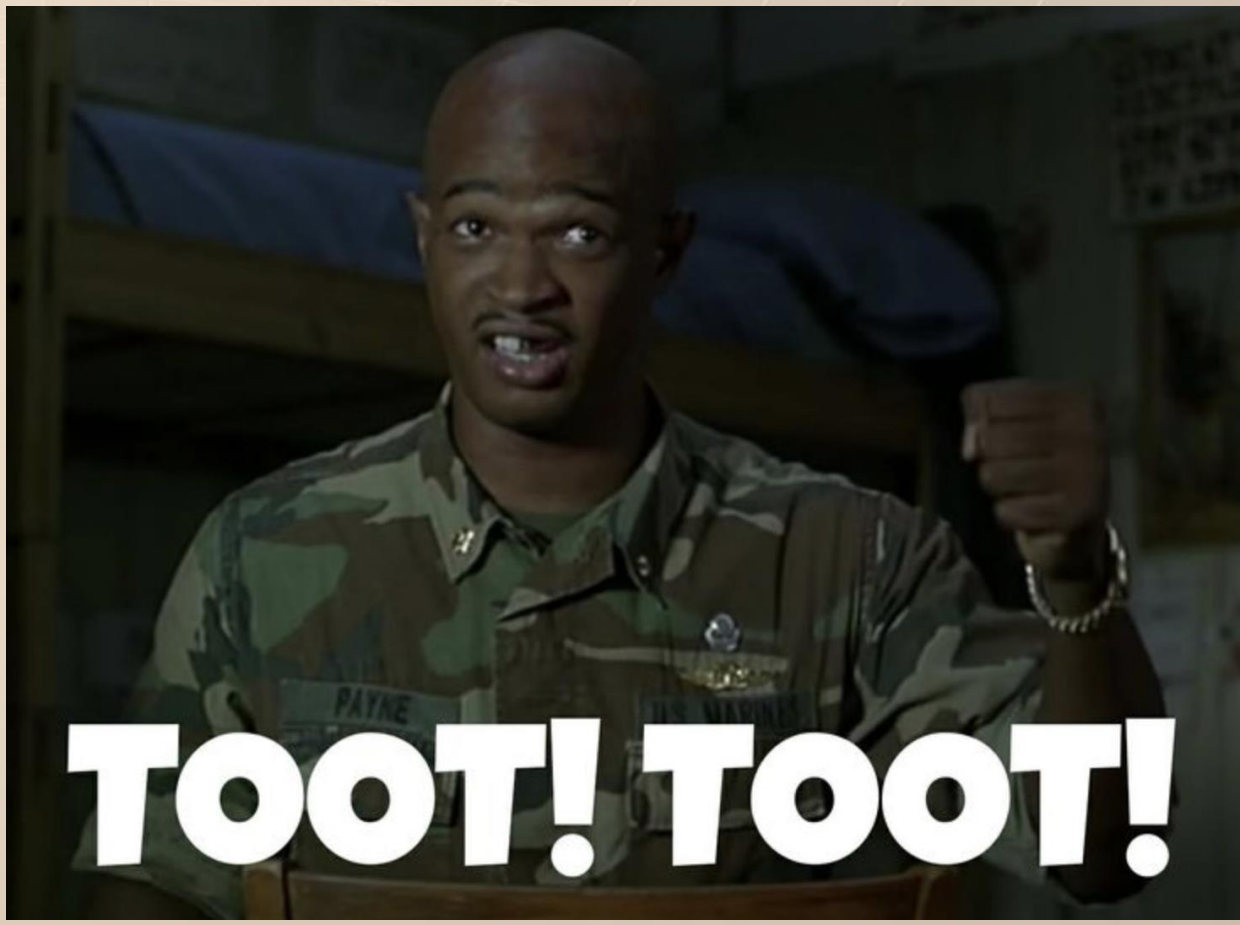
SIMA LAND

Слоник, который смог в ZFS

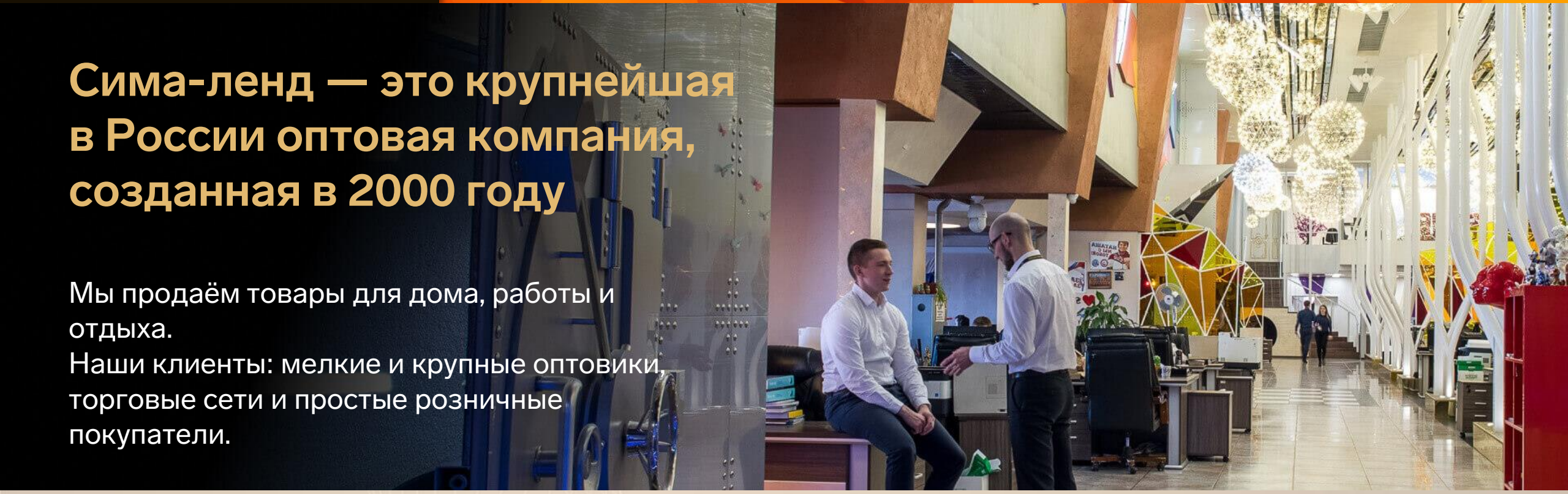


Артём Свяжин

Давайте познакомимся

- 8 лет опыта администрирования и разработки в различных высоконагруженных СУБД в т.ч. Postgresql
- Почти 14 лет опыта программирования на разных языках
- 8 лет опыта администрирования и разработки в UNIX операционных системах
- Опыт технической экспертизы 1с
- Нет ничего невозможного - девиз

SIMA LAND



Сима-ленд — это крупнейшая в России оптовая компания, созданная в 2000 году

Мы продаём товары для дома, работы и отдыха.
Наши клиенты: мелкие и крупные оптовики, торговые сети и простые розничные покупатели.

Более 40ТВ
Суммарный объем БД

8ТБ
Объем основной БД

450+ чел
Сотрудников IT департамента

75К TPS
На самой главной БД

Более 100
Серверов СУБД Postgres

900
Микросервисов в k8s

План презентации

1. Причины перехода на ZFS
2. Тернистый путь перехода на COW
3. Теоретическая часть
4. Схема как оно работает у нас и нюансы настройки ZFS
5. Экспериментальная часть и выводы

Проблемы которые нас привели к COW вследствие большого роста компании

1. Большое количество машин в ЦОД

Проблемы которые нас привели к COW вследствие большого роста компании

1. Большое количество машин в ЦОД
2. Необходимость большого дискового пространства (дисков)

Проблемы которые нас привели к COW вследствие большого роста компании

1. Большое количество машин в ЦОД
2. Необходимость большого дискового пространства (дисков)
3. Требуемое время на восстановление для каждой машины

Проблемы которые нас привели к COW вследствие большого роста компании

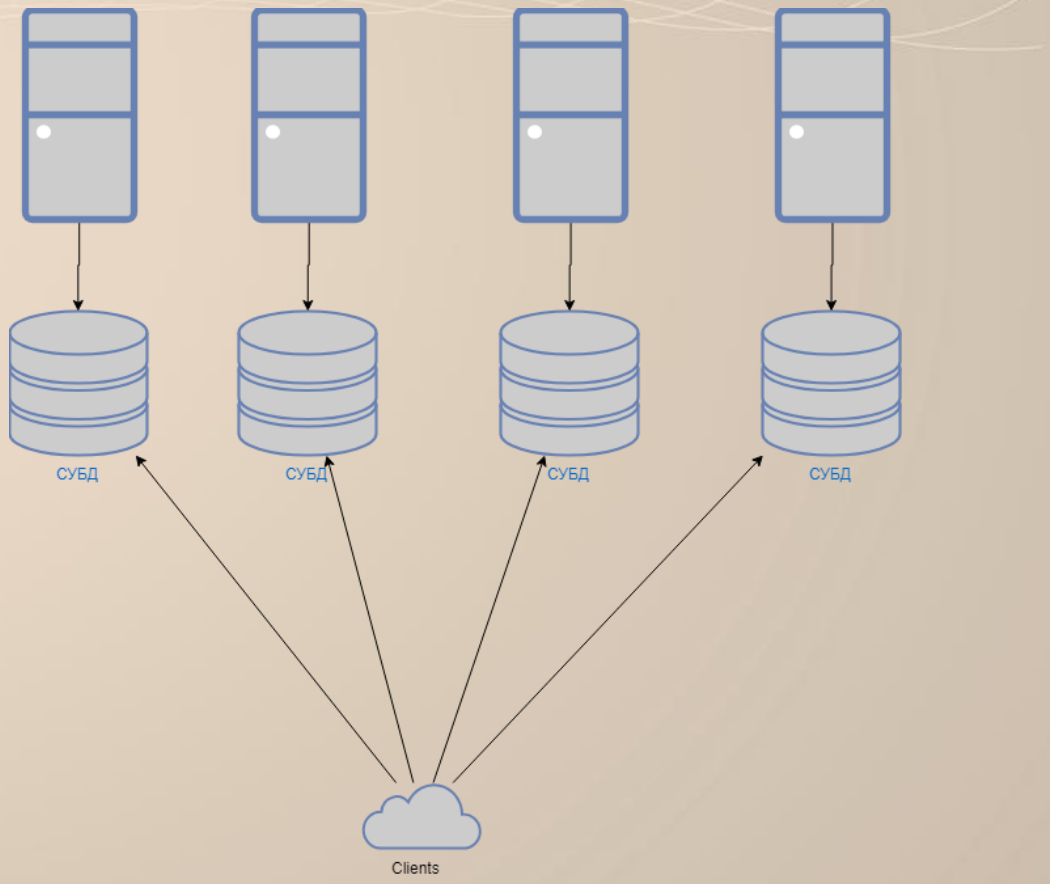
1. Большое количество машин в ЦОД
2. Необходимость большого дискового пространства (дисков)
3. Требуемое время на восстановление для каждой машины
4. Снижение скорости и качества тестирования кода

Проблемы которые нас привели к COW вследствие большого роста компании

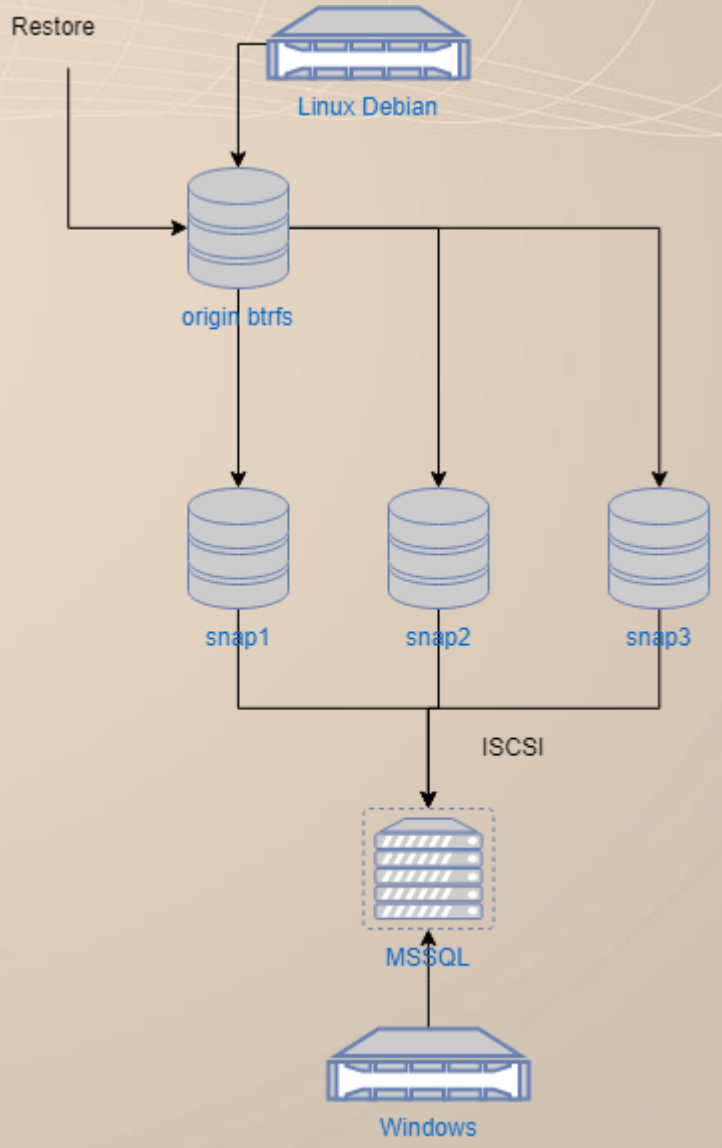
1. Большое количество машин в ЦОД
2. Необходимость большого дискового пространства (дисков)
3. Требуемое время на восстановление для каждой машины
4. Снижение скорости и качества тестирования кода
5. неудобство и неуправляемость

Как мы сдружились с COW?

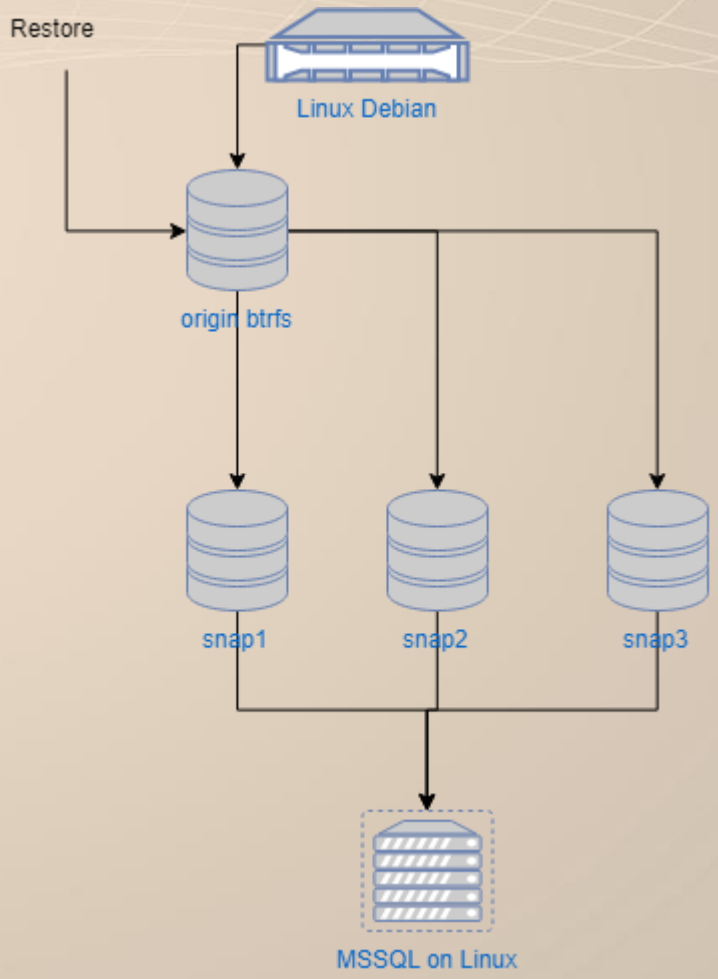
Первоначально схема с тестовыми контурами выглядела так



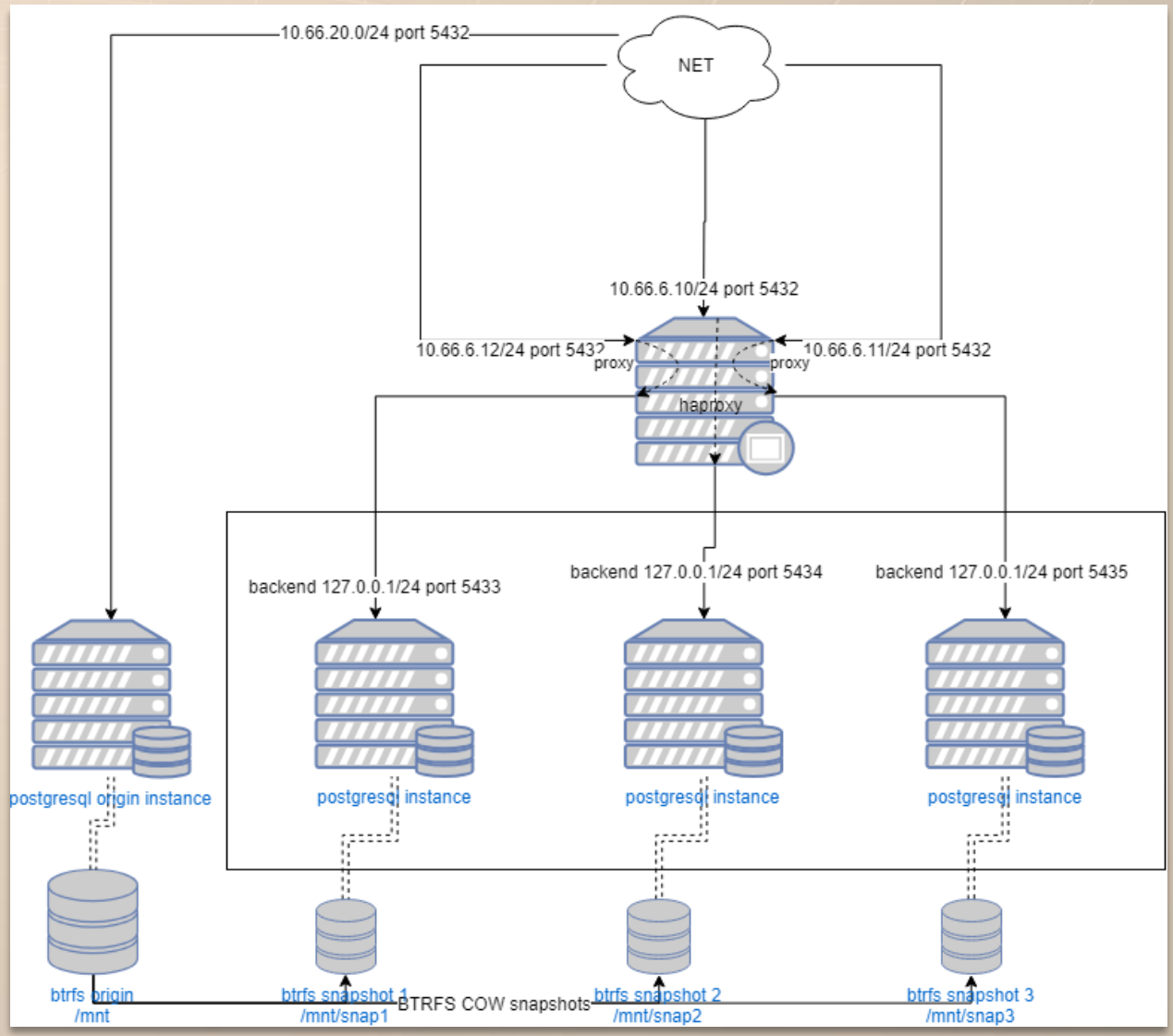
Далее произошла модификация пришли к COW btrfs



Убрали сетевые задержки ISCSI перешли на MSSQL on linux



Почему бы не сделать с postgresql?



ZFS vs BTRFS

ZFS – COW?

- Read-Modify-Write: прочесть весь экстент (Read), изменить его в середине (Modify), просчитать новую контрольную сумму и выполнить запись экстента целиком (Write).

- Redirect-Write: данные экстента записываются в новое (пустое) место, и затем в метаданные производится атомарная запись, включающая в себя новую контрольную сумму и новое расположение экстента. А старое место где данные экстента лежали раньше, помечаеся как свободное.

1 относительная производительность по сравнению с другими CoW fs

Почему же все-таки

?

SIMA LAND

Почему же все-таки

?

1. относительная производительность по сравнению с другими CoW fs
2. умеет в блочные устройства

SIMA LAND

Почему же все-таки

?

1. относительная производительность по сравнению с другими CoW fs
2. умеет в блочные устройства
3. быстрое создание и удаление снапшотов данных

Почему же все-таки

?

1. относительная производительность по сравнению с другими CoW fs
2. умеет в блочные устройства
3. быстрое создание и удаление снапшотов данных
4. есть кэш в оперативной памяти

Почему же все-таки

?

1. относительная производительность по сравнению с другими CoW fs
2. умеет в блочные устройства
3. быстрое создание и удаление снапшотов данных
4. есть кэш в оперативной памяти
5. RMW + RW

Но есть и минусы

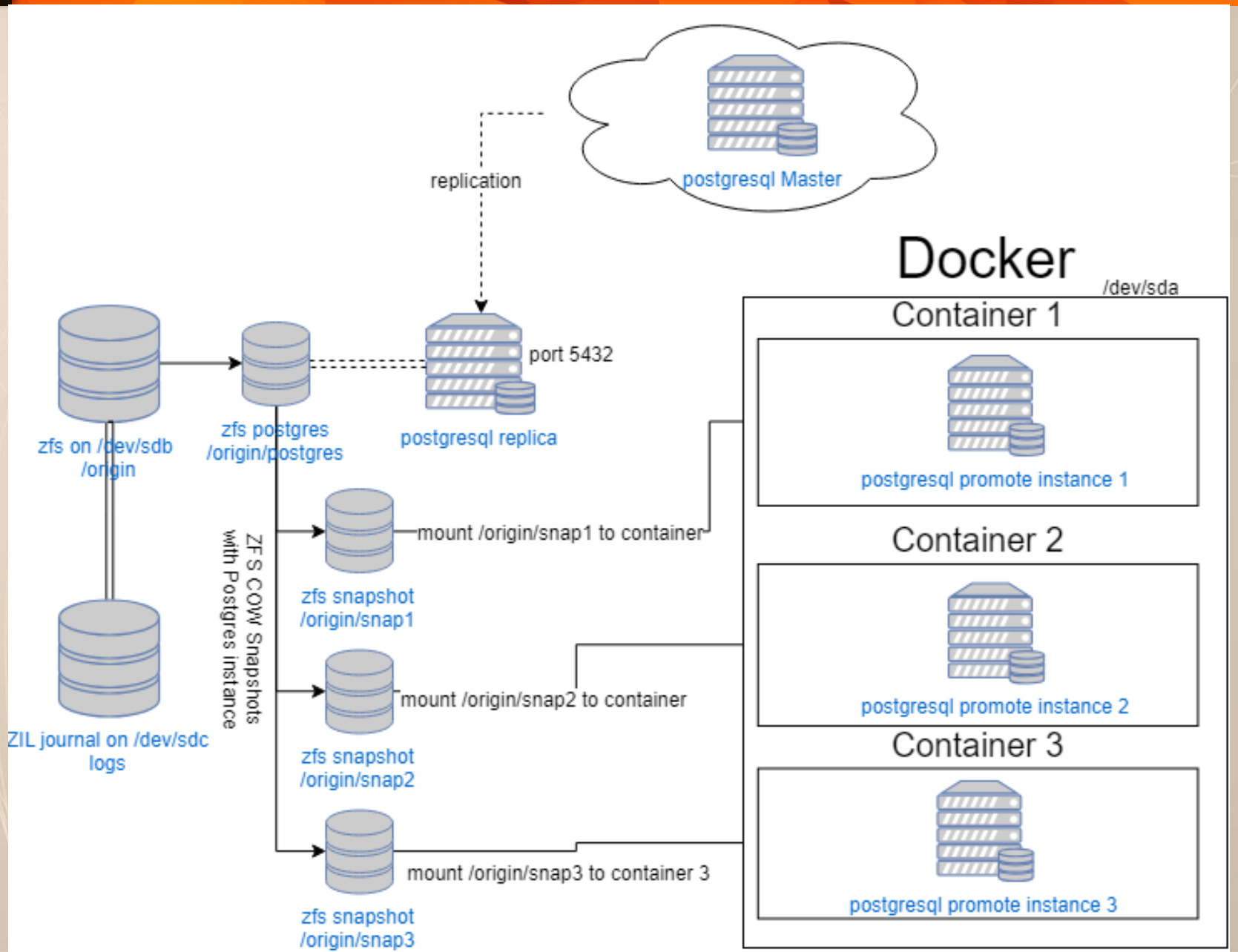
1. надо больше оперативной памяти
2. отдельный диск под журналы
3. резервирование свободного места на диске

SIMA LAND

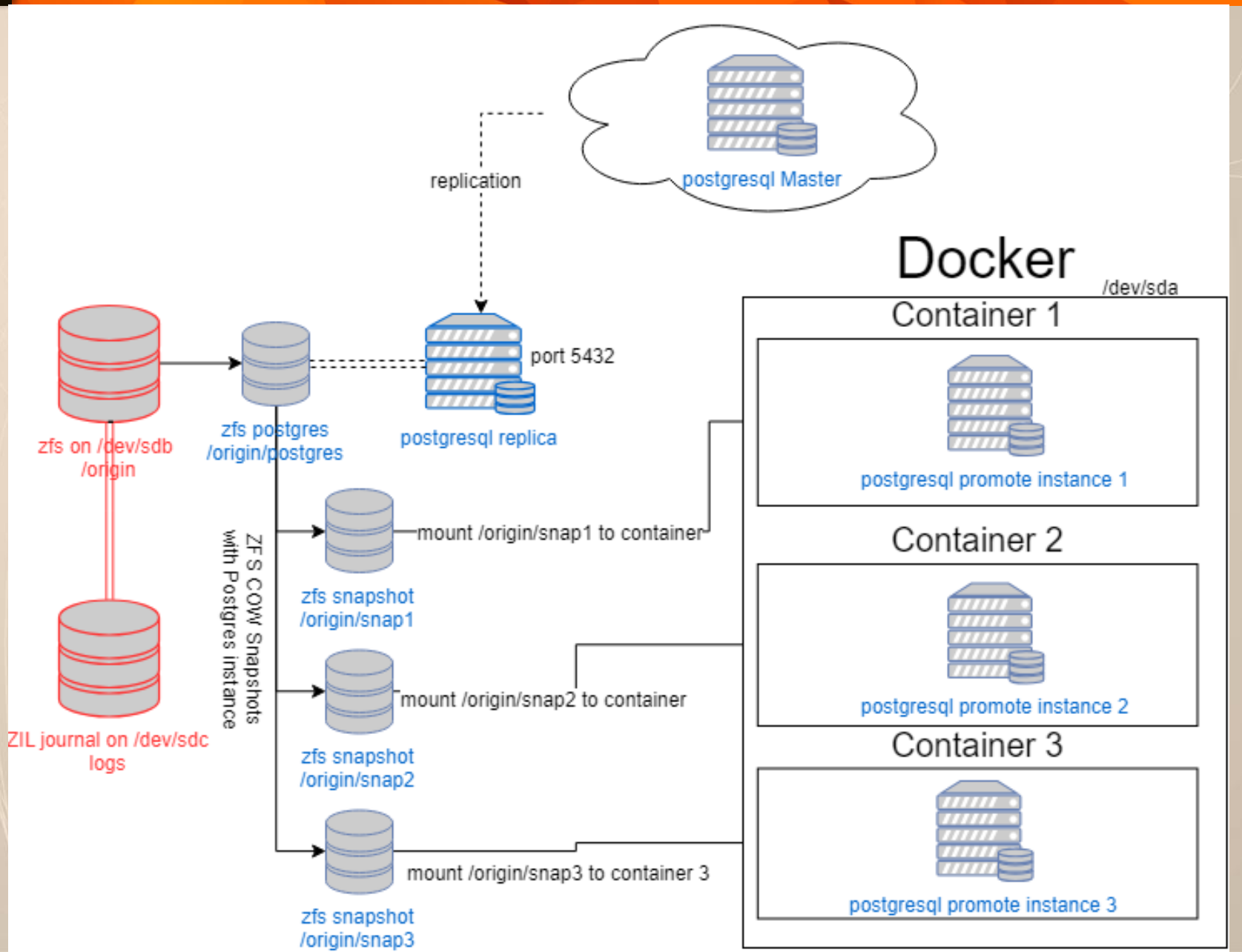
Перейдем к практическому применению совместно с

PostgreSQL

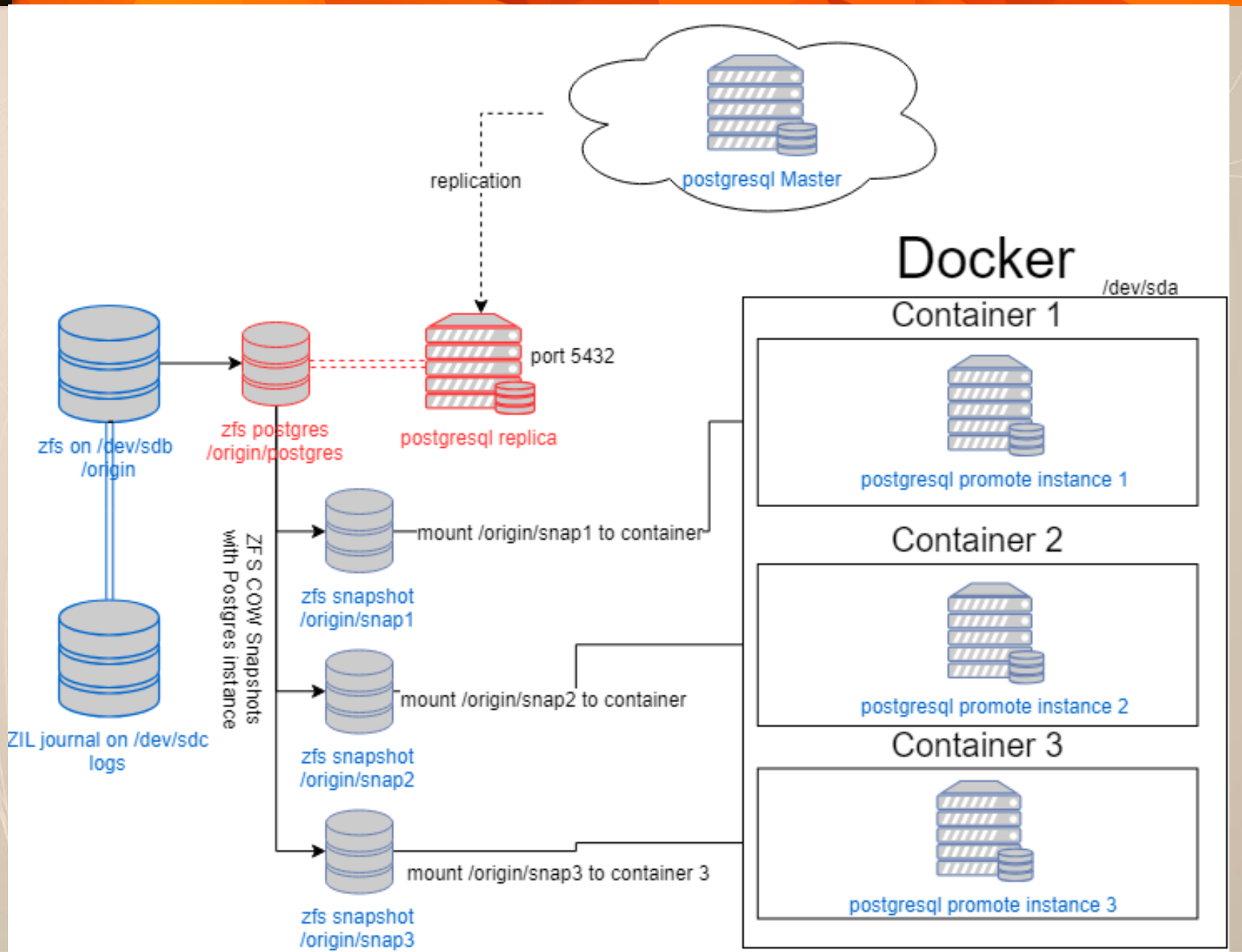
Схема, как оно работает у нас



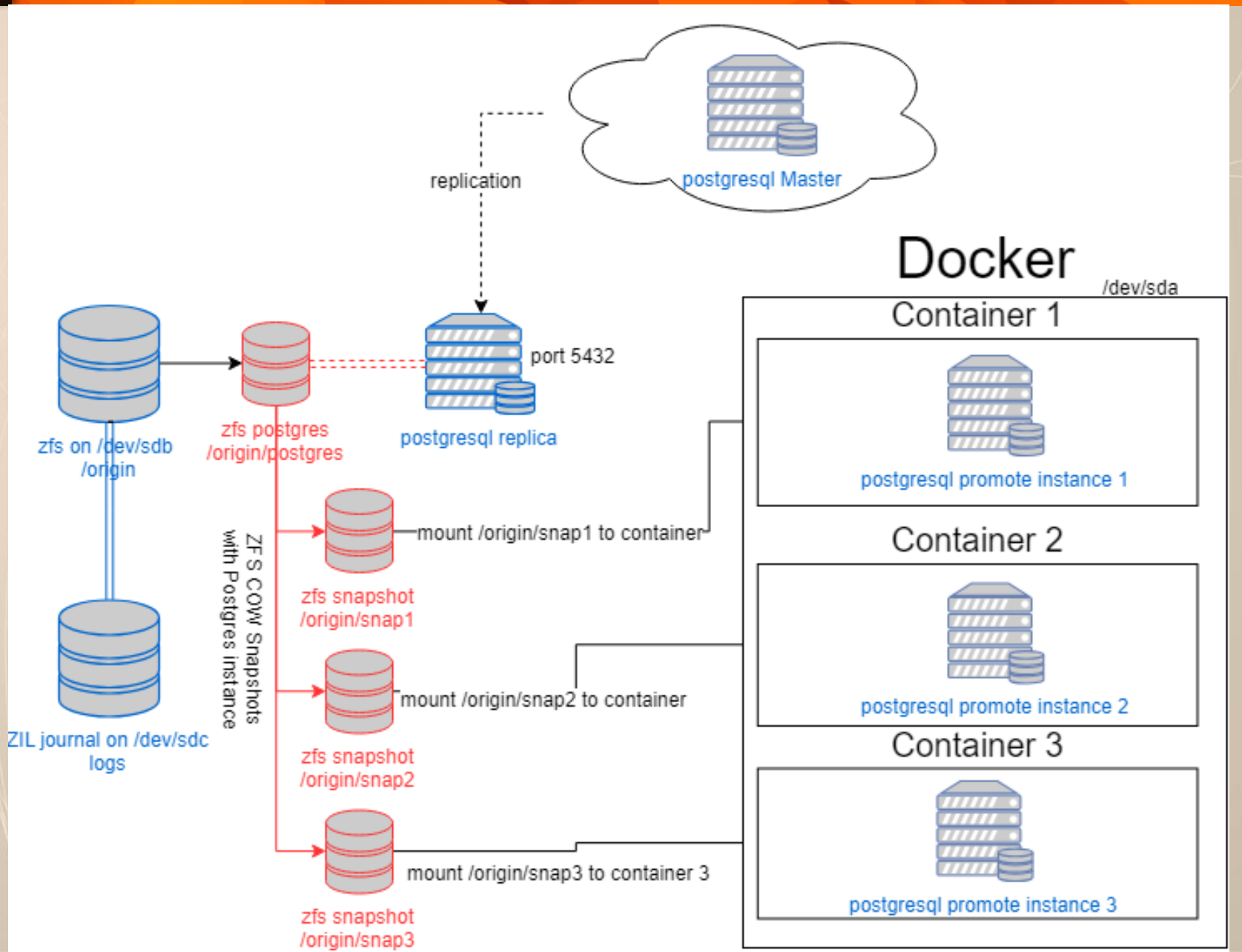
Схема, как оно работает у нас



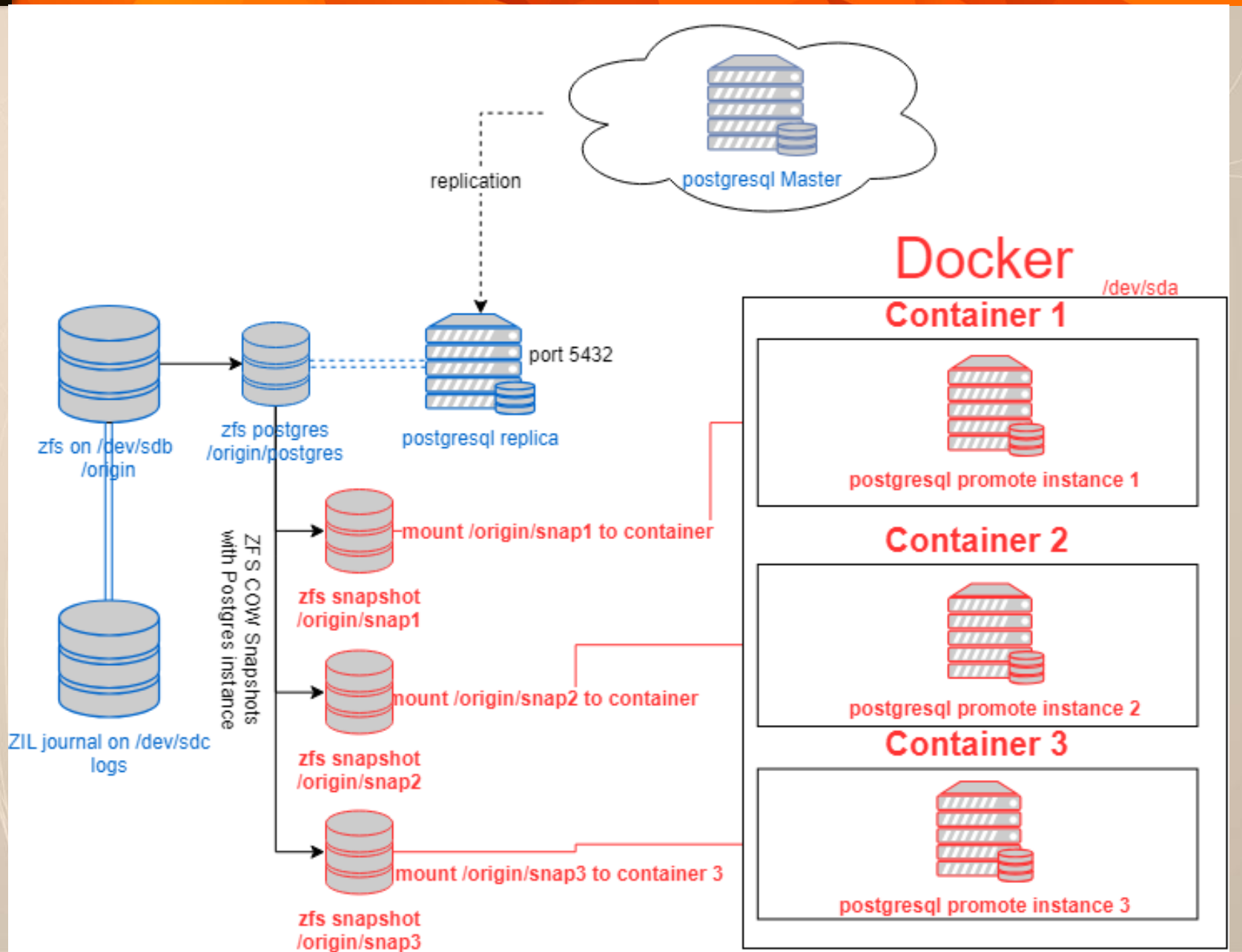
Схема, как оно работает у нас



Схема, как оно работает у нас



Схема, как оно работает у нас



Настройка zfs и zpool

- По умолчанию zfs забирает **50%** объема доступной оперативной памяти. Поэтому необходимо вычислить нужный объем ARC кэша для текущего пула по формуле 1GB+(4GB * на каждый 1TB из ZPOOL)

```
root@zfstest:/# echo "options zfs zfs_arc_min=4294967296" > /etc/modprobe.d/zfs.conf
root@zfstest:/# echo "options zfs zfs_arc_max=8589934592" >> /etc/modprobe.d/zfs.conf
```

Настройка zfs и zpool

- По умолчанию zfs забирает **50%** объема доступной оперативной памяти. Поэтому необходимо вычислить нужный объем ARC кэша для текущего пула по формуле 1GB+(4GB * на каждый 1TB из ZPOOL)

```
root@zfstest:/# echo "options zfs zfs_arc_min=4294967296" > /etc/modprobe.d/zfs.conf
root@zfstest:/# echo "options zfs zfs_arc_max=8589934592" >> /etc/modprobe.d/zfs.conf
```

- По умолчанию zfs размечает ZIL на одно блочное устройство с zpool. Так как ZIL это журнал намерений, то для снижения нагрузки ввода — вывода на zpool рекомендуется выделить отдельный диск и разметить на него ZIL.

Настройка zfs и zpool

- По умолчанию zfs забирает **50%** объема доступной оперативной памяти. Поэтому необходимо вычислить нужный объем ARC кэша для текущего пула по формуле 1GB+(4GB * на каждый 1TB из ZPOOL)

```
root@zfstest:/# echo "options zfs zfs_arc_min=4294967296" > /etc/modprobe.d/zfs.conf
root@zfstest:/# echo "options zfs zfs_arc_max=8589934592" >> /etc/modprobe.d/zfs.conf
```

- По умолчанию zfs размечает ZIL на одно блочное устройство с zpool. Так как ZIL это журнал намерений, то для снижения нагрузки ввода — вывода на zpool рекомендуется выделить отдельный диск и разметить на него ZIL.

- По умолчанию ZFS выделяет блоки динамически от 512 до 16M, т.к. postgres оперирует страницами размером 8k, то рекомендуется для получения большего кол-ва TPS «жестко» задать параметр выделения блоков соответствующим размером

```
zfs set recordsize="8k" origin
```

В эфире рубрика эээээээээксперименты

Этапы тестирования

1. Создание snapshot с реплики
2. Вставка данных в первичный сервер и snapshot, оценка результата влияния на snapshot и реплику
3. Удаление данных в реплике и snapshot соответственно, оценка влияния на snapshot и реплику
4. ANALYZE на snapshot
5. VACUUM на snapshot
6. Вывод, заключение

Создание снепшота с реплики

```
zfs snapshot origin/postgres@postgres_snap1
zfs clone origin/postgres@postgres_snap1 origin/postgres_snap1
```

Создание снепшота с реплики

```
zfs snapshot origin/postgres@postgres_snap1
zfs clone origin/postgres@postgres_snap1 origin/postgres_snap1
```

Далее необходимо запустить docker контейнер и сделать pg_promote

```
docker-compose --compatibility -f docker-compose.yml "exec" -u postgres dbtest psql -d postgres -c "SELECT pg_promote();"
```

Выполнение команды pg_promote после запуска контейнера

```yaml
version: '3.1'

services:

  temchik:
    image: postgres:15.5-bookworm
    restart: always
    shm_size: "2gb"
    ports:
      - "5434:5432"
    environment:
      PGDATA: /postgres
    volumes:
      - /origin/postgres_snap1/:/postgres
      - /docker-pg/postgresql.conf:/postgres/postgresql.conf
      - /docker-pg/pg_hba.conf:/postgres/pg_hba.conf
      - /docker-pg/pg_ident.conf:/postgres/pg_ident.conf
```

Docker-compose.yml

График изменения размера реплики

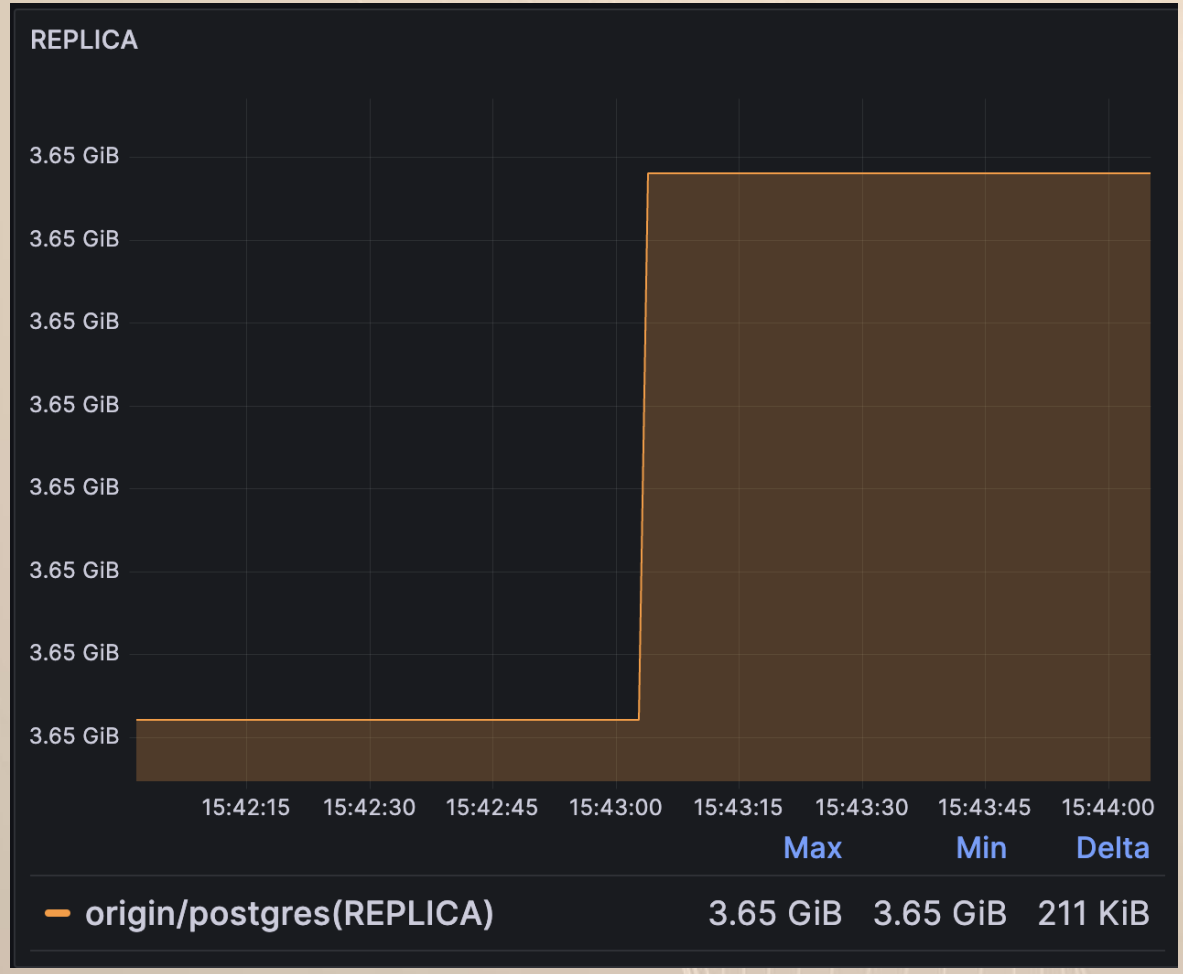


График изменения размера снапшота

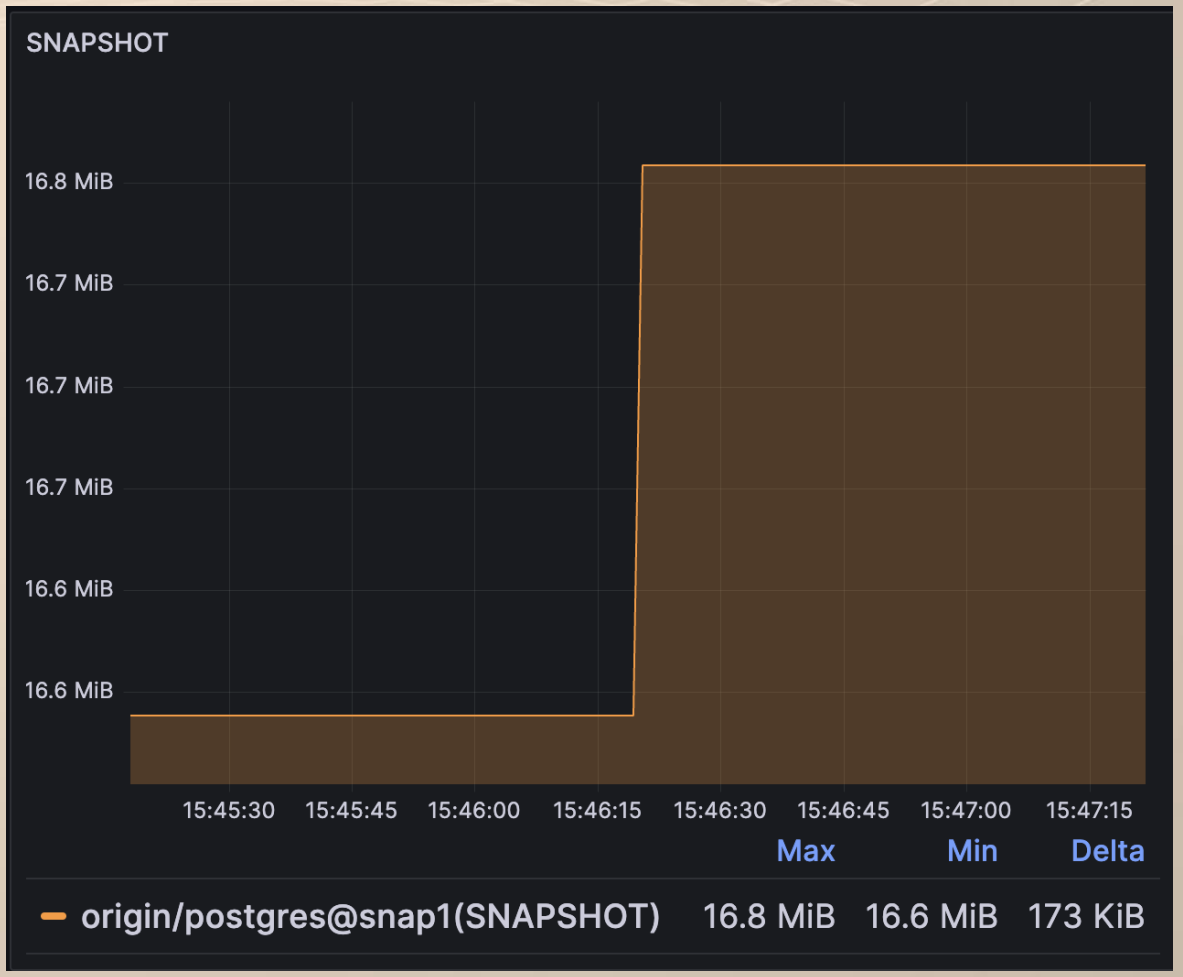


График изменения размера реплики

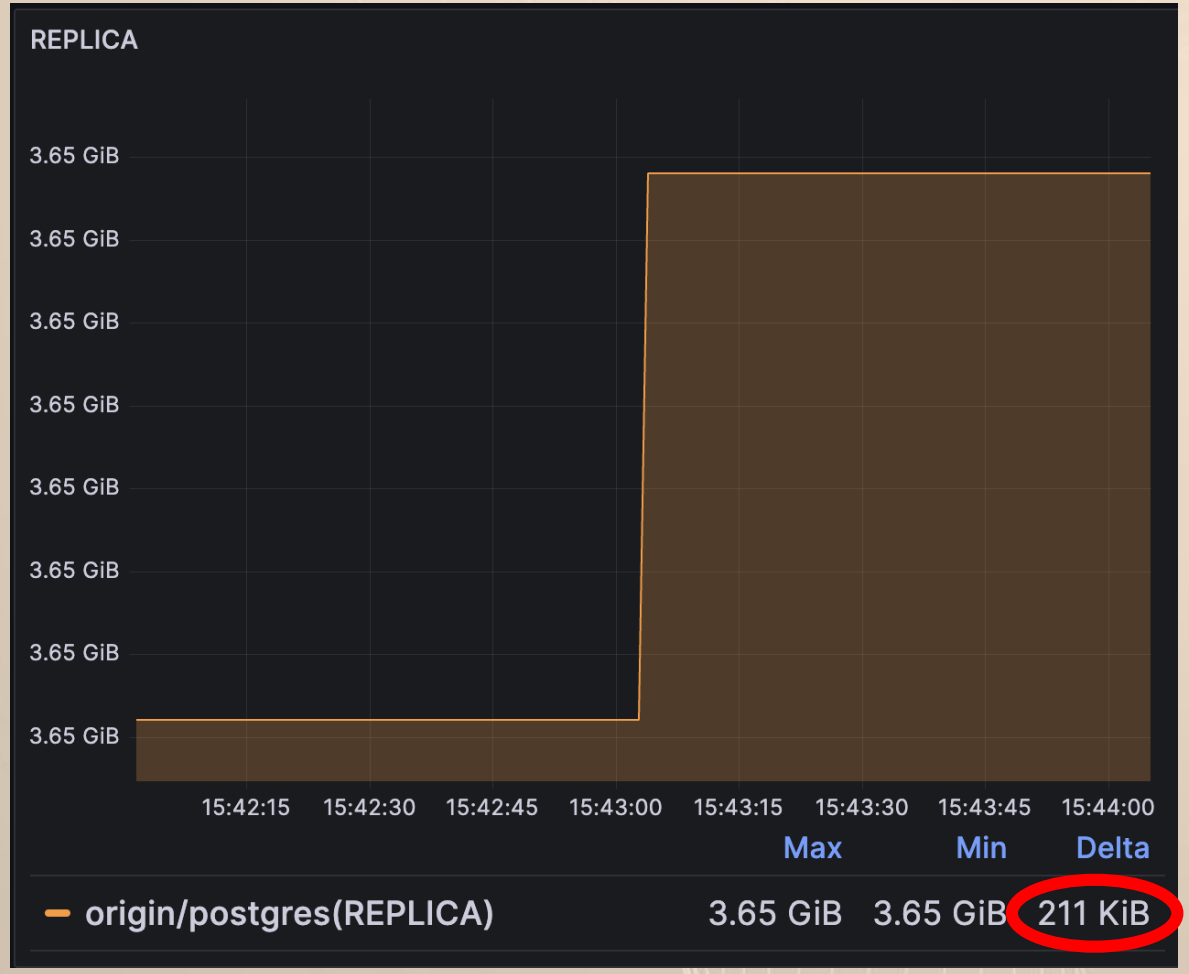
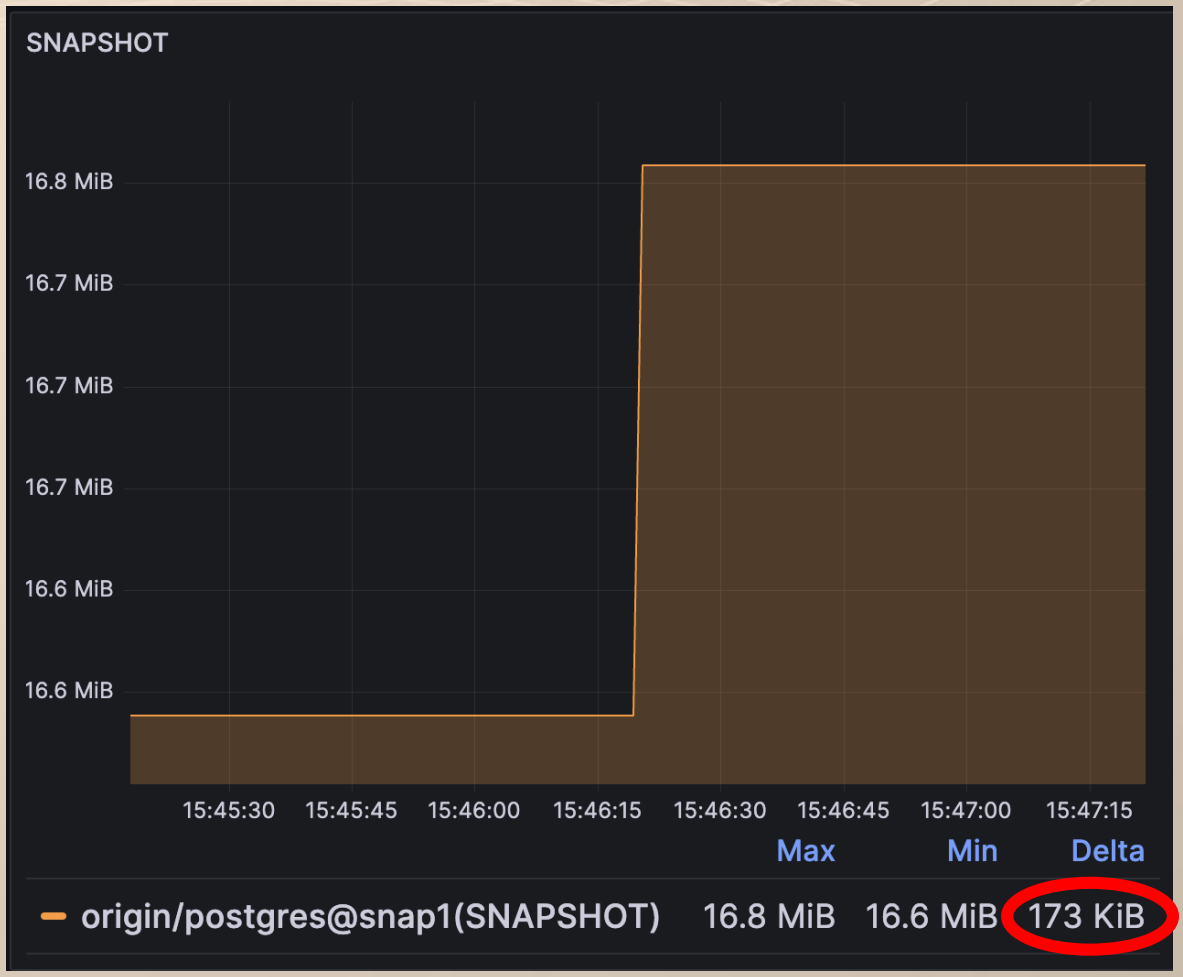
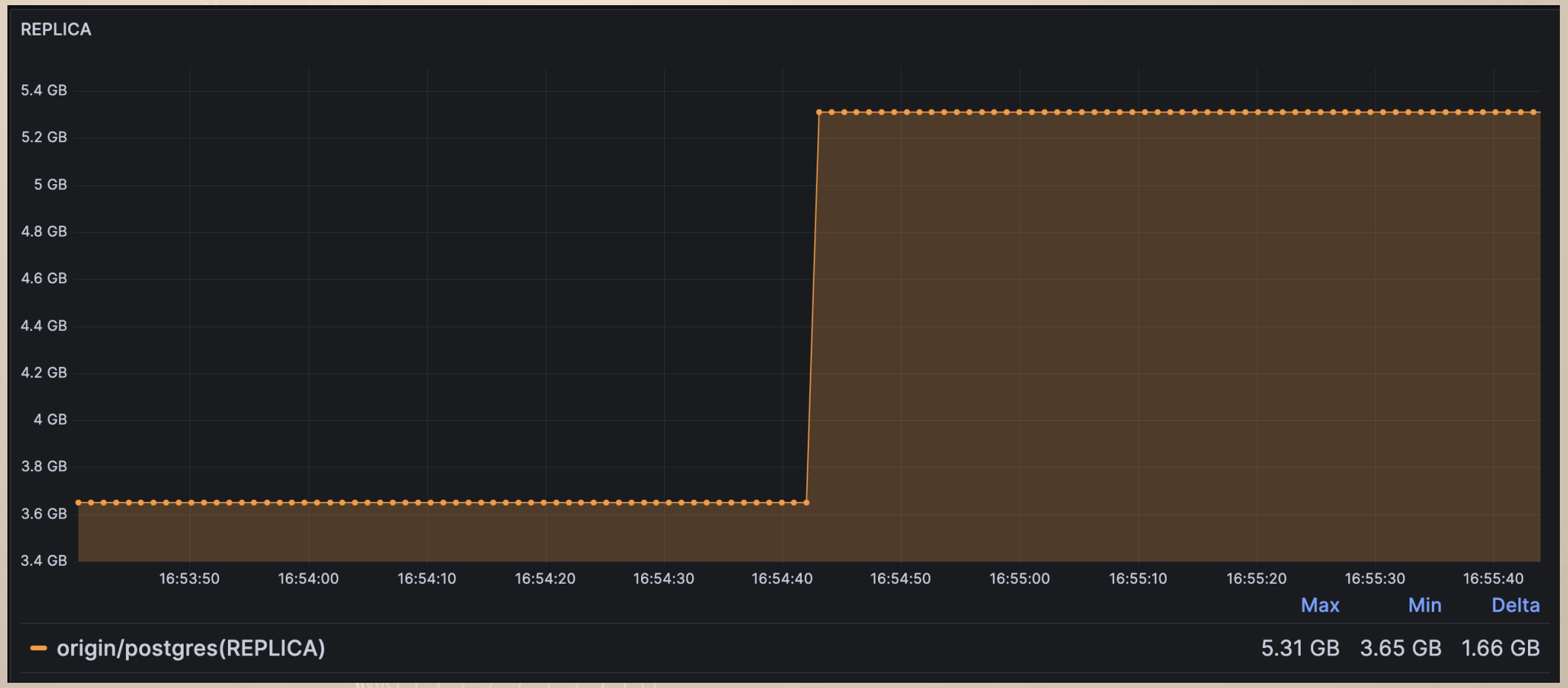


График изменения размера снапшота



Удаление данных с реплики

```
DELETE FROM boarding_passes;
```



Удаление данных с реплики

```
DELETE FROM boarding_passes;
```

Почему так? Удаление данных приводит к росту?
Объяснение кроется после удаления зависимого snapshot реплики

NAME	USED	AVAIL	REFER
origin	5.35G	187G	26K
origin/postgres	5.31G	187G	3.22G
origin/postgres_snap1	17.7M	187G	2.83G

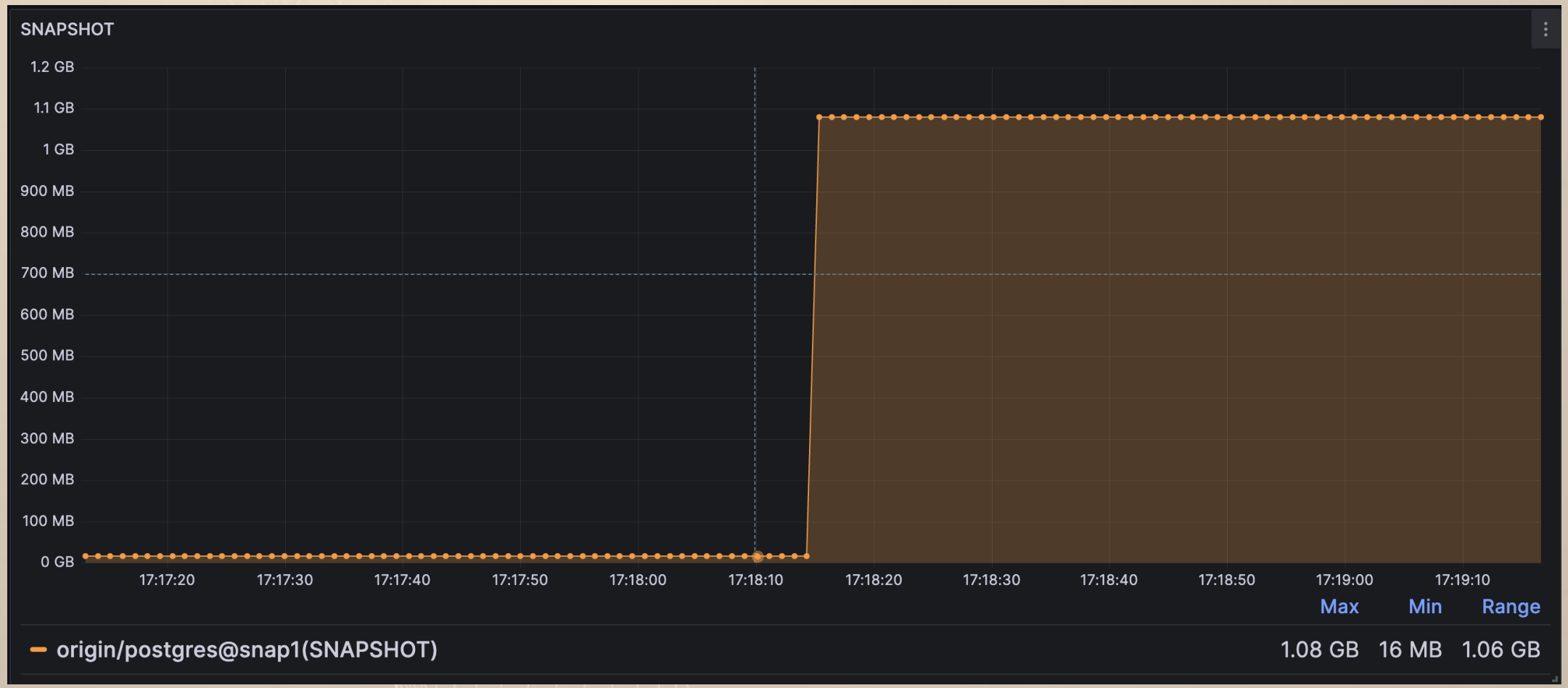
удаление зависимого snapshot реплики

```
zfs destroy -r origin/postgres@postgre_snap1
```

NAME	USED	AVAIL	REFER
origin	3.21G	190G	26K
origin/postgres	3.22G	190G	3.20G

Удаление данных в снапшоте

```
DELETE FROM boarding_passes;
```



Удаление данных в снапшоте

```
DELETE FROM boarding_passes;
```

Опять удаление и опять рост, но в данном случае в снапшоте почему так?

```
demo=# select pg_size_pretty(pg_total_relation_size('boarding_passes'::regclass));
  pg_size_pretty
 --------------
  1102 MB
```

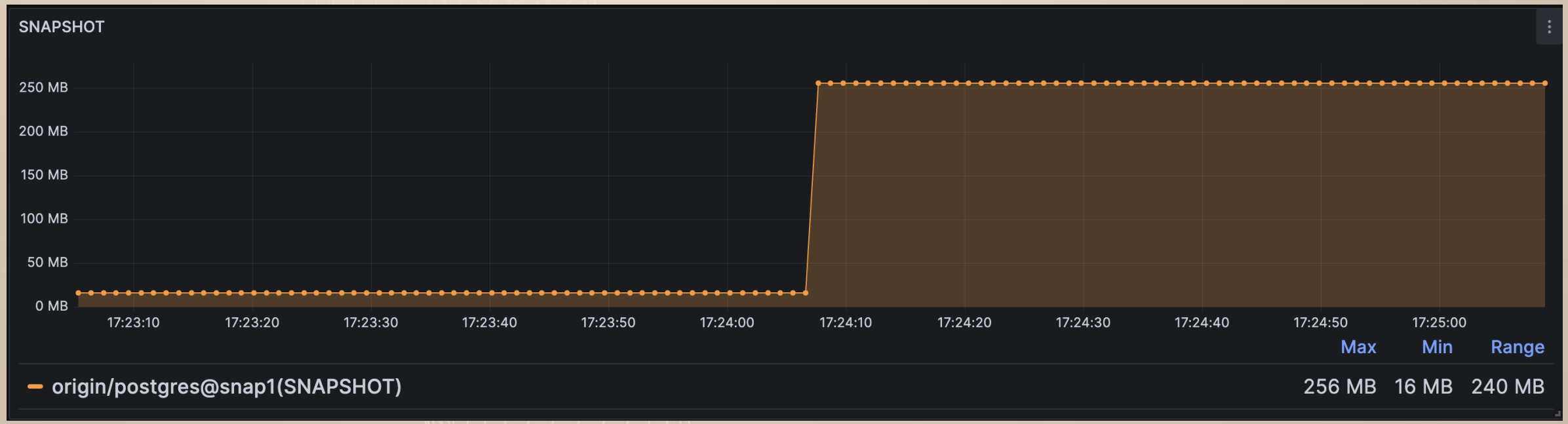
Все блоки были изменены которые касались данной таблицы включая индексы и TOAST + кусок wal

ANALYZE

Исходный размер снапшота

NAME	USED	AVAIL	REFER
origin	3.66G	189G	26K
origin/postgres	3.63G	189G	3.63G
origin/postgres_snap1	16.6M	189G	2.83G

```
ANALYZE boarding_passes;
```



ANALYZE

Исходный размер снапшота

NAME	USED	AVAIL	REFER
origin	3.66G	189G	26K
origin/postgres	3.63G	189G	3.63G
origin/postgres_snap1	16.6M	189G	2.83G

```
ANALYZE boarding_passes;
```

ANALYZE

Видим рост почему так? Всему виной реплика и promote? Для того чтобы разобраться создадим еще 1 snapshot и запромоутим его

NAME	USED	AVAIL	REFER	MOUNTPOINT
origin	3.92G	189G	26K	/origin
origin/postgres	3.63G	189G	3.63G	/origin/postgres
origin/postgres_snap1	256M	189G	2.83G	/origin/postgres_snap1
origin/postgres_snap2	16.6M	189G	2.83G	/origin/postgres_snap2

ANALYZE

Видим рост почему так? Всему виной реплика и promote? Для того чтобы разобраться создадим еще 1 snapshot и запромоутим его

NAME	USED	AVAIL	REFER	MOUNTPOINT
origin	3.92G	189G	26K	/origin
origin/postgres	3.63G	189G	3.63G	/origin/postgres
origin/postgres_snap1	256M	189G	2.83G	/origin/postgres_snap1
origin/postgres_snap2	16.6M	189G	2.83G	/origin/postgres_snap2

Получим путь таблицы в файловой системе

```
root@zfstest:/docker-pg# psql -p 5433 -U postgres -h localhost -d demo -c "SELECT
pg_relation_filepath ('boarding_passes')"
  pg_relation_filepath
------------------------------
  base/25293/25316
(1 row)
```

ANALYZE

Воспользуемся утилитой bindiff чтобы понять размер различных блоков в этом бинарном файле
https://www.dr-lex.be/software/bindiff.html

```
./bindiff –b 8192 -t /origin/postgres_snap2/base/25293/25316
/origin/postgres_snap1/base/25293/25316
```

Где –b block size в zfs pool или zvol

```
Total size of differing blocks: 245760000B
```

Собственно получаем логичную картину.
Но что же поменялось в самих страницах при выполнении **ANALYZE**

ANALYZE

Воспользуемся утилитой bindiff чтобы понять размер различных блоков в этом бинарном файле
https://www.dr-lex.be/software/bindiff.html

```
./bindiff –b 8192 -t /origin/postgres_snap2/base/25293/25316
/origin/postgres_snap1/base/25293/25316
```

Где –b block size в zfs pool или zvol

```
Total size of differing blocks: 245760000B
```

Собственно получаем логичную картину.
Но что же поменялось в самих страницах при выполнении **ANALYZE**

```
cmp -1 /origin/postgres_snap2/base/25293/25316 /origin/postgres_snap1/base/25293/25316 |
gawk '(printf "408X %02X %02X\n", $1-1, strtonum(0S2), strtonum (0$3)}'
```

```
1C74BEC5 08 09
1C74BEFD 08 09
1C748F35 08 09
1C74BF6D 08 09
1C74DFDD 08 09
```

Поменялось по 1байту в определенных позициях файла.
До сих пор непонятно что меняет **ANALYZE** в самих страницах.
Нам поможет расширение pageinspect

ANALYZE

Запрос на snapshot1 с ANALYZE

```
demo=# SELECT t_infomask, t_infomask2, raw_flags, combined_flags
FROM heap_page_items (get_raw_page( 'boarding_passes', 0)),
LATERAL heap_tuple_infomask_flags(t_infomask, t_infomask2) limit 10;
```

```
t_infomask | t_infomask 2 |                          raw_flags                                  | combined_flags |
-----------+--------------+---------------------------------------------------------------------+----------------+
2306       | 4            |{HEAP_HASVARWIDTH, HEAP_XMIN_COMMITTED, HEAP _XMAX_INVALID} | {}             |
2306       | 4            |{HEAP_HASVARWIDTH, HEAP_XMIN_COMMITTED, HEAP _XMAX_INVALID} | {}             |
2306       | 4            |{HEAP_HASVARWIDTH, HEAP_XMIN_COMMITTED, HEAP _XMAX_INVALID} | {}             |
```

Запрос на snapshot2 без ANALYZE

```
t_infomask | t_infomask 2 |                raw_flags                | combined_flags |
-----------+--------------+-----------------------------------------+----------------+
2050       | 4            |{HEAP_HASVARWIDTH, HEAP _XMAX_INVALID} | {}             |
2050       | 4            |{HEAP_HASVARWIDTH, HEAP _XMAX_INVALID} | {}             |
2050       | 4            |{HEAP_HASVARWIDTH, HEAP _XMAX_INVALID} | {}             |
```

ANALYZE

Запрос на snapshot1 с ANALYZE

```
demo=# SELECT t_infomask, t_infomask2, raw_flags, combined_flags
FROM heap_page_items (get_raw_page( 'boarding_passes', 0)),
LATERAL heap_tuple_infomask_flags(t_infomask, t_infomask2) limit 10;
```

```
t_infomask | t_infomask 2 |                          raw_flags                                  | combined_flags |
-----------+--------------+---------------------------------------------------------------------+----------------+
2306       | 4            |{HEAP_HASVARWIDTH, HEAP_XMIN_COMMITTED, HEAP _XMAX_INVALID} | {}             |
2306       | 4            |{HEAP_HASVARWIDTH, HEAP_XMIN_COMMITTED, HEAP _XMAX_INVALID} | {}             |
2306       | 4            |{HEAP_HASVARWIDTH, HEAP_XMIN_COMMITTED, HEAP _XMAX_INVALID} | {}             |
```

Запрос на snapshot2 без ANALYZE

```
t_infomask | t_infomask 2 |                raw_flags                | combined_flags |
-----------+--------------+-----------------------------------------+----------------+
2050       | 4            |{HEAP_HASVARWIDTH, HEAP _XMAX_INVALID} | {}             |
2050       | 4            |{HEAP_HASVARWIDTH, HEAP _XMAX_INVALID} | {}             |
2050       | 4            |{HEAP_HASVARWIDTH, HEAP _XMAX_INVALID} | {}             |
```

postgres/src/include/access/**htup_details.h**

```
204     #define HEAP_XMIN_COMMITTED     0x0100  /* t_xmin committed */
205     #define HEAP_XMIN_INVALID       0x0200  /* t_xmin invalid/aborted */
206     #define HEAP_XMIN_FROZEN        (HEAP_XMIN_COMMITTED|HEAP_XMIN_INVALID)
207     #define HEAP_XMAX_COMMITTED     0x0400  /* t_xmax committed */
```

VACUUM

Исходный размер

```
NAME                    USED    AVAIL   REFER   MOUNTPOINT
origin                  3.92G   189G    26K     /origin
origin/postgres         3.63G   189G    3.63G   /origin/postgres
origin/postgres_snap1    370M   189G    2.83G   /origin/postgres_snap1
origin/postgres_snap2   16.6M   189G     2.83G  /origin/postgres_snap2
```

```
DELETE FROM boarding_passes;
```

Размер после удаления

```
NAME                    USED    AVAIL   REFER   MOUNTPOINT
origin                  3.92G   189G    26K     /origin
origin/postgres         3.63G   189G    3.63G   /origin/postgres
origin/postgres_snap1   1.35G   189G    2.83G   /origin/postgres_snap1
origin/postgres_snap2   16.6M   189G     2.83G  /origin/postgres_snap2
```

Размер после VACUUM

```
NAME                    USED    AVAIL   REFER   MOUNTPOINT
origin                  3.92G   189G    26K     /origin
origin/postgres         3.63G   189G    3.63G   /origin/postgres
origin/postgres_snap1   1.16G   189G    3.20G   /origin/postgres_snap1
origin/postgres_snap2   16.6M   189G     2.83G  /origin/postgres_snap2
```

VACUUM

Исходный размер

```
NAME                    USED        AVAIL       REFER    MOUNTPOINT
origin                  3.92G       189G        26K      /origin
origin/postgres         3.63G       189G        3.63G    /origin/postgres
origin/postgres_snap1    370M       189G        2.83G    /origin/postgres_snap1
origin/postgres_snap2   16.6M       189G         2.83G   /origin/postgres_snap2
```

```
DELETE FROM boarding_passes;
```

Размер после удаления

```
NAME                    USED        AVAIL       REFER    MOUNTPOINT
origin                  3.92G       189G        26K      /origin
origin/postgres         3.63G       189G        3.63G    /origin/postgres
origin/postgres_snap1   1.35G       189G        2.83G    /origin/postgres_snap1
origin/postgres_snap2   16.6M       189G         2.83G   /origin/postgres_snap2
```

Размер после VACUUM

```
NAME                    USED        AVAIL       REFER    MOUNTPOINT
origin                  3.92G       189G        26K      /origin
origin/postgres         3.63G       189G        3.63G    /origin/postgres
origin/postgres_snap1   1.16G       189G        3.20G    /origin/postgres_snap1
origin/postgres_snap2   16.6M       189G         2.83G   /origin/postgres_snap2
```

VACUUM

Исходный размер

```
NAME                    USED    AVAIL   REFER   MOUNTPOINT
origin                  3.92G   189G    26K     /origin
origin/postgres         3.63G   189G    3.63G   /origin/postgres
origin/postgres_snap1    370M   189G    2.83G   /origin/postgres_snap1
origin/postgres_snap2   16.6M   189G     2.83G  /origin/postgres_snap2
```

```
DELETE FROM boarding_passes;
```

Размер после удаления

```
NAME                    USED    AVAIL   REFER   MOUNTPOINT
origin                  3.92G   189G    26K     /origin
origin/postgres         3.63G   189G    3.63G   /origin/postgres
origin/postgres_snap1   1.35G   189G    2.83G   /origin/postgres_snap1
origin/postgres_snap2   16.6M   189G     2.83G  /origin/postgres_snap2
```

Размер после VACUUM

```
NAME                    USED    AVAIL   REFER   MOUNTPOINT
origin                  3.92G   189G    26K     /origin
origin/postgres         3.63G   189G    3.63G   /origin/postgres
origin/postgres_snap1   1.16G   189G    3.20G   /origin/postgres_snap1
origin/postgres_snap2   16.6M   189G     2.83G  /origin/postgres_snap2
```

VACUUM

```
root@zfstest:/docker-pg# ./bindiff -b 8192 -t /origin/postgres_snap2/base/25293/25316
/origin/postgres_snap1/base/25293/25316
File 1 is 477421568 bytes longer than file 2.
Total size of differing blocks: 0
```

At the end of the war the sunshine begins

- При частом удалении и изменении данных надо закладывать x2 физического места на диске
- Реплика не влияет на основной ZVOL
- При изменении в снапшоте все блоки данных, которые зависят будут скопированны в снапшот
- У ANALYZE есть нюансы после promote
- Применимость системы

Ссылки на источники информации и программные продукты

- https://www.dr-lex.be/software/bindiff.html
- https://habr.com/ru/companies/vk/articles/529516/
- https://www.linux.org.ru/articles/admin/17333839
- https://github.com/openzfs/
- https://www.docker.com
- https://github.com/postgres/postgres

Спасибо за внимание!

Q&A

Мои контакты

- Telegram - @moytra_nihzayvs
- Email – svyazhin_a@sima-land.ru