

# Troubleshooting



Роман  
Гордиенко



**Роман Гордиенко**

Backend Developer, Factory5



[Роман Гордиенко](#)

---

# План занятия

1. [Введение](#)
2. [MongoDB](#)
3. [Redis](#)
4. [MySQL](#)
5. [PostgreSQL](#)
6. [Elasticsearch](#)
7. [Итоги](#)
8. [Домашнее задание](#)



# Введение



# Введение

**Troubleshooting** - это систематический, опосредованный определённой логикой поиск источника проблемы с целью её решения.

Troubleshooting как поиск и устранение неисправностей необходим для поддержания и развития сложных систем, где проблема может иметь множество различных причин.

---

# Введение

**Troubleshooting БД подразделяется на:**

- устранение проблем (непосредственный troubleshooting)
- performance tuning

**Поиск и устранение ошибок можно разделить на следующие этапы:**

- чтение логов
- профилирование
- анализ полученных данных
- устранение неисправности
- тестирование правок

---

# Введение

**В БД чаще всего ошибки появляются на следующих этапах:**

- установка дистрибутива (invalid installation);
- конфигурация сервера (invalid configuration);
- настройка пользователей (invalid user policy);
- безопасность (invalid security policy);
- сетевые сбои (network failures).

---

# Введение

**Ошибки при установке дистрибутива** чаще всего возникают при:

- сбоях пакетного менеджера;
- отсутствии каких-либо прав пользователя на сервере;
- нехватки системных ресурсов;
- “битых” дистрибутивов.

Описание решений данного вида проблем чаще всего приведено в документации на приложение, в разделах Installation или Administration.



---

# Введение

**Ошибки конфигурации** - это довольно большой слой. Примеры типичных ошибок конфигурации:

- JVM GC некорректно обрабатывает данные;
- Master-узел не может найти Slave-узлы;
- Данные хранятся не в нужном разделе;
- Сервер БД пытается принимать входящие соединения на занятом порту;
- Превышено количество соединений на сервер БД.

Нужно быть предельно внимательным при настройке сервера БД.

---

# Введение

**Ошибки настроек пользователей и security** можно объединить в единый слой. К таким ошибкам относятся:

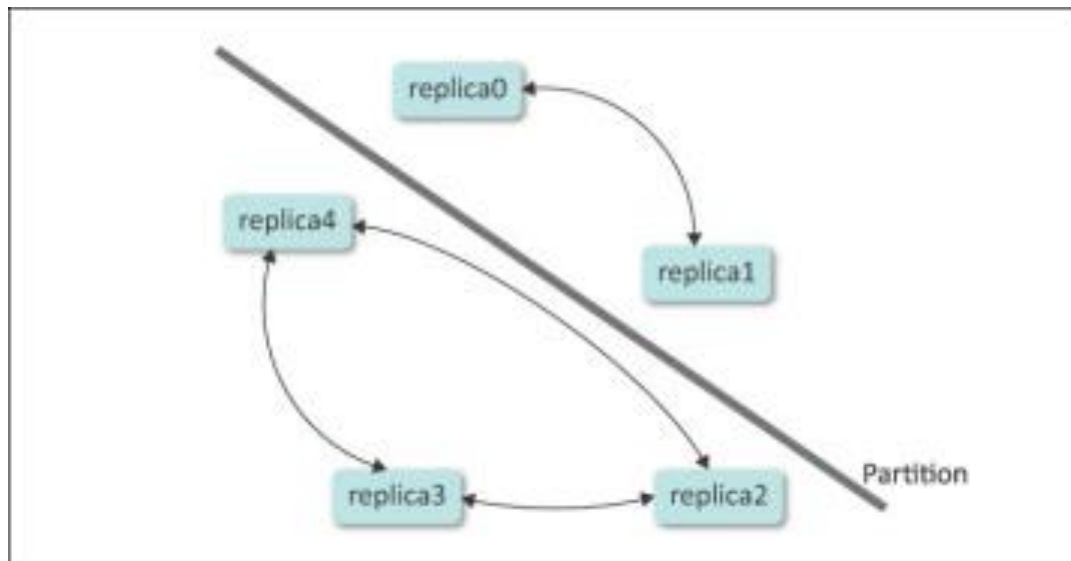
- несанкционированный доступ пользователей до баз данных и таблиц;
- возможность проводить CUD над данными (например в production слоях);
- разрешение на выполнение запросов, требующих множество вычислительных ресурсов (например в SQL - LIKE запросы).

# Введение

**Сетевые сбои** чаще всего происходят внезапно.

Этот тип сбоев требуется наиболее внимательно отслеживать.

Например, в СА-системах это может привести к разбиению одного кластера на два.





# MongoDB

---

# MongoDB

MongoDB имеет достаточно подробную [документацию](#), в которой описаны:

- требования для развертки сервера;
- профилирование работы;
- best-practice для конфигурирования;
- методы бэкапа и восстановления данных;
- проведение мониторинга.

---

# MongoDB

Основные метрики мониторинга MongoDB:

- время исполнения операций;
- количество операций.

Данные метрики контролируются, посредством средств мониторинга, поставляемых с MongoDB, либо сторонними приложениями для мониторинга.

# MongoDB

**Free monitoring** - поставляемый вместе с экземпляром MongoDB облачный инструмент мониторинга состояния БД.

Для включения Free monitoring необходимо выполнить следующую команду в **mongo-shell**:


```
db.enableFreeMonitoring()
```

Пример ответа на данный вызов:

```
{
  "state" : "enabled",
  "message" : "To see your monitoring data, navigate to the unique URL below.\nAnyone yo",
  "url" : "https://cloud.mongodb.com/freemonitoring/mongo/MSBjZTZhNTJmOS0yODg1",
  "userReminder" : "",
  "ok" : 1
}
```

# MongoDB

Free Monitoring

Powered by  MongoDB Atlas

5b7412725cf6

STANDALONE

VERSION  
4.4.1

[Helpful Information](#)

GRANULARITY: 1 MINUTE CHART TIME RANGE: 1 DAY

System CPU  
Usage ⓘ

73.28%

Read Operation Execution  
Time ⓘ

0.00 ms

Query  
Targeting ⓘ

0.00

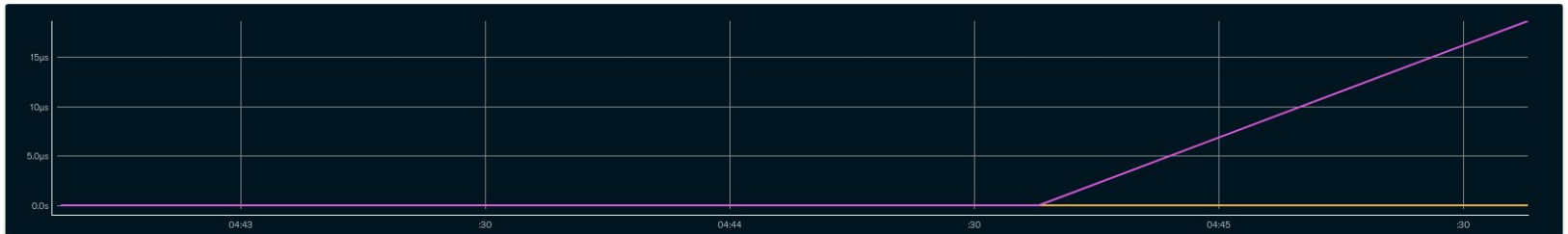
Read  
Operations/Second ⓘ

0.00/s

5b7412725cf6:

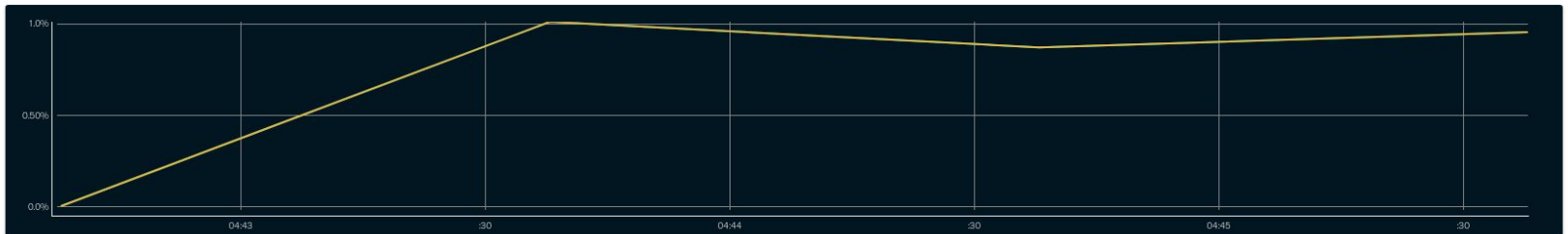
Operation Execution  
Times ⓘ

- READS
- WRITES
- COMMANDS



Disk Utilization ⓘ

- MAX UTIL % OF ANY DRIVE
- AVERAGE UTIL % OF ALL DRIVES





# MongoDB

При установлении факта деградации производительности системы, вычислить проблемные запросы возможно используя следующий вызов **mongo-shell**:

```
db.currentOp({"secs_running": {$gte: 5}})
```

Возвращаемый на данный запрос документ будет иметь поля:

- **query** (выполняемый запрос)
- **active** (выполняются ли сейчас операции по запросу)
- **ns** (имя коллекции, к которой относится запрос)
- **secs\_running** (время выполнения запроса)

# MongoDB

Узнать информацию о исполнении конкретного запроса возможно, используя на запросе метод:

```
.explain("executionStats")
```

Пример вызова метода *explain*:

```
db.inventory.find(  
  { quantity: { $gte: 100, $lte: 200 } }  
)<pre>.explain("executionStats")</pre>
```



# Redis

# Redis

Основной метрикой контроля работоспособности Redis является latency.

Latency можно измерять, используя **redis-cli**:

```
redis-cli --latency -h `host` -p `port`
```

При деградации данной величины, необходимо произвести контроль настроек окружения.

**Redis является однопоточным приложением.**

Взято с сайта: [redis.io](https://redis.io)

# Redis

Чек лист контроля основных настроек окружения Redis:

- проверка Redis на наличие блокирующих slow команд:  
*redis- cli SLOWLOG GET N*
- проверка отключения huge\_page на уровне ядра:  
*echo never > /sys/kernel/mm/transparent\_hugepage/enabled  
&& \systemctl restart redis*
- проверка задержки на уровне VM:  
*redis-cli --intrinsic-latency 100*



MySQL

---

# MySQL

Типовые проблемы, возникающие при использовании MySQL:

- замедление выполнения пользовательских запросов;
- неверная настройка удаленного доступа к БД;
- нехватка серверных ресурсов;
- повреждение таблиц;
- нарушение сокетного обмена с сервером MySQL.

# MySQL

Причины замедления пользовательских запросов описаны в `slow_log`. Для включения `slow_log` в конфигурации `mysqld` необходимо описать следующие директивы:

```
slow_query_log = 1
slow_query_log_file = /var/log/mysql-slow.log
long_query_time = 2
log_queries_not_using_indexes
```

После чего сделать рестарт процесса `mysql`:

```
$ sudo systemctl restart mysql
```

Также выполнение запроса можно изучить, посредством использования SQL директивы `EXPLAIN`.

Взято с сайта: [digitalocean.com](http://digitalocean.com)



# MySQL

Пример записи из slow\_log:

```
# Time: 140905 6:33:11
# User@Host: dbuser[dbname] @ hostname [1.2.3.4]
# Query_time: 0.116250 Lock_time: 0.000035 Rows_sent: 0 Rows_examined: 20878
use dbname;
SET timestamp=1409898791;
...SLOW QUERY HERE...
```

---

# MySQL

Причины нарушения удаленного доступа чаще всего заключаются в:

- неверно выставленных сетевых настройках сервера БД;
- некорректных правах доступа до сущностей БД.

# MySQL

Сетевые настройки сервера БД устанавливаются в конфигурационном файле, посредством изменения следующей директивы:

```
bind-address      = 0.0.0.0
```

Из соображений безопасности не рекомендуется выставлять внешний ip адрес или wildcard хоста в данной директиве.

Проброс соединений до сервера БД рекомендуется осуществлять через балансировщик запросов.

# MySQL

Синтаксис запроса просмотра привилегий на сущности БД для пользователя:

```
SHOW GRANTS
  [FOR user_or_role
    [USING role [, role] ...]]

user_or_role: {
  user (see Section 6.2.4, "Specifying Account Names")
| role (see Section 6.2.5, "Specifying Role Names".
}
```

Пример запроса привилегий пользователя:

```
mysql> SHOW GRANTS FOR 'jeffrey'@'localhost';
+-----+
| Grants for jeffrey@localhost |
+-----+
| GRANT USAGE ON *.* TO `jeffrey`@`localhost` |
| GRANT SELECT, INSERT, UPDATE ON `db1`.* TO `jeffrey`@`localhost` |
+-----+
```

Очистка “кешей” привилегий производится следующим образом:

```
FLUSH PRIVILEGES;
```

# MySQL

Нехватка серверных ресурсов в основном заключается в достижении пределов памяти дисковых устройств, либо переполнению inodes.

Пример записей в mysql.log при данном виде проблем:

```
Out of memory or mmap can't allocate
```

Данные характеристики являются компонентами мониторинга хоста.

Дополнительно можно произвести тонкую настройку mysql сервера, используя конфигурационный файл для оптимальной настройки использования памяти сервером MySQL.

---

# MySQL

Основные причины повреждения таблиц с данными:

- Непредвиденная остановка сервера MySQL во время операции записи;
- Одновременное изменение внешним ПО записей и сервером БД;
- Непредвиденное отключение сервера БД;
- Выход из строя хост-машины;
- Программная ошибка MySQL.

# MySQL

При возникновении повреждения таблиц с данными необходимо (пример для MyISAM):

- остановить сервер БД

```
$ sudo systemctl stop mysql
```

- скопировать все данные сервера БД на хосте

```
$ cp -r /var/lib/mysql /var/lib/mysql_bkp
```

- зайти в mysql-cli и запустить SQL команду для проверки таблицы

```
mysql> CHECK TABLE table_name;
```

- если таблица повреждена, необходимо ее восстановить

```
mysql> REPAIR TABLE table_name;
```

- в случае успешного восстановления, будет следующий вывод

```
+-----+-----+-----+-----+
| database_name.table_name | repair | status | OK |
+-----+-----+-----+-----+
```

---

# MySQL

В основном проблемы с сокетным взаимодействием возникает в случае:

- искажения конфигурационного файла сервера MySQL;
- нарушения прав доступа до сокет-файла.



# MySQL

Для выставления корректных прав доступа и проверки корректности конфигурации:

- Определите местоположение в файловой системе файла сокета и идентификатора процесса из конфигурационного файла

```
pid-file      = /var/run/mysqld/mysqld.pid  
socket        = /var/run/mysqld/mysqld.sock
```

- Изучите права на директорию с этими файлами

```
$ ls -la /var/run/mysqld/
```

- В случае некорректно выставленных прав, установите необходимые

```
$ sudo chown mysql:mysql /var/run/mysqld/
```

```
$ sudo chmod -R 755 /var/run/mysqld/
```

**Важно!** Перед проведением данных процедур - необходима остановка сервера БД.



# PostgreSQL

---

# PostgreSQL

Типовые проблемы, возникающие при использовании PostgreSQL схожи с MySQL:

- замедление выполнения пользовательских запросов;
- неверная настройка удаленного доступа к БД;
- нехватка серверных ресурсов;
- повреждение таблиц.

# PostgreSQL

Причины замедления пользовательских запросов описаны в `slow_log`, аналогично MySQL. По умолчанию в PostgreSQL `slow_log` отключен.

Для включения `slow_log` необходимо в конфигурационном файле объявить временную величину, дольше которой записи будут считаться `slow` и записываться в `log`, например 5 секунд:

```
log_min_duration_statement = 5000
```

Также можно для каждой БД присвоить индивидуальную величину времени, для записи в `slow_log`:

```
ALTER DATABASE test SET log_min_duration_statement = 5000;
```

Пример записи в `slow log`:

```
LOG: duration: 10010.353 ms statement: SELECT pg_sleep(10);
```

Анализ причин замедления запроса можно провести, используя директиву **EXPLAIN**

# PostgreSQL

Проблемы удаленного доступа чаще всего связаны со следующим:

- Доступ на фаерволе

*Исправляется настройкой фаервола на хост-машине*

- Некорректная привязка сервера БД к сетевому интерфейсу

```
listen_address = '*'           # Listen on all network interfaces
listen_address = '192.168.1.1' # Listen on the interface having this
address
listen_address = 'localhost, 192.168.1.1' #Listen on specified
address and loopback
```

- Некорректно указаны методы аутентификации (в файле pg\_hba.conf)

```
host    all    all    127.0.0.1/32    md5
```



# PostgreSQL

Восстановление поврежденных таблиц является довольно сложной и нетривиальной процедурой в рамках postgresql.

“Best-practice” для решения таких проблем - является регулярный backup данных и тестирование этих backup. В случае повреждения таблиц, всегда можно восстановиться из такого backup и WAL.

**Важно!** Перед операциями восстановления - произведите создание резервной копии данных на уровне файловой системы.



# Elasticsearch

---

# Elasticsearch

Типовые проблемы, возникающие при использовании Elasticsearch:

- утечка памяти на уровне JVM;
- отсутствие привязки индексов и шард.



---

# Elasticsearch

**Профилирование - способ нахождения утечек памяти JVM.**

**Оптимальные показатели профилировки GC:**

- графики имеют “пилообразный” характер;
- нет резких скачков потребления памяти - “climbing”;
- частота “пил” на графике - стремится к константе;
- значения потребления памяти не доходят до граничных, установленных в *Xms* и *Xmx*.



# Elasticsearch

## Garbage Collector Logging

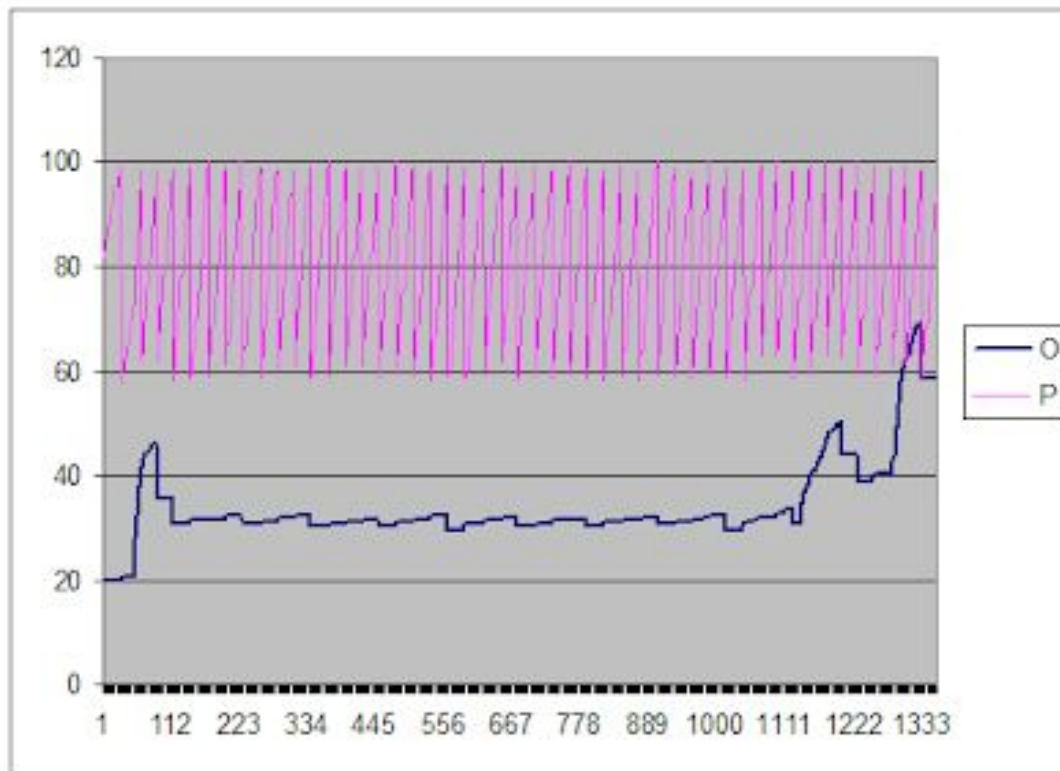
Пример настроек:

- XX:+PrintGCDetails;**
- XX:+PrintGCDateStamps;**
- Xloggc:/opt/app/gc.log.**

В штатной работе - обычно логирование GC не используется.

# Elasticsearch

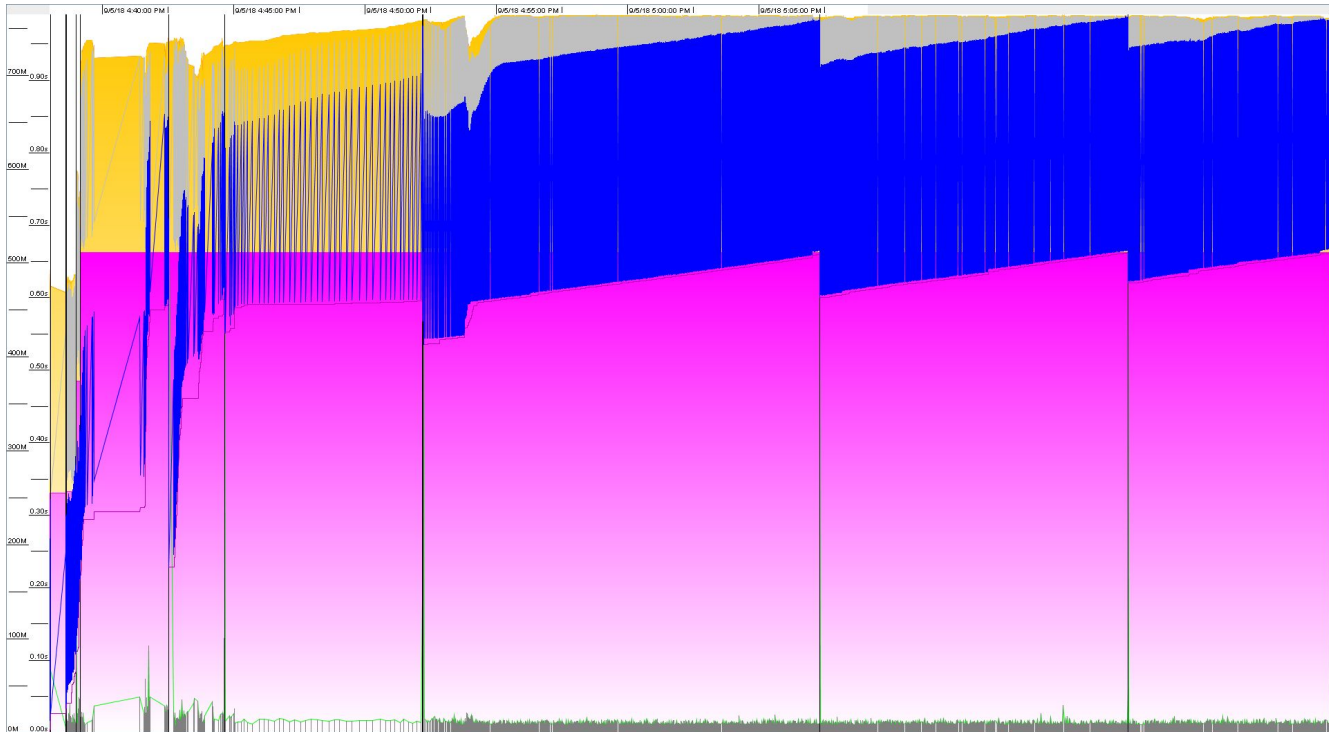
## Пример профилировки с утечкой памяти



Взято с сайта: [threeturves.blogspot.com/2009/08/how-to-find-java-memory-leaks-part-1.html](http://threeturves.blogspot.com/2009/08/how-to-find-java-memory-leaks-part-1.html)

# Elasticsearch

Пример профилировки с штатной работой:



Взято с сайта: <https://confluence.atlassian.com/kb/how-to-define-xmx-based-on-gc-logs-960142303.html>

# Elasticsearch

В случае появления UNASSIGNED шард и индексов (например при потере какой-либо ноды кластера), можно осуществить их перепривязку к конкретной кластерной ноде, используя API.

Поиск шард в состоянии UNASSIGNED:

```
curl -XGET localhost:9200/_cat/shards?h=index,shard,prirep,state,unassigned.reason| grep UNASSIGNED
```

Ответное сообщение будет содержать:

```
constant-updates 0 p UNASSIGNED NODE_LEFT node_left
```

# Elasticsearch

Вызов API для ручной  
перепривязки индексов.

```
POST /_cluster/reroute
{
  "commands": [
    {
      "move": {
        "index": "test", "shard": 0,
        "from_node": "node1", "to_node": "node2"
      }
    },
    {
      "allocate_replica": {
        "index": "test", "shard": 1,
        "node": "node3"
      }
    }
  ]
}
```



# Итоги

---

# Итоги

В рамках данной лекции мы рассмотрели типовые проблемы с серверами БД:

- MongoDB;
- Redis;
- MySQL;
- PostgreSQL;
- Elasticsearch.

И узнали возможные пути решения этих проблем.





## Домашнее задание

Давайте посмотрим ваше [домашнее задание](#).

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

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

**Роман Гордиенко**