Базы данных и Kubernetes.

Не забудь включить запись!

План

- Назначение баз данных
- Теорема CAP
- Типы баз данных
- Кластеризация
- Шардинг
- Варианты использования БД в k8s
- Выделение ресурсов: RAM, SDD/HDD, CPU
- Существующие k8s операторы для БД
- Проблемы обслуживания: backup, миграции, безопасность, upgrade

Назначение баз данных

- OLTP
  - оперирует текущими данными
  - большой процент операций записи
  - требуется высокая скорость обработки транзакция (ms)
  - много конкурентных сессий
  - требуется консистентность данных -> (нормализация, блокировки)

Назначение баз данных

- DWH
  - оперирует данными за несколько лет
  - центральная БД для сбора информации из OLTP систем
  - данные собираются через ETL процессы
  - скорость обработки запроса может измеряться часами
  - OLAP
    - Технология для построения аналитичских отчетов

Примеры использования баз данных

- OLTP
  - магазин, биржа, биллинг в телекоме, банковская система
  - ERP, CRM
  - Витрины товаров, каталоги
- DWH
  - аналитика выше перечисленных систем
  - системы логгирования
- БД для микросервисов
- Кэш системы
- Системы управления конфигурациями

Важные характеристики/требования при выборе СУБД

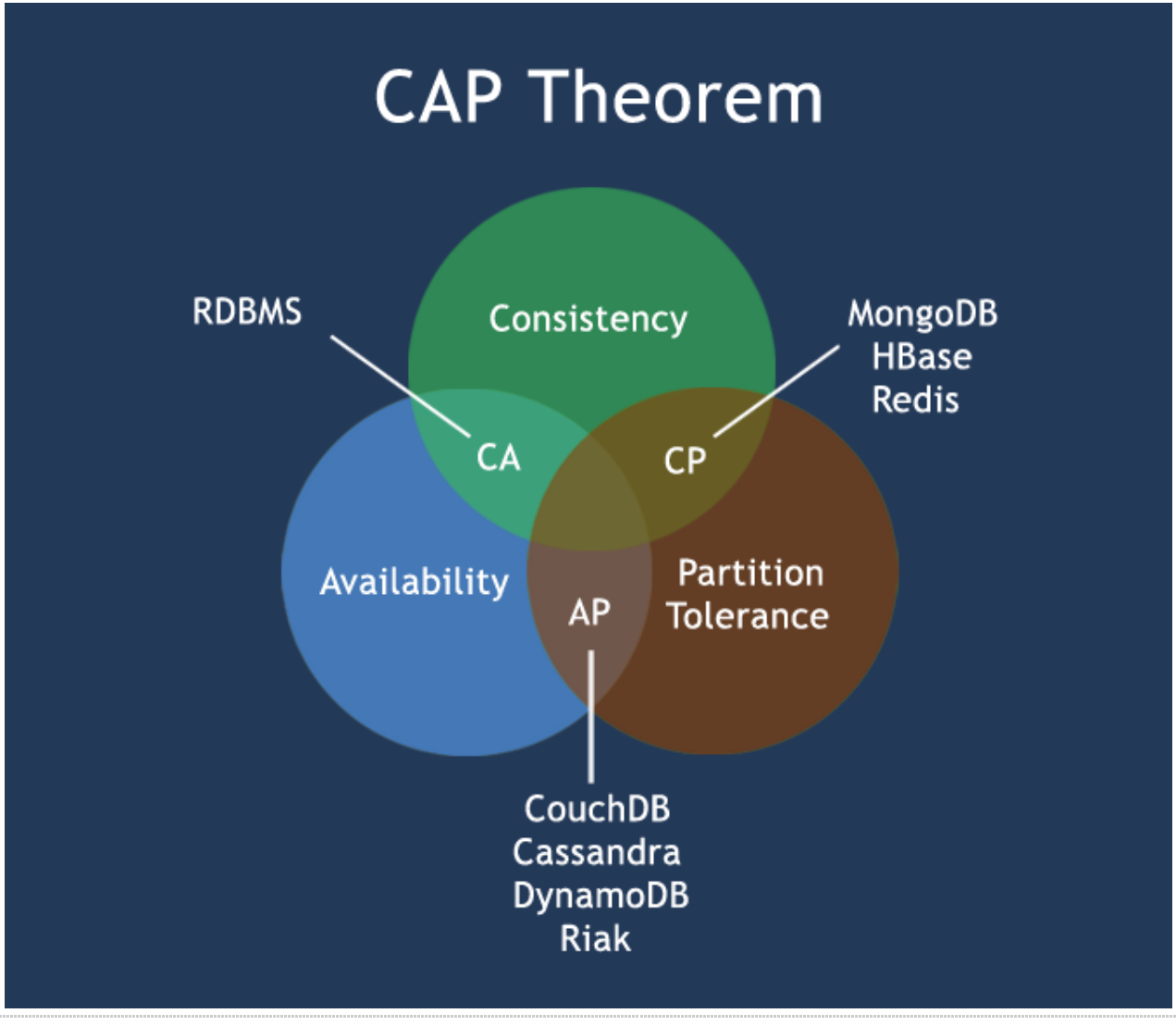
- Консистентность
  - Консистентность реплик
- Объем данных
- Размер транзакций (длительность, объем данных, кол-во одновременных транзакций)
- Кол-во одновременных соединений/сессий
- Соотношение операций чтения/записи
- Масштабирование
  - Шардинг

Теорема CAP

CAP-теорема — утверждение о том, что в **распределённых системах** нельзя одновременно добиться трёх свойств:

- Согласованность (Consistency)
  - все рабочие узлы содержат одинаковую информацию
- Доступность (Availability)
  - возможность доступа к кластеру, даже если узел в кластере выходит из строя.
- Терпимость к разделению сети. (Partition Tolerance)
  - Независимо от сбоев в работе сети узлы продолжают работать/

Диаграмма CAP



Теорема CAP. Комбинации

- CA - система доступна и консистентна. Нежизнепособна в ненадежной сети.
- CP - не будет доступна пока нет полной синхронизации между всеми узлами
- AP - Данные на разных работающих узлах могут отличаться

Типы баз данных

- Реляционные (SQL)
  - Oracle, MSSQL, PostgreSQL, MySQL
  - построчное хранение данных
  - ACID, MVCC
- Key-value
  - Redis, Etcd, Consul, Memcached
  - Быстрый доступ к данным

Типы баз данных

- Column-oriented
  - Clickhause, HP Vertica
  - Предназначены для аналитики
- wide column store (two-dimensional key-value store)
  - HBase, Cassandra, ScyllaDB
  - Предназначены для распределнных вычислений BigData
- Document-oriented
  - MongoDb, CouchDB
  - JSON формат хранения, REST API

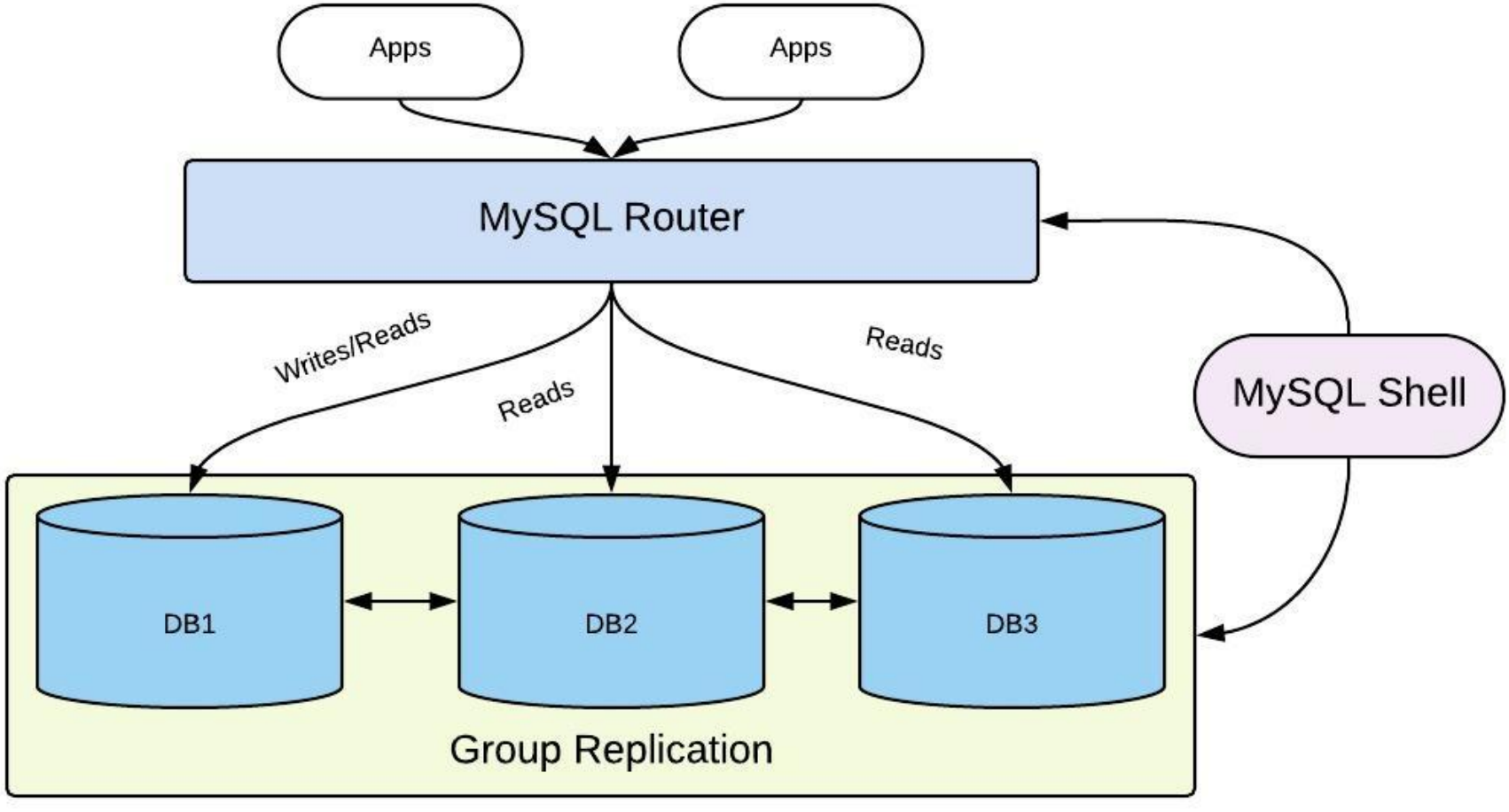
Репликация.

- Журналы (REDO logs, WAL, transactional logs)
- Репликация*
  - Процесс синхронизации данных путем доставки журналов на другие ноды
  - Задачи: баласировка чтения(/записи?), fault tolerance
  - Проблемы: организация автоматического переключения
- Разновидности:
  - Асинхронная
  - Синхронная
  - Полусинхронная

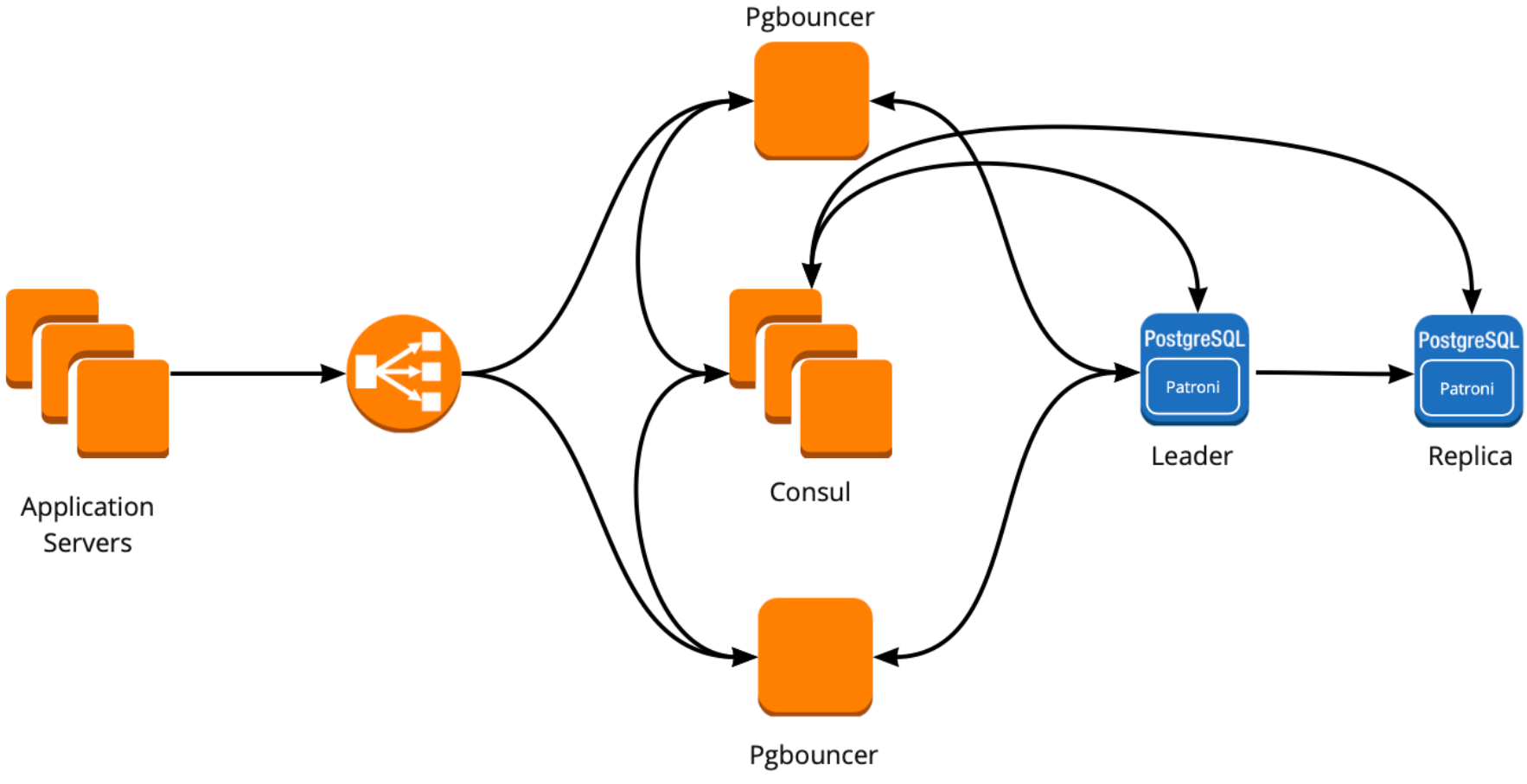
Кластеризация.

- Кластеризация
  - Полноценная масштабируемая, отказоустойчивая конфиграция
  - MySQL: Galera cluster, InnoDB cluster
  - PostgreSQL: Patroni, Slony
  - Oracle: RAC (требует общего хранилища)
  - MSSQL: AlwaysON
  - Redis Cluster

InnoDB cluster



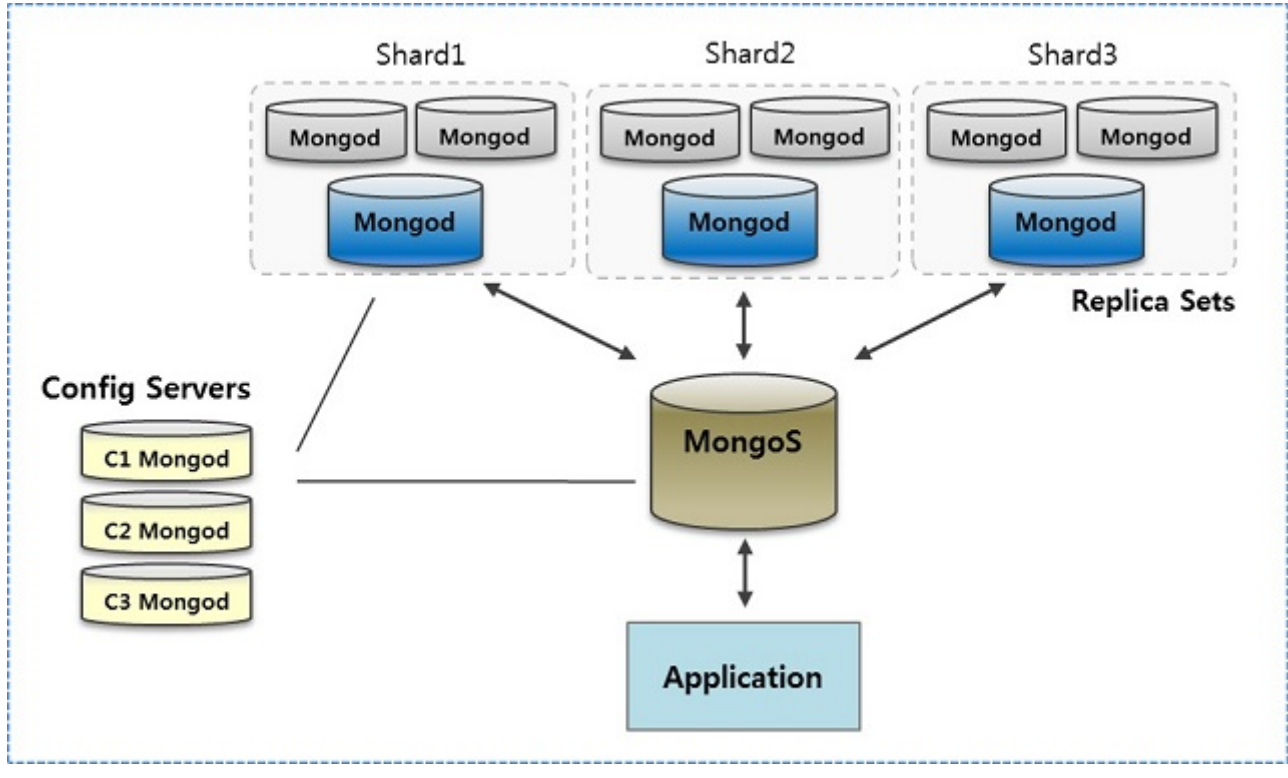
Patroni cluster



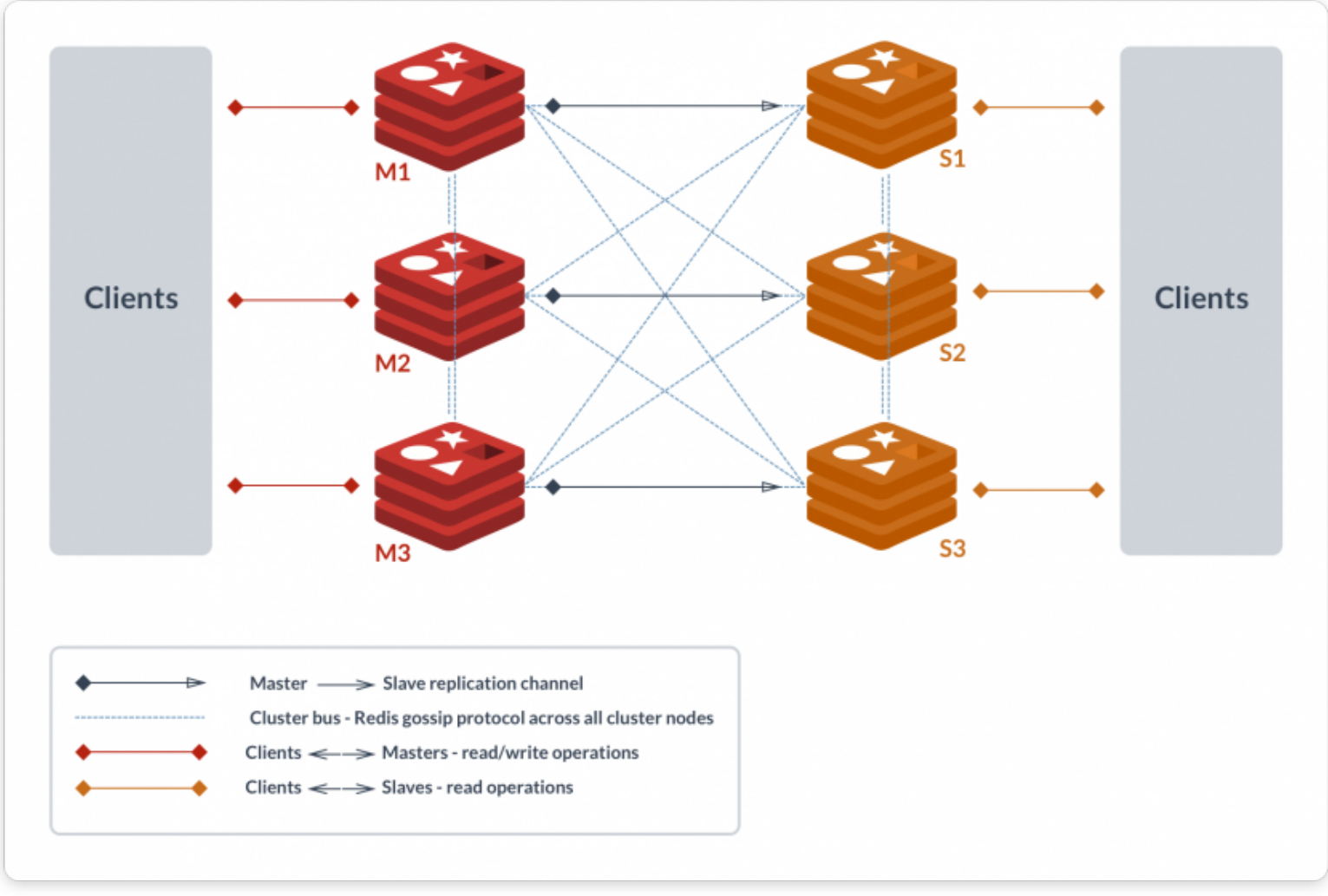
Proxy, pools, balancers

- Экономия времени на дорогой операции коннекта к базе
- Поддержка очередей
- Ограничение кол-ва подключений
- Балансировка запросов на чтение/запись между нодами
- Балансировка запросов на запись в разные базы на разные ноды в режиме мульти-мастер
- Анализ запросов
- Примеры:
  - MaxScale (Galera cluster)
  - ProxySQL (PXC)
  - InnoDB Cluster Router (Mysql)
  - PgBouncer (PostgreSQL)

MongoDb Шардинг



Redis cluster Шардинг



Варианты использования БД в k8s

- Облачная БД как сервис
- Отдельная установка с доступом из k8s
- Установка в кубернетес на выделенные ноды
- Установка на ноды совместно с другими сервисами со строгим выделением ресурсов

Варианты деплоя в кубернетес

- Pods/Deployments
  - тестовые базы
  - маленькие кластеры, которые легко восстановить
  - можно использовать шаредный PV для рестарта пода
- StatefulSets
  - подходит для большинства БД
  - есть много реализаций для cloud-native БД
- Operators
  - наилучший способ
  - много нужного дополнительного функционала

Механизмы k8s для организации HA

- StatefulSet - Контроллер деплоя. Описан вся логика действий, предпринимаемых в случае падения узла (недоступности pod'a).
  - каждая реплика модуля будет иметь свое собственное состояние и будет использовать собственный том.
  - стабильный уникальный порядковый сетевой идентификатор, который можно использовать для обнаружения всех услов сета
- PodAntiAffinity — возможность указывать определённым pod'ам, чтобы они не находились на одном и том же узле.
- PodDisruptionBudgets — лимит на количество экземпляров pod'ов, которые можно одновременно выключить в случае плановых работ.

Примеры StatefulSets

- Prometheus
- MySQL
- Cassandra

Операторы для БД

- Позволяют создавать новые ресурсы (custom resources)
- Позволяют прописывать дополнительную логику для работы этих ресурсов
- Репозиторий с ссылками на операторы

Демо. Смотрим как устроены операторы для различных СУБД

Хранение данных

- LocalStorage
  - ReadWriteOnce
  - Sata, SAS, NVME, SSD
- Network Block Device
  - ReadWriteOnce
  - ISCSI, FC, Ceph RBD
  - AWS EBS, GCEPersistentDisk
- Network File System
  - ReadWriteMany
  - NFS, CephFS, GlusterFS
  - AWS EFS

IOPS и latecy

- Использование облачных хранилищ увеличивает latency
- использование LocalStorage не отличается от использования на голой системе
- ISCSI, FC не отличается от использования на голой системе
- есть возможность лимитировать iops для подов
    - storage-iops

Память

- СУБД требуют много памяти
  - память у СУБД лимитируется настройками
  - buffer pool - 50-80%
  - каждый коннект к базе имеет свою память
- OOM killer не желателен
- Операции сортировки и группировки часто используют диск

Сеть

- Для запросов к БД должны быть реализованы балансеры
- Реплики могут создавать достаточно массивный обмен данными по сети
    - Зависит от объема операций записи
- Бэкапы также будут грузить сеть
- Стоит задумать о разделении сети на служебную и паблик
- Стоит задуматься над включением TLS для доступа к базе и балансерам

CPU

- Использование напрямую зависит от кол-во конкурентных подключений
- IO waitings также могут влиять на высокий LA

Backup & restore

- Выбор стратегии бэкапа
- Выбор места для сохранения
- Выбор расписания
- Подготовка сценариев восстановления
- В операторах уже есть реализованые возможности

Миграции

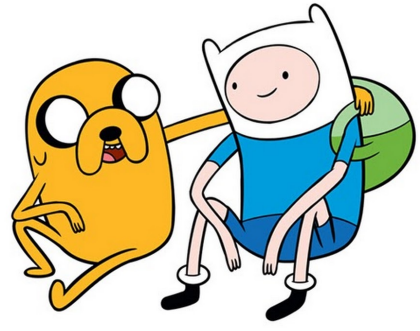
- Использование средств CI/CD для БД
  - Flyway, Liquibase, Alembic, etc...
- В момент накатки миграций функционал приложения может вести себя некорректно
- Миграции с изменением структуры могут занять продолжительное время

Upgrade

- Обновление версии СУБД - нетривиальная задача
- Операторы могут включать в себя такую возможность
  - Пошаговое обновление на следующую версию
- Следует избегать попыток обновлений на несколько версий вперед
- Иногда проще поднять новый кластре и перелить данные

Безопасность

- Включайте TLS для доступа к БД
  - Во многих операторах это реализовано
- Учитывайте что приложение может коннектится к БД под разными пользователями
  - Это нужно учитывать при настройке пулов
- Вероятно может понадобится шифрование БД
- Также может понадобится шифрование бэкапов



Спасибо за внимание!

Время для ваших вопросов!