失敗しない！
Kubernetes向け
ストレージ選び5つのポイント

Yukinori Sakashita

Z Lab

自己紹介

坂下 幸徳 (博士)

Yukinori Sakashita (Ph.D)

- ヤフー株式会社/ゼットラボ株式会社
- SNIA日本支部 技術委員会 副委員長

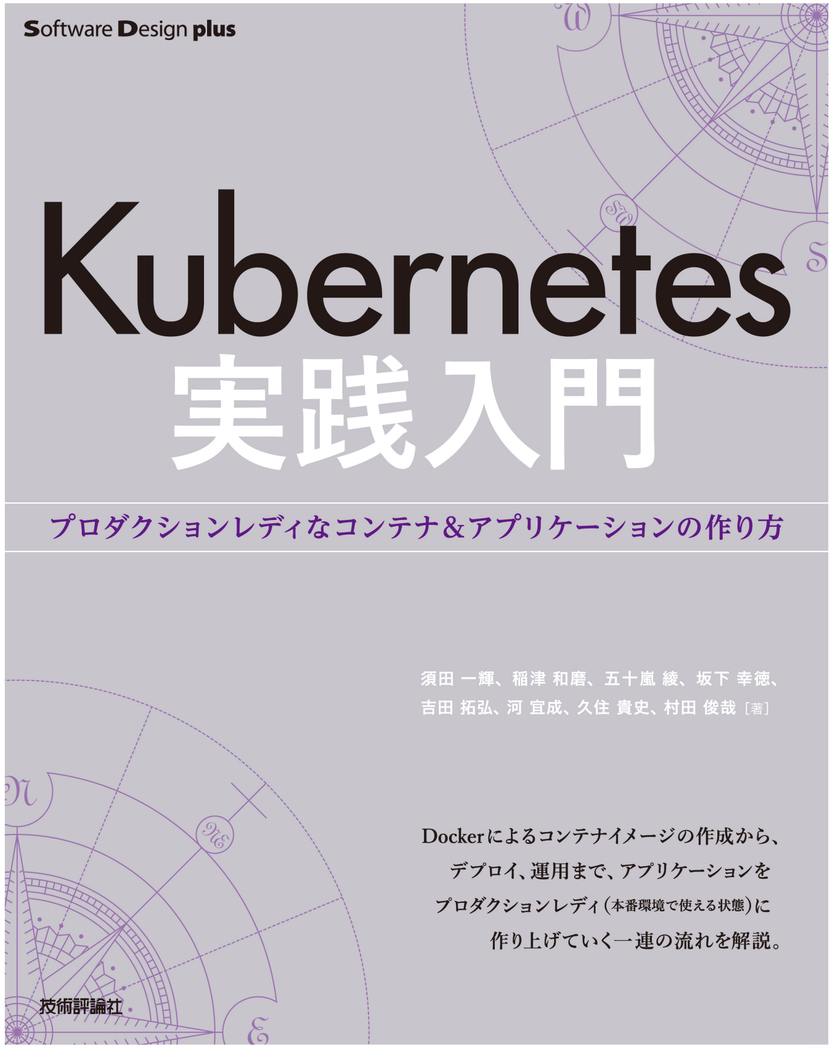
- 技術/研究エリア
  - 運用管理技術
  - クラウド
  - サーバ/ストレージ
  - 自動/自律コンピューティング
  - AI/ML (NN, HMM, Bayesian Network)

- 職歴
  - 総合電機メーカー
    - ✓ 研究所 (主任研究員)
    - ✓ 海外研究所@シリコンバレー (ラボ長)
  - サーバベンダー(テクニカルSE)

Z Lab

出版

- 2019年3月出版
- コンテナイメージの作成からデプロイ・運用までアプリケーションをプロダクションレディに作り上げる一連の流れを解説



Z Lab

データ保証

セキュリティ

性能

コスト

運用体制

スキル

バックアップ

DR/BCP

アプライアンス製品

SDS

OSSストレージ

SSD/HDD

標準仕様

Agenda

1. ヤフー/Z Labの取り組み
2. Kubernetesにおけるデータ永続化の取り組み
3. Kubernetes向けストレージ選び5つのポイント
4. まとめ

Z Lab

ヤフー/Z Labの取り組み

Z Lab

- 2015年に設立されたヤフー株式会社の100%子会社
- ヤフーの次世代インフラを研究開発
- ヤフー株式会社向けにKubernetes as a Serviceを開発
- https://zlab.co.jp

Z Lab

Yahoo! JAPAN

- 100以上のWebサービス(ニュース, ヤフオク、天気など)を提供
- Webサービスを支えるインフラ環境
  - 400+ Kubernetes クラスタ
  - 50,000+ コンテナ

※2019年7月時点

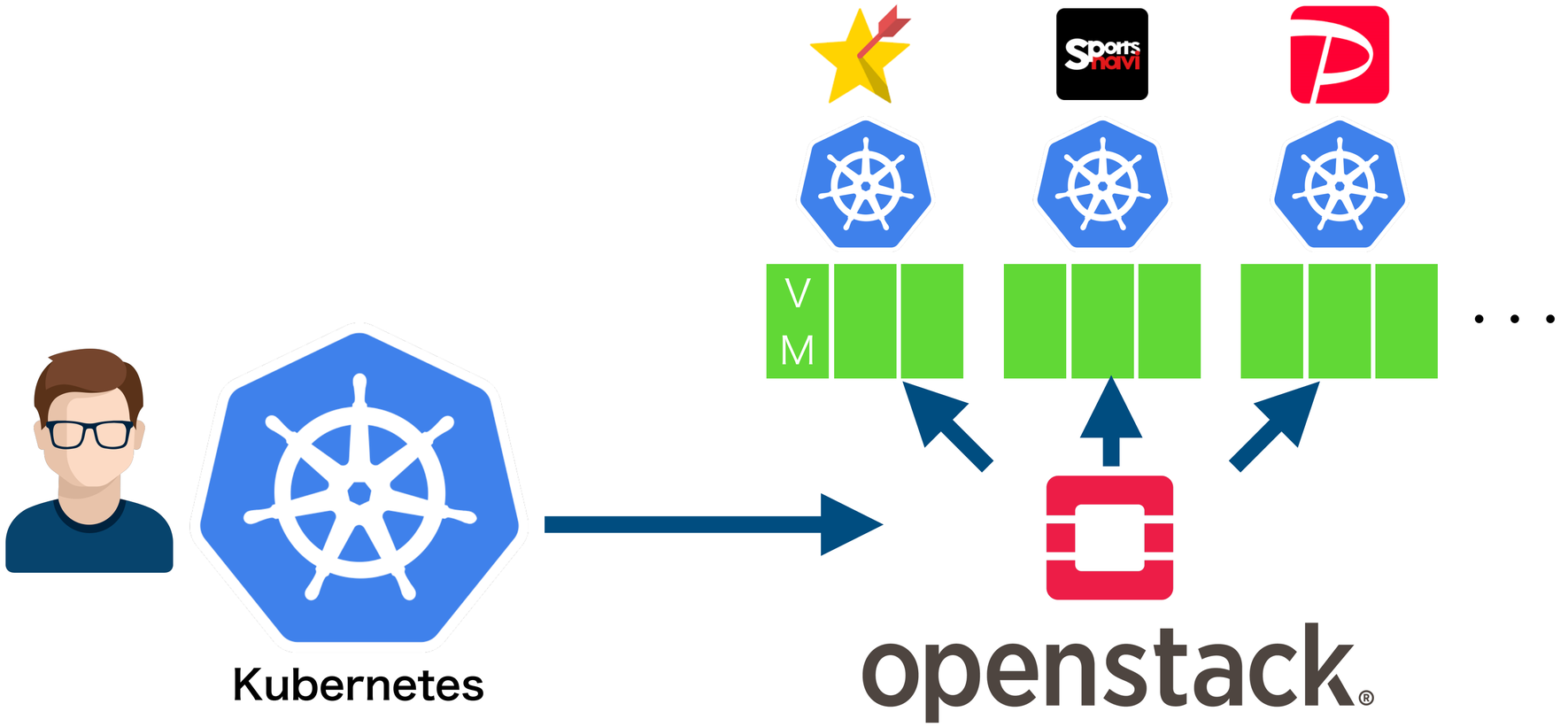
Z Lab

Kubernetes as a Service

- Yahoo! JAPANのWebサービスの一部はKubernetes上で稼働
- マネージドKubernetesサービス(Kubernetes as a Service)を開発
  - 2017/8 β
  - 2018/8 GA
- Kubernetes as a Service(CaaS)によりProductivityとSustainabilityが向上
- 400+ Kubernetesクラスタを運用中

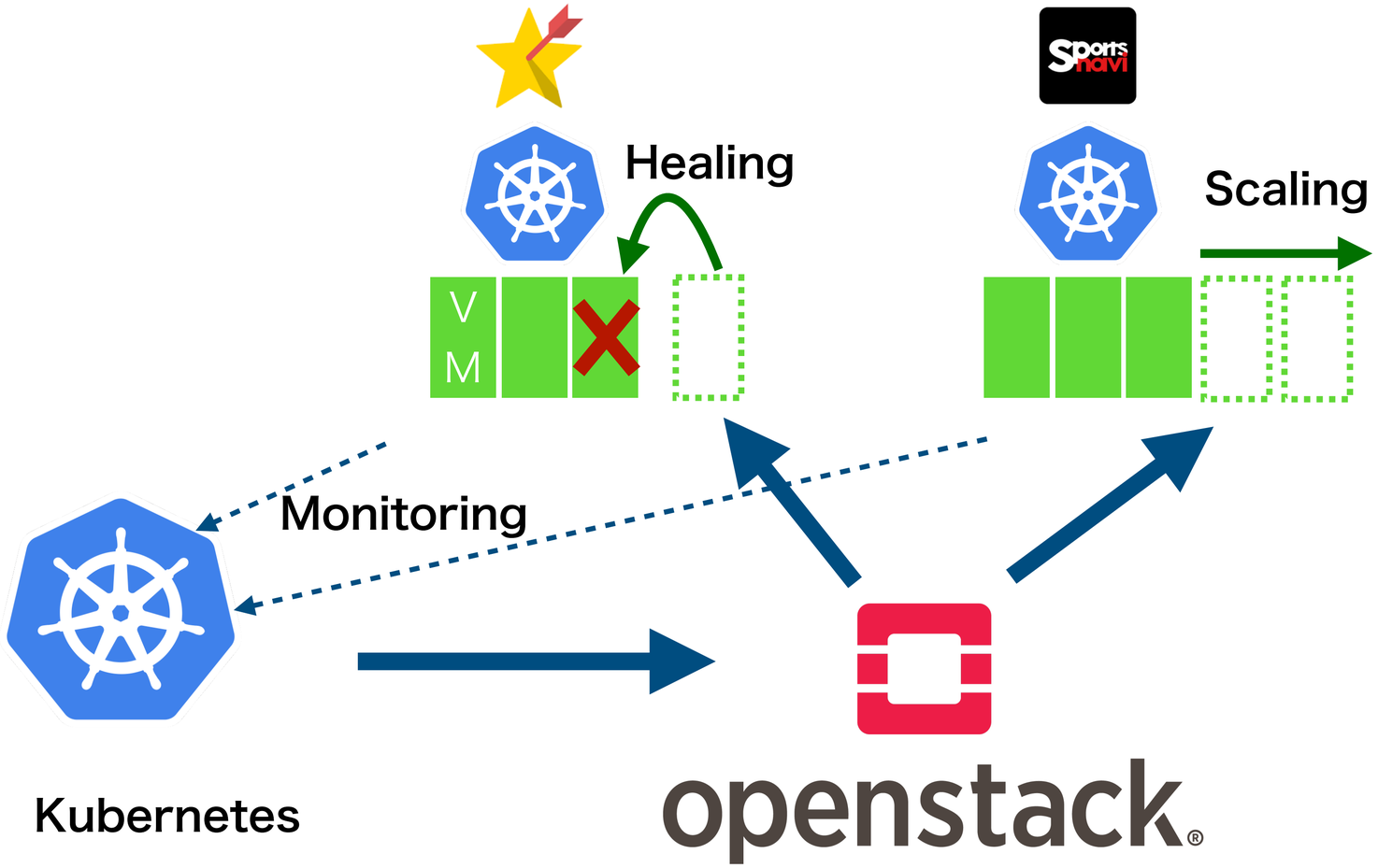
Z Lab

Architecture of CaaS



Z Lab

Self-healing & Scaling



Z Lab

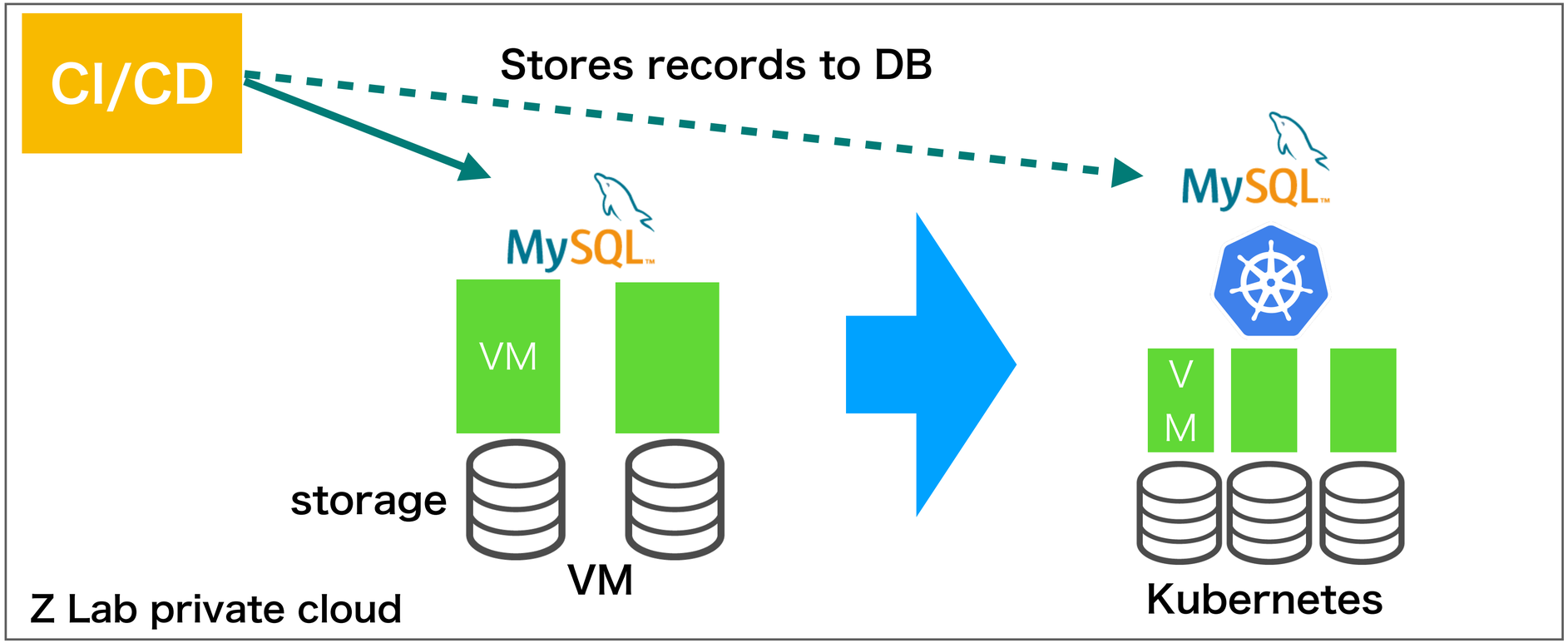
CaaSのターゲットアプリ

- 現時点ではステートレスアプリケーションをKubernetes上で動作
- Webサービスはステートレスとステートフルのアプリケーションで構成
- Z Lab にてCaaS向けのステートフル環境を試作
- ステートフルアプリケーションへ挑戦

Z Lab

ステートフル環境の事例

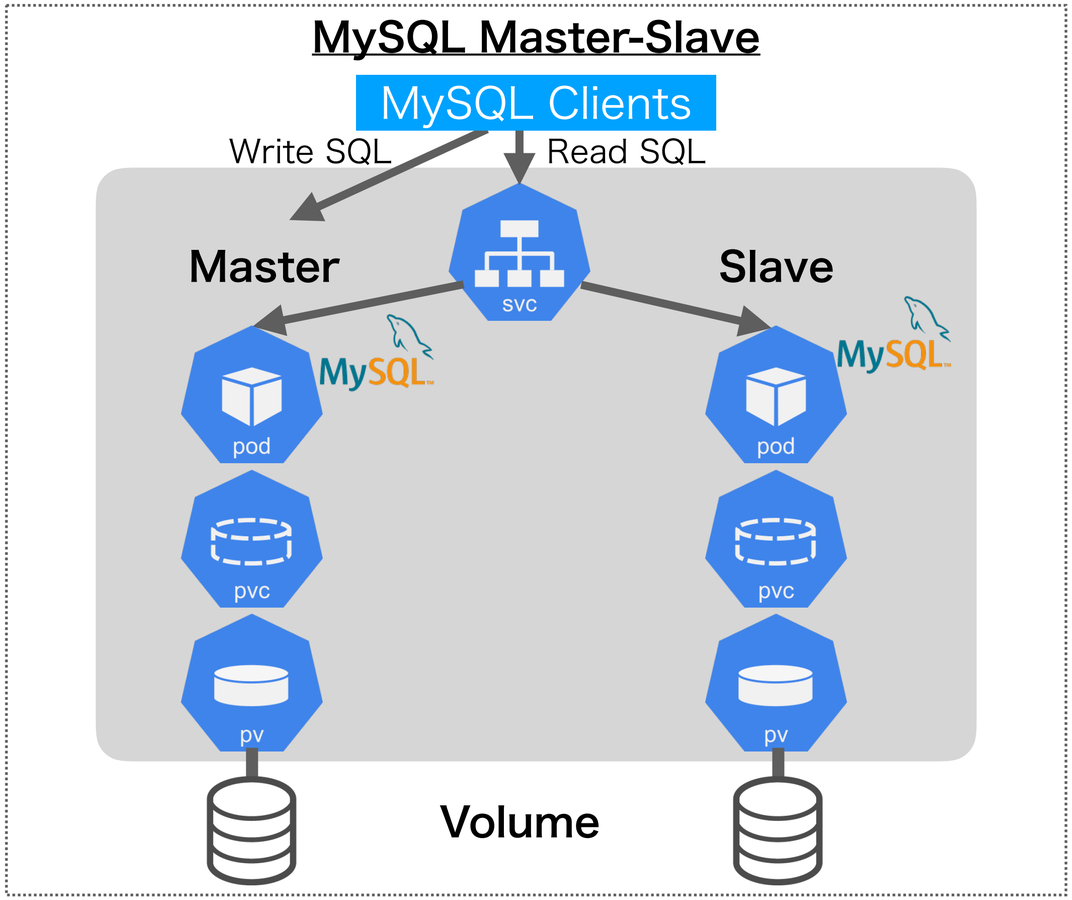
- CaaS向けにステートフル環境を構築
- Z Labにて運用中のMySQLをVMからKubernetesへ移行
- 動作は問題なし。スケールアウトやアップデートの管理性が向上

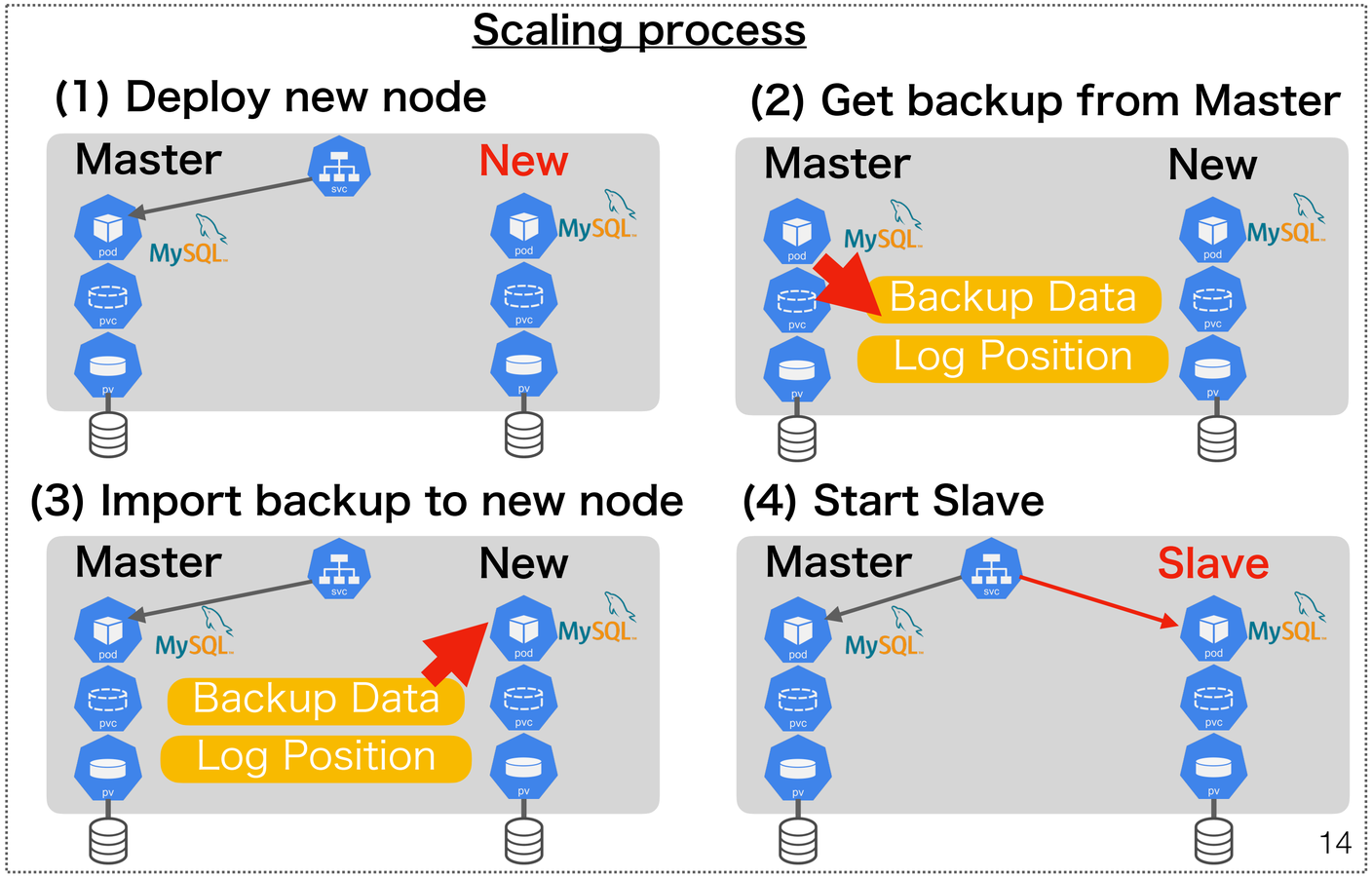


Z Lab

スケールするMySQLの一例

- MySQL(Master-Slave構成)のSlave追加時の構築ステップを実装
- MySQLの構築ステップの知識がない管理者でもスケール可能
  (e.g. $ kubectl scale —replicas=3 sts/mysql)

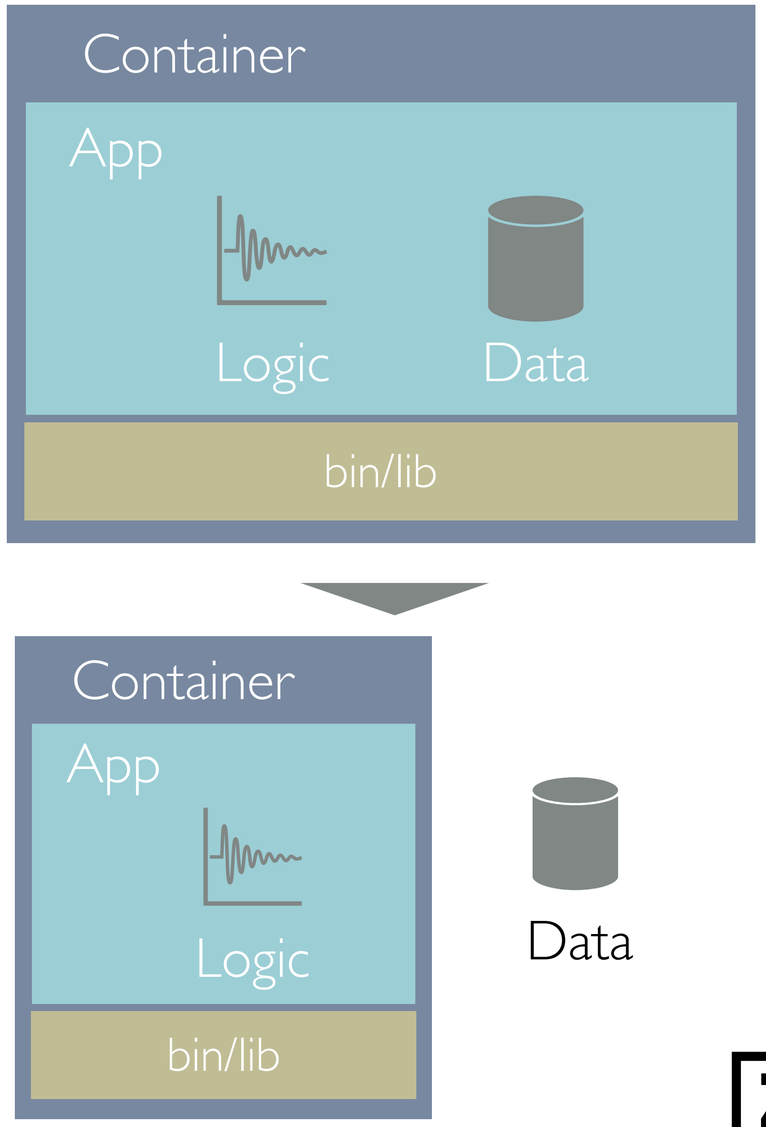




Kubernetesにおける
データ永続化の取り組み

コンテナでのデータ永続化とは

- 過去
  - コンテナはステート(データ)を持つ
    アプリが苦手
  - コンテナがデプロイされる都度
    データが初期化
- 現在 (2017年頃〜)
  - ロジックとデータを分離
  - データを外部ストレージに保存(永続化)

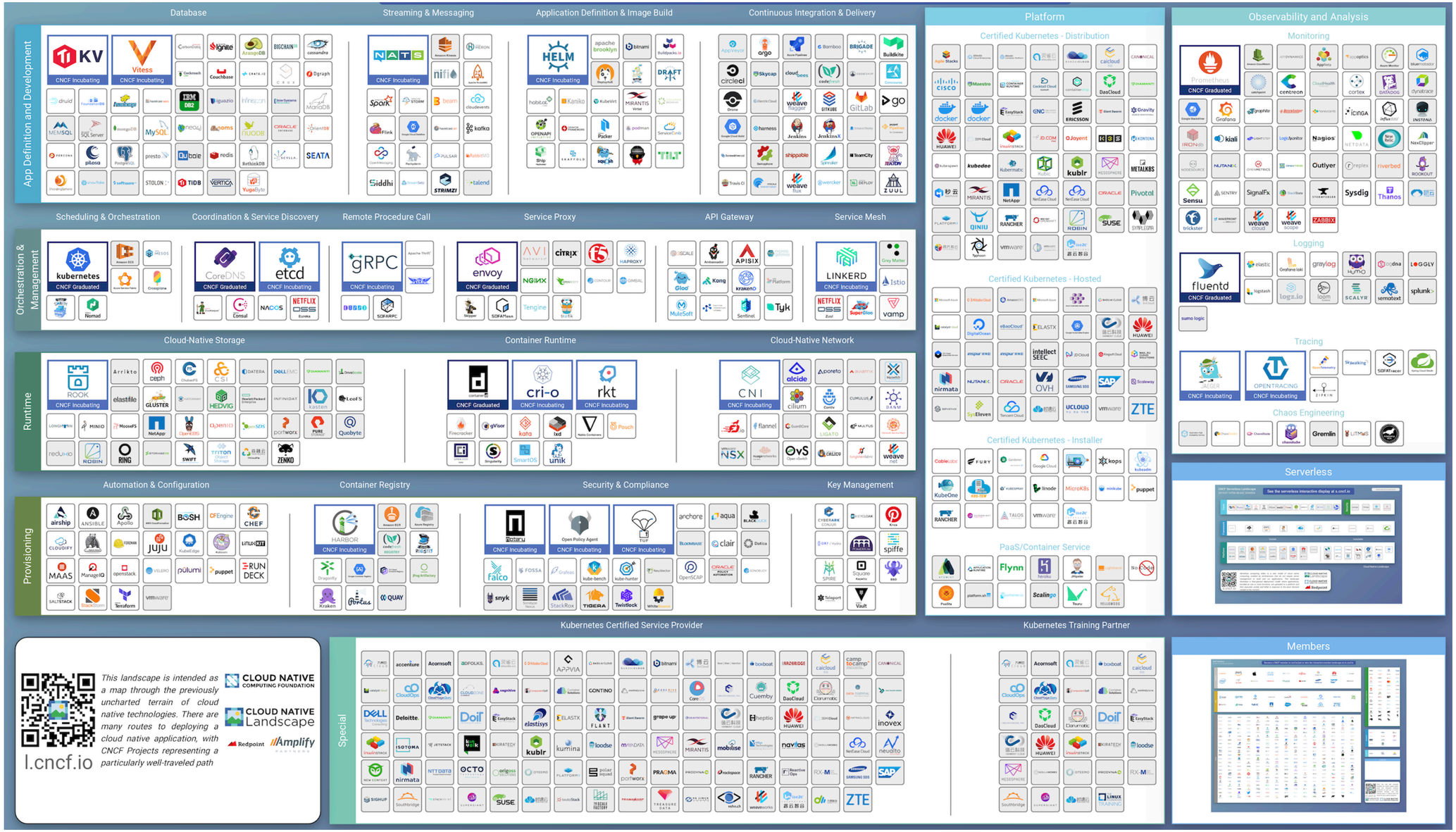


Z Lab

Kubernetes

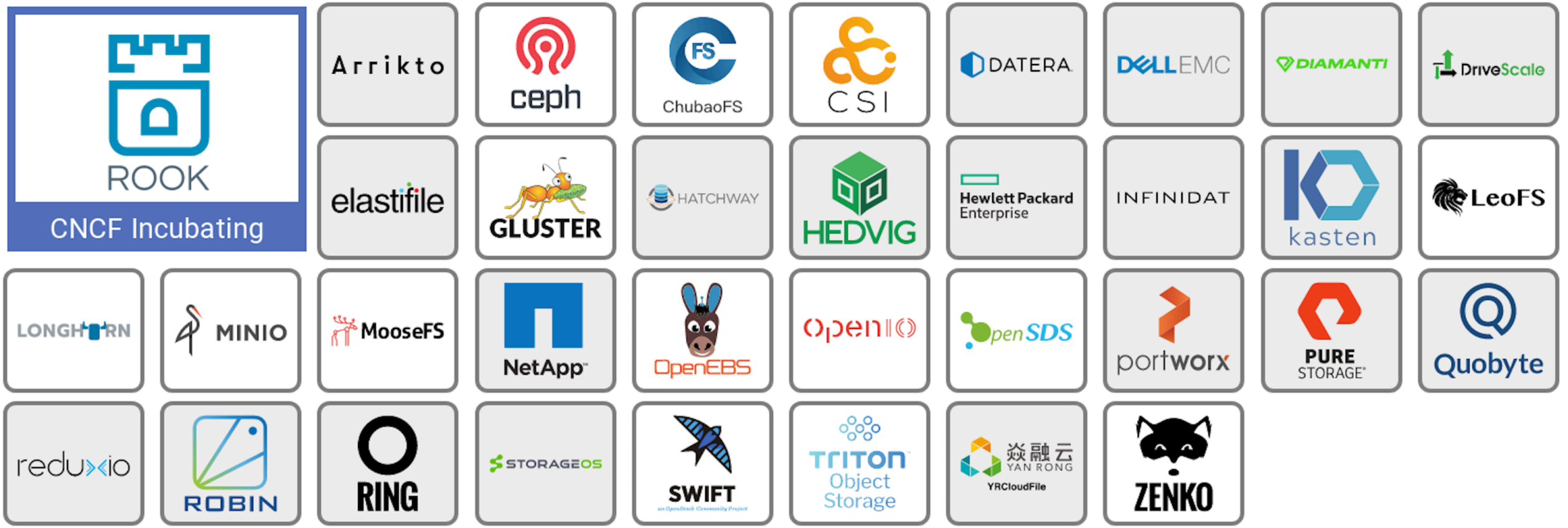
- コンテナオーケストレーションの
  デファクトスタンダード
- CNCFのエコシステムの中心的存在
  (Cloud Native Computing Foundation)
- 主な機能
  - Self-healing
  - Horizontal scaling
  - Service discovery and load balancing
  - Automated rollouts and rollbacks
  - Storage orchestration

Cloud Native Landscape



Z Lab

Cloud-Native Storage



Ref: CNCF landscape(https://github.com/cncf/landscape)

Z Lab

ストレージ機能のサポート履歴

- Kubernetes v1.6 (’17/3)
  - Dynamic Provisioning
- Kubernetes v1.8 (’17/9)
  - Flex Volume
- Kubernetes v1.9 (’17/12)
  - CSI(Alpha)
- Kubernetes v1.10 (’18/3)
  - Local Volume (Beta)
- Kubernetes v1.11 (’18/6)
  - Volume Resizing
- Kubernetes v1.12 (’18/9)
  - Volume Snapshot (Alpha)
  - Topology-Aware Volume Provisioning (Beta)
- Kubernetes v1.13 (’18/12)
  - CSI (GA)
- Kubernetes v1.14 (’19/3)
  - Local Volume (GA)
- Kubernetes v1.15 (’19/6)
  - CSI Volume Cloning (Alpha)

サポートしているVolumeタイプ

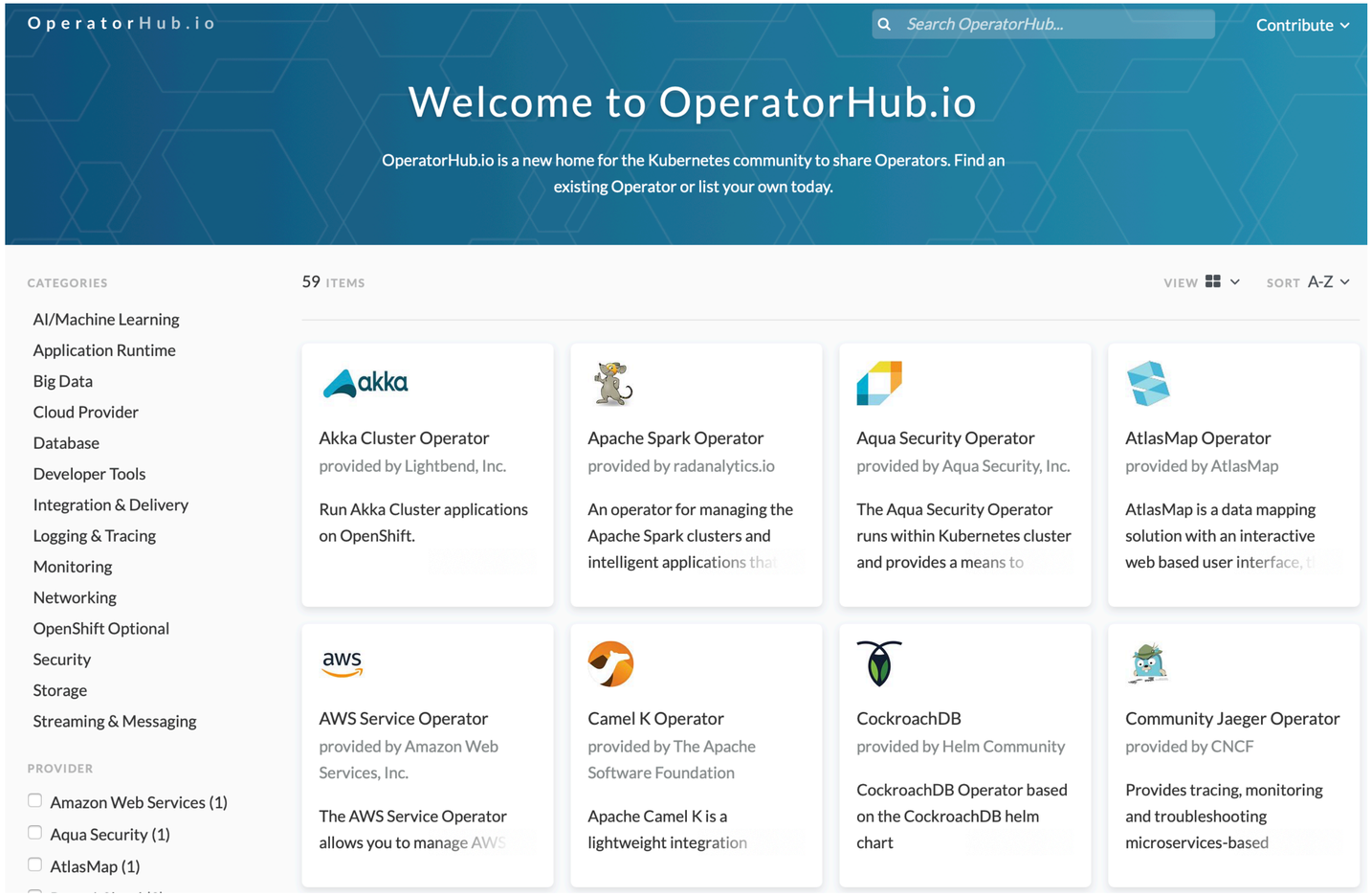
- awsElasticBlockStore
- azureDisk
- azureFile
- cephfs
- cinder
- configMap
- csi
- downwardAPI
- emptyDir
- fc(fibre channel)
- flexVolume
- flocker
- gcePersistentDisk
- gitRepo(deprecated)
- glusterfs
- hostPath
- iscsi
- local
- nfs
- projected
- portworxVolume
- quobyte
- rbd
- scaleIO
- secret
- storages
- vsphereVolume

※2019/7現在

OperatorHub.io

- Kubernetes Operatorのパブリックレジストリ
- 2019/3公開

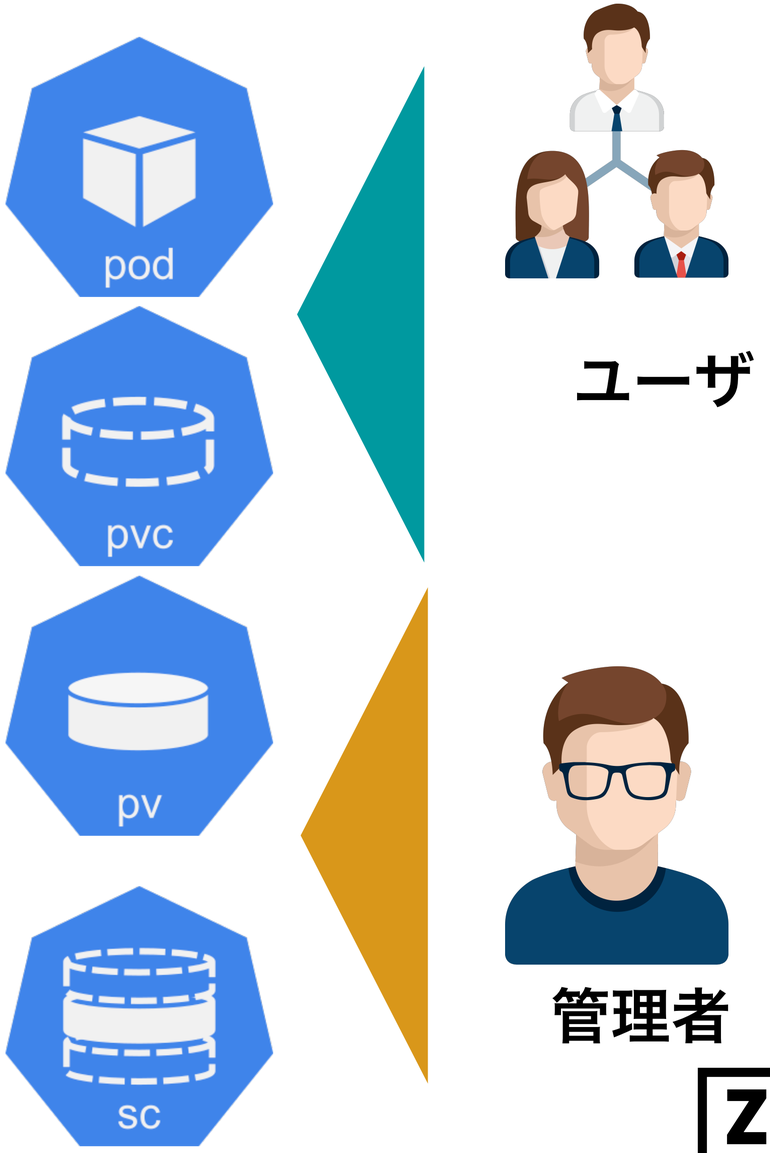
※2019/7現在



カテゴリー	登録数
AI/Machine Learning	2
Application Runtime	3
Big Data	4
Cloud Provider	2
Database	16
Developer Tools	2
Integration&Delivery	6
Logging&Tracing	6
Monitoring	8
Networking	1
OpenShift Optional	5
Security	9
Storage	7
Streaming&Messaging	2

ストレージのモデル

- ベンダーニュートラルなモデル
  - PersistentVolumeClaim
  - PersistentVolume
  - StorageClass
- 管理者とユーザの役割を考慮したモデル
- ストレージ装置の操作なしでVolumeを
  コンテナに自動でマウント



Z Lab

PersistentVolumeClaim(PVC)

- Volumeの要求仕様を表すリソース
  - Storage type (StorageClass名)
  - Volume size
  - Access mode

AccessMode	説明
ReadWriteOnce	1つのNodeからRead/Writeでマウントできる
ReadOnlyMany	複数のNodeからRead Onlyでマウントできる
ReadWriteMany	複数のNodeからRead/Writeでマウントできる

Z Lab

PersistentVolume(PV)

- Volumeを表すリソース
  - Storage type (StorageClass名)
  - Size
  - Access modes
  - Reclaim policy
    - Retain
    - Recycle
    - Delete

Z Lab

StorageClass(SC)

- ストレージプールを表すモデル
  - Name
  - Reclaim policy
  - Provisioner
- Provisionerがストレージ装置のAPIを呼び出し

Z Lab

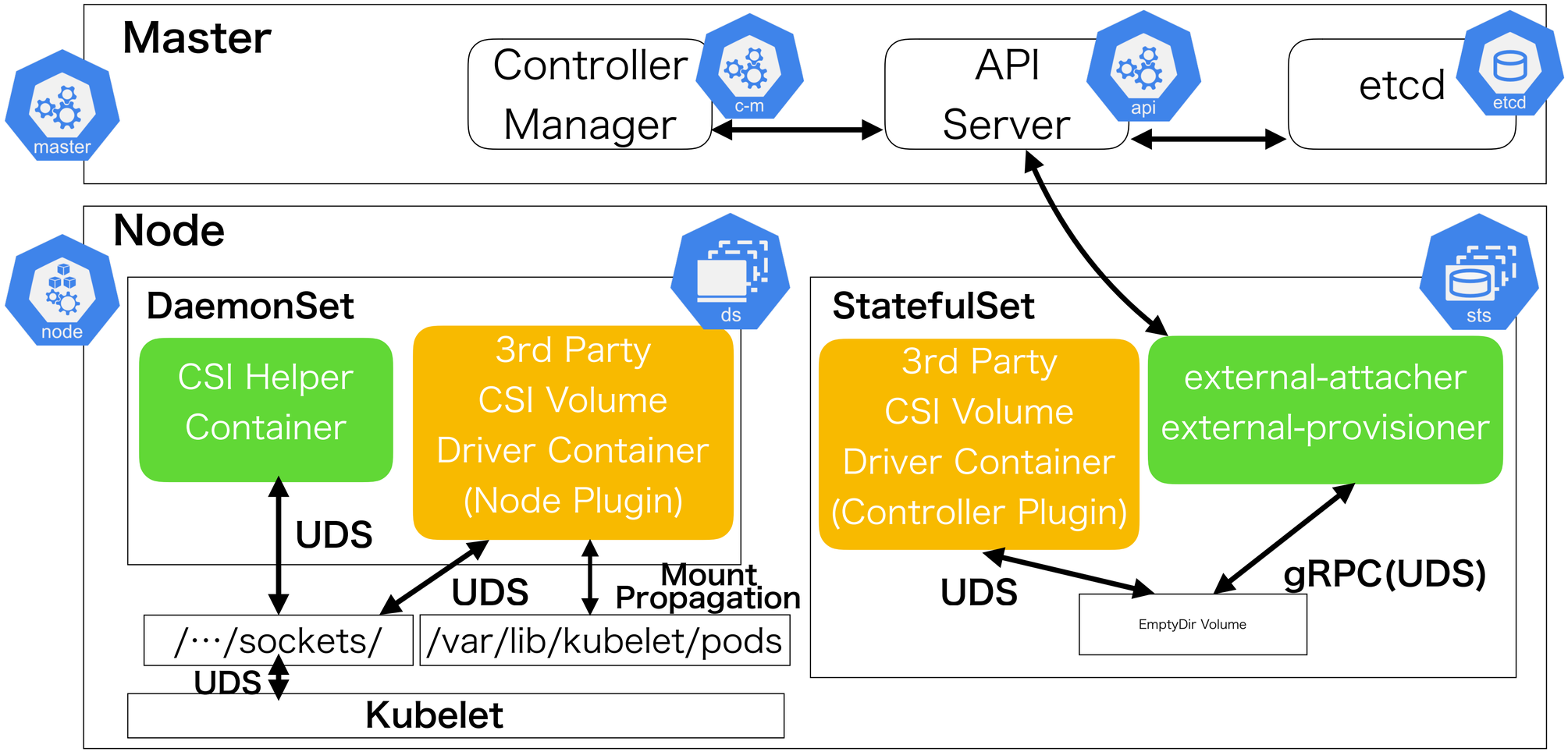
CSI(Container Storage Interface)

- コンテナオーケストレーション向けの標準仕様
- KubernetesではStorageClassのProvisionerのインターフェースとして採用
- 2017年12月に最初のパブリックリリース
- 主要なコンテナオーケストレーションが採用
  (Kubernetes, CloudFoundry, Mesos, Docker, etc)
- Kubernetesのソースツリー(in-tree)で管理されていたストレージ装置に
  関連する実装が、3rdベンダで独自に開発/提供することが可能

CONTAINER
STORAGE
INTERFACE

Z Lab

CSIのアーキテクチャ



External Component - Created by Kubernetes team

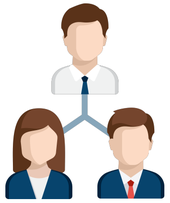
External Component - Created by 3rd Party Vendor

Z Lab

ストレージの使い方

- Volumeの割り当てには大きく2つの方法
- Manual Provisioning
  - 管理者がStorageClassとPersistentVolumeを事前にデプロイ
  - ユーザはPersistentVolumeClaimを使ってPersistentVolumeを選択
- Dynamic Provisioning
  - 管理者はStorageClassのみ事前に準備
  - ユーザがPersistentVolumeClaimをデプロイすると PersistentVolumeが自動生成

Dynamic Provisioning

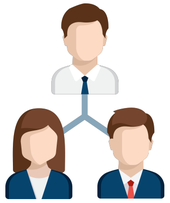


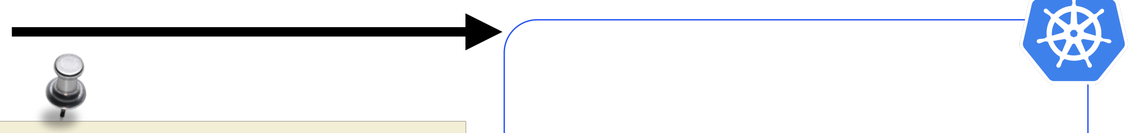
1. Deploy

```
...
 spec:
     containers:
     - name: mysql
       image: mysql:5.7.22
...
       volumeMounts:
       - name: mysql-data
         mountPath: /var/lib/mysql
 volumeClaimTemplates:
 - metadata:
     name: mysql-data
   spec:
     accessModes: ["ReadWriteOnce"]
     storageClassName: gold
     resources:
       requests:
         storage: 1Gi
```

Z Lab

Dynamic Provisioning



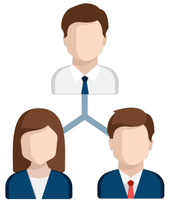


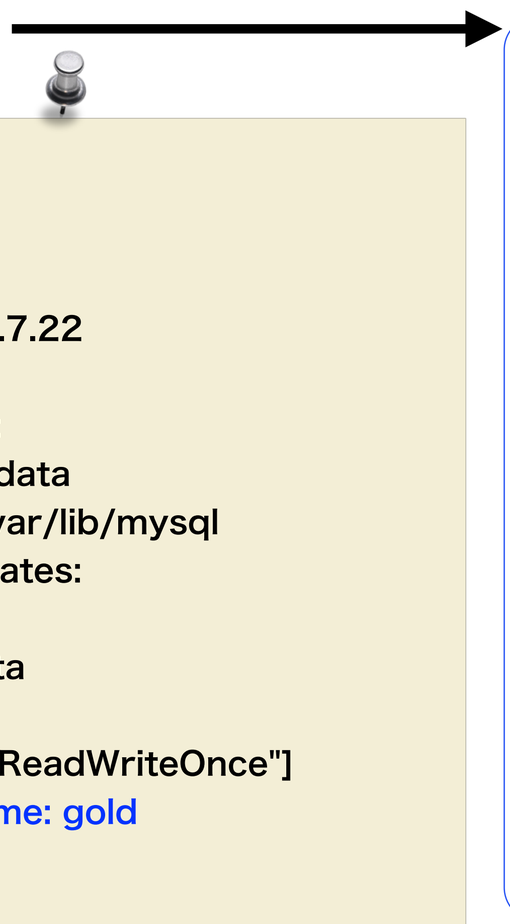
```
...
 spec:
     containers:
     - name: mysql
       image: mysql:5.7.22
...
       volumeMounts:
       - name: mysql-data
         mountPath: /var/lib/mysql
 volumeClaimTemplates:
 - metadata:
     name: mysql-data
   spec:
     accessModes: ["ReadWriteOnce"]
     storageClassName: gold
     resources:
       requests:
         storage: 1Gi
```


2. Create PVC

Z Lab

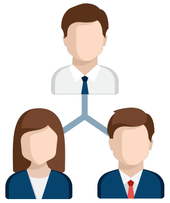
Dynamic Provisioning



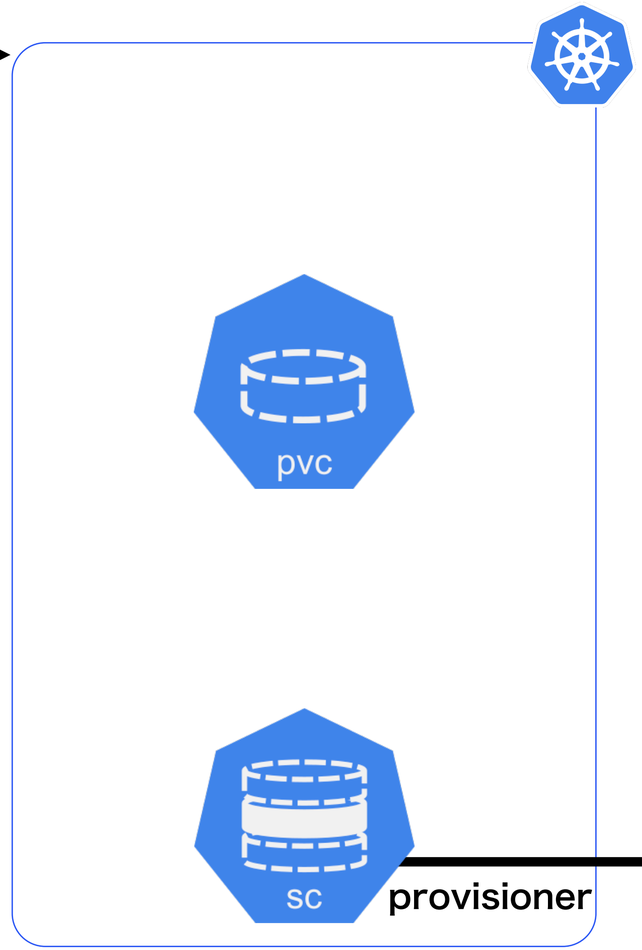


Z Lab

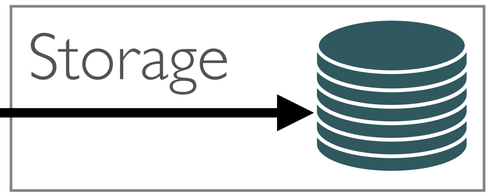
Dynamic Provisioning



```
...
 spec:
    containers:
    - name: mysql
      image: mysql:5.7.22
...
      volumeMounts:
      - name: mysql-data
        mountPath: /var/lib/mysql
 volumeClaimTemplates:
 - metadata:
    name: mysql-data
   spec:
    accessModes: ["ReadWriteOnce"]
    storageClassName: gold
    resources:
      requests:
        storage: 1Gi
```

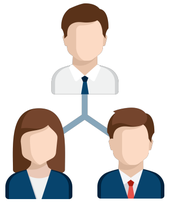


4. Create Volume



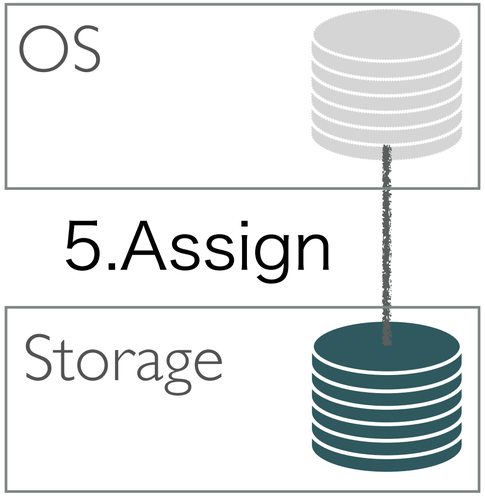
Z Lab

Dynamic Provisioning



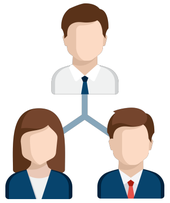
```
...
 spec:
    containers:
    - name: mysql
      image: mysql:5.7.22
...
       volumeMounts:
       - name: mysql-data
         mountPath: /var/lib/mysql
 volumeClaimTemplates:
 - metadata:
    name: mysql-data
   spec:
    accessModes: ["ReadWriteOnce"]
    storageClassName: gold
    resources:
      requests:
        storage: 1Gi
```

6.Create PV



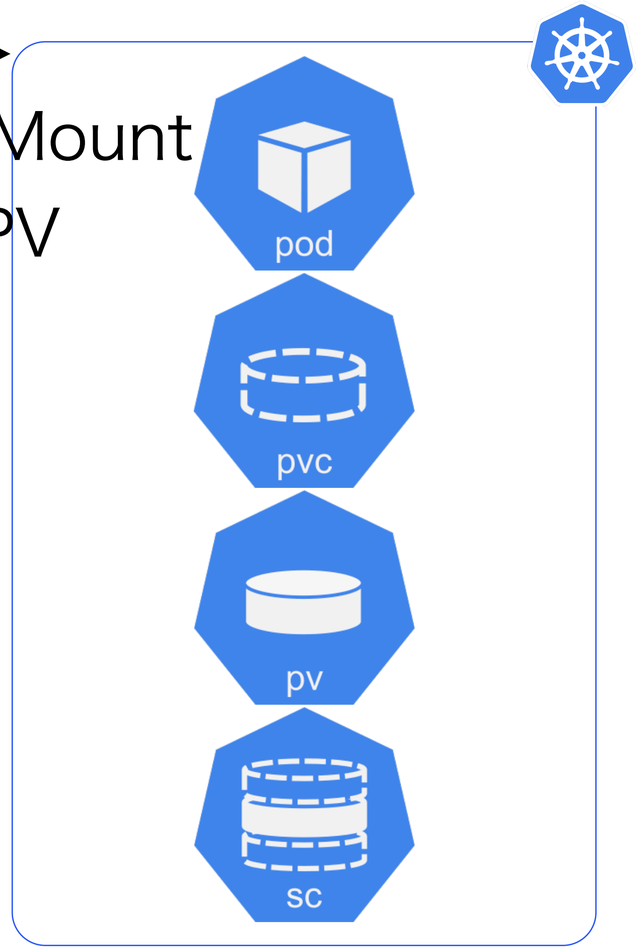
Z Lab

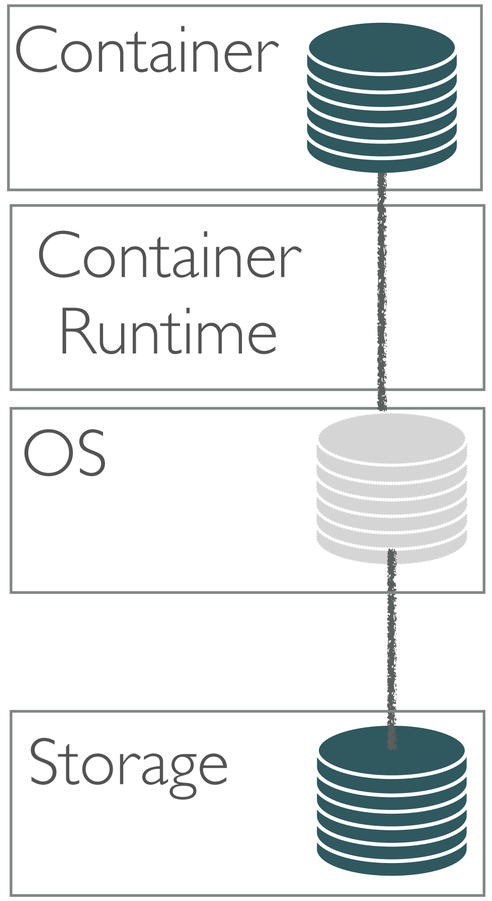
Dynamic Provisioning



```
...
 spec:
    containers:
    - name: mysql
      image: mysql:5.7.22
...
      volumeMounts:
      - name: mysql-data
        mountPath: /var/lib/mysql
 volumeClaimTemplates:
 - metadata:
    name: mysql-data
   spec:
    accessModes: ["ReadWriteOnce"]
    storageClassName: gold
    resources:
      requests:
        storage: 1Gi
```

7. Mount PV





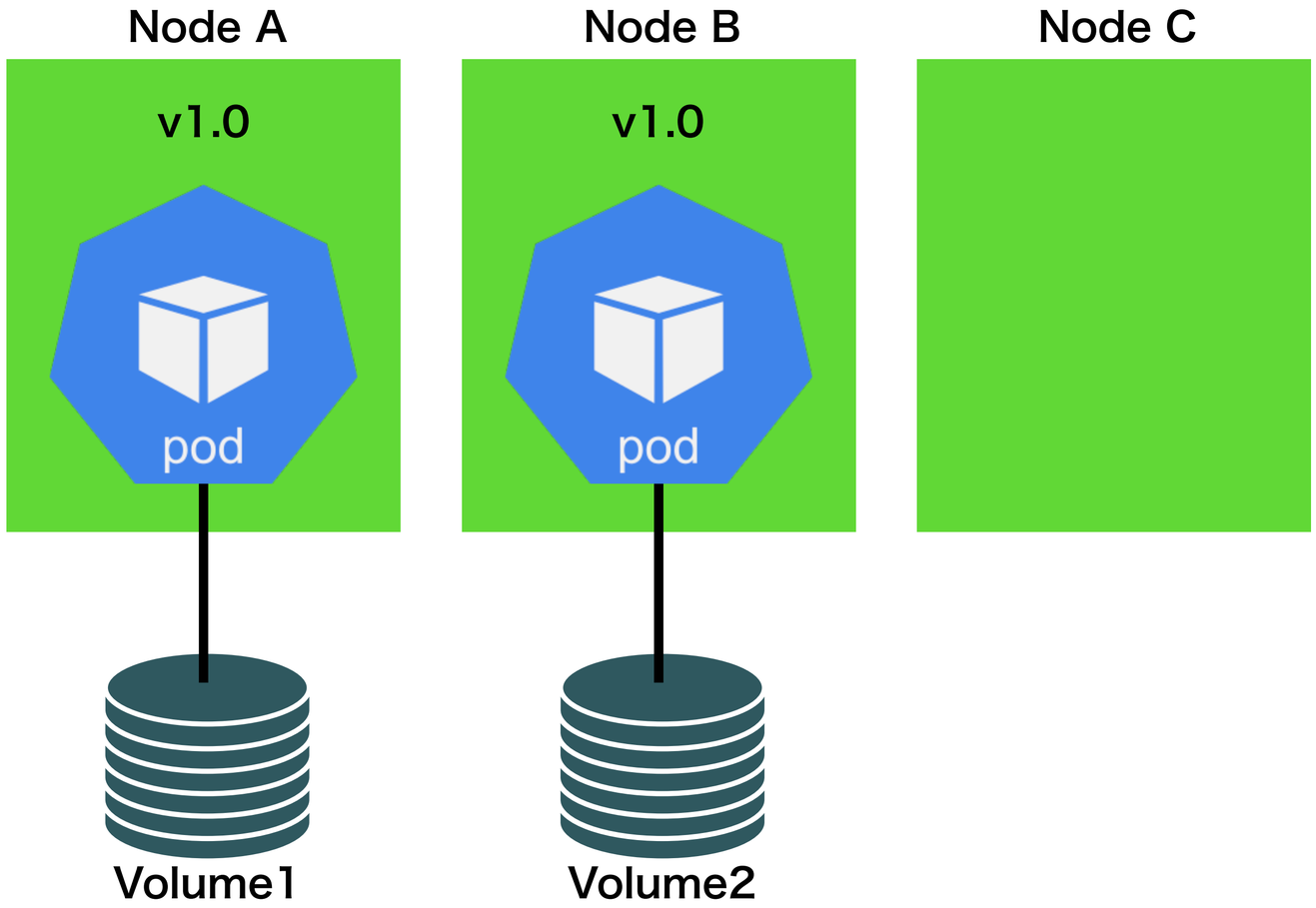
Z Lab

Rolling Update

- ローリングアップデートによりPodは再作成
- PersistentVolumeとPersistentVolumeClaimは再作成されない
- つまり、ストレージ装置のVolumeも削除されない
- Volumeに対しdetach/attachが実行

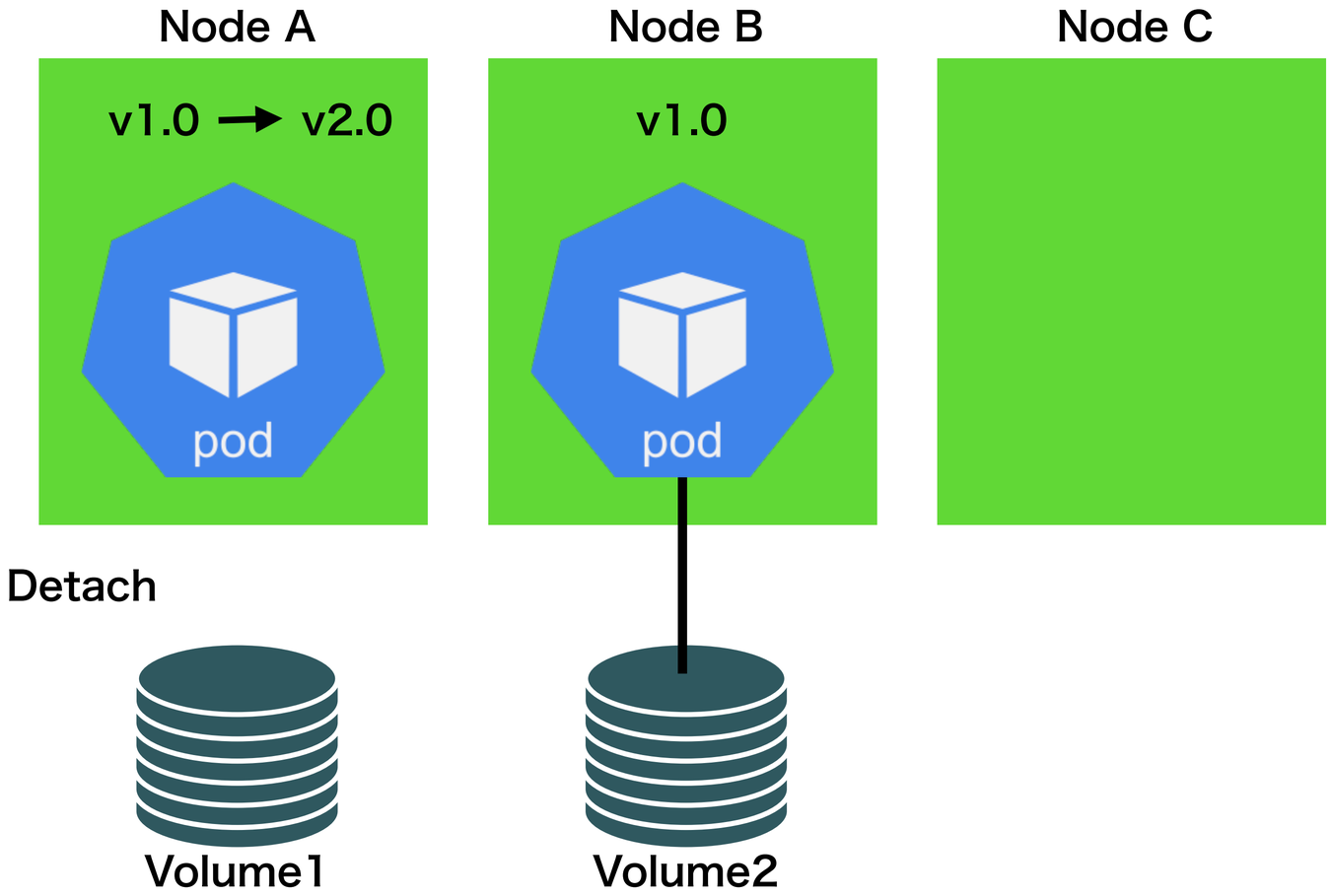
Z Lab

Rolling Update



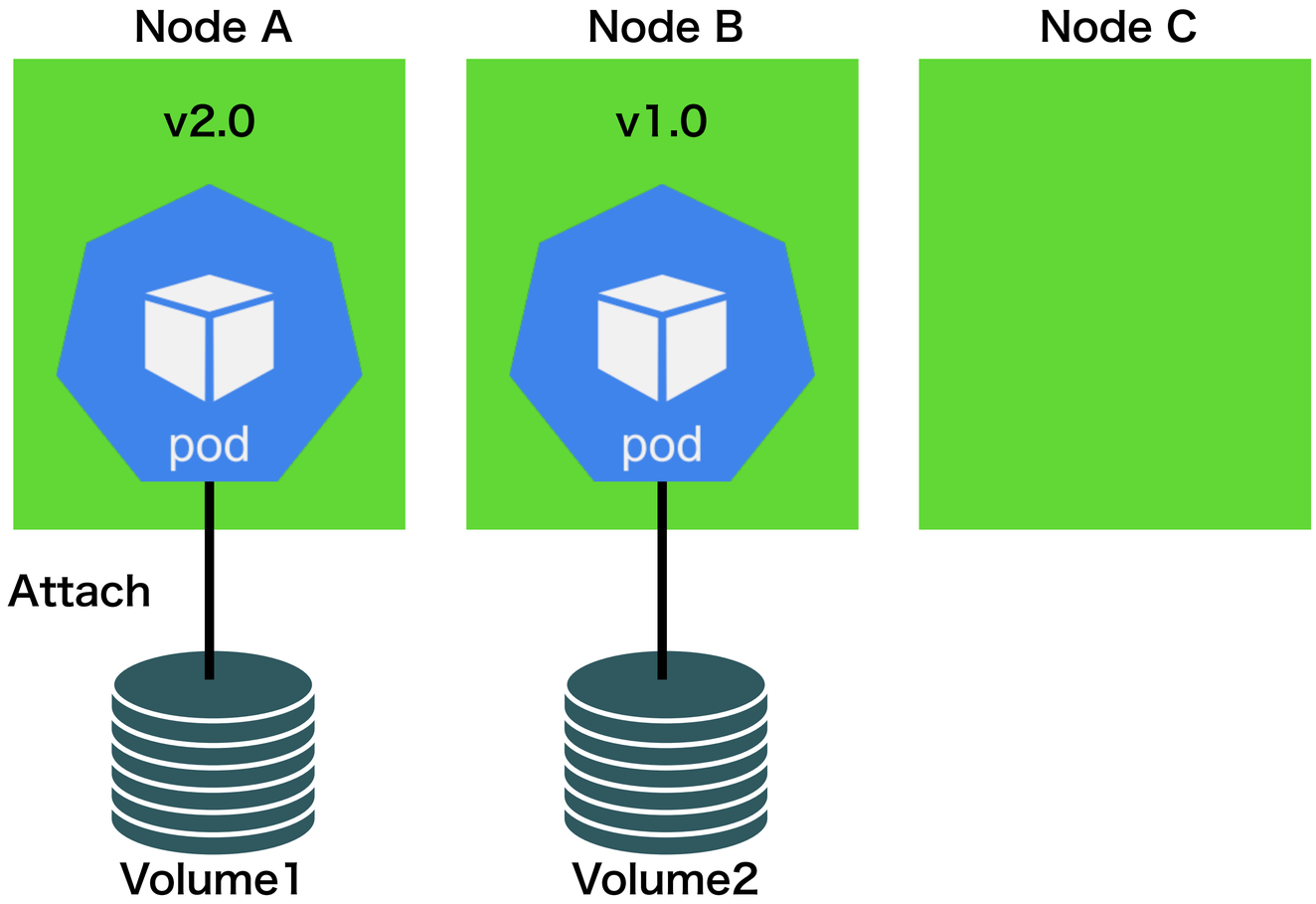
Z Lab

Rolling Update



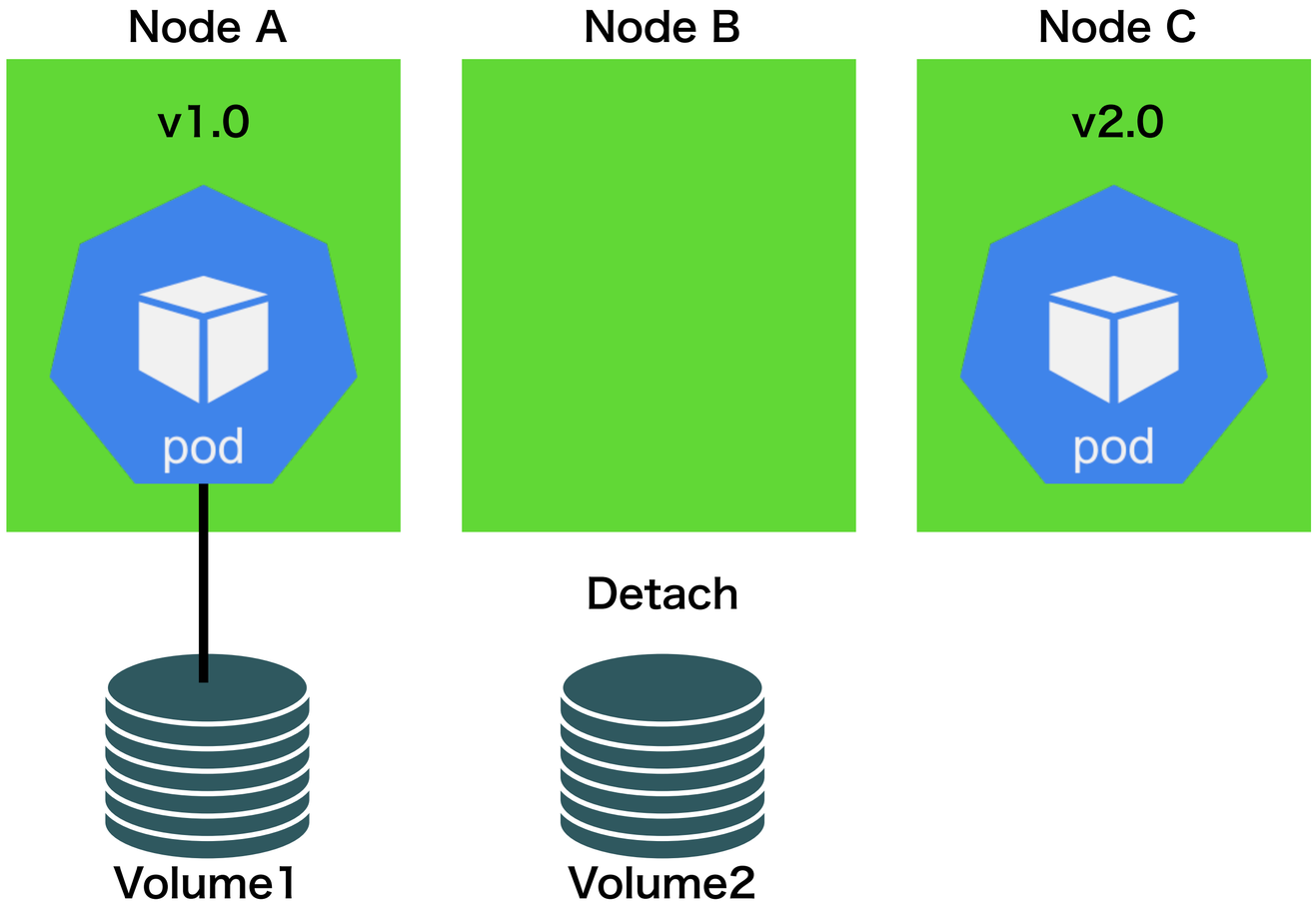
Z Lab

Rolling Update



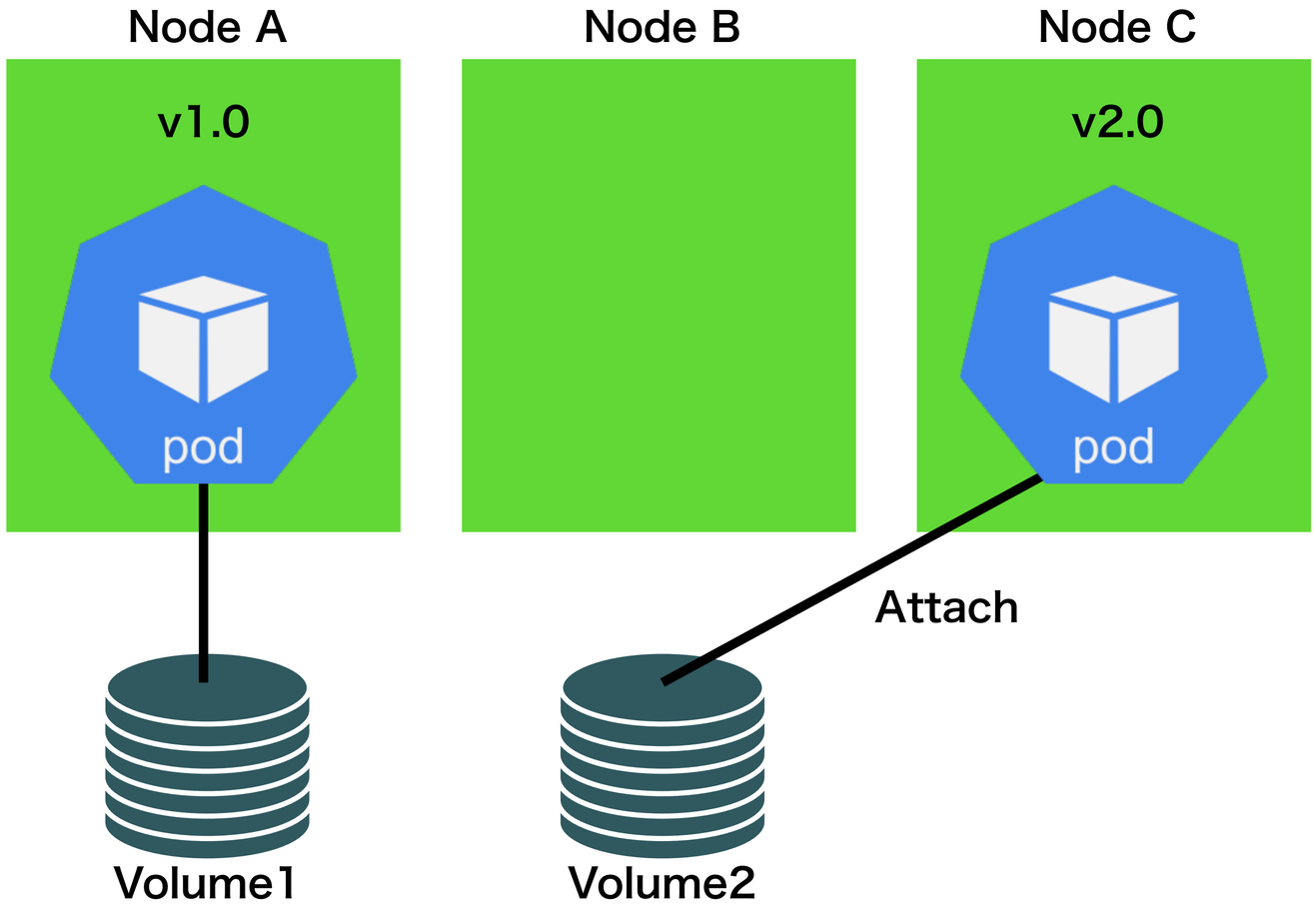
Z Lab

Rolling Update



Z Lab

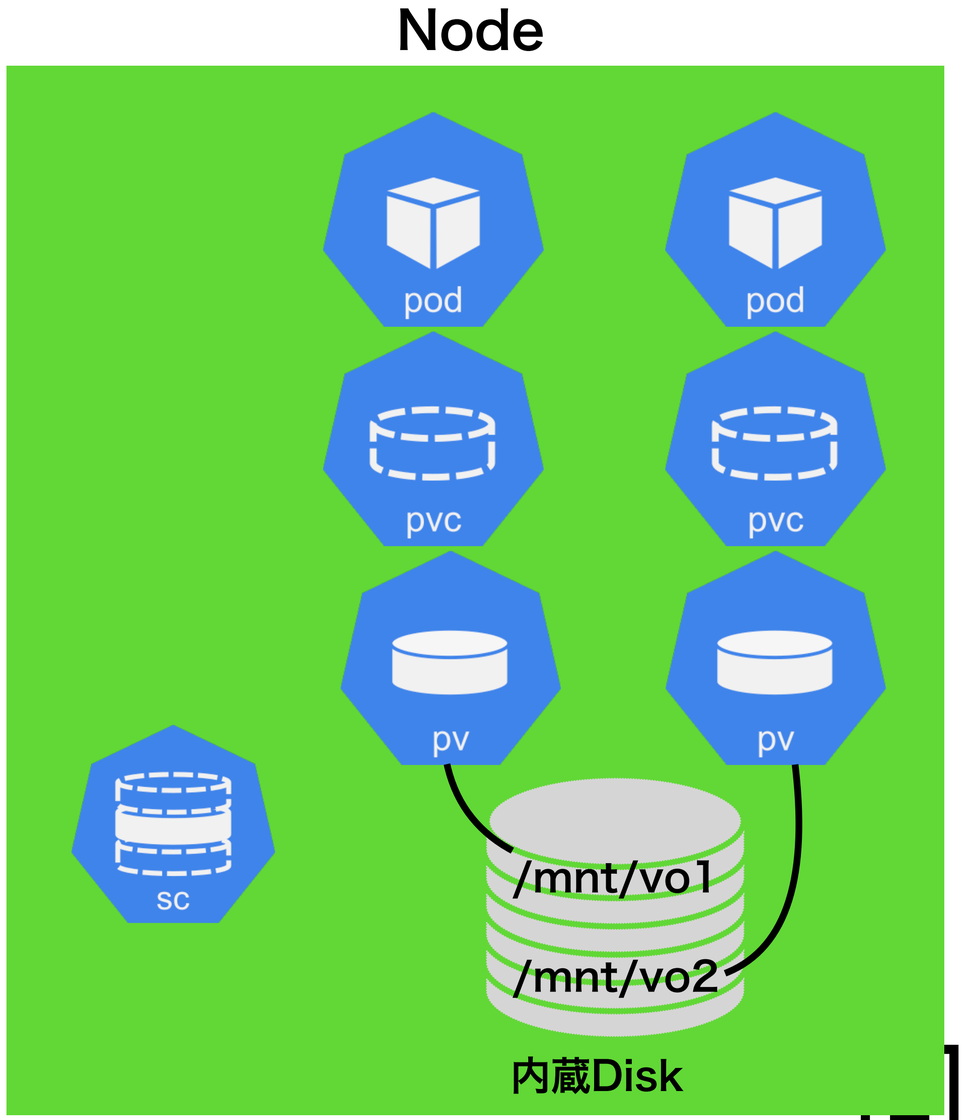
Rolling Update



Z Lab

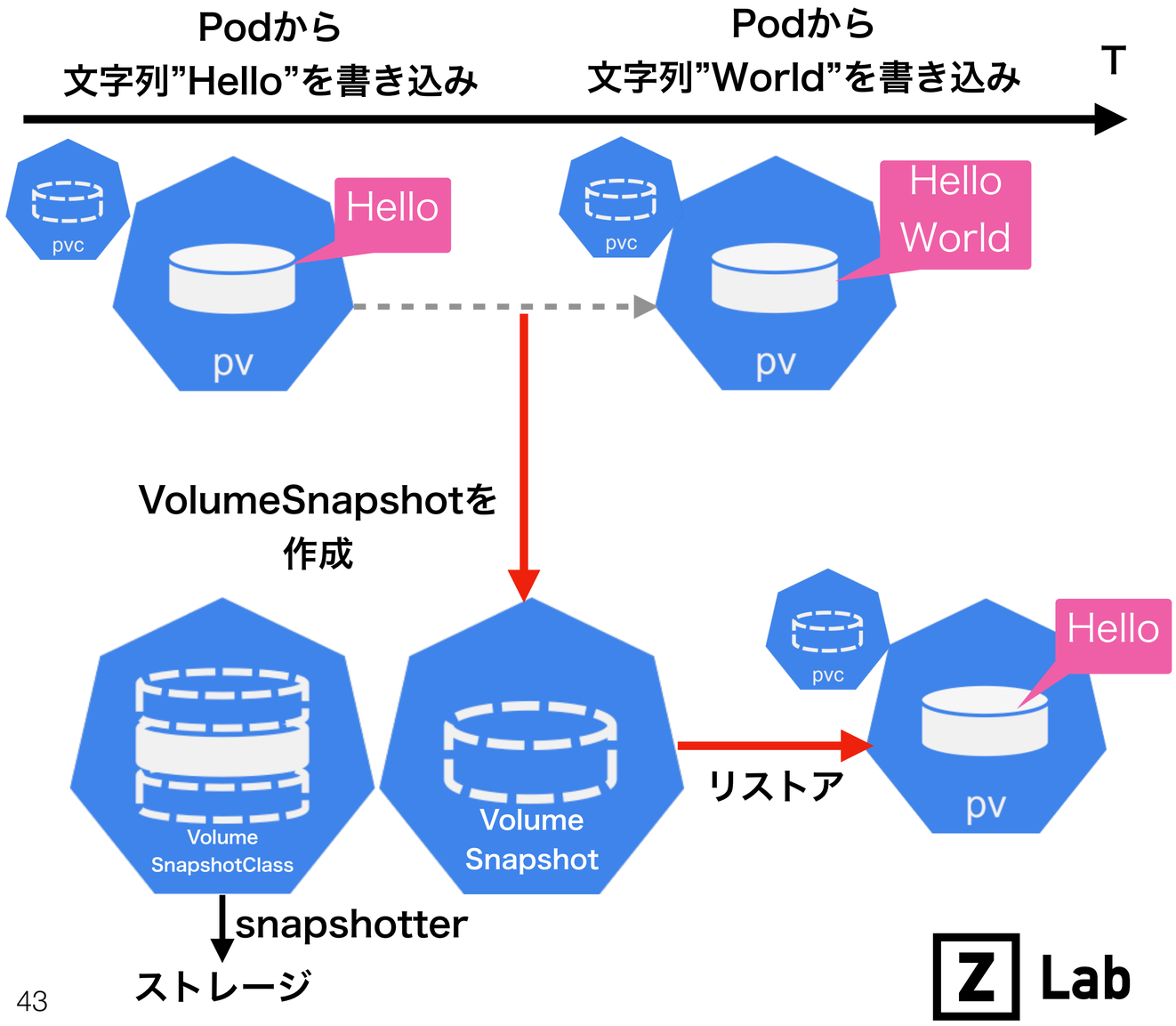
新機能1: Local Volume

- Kubernetes v1.14でGA
- Nodeの内蔵Diskのディレクトリを Podからマウント
- 外部ストレージの場合と同様に SC,PV,PVCを使った操作
- Node障害や削除などで保存したデータが削除されることに注意
- PodのNode移動も対応不可
- Dynamic Provisioningは未サポート



新機能2: Volume Snapshot

- Kubernetes v1.12にてAlpha
- VolumeSnapshotClass、VolumeSnapshotが新登場
- VolumeSnapshotをデプロイすることで、PersistentVolumeのデータをスナップショットとして保存
- リストアはPVCにてVolumeSnapshotをdataSourceとして指定しPersistentVolumeを作成



Z Lab

Kubernetes向けストレージ選び5つのポイント

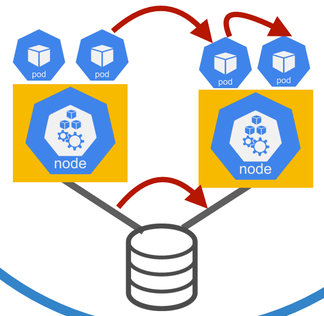
ストレージのシステム設計

- Public Cloudのサービスではストレージサービスが提供
  - AWS EBS Volume, Azure Disk, AzureFile, GCE Persistent Disk etc
- Private Cloudではインフラ管理者がストレージを準備
- ユーザはストレージの知識なしで利用
- ステートフルなアプリケーションを安心・安全にKubernetes上で
  動作させるために、管理者にはストレージの知識が必要
  - Kubernetesの知識＋ストレージの知識

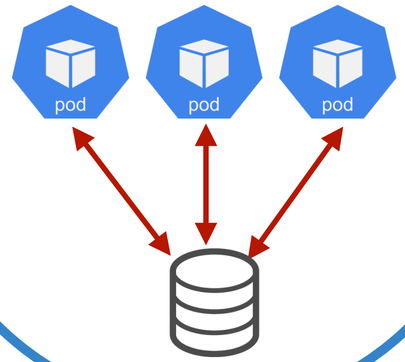
Z Lab

5つのポイント

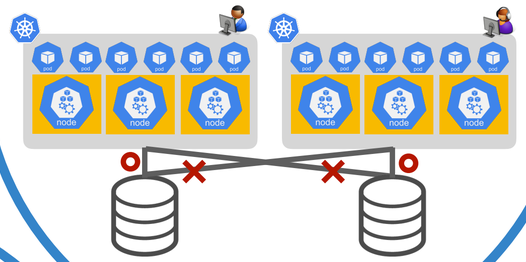
1. Rolling Update



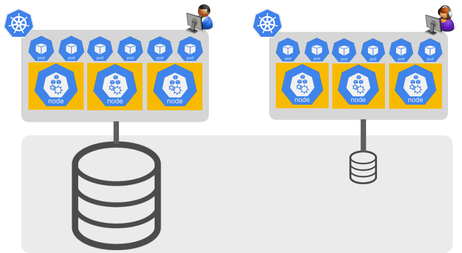
2. Locking



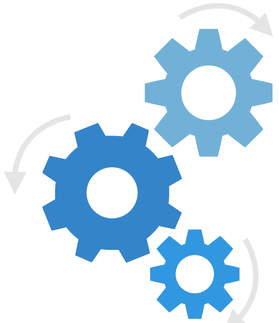
3.Security



4.Quota

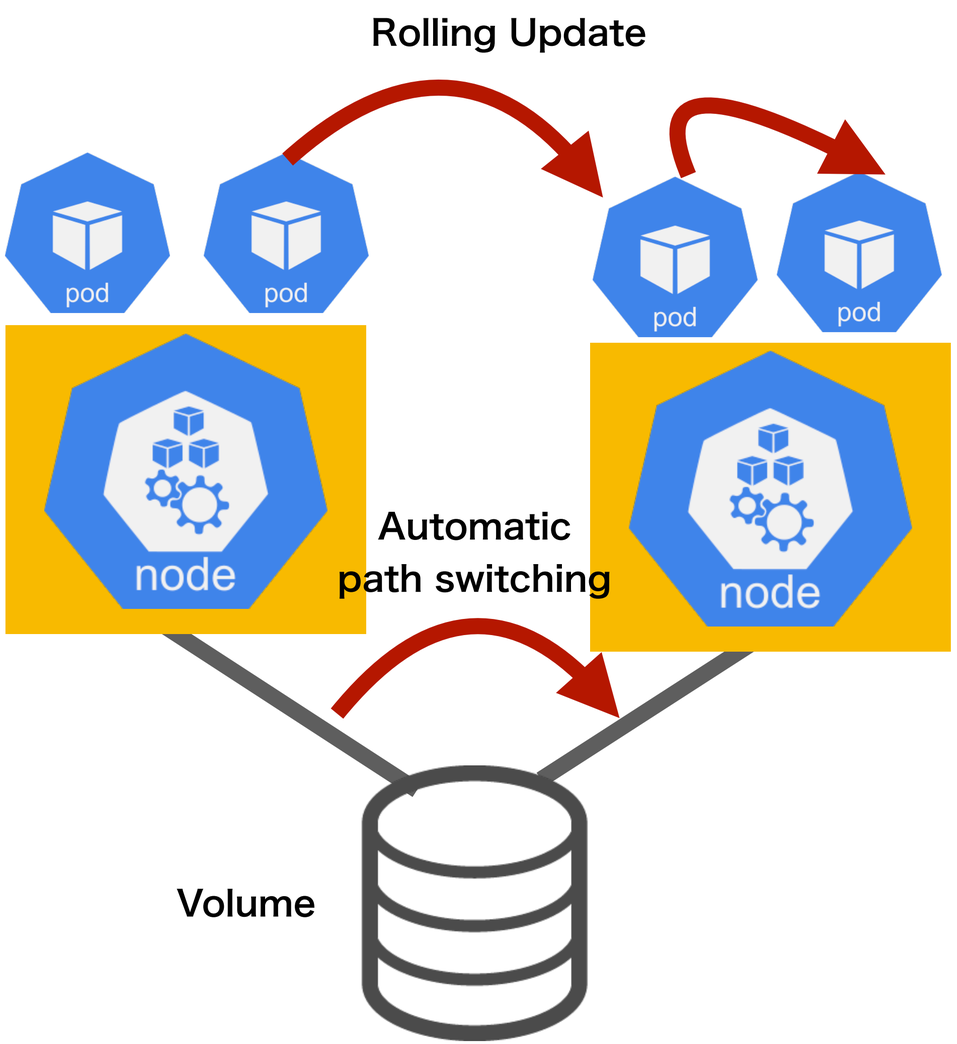


5.Value



Z Lab

1. Rolling Update

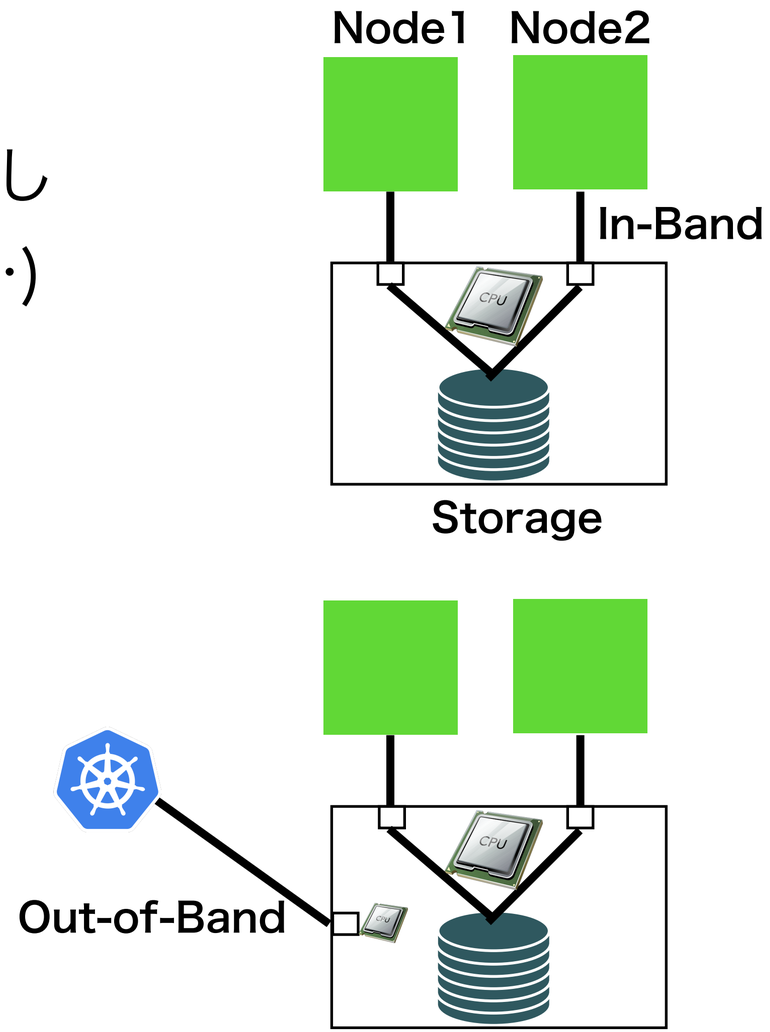


- Podが再作成する契機
  - Rolling update
  - 障害 etc.
- 再作成する場所
  - 同一Node
  - 別Node
- Detach/Attachの自動実行

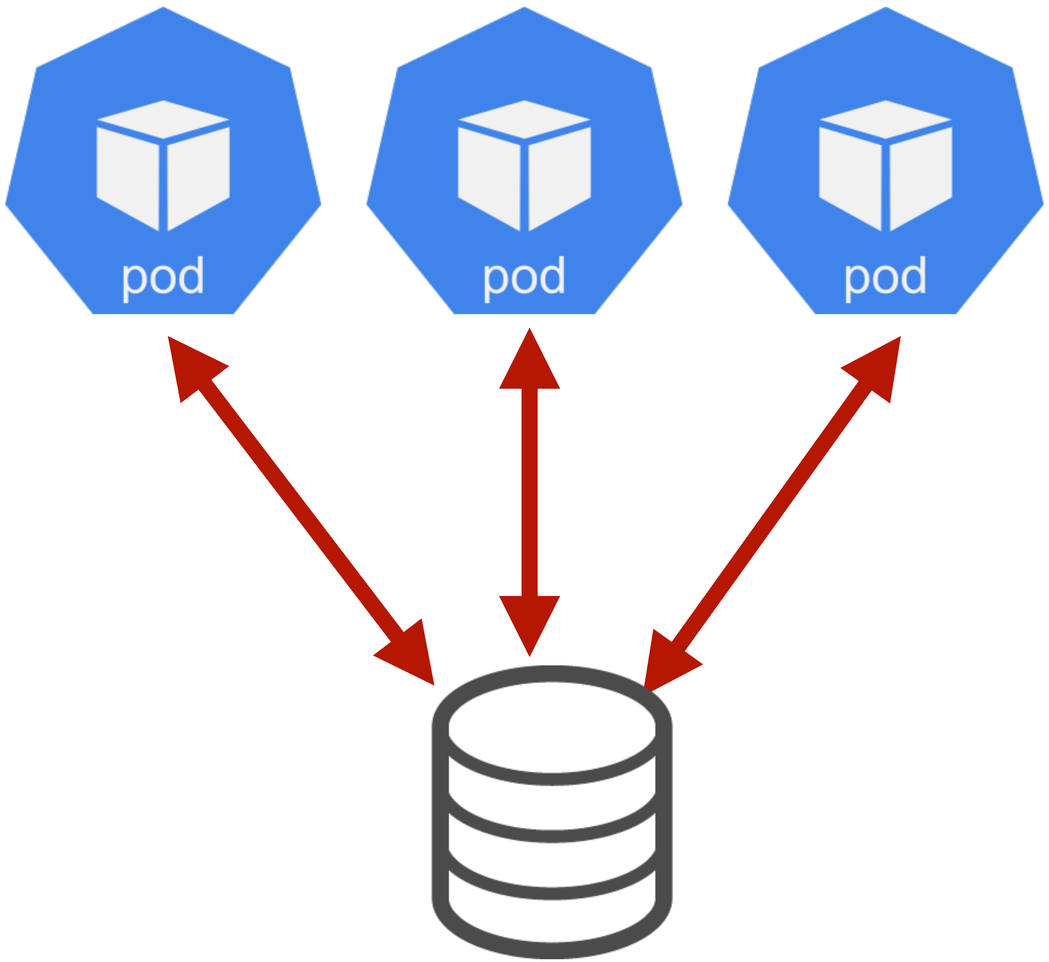
Z Lab

設計ポイント1

- Detach/Attach オペレーションの性能
  - 一度に多量のDetach/Attachオペレーション呼び出し
    (例えば 110 pods/nodeが1Volume持っていると…)
  - 従来高速であったIn-Bandオペレーションの
    ベンダ独自ビットによるパス切り替えは
    Kubernetesでは利用不可(e.g. SCSI VPD83)
- マルチパス設定
  - Non-disruptiveな切り替えでは必要
  - Node追加時に管理者の操作なしでパス設定
    ができればベター



2. Locking



- サポートしているAccess Modes
  - ReadWriteOnce
  - ReadOnlyMany
  - ReadWriteMany
- BlockストレージはLockを持たない
- 同時に1つのVolumeに書き込みを行うとデータ破損の危険性あり

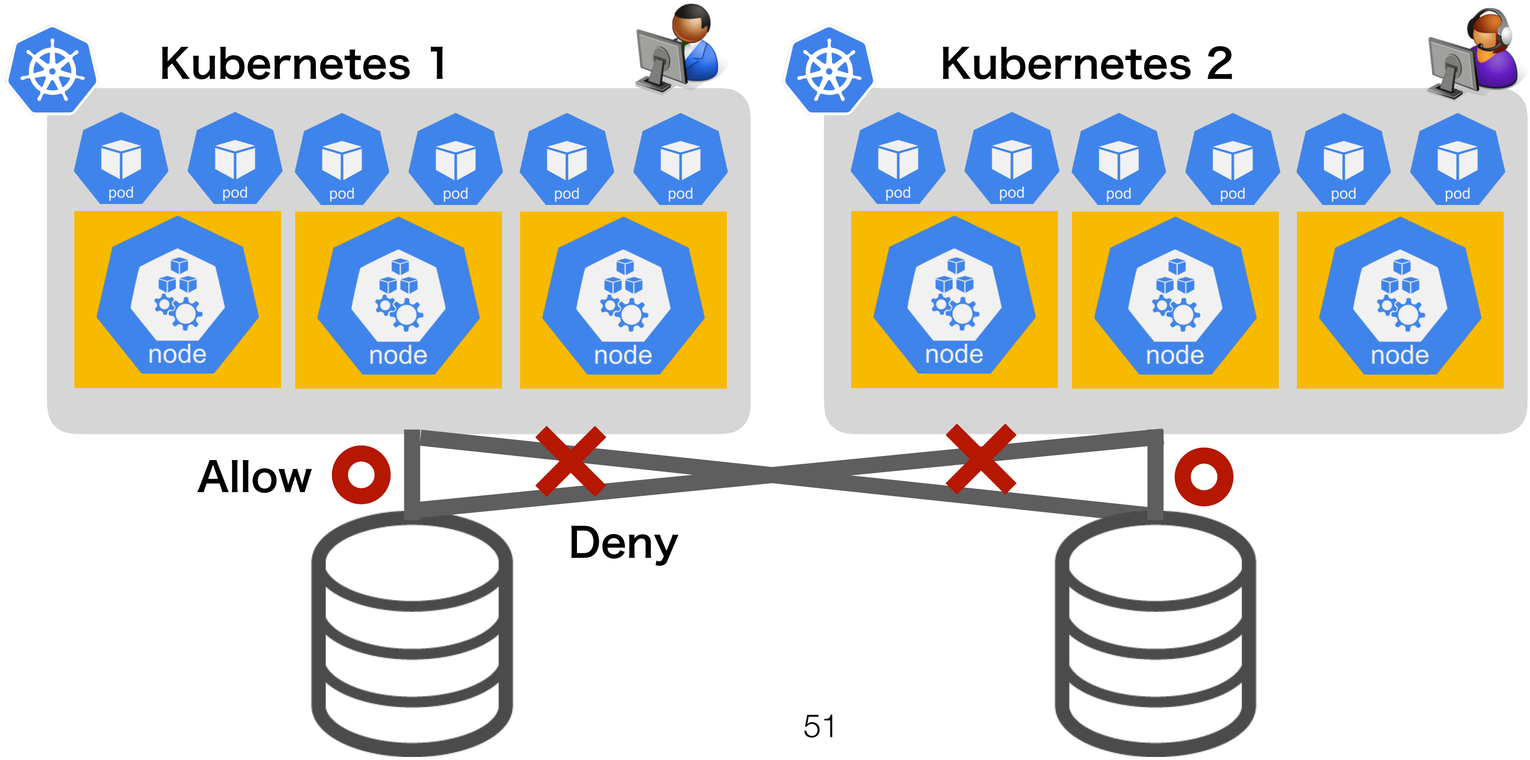
Z Lab

設計ポイント2

- サポートするAccess Modesをチェック
  - Lock制御しないアプリケーションではReadWriteManyを避けるべき
- ミドルウェアのアプリケーション(e.g. RDB)か
  フロントエンドのアプリケーション(e.g. Web front)のどちらでLock制御されているかをチェック
  - フロントエンドのアプリケーションでLock制御するサービスの場合はReadWriteManyは避けるべき

3. Security

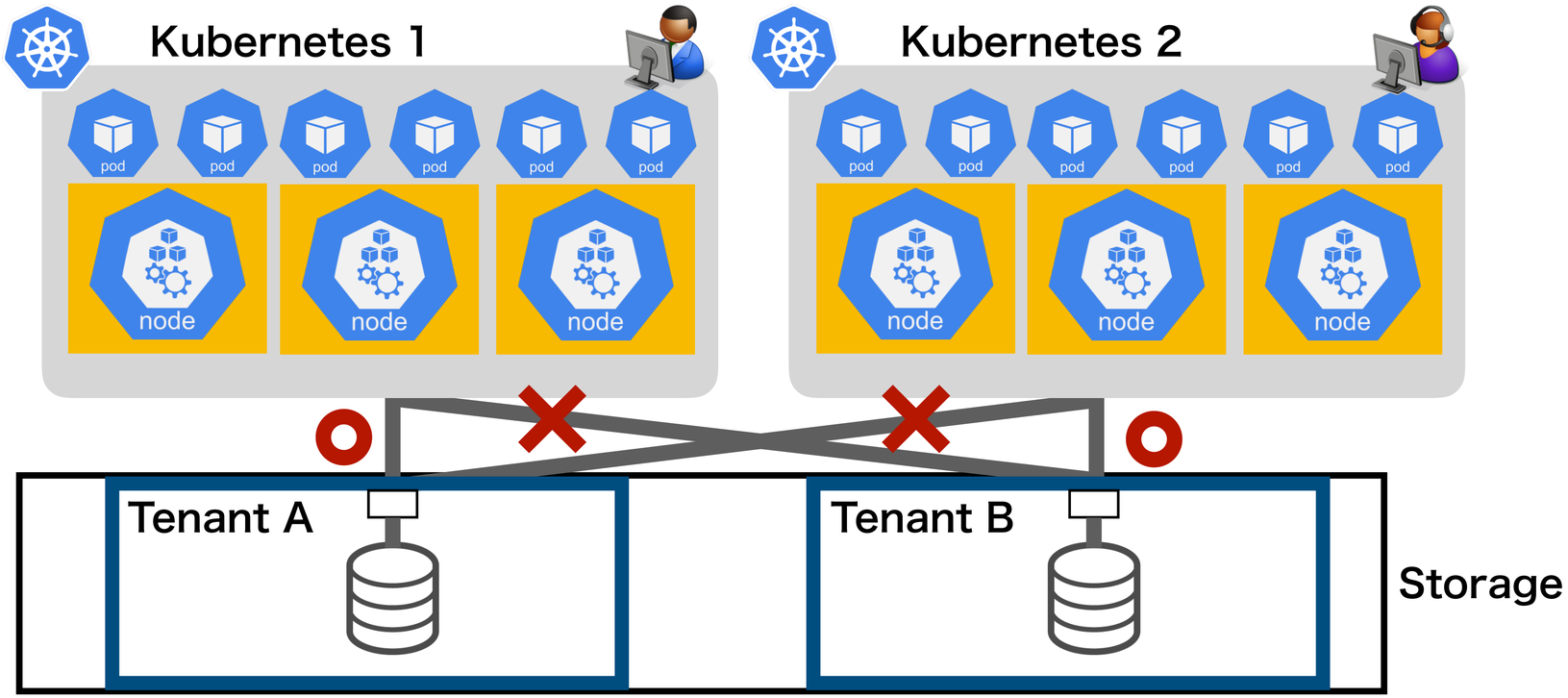
- 不必要なユーザからのアクセスは禁止
- 複数Kubernetesがある環境ではKubernetes自身での
  アクセスコントロールは不可



Z Lab

設計ポイント3

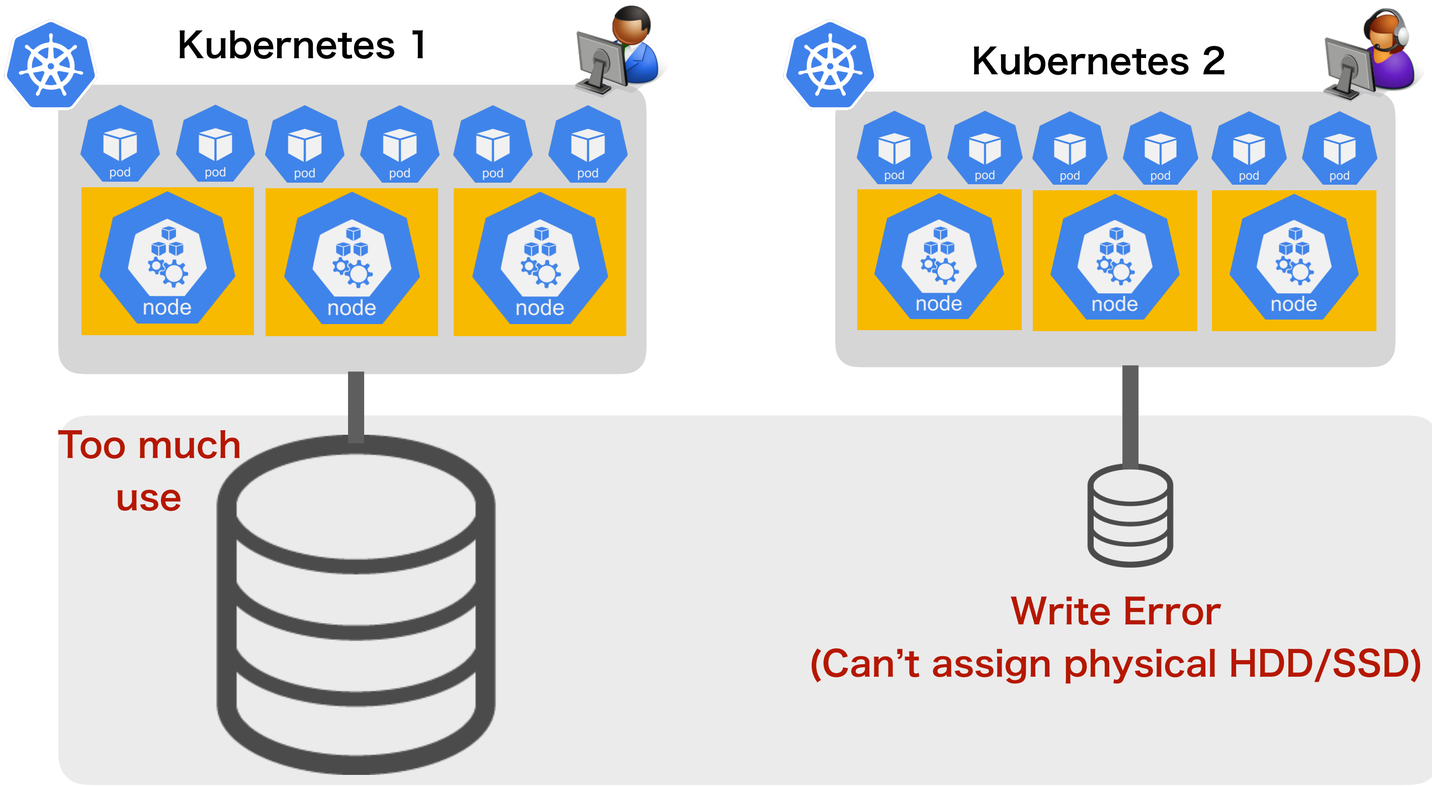
- ストレージのマルチテナントでKubernetes毎のアクセスをコントロール
  - コントロールできるリソースを確認 (e.g. volume, port, node's IP/IQN)
- テナントのACLにNodeのIP(or IQN)が自動登録できればベター



Z Lab

4. Quota

- 特定ユーザによるリソースの使いすぎを禁止
- Thin Provisioning Volumeのように物理SSD/HDDを共有する場合は特に要注意



Z Lab

設計ポイント4

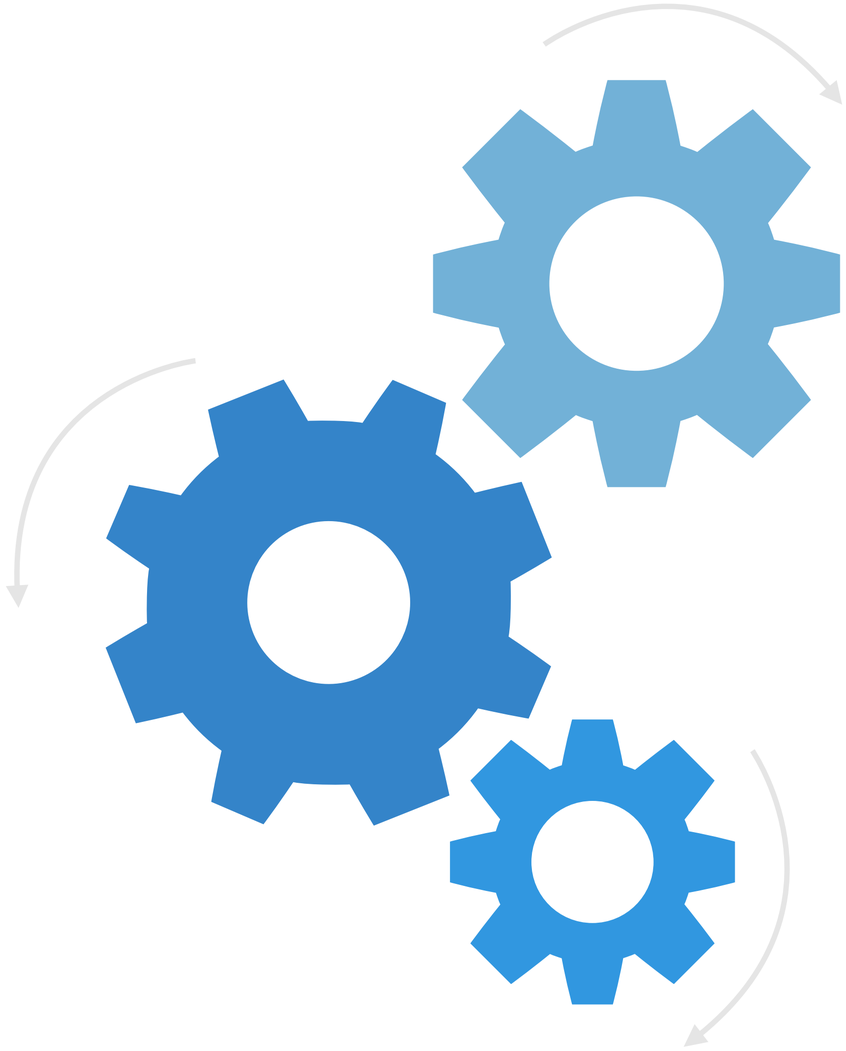
- Quotaを使い分けて設計
- KubernetesのQuota
  - Namespace毎での制限
- ストレージのQuota
  - Kubernetes毎での制限
  - VolumeやPoolのサイズ制限
  - Volumeの生成数の制限

KubernetesのStorage Resource Quota

パラメータ	説明
requests.storage	Namespace内の全てPVCのサイズの合計
persistentvolumeclaims	Namespace内のPVCの数
<storage-class-name>.storageclass.storage.k8s.io/requests.storage	特定のSCから割り当てられたNamespace内の全てのPVCのサイズの合計
<storage-class-name>.storageclass.storage.k8s.io/persistentvolumeclaims	特定のSCから割り当てられたNamespace内のPVCの数

Z Lab

5. Value



- ストレージには様々なタイプが存在
  - アプライアンス製品 (ASIC利用)
  - SDS(Software Defined Storage)
  - CloudNative Storage
- 一長一短
- 全データを特定のストレージのみで
  サポートすると可用性低下や
  コスト増大のリスクあり

Z Lab

設計ポイント5

- 保存するデータの価値の見定め
- データの価値に応じ、複数のStorageClassを使い分けがオススメ
- ストレージのコストは保険と同じようなもの(？)

```
$ kubectl get sc
NAME   PROVISIONER       AGE
Gold   A/provisioner     26d
Silver SDS/provisioner   26d
Bronze CNS/provisioner   26d
```

比較項目	アプライアンス製品	SDS	CloudNative Storage
機器コスト	$$$$$	$$$	$$$
セットアップの容易さ			✓
公開情報の多さ		✓	
スケールアウトのし易さ			✓
障害の発生しにくさ	✓		
データ復旧の容易さ	✓		
ストレージ新技術の採用の早さ	✓		
バージョンアップの頻度		✓	✓

※発表者の主観による評価です

Z Lab

まとめ

- ゼットラボでの事例、Kubernetesでのストレージを紹介
- Kubernetes向けストレージ選び5つのポイント
  1. Rolling Update
  2. Locking
  3. Security
  4. Quota
  5. Value
- VMが登場した時もステートフルアプリのVMへの移行は慎重だった
- Containerでの動作事例も2018年より徐々に増えてきている印象

Z Lab

We are Hiring!

Kubernetes, Docker, Prometheus, Golang, etc.

ご興味のある方は、ゼットラボ社員に直接ご連絡ください。

Z Lab