Persistent Storage with Kubernetes in Production

Which solution and why?

Kubernetes London, 30 Nov 2017

Cheryl Hung, Product Manager

© StorageOS Ltd.

Cheryl

Product manager at StorageOS

@oicheryl

© StorageOS Ltd.

STORAGEOS

Objectives

- Why is state tricky in Kubernetes?
- How do I evaluate storage?
- What storage options are there for Kubernetes?

© StorageOS Ltd.

Objectives

- Why is state tricky in Kubernetes?
- How do I evaluate storage?
- What storage options are there for Kubernetes?

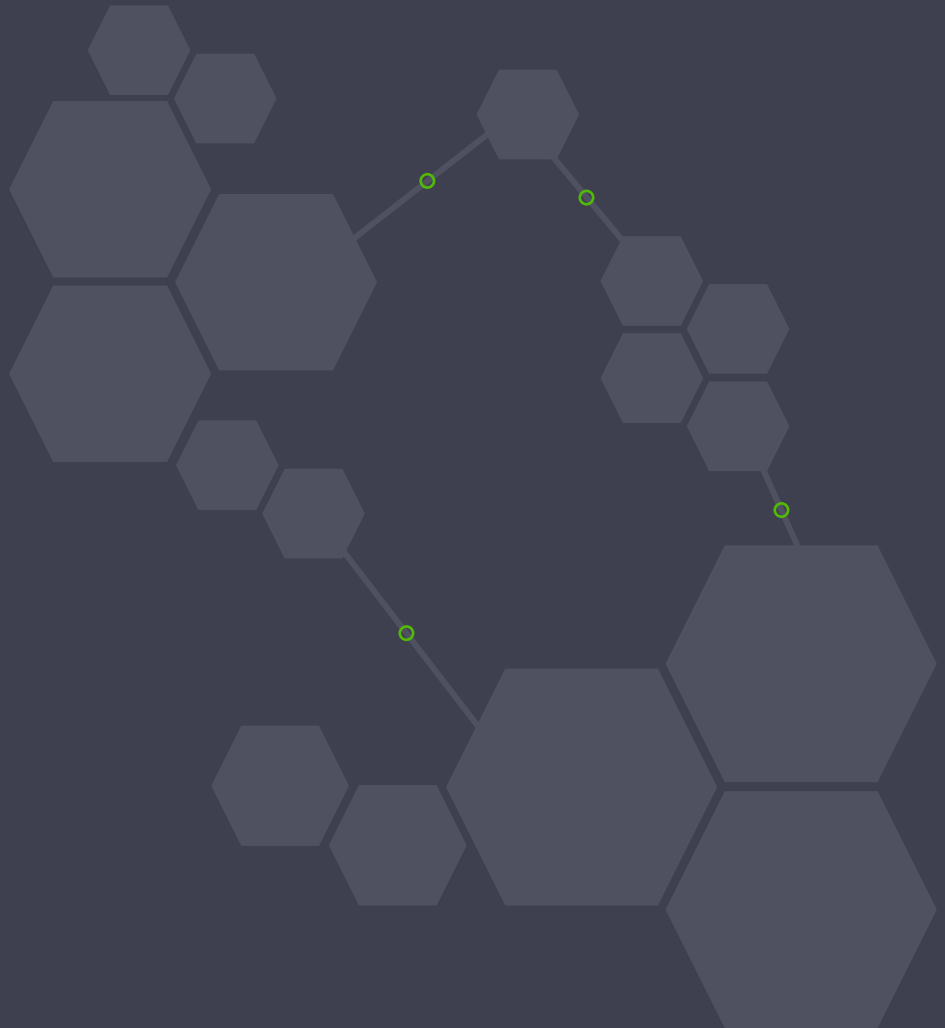
Anti-objective:

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

© StorageOS Ltd.

STORAGEOS

Why is state so tricky?



© StorageOS Ltd.

Why do I need storage?

Why do I need storage?

© StorageOS Ltd.

Problem 1: No pet storage

© 2013-2017 StorageOS Ltd. All rights reserved.

Problem 2: Data needs to follow

© 2013-2017 StorageOS Ltd. All rights reserved.

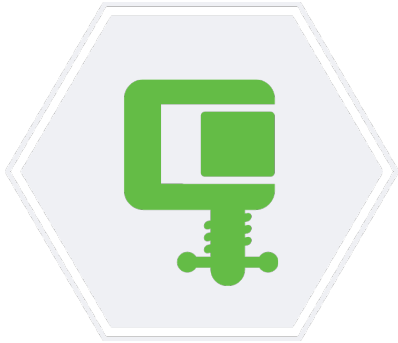
Problem 3: Gravity can be a problem

© StorageOS Ltd.

Problem 4: Storage isn't just storage

App Binaries

App data



Config

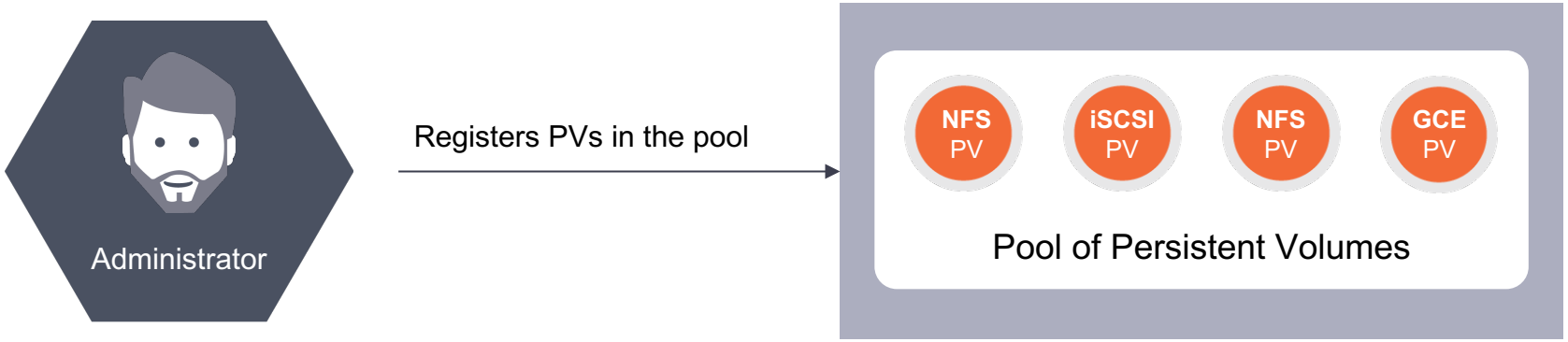
Backup

© StorageOS Ltd.

Kubernetes Storage Model: Persistent Volumes and Claims

© StorageOS Ltd.

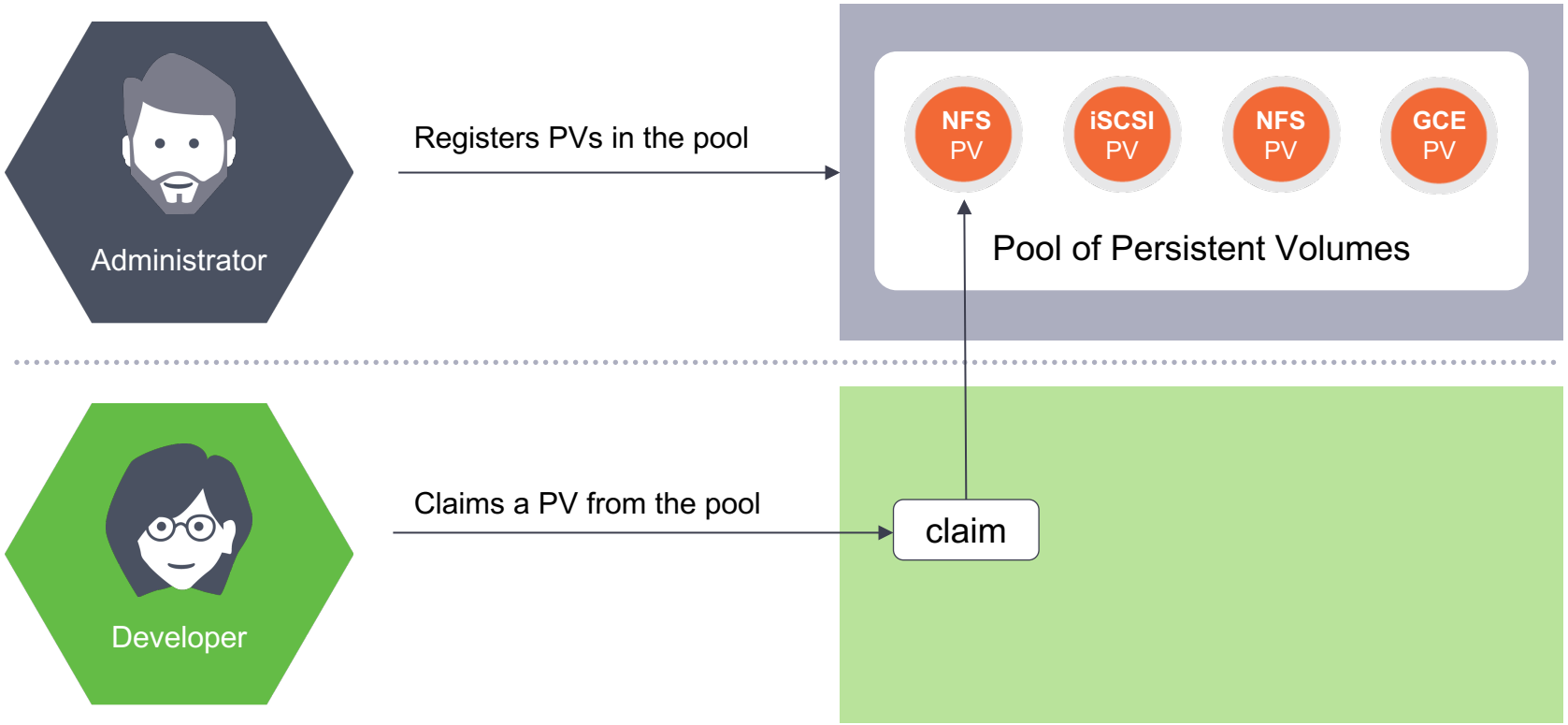
Kubernetes Storage Model: Persistent Volumes and Claims



© StorageOS Ltd.

Kubernetes Storage Model: Persistent Volumes and Claims

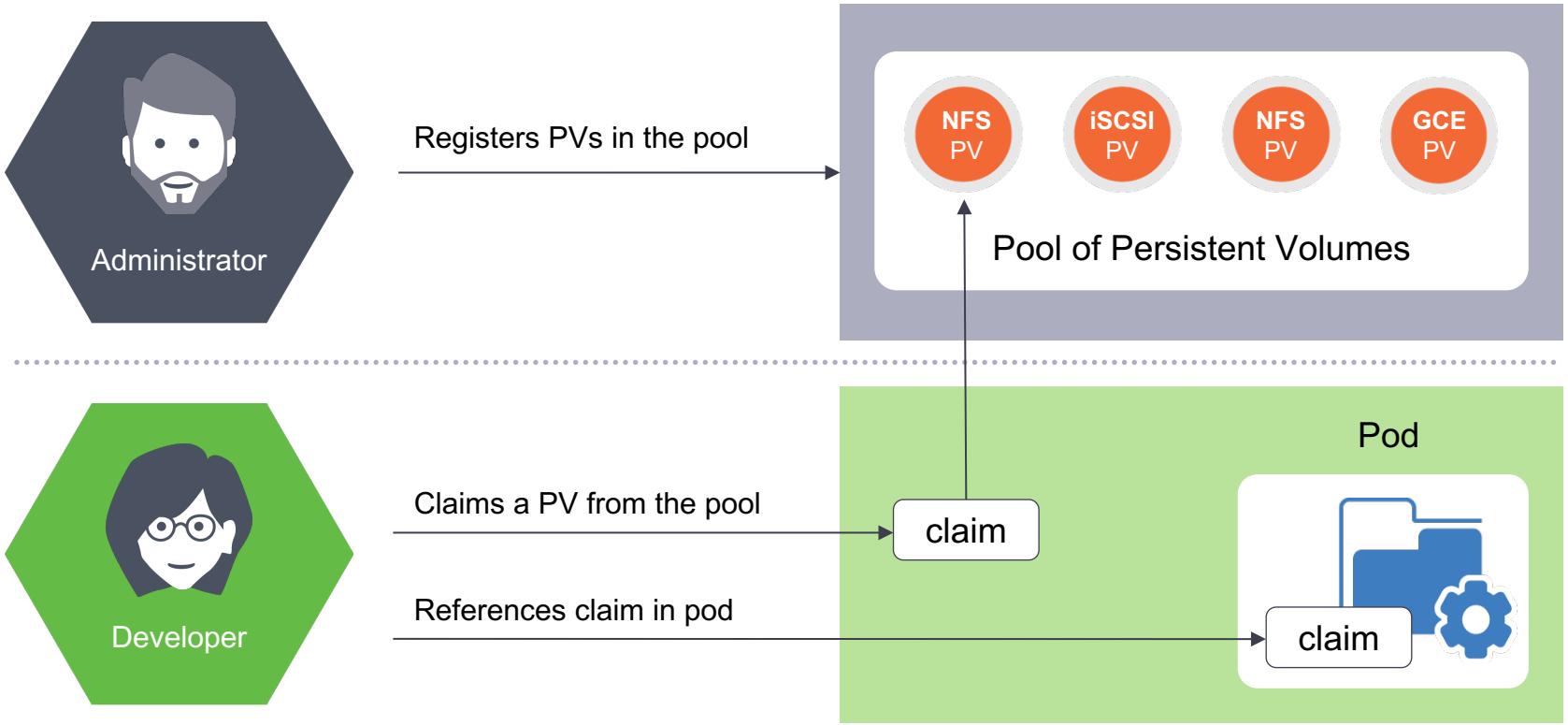
@oicheryl



© StorageOS Ltd.

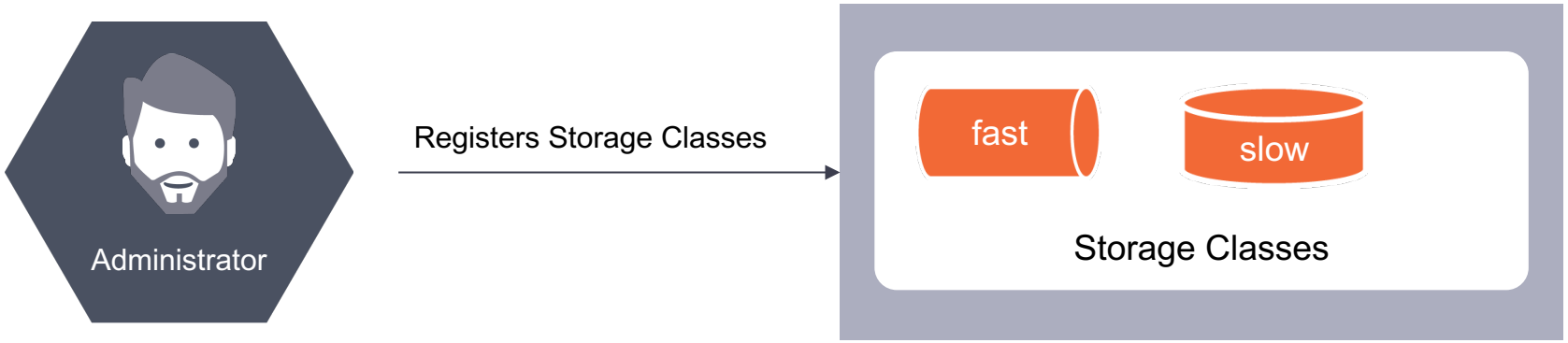
Kubernetes Storage Model: Persistent Volumes and Claims

@oicheryl



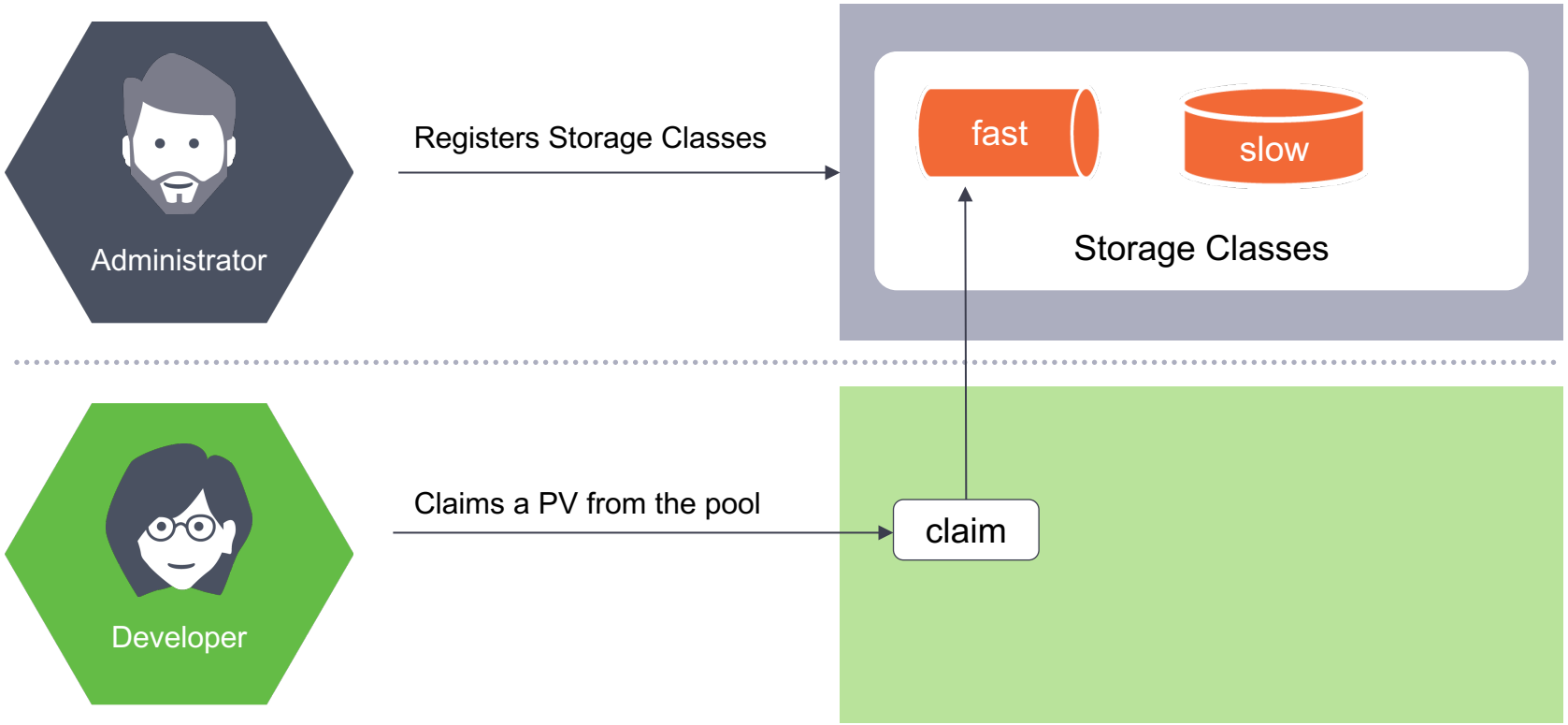
© StorageOS Ltd.

Dynamic provisioning with Storage Classes



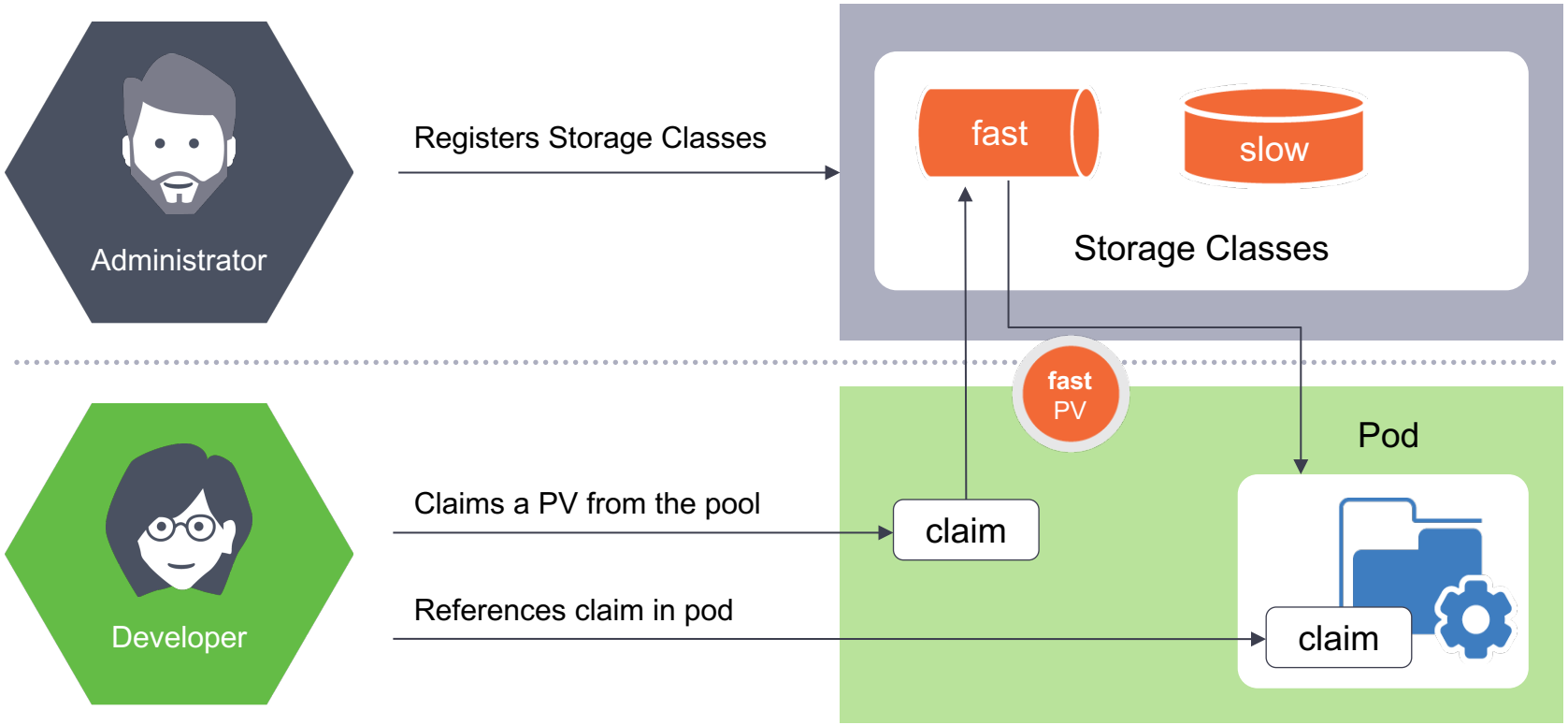
© StorageOS Ltd.

Dynamic provisioning with Storage Classes



© StorageOS Ltd.

Dynamic provisioning with StorageClass



© StorageOS Ltd.

Eight Principles of Cloud Native Storage

What is Cloud Native?

Horizontally scalable

No single point of failure

Resilient and self healing

Minimal operator overhead

Decoupled from the underlying platform

© StorageOS Ltd.

Meet Jane

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

© StorageOS Ltd.

1. Application centric

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

© StorageOS Ltd.

2. Platform agnostic

The storage platform should be able to run anywhere.

Upgrades and scaling is non-disruptive.

© StorageOS Ltd.

3. Declarative and composable

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

© StorageOS Ltd.

4. API Driven

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

© StorageOS Ltd.

5. Natively secure

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

© StorageOS Ltd.

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.

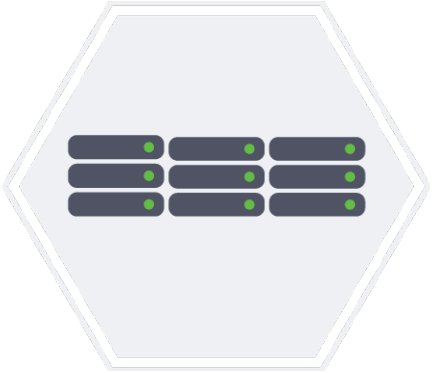
© StorageOS Ltd.

7. Performant

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

© StorageOS Ltd.

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

© StorageOS Ltd.

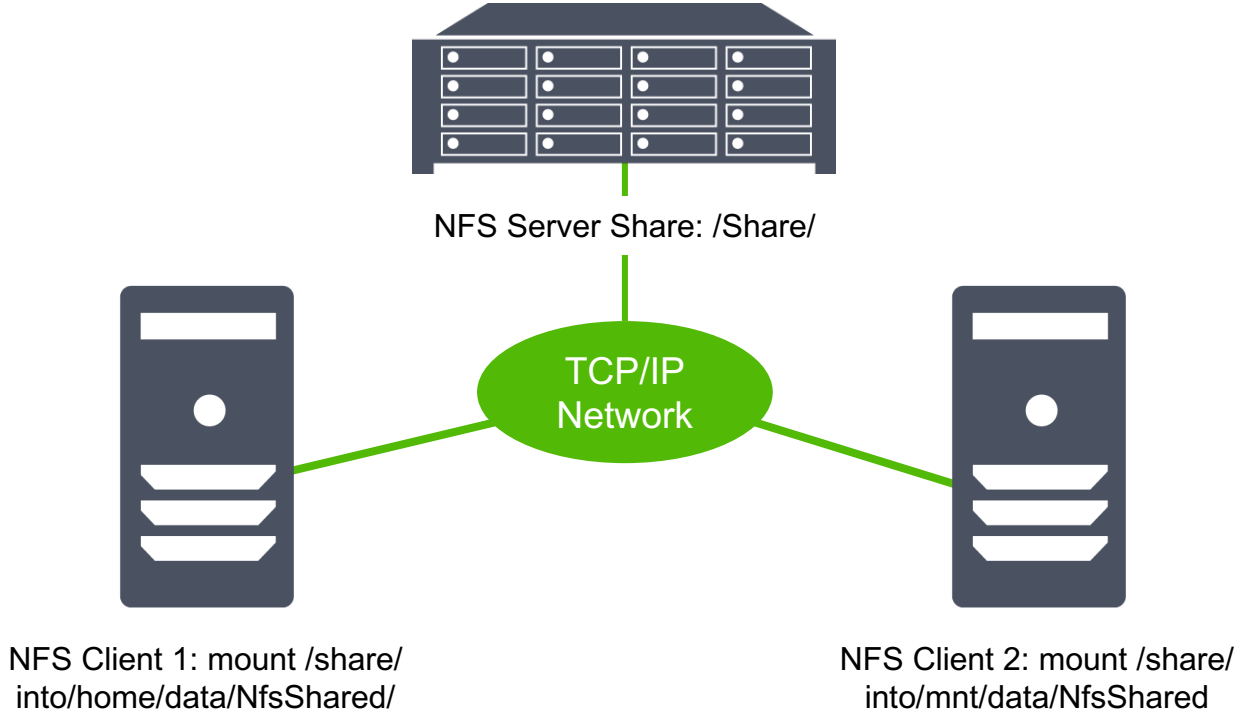
8. Consistently available

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

© StorageOS Ltd.

Storage Landscape

Centralized file system



© StorageOS Ltd.

Centralized file system

Single point of failure
Hard to scale horizontally
No Docker/Kubernetes integration

0

© StorageOS Ltd.

Proprietary storage array

© StorageOS Ltd.

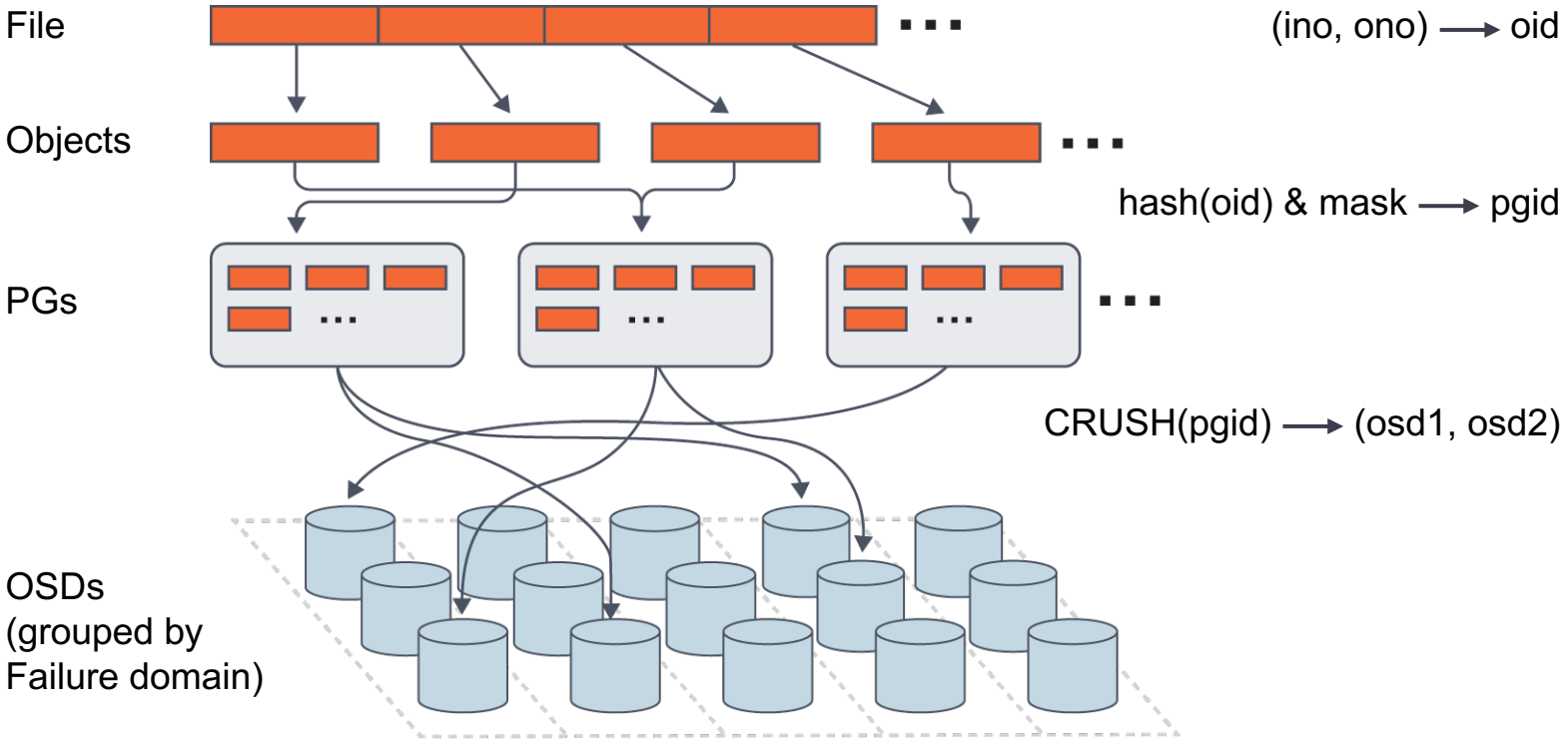
Proprietary storage array

2

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

© StorageOS Ltd.

Distributed storage

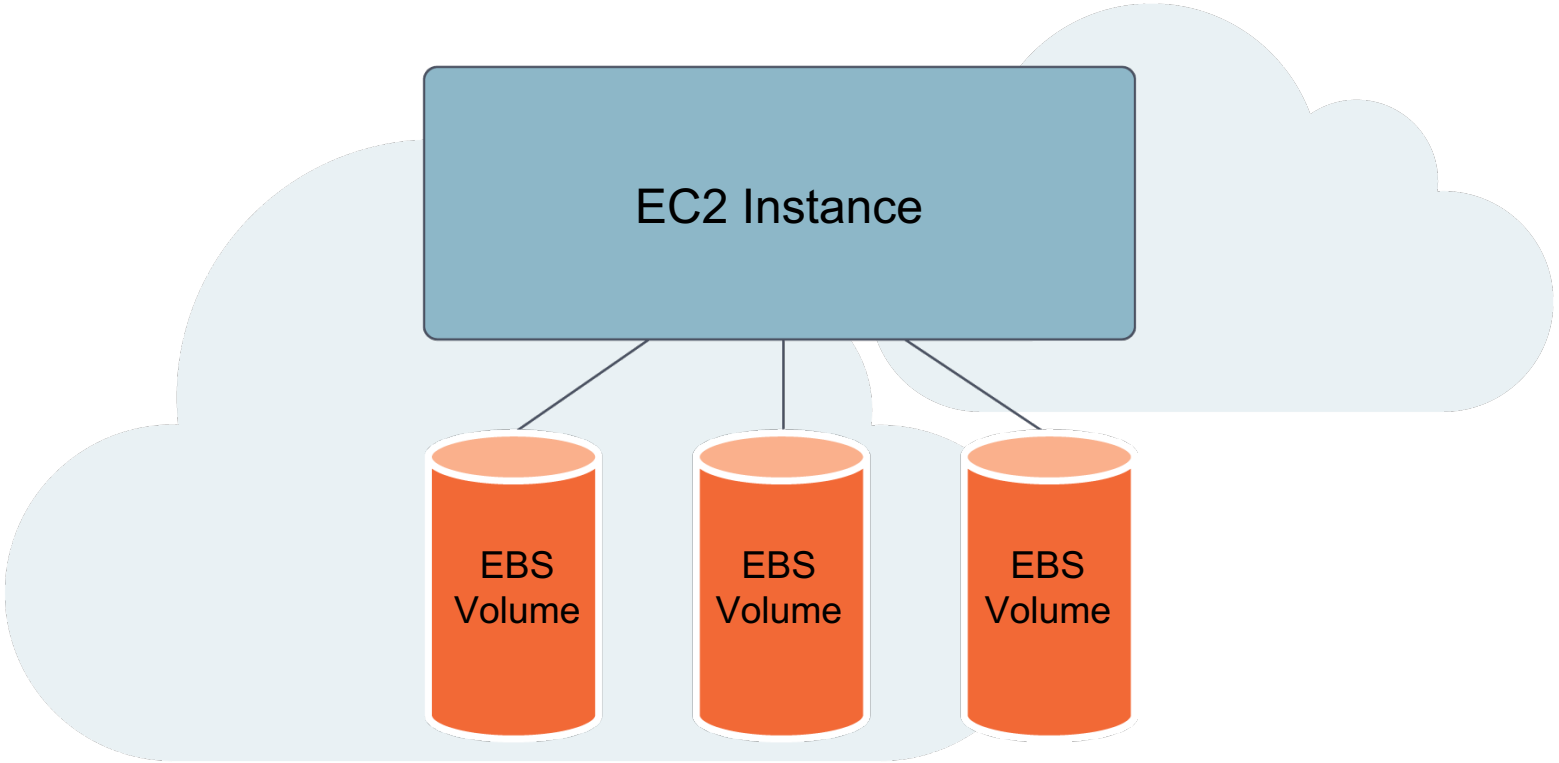


© StorageOS Ltd.

Horizontally scalable
Hardware agnostic
Write amplification
Failures are expensive
Complicated to set up

© StorageOS Ltd.

Public cloud SAN



© StorageOS Ltd.

Public cloud SAN

6

Horizontally scalable

Consistent and performant

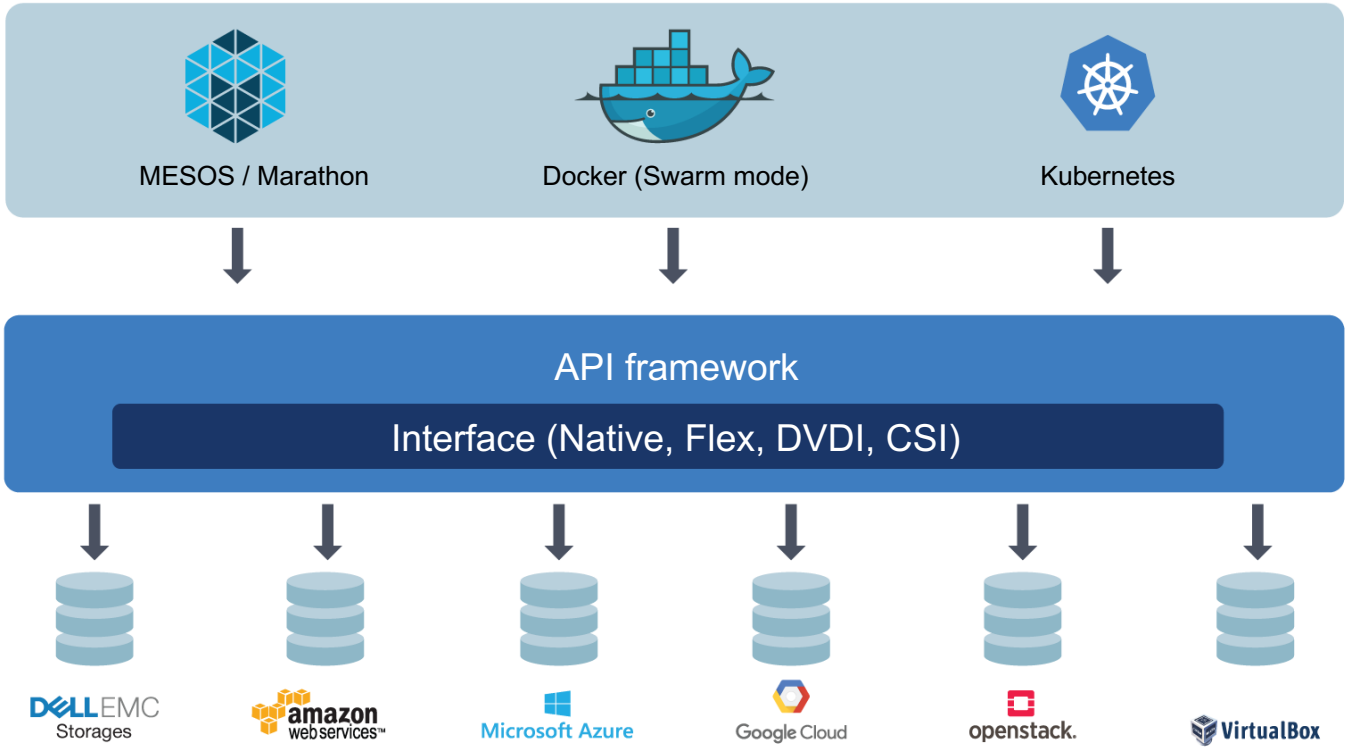
Attach/detach times on physical block devices is slow

Expensive, vendor lock in

Compliance

© StorageOS Ltd.

API framework



© StorageOS Ltd.

5

Docker/Kubernetes integration

No guarantees for performance, replication, failover, scalability

© StorageOS Ltd.

Orchestrated storage (StorageOS)

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

© StorageOS Ltd.

Orchestrated storage (StorageOS)

Platform agnostic

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

© StorageOS Ltd.

Orchestrated storage (StorageOS)

Platform agnostic

Horizontally scalable

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

© StorageOS Ltd.

Orchestrated storage (StorageOS)

Platform agnostic

Horizontally scalable

Database (ie. block)

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

© StorageOS Ltd.

Orchestrated storage (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

© StorageOS Ltd.

Orchestrated storage (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

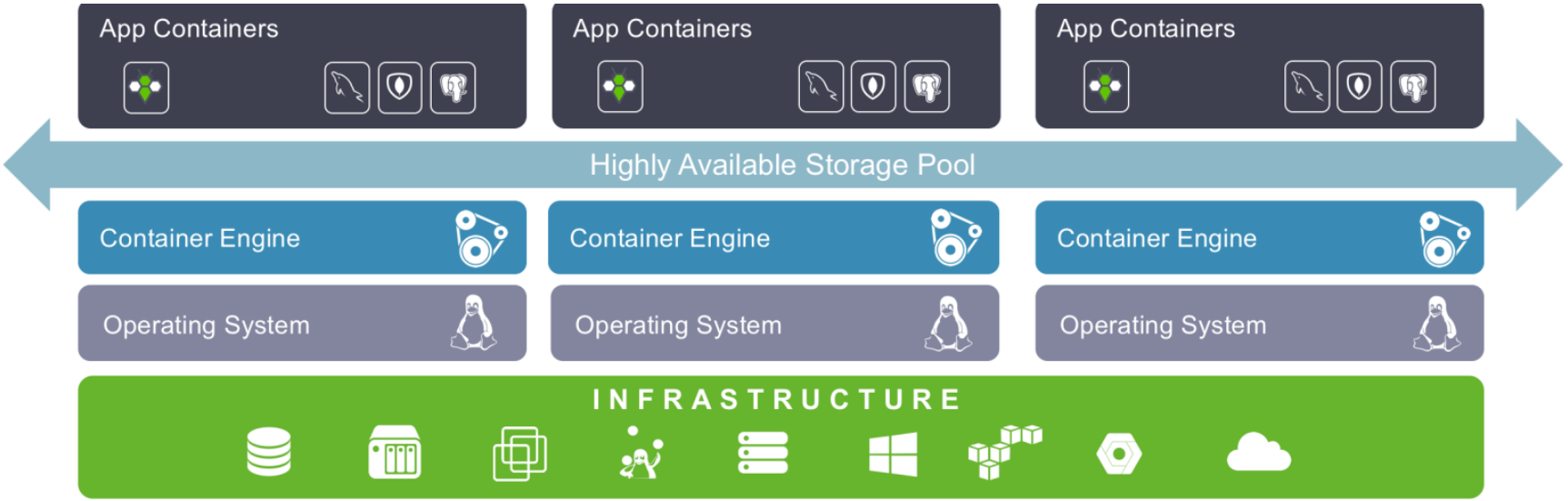
High availability

© StorageOS Ltd.

KEEP CALM IT IS DEMO TIME

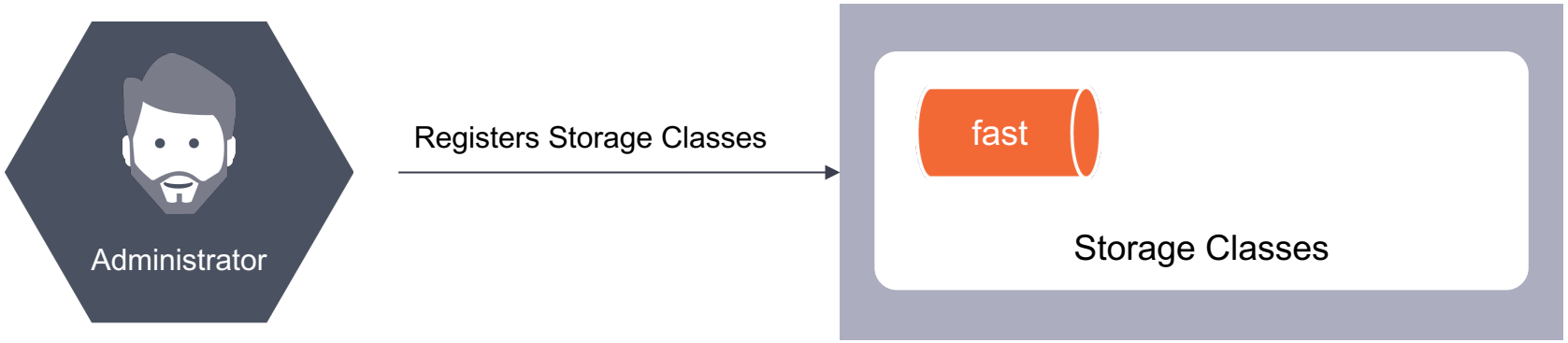
© StorageOS Ltd.

Orchestrated storage (StorageOS)



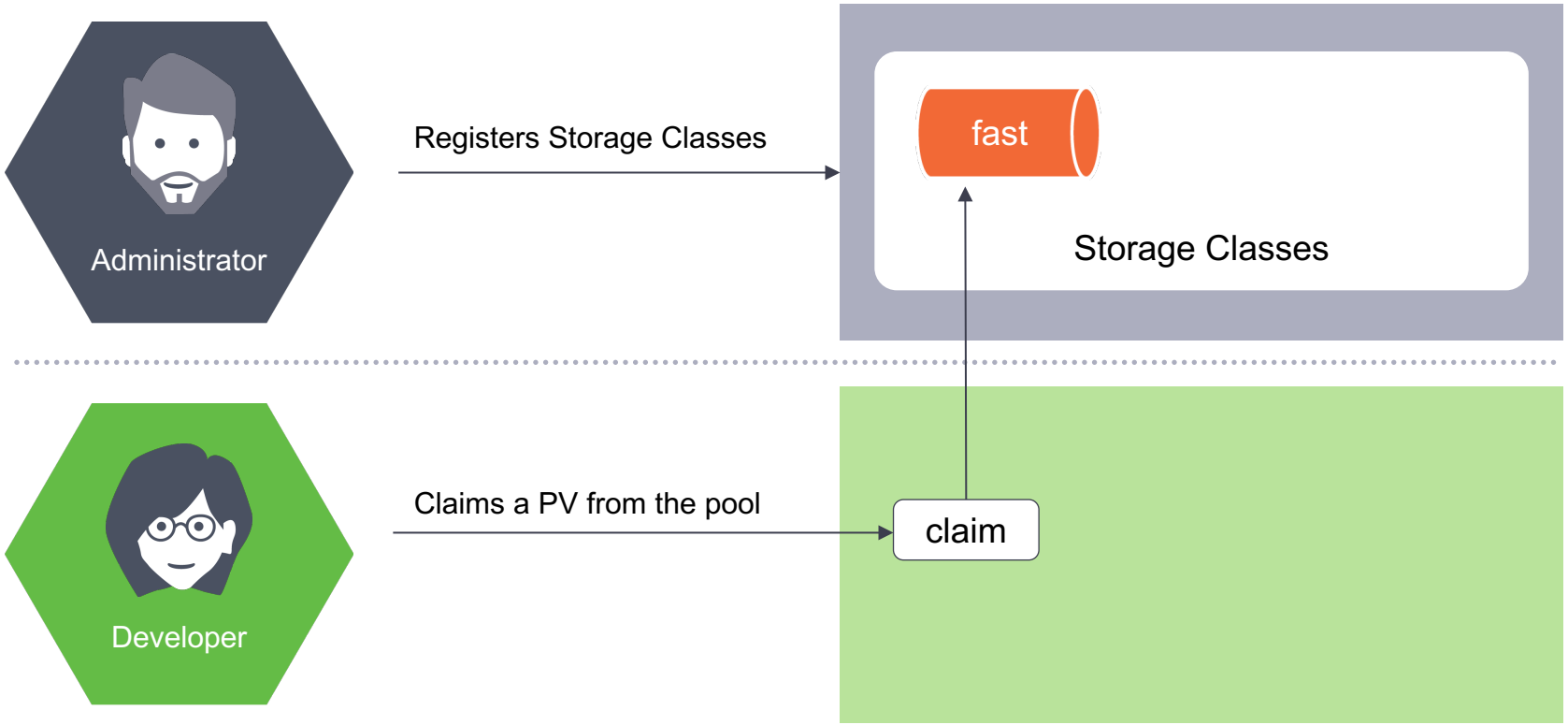
© StorageOS Ltd.

Create the fast StorageClass



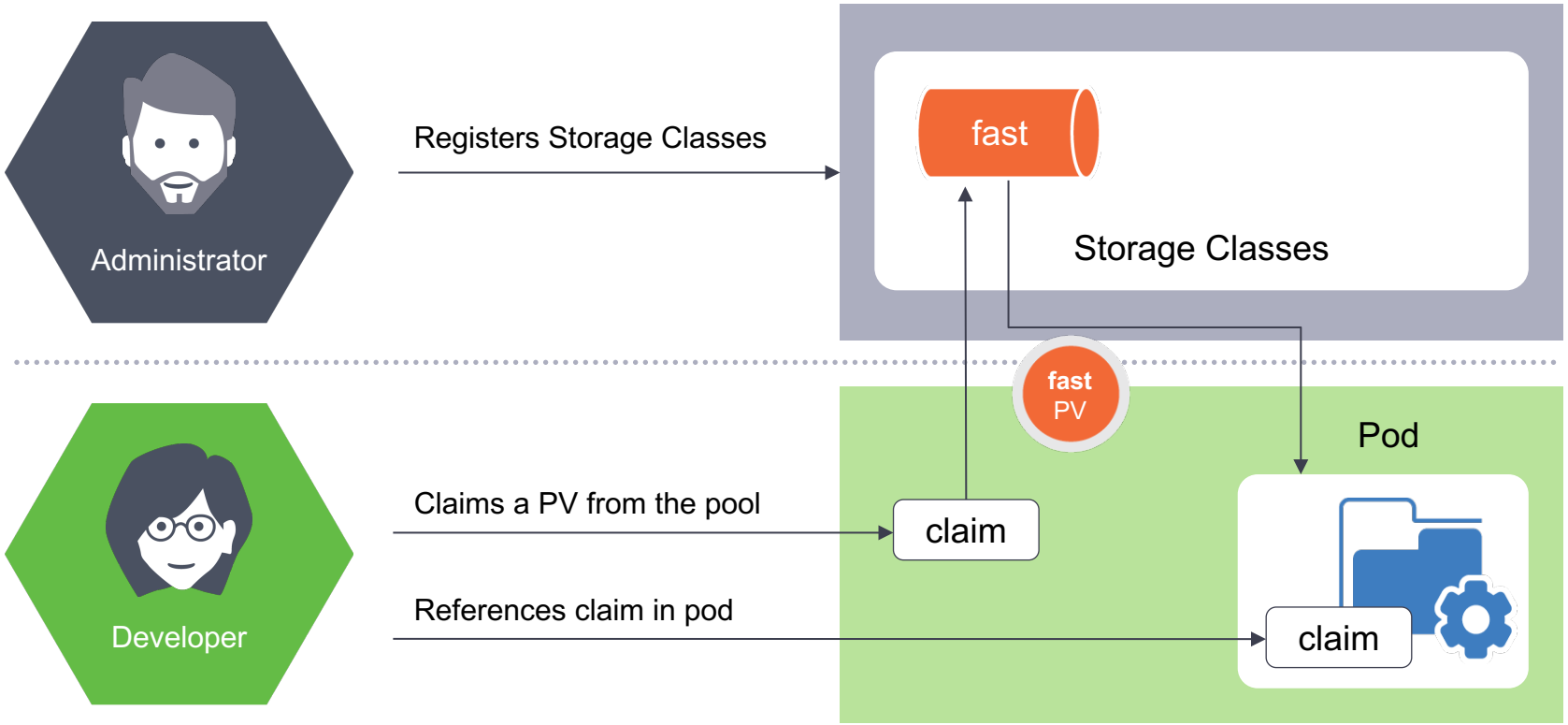
© StorageOS Ltd.

Claim a persistent volume



© StorageOS Ltd.

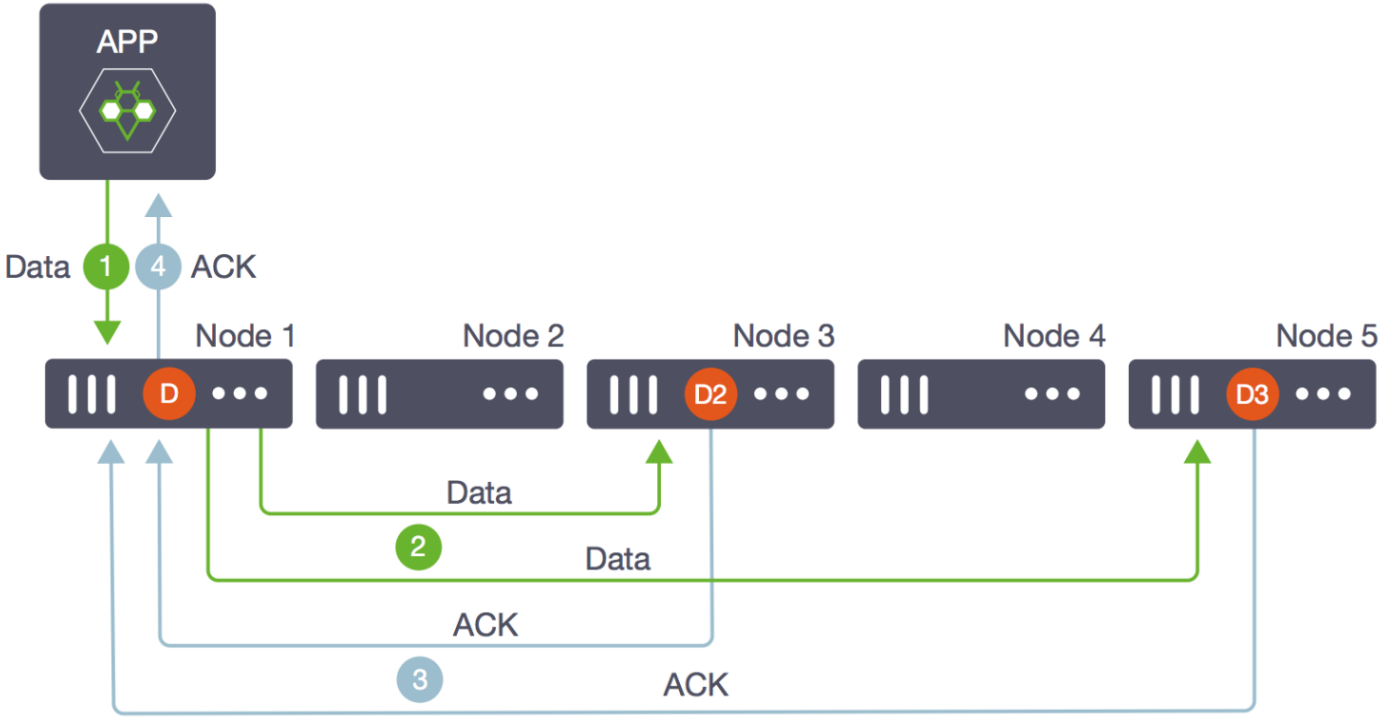
Claim a persistent volume



© StorageOS Ltd.

High availability with StorageOS

@oicheryl

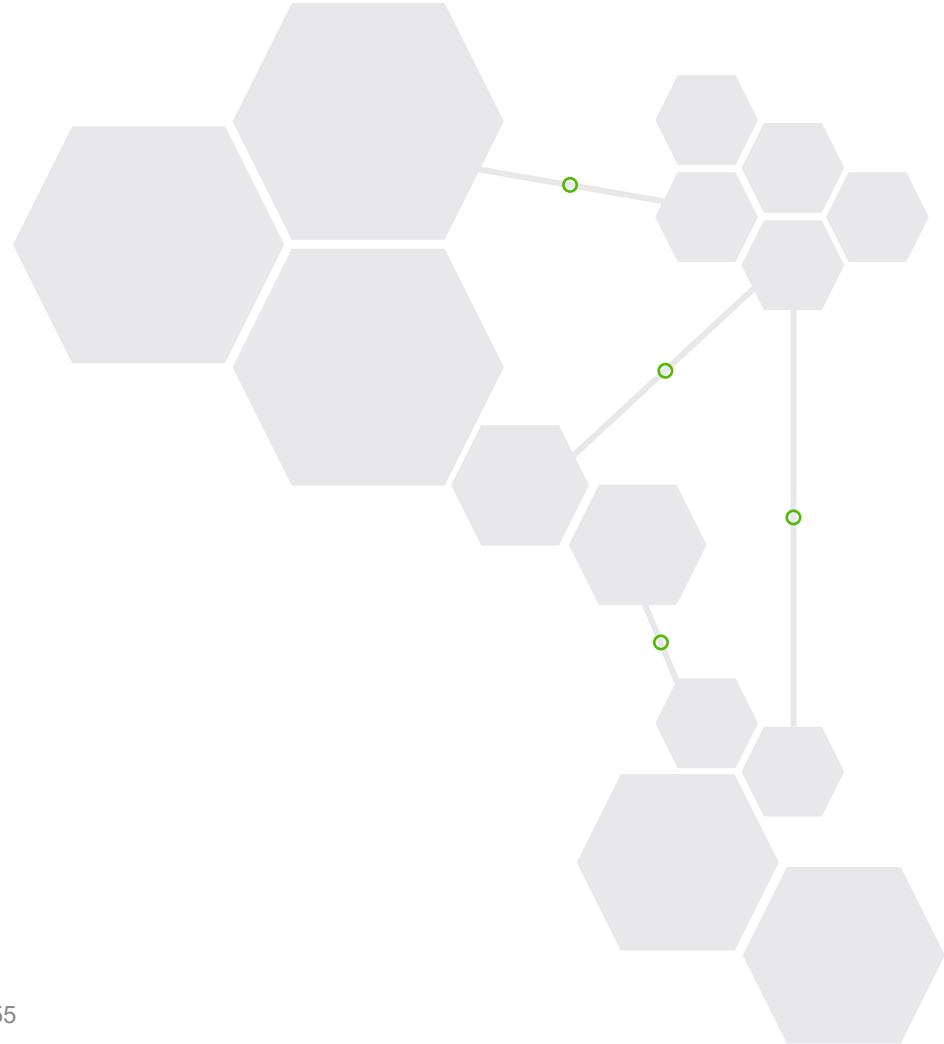


© StorageOS Ltd.

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 evaluate options using the eight principles of cloud native storage:

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

© StorageOS Ltd.

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.

© StorageOS Ltd.

StorageOS resources

Browser-based tutorials

- my.storageos.com/main/tutorials

Quickstart

- storageos.com/kubernetes

Slack channel

- slack.storageos.com

We’re hiring!

© StorageOS Ltd.

STORAGEOS

Cloud Native London, Tue 12 Dec

Speakers from UK Gov Digital Service, Bitnami, RealKinetic

meetup.com/Cloud-Native-London

STORAGEOS

Questions

Slides at oicheryl.com

© StorageOS Ltd.