Persistent Storage with Kubernetes in Production

Which solution and why?

Cheryl Hung, Product Manager
@oicheryl

Cheryl

Product manager

@oicheryl

STORAGEOS

Objectives

- Why state is tricky in Kubernetes
- How do I evaluate my storage needs?
- How do I choose a storage solution for Kubernetes?

Objectives

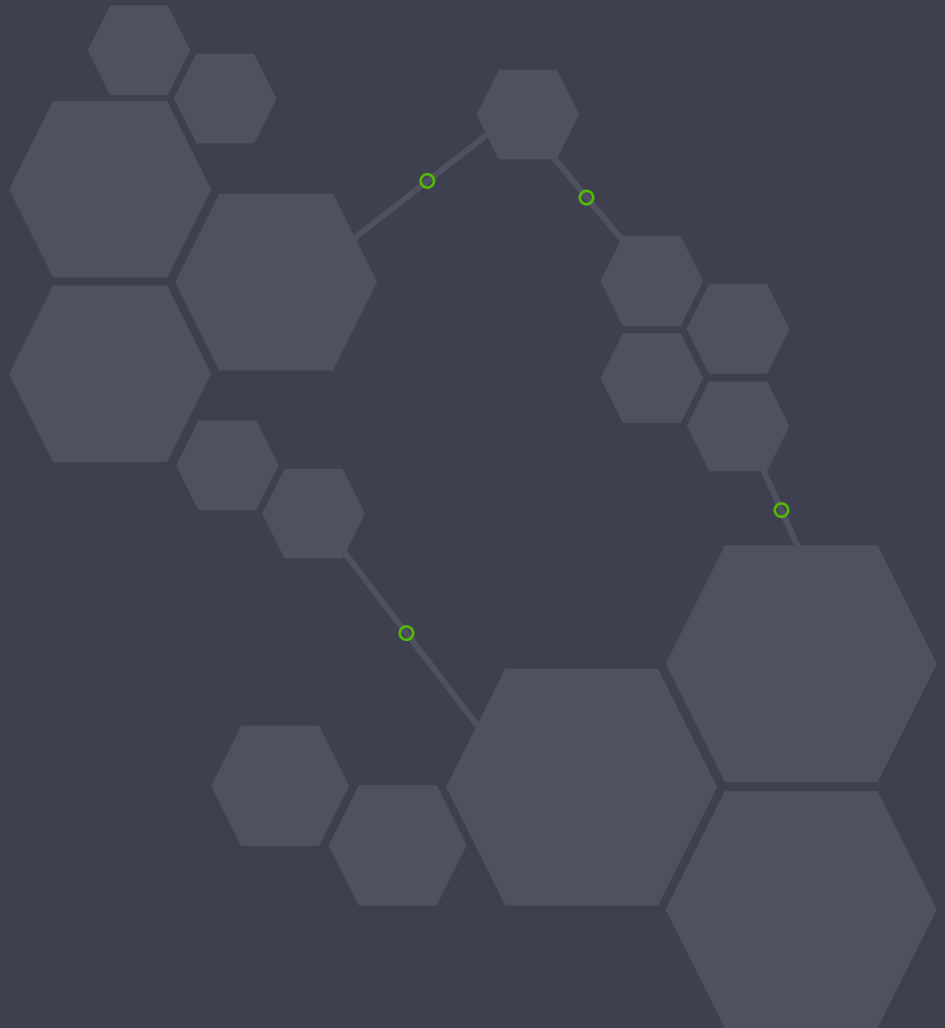
- Why state is tricky in Kubernetes
- How do I evaluate my storage needs?
- How do I choose a storage solution for Kubernetes?

Anti-objective:

- Should I use a database, message queue, NoSQL db… for my app?

STORAGEOS

Why is state so tricky?



Why do I need storage?

Why do I need storage?

Problem 1: No pet storage

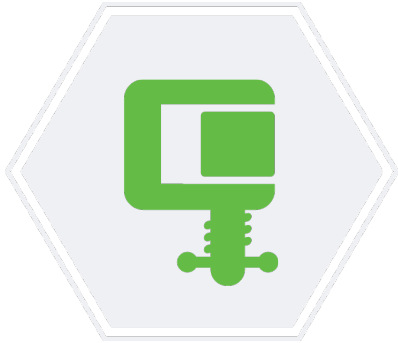
Problem 2: Data needs to follow

Problem 3: Gravity can be a problem

Problem 4: Storage isn't just storage

App Binaries

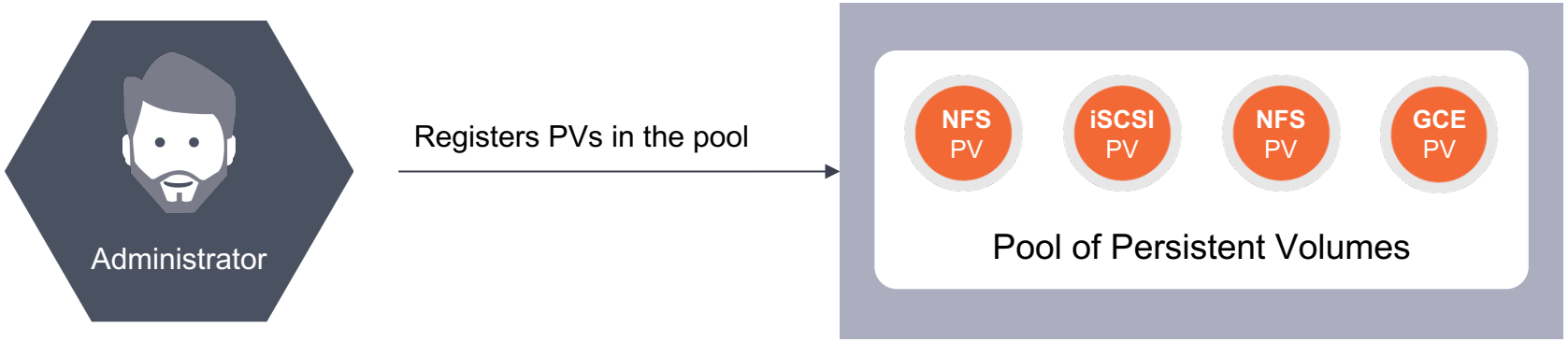
App data



Config

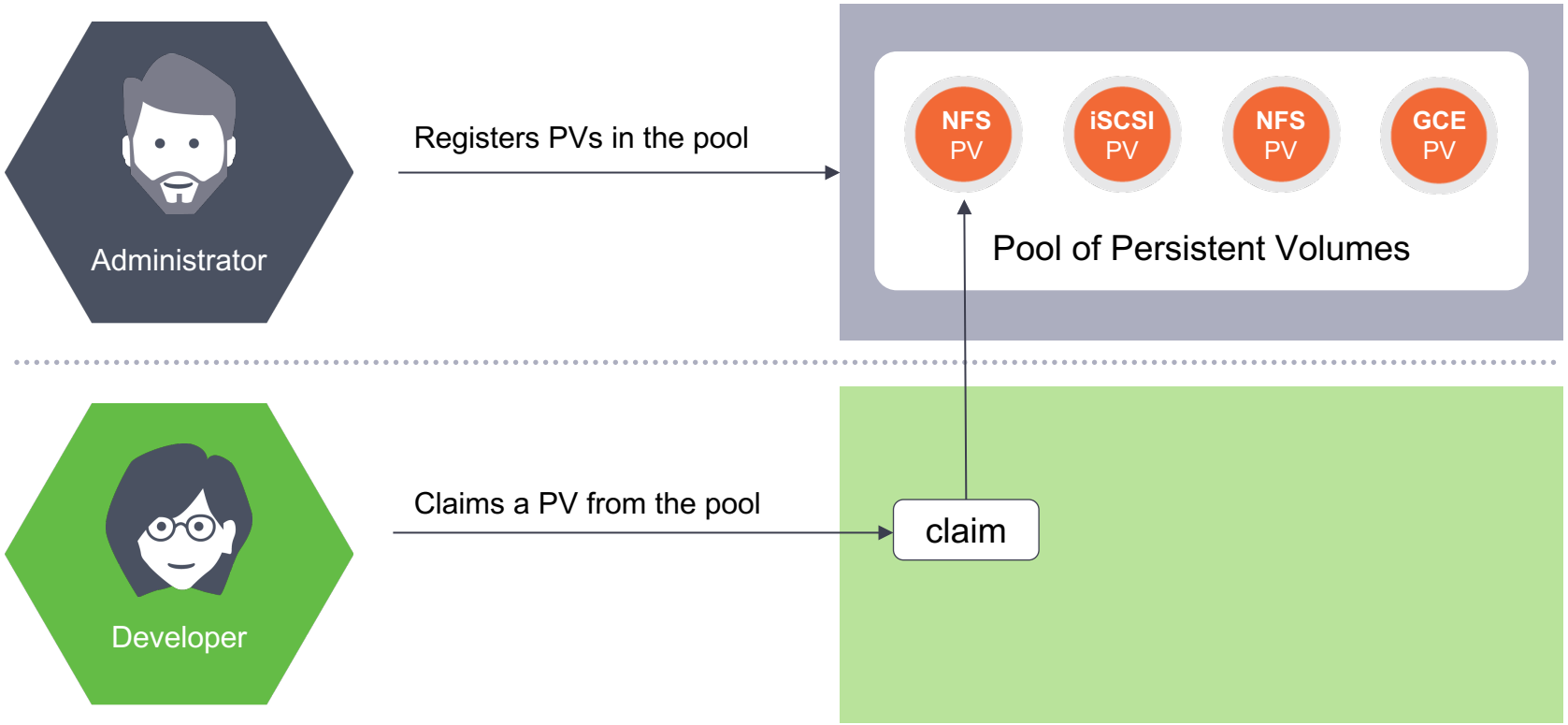
Backup

Kubernetes Storage Model: Persistent Volumes and Claims



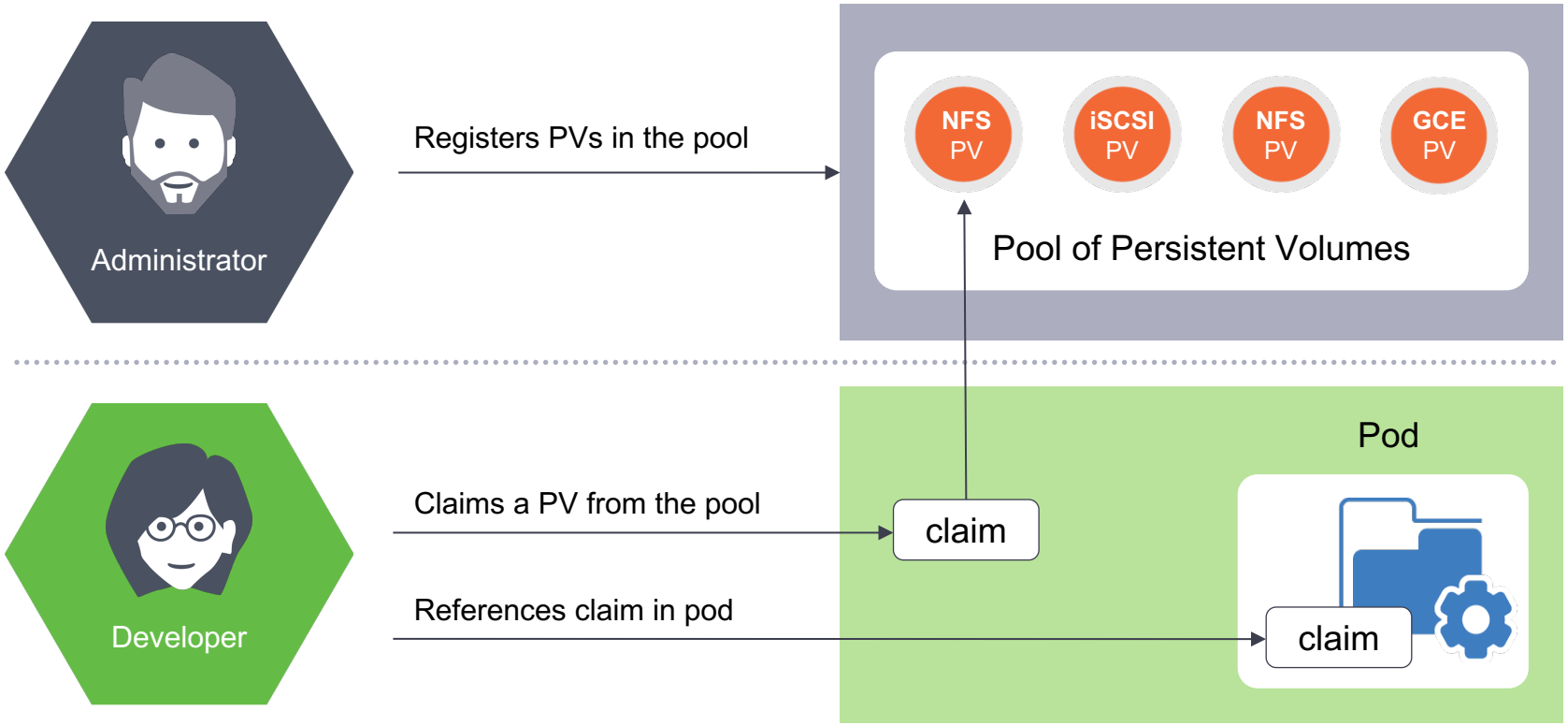
Kubernetes Storage Model: Persistent Volumes and Claims

@oicheryl

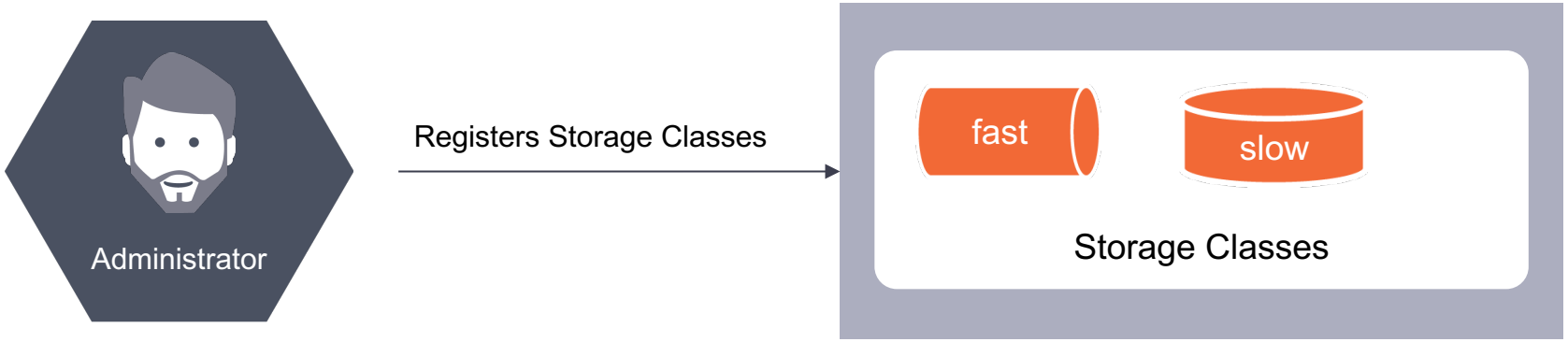


Kubernetes Storage Model: Persistent Volumes and Claims

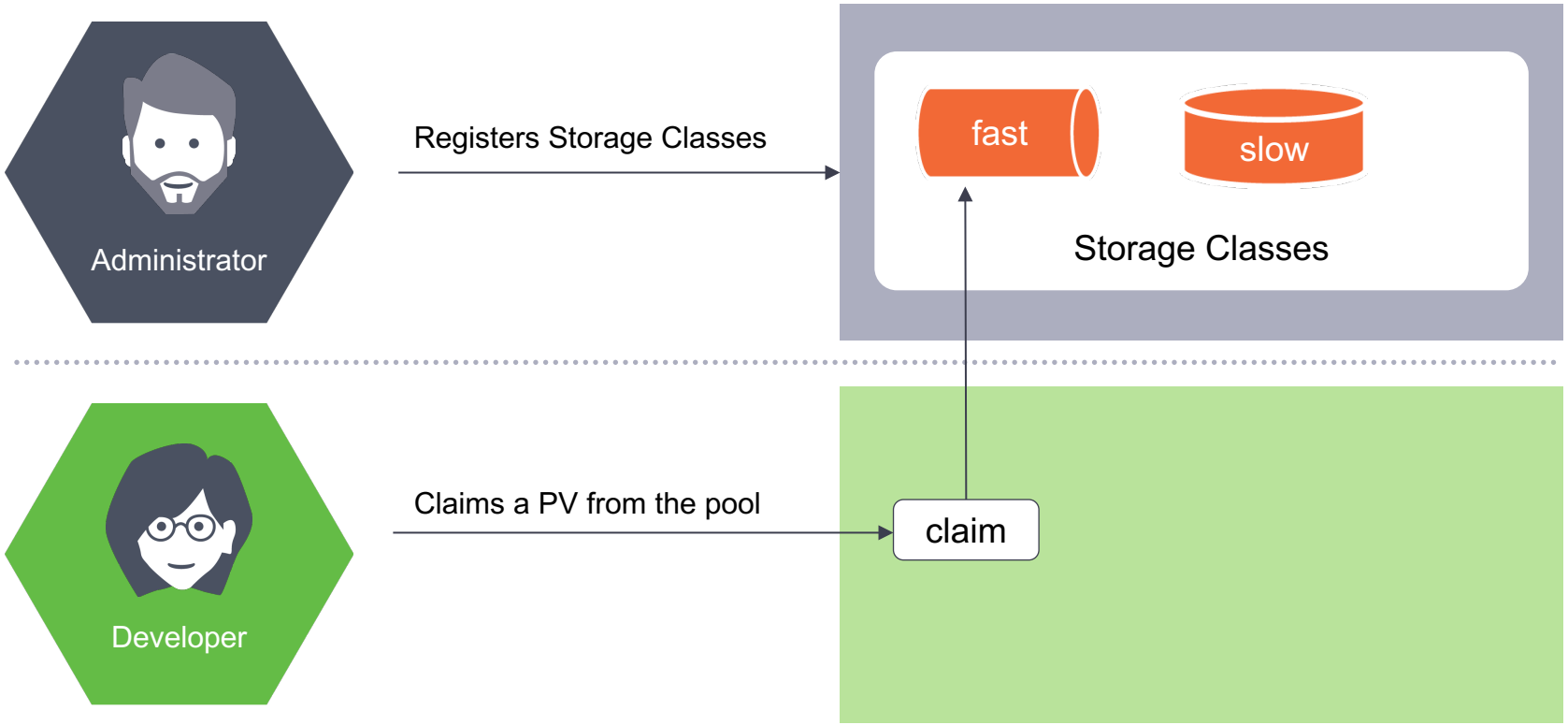
@oicheryl



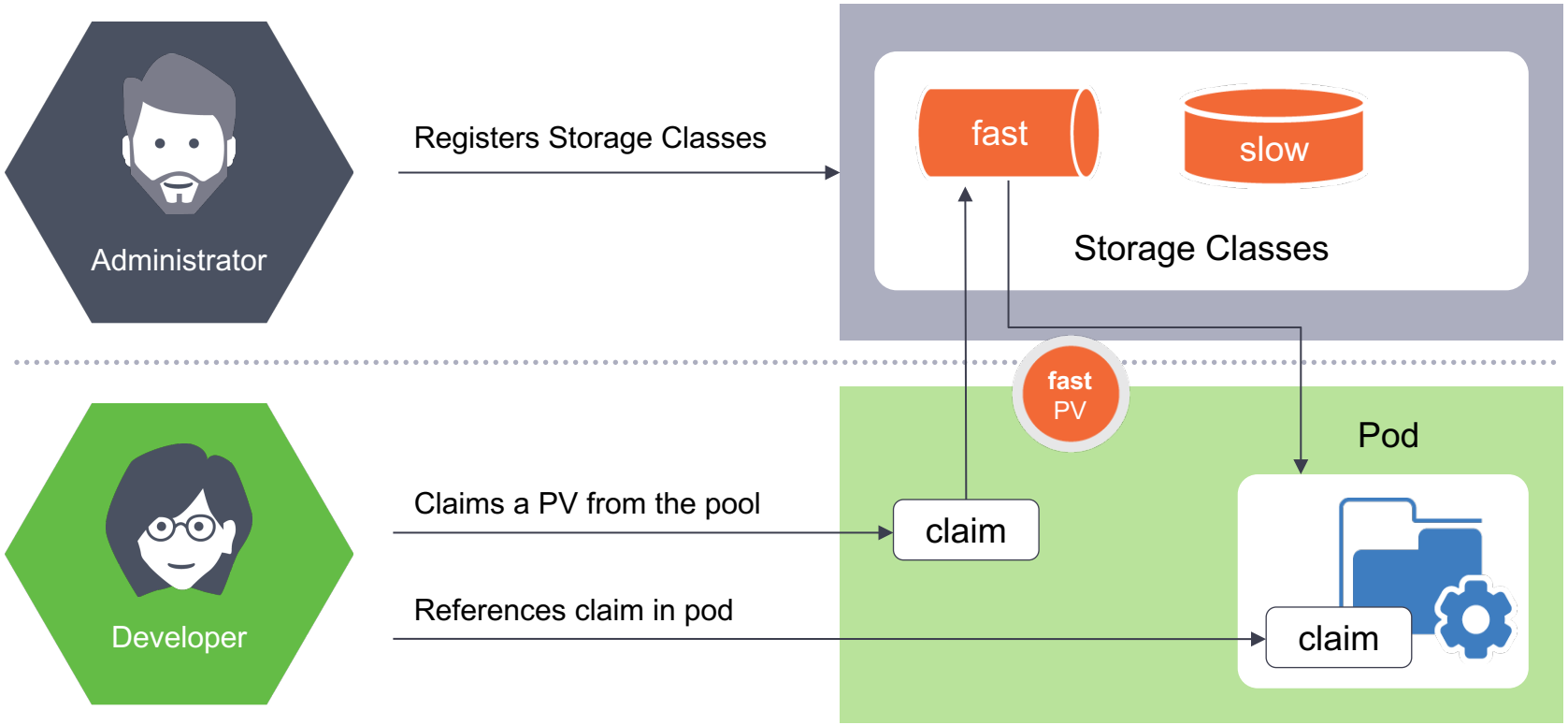
Dynamic provisioning with Storage Classes



Dynamic provisioning with Storage Classes



Dynamic provisioning with StorageClass



Eight Principles of Cloud Native Storage

Meet Jane

- A DevOps engineer in a bank
- She wants to migrate a Postgres database to containers

What is Cloud Native?

- Horizontally scalable
- No single point of failure
- Resilient and self healing
- Minimal operator overhead
- Decoupled from the underlying platform

1. Application centric

Storage should be presented to and consumed by applications, not by operating systems or hypervisors.

Storage needs to be able to follow an application as it scales, grows, and moves between platforms and clouds.

2. Platform agnostic

The storage platform should be able to run anywhere.

Upgrades and scaling is non-disruptive.

3. Declarative and composable

Storage resources should be declared and composed just like all other resources required by applications and services.

4. API Driven

Storage resources and services should be easy to be provisioned, consumed, moved and managed via an API.

5. Natively secure

Storage services should integrate and inline security features such as encryption and RBAC.

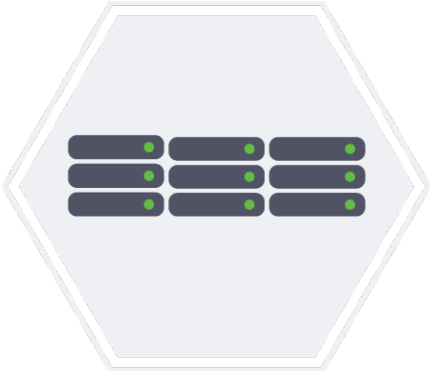
6. Agile

The platform should be able to move application data between locations, dynamically resize volumes for growth, take point in time copies of data for data retention or to facilitate rapid recovery of data.

7. Performant

The storage platform should be able to offer deterministic performance in complex distributed environments.

Block vs file vs object



Block Storage
Data stored in a fixed-size ‘blocks’ in a rigid arrangement – ideal for enterprise databases

File Storage
Data stored as ‘files’ in hierarchically nested ‘folders’ – ideal for active documents

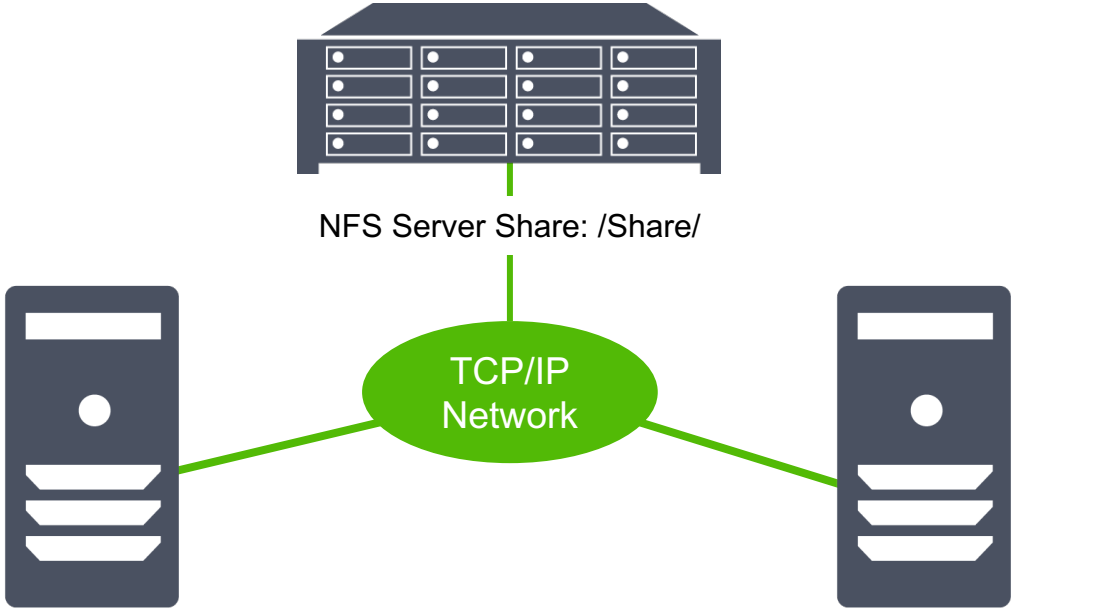
Object Storage
Data stored as ‘objects’ in scalable ‘buckets’ – ideal for unstructured big data, analytics and archiving

8. Consistently available

The storage platform should manage data distribution with a predictable, proven data model to ensure high availability, durability, consistency of application data.

Storage Landscape

Centralized file system



Centralized file system

Single point of failure
Hard to scale horizontally
No native integration

0

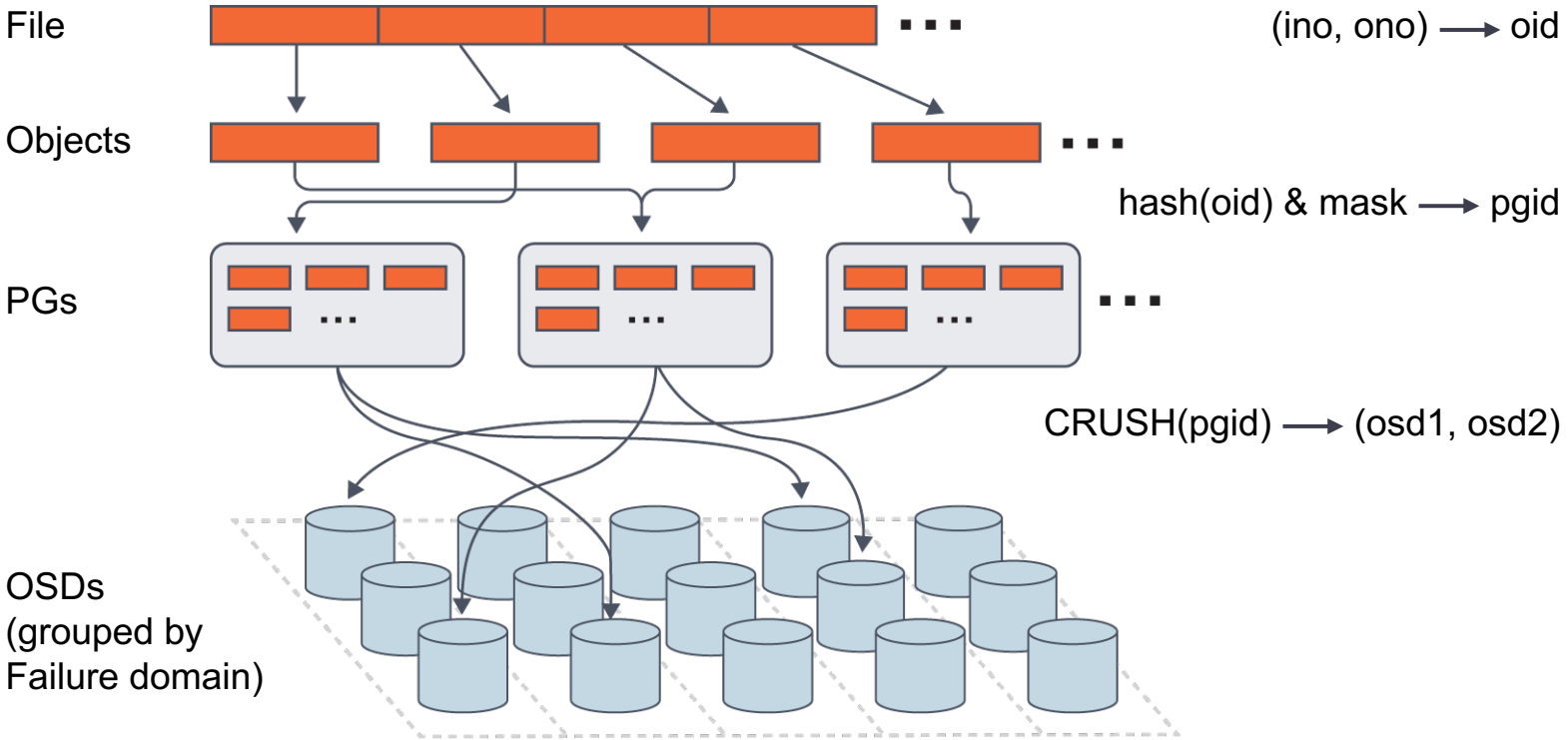
Proprietary storage array

Proprietary storage array

2

Deterministic performance
Vendor lock in
No thin provisioning
Hard to scale horizontally
Expensive and long lead times

Distributed object store

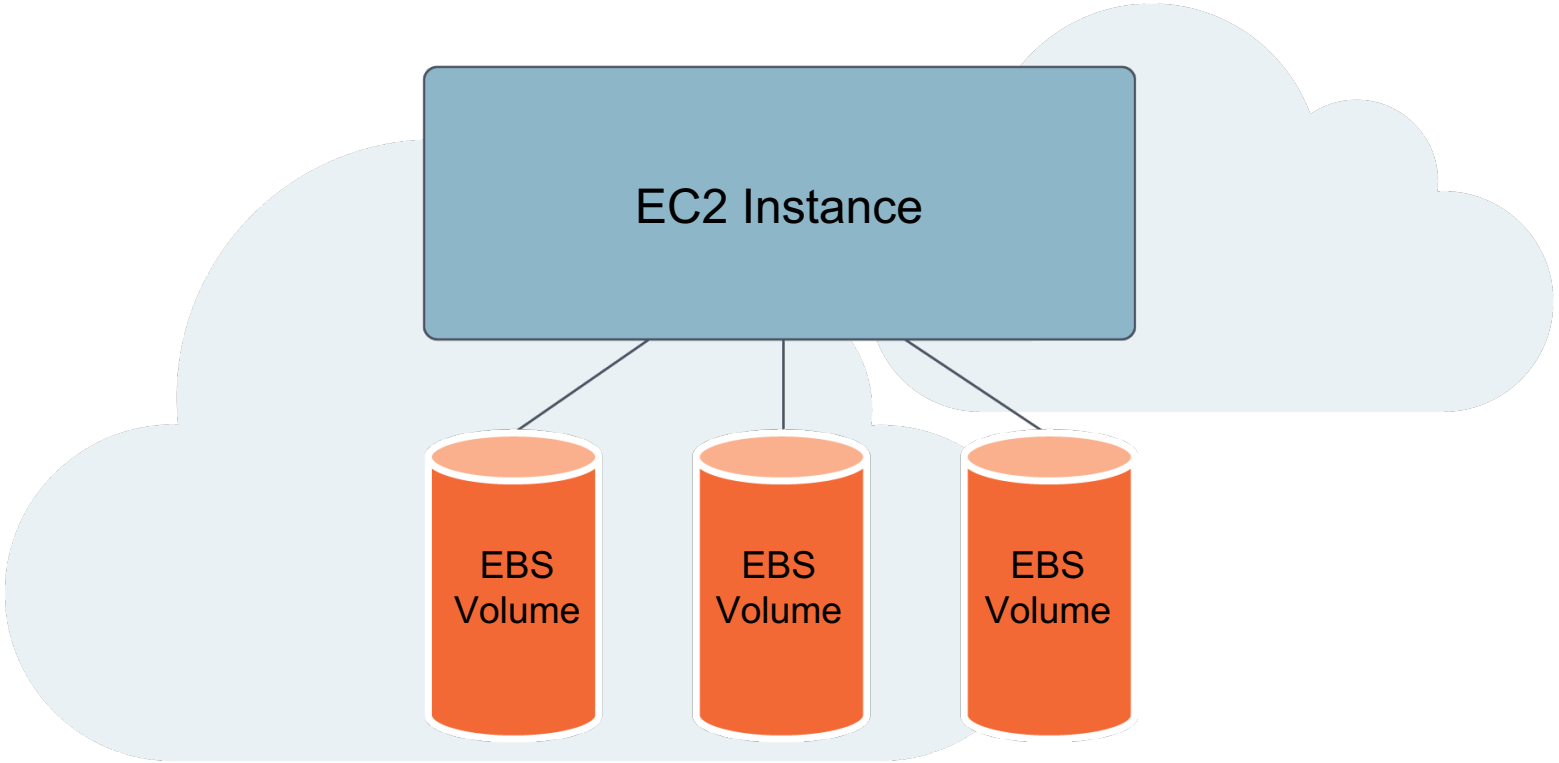


Distributed object store

4

Horizontally scalable
Hardware agnostic
Complicated to set up (see: Rook)
Writes fan out 13-40 times
Failures are expensive

Public cloud / managed service



6

Horizontally scalable

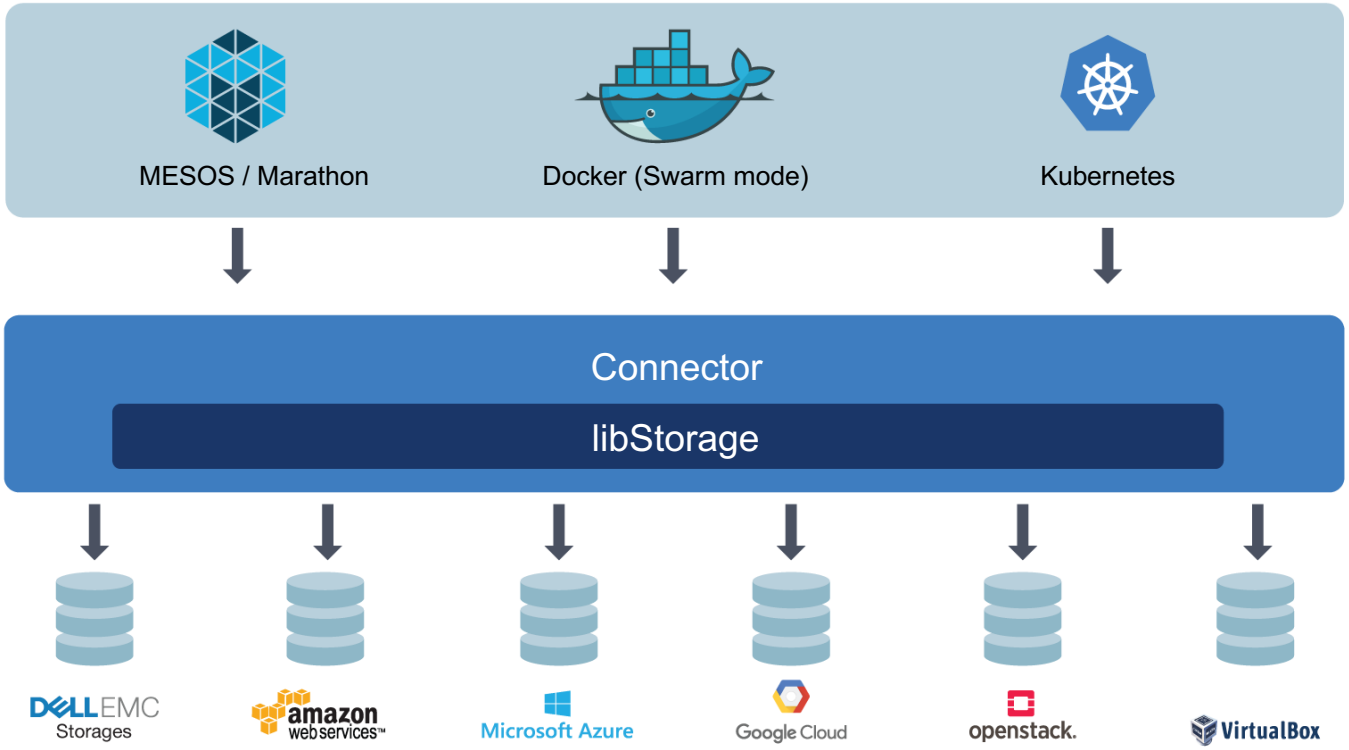
Consistent and performant

40 EBS volumes per EC2 instance

Mounting physical block devices is slow

Expensive, vendor lock in, compliance

API framework



API framework

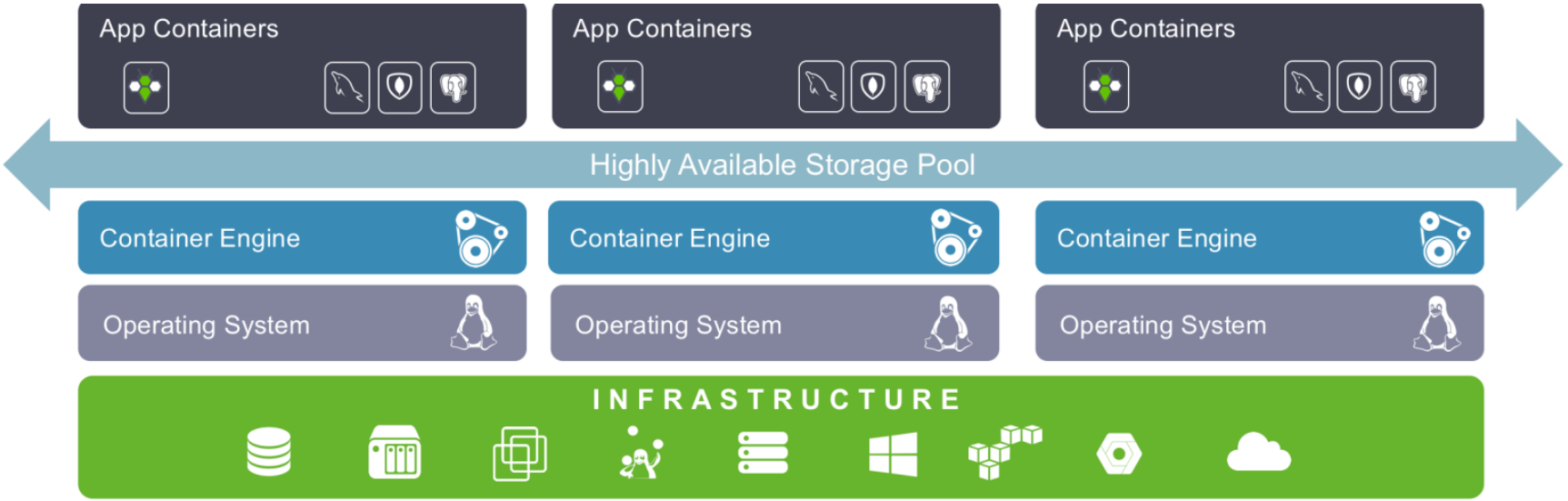
5

Integrated with Docker/Kubernetes

No guarantees for performance, replication, failover, scalability

Orchestrated storage (StorageOS)

@oicheryl

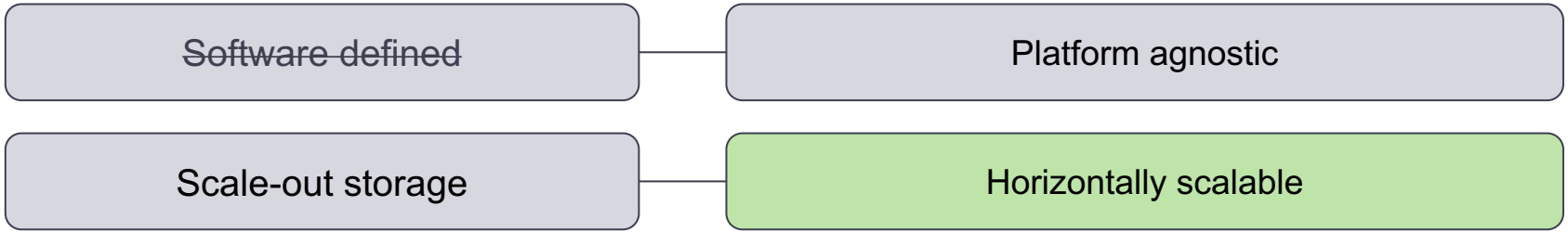


Orchestrated storage (StorageOS)

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

Orchestrated storage (StorageOS)

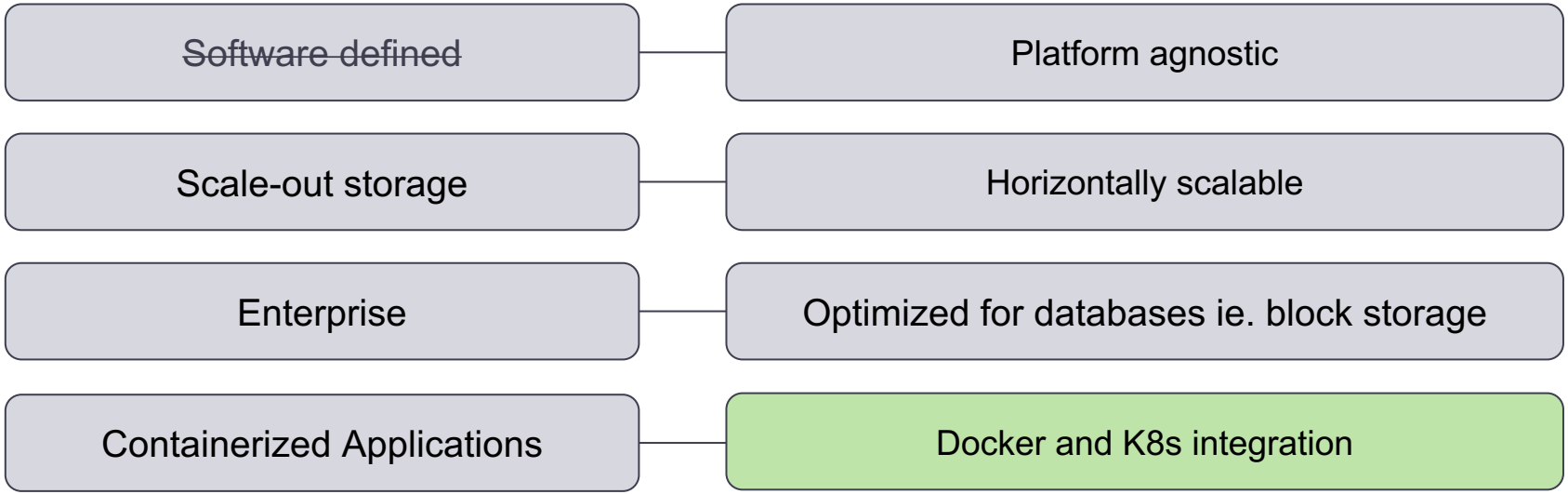
Orchestrated storage (StorageOS)



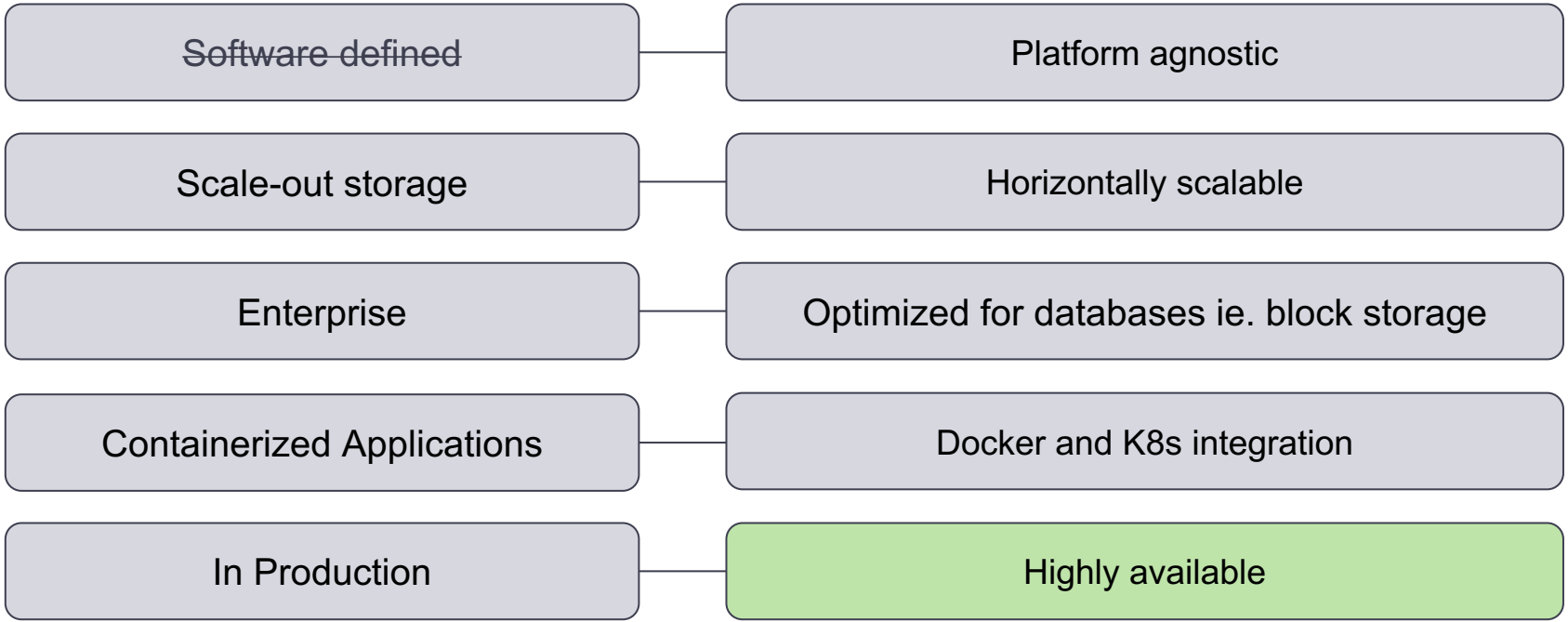
Orchestrated storage (StorageOS)

~~Software defined~~	Platform agnostic
Scale-out storage	Horizontally scalable
Enterprise	Optimized for databases ie. block storage

Orchestrated storage (StorageOS)



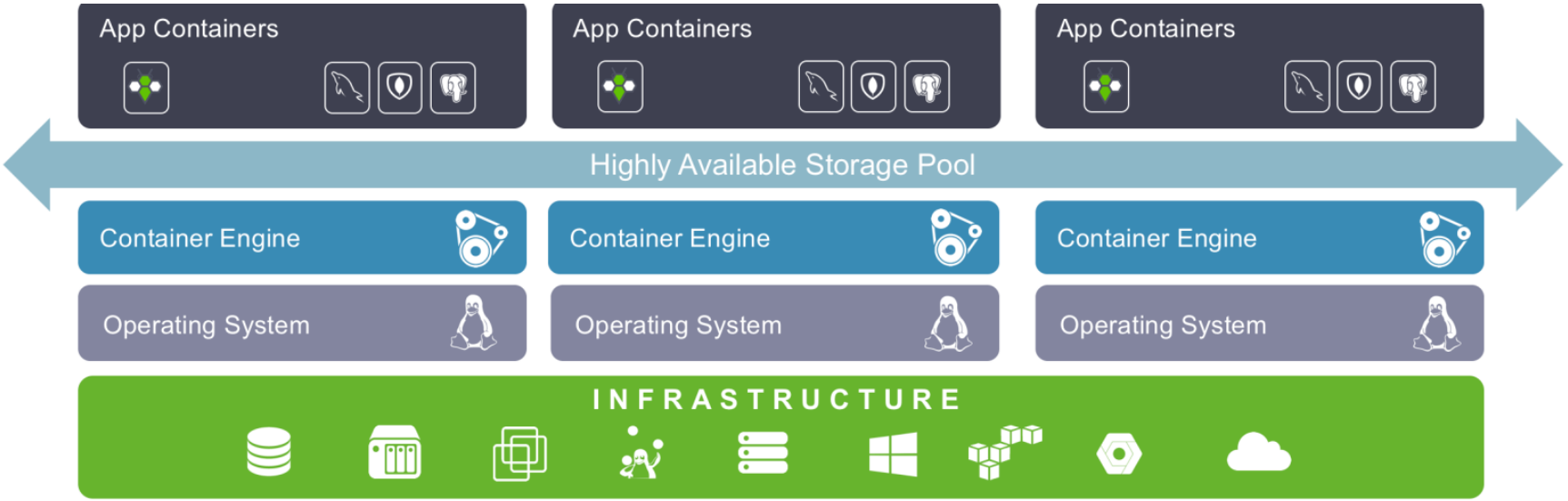
Orchestrated storage (StorageOS)



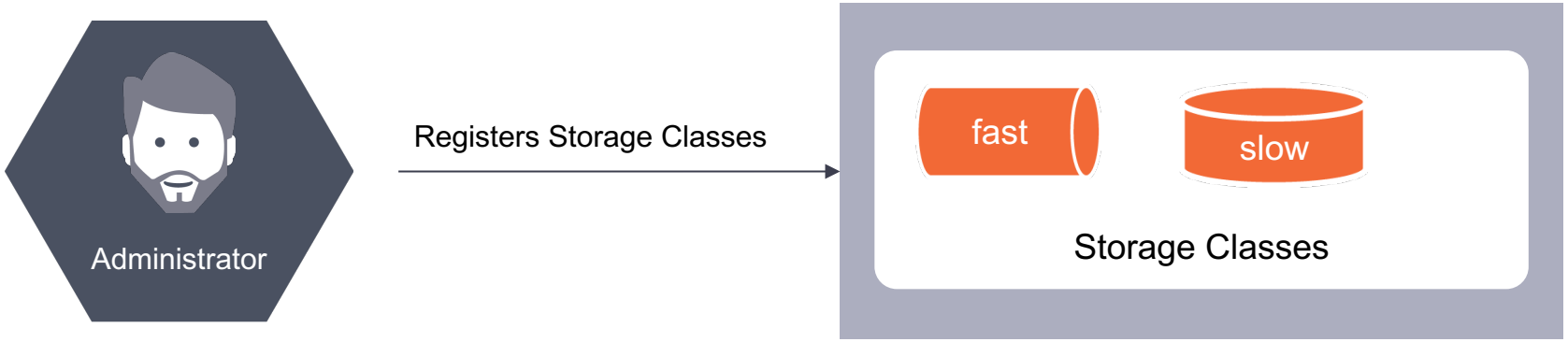
KEEP CALM IT IS DEMO TIME

Orchestrated storage (StorageOS)

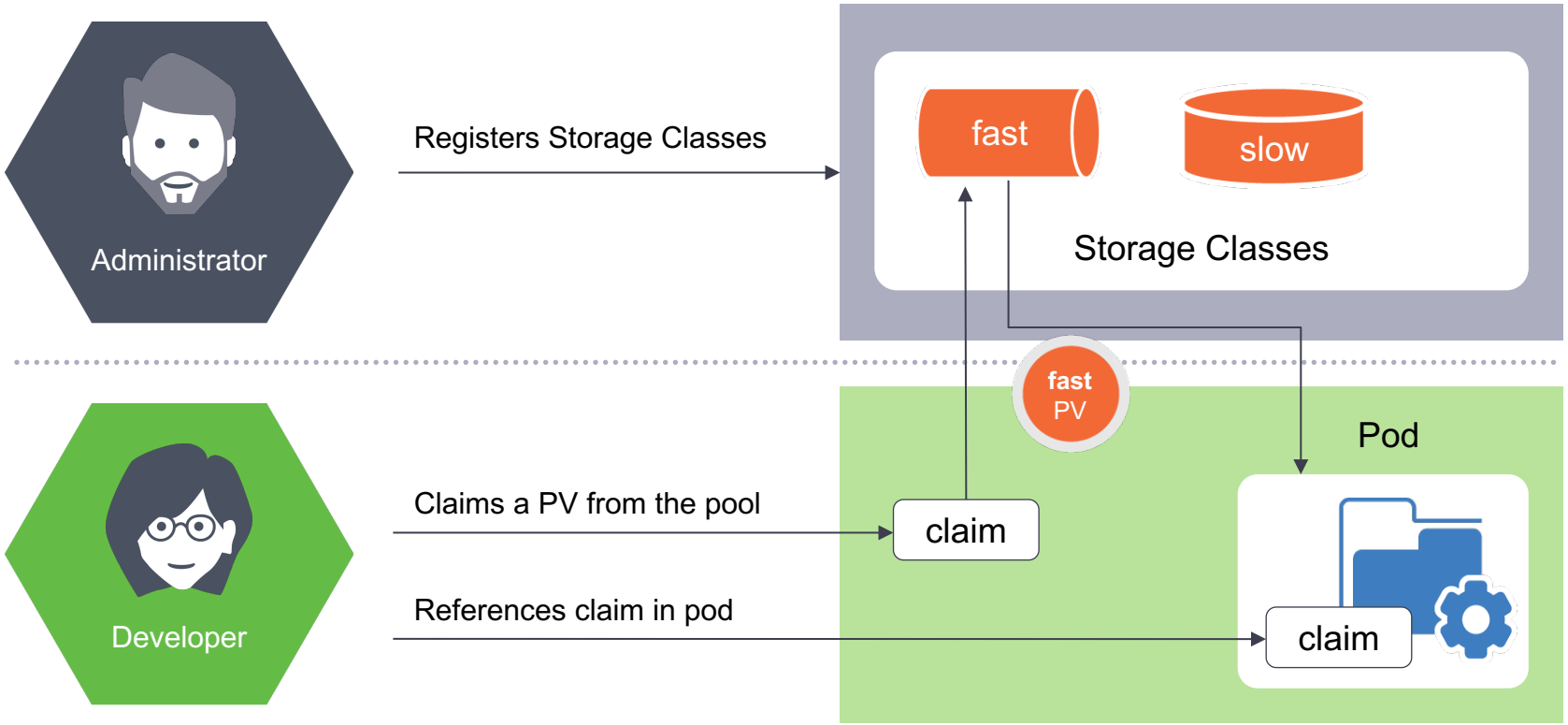
@oicheryl



Create a StorageClass

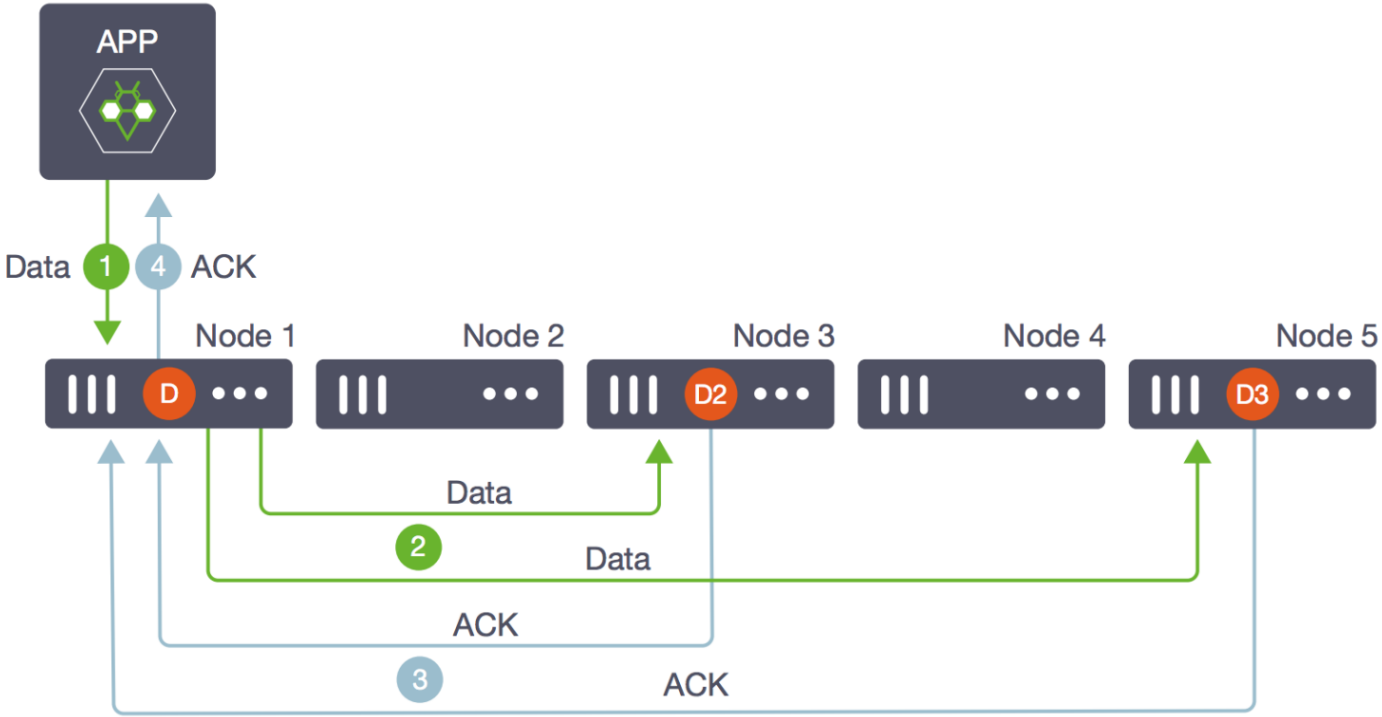


Claim a persistent volume



High availability with StorageOS

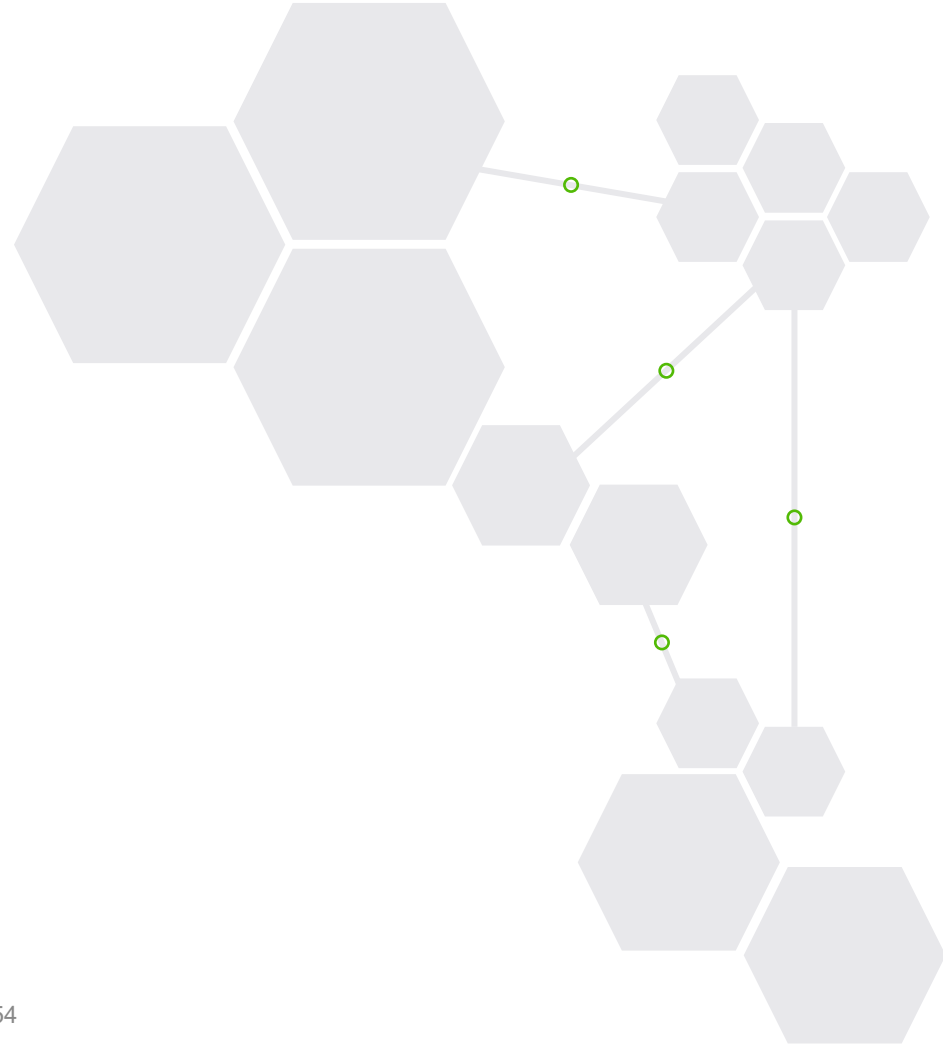
@oicheryl



8

Horizontally scalable
Consistent and performant
Platform agnostic
Synchronous replication
Volume is limited to the size of one node

To Recap…



Storage is tricky…!

But you can compare storage using the eight principles:

1. Application centric	5. Natively secure
2. Platform agnostic	6. Agile
3. Declarative and composable	7. Performant
4. API driven	8. Consistently available

K8S Storage SIG / CNCF Storage WG

Objective is to define an industry standard “Container Storage Interface” (CSI) that will enable storage vendors to develop a plugin once and have it work across a number of container orchestration systems.

Storage is tricky…!

StorageOS gives you the reliability, scalability and developer ease of managed service, at the performance and price of local storage.

STORAGEOS

In-browser tutorials
my.storageos.com/main/tutorials

Quickstart
storageos.com/kubernetes

Hiring in London!

STORAGEOS

Questions

Slides at oicheryl.com