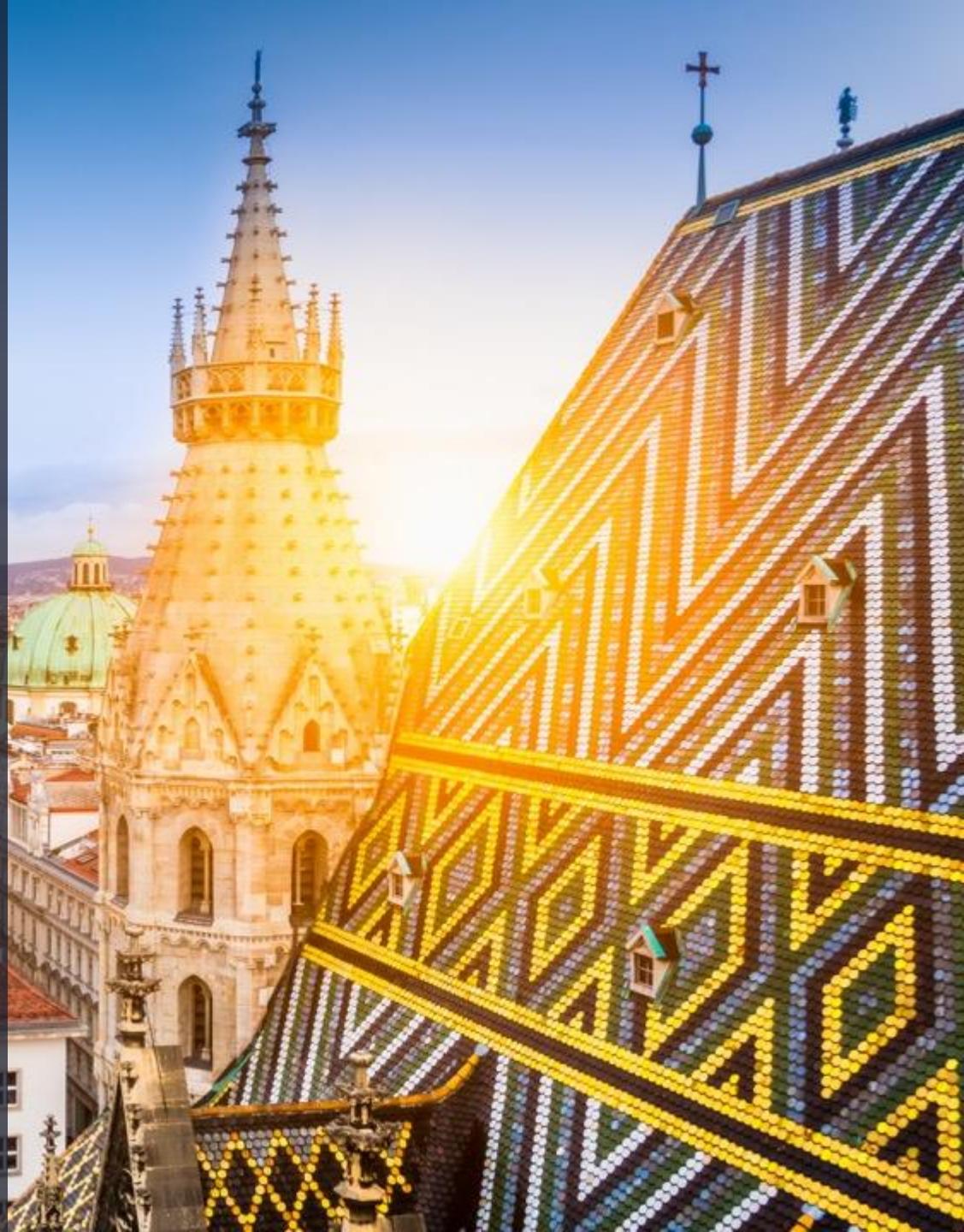


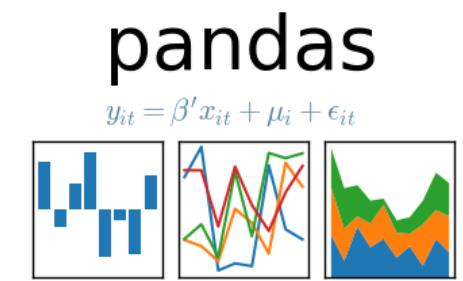
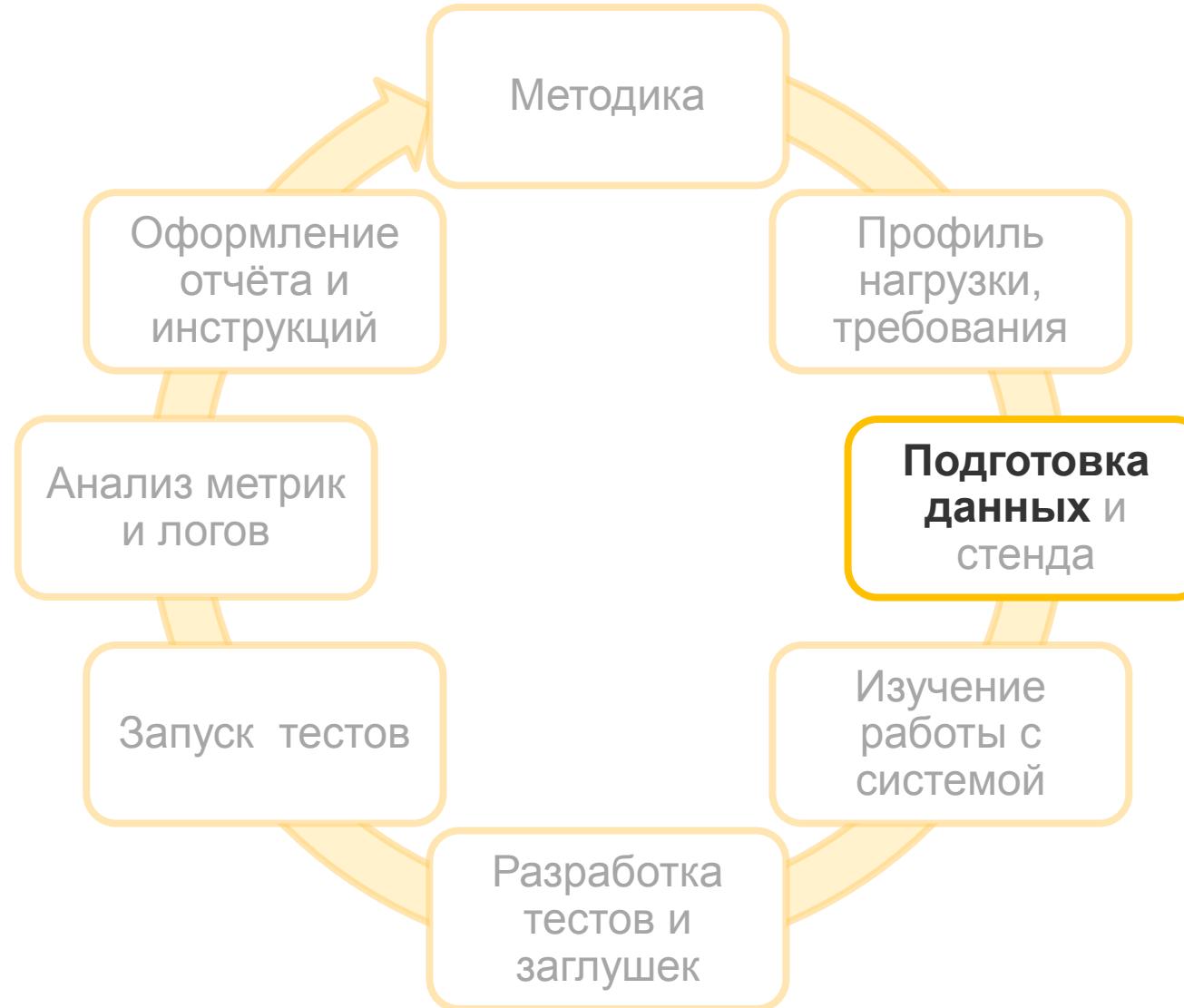
# Готовим тестовые данные

Опыт команды ELBRUS Team





# Готовим тестовые данные



# Подготовка тестовых данных

Для нагрузочного тестирования  
новых микросервисов  
в банке



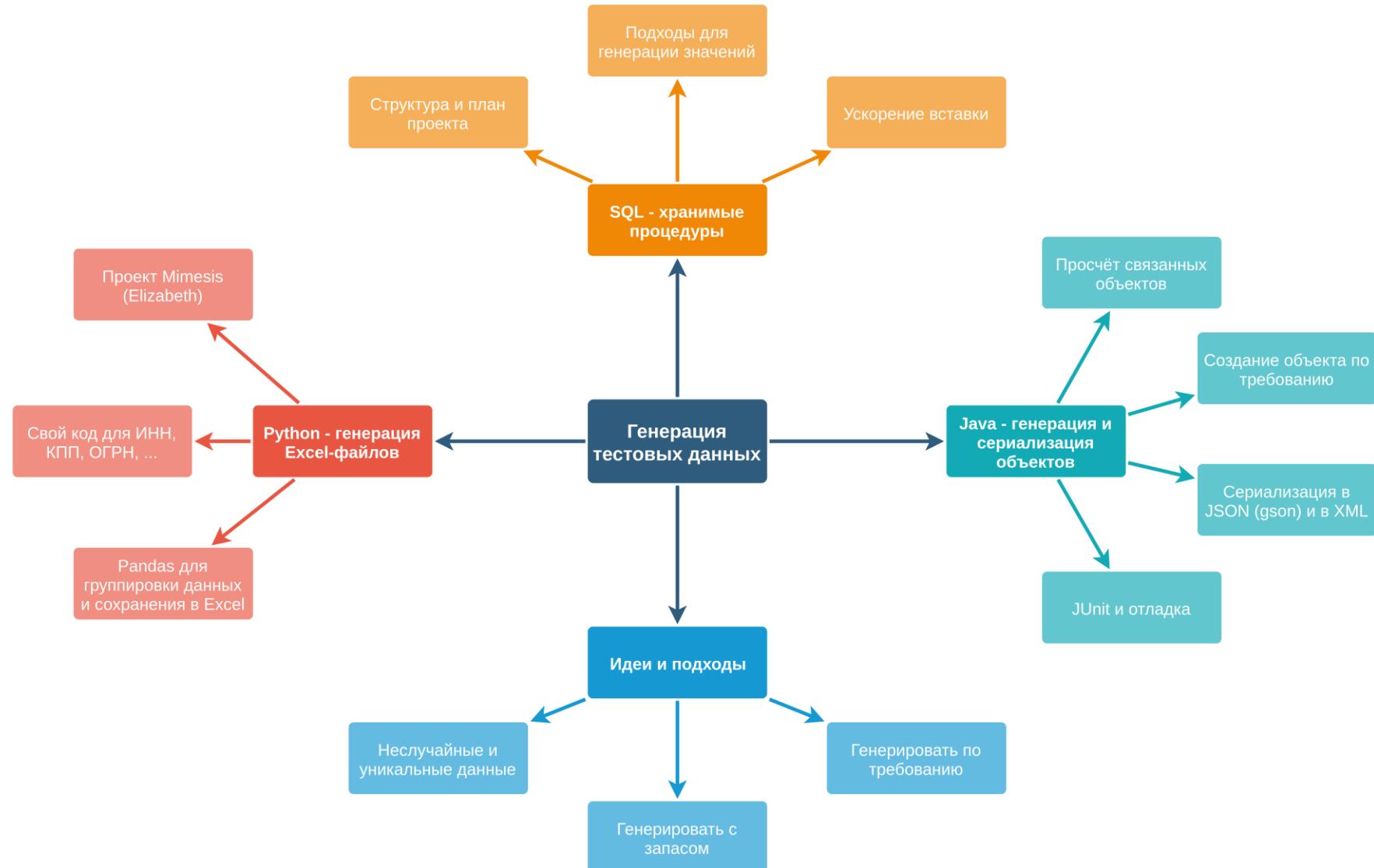
# Подготовка тестовых данных

Для нагрузочного тестирования  
новых микросервисов  
в банке  
в команде ELBRUS Team





# Генерация тестовых данных



Основа генератора пишется на SQL, а  
Java/Scala помогают  
Данных должно быть много

# SQL для генерации данных

Почему SQL, а не API: Java/Scala/Python/C#/...



## Скорость SQL

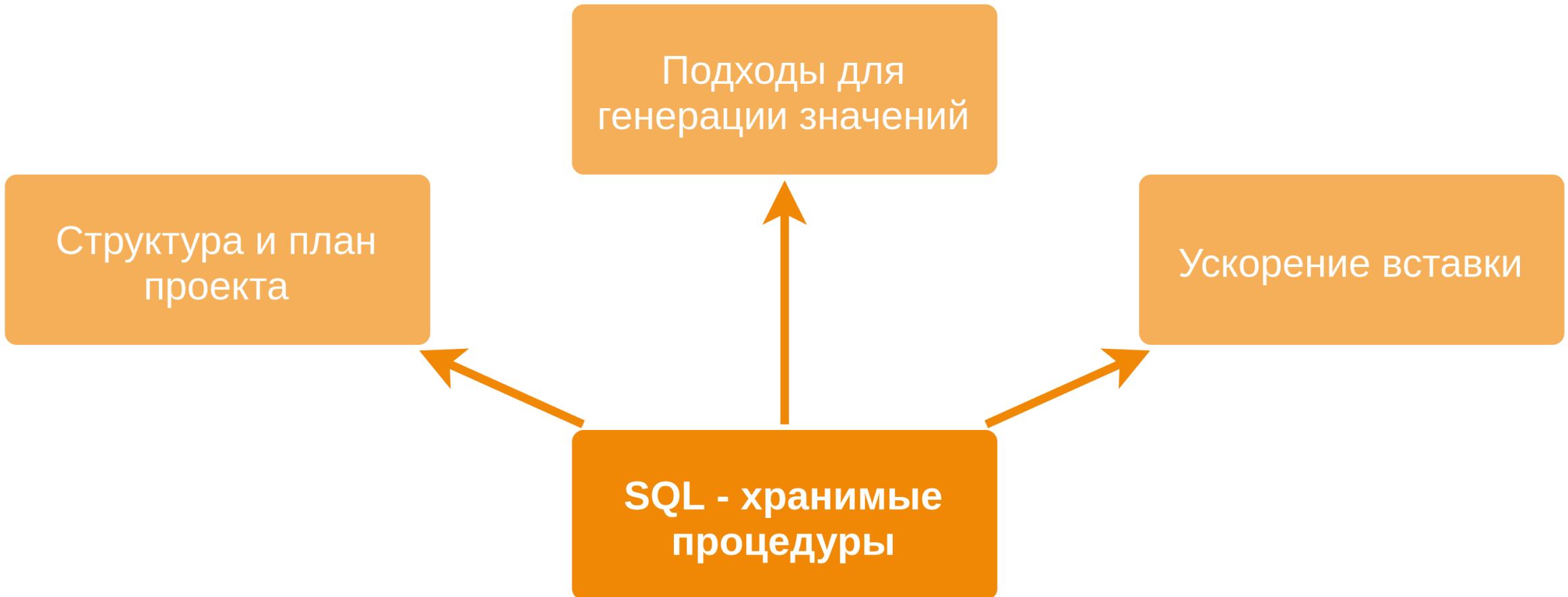
- можно сгенерировать 100 Гбайт
- не нужны интеграции
- SQL может многое



## Гибкость и надежность API

- генерация на активной системе
- быстрый отклик по корректности
- Java/Scala/Python/C#/... могут всё

# Опыт использования SQL-генераторов





# Сервис аутентификации

Разделение тестовых данных по сценариям

Сценарий	Домен
Login_logout	@ok.org
Try_fail_pass	@bad.org
Reset_password	@password.com

нбанк

айффайзен  
БАНК

Авторизация

RBO/ELBRUS

[Проверка работы плагина.](#)  
[Первоначальная установка.](#)

Логин или адрес электронной почты

Пароль

Войти

Забыли пароль?

Рекомендации по безопасной работе с банком

Не отвечайте на электронные письма, СМС или другие сообщения с просьбой уточнить конфиденциальные данные — скорее всего, это мошенники.



# Планирование данных

Продумать шаблоны и количество

Сценарий	Шаблоны	Количество
Login_logout	@ok.org loadUser{N}	Users – 100к History – x3
Try_fail_pass	@bad.org incorrect{N}pass	Users – 50к History – x1
Reset_password	@password.com user{N}Reset	Users – 10к History – x5

Готовим тестовые данные

	refresh_token	да
новление пароля	update_password	да

ИИЯ

необходимые тестовые данные		Шаблоны тестовых	
ailed_login_attempts	user_password_history	login	email
00 000, Nx3	900 000, Nx3	sms{N}enable	*@factor2enable.com
500 000, Nx5	1 500 000, Nx5	change{N}Password	*@changePass.com
-	inviteUser{N}		*@invite.ru
00 000, Nx1	0	migration{N}user	*@company.ru
50 000 Nx3	150 000 Nx3	update{N}userPass	*@old.ru



# Шаблоны и количество

Передаются как параметры в хранимые процедуры

load\_fill\_database\_{scenario}

load\_fill\_database

load\_get\_name,  
load\_get\_password, ...

```
-- Function: public.load_fill_database_change_password(integer, integer)
-- DROP FUNCTION public.load_fill_database_change_password(integer, integer);

CREATE OR REPLACE FUNCTION public.load_fill_database_change_password(
    "userIndexStart" integer DEFAULT 1,
    "userIndexEnd" integer DEFAULT 300000)
RETURNS text AS
$BODY$
BEGIN
    return public.load_fill_database(
        "userIndexStart",
        "userIndexEnd",
        'change%sPassword', -- login_format
        '%s@changeWord.com', -- email_format
        '+7960', -- phone_prefix
        'change_password', -- test_name
        '106.%s.%s.%s',
        '136.%s.%s.%s',
        'https://testStandName.raiffeisen.ru/profile',
        false);
END
$BODY$
LANGUAGE plpgsql VOLATILE
COST 100;
ALTER FUNCTION public.load_fill_database_change_password(integer, integer)
OWNER TO admin;

-- Function: public.load_fill_database_login_logout(integer, integer)
-- DROP FUNCTION public.load_fill_database_login_logout(integer, integer);

CREATE OR REPLACE FUNCTION public.load_fill_database_login_logout(
    "userIndexStart" integer DEFAULT 1,
    "userIndexEnd" integer DEFAULT 1500000)
RETURNS text AS
$BODY$
BEGIN
    return public.load_fill_database(
        "userIndexStart",
        "userIndexEnd",
        'good%$login', -- login_format
        '%s@ok.org', -- email_format
        '+790', -- phone_prefix
        'login_logout', -- test_name
        '100.%s.%s.%s'.
```



# Шаблоны и количество

Передаются как параметры в хранимые процедуры

load\_fill\_database\_{scenario}

load\_fill\_database

load\_get\_name,  
load\_get\_password, ...

```
CREATE OR REPLACE FUNCTION public.load_fill_database(
    "userIndexStart" integer,
    "userIndexEnd" integer,
    login_format character varying,
    email_format character varying,
    phone_prefix character varying,
    test_name character varying,
    failed_login_ip_format character varying,
    back_ip_format character varying,
    redirect_uri character varying,
    two_factor_auth_on_value boolean)
RETURNS text AS
$BODY$
DECLARE
    caseIndex integer;
    timeDiffMinutes integer := 1;
    nowTime timestamp;
    date_create_value timestamp;
    nick_name_value character varying(255);
    user_id_value character varying(255);
    ip_address_index integer;
    randomStr text;
    old_token_id character varying(36);
    old_token_date_expires timestamp;
    phone_suffix_len integer;

BEGIN
    nowTime = NOW();
    phone_suffix_len = 12 - length(phone_prefix);

    FOR i IN "userIndexStart".."userIndexEnd" LOOP

        timeDiffMinutes = i % 80000;
        date_create_value = nowTime - (timeDiffMinutes * interval '60 second') - (3 * int
            nick_name_value = format(login_format, i::text)
            user_id_value = md5('users ' || test_name || lpad(i::text, 12, '0'))::uuid;

            insert into Users (
                id,
                nick_name,
                name,
                given_name,
                family_name,
                middle_name,
```



# Подходы для генерации значений

Объект – функция от параметров и номера

FOR index IN  
"indexStart".."indexEnd" LOOP

value = function(index, params)

Использование value для  
вставки в таблицы

```
CREATE OR REPLACE FUNCTION public.load_fill_database(  
    "userIndexStart" integer,  
    "userIndexEnd" integer,  
    login_format character varying,  
    email_format character varying,  
    phone_prefix character varying,  
    test_name character varying,  
    failed_login_ip_format character varying,  
    back_ip_format character varying,  
    redirect_uri character varying,  
    two_factor_auth_on_value boolean)  
RETURNS text AS  
$BODY$  
DECLARE  
    caseIndex integer;  
    timeDiffMinutes integer := 1;  
    nowTime timestamp;  
    date_create_value timestamp;  
    nick_name_value character varying(255);  
    user_id_value character varying(255);  
    ip_address_index integer;  
    randomStr text;  
    old_token_id character varying(36);  
    old_token_date_expires timestamp;  
    phone_suffix_len integer;  
  
BEGIN  
    nowTime = NOW();  
    phone_suffix_len = 12 - length(phone_prefix);  
  
    FOR i IN "userIndexStart".."userIndexEnd" LOOP  
  
        timeDiffMinutes = i % 80000;  
        date_create_value = nowTime - (timeDiffMinutes * interval '60 second') - (3 * int  
  
        nick_name_value = format(login_format, i::text)  
        user_id_value = md5('users ' || test_name || lpad(i::text, 12, '0'))::uuid;  
  
        insert into Users (  
            id,  
            nick_name,  
  
            name,  
            given_name,  
            family_name,  
            middle_name,
```



# Подходы для генерации значений

[0; N-1] = Остаток от деления на N

timeDiffMinutes  
= i % 80000

Готовим тестовые данные

```
CREATE OR REPLACE FUNCTION public.load_fill_database(
    "userIndexStart" integer,
    "userIndexEnd" integer,
    login_format character varying,
    email_format character varying,
    phone_prefix character varying,
    test_name character varying,
    failed_login_ip_format character varying,
    back_ip_format character varying,
    redirect_uri character varying,
    two_factor_auth_on_value boolean)
RETURNS text AS
$BODY$
DECLARE
    caseIndex integer;
    timeDiffMinutes integer := 1;
    nowTime timestamp;
    date_create_value timestamp;
    nick_name_value character varying(255);
    user_id_value character varying(255);
    ip_address_index integer;
    randomStr text;
    old_token_id character varying(36);
    old_token_date_expires timestamp;
    phone_suffix_len integer;

BEGIN
    nowTime = NOW();
    phone_suffix_len = 12 - length(phone_prefix);

    FOR i IN "userIndexStart".."userIndexEnd" LOOP

        timeDiffMinutes = i % 80000;
        date_create_value = nowTime - (timeDiffMinutes * interval '60 second') - (3 * int
        nick_name_value = format(login_format, i::text)
        user_id_value = md5('users ' || test_name || lpad(i::text, 12, '0'))::uuid;

        insert into Users (
            id,
            nick_name,
            name,
            given_name,
            family_name,
            middle_name,
```



# Подходы для генерации значений

Функция format для шаблонных строк

```
nick_name =  
format('user%$name',  
i::text)
```

user123name

Готовим тестовые данные

```
CREATE OR REPLACE FUNCTION public.load_fill_database(  
    "userIndexStart" integer,  
    "userIndexEnd" integer,  
    login_format character varying,  
    email_format character varying,  
    phone_prefix character varying,  
    test_name character varying,  
    failed_login_ip_format character varying,  
    back_ip_format character varying,  
    redirect_uri character varying,  
    two_factor_auth_on_value boolean)  
RETURNS text AS  
$BODY$  
DECLARE  
    caseIndex integer;  
    timeDiffMinutes integer := 1;  
    nowTime timestamp;  
    date_create_value timestamp;  
    nick_name_value character varying(255);  
    user_id_value character varying(255);  
    ip_address_index integer;  
    randomStr text;  
    old_token_id character varying(36);  
    old_token_date_expires timestamp;  
    phone_suffix_len integer;  
  
BEGIN  
    nowTime = NOW();  
    phone_suffix_len = 12 - length(phone_prefix);  
  
    FOR i IN "userIndexStart".."userIndexEnd" LOOP  
  
        timeDiffMinutes = i % 80000;  
        date_create_value = nowTime - (timeDiffMinutes * interval '60 second') - (3 * int  
  
        nick_name_value = format(login_format, i::text)  
        user_id_value = md5('users ' || test_name || lpad(i::text, 12, '0'))::uuid;  
  
        insert into Users (  
            id,  
            nick_name,  
  
            name,  
            given_name,  
            family_name,  
            middle_name,
```



# Подходы для генерации значений

Комбинации format и остатка от деления для IPv4

```
format('106.%s.%s.%s',
(((ip_index / 250) / 250) % 250)::text,
((ip_index / 250) % 250)::text,
(1 + ip_index % 250)::text )
```

106.0.1.2

15 625 000 значений

```
CREATE OR REPLACE FUNCTION public.load_fill_database(
    "userIndexStart" integer,
    "userIndexEnd" integer,
    login_format character varying,
    email_format character varying,
    phone_prefix character varying,
    test_name character varying,
    failed_login_ip_format character varying,
    back_ip_format character varying,
    redirect_uri character varying,
    two_factor_auth_on_value boolean)
RETURNS text AS
$BODY$
DECLARE
    caseIndex integer;
    timeDiffMinutes integer := 1;
    nowTime timestamp;
    date_create_value timestamp;
    nick_name_value character varying(255);
    user_id_value character varying(255);
    ip_address_index integer;
    randomStr text;
    old_token_id character varying(36);
    old_token_date_expires timestamp;
    phone_suffix_len integer;

BEGIN
    nowTime = NOW();
    phone_suffix_len = 12 - length(phone_prefix);

    FOR i IN "userIndexStart".."userIndexEnd" LOOP

        timeDiffMinutes = i % 80000;
        date_create_value = nowTime - (timeDiffMinutes * interval '60 second') - (3 * int
nick_name_value = format(login_format, i::text)
user_id_value = md5('users ' || test_name || lpad(i::text, 12, '0'))::uuid;

        insert into Users (
            id,
            nick_name,
            name,
            given_name,
            family_name,
            middle_name,
```



# Подходы для генерации значений

GUID = md5 от имени таблицы, теста и номера

```
guid = md5('users' ||  
           test_name ||  
           lpad(i::text, 12, '0'))::uuid;
```

faf154aa-54d1-0b07-  
3c8e-fe8921f96c45

Готовим тестовые данные

```
CREATE OR REPLACE FUNCTION public.load_fill_database(  
    "userIndexStart" integer,  
    "userIndexEnd" integer,  
    login_format character varying,  
    email_format character varying,  
    phone_prefix character varying,  
    test_name character varying,  
    failed_login_ip_format character varying,  
    back_ip_format character varying,  
    redirect_uri character varying,  
    two_factor_auth_on_value boolean)  
RETURNS text AS  
$BODY$  
DECLARE  
    caseIndex integer;  
    timeDiffMinutes integer := 1;  
    nowTime timestamp;  
    date_create_value timestamp;  
    nick_name_value character varying(255);  
    user_id_value character varying(255);  
    ip_address_index integer;  
    randomStr text;  
    old_token_id character varying(36);  
    old_token_date_expires timestamp;  
    phone_suffix_len integer;  
  
BEGIN  
    nowTime = NOW();  
    phone_suffix_len = 12 - length(phone_prefix);  
  
    FOR i IN "userIndexStart".."userIndexEnd" LOOP  
  
        timeDiffMinutes = i % 80000;  
        date_create_value = nowTime - (timeDiffMinutes * interval '60 second') - (3 * int  
  
        nick_name_value = format(login_format, i::text)  
        user_id_value = md5('users' || test_name || lpad(i::text, 12, '0'))::uuid;  
  
        insert into Users (  
            id,  
            nick_name,  
  
            name,  
            given_name,  
            family_name,  
            middle_name,
```



# Подходы для генерации значений

GUID = код(имени таблицы), код(теста) и номер

Номер = 0 .. 999 999 999 999

```
guid = ('22-33-44-55' ||  
'aa-bb-cc-dd-ee-ff' ||
```

```
lpad(i::text, 12, '0'))::uuid;
```

```
22334455-aabb-ccdd-  
eeff-000000000001
```

Готовим тестовые данные

```
CREATE OR REPLACE FUNCTION public.load_fill_database(  
    "userIndexStart" integer,  
    "userIndexEnd" integer,  
    login_format character varying,  
    email_format character varying,  
    phone_prefix character varying,  
    test_name character varying,  
    failed_login_ip_format character varying,  
    back_ip_format character varying,  
    redirect_uri character varying,  
    two_factor_auth_on_value boolean)  
RETURNS text AS  
$BODY$  
DECLARE  
    caseIndex integer;  
    timeDiffMinutes integer := 1;  
    nowTime timestamp;  
    date_create_value timestamp;  
    nick_name_value character varying(255);  
    user_id_value character varying(255);  
    ip_address_index integer;  
    randomStr text;  
    old_token_id character varying(36);  
    old_token_date_expires timestamp;  
    phone_suffix_len integer;  
  
BEGIN  
    nowTime = NOW();  
    phone_suffix_len = 12 - length(phone_prefix);  
  
    FOR i IN "userIndexStart".."userIndexEnd" LOOP  
  
        timeDiffMinutes = i % 80000;  
        date_create_value = nowTime - (timeDiffMinutes * interval '60 second') - (3 * int  
  
        nick_name_value = format(login_format, i::text)  
        user_id_value = md5('users ' || test_name || lpad(i::text, 12, '0'))::uuid;  
  
        insert into Users (  
            id,  
            nick_name,  
  
            name,  
            given_name,  
            family_name,  
            middle_name,
```



# Подходы для генерации значений

Выбор из словаря по номеру

```
array[ 1 +  
mod(abs("Index"),  
arrayLength) ]
```

предел 1 ГБайт,  
тестировал 10 МБайт,  
использую 64 кБайт

```
-- Function: public.load_get_name(integer)  
  
-- DROP FUNCTION public.load_get_name(integer);  
  
CREATE OR REPLACE FUNCTION public.load_get_name("userIndex" integer DEFAULT 0)  
RETURNS text AS  
$BODY$  
DECLARE  
    nameArray text[] := array[  
        'Авдей',  
        'Авдий',  
        'Авенир',  
        'Аверкий',  
        'Аксентий',  
        'Агафон',  
  
        '...',  
  
        'Фёдор',  
        'Харитон',  
        'Христофор',  
        'Эдуард',  
        'Эраст',  
        'Юlian',  
        'Юлий',  
        'Юрий',  
        'Юстин',  
        'Яков',  
        'Якун',  
        'Ярослав'  
    ];  
    nameArrayLenght integer := array_length(nameArray, 1);  
BEGIN  
    RETURN nameArray[ 1 + mod(abs("userIndex"), nameArrayLenght) ];  
END  
$BODY$  
LANGUAGE plpgsql VOLATILE  
COST 100;  
ALTER FUNCTION public.load_get_name(integer)  
OWNER TO postgres;
```



# Подходы для генерации значений

Выбор из сгенерированной тестовой таблицы

загрузка таблицы  
из CSV-фала

генератор CSV-файла  
на java/scala/...

```
CREATE FUNCTION public.load_get_password("userLogin")
RETURNS text AS
$BODY$
DECLARE
    password varchar(255);
BEGIN
    password = (select password_hash
        from test_passwords
        where user_name="userLogin"
        limit 1);

    RETURN password;
END
$BODY$
LANGUAGE plpgsql VOLATILE;
```



# Ускорение вставки

Можно отключить индексы, удалить ключи, ...



Сохранить схему БД



Отключить индексы, удалить ключи, ...



Сгенерировать данные



Включить индексы, вернуть ключи, ...



Сравнить схему БД (до и после)

... ИЛИ ВСЁ УДАЛИТЬ И  
ВОССТАНОВИТЬ ПОСЛЕ  
ВСТАВКИ ДАННЫХ

Готовим тестовые данные

```
-- Отключить индексы и внешний ключи, ...
UPDATE pg_index
SET indisready=false
WHERE indrelid = (
    SELECT oid
    FROM pg_class
    WHERE relname in ('users', 'sessions')
);
-- удалить первичные ключи, ...
ALTER TABLE public.sessions
DROP CONSTRAINT sessions_pkey;
-- отключить внешние ключи
ALTER TABLE public.users DISABLE TRIGGER ALL;
ALTER TABLE public.sessions DISABLE TRIGGER ALL;

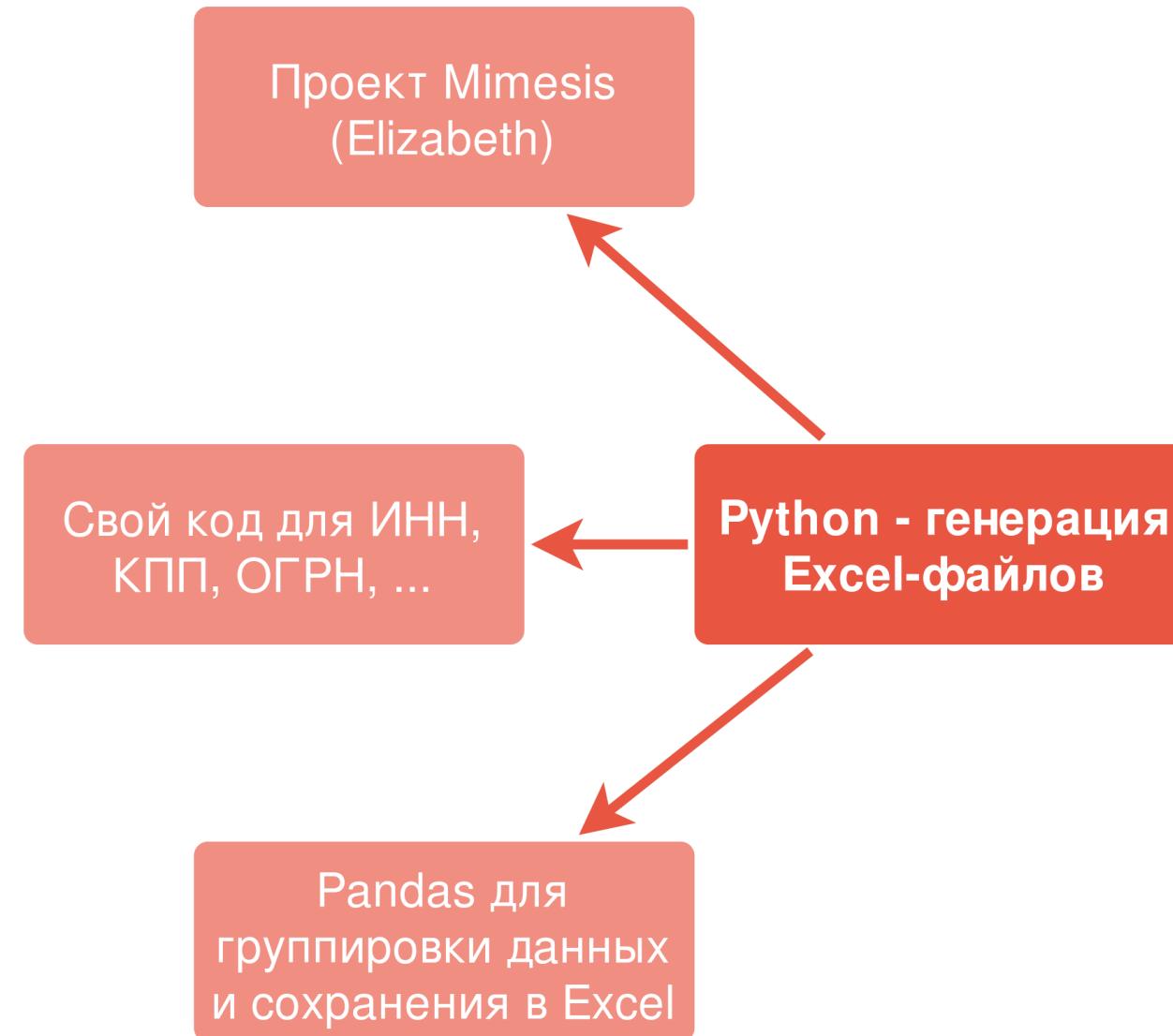
-- Включить индексы...
UPDATE pg_index
SET indisready=true
WHERE indrelid = (
    SELECT oid
    FROM pg_class
    WHERE relname in ('users', 'sessions')
);
ALTER TABLE public.users ENABLE TRIGGER ALL;
ALTER TABLE public.sessions ENABLE TRIGGER ALL;
-- ... вернуть ключи ...
ALTER TABLE public.sessions
ADD CONSTRAINT sessions_pkey PRIMARY KEY(id);
-- ... и переиндексировать
REINDEX public.users;
REINDEX public.sessions;
```

Для специальных форматов данных  
используются специальные библиотеки

Pandas для табличных данных и  
Excel-файлов



# Генерация тестовых данных: pandas, elizabeth



X





```
1 from elizabeth import Personal
2 import pandas as pd
3 import gc
4
5 gc.collect()
6
7 simpleChars = "кнгзфвпрлдчсмтб"
8 fatherNameExclude = {}
9 fatherNameExclude["Павел"] = "Павл"
10 fatherNameExclude["Лев"] = "Лъв"
11
12 #Генерация женщин
13 rowList = []
14 for index in range(1, 600000):
15     row = {}
16     person = Personal('ru')
17     row["01.Фамилия"] = person.surname()
18     row["02.Имя"] = person.name()
19     while True:
20         fatherName = person.name(gender='male')
21         if (fatherName[-1:] in simpleChars):
22             if(fatherName in fatherNameExclude):
23                 row["03.Отчество"] = fatherNameExclude[fatherName] + "овна"
24             else:
25                 row["03.Отчество"] = fatherName + "овна"
26             break
27     row["04.Имя для написания на карте"] = transliterate(row["02.Имя"] + " " + row["01.Фамилия"])[0:23]
28     row["05.Пол"] = "Женский"
29
30     rowList.append(row)
```

## Просто

- создать Person

## Не очень сложно

- сгенерировать отчество

```
1
2
3
4 #Генерация мужчин
5 for index in range(1, 600000):
6     row = {}
7     person = Personal('ru')
8     row["02.Имя"] = person.name(gender='male')
9     row["01.Фамилия"] = person.surname(gender='male')
10    while True:
11        fatherName = person.name(gender='male')
12        if (fatherName[-1:] in simpleChars):
13            if(fatherName in fatherNameExclude):
14                row["03.Отчество"] = fatherNameExclude[fatherName] + "ович"
15            else:
16                row["03.Отчество"] = fatherName + "ович"
17            break
18    row["04.Имя для написания на карте"] = transliterate(row["02.Имя"] + " " + row["01.Фамилия"])[0:23]
19    row["05.Пол"] = "Мужской"
20
21    rowList.append(row)
22
23
```

## Дополнительно нужна

- транслитерация



```
1 df = pd.DataFrame(rowList)
2 df = df.drop_duplicates(keep="first")
3
4 pSize = 20000
5 dfList = split_list(df, pSize)
6 length = len(df) // pSize
7
8
9 rowList = None
10 df = None
11 gc.collect()
12
13 for i in range(length):
14     dfPart = dfList[i]
15     dfPart["06.Дата рождения"] = "25.11.1986"
16     dfPart["07.Страна рождения"] = "РОССИЯ"
17     dfPart["08.Место рождения"] = "г. Москва"
18     dfPart["09.Страна гражданства"] = "РОССИЯ"
19     dfPart["10.ИНН"] = [get_inn_f1(i*pSize + index, 3, 0) for index in range(len(dfPart))]
20     dfPart["11.Семейное положение"] = "ДА"
21     dfPart["12.Паспорт.Тип"] = "01. Паспорт гражданина Российской Федерации - для гражданина Российской Федерации"
22     dfPart["13.Паспорт.Серия"] = [1337 + ((i*pSize + index) // 900000) for index in range(len(dfPart))]
23     dfPart["14.Паспорт.Номер"] = [str(100000 + (i*pSize + index) % 900000).zfill(6) for index in range(len(dfPart))]
24     dfPart["15.Паспорт.Страна выдавшая"] = "РОССИЯ"
25
26 dfPart.to_csv("testData." + str(i).zfill(3) + ".txt", index=False, quoting=1, encoding="windows-1251")
27 dfPart = None
28 dfList[i] = None
29 gc.collect()
```

## Удаление дублей

- Pandas

## Генерация серий

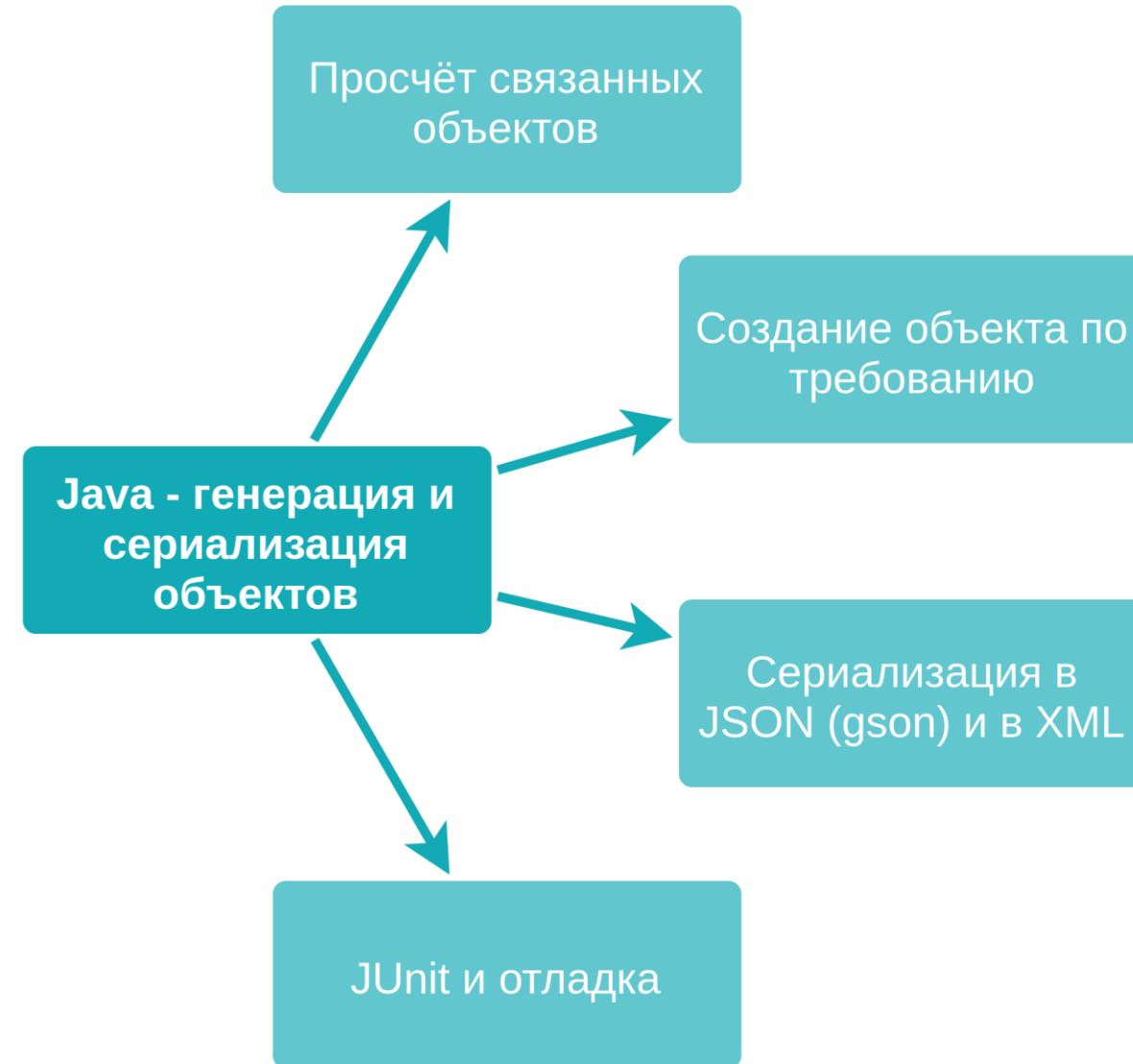
- Python

# Переиспользую исходный код приложений для сериализации в XML и JSON

Если нужно сгененировать XML и  
JSON сложной структуры



# Java, Scala когда важно не ошибиться





# Сериализация в JSON и XML

DataTransferObject, Model, XSD, ... для теста

Разработчики уже всё написали

Система преобразует входные данные из JSON и XML в объекты и валидирует их.

А тест – объекты в JSON и XML.

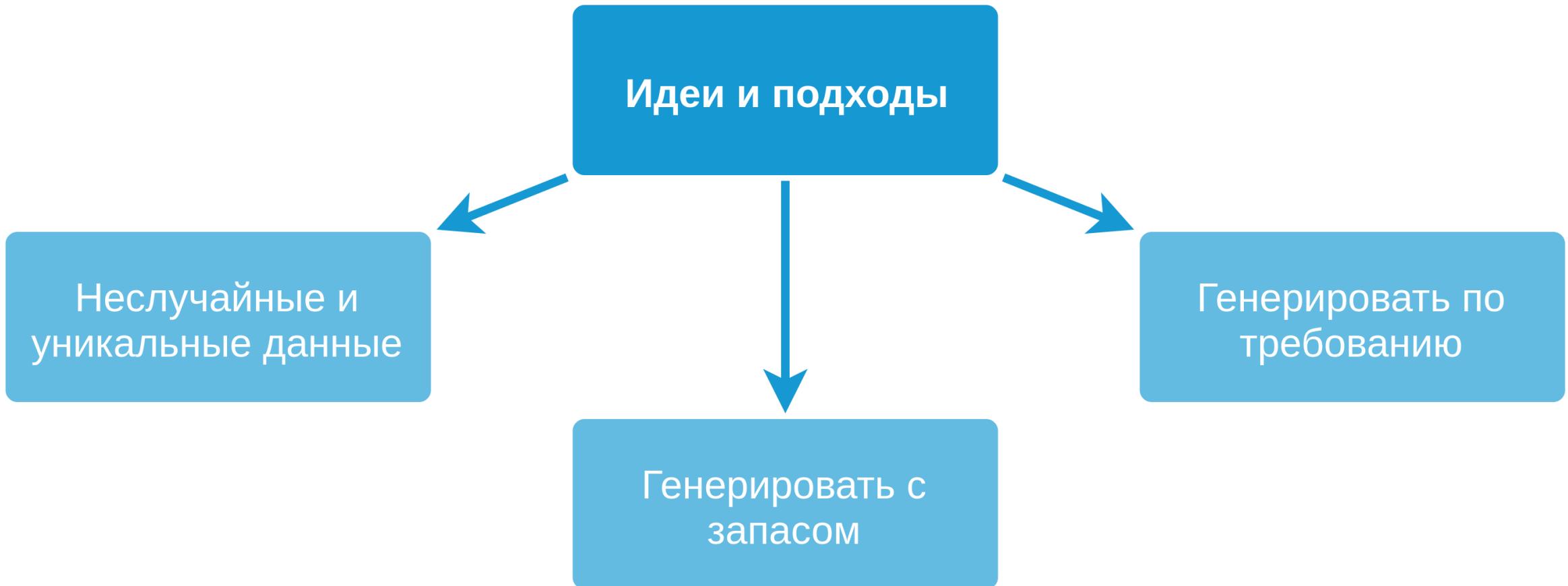
The screenshot shows a code editor with Java code for a `Transaction` class. The code uses annotations from `com.google.gson.annotations` and `org.apache.commons.lang3.builder` to handle serialization. The `dc` field is annotated with `@SerializedName("dc")` and `@Expose`. The `docguid` and `docnum` fields are also annotated with `@SerializedName` and `@Expose`. The code is part of a larger file named `ts/bitbucket.ra`.

```
1 package ru.raiffeisen.rbp.statement.lo  
2  
3 import java.math.BigDecimal;  
4 import com.google.gson.annotations.Expose;  
5 import com.google.gson.annotations.SerializedName;  
6 import org.apache.commons.lang3.builder.*;  
7 import org.apache.commons.lang3.builder.ToStringBuilder;  
8 import org.apache.commons.lang3.builder.ToStringStyle;  
9  
10 public class Transaction {  
11  
12     /**  
13      * Признак дебета / кредита. UAS:  
14      * (Required)  
15      */  
16  
17     @SerializedName("dc")  
18     @Expose  
19     private BigDecimal dc;  
20  
21     /**  
22      * Внутренний референс проводки. UAS:  
23      */  
24  
25     @SerializedName("docguid")  
26     @Expose  
27     private String docguid;  
28  
29     /**  
30      * Номер документа. UAS: P2BLSTPF.  
31      */  
32     @SerializedName("docnum")
```

# Затраты времени на генерацию данных окупаются тестированием без сбоев

Тест производительности находит  
дефекты производительности, а не  
дефекты теста, как бывает иногда

# Вкладываем в большие качественные данные



# Подготовка тестовых данных

Основа генератора пишется на SQL, а Java/Scala/... помогают

Для специальных форматов используются специальные библиотеки

Для простоты использую исходный код системы, формируя JML и JSON

Затраты времени на генерацию данных окупаются стабильностью

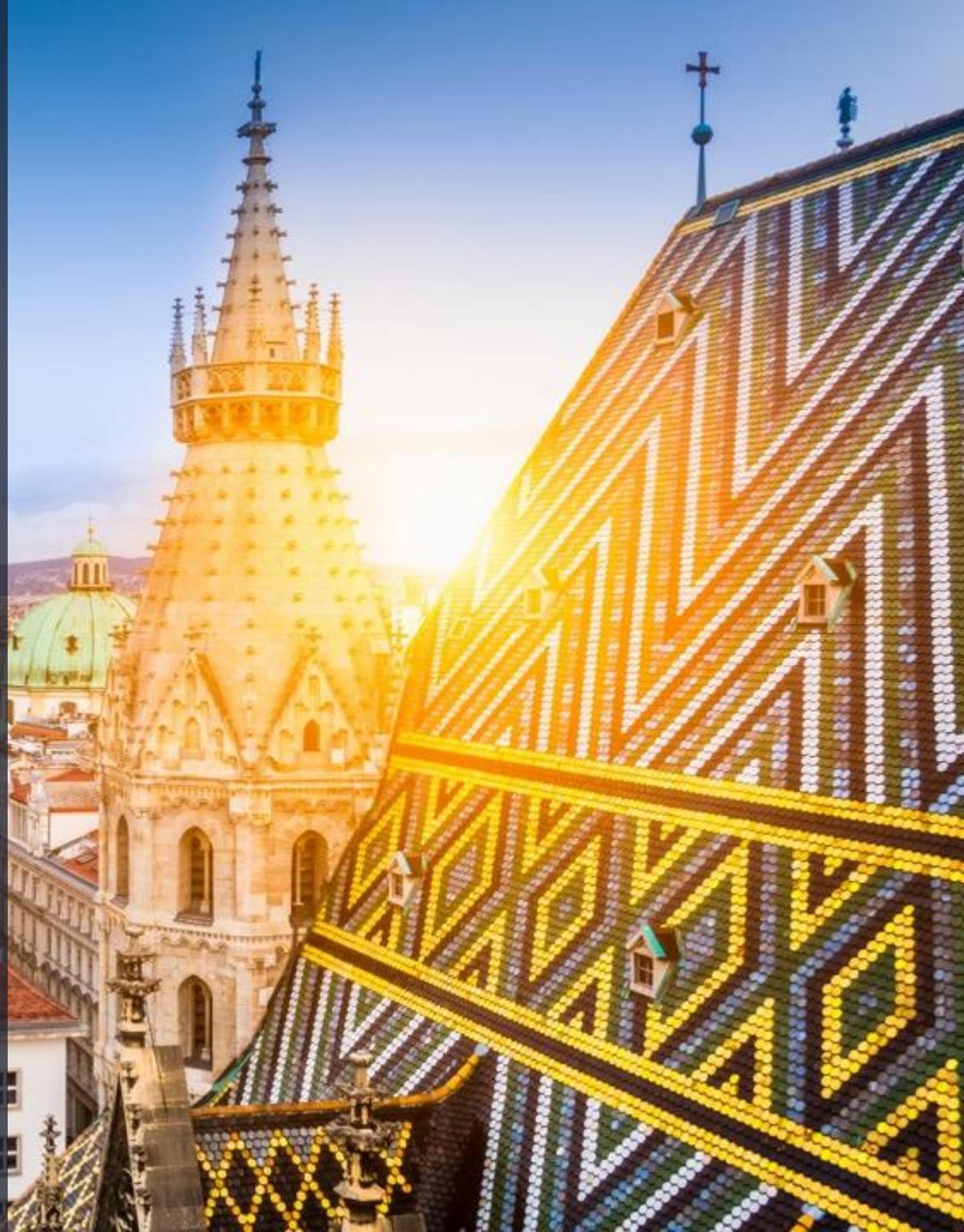


Спасибо



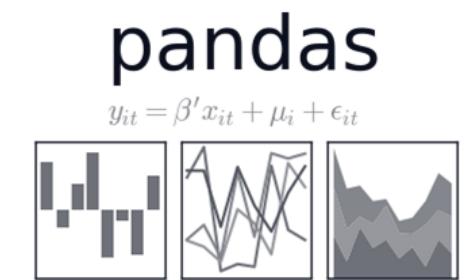
# Сервируем отчёт по тестированию

Опыт команды ELBRUS Team





# Сервируем отчёт по тестированию



# Оформление отчёта по тестированию

Ручным способом с автоматизацией  
по инцидентам и дефектам  
а также новым системам  
для гибкой команды

# Оформление отчёта по тестированию

Ручным способом с автоматизацией  
по инцидентам и дефектам  
а также новым системам  
для гибкой команды  
ELBRUS Team



# Использование шаблонов и цепочек

Нефункциональные требования

Методика тестирования производительности

Отчёт по тестированию производительности



# Шаблоны

The screenshot shows a Confluence page with the following details:

- Page Title:** [Шаблон (в разработке)] Отчёт по тестированию производительности
- Page URL:** https://confluence.raiffeisen.ru/pages/viewpage.action?pageId=140887135
- Page Content:** A table with the following information:

Description	
Version	Версия 2.0.7, модуль mobile 2.0
Status	DRAFT APPROVED
Last modified	25.11.2018 by SMIRNOV Vyacheslav
- Page Structure:** The page is part of a category 'Шаблоны документов по тестированию производительности'. It includes a sidebar with various navigation links such as 'Автоматизированное тестирование', 'Тестирование производительности', 'База знаний', etc.
- Page Footer:** Сервируем отчёт по тестированию
- Page Header:** Confluence, Spaces, People, Calendars, Create, ...
- Page Header Buttons:** Search, Edit, Save for later, Watching, Share, ...

The right side of the page displays a hierarchical list of topics:

- 1 Результаты
  - 1.1 Объекты тестирования
  - 1.2 Цели и итоги тестирования
    - 1.2.1 Причины тестирования
    - 1.2.2 Виды тестирования производительности
    - 1.2.3 Дополнительные цели тестирования производительности
  - 1.3 Проверенные гипотезы
  - 1.4 Обнаруженные проблемы
  - 2 Результаты по видам тестирования производительности
    - 2.1 Нагрузочное тестирование (поиск максимума)
    - 2.2 Тестирование стабильности
    - 2.3 Объёмное тестирование
    - 2.4 Стressовое тестирование
  - 3 Архитектура и конфигурация системы
    - 3.1 Структура тестового стенда
    - 3.2 Конфигурация балансировщика нагрузки
    - 3.3 Конфигурация фронта
    - 3.4 Конфигурация сервера приложений
    - 3.5 Конфигурация сервера заглушек
    - 3.6 Конфигурация сервера баз данных
    - 3.7 Конфигурация нагрузочных станций
  - 4 Результаты синтетического тестирования узлов тестового стенда
  - 5 Рекомендации по конфигурации системы
    - 5.1 Балансировщик нагрузки
    - 5.2 Фронт
    - 5.3 Сервер приложений
    - 5.4 Сервер баз данных
    - 5.5 Нагрузочные станции
    - 5.6 Вспомогательные станции (заглушки, тестовая инфраструктура)
    - 5.7 Тестируемые операции
    - 5.8 Эмуляция работы пользователей системы
    - 5.9 Требования к производительности
      - 5.9.1 Требования к временем отклика
        - 5.9.1.1 Таблица. Требования к времени выполнения операций
        - 5.9.1.2 Таблица. Требования к времени выполнения операций при пиковом режиме работы
    - 5.10 Наполнение БД
    - 5.11 Профили нагрузки
      - 5.11.1 Таблица. Профиль стандартной нагрузки
      - 5.11.2 Таблица. Профиль нагрузки при пиковом режиме работы

# Выгрузка графиков из Grafana

Выверенные доски  
с гибкими настройками и фильтрами  
с отметками значимых событий и лимитов  
удобно автоматически выгружать, как набор картинок



# Подготовлен набор досок

Для инструментов, логов и метрик

Доски подготовлены самостоятельно

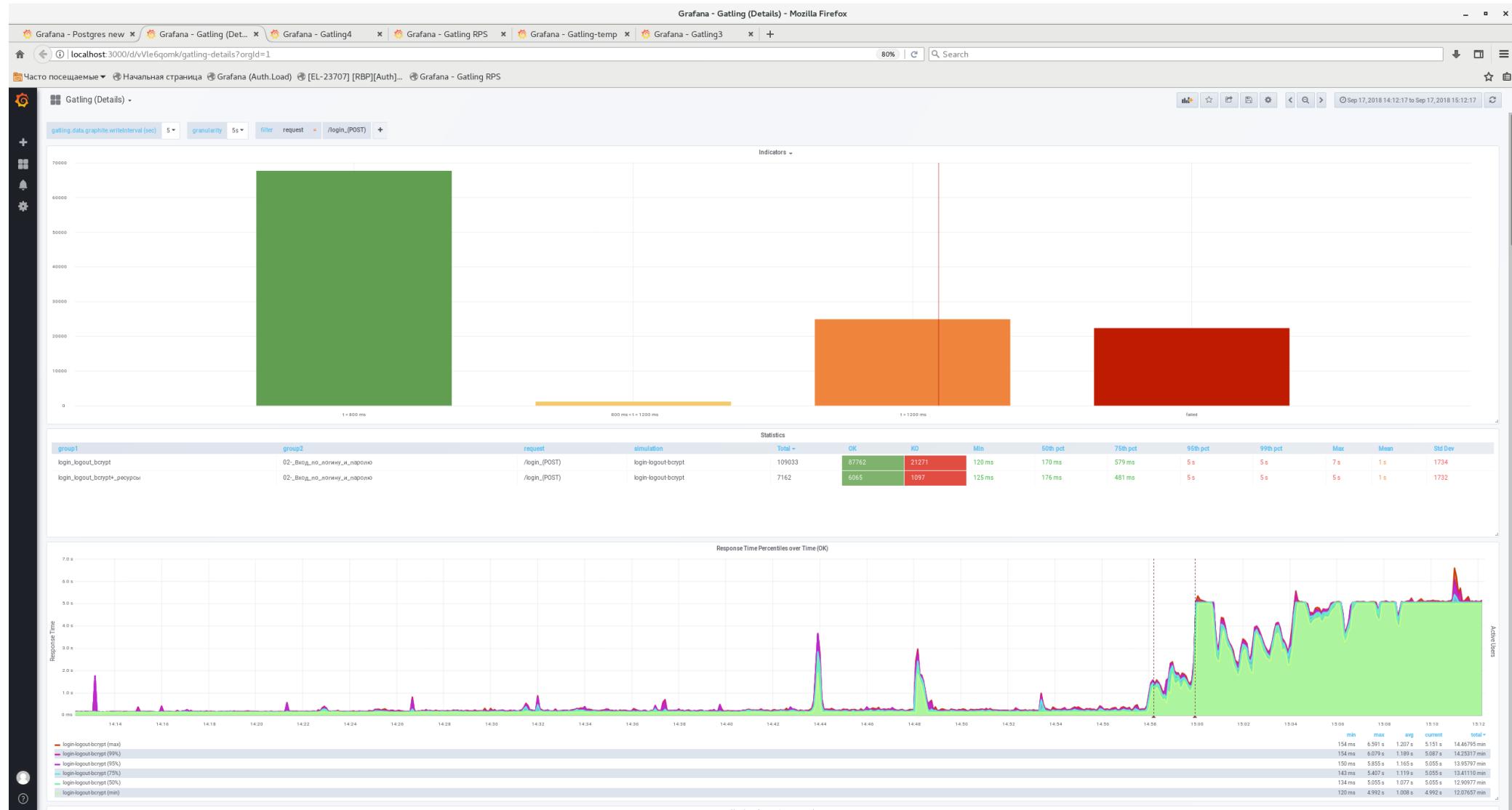
Сервируем отчёт по тестированию



- Gatling
- Gatling (Details)
- Gatling RPS
- Gatling-temp
- Gatling3
- Gatling4
- Log
- Auth Logs
- logs
- logs2
- General
  - HAproxy Pre-release haproxy
  - ibm mq
  - java
  - Postgres new
  - Processes
  - ProcStat
  - Server - Operating System Metrics linux

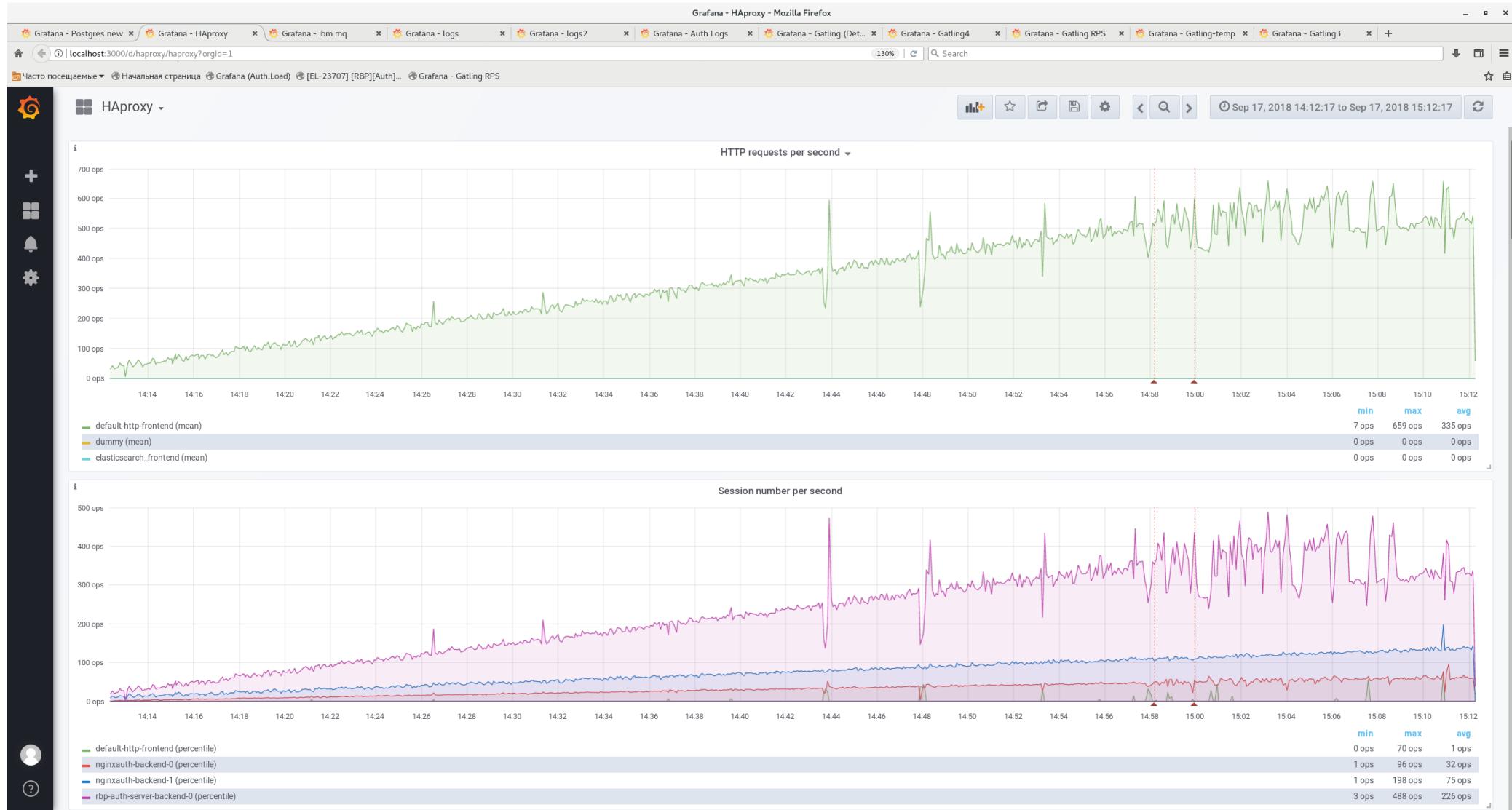


# Графики выверены



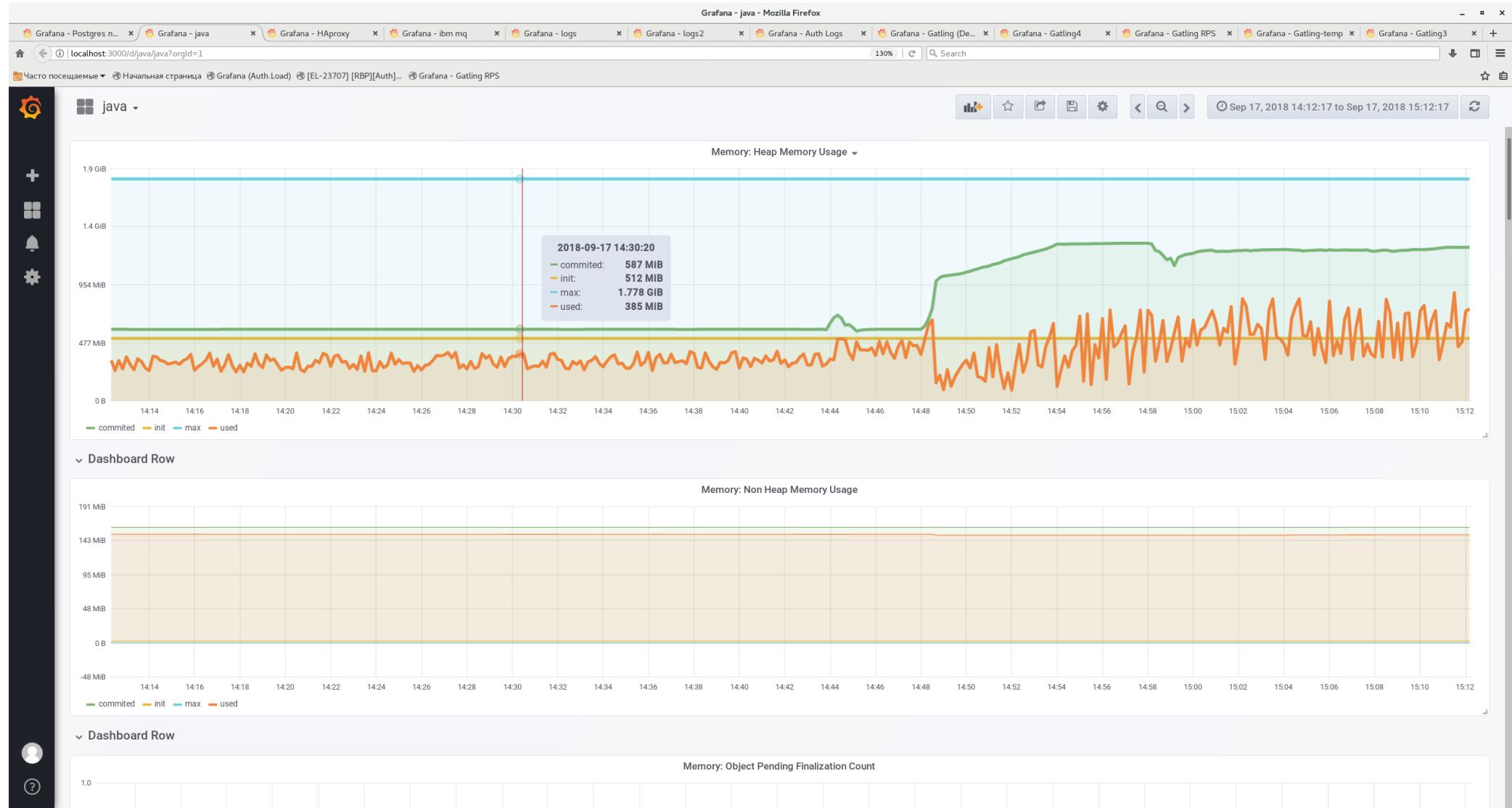


# Графики выверены



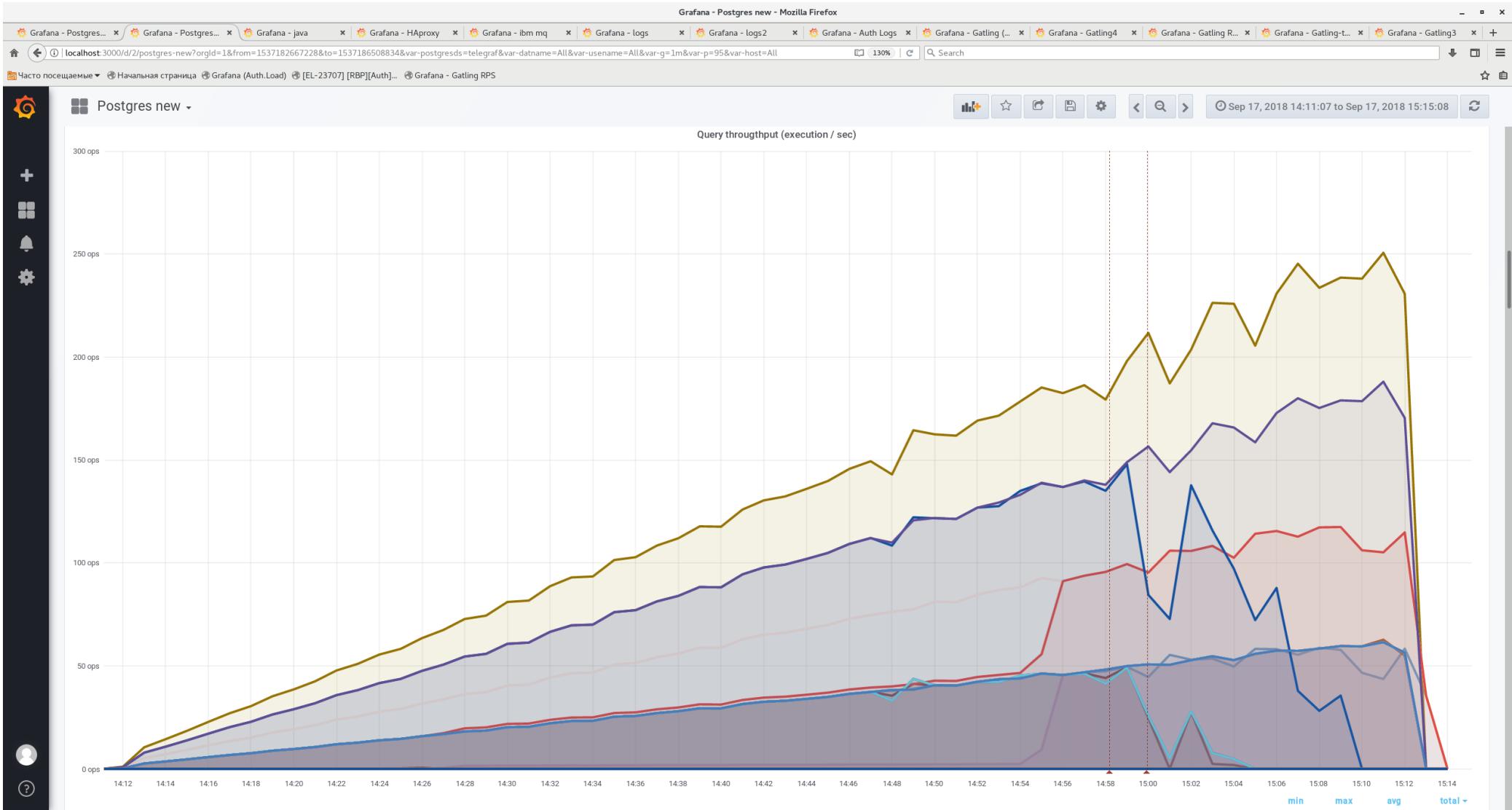


# Графики выверены



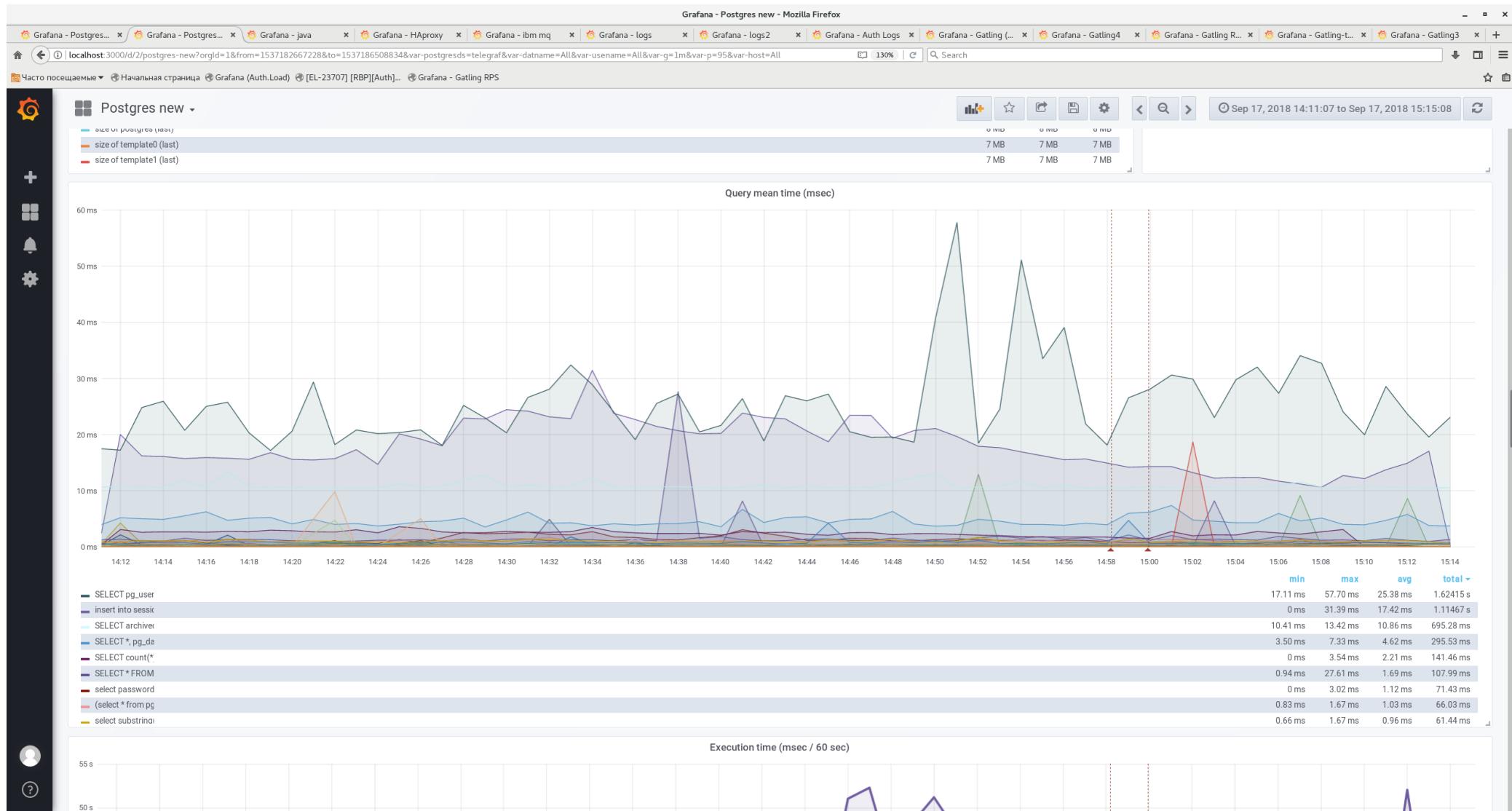


# Графики выверены



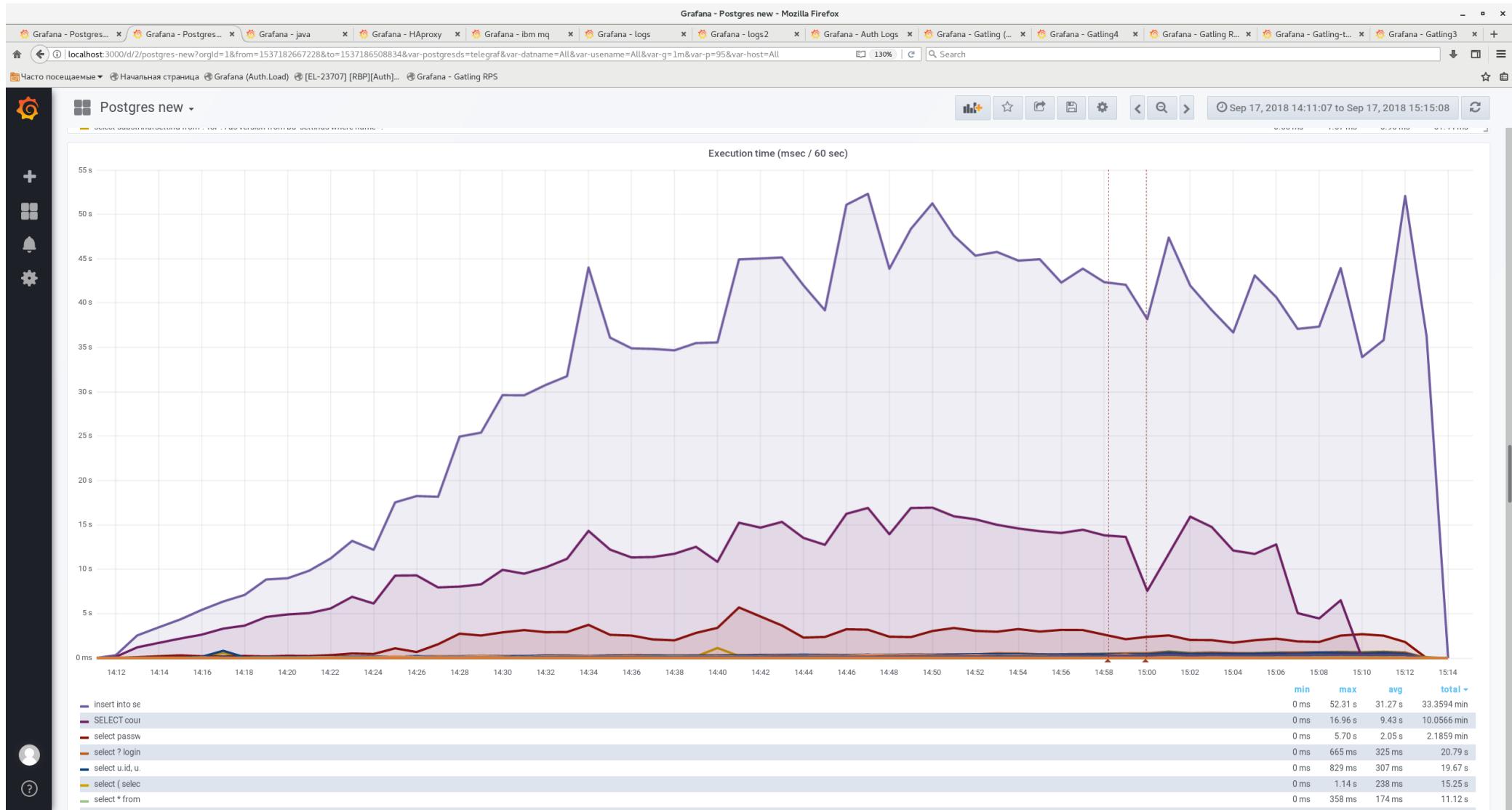


# Графики выверены



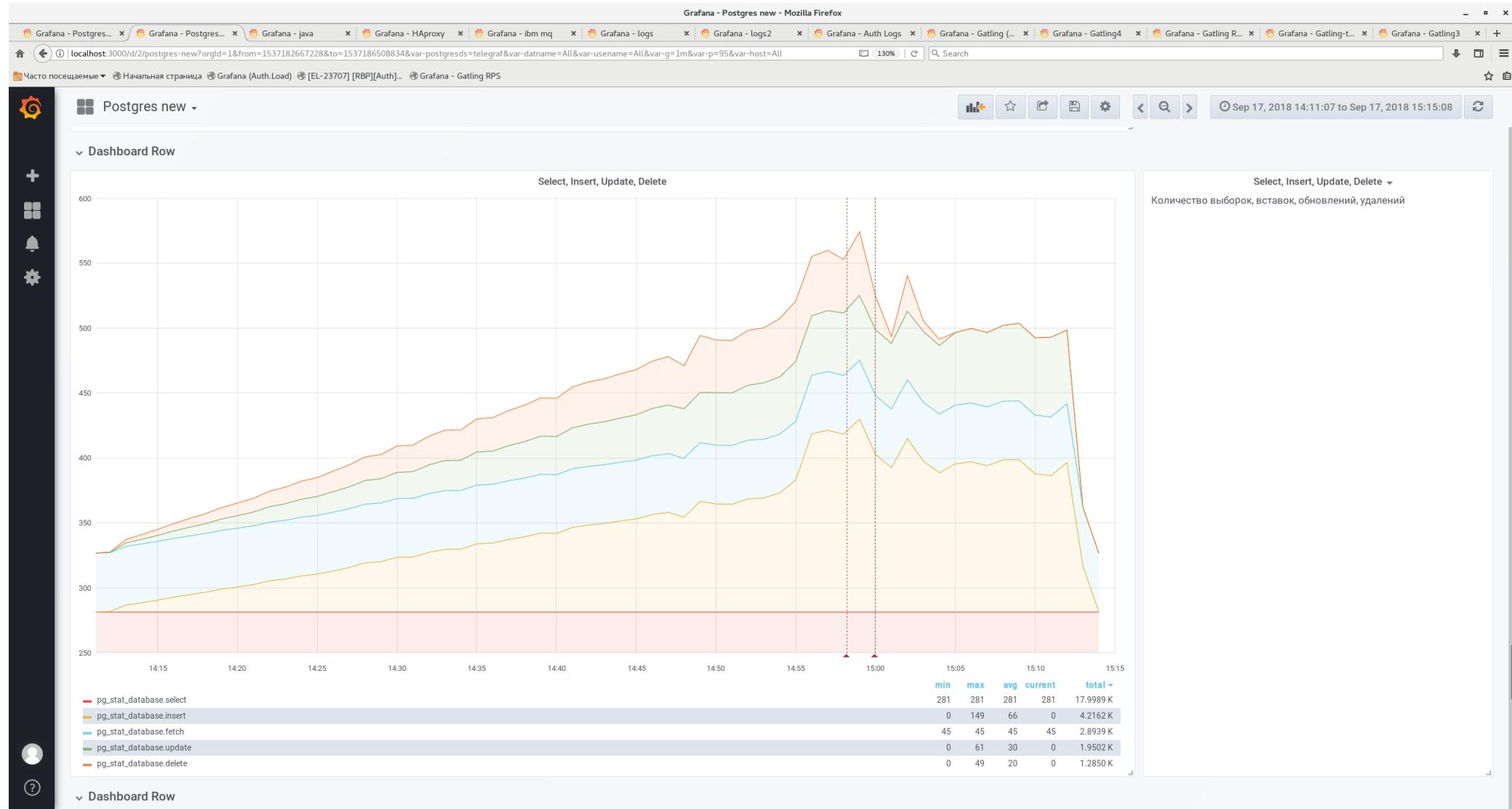


# Графики выверены



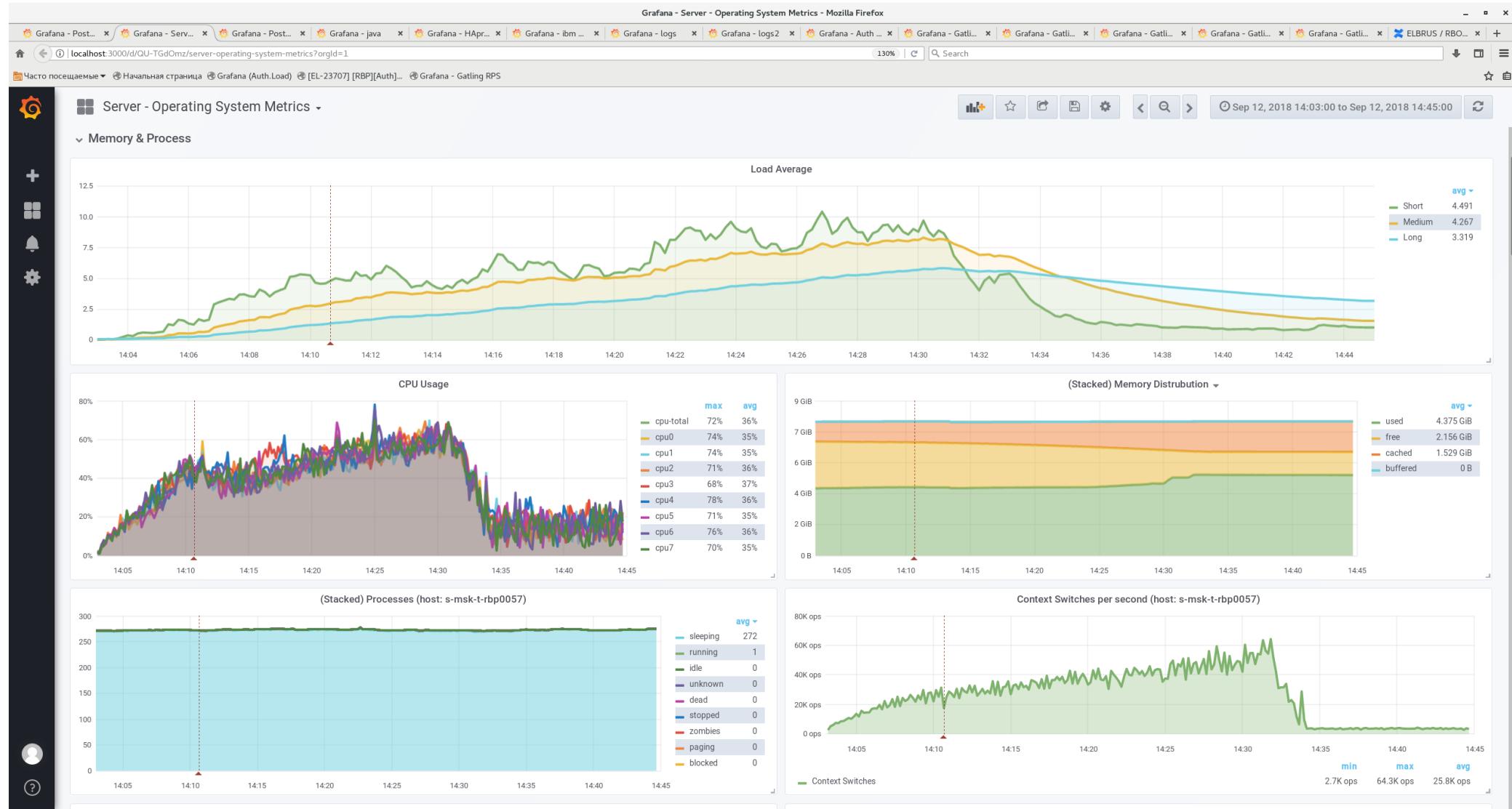


# Графики выверены





# Графики выверены



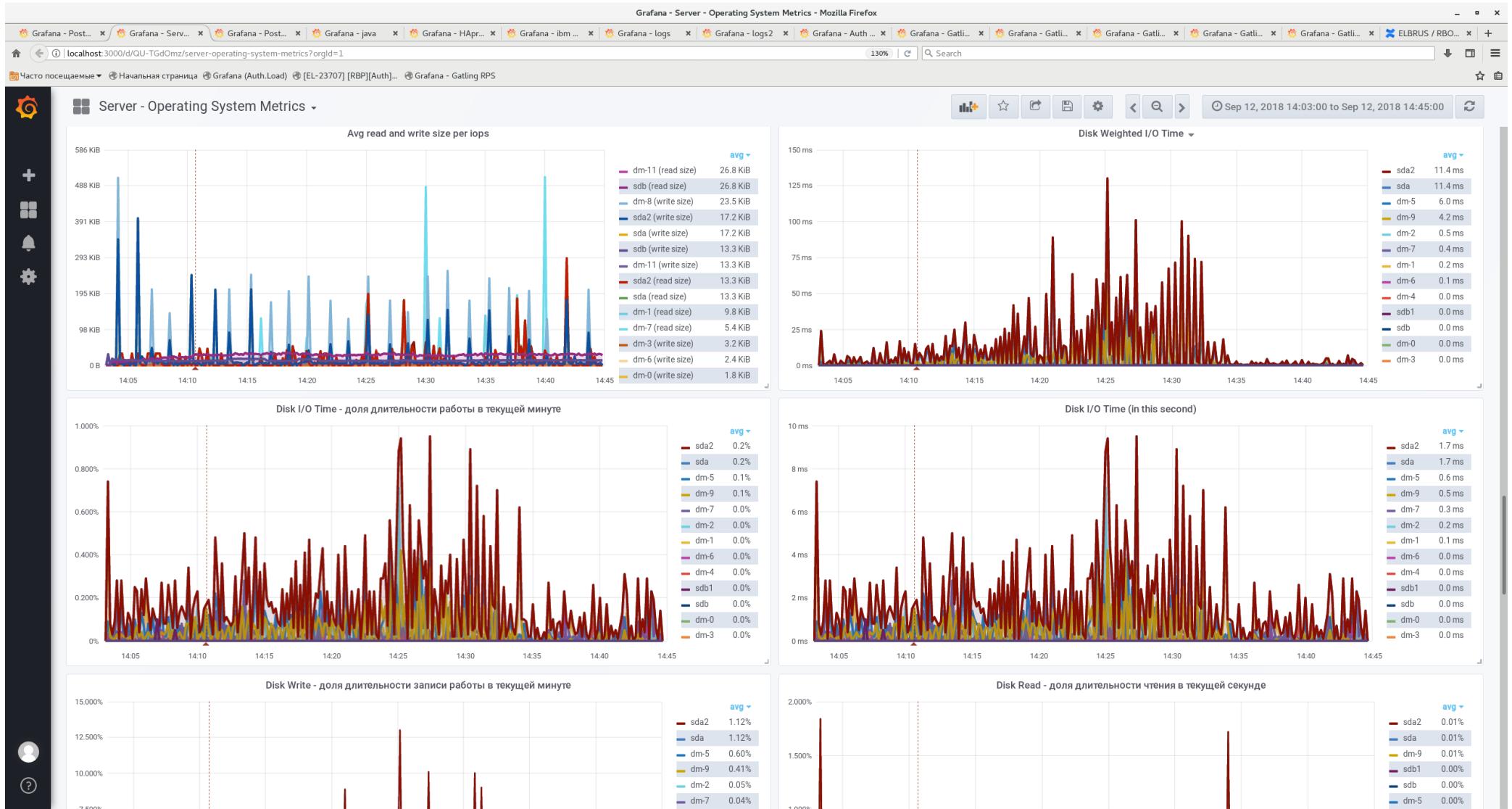


# Графики выверены



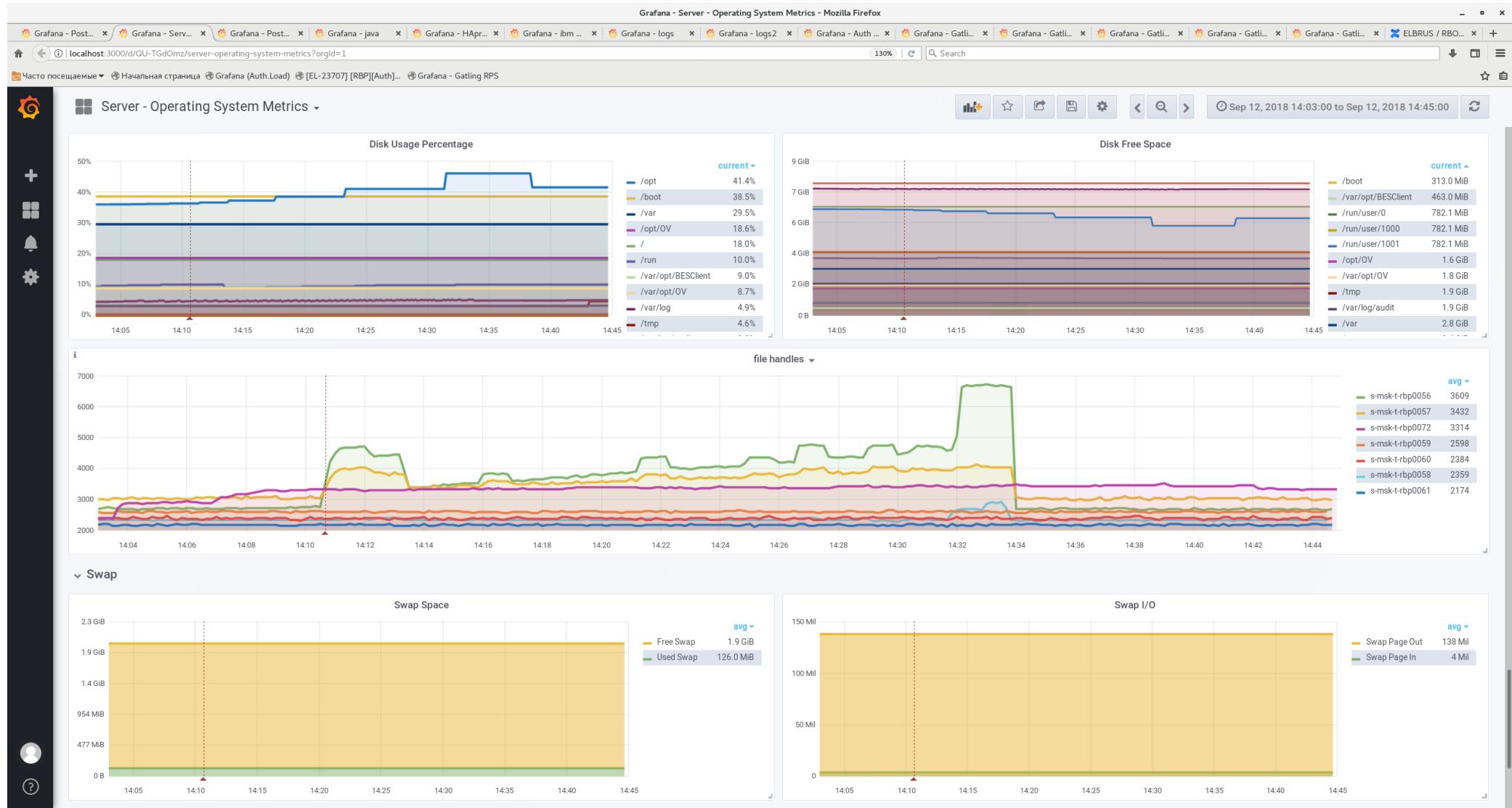


# Графики выверены





# Графики выверены





# Графики выгружаются автоматизировано

The screenshot shows a Confluence page titled 'get.image.sh'. The page contains a shell script with the following content:

```
#!/bin/bash
# объявляем переменную для хранения данных о графиках, которые будем сохранять в файл
declare -A graphs
# назначаем переменную host, для экспорта графиков по заданному хосту (передается в качестве параметра запуска, по умолчанию - s-msk-t-elweb10)
host=${1:-s-msk-t-elweb10}

# экспортируем графики из дашборда server-operating-system-metrics
# заполняем информацию о графиках, которые будем сохранять в файл (id и описание, кот. будет подставляться в имя файла)
graphs=( \
["32"]="CPU (stacked) for host (granularity)" \
["5"]="CPU Usage" \
["38"]="Memory used%" \
["26"]="Disk Usage Percentage" \
["28"]="Disk Weighted IO Time" \
)
# в цикле проходим по графикам и отправляем запрос на сохранение в файл
for i in "${!graphs[@]}"
do
curl "http://localhost:3000/render/dashboard-solo/db/server-operating-system-metrics?orgId=1&from=1530801600000&to=1530802800000&var-host=${host}&var-granularity=1m&panelId=${i}&width=1800&height=600&tz=U
-H 'Cookie: grafana_sess=40bcfd2c51cff94c; grafana_user=admin; grafana_remember=9979e21e0410d6f196c607b564c666217cf9199bcd4b57079480549b83d1a4fb6b'
-H 'Host: localhost:3000'
-H 'Connection: keep-alive'
--compressed
-o "${host}-os-${graphs[$i]}.png"
done

# экспортируем графики из дашборда procstat
graphs=( ["1"]="CPU Usage" ["3"]="Memory Usage")
for i in "${!graphs[@]}"
do
curl "http://localhost:3000/render/dashboard-solo/db/procstat?orgId=1&from=1530801600000&to=1530802800000&var-granularity=1m&var-host=${host}&var-process_name=java&var-process_name=rbp-notification-service
-H 'Cookie: grafana_sess=40bcfd2c51cff94c; grafana_user=admin; grafana_remember=9979e21e0410d6f196c607b564c666217cf9199bcd4b57079480549b83d1a4fb6b'
-H 'Host: localhost:3000'
-H 'Connection: keep-alive'
--compressed
-o "${host}-procstat-${graphs[$i]}.png"
done
```

Below the code, there is a note: 'Для получения id графиков Grafana можно экспортовать дашборд в json файл и выполнить следующую команду:' followed by a command line example: 'grep '"title":\|'"id":' ~/Downloads/RabbitMQ-1530886819301.json'

# Confluence для отчёта, InfluxDB для метрик

Все данные за несколько часов теста сохраняются в InfluxDB, из которого можно сделать выгрузку временного отрезка, а финальный результат останется в Confluence



## Выгрузка метрик

```
1 # Выгрузка
2 influx_inspect export -compress -database ""
  -datadir /var/lib/influxdb/data -waldir /
  var/lib/influxdb/wal -start 2018-09-17T14:
  12:17+03:00 -end 2018-09-17T15:12:17+03:00
  -out /tmp/backup.gz
```

3

```
4 # Загрузка
5 influx -import -compressed -path=./backup.gz
```

# Оформление отчёта по тестированию

Использование шаблонов и цепочек документов

Графики из Grafana можно выгружать автоматически

Confluence удобен для хранения отчёта, а диск для бекапов Influx

Нужна автоматизация, скорость и простота для регресса нагрузки



Спасибо