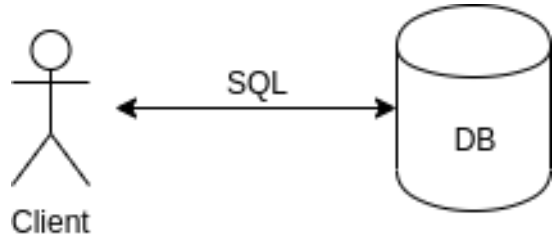
Как рулить пароходом MySQL

Николай Ихалайнен

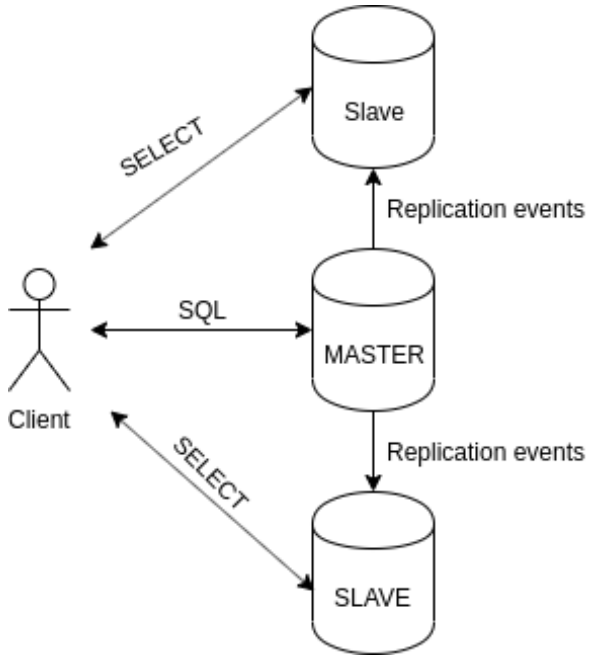
Репликация лучше одного сервера

Один сервер

- Меньше составных частей меньше падений
- Новые транзакции читают закоммиченные данные без задержек
- Приложение знает где искать базу



Репликация лучше одного сервера



Репликация

- Сервис жив и обслуживает запросы после падения одного или нескольких узлов
- Больше серверов, больше производительность чтения
- Прокси-сервера гибко распределяют нагрузку

Репликация лучше одного сервера

Один сервер

- Меньше составных частей меньше падений
- Новые транзакции читают закоммиченные данные без задержек
- Приложение знает где искать базу

Репликация

- Сервис жив и обслуживает запросы после падения одного или нескольких узлов
- Больше серверов, больше производительность чтения
- Прокси-сервера гибко распределяют нагрузку

Репликация нужна облакам

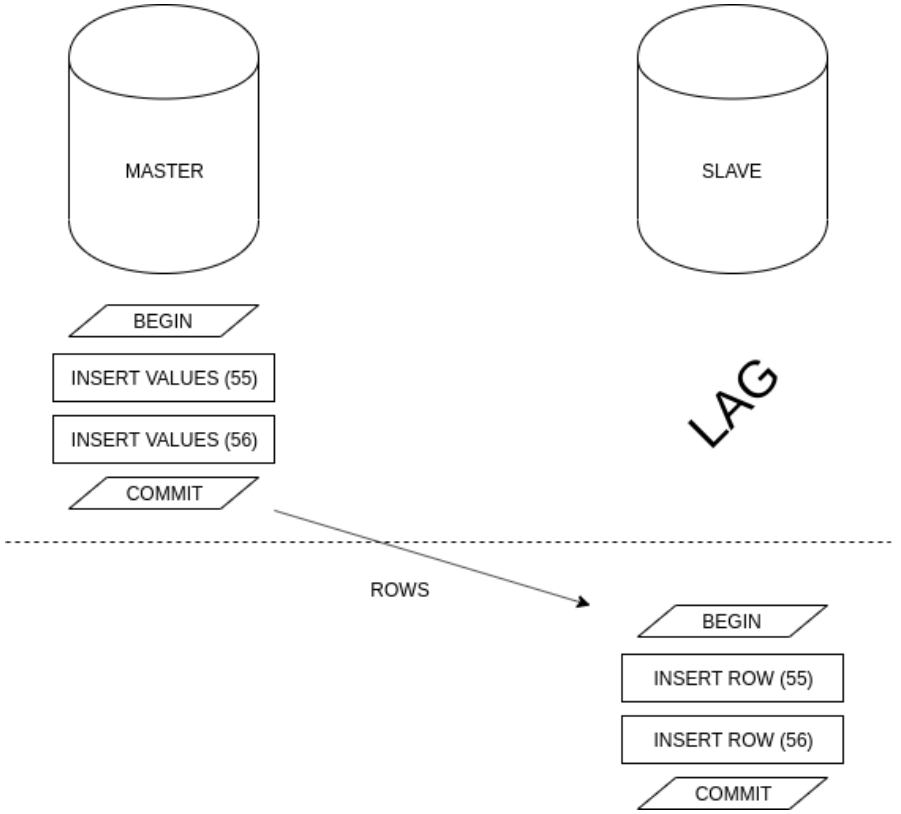
- Новый узел с такими же данным – легче чем дорогой инстанс
- Узел умер? Данные всё ещё доступны. Нет простоя
- Плавное обновление: новая версия MySQL на одной реплике

Kubernetes + MySQL <> vendor lock

- Свободная миграция между облачными провайдерами
- Отказоустойчивость при потере узлов
- Простые бекапы и востановление
- Масштабируемость на чтение (SELECT)
- Изоляция OLTP запросов от сложной аналитики

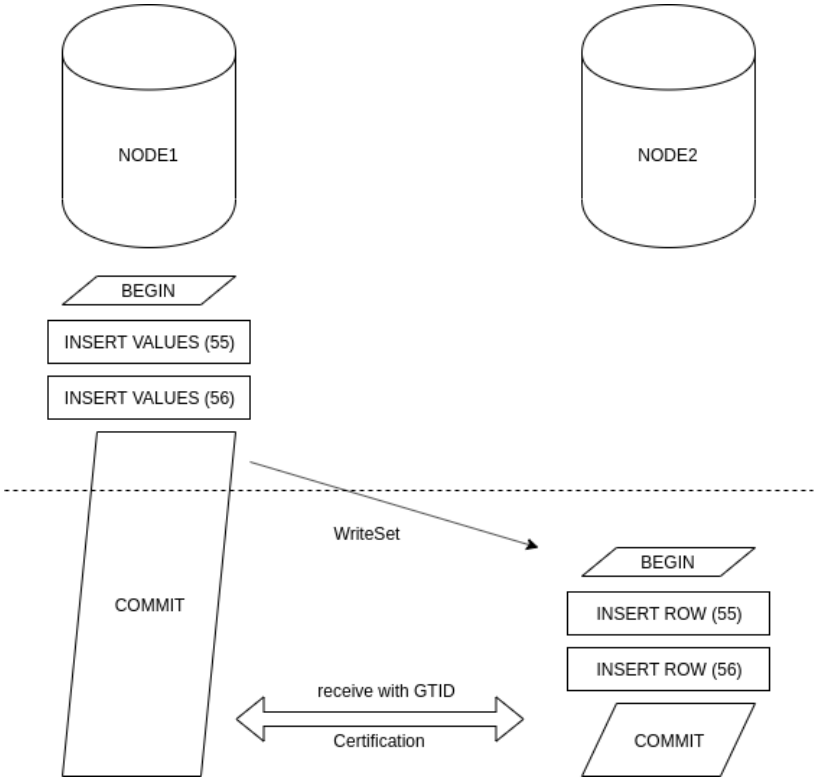
Много непараллельной записи – лучше ассинхронная репликация

- Скорость запись как на одиночном сервере
- Можно чинить реплики вручную (pt-table-checksum/sync)
- Читаем с реплик снимки прошлого
- Легко разваливается и отстаёт



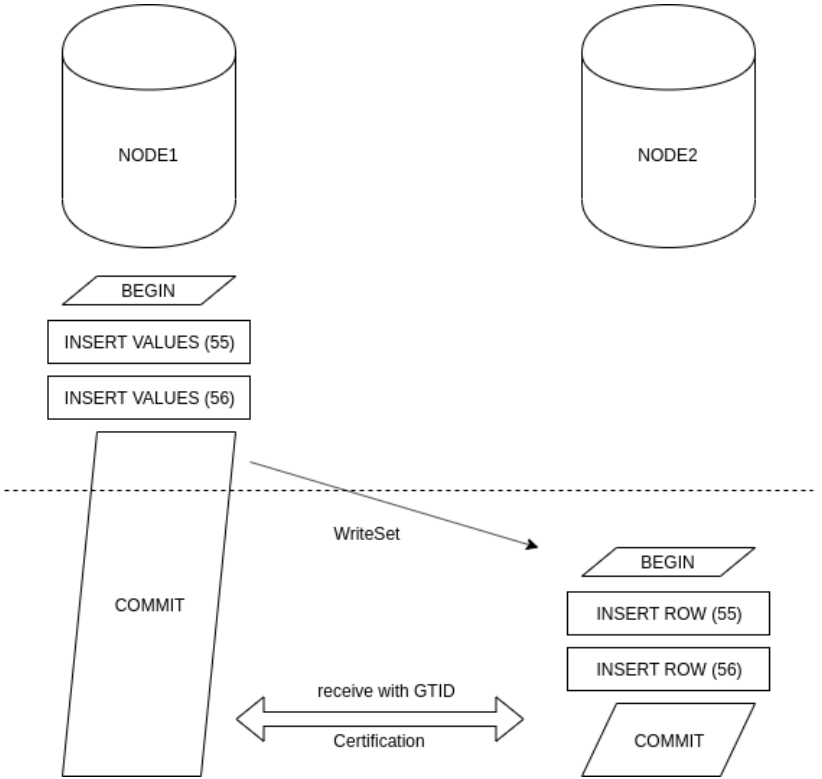
Синхронная репликация - лучшая автономность

- Новый узел CAM создаётся из бекапа
- CAM востанавливается после сбоя
- Сертификация коммита
- ROW-based



Синхронная репликация - лучшая автономность

- не “пропускайте” ошибки
- на каждом узле:
  идентичные снимки данных



Синхронная репликация – лучшая консистентность

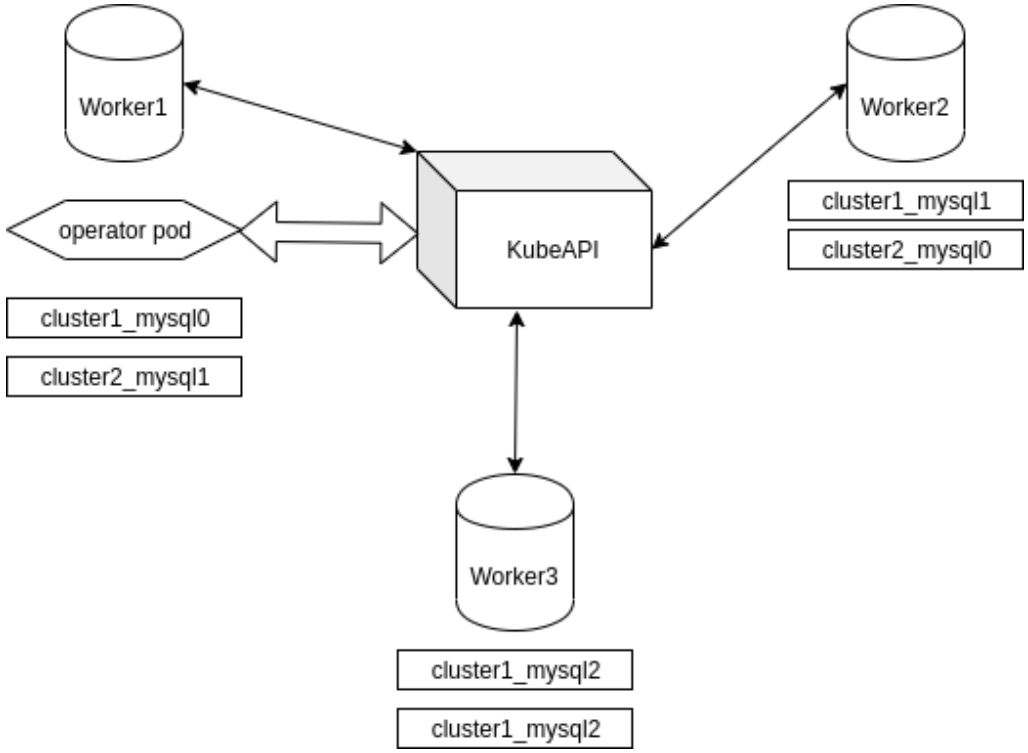
- После нескольких RTT закоммиченное на всех узлах
- Узел потерял кворум – ошибка вместо результата запроса
- 3 и больше узлов

MySQL в K8S это...

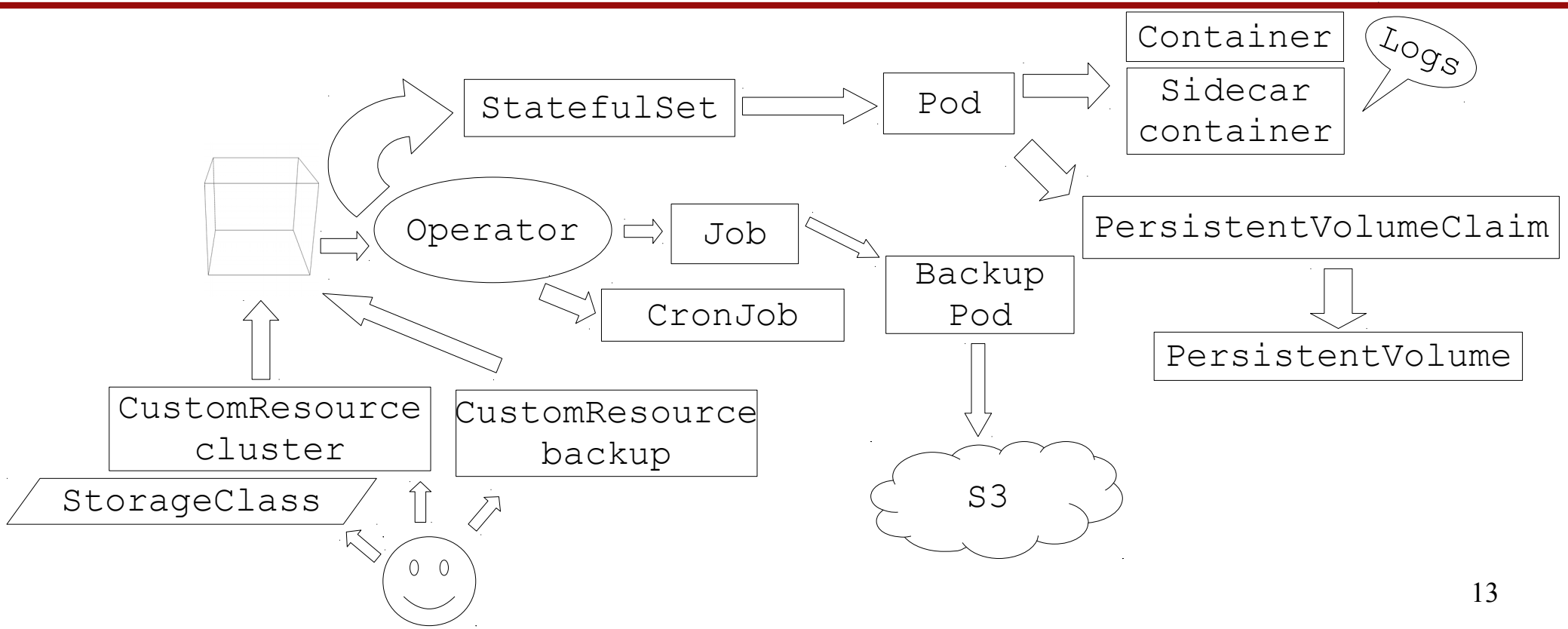
- Быстро инсталлировать
- Подключиться из приложения
- Подкрутить настройки через yaml
- Расширить/сжать
- Востановить после сбоя узла
- Резервные копии
- Мониторинг

Слой управления: операторы

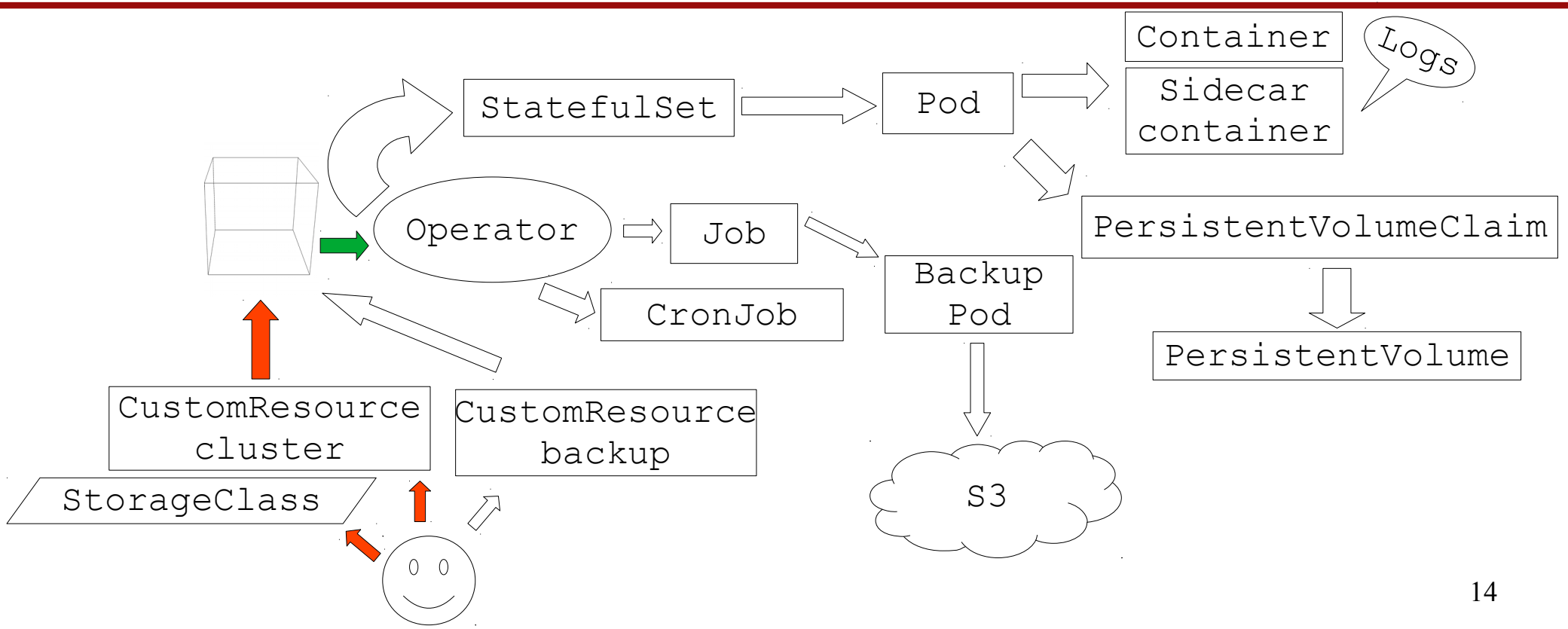
- Расширения на уровне кластера K8S
- Автоматизируют рутинные задачи
- Глобальны для всех кластеров MySQL в K8S



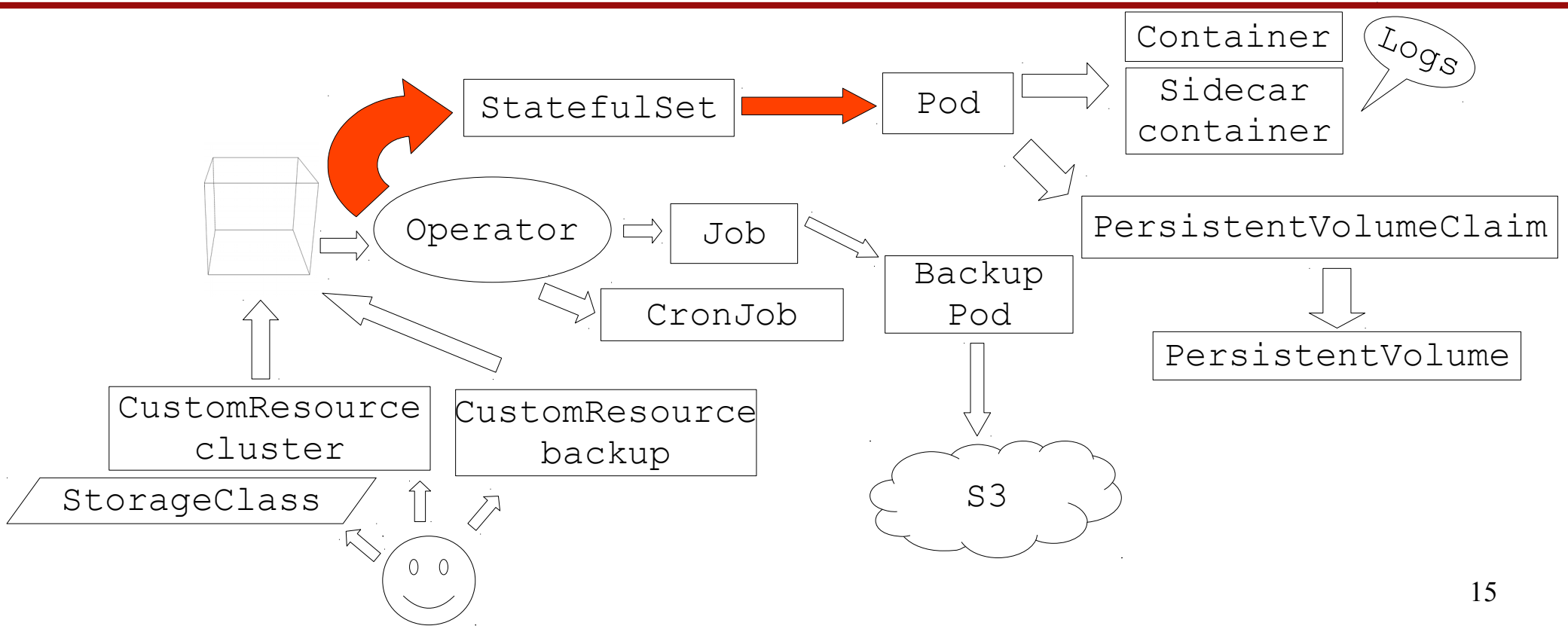
Операторы: объекты



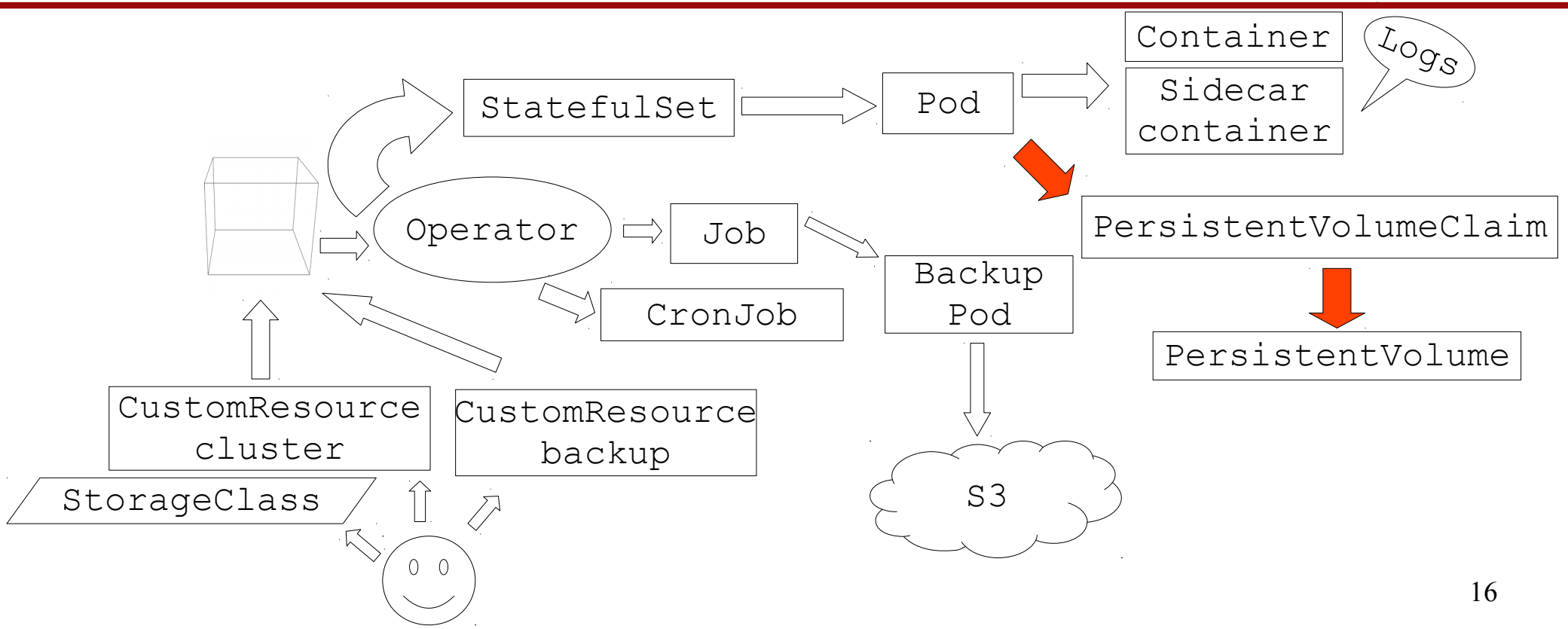
Операторы: декларация кластера



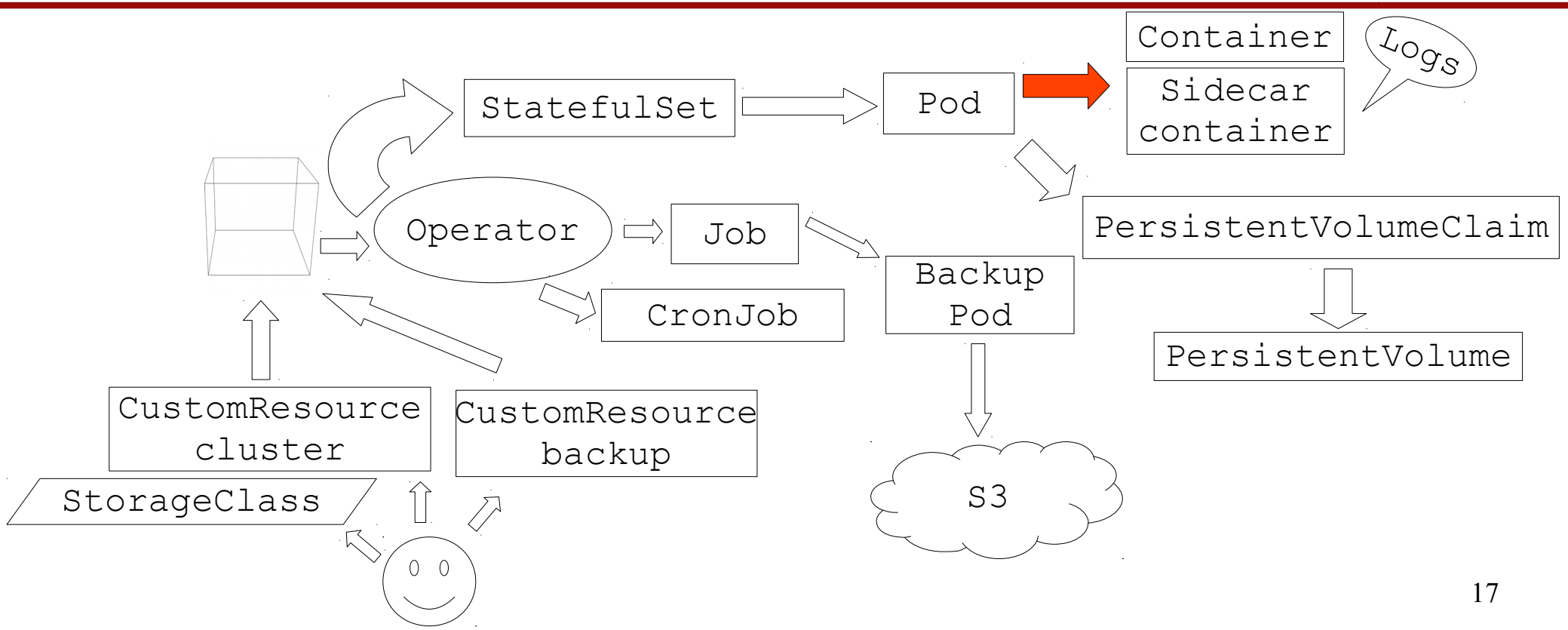
Операторы: запуск Pod



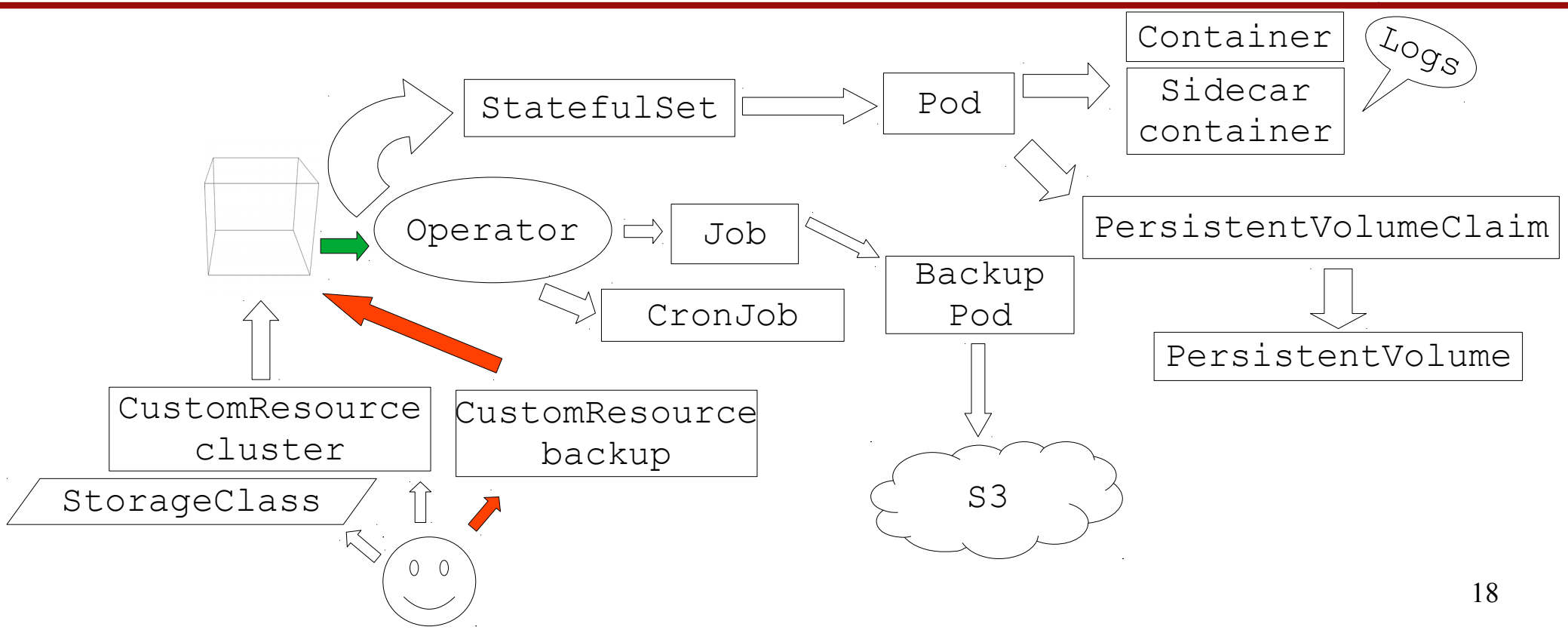
Операторы: подключение дисков



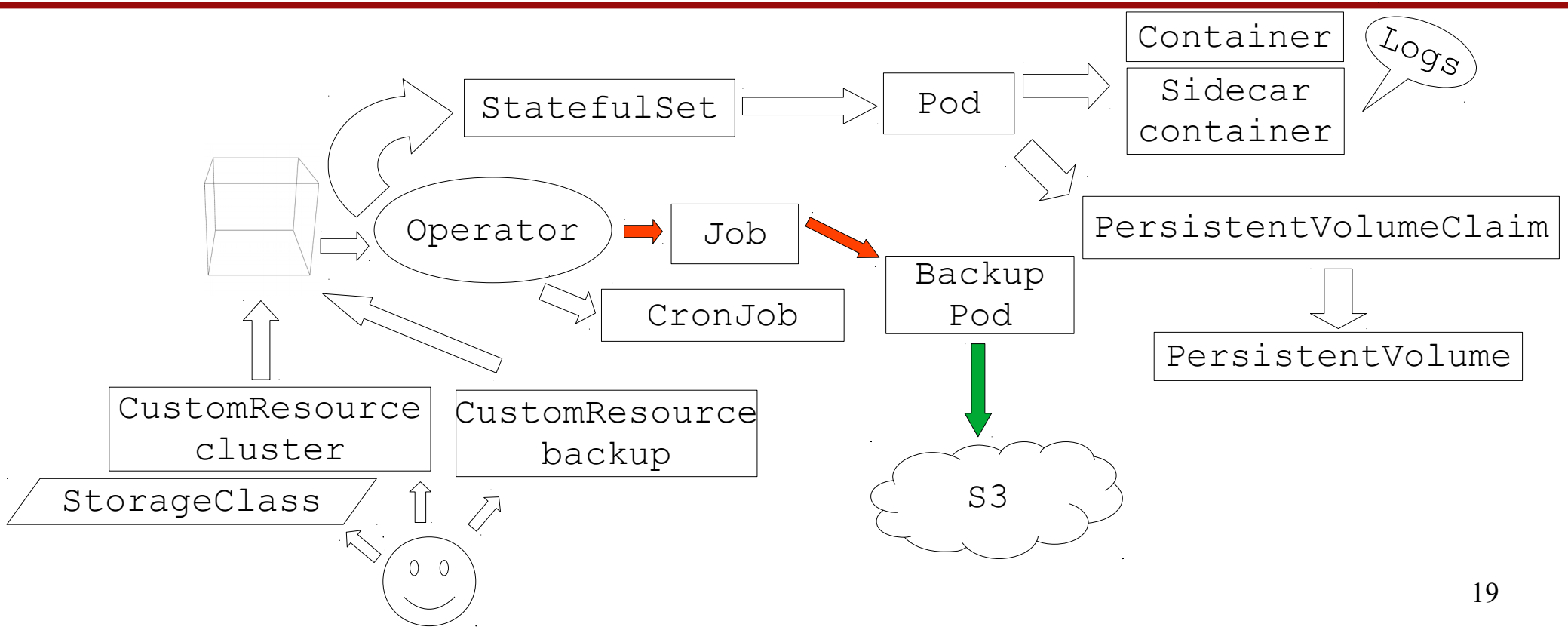
Операторы: запуск контейнеров



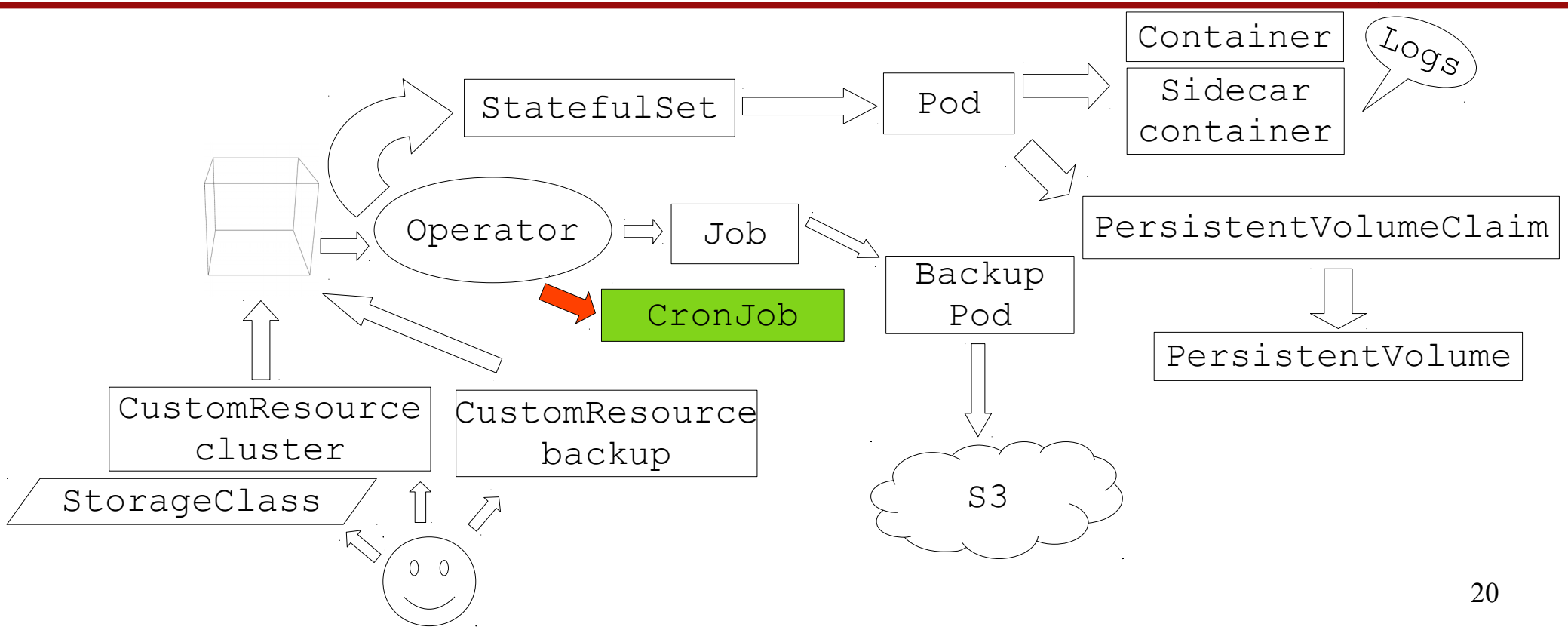
Операторы: декларация бекапа



Операторы: одноразовые задачи



Операторы: периодические бекапы



Отладка

- kubectl get pods
- kubectl get -o yaml pod cluster1-mysql
- kubectl describe pod cluster1-mysql
- kubectl logs backup-abc85683

Доступные Open Source операторы

Асинхронная

- Presslabs operator для MySQL
  github.com/presslabs/mysql-operator

Синхронная

- K8S operator для Percona Xtradb Cluster
  github.com/percona/percona-xtradb-cluster-operator
- Oracle MySQL Operator: InnoDB Cluster
  github.com/oracle/mysql-operator

Presslabs Operator for MySQL

```
$ helm repo add presslabs https://presslabs.github.io/charts
$ helm install presslabs/mysql-operator --name mysql-operator
```

- Позволяет настроить репликацию типа Master-Slave, которая управляется Orchestrator
  - Утилита для автоматической настройки репликации и переключения роли мастера на слейв
- Создаёт/уничтожает слейвы по вашему запросу

Запуск Master-Slave кластера

```
$ cat <<EOF | kubectl apply -f-
apiVersion: mysql.presslabs.org/v1alpha1
kind: MysqlCluster
metadata:
  name: my-cluster
spec:
  replicas: 1
  secretName: my-cluster-secret
---
apiVersion: v1
kind: Secret
metadata:
  name: my-cluster-secret
type: Opaque
```

А где репликация?

```
$ kubectl get pods
NAME                READY   STATUS    RESTARTS   AGE
my-cluster-mysql-0   4/4     Running   0          24m
mysql-operator-0     2/2     Running   0          29m

$ kubectl get mysqlclusters.mysql.presslabs.org
NAME         READY   REPLICAS   AGE
my-cluster   True    1          22m

$ kubectl get mysql/my-cluster -o yaml | \
  sed -e 's/replicas: 1/replicas: 3/' | kubectl apply -f -
```


- Replicas = 1 => только один сервер, мастер

Репликация?

```
$ kubectl get pods
NAME                READY   STATUS    RESTARTS   AGE
my-cluster-mysql-0  4/4     Running   0          29m
my-cluster-mysql-1  4/4     Running   0          2m54s
my-cluster-mysql-2  4/4     Running   0          104s
mysql-operator-0    2/2     Running   0          34m
```

- Кто есть кто:
  - Где мастер?
  - Где слейвы?

Orchestrator

```
$ kubectl port-forward service/mysql-operator 8080:80
$ wget https://raw.githubusercontent.com/github/orchestrator\
/master/resources/bin/orchestrator-client
$ export ORCHESTRATOR_API=http://127.0.0.1:8080/api

$ orchestrator-client -c clusters
my-cluster-mysql-0.mysql.default:3306
$ orchestrator-client -c topology \
    -i my-cluster-mysql-0.mysql.default:3306
my-cluster-mysql-0.mysql.default:3306   [0s,ok,5.7.26-29-log,rw,ROW,>>,GTID]
+ my-cluster-mysql-1.mysql.default:3306 [0s,ok,5.7.26-29-log,ro,ROW,>>,GTID]
+ my-cluster-mysql-2.mysql.default:3306 [0s,ok,5.7.26-29-log,ro,ROW,>>,GTID]
```


- Один демон на весь K8S кластер

Подключение к mysql

```
$ kubectl run -i --rm --tty  --image=percona:5.7 \
  --restart=Never percona-client   --   bash -il
$ mysql --host my-cluster-mysql-1.mysql.default \
    --user=root –password=not-so-secure
mysql> select @@read_only;
| @@read_only |
|           1 |
```

- Слейвы только для чтения
- CLUSTER_NAME-mysql-N.mysql.NAMESPACE
- CLUSTER_NAME-mysql-master
- CLUSTER_NAME-mysql : случайный SLAVE

Переключение роли мастера

```
$ orchestrator-client -c graceful-master-takeover \
   -i my-cluster-mysql-0.mysql.default:3306 \
   -d my-cluster-mysql-1.mysql.default:3306
$ orchestrator-client -c topology \
    -i my-cluster-mysql-0.mysql.default:3306
my-cluster-mysql-1.mysql.default:3306   [0s,ok,5.7.26-29-log,rw,ROW,>>,GTID]
- my-cluster-mysql-0.mysql.default:3306 [null,nonreplicating,5.7.26-29-
log,ro,ROW,>>,GTID,downtimed]
+ my-cluster-mysql-2.mysql.default:3306 [0s,ok,5.7.26-29-log,ro,ROW,>>,GTID]
```

- read_only становится 0
- START SLAVE на старом мастере чинит его.
- не отслеживается оператором

Запуск Master-Slave кластера

```
$ export
GH=https://raw.githubusercontent.com/presslabs/mysql-operator/master/
examples

$ kubectl apply -f $GH/example-cluster-secret.yaml

$ kubectl apply -f $GH/example-cluster.yaml
```

- Создать тестоый кластер на два узла (master+slave) просто
- Не подходит для боевого использования

Из коробки

- podSpec ограничения процессора и памяти небольшие
- Настройки MySQL по-умолчанию: очень медленно
- Бекапы не настроены
- “Какие-то” диски выбраны
  - SSD/Flash очень полезны для современных баз данных

Ограничение на CPU & MEM

```
$ kubectl describe nodes
Allocatable:
 attachable-volumes-gce-pd:  127
 cpu:                        3920m
 ephemeral-storage:          47093746742
 hugepages-2Mi:              0
 memory:                     12699300Ki
 pods:                       110
```

- Надо иметь представление о своих узлах

CPU & MEM в example-cluster.yaml

```
podSpec:
  resources:
    requests:
      memory: 8G
      cpu: 3000m
```

- Каждый запрос, который работает 1 секунду потребляет 1000m
- Идеально, если база помещается в буфера MySQL целиком
  - Миллионы строк - сотни мегабайт -> гигабайты

Настройка MySQL в example-cluster.yaml

```
MysqlConf:
  innodb-pool-buffer-size: 7G
  innodb_log_file_size: 4G
  innodb_flush_log_at_trx_commit: 2
```

- Опции будут добавлены в my.cnf
- https://www.percona.com/blog/2016/10/12/mysql-5-7-performance-tuning-immediately-after-installation/

Настройка MySQL в example-cluster.yaml

MysqlConf:
  innodb-pool-buffer-size: 7G
  innodb_log_file_size: 4G
  innodb_flush_log_at_trx_commit: 2
  sync_binlog=0

- Потеря до 1 секунды закоммиченных транзакций
  - При падении всего кластера разом
- sync_binlog=N : fsync каждые N транзакций

Типы “дисков”

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ssd
provisioner: kubernetes.io/gce-pd
parameters:
  type: pd-ssd
```

- Каждый облачный провайдер предоставляет свой “provisioner” чтобы выбрать SSD или задать лимит на IOPS

Типы “дисков” в example-cluster.yaml

```
VolumeSpec:
  persistentVolumeClaim:
    accessModes: [ "ReadWriteOnce" ]
    resources:
      requests:
        storage: 16Gi
    storageClassName: ssd
```

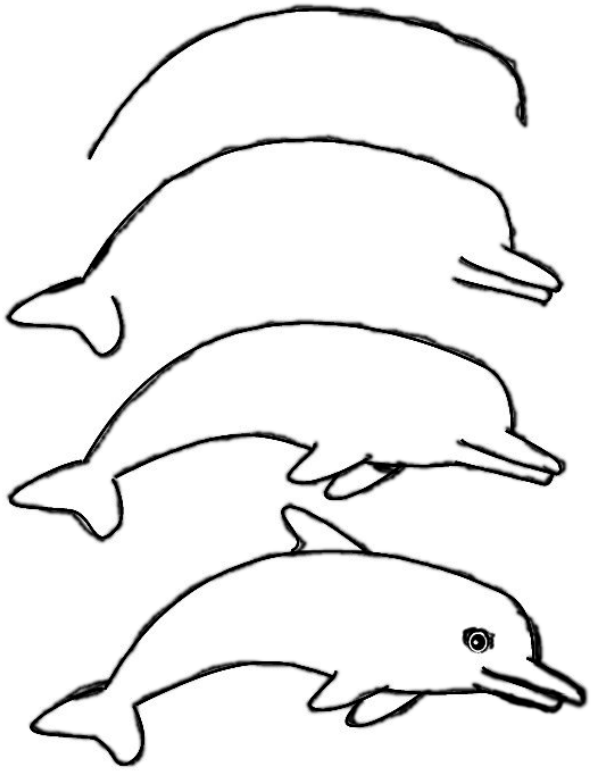
- Включаем SSD
- Меняем размер тома

Резервные копии

- Надо создать “secret” для доступа в S3 или GCS
  - minio - локально
- Создать ресурс MysqlBackup для однократного бекапа
- Поменять example-cluster.yaml для периодических бекапов
- Бекапы будут сделаны с помощью xtrabackup
- Presslabs не использует K8S CronJob
  - создаёт и удаляет задания самостоятельно

Пуленепробиваемый MySQL

- Presslabs оператор использует старые проверенные технологии
- Бекапы с минимальным простоем (Backup Locks)
- Лёгкая миграция с одиночного MySQL сервера
- Отличная автоматизация репликации MySQL



Доступные Open Source операторы

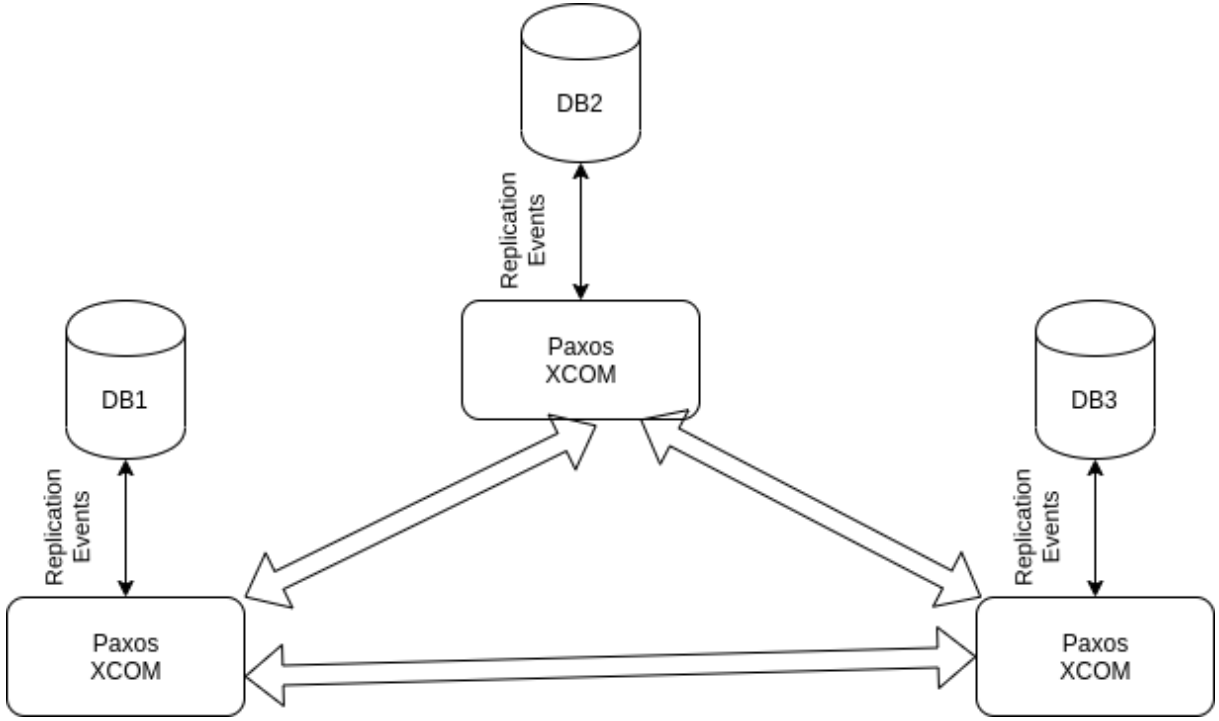
Асинхронная

- Presslabs operator для MySQL
  github.com/presslabs/mysql-operator

Синхронная

- K8S operator для Percona Xtradb Cluster
  github.com/percona/percona-xtradb-cluster-operator
- **Oracle MySQL Operator: InnoDB Cluster**
  github.com/oracle/mysql-operator

InnoDB Cluster: синхронная репликация



Что такое InnoDB Cluster?

- Синхронная репликация
- Плагин MySQL Group Replication
- Частично базируется на “старых” технологиях репликации
- WRITESET

Что такое InnoDB Cluster?

- WRITESET
  - Транзакции меняют строки
  - Для каждой строки считаем XXHASH64 от Primary Key
  - ПАРАЛЛЕЛЬНО: транзакции меняющие разные строки

Что такое InnoDB Cluster?

- WRITESET
  - Конфликтующие транзакции
- Синхронный Multi-Master
  - Для одной из конфликтующих в кластере транзакций делаем ROLLBACK

Что такое InnoDB Cluster?

- Накладно: подтверждать каждую транзакцию
- XCOM: eXtended communications
  - Реализация Paxos: консенсус
  - Сетевое взаимодействие
- Сертификация транзакций

InnoDB Cluster – хорошая отказоустойчивость

- Автоматическое переключение на живой узел
- Включая сбои кода, железа, реконнект соединений между узлами
- Высокодоступная база данных (архитектурно)

Что такое InnoDB Cluster?

- Плагин MySQL Group Replication
- Single/Multi-primary (Изменения базы из любого узла)
- Автоматическая распределённая “регенерация” (recovery)
- Обнаружение конфликтов (WRITESET)
- Отслеживание членства в группе

InnoDB Cluster: инсталляция

```
$ git clone git@github.com:oracle/mysql-operator.git
$ cd mysql-operator
$ helm repo update
$ kubectl create ns mysql-operator
$ helm install --name mysql-operator mysql-operator
$ kubectl get --namespace mysql-operator all
```

- Используется Helm
- Сложные настройки безопасности
- Такое же короткое имя ресурса как и для оператора Presslabs

Запуск кластера

```
$ cat <<EOF | kubectl apply -f -
apiVersion: mysql.oracle.com/v1alpha1
kind: Cluster
metadata:
  name: my-app-db
```

- В namespace должна быть сервисная учётная запись mysql-agent
- Нет документации на все поля в yaml ресурса

Приложение находит базу по DNS

```
$ kubectl get pods
NAME         READY   STATUS    RESTARTS   AGE
my-app-db-0  2/2     Running   0          5m40s
my-app-db-1  2/2     Running   0          4m53s
my-app-db-2  2/2     Running   0          3m52s
$ PASS=$(kubectl get secret my-app-db-root-password \
  -o jsonpath="{.data.password}" | base64 --decode)
$ kubectl run mysql-client --image=mysql:5.7 -it --rm --restart=Never \
  -- mysql -h my-app-db -uroot -p$PASS -e 'SELECT 1'
mysql: [Warning] Using a password on the command line interface can be insecure.
| 1 |
| 1 |
```

- Балансировка соединений через K8S, можно установить MySQL Router

Настройка MySQL

```
kind: ConfigMap
apiVersion: v1
metadata:
  name: mysql-config
data:
  my.cnf: |-
    [mysqld]
    innodb_buffer_pool_size=256M
apiVersion: mysql.oracle.com/v1alpha1
kind: Cluster
metadata:
  name: my-app-db
spec:
  config:
    name: mysql-config
  volumeClaimTemplate:
    metadata:
      name: data
```

Лимиты CPU и памяти в K8S

- Есть поле Cluster.spec.resources.server.requests.memory в yaml
- Но оно игнорируется оператором.
- Исправлено в master, ждём релиза.

Резервное копирование

```
apiVersion: mysql.oracle.com/v1alpha1
kind: Backup
metadata:
  name: mysql-backup
spec:
  executor:
    mysqldump:
      databases:
        - name: test
  storageProvider:
    s3:
      endpoint: ocitenancy.compat.objectstorage.ociregion.oraclecloud.com
      region:   ociregion
      bucket:   mybucket
      credentialsSecret:
        name: s3-credentials
  cluster:
    name: mysql
```

Регулярные бекапы

```
apiVersion: mysql.oracle.com/v1alpha1
kind: BackupSchedule
metadata:
  name: mysql-backup-schedule
spec:
  schedule: '30 * * * *'
  backupTemplate:
    executor:
      mysqldump:
        databases:
          - name: test
    storageProvider:
      s3:
        endpoint: ocitenancy.compat.objectstorage.ociregion.oraclecloud.com
        region: ociregion
        bucket: mybucket
        credentialsSecret:
          name: s3-credentials
```

MySQL Router: sidecar

```
containers:
- name: mysqlrouter
  image: mysql/mysql-router:8.0.12
  env:
  - name: MYSQL_PASSWORD
    valueFrom:
      secretKeyRef:
        name: wordpress-mysql-root-password
        key: password
  - name: MYSQL_USER
    value: root
  - name: MYSQL_PORT
    value: "3306"
  - name: MYSQL_HOST
    value: mysql-wordpress-0.mysql-wordpress
  - name: MYSQL_INNODB_NUM_MEMBERS
    value: "3"
  command:
```

Oracle MySQL Operator

- Пожалуйста попробуйте его!
- Редко обновляется: Последний коммит 15 мая 2019 [сейчас октябрь 2019]
- Это очень простой и быстрый способ попробовать InnoDB Cluster 8.0 [ещё есть dbdeployer]

Доступные Open Source операторы

Асинхронная

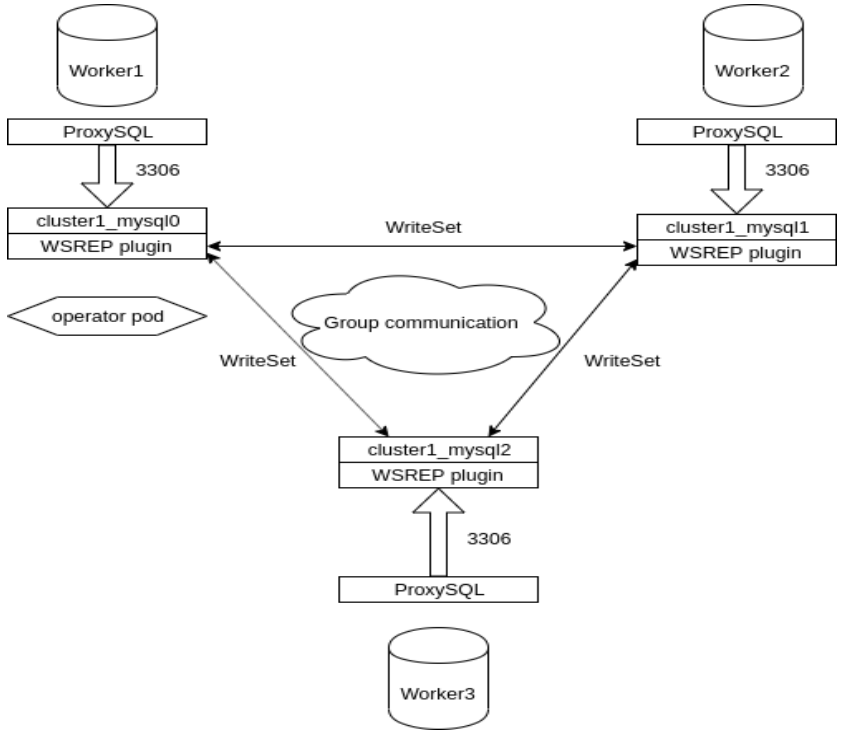
- Presslabs operator для MySQL
  github.com/presslabs/mysql-operator

Синхронная

- **K8S operator для Percona Xtradb Cluster**
  github.com/percona/percona-xtradb-cluster-operator
- Oracle MySQL Operator: InnoDB Cluster
  github.com/oracle/mysql-operator

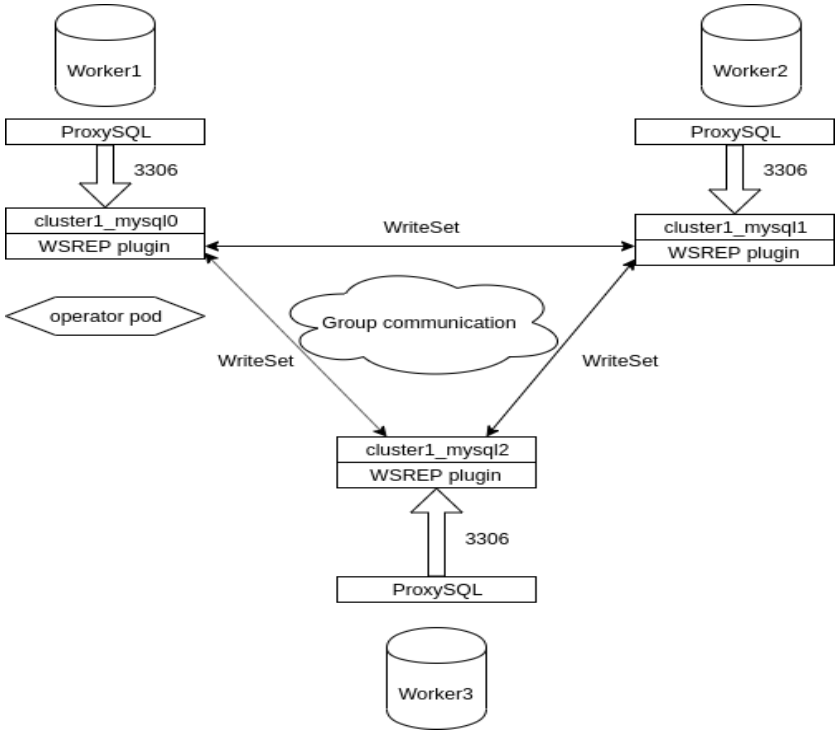
Percona XtraDB Cluster: синхронная

- С мая 2012 (5.5),
  18 релизов 5.7
- Multi-master
- Автоматическое управление членством в группе
- Востановление данных на узле после сбоя



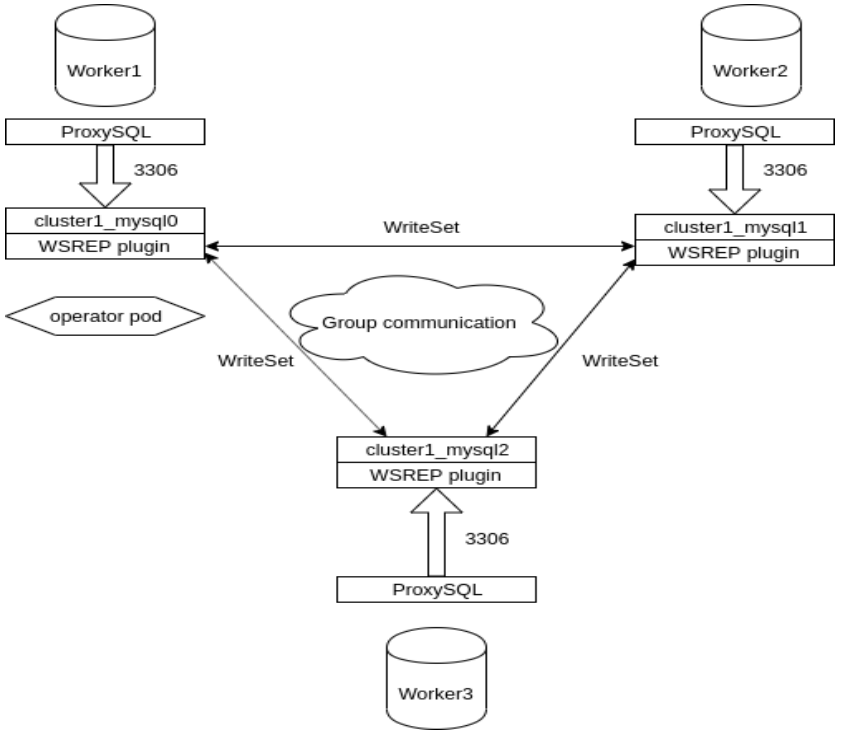
Percona XtraDB Cluster: клонирование

- Bootstrap
- State Snapshot Transfer (SST)
  - Xtrabackup
- Incremental State Transfer (IST)
  - gcache.size



Percona XtraDB Cluster: выполнение запросов на запись

- BEGIN;
- Запросы
- COMMIT:
  - Выделить write-set
  - Получить Transaction ID
  - Передать write-set
  - Подождать сертификации
  - Вернуть OK клиенту



Percona XtraDB Cluster: *_seqno

- global_seqno (x,y,z)
- local_seqno n1(a,b,c) n2(a,p,r) n3(m, n, o)
- last_seen_seqno
  - Для сертифицируемой trx
  - Определяет нижнюю границу диапазона сертификации
- depends_seqno

Percona XtraDB Cluster: Flow Control

- Асинхронное копирование транзакций на узлы
- Асинхроное применение
  - Глобальный порядок
- Очередь получения
  - Flow Control

Percona XtraDB Cluster: особенности

- Каждый узел – полная копия данных
- Запись со скоростью самого медленного сервера
- Виртуально-синхронная репликация вместо двух-фазного коммита
- Узел без кворума не выполняет запросы
- COMMIT – может выдать ошибку
- COMMIT – долгий, как минимум RTT
- Не любит больших транзакций

PXC оператор: запуск кластера

```
$ git clone -b release-1.2.0 \
https://github.com/percona/percona-xtradb-cluster-operator
$ cd percona-xtradb-cluster-operator
$ kubectl apply -f deploy/bundle.yaml

$ kubectl apply -f deploy/secrets.yaml
$ kubectl apply -f deploy/ssl-secrets.yaml
$ kubectl apply -f deploy/cr.yaml
```

PXC оператор: запуск кластера

```
$ kubectl get pods
NAME                                               READY   STATUS    AGE
cluster1-proxysql-0                                4/4     Running   4m4s
cluster1-proxysql-1                                4/4     Running   5m3s
cluster1-proxysql-2                                4/4     Running   4m51s
cluster1-pxc-0                                     2/2     Running   10m
cluster1-pxc-1                                     2/2     Running   4m24s
cluster1-pxc-2                                     2/2     Running   3m33s
percona-xtradb-cluster-operator-745476cd76-p2qlw   1/1     Running   13m
```

Подключение из приложения

```
$ kubectl run -i --rm --tty --image=percona:5.7 \
  --restart=Never  percona-client \
  -- mysql -h cluster1-proxysql -uroot -proot_password
```

- В том же пространстве имён: имя-кластера-proxysql
- Можно задать полный путь с пространством имён: cluster1-proxysql.default.svc.cluster.local
- Сервис в K8S Service с типом ClusterIP балансирует соединения между всеми mysql портами ProxySQL

ProxySQL нужен каждому

- Балансировщик нагрузки
- Работает по протоколу MySQL
- На уровне:
  - Запросов
  - Транзакций
  - Пользователей
- Разделение SELECT / INSERT+UPDATE+DELETE

ProxySQL: настройка

```
$ kubectl exec -it cluster1-proxysql-0 \
  -- mysql --port=6032 --protocol=tcp \
  -uproxyadmin -padmin_password
```

- Кластеризация: много ProxySQL серверов с общими настройками
- Все настройки можно менять по MySQL протоколу, TCP/6032.

ProxySQL: настройка

- Оператор поддерживает список серверов в актуальном состоянии
- Все сервера в таблице **mysql_servers**
- Роли серверов (писатель, читатель, запасной писатель) в **mysql_galera_hostgroups**

ProxySQL: настройка, пользователи

```
mysql> GRANT ALL PRIVILEGES ON database1.* \
 TO 'user1'@'%' IDENTIFIED BY 'password1';
$ kubectl exec -it cluster1-proxysql-0 \
  -- proxysql-admin \
   --config-file=/etc/proxysql-admin.cnf --syncusers
```

- Пользователи должны быть в ProxySQL **mysql_users**
- Такие же имена и пароли должны быть в MySQL
  - Достаточно добавить на одном узле PXC
  - Права пользователей (GRANT/REVOKE) – в MySQL

ProxySQL: настройка, роутинг

- **mysql_query_rules**:
  SELECTы обрабатываются читателями,
  остальные запросы (включая select for update) обрабатываются единственным писателем.
- Роутинг запросов можно делать на основе
  - RegExp
  - Имени пользователя (prod_ro, prod_rw)

Вандализм: проверка надёжности

```
$ kubectl delete pv \
  pvc-faeafbd9-ee5d-11e9-a1ba-42010af000de
$ kubectl delete pvc datadir-cluster1-pxc-1
$ kubectl delete pod cluster1-pxc-1 # 2X
```

- Удаляем диск одного из узлов

Вандализм: было

```
mysql> select @@hostname;
+---------------+
| @@hostname    |
+---------------+
| cluster1-pxc-1 |
+---------------+
1 row in set (0.01 sec)
```

- Удаляем диск одного из узлов

Вандализм: проблема обнаружена

mysql> select @@hostname;
ERROR 2013 (HY000): Lost connection to MySQL server during query

- Запрос возвращает ошибку
- Proxysql закрывает соединение к приложению

Вандализм: failover

```
mysql> select @@hostname;
ERROR 2006 (HY000): MySQL server has gone away
No connection. Trying to reconnect...
Connection id:    309
Current database: *** NONE ***

+---------------+
| @@hostname    |
+---------------+
| cluster1-pxc-2 |
+---------------+
1 row in set (0.03 sec)
```

- При реконнекте, proxysql выберет живой узел

Вандализм: recovery

```
$ kubectl get pods
NAME              READY   STATUS    RESTARTS   AGE
cluster1-pxc-0    1/1     Running   0          34m
cluster1-pxc-1    1/1     Running   0          13m
cluster1-pxc-2    1/1     Running   0          32m
```

- Узел PXC с чистым диском сам просит данные с другого узла: State Snapshot Transfer
- Если диски с данными, то нужна только разница: Incremental State Transfer
  - wsrep_provider_options="gcache.size=512M"

Горизонтальное масштабирование

Приложение

- Быстрый старт нового узла
- Больше процессора, больше нагрузка

База данных

- Медленный запуск, надо клонировать данные
- Быстрые и медленные запросы

Горизонтальное масштабирование

```
$ kubectl get pxc/cluster1 -o yaml \
  | sed -e 's/size: 3/size: 5/' \
  | kubectl apply -f -
```

- Меняет конфигурацию для кластера
- Технически почти `kubectl scale StatefulSet_name`

Ой не сработало

```
$ kubectl get pods
NAME           READY   STATUS    RESTARTS   AGE
cluster1-pxc-1   1/1     Running   0          20m
cluster1-pxc-2   1/1     Running   0          19m
cluster1-pxc-3   1/1     Running   0          5m50s
cluster1-pxc-4   0/1     Pending   0          4m28s
```

- Количество узлов в кластере K8S недостаточно
- kubectl describe pods cluster1-pxc-4
  - 0/4 nodes are available: 4 node(s) didn't match pod affinity/anti-affinity
- gcloud container clusters resize CLUSTER_NAME --num-nodes 5

Горизонтальное масштабирование

```
mysql> show status like 'wsrep_cluster_size';
+--------------------+-------+
| Variable_name      | Value |
+--------------------+-------+
| wsrep_cluster_size | 5     |
+--------------------+-------+
```

- Как только у K8S будут ресурсы, всё начинает работать

Горизонтальное масштабирование

```
$ for i in $(seq 200) ; do \
  mysql -N -h cluster1-proxysql -uroot -proot_password \
   -e 'select @@hostname' 2>/dev/null ; done | sort -u
cluster1-pxc-0
cluster1-pxc-1
cluster1-pxc-2
cluster1-pxc-3
```

- Один “писатель”
- Четыре “читателя”

Масштабирование вниз

```
$ kubectl get pxc/cluster1 -o yaml \
  | sed -e 's/size: 5/size: 3/' \
  | kubectl apply -f -
```

- Не освобождает диски.
- Удаляет в порядке создания: остаются 0, 1, 2
- kubectl delete PersistentVolumeClaim \
  datadir-cluster1-pxc-{3,4} \
  proxydata-cluster1-proxysql-{3,4}

Оператор PXC

- Стабильный
- Лёгкий в использовании
- Гибкий в настройке
- Готов “ещё вчера”

Выводы

- Нужно использовать операторы MySQL
- Одиночные инстансы – только вне “Прода”
- Presslabs – да! Взамен текущей асинхронной реплики
  - и одиночных серверов
- InnoDB Cluster – всё ещё с осторожностью
- PXC Operator – ДА! Быстно, надёжно, автоматизированно.

Вопросы?

?