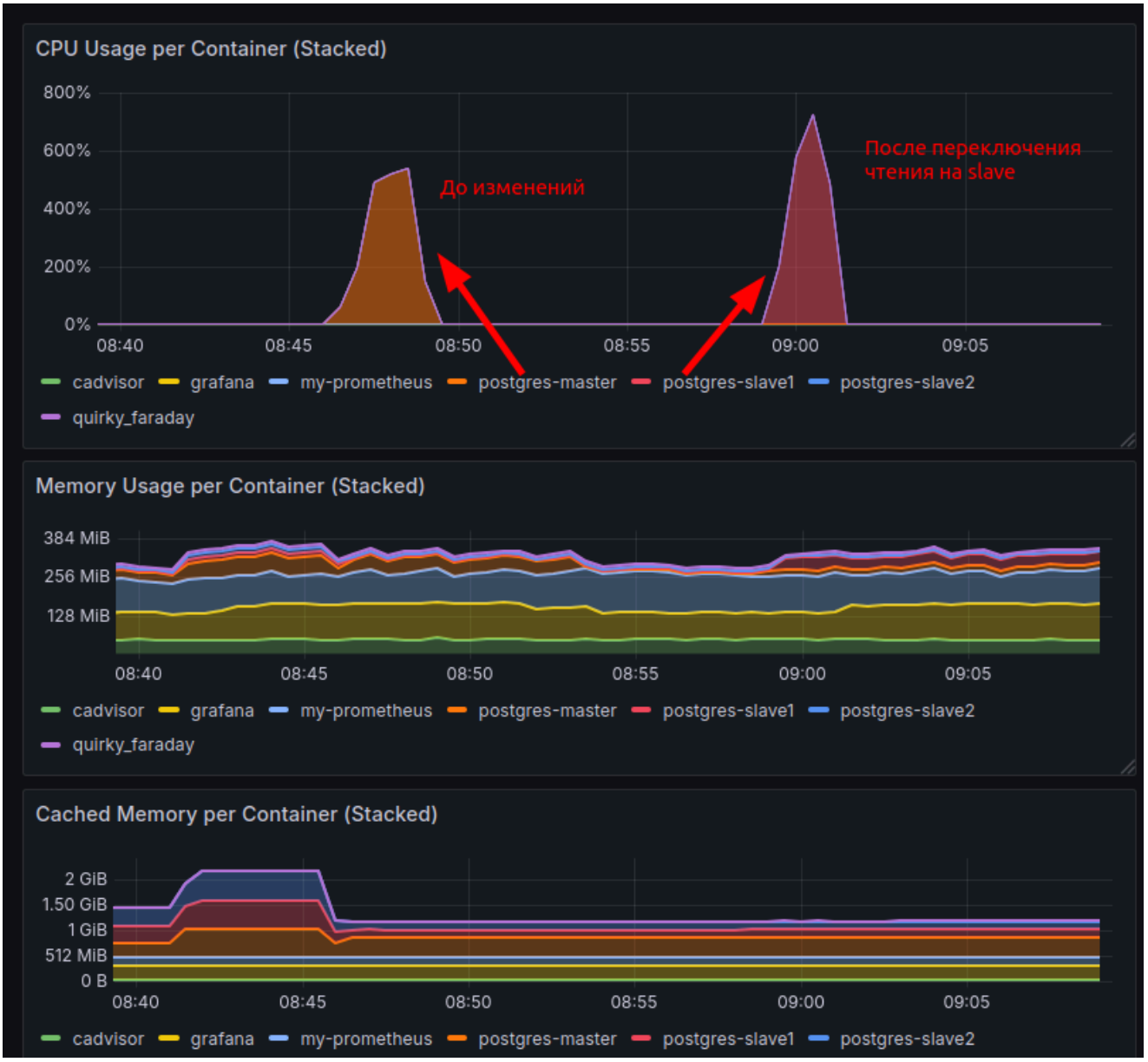
Задание 3. Репликация

1. Составлен план нагрузочного тестирования исходя из двух запросов на чтение (/user/get/{id} и /user/search из спецификации).
2. Создана нагрузка на чтение с помощью составленного на предыдущем шаге плана, делаем замеры. См. левую часть графика на картинке ниже.
3. Настроена потоковая асинхронная репликация 2 слейва и 1 мастер. Скрипты настройки и docker-compose доступны в github.

```
hl1=# SELECT application_name, client_addr, state, sync_state FROM pg_stat_replication;
 application_name | client_addr |   state   | sync_state
------------------+-------------+-----------+------------
 postgres-slave2  | 172.19.0.4  | streaming | async
 postgres-slave1  | 172.19.0.3  | streaming | async
(2 rows)
```

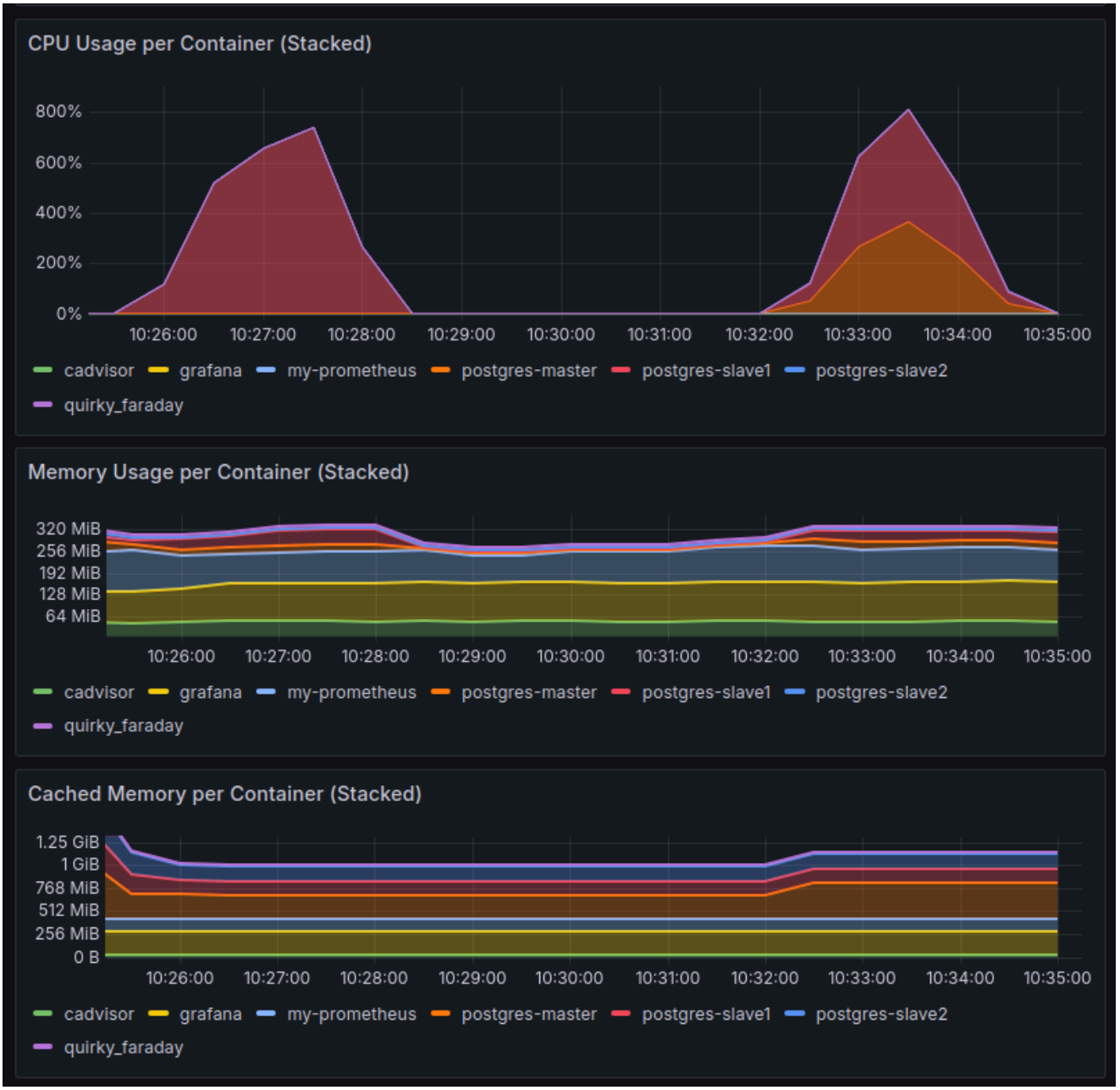
4. Добавлен в проект Replication DataSource: kwon37xi/replication-datasource-boot, настроено чтение на slave1.
5. Создана нагрузка как в п.2. См. правую часть графика на картинке ниже.

Картинка 1. Тесты на master и slave



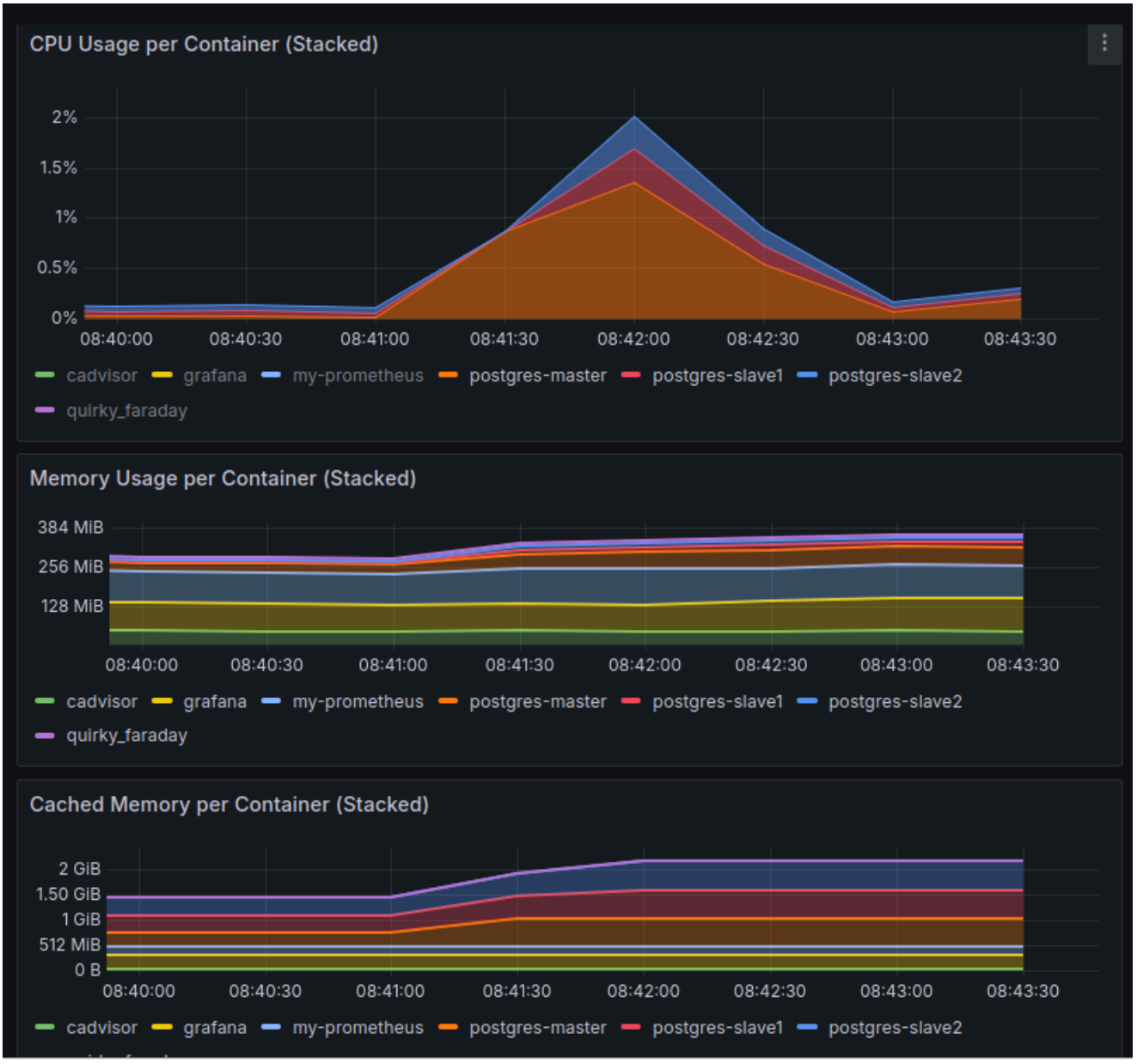
Ниже показано распределение нагрузки, если у одного запроса выставить readOnly = true, а у второго - readOnly = false. В левой части оба - readOnly = true, в правой - один readOnly = true, а у второго - readOnly = false.

Картинка 1.2. Один запрос на master, второй - на slave.



Картинка 2. Момент синхронизации при заливке первичной базы с 1 000 000 записей пользователей. Видно основную работу master и работу slave1 и slave2, но их работа в

разы меньше, чем у master.



6. Настроить кворумную синхронную репликацию.

```
alter system set synchronous_standby_names='ANY 1 ("postgres-slave1",
"postgres-slave2")';
```

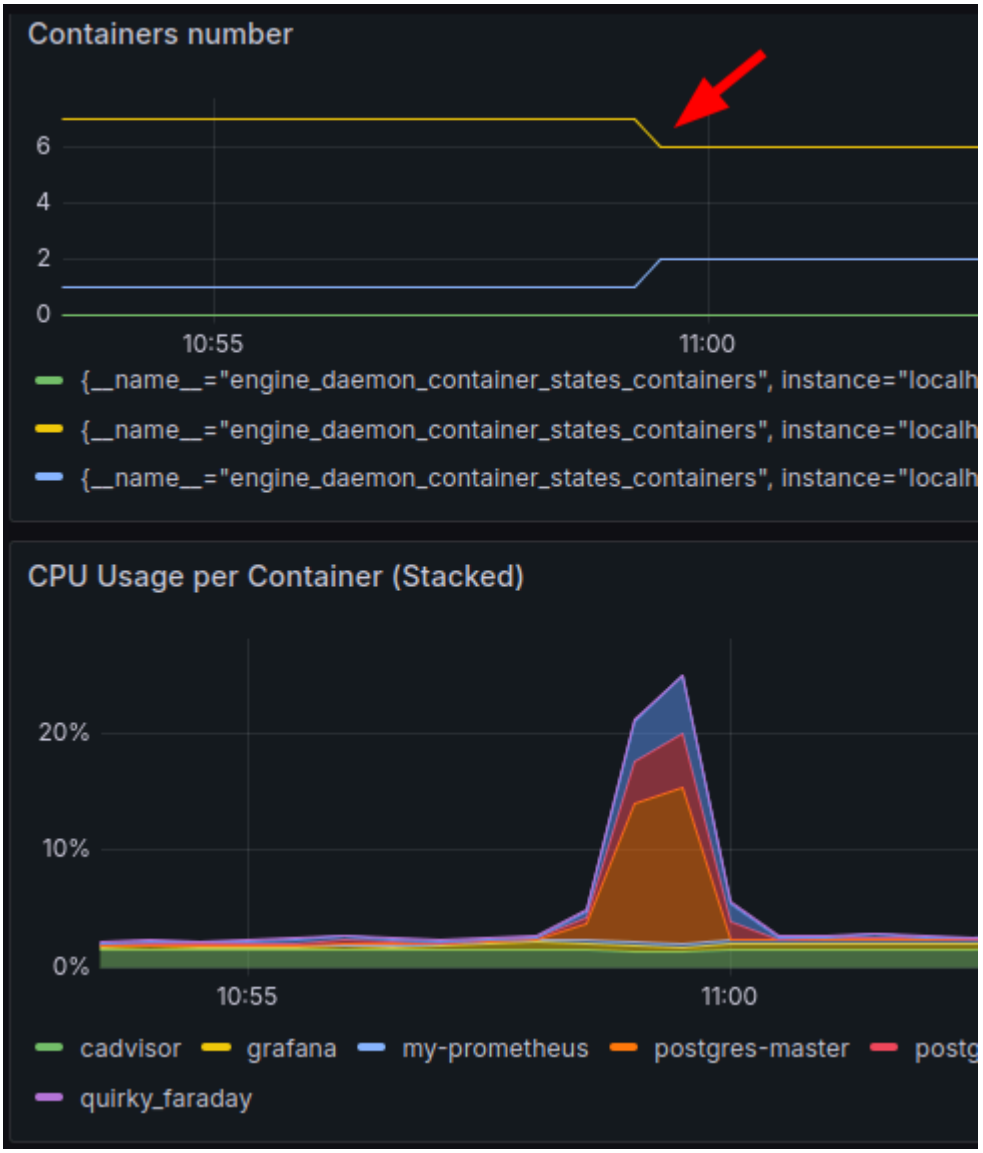
```
hl1=# alter system set synchronous_standby_names='ANY 1 ("postgres-slave1", "postgres-slave2")';
ALTER SYSTEM
hl1=# select pg_reload_conf();
 pg_reload_conf
----------------
 t
(1 row)

hl1=# SELECT application_name, client_addr, state, sync_state FROM pg_stat_replication;
 application_name | client_addr |   state   | sync_state
------------------+-------------+-----------+------------
 postgres-slave2  | 172.19.0.4  | streaming | quorum
 postgres-slave1  | 172.19.0.3  | streaming | quorum
(2 rows)
```

7. Создать нагрузку на запись в любую тестовую таблицу.

8. Убиваем мастер узел.

Картинка 3. Отключение master.



9. Заканчиваем нагрузку на запись.
10. Выбираем самый свежий слейв. Промоутим его до мастера. Переключаем на него второй слейв.

На slave1 выполняем:

```
select pg_promote();

alter system set synchronous_standby_names='ANY 1 ("postgres-slave2")';
select pg_reload_conf();
```

На slave2 меняем конфиг и перезапускаем:

```
primary_conninfo = 'host=postgres-slave1 port=5432 user=replicator
password=replicator123 application_name=postgres-slave2'
```

Проверяем число пользователей в таблице на slave1:

```
hl1=# select count(*) from public.user;
  count
---------
 1052340
```

Проверяем число пользователей в таблице на slave2:

```
hl1=# select count(*) from public.user;
  count
---------
 1052340
```

Останавливаем оба slave1 и slave2, запускаем master и смотрим, сколько транзакций было изначально записано в нем:

```
hl1=# select count(*) from public.user;
  count
---------
 1052340
```

Вывод: потери транзакций при синхронной синхронизации нет.

Проверяем, работает ли новый слейв и новый мастер.
Снова останавливаем старый master.
Запускаем slave1 и slave2.

Добавляем запись в slave1.

```
hl1=# insert into "user" values ('user3', 'FIRST3', 'SECOND3', '1978-05-
04', 'SEX2', 'BIO2', 'MSK2');
INSERT 0 1
hl1=# select count(*) from public.user;
  count
---------
 1052341
```

Проверяем, что запись добавилась в slave2

```
hl1=# select count(*) from public.user;
  count
---------
 1052341
(1 row)
```

```
hl1=# select * from public.user where id='user3';
  id   | first_name | second_name | birthdate  | sex  | biography | city
-------+------------+-------------+------------+------+-----------+------
 user3 | FIRST3     | SECOND3     | 1978-05-04 | SEX2 | BIO2      | MSK2
(1 row)
```

Вывод: переключение прошло успешно.