Think like a storage architect, in four questions

GDG Cloud London, 18 April 2018

Cheryl Hung, Product Manager

Cheryl (@oicheryl)

Ex-Google software engineer

Product manager, StorageOS

Cloud Native London meetup

Objectives

- Why is state so tricky?
- How does storage work with Kubernetes?
- How do I choose a storage solution?

Objectives

- Why is state so tricky?
- How does storage work with Kubernetes?
- How do I choose a storage solution?

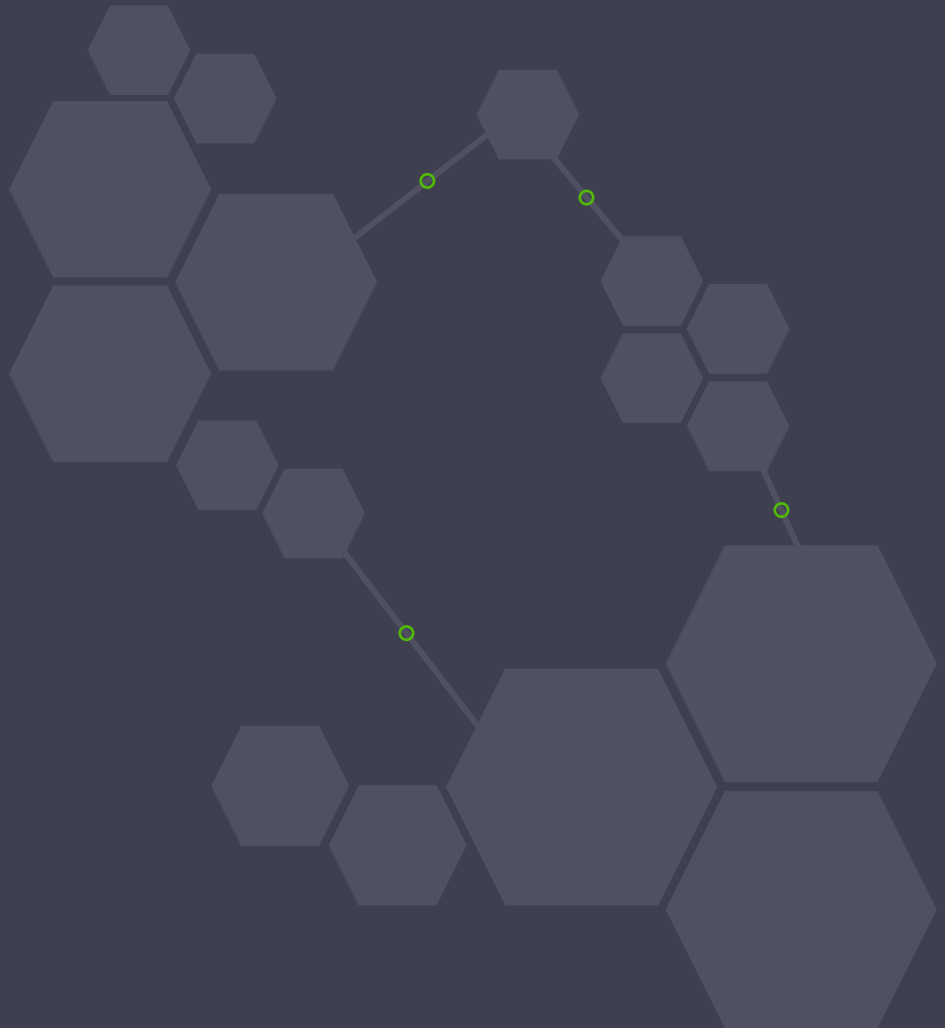
Anti-objective:

- Should I use a database/message queue/key-value store... for my app?

Audience poll

STORAGEOS

Why is state so tricky?



Why do I need storage?

Why do I need storage?

First challenge: No pet storage

Second challenge: Data needs to follow

Third challenge: Humans are fallible

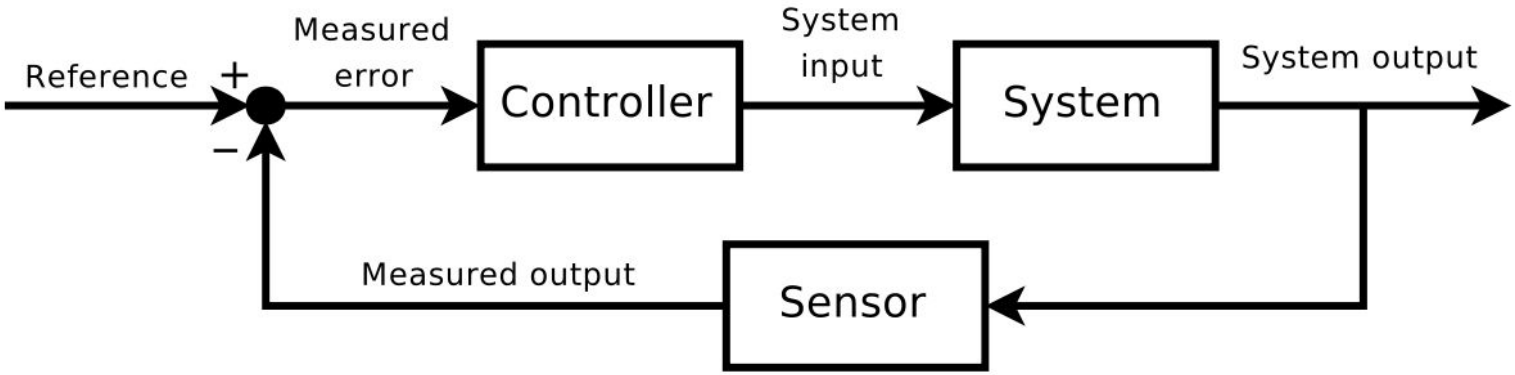
STORAGEOS

How does storage work with Kubernetes?

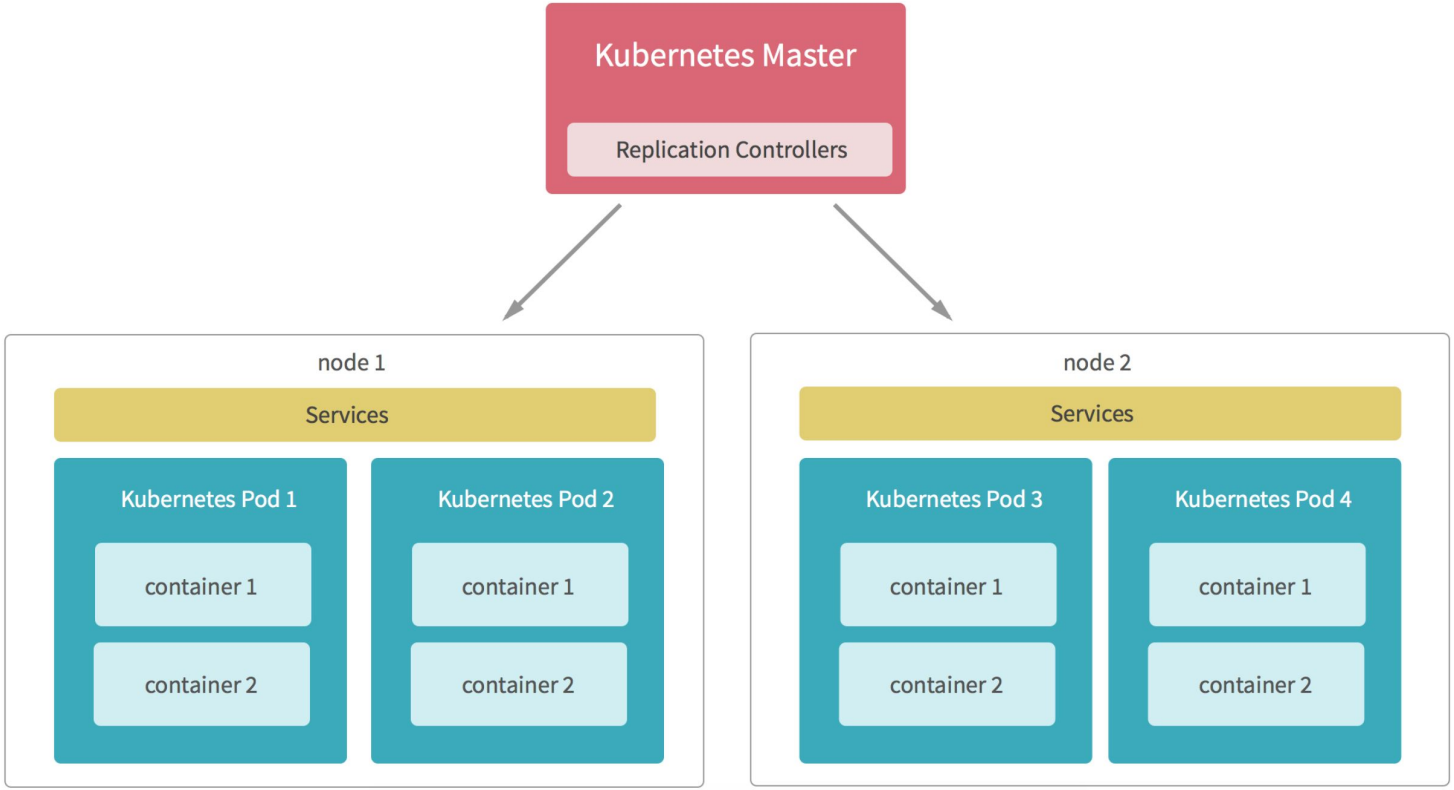
Extremely quick intro to Kubernetes

- An open source container orchestrator platform, originally developed by Google
- One of the fastest moving projects in open source; “The Linux of the cloud”
- Abstracts away infrastructure in a cluster

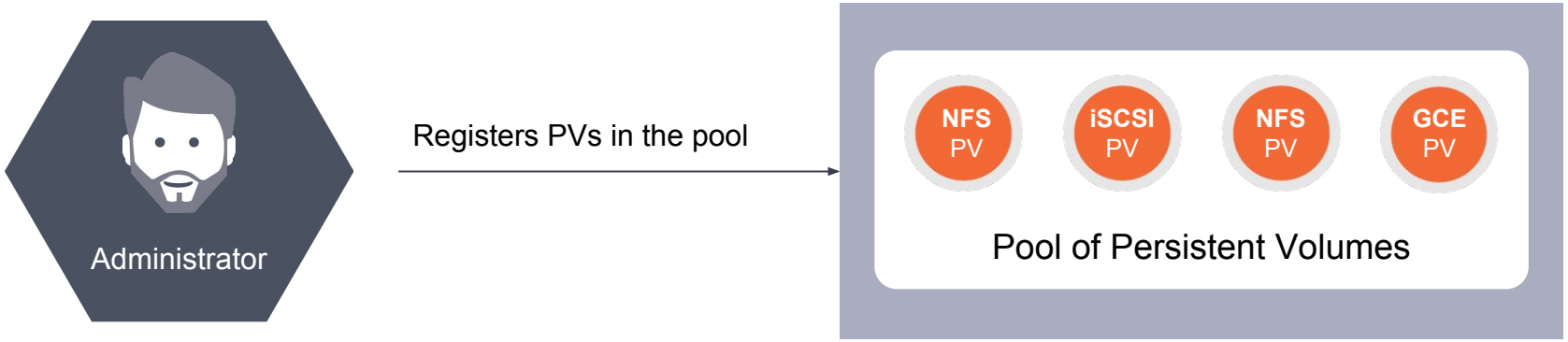
“Always run my application (packaged as a container/pod) with four replicas”



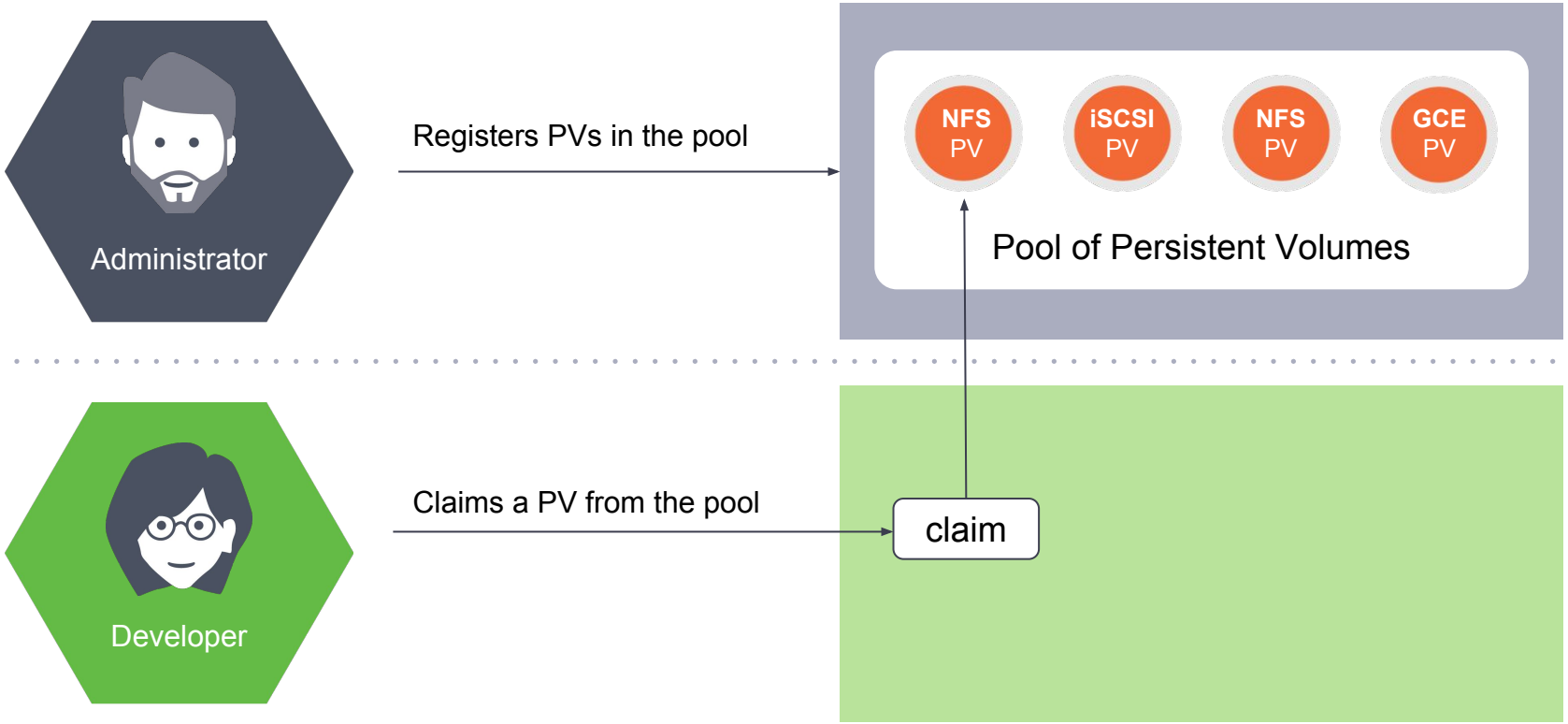
Extremely quick intro to Kubernetes



Kubernetes Storage Model: Persistent Volumes and Claims

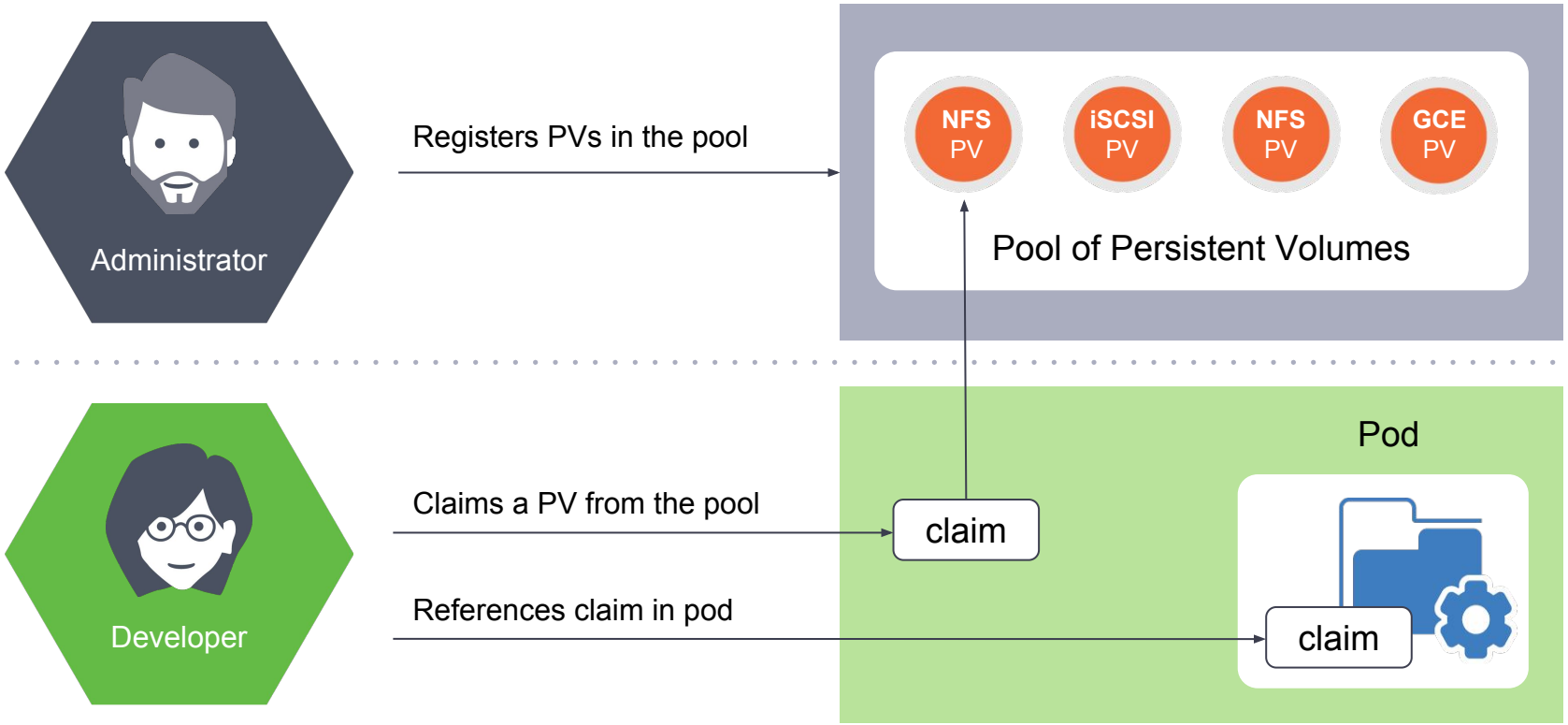


Kubernetes Storage Model: Persistent Volumes and Claims

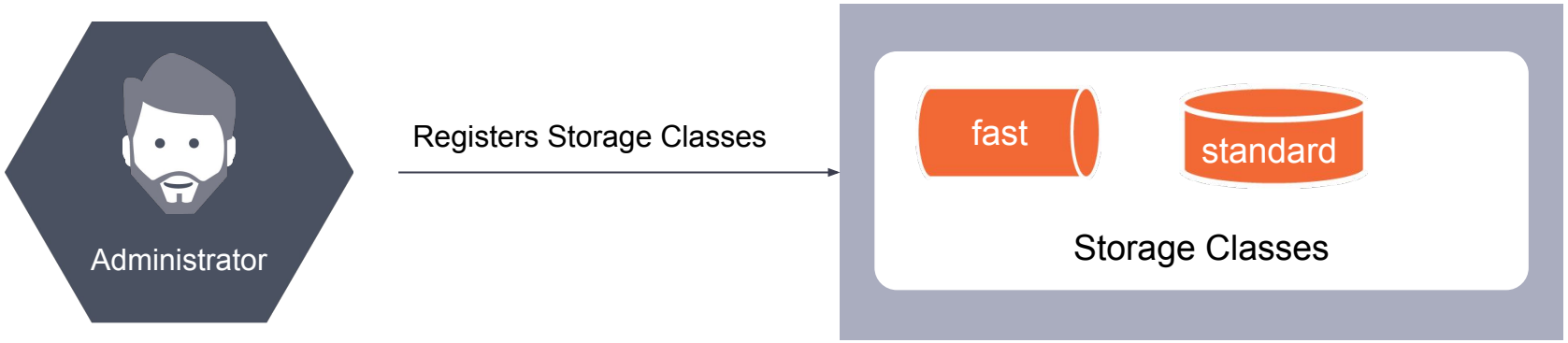


Kubernetes Storage Model: Persistent Volumes and Claims

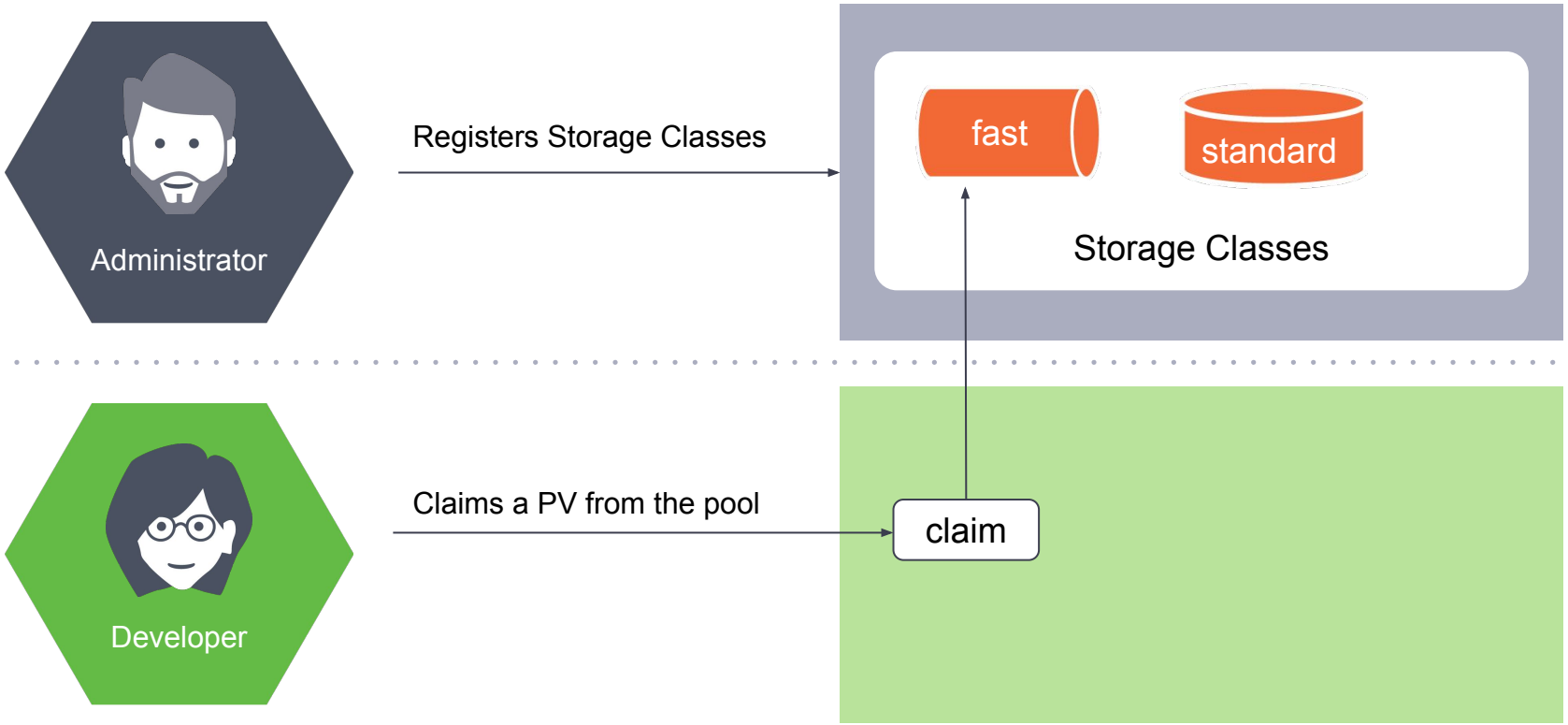
@oicheryl



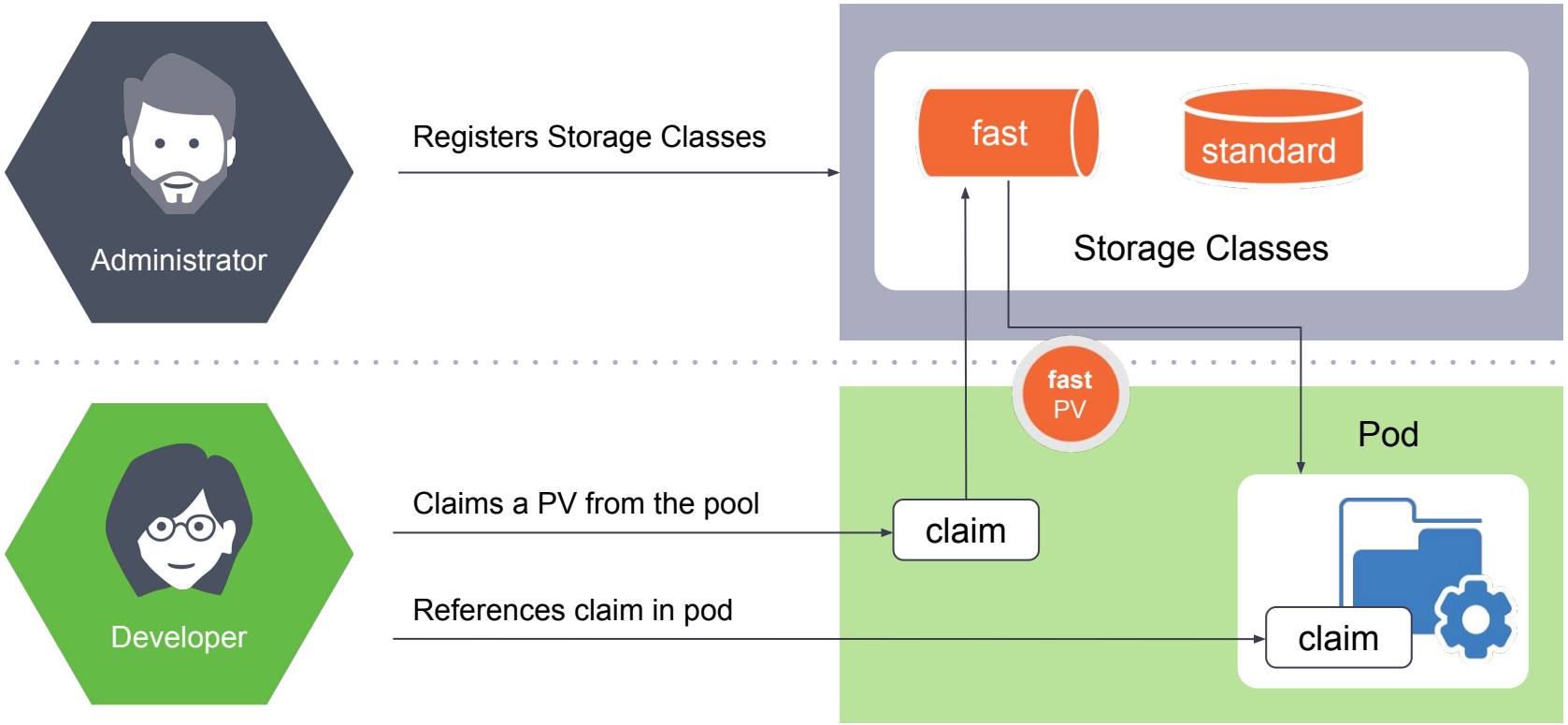
Dynamic provisioning with Storage Classes



Dynamic provisioning with Storage Classes



Dynamic provisioning with Storage Classes



STORAGEOS

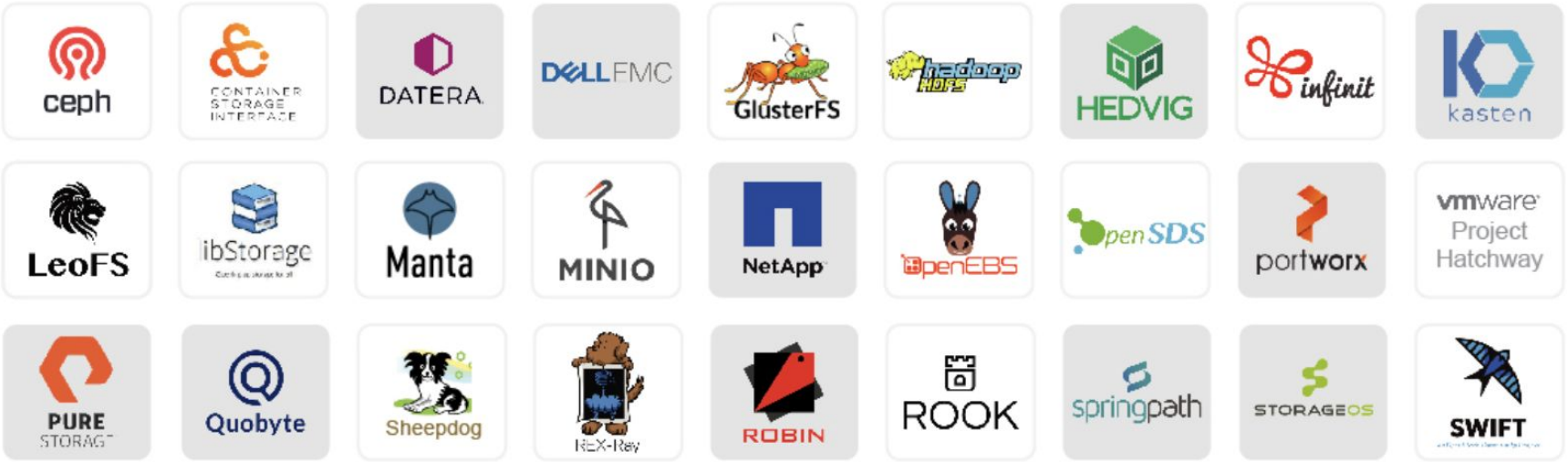
Which storage solution should I choose?

Kubernetes storage plugins

Volume Plugin	Internal Provisioner	Config Example
AWSElasticBlockStore	✓	AWS
AzureFile	✓	Azure File
AzureDisk	✓	Azure Disk
CephFS	-	-
Cinder	✓	OpenStack Cinder
FC	-	-
FlexVolume	-	-
Flocker	✓	-
GCEPersistentDisk	✓	GCE
Glusterfs	✓	Glusterfs
iSCSI	-	-
PhotonPersistentDisk	✓	-
Quobyte	✓	Quobyte
NFS	-	-
RBD	✓	Ceph RBD
VsphereVolume	✓	vSphere
PortworxVolume	✓	Portworx Volume
ScaleIO	✓	ScaleIO
StorageOS	✓	StorageOS

From the CNCF Landscape

Cloud-Native Storage



Cloud Storage

	Azure	Google	AWS
Object Storage	Azure Blob Storage	Google Cloud Storage	Amazon Simple Storage Service (S3)
Virtual Machine / Block Storage	Azure Page Blobs / Premium Storage	Persistent Disk	Amazon Elastic Block Storage (EBS)
File Storage	Azure File Storage	⊘	Amazon Elastic File System (EFS)
Long Term Cold Storage	Azure Cool Storage	Google Coldline Storage	Amazon Glacier
Hybrid / Gateway Storage	Azure Storsimple	⊘	AWS Storage Gateway

Meet Jane

- A DevOps engineer at a media company
- Migrating client Wordpress websites into Kubernetes
- Needs to decide what storage to use

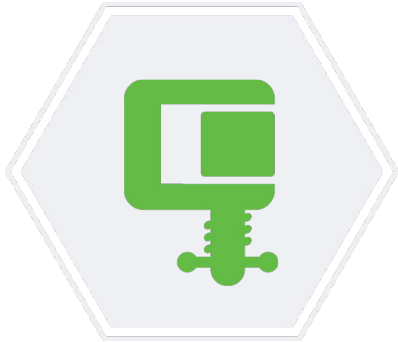
The four questions

1. What is my **use case**?
2. What are my **performance requirements**?
3. How should developers **access** storage?
4. Where is the storage **deployed and managed**?

1. What is my use case?

App binaries

App data



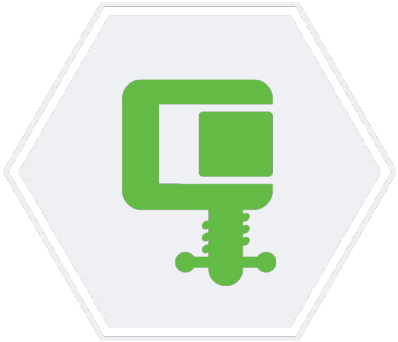
Config

Backup

2. What are my performance requirements?

App binaries
Ephemeral

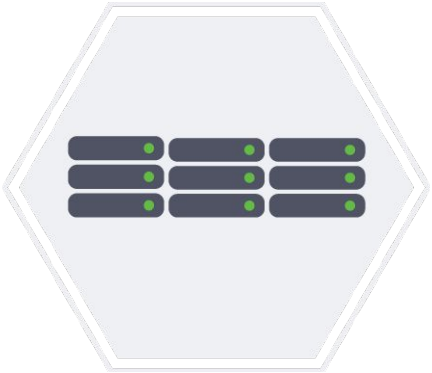
App data
Latency, availability, performant



Config
Shared

Backup
Cost efficient, cloud

3. How should developers access storage?

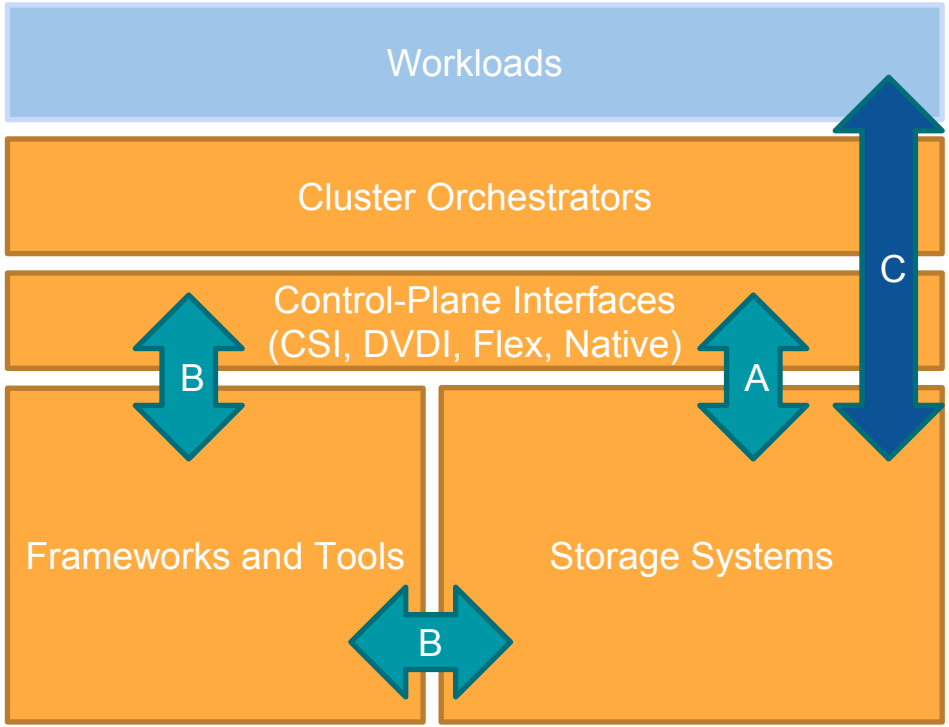


Block
Fixed-size ‘blocks’ in a rigid arrangement – ideal for enterprise databases

File
‘Files’ in hierarchically nested ‘folders’ – ideal for active documents

Object
‘Objects’ in scalable ‘buckets’ – ideal for unstructured big data and archiving

4. Where is the storage deployed and managed?



- CO supports one or more **Interfaces** to interact with the Storage System
- Storage System can **(A)** support control-plane interface API directly and interact directly with the CO or can **(B)** interact with the CO via an **API framework layer** or other **Tools**.
- Storage system must support the ability to provision and consume (C) volumes through a standard interface to be considered **Interoperable**
- Workloads interact (C) with storage systems over various data-plane methods

Jane’s storage requirements

- Postgres database for application data
- Database location, credentials
- Database and website backups
- User uploaded media

Database location and credentials

1. **Use case?** Configuration
2. **Performance requirements?** Shared across instances
3. **Access?** Kubernetes provides Secrets for sensitive data such as passwords, and ConfigMap for arbitrary config. Both can be accessed by the application through environment variables
4. **Deployed and managed?** Tight integration with Kubernetes

User uploaded media

1. **Use case?** Shared media
2. **Performance requirements?** Large blobs of data, shared across pods
3. **Access?** Shared filesystem
4. **Deployed and managed**?

Cloud: Managed NFS, or object store if the app can support it

On prem: Distributed FS (not NFS)

Database and website backup

1. **Use case?** Backup and archival
2. **Performance requirements?** Durability, cost, snapshots
3. **Access?** Object store
4. **Deployed and managed?**

Cloud: Managed object store (Google Cloud Storage), long term cold storage (Google Coldline)

On prem: Object store, NAS

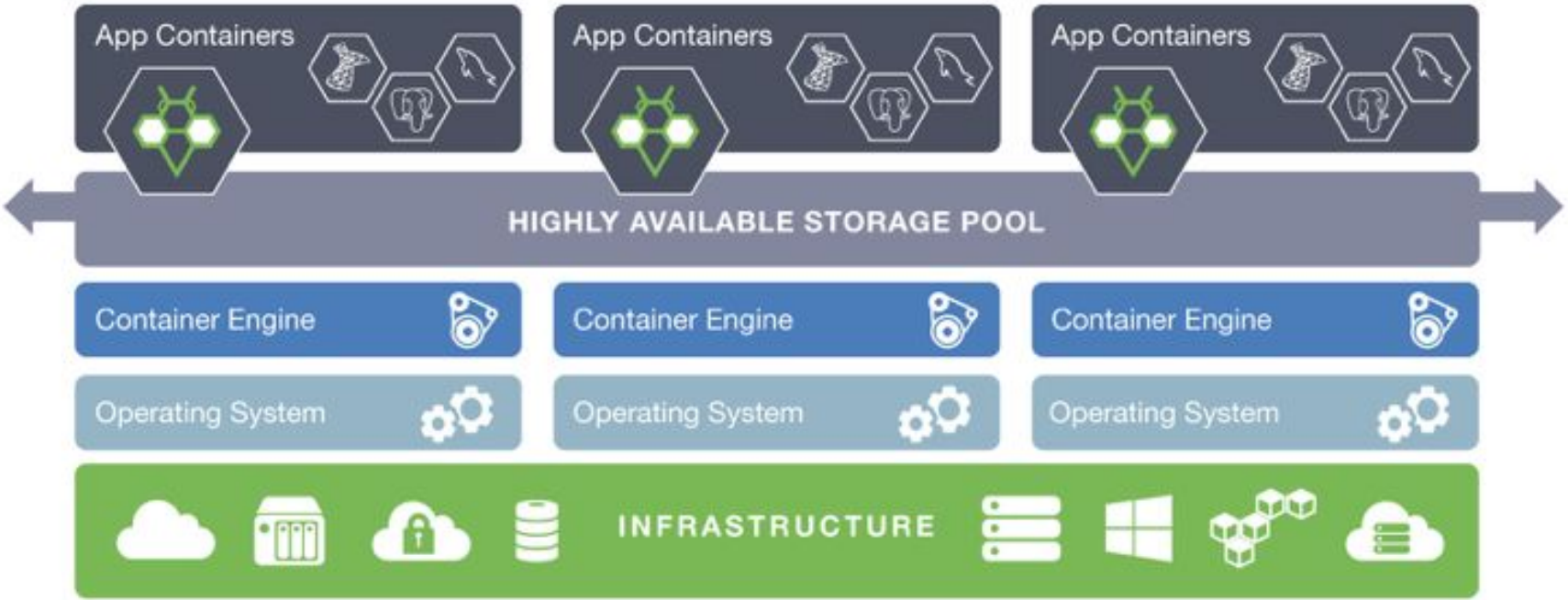
Postgres for application data

1. **Use case?** Transactional database
2. **Performance requirements?** High availability, low latency, deterministic performance
3. **Access?** Database connector
4. **Deployed and managed?**

Cloud: Block storage (but Google persistent disks cannot be detached from running GCE instances) or managed db

On prem: Software defined storage

Software-defined storage



STORAGEOS

Thanks

Slides at oicheryl.com

What is StorageOS?

A software-defined, scale-out storage platform for running enterprise containerized applications in production

What is StorageOS?

Platform agnostic

Horizontally scalable

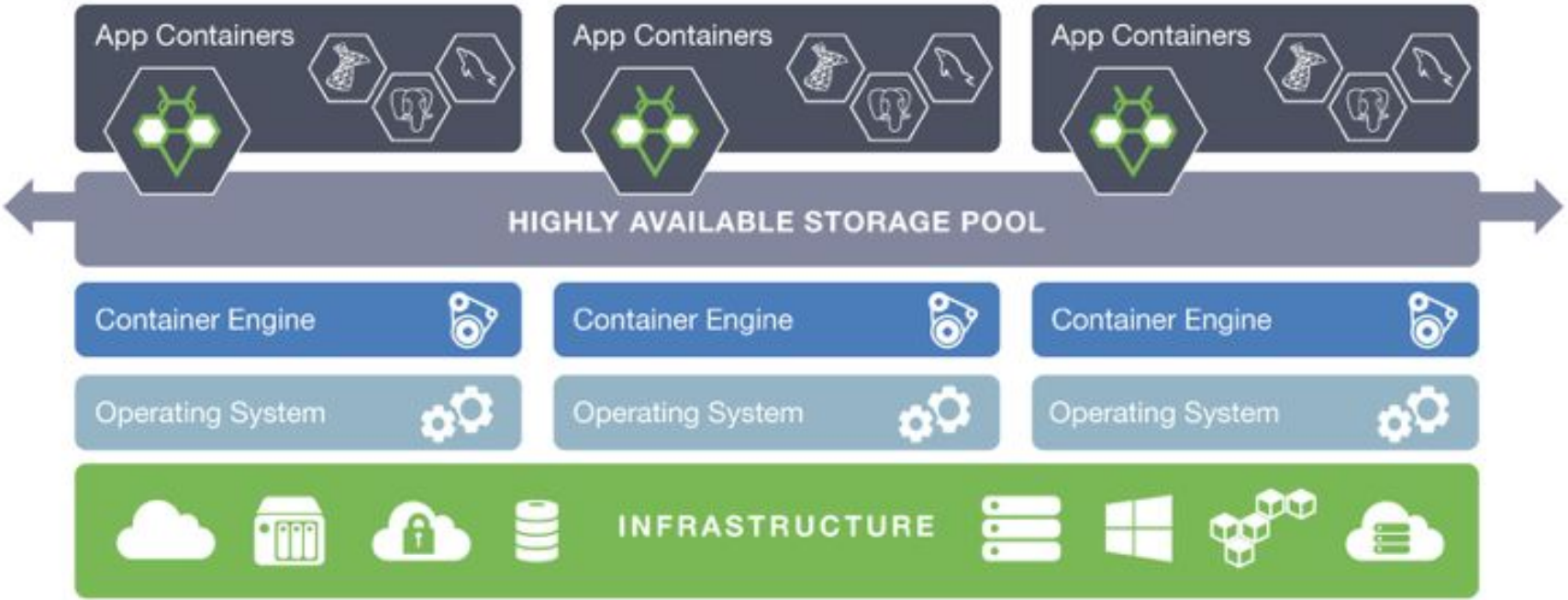
Database (ie. block)

A software-defined, scale-out storage platform for running enterprise containerized applications in production

Docker/K8s integration

High availability

StorageOS architecture



StorageOS architecture

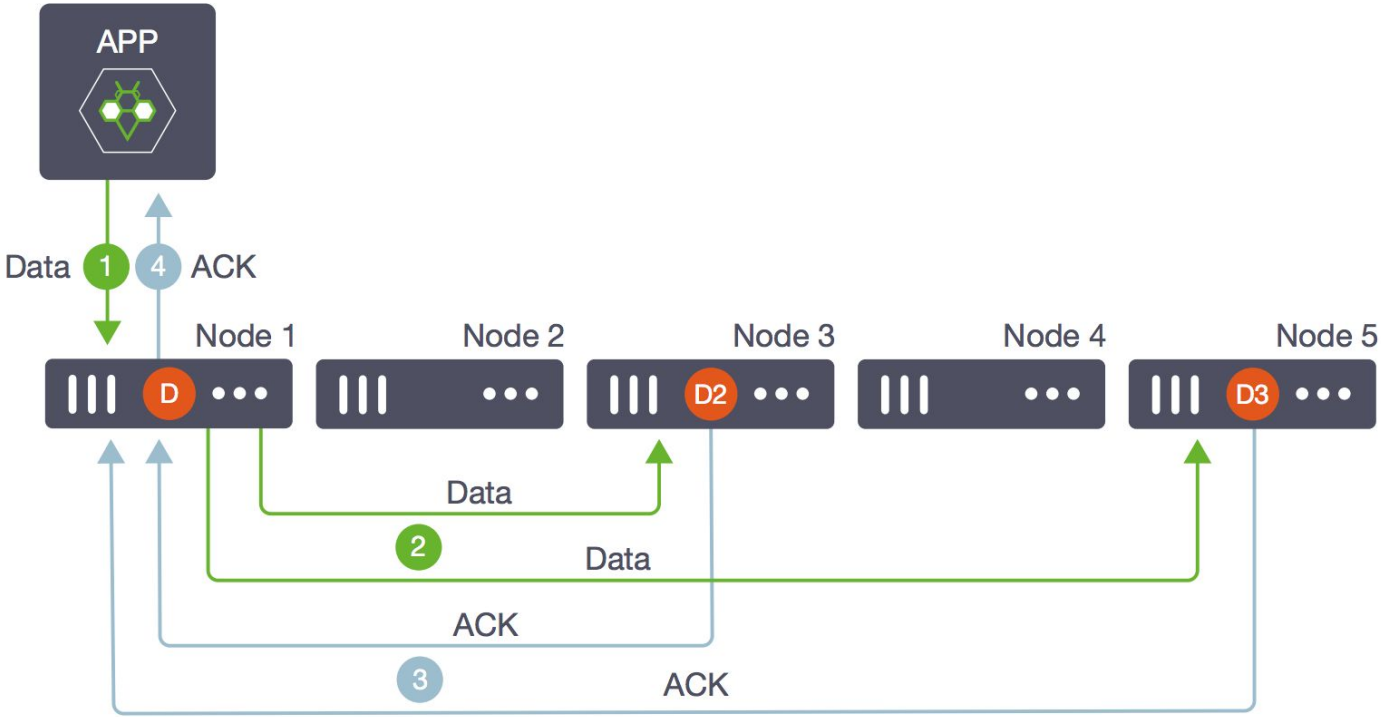
StorageOS is conceptually pretty simple; it's a virtualization layer on top of any commodity or cloud storage. It's deployed as one container per node, similar to a DaemonSet.

1. Nodes contribute local block storage to the storage pool.
2. Virtual volumes (block storage formatted with a standard filesystem) are created using the StorageOS volume plugin.
3. Any pods can mount the virtual volumes from any node. If a pod is rescheduled to a different node, StorageOS simply redirects reads and writes so the pod can continue to access the storage.

StorageOS architecture

It’s designed to scale horizontally by adding more nodes. New nodes contribute their storage into the storage pool, or, if they don’t have storage themselves, can access storage on other nodes.

High availability with StorageOS



High availability with StorageOS

StorageOS uses a hybrid master/replica architecture, where replicas are distributed across nodes.

Replication is very simple in StorageOS. Volume D is created with two replicas. StorageOS creates the replicas (D2, D3) and schedules them to two different nodes (N3, N5). Incoming writes to D are synchronously replicated to D2 and D3, ie. writes are not persisted until acknowledged by both replicas.

If N1 fails, one of D2 or D3 gets promoted to master, providing instant failover and no interruption of service. StorageOS creates and resyncs a new replica on N2 or N4 in the background.

StorageOS resources

Browser-based demo

- my.storageos.com/main/tutorials

Quickstart

- storageos.com/kubernetes

More reading

Download the technical architecture overview at storageos.com/storageos-platform-architecture-overview.

Try out in your browser, with zero downloads or configuration: my.storageos.com/main/tutorials

Full documentation at docs.storageos.com.