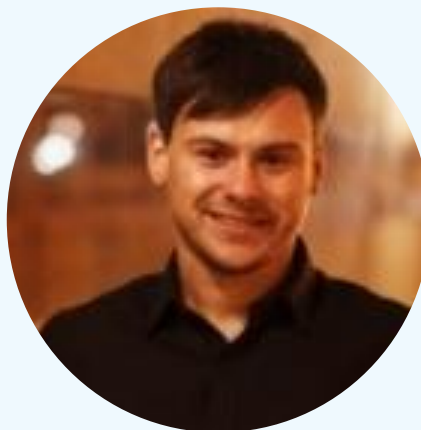


Реляционные базы данных: Репликация баз данных.



Сергей
Андрюнин



Сергей Андрюнин

DevOps инженер

RTLabs

Предисловие

Сегодня мы:

- поговорим о **репликации** баз данных;
- разберем, какие **методы и технологии репликации** существуют, как они работают;
- настроим репликацию **master-slave**.





План занятия

1. [Репликация](#)
2. [Репликация master-slave](#)
3. [Репликация master-master](#)
4. [Настройка master-slave репликации](#)
5. [Настройка master-master репликации](#)
6. [Тестирование режима работы](#)
7. [Итоги](#)
8. [Домашнее задание](#)



Репликация



Репликация

Репликация — это процесс, под которым понимается копирование данных из одного источника на другой (или на множество других) и наоборот.

С точки зрения базы данных — это механизм копирования базы данных и создания копий или дополнений существующих объектов.

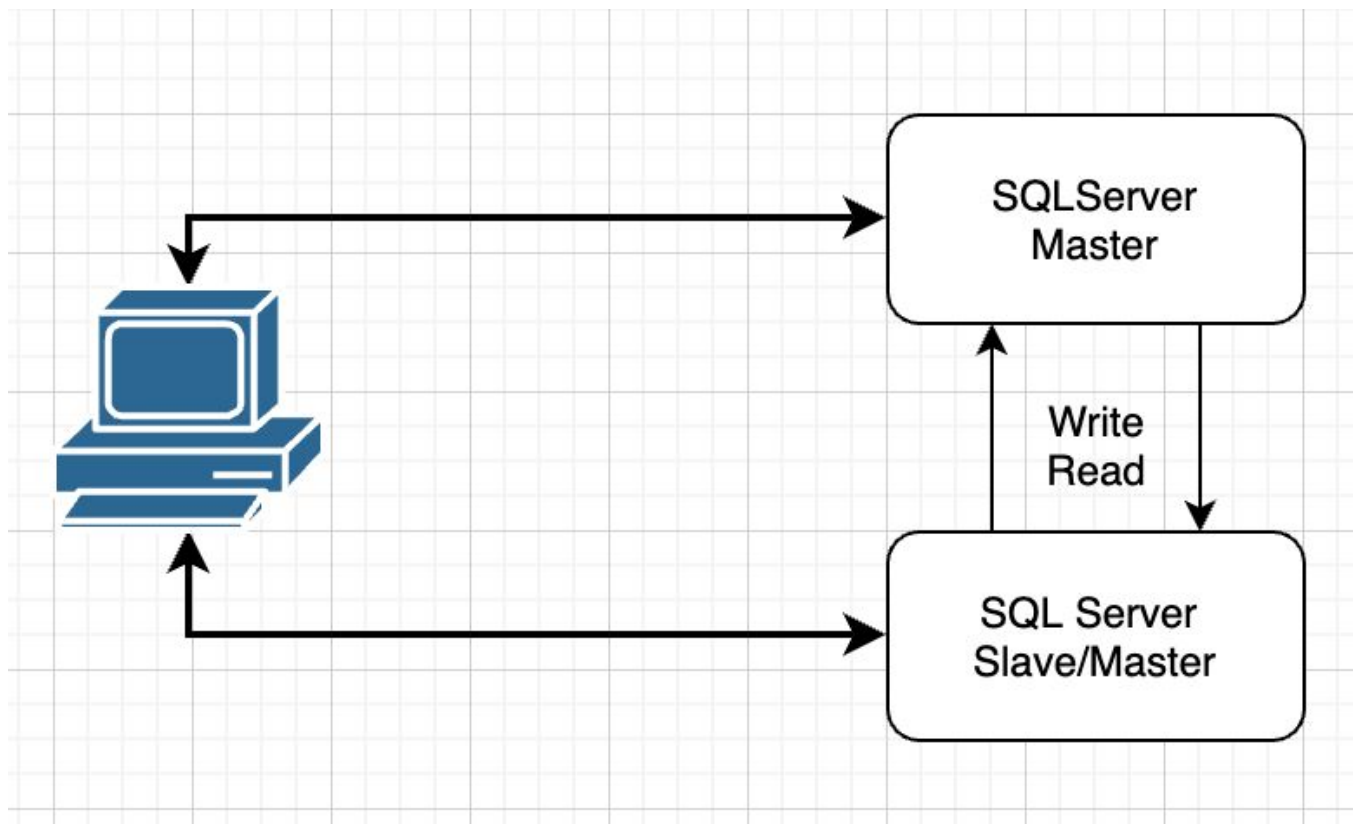
Виды репликации

- **Синхронная**, если данная реплика обновляется, все другие реплики того же фрагмента данных также должны быть обновлены в одной и той же транзакции. Логически это означает, что существует лишь одна версия данных.
- **Асинхронная** — обновление одной реплики распространяется на другие спустя некоторое время, а не в той же транзакции. Таким образом, при асинхронной репликации вводится задержка или время ожидания, в течение которого отдельные реплики могут быть фактически не идентичными.

Преимущества репликации

- повышение производительности чтения данных;
- повышение отказоустойчивости;
- распространение данных между серверами повышает надежность, доступность и скорость;
- распределение нагрузки;
- тестирование новых конфигураций;
- резервное копирование.

Общая схема репликации





Репликация master-slave



Репликация master-slave

Репликация типа Master-Slave часто используется для обеспечения отказоустойчивости приложений.

Кроме этого, она позволяет распределить нагрузку на базу данных между несколькими серверами (репликами).

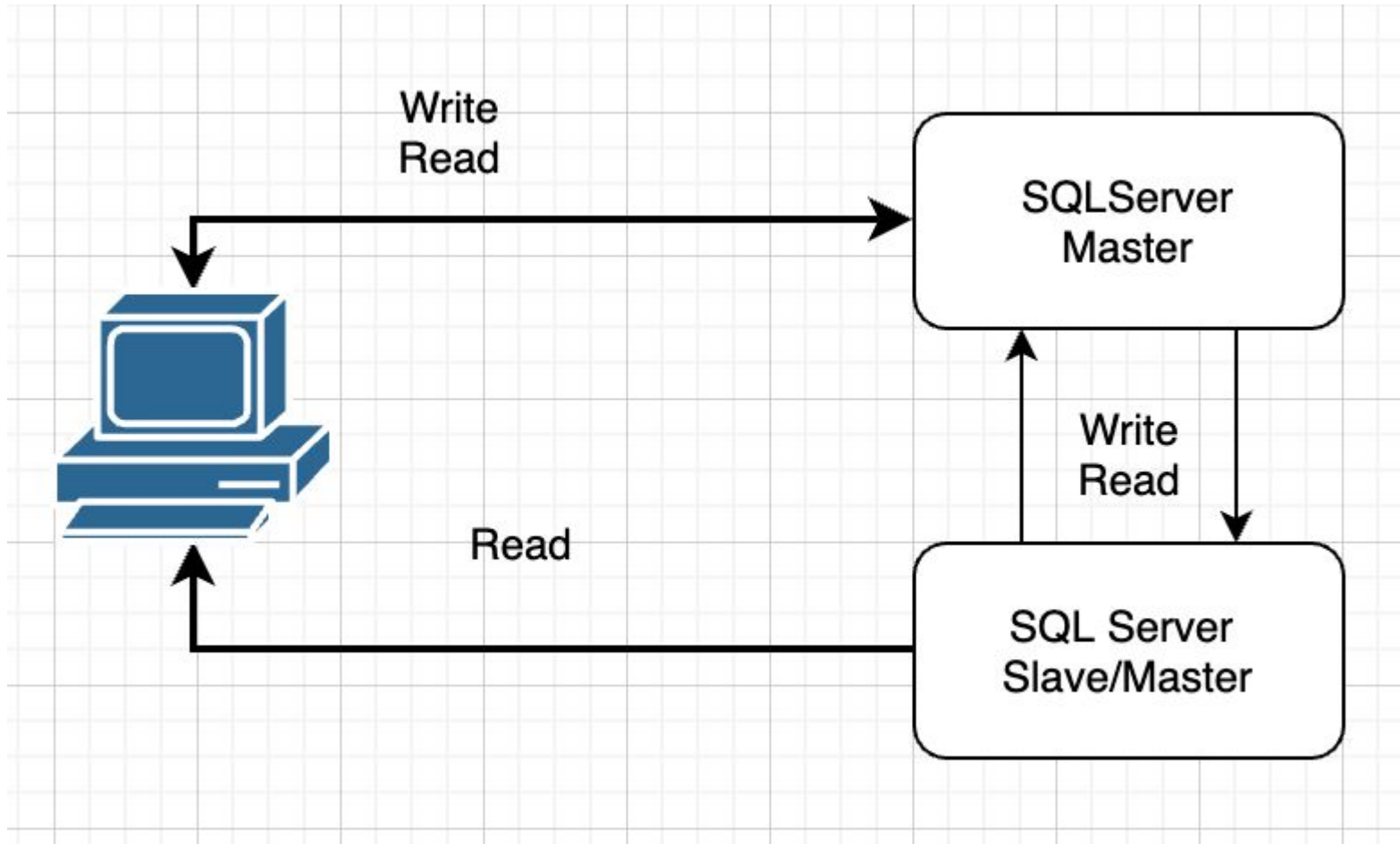


Репликация master-slave

Master — это основной сервер БД, куда поступают все данные. Все изменения в данных (добавление, обновление, удаление) должны происходить на этом сервере.

Slave — это вспомогательный сервер БД, который копирует все данные с мастера. С этого сервера следует читать данные. Таких серверов может быть несколько.

Репликация master-slave



Репликация master-slave

Репликация выполняется по следующим шагам:

- **Master** записывает изменения данных в журнал (**binary log**);
- **Slave** копирует изменения двоичного журнала в свой, который называется журналом ретрансляции (**relay log**);
- **Slave** воспроизводит изменения из журнала ретрансляции, применяя их к собственным данным.



Репликация master-slave

Репликация бывает двух подходов: **покомандная** и **потокковая**.

Покомандная репликация — в журнал **master** протоколируются запросы изменения данных (INSERT, UPDATE, DELETE), на **slave** повторяются.

Построчной репликации в журнале запишутся изменения данных, тоже произойдет и на **slave**.



Репликация master-master

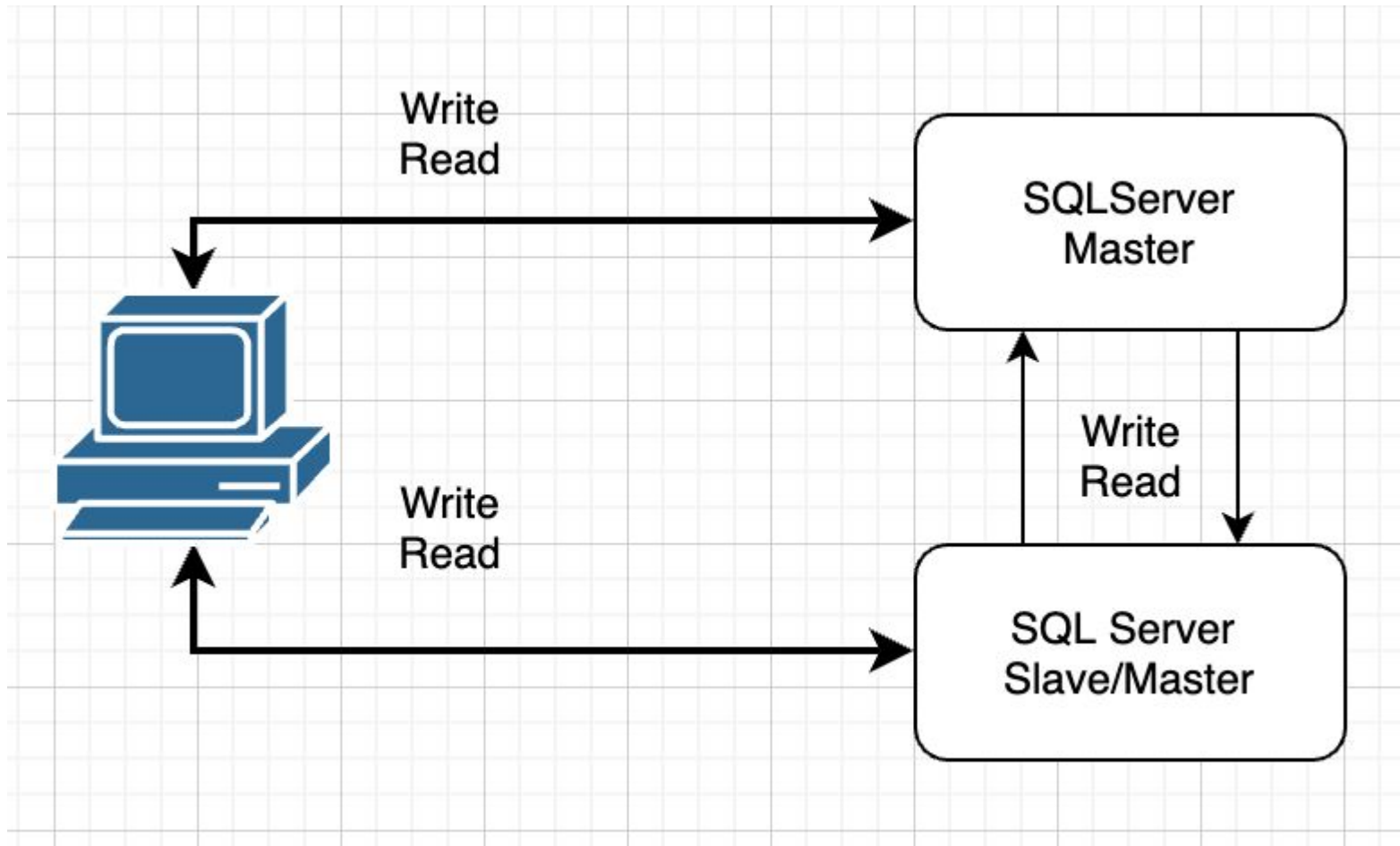



Репликация Master-Master

Репликация master-master позволяет копировать данные с одного сервера на другой. Эта конфигурация добавляет избыточность и повышает эффективность при обращении к данным.

Master-Master репликации – это настройка обычной Master-Slave репликации, только в обе стороны (каждый сервер является мастером и слейвом одновременно).

Репликация Master-Master

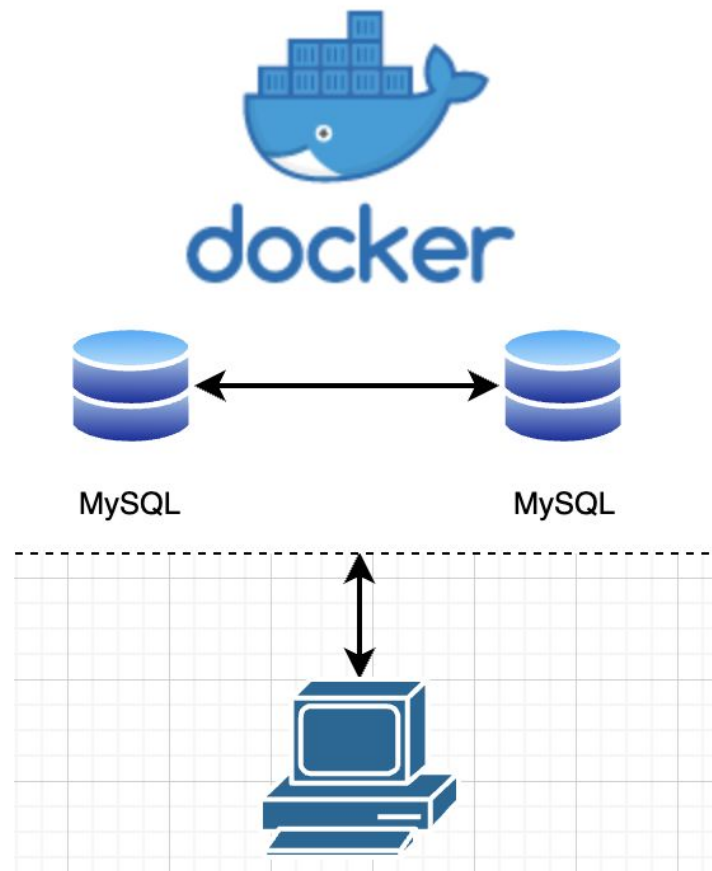




Настройка master-slave репликации

Подготовка модели

Для работы требуется
установить **docker**.



Установка контейнеров mysql

Установка master:

```
docker run -d --name replication-master -e MYSQL_ALLOW_EMPTY_PASSWORD=true -v  
~/path/to/world/dump:/docker-entrypoint-initdb.d mysql:8.0
```

Установка slave:

```
docker run -d --name replication-slave -e MYSQL_ALLOW_EMPTY_PASSWORD=true  
mysql:8.0
```

Для реализации взаимодействия создадим мост и сеть:

```
docker network create replication  
docker network connect replication replication-master  
docker network connect replication replication-slave
```

Настройка компонентов

Для управления реализацией настроек обновим и установим в контейнеры инструменты.

Для Master:

```
docker exec replication-master apt-get update && docker exec replication-master  
apt-get install -y nano
```

Для Slave:

```
docker exec replication-slave apt-get update && docker exec replication-slave  
apt-get install -y nano
```

Настройка компонентов

Создадим учетную запись Master для сервера репликации:

```
docker exec -it replication-master mysql
```

В контейнере выполним:

```
mysql> CREATE USER 'replication'@'%';  
mysql> GRANT REPLICATION SLAVE ON *.* TO 'replication'@'%';
```

Изменим конфигурацию сервера:

```
docker exec -it replication-master bash  
~ nano /etc/mysql/my.cnf
```

Настройка компонентов

my.cnf -> секция [mysqld] добавляем следующие параметры:

```
server_id = 1  
log_bin = mysql-bin
```

При изменении конфигурации сервера требуется перезагрузка:

```
docker restart replication-master
```

После требуется зайти в контейнер и проверить состояние:

```
docker exec -it replication-master mysql  
mysql> SHOW MASTER STATUS;
```


Настройка компонентов

Следующим шагом требуется выполнить слепок системы и заблокировать все изменения на сервер:

```
mysql> FLUSH TABLES WITH READ LOCK;
```

После данных манипуляций выхода из контейнера и выполняем процесс mysqldump для экспорта базы данных, например:

```
docker exec replication-master mysqldump world > /path/to/dump/on/host/world.sql
```

После, следует зайти обратно в контейнер и вывести настройки master сервера (они понадобятся при настройке slave):

```
docker exec -it replication-master mysql  
mysql> SHOW MASTER STATUS;
```

!!! Важно запомнить File и Position

Настройка компонентов

Снимаем блокировку базы данных:

```
mysql> UNLOCK TABLES;
```

Master готов, переходим к slave:

```
docker cp /path/to/dump/on/host/world.sql replication-slave:/tmp/world.sql
docker exec -it replication-slave mysql
mysql> CREATE DATABASE `world`;
docker exec -it replication-slave bash
~ mysql world < /tmp/world.sql
```

Открываем конфигурационный файл на Slave my.cnf:

```
docker exec -it replication-slave bash
~ nano /etc/mysql/my.cnf
```

Настройка компонентов

my.cnf -> секция [mysqld] добавляем следующие параметры:

```
log_bin = mysql-bin  
server_id = 2  
relay-log = /var/lib/mysql/mysql-relay-bin  
relay-log-index = /var/lib/mysql/mysql-relay-bin.index  
read_only = 1
```

Перезагружаем Slave:

```
docker restart replication-slave
```

Настройка компонентов

Следующим шагом требуется прописать в базе данных на сервер slave, кто является master и данные полученные в File и Position:

```
docker exec -it replication-slave mysql
```

```
mysql> CHANGE MASTER TO MASTER_HOST='replication-master',  
MASTER_USER='replication', MASTER_LOG_FILE='mysql-bin.000001',  
MASTER_LOG_POS=156;
```

Далее запускаем журнал ретрансляции, и проверим статус операций:

```
mysql> START SLAVE;  
mysql> SHOW SLAVE STATUS\G
```

Ключевые настройки

- **Slave_IO_State, Slave_SQL_State** — состояние IO потока, принимающего двоичный журнал с мастера, и состояние потока, применяющего журнал ретрансляции.
- **Read_Master_Log_Pos** — последняя позиция, прочитанная из журнала мастера.
- **Relay_Master_Log_File** — текущий файл журнала мастера.
- **Seconds_Behind_Master** — отставание данных в секундах.
- **Last_IO_Error, Last_SQL_Error** — ошибки репликации, если они есть.



Настройка master-master репликации

Тестирование

Режим Master-Master:

```
docker run -d --name replication-master-one -e MYSQL_ALLOW_EMPTY_PASSWORD=true  
-v ~/path/to/world/dump:/docker-entrypoint-initdb.d mysql:8.0  
docker run -d --name replication-master-two -e MYSQL_ALLOW_EMPTY_PASSWORD=true  
-v ~/path/to/world/dump:/docker-entrypoint-initdb.d mysql:8.0
```

Конфигурационный файл my.conf должен содержать:

```
server_id = 1  
log_bin = mysql-bin #на первом  
server_id = 2  
log_bin = mysql-bin #на втором
```

Тестирование

На втором сервере выполнить:

```
slave stop;  
CHANGE MASTER TO MASTER_HOST = 'replication-master-one', MASTER_USER =  
'replicator', MASTER_PASSWORD = 'password', MASTER_LOG_FILE =  
'mysql-bin.000001', MASTER_LOG_POS = 107;  
slave start;
```

На первом:


```
slave stop;  
CHANGE MASTER TO MASTER_HOST = 'replication-master-two', MASTER_USER =  
'replicator', MASTER_PASSWORD = 'password', MASTER_LOG_FILE =  
'mysql-bin.000001', MASTER_LOG_POS = 107;  
slave start;
```


Тестирование

Важно добавить пользователей на обоих серверах:

```
create user 'replicator'@'%' identified by 'password';  
create database example;  
grant replication slave on *.* to 'replicator'@'%';
```

Перезагружаем и проверяем.



Тестирование режима работы

Тестирование

Меняем данные на Server-Master:

```
docker exec -it replication-master mysql  
  
mysql> USE world;  
mysql> INSERT INTO city (Name, CountryCode, District, Population) VALUES  
( 'Test-Replication', 'ALB', 'Test', 42);
```

Проверяем на slave:

```
docker exec -it replication-slave mysql  
  
mysql> USE world;  
mysql> SELECT * FROM city ORDER BY ID DESC LIMIT 1;
```



Итоги

Итоги

Сегодня мы:

- рассмотрели принципы репликации,
- настроили режимы работы master-slave и протестировали его работу,
- рассмотрели master-master режим.





Домашнее задание



Домашнее задание

Давайте посмотрим ваше [домашнее задание](#).

- Вопросы по домашней работе задавайте **в чате** мессенджера .
- Задачи можно сдавать **по частям**.
- Зачёт по домашней работе проставляется после того, как **приняты все задачи**.

**Задавайте вопросы и
пишите отзыв о лекции!**

Сергей Андрюнин