level only on Cold-Data. The checksums are not managed on hot-data, the blocks go in and out of chunks on the hot-data without the need of checksum calculation on the fly.
- Deduplication-while-tiering: Deduplication has capacity benefits but kills performance (either inline or offline). But in OpenEBS, we do this while moving the data from hot-to-cold tiers. In effect, the benefits of deduplication without the performance penalty.

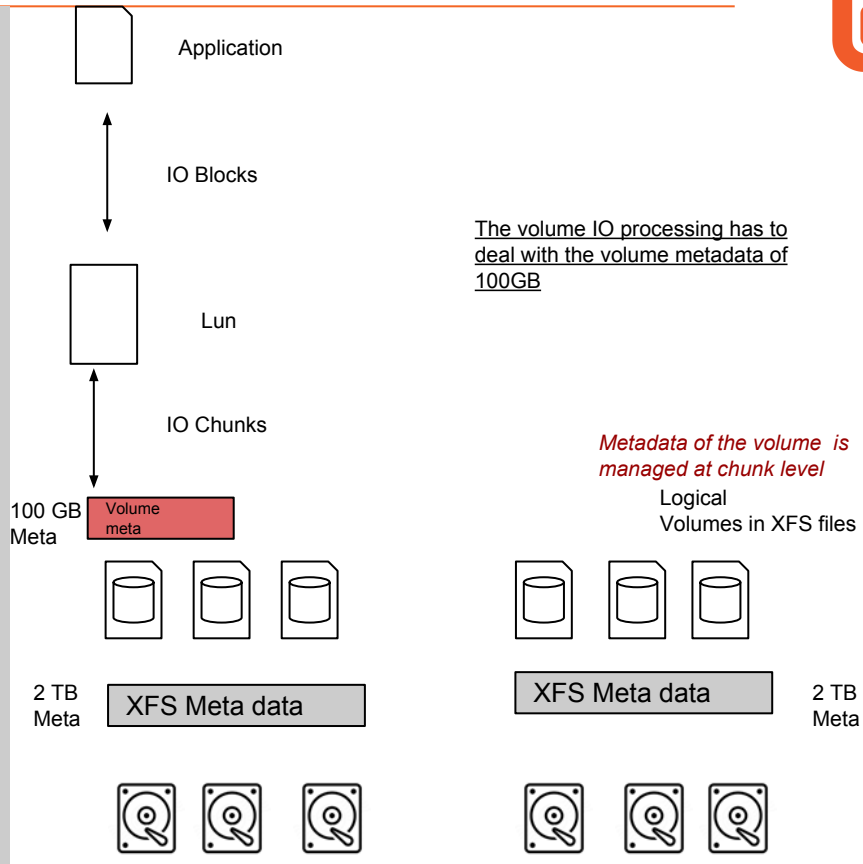
Cchchchcunking



OpenEBS - Metadata at scale is not an issue

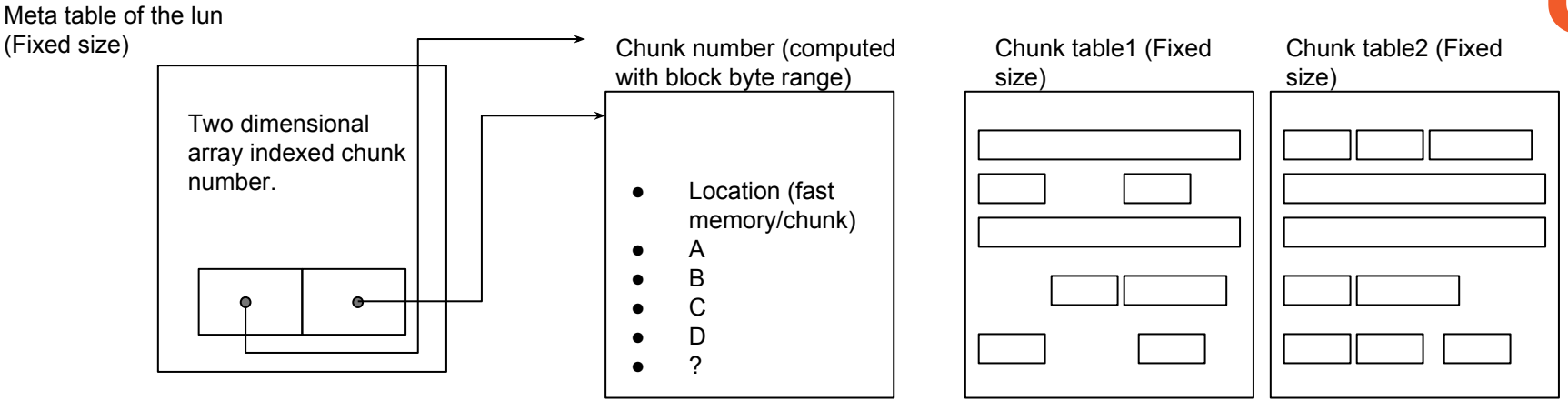


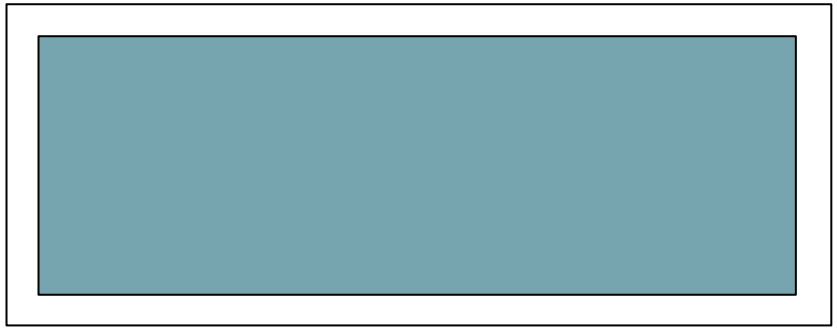
The volume IO processing has to deal with the global metadata of 8TB



The volume IO processing has to deal with the volume metadata of 100GB

Storage Data format





Block layout of the lun inside the xfs file

/xfs/outside/lun1 (sparse file)

Storage Interface

- HardDisks
- SAS/SATA Flash
- NVMe Flash
- PCIe Flash
- S3
- Cloud Block Storage

VSM Network Interface

- Host Networking
- VLANs / IPSpaces

Ease of Configuration

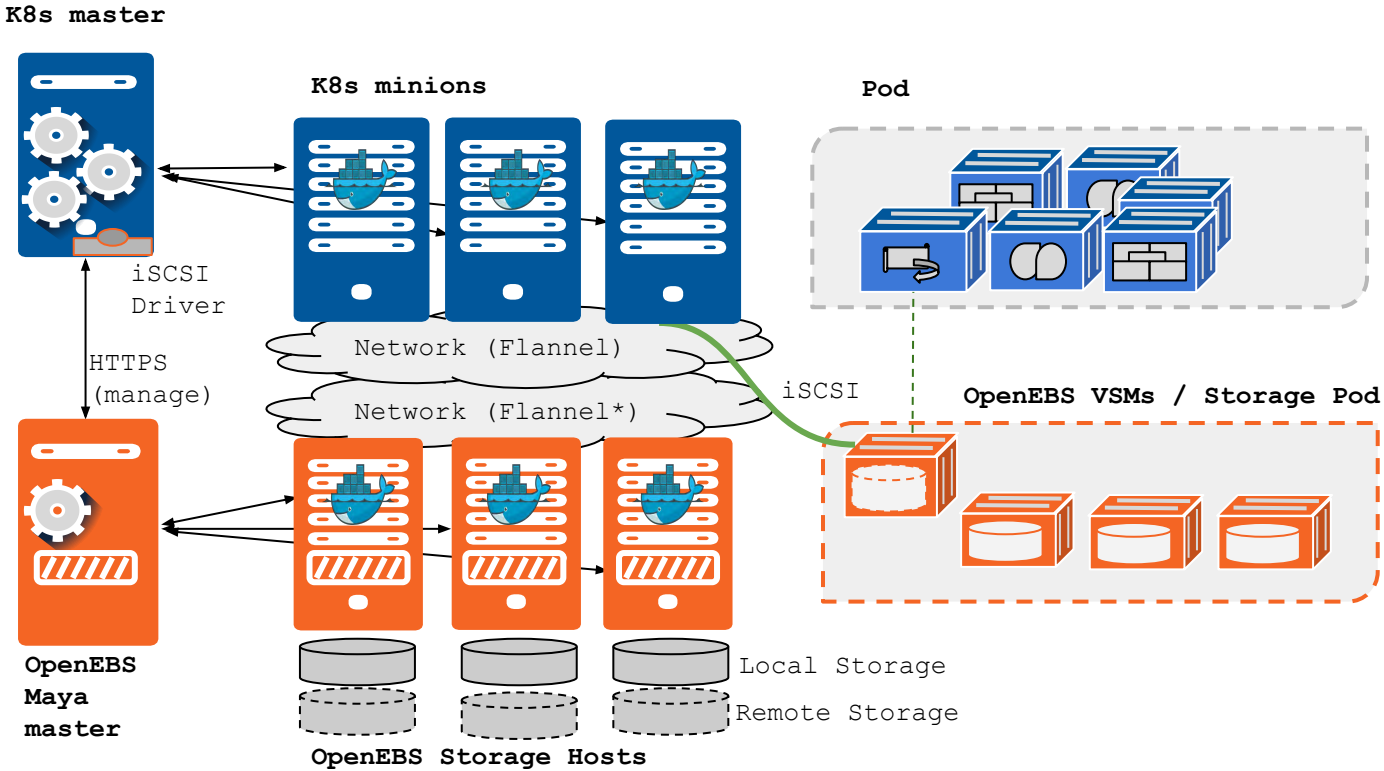
- VSM Configuration Spec
- Infra Spec
- Integrate into K8s / EBS Compatible

Integration to Orchestration

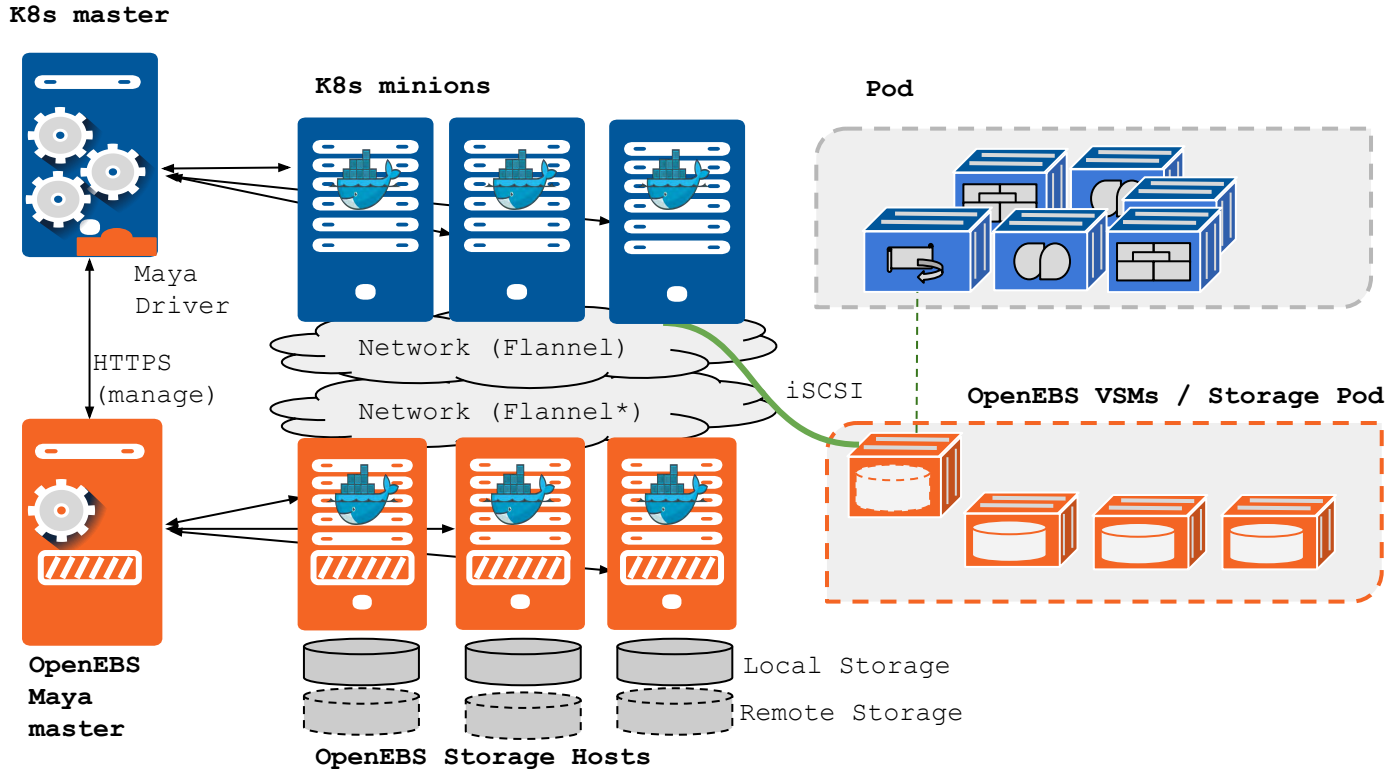
Options to consume the storage by containers:

- iSCSI Driver (Pre-provisioned)
- Maya Volume Driver
- Integrated Orchestration

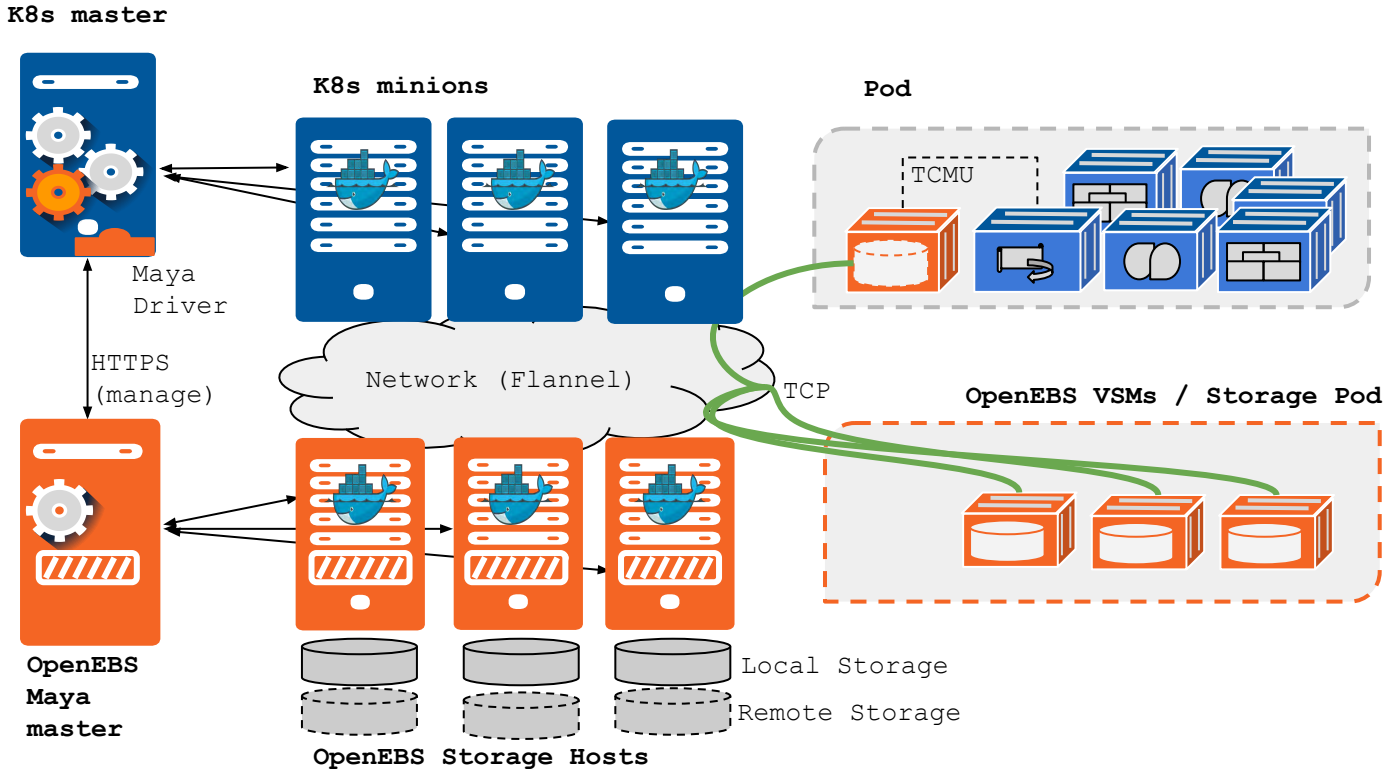
Storage Connectivity - iSCSI Claims



Storage Connectivity - Maya Driver



Storage Connectivity - Shared Orchestration



Resiliency and Fault Tolerance

- Scaleout
- Blue-Green Upgrades - Infra
- Rolling Upgrades - VSMs
- High-Availability

Security

- Data Security
- Encryption
- Secure Delete

Telemetry

- Monitoring and Troubleshooting
- Analytics

Performance

- IO Latency
- Provisioning
- Analytics

Scale

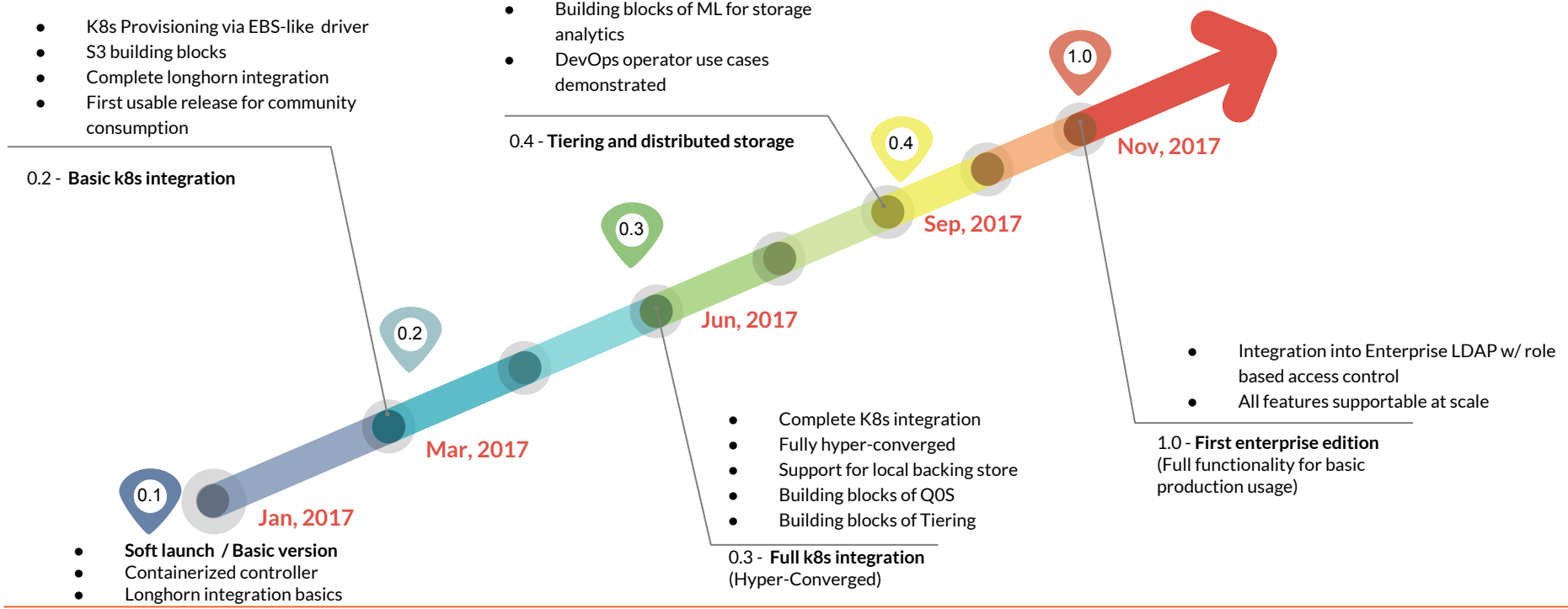
- Capacity
- Number of Volumes

Deployment Flexibility

OpenEBS Deployment Options for:

- Dedicated Storage (External)
- Hyper-converged
- Hybrid-Cloud (AWS)

OpenEBS Roadmap



- **Soft launch / Basic version**
- Containerized controller
- Longhorn integration basics

- K8s Provisioning via EBS-like driver
- S3 building blocks
- Complete longhorn integration
- First usable release for community consumption

0.2 - **Basic k8s integration**

- Complete K8s integration
- Fully hyper-converged
- Support for local backing store
- Building blocks of Q0S
- Building blocks of Tiering

0.3 - **Full k8s integration**
(Hyper-Converged)

- Tiering and QoS demonstrated
- Building blocks of ML for storage analytics
- DevOps operator use cases demonstrated

0.4 - **Tiering and distributed storage**

- Integration into Enterprise LDAP w/ role based access control
- All features supportable at scale

1.0 - **First enterprise edition**
(Full functionality for basic production usage)

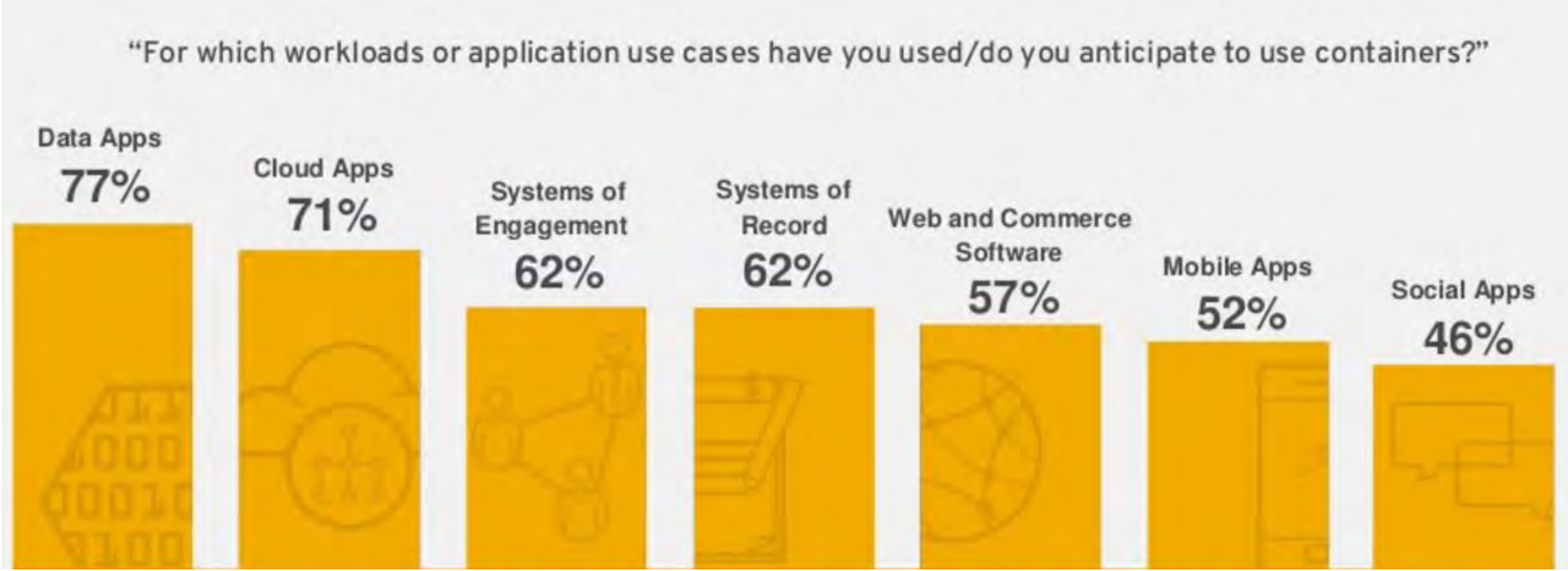
blog.openebs.io

https://github.com/openebs

Join the
community
#slack
slack.openebs.io

@openebs

Stateful containers?!



http://www.slideshare.net/Red_Hat_Storage/red-hat-storage-day-boston-persistent-storage-for-containers

Stateful containers?!

http://www.slideshare.net/Red_Hat_Storage/red-hat-storage-day-boston-persistent-storage-for-containers