google-sheet-to-db

How to save data from google sheet to mysql database using google Apps script

The task is:

- 1. Create Mysql DB with any 10 columns
- 2. Write CRUD API server in node.js to manipulate data in created at step 1 DB via API
- 3. Create 10 columns table in Google sheet, fill 3 rows with dummy data
- 4. Write Google App Script application to connect spreadsheet with db via API created in step 2
- 5. You should be able to add/update all data from table to DB

Node.js application and mysql db server will be deployed on Red Hat Openshift cluster using openshift sendbox.

Google sheet and Google Apps Script will be on google. It is obvious

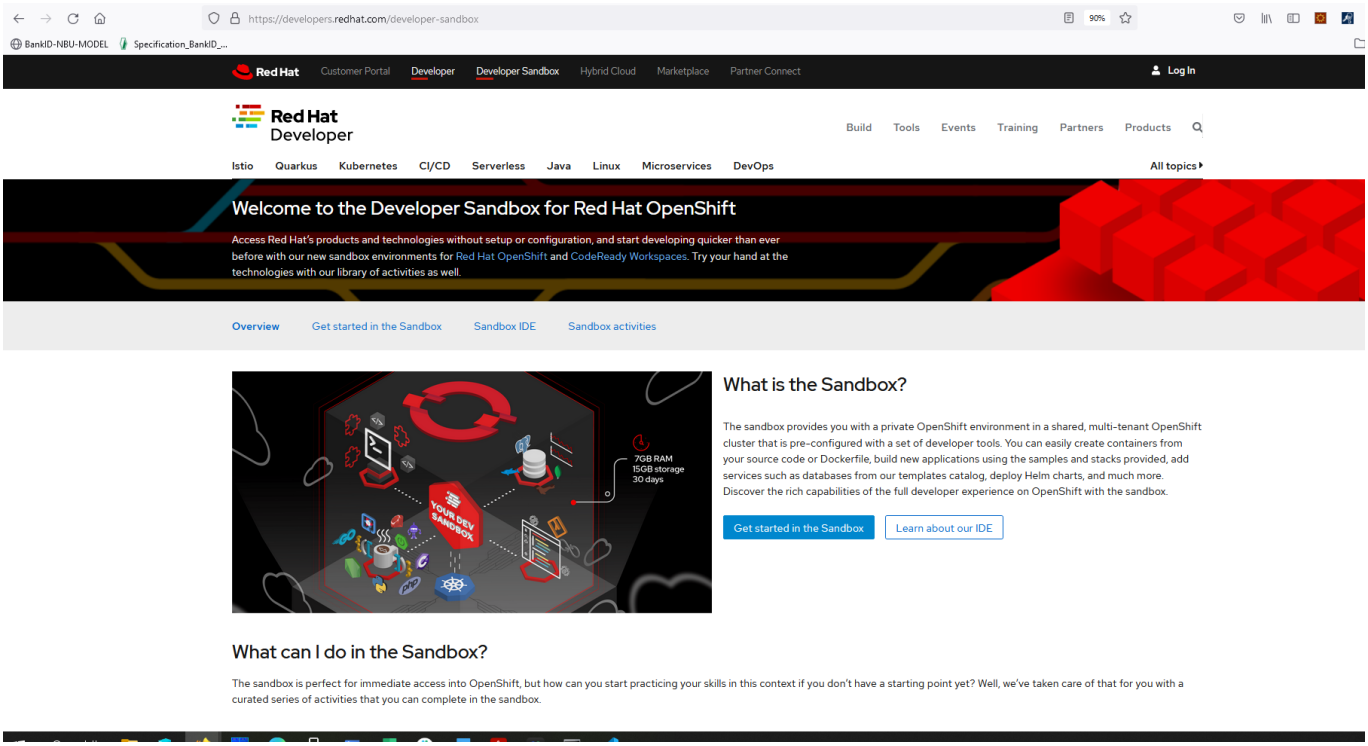
1. Create Mysql DB with any 10 columns

To acomplish this step, we must:

- create openshift sendbox;
- deploy into you senbox project MySQL server
- make conection from your laptop to MySQL server on openshift
- write DDL scritps end create database structure

create openshift sendbox

It is easy. You have to regester youself as developer on openshift sendbox pic-01.

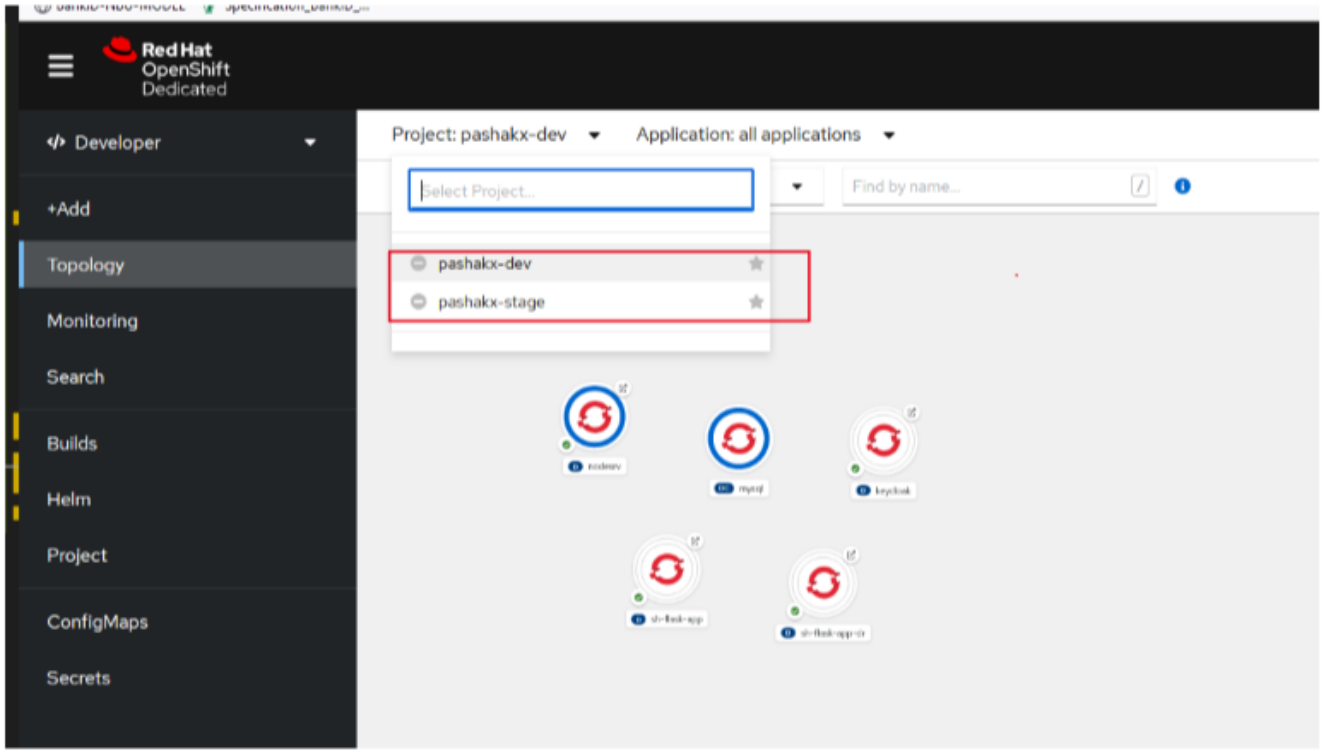


pic-01

As a result you will get openshift cluster with 2 projects (with 2 namespace in kubernetes terms).

> Q: What kind of resources do I get with my sandbox?
>
> A: Your private OpenShift environment includes two projects (namespaces) and a resource quota of 7 GB RAM, 15GB storage. The two namespaces can be used to emulate "development" and "stage" phases for your application.

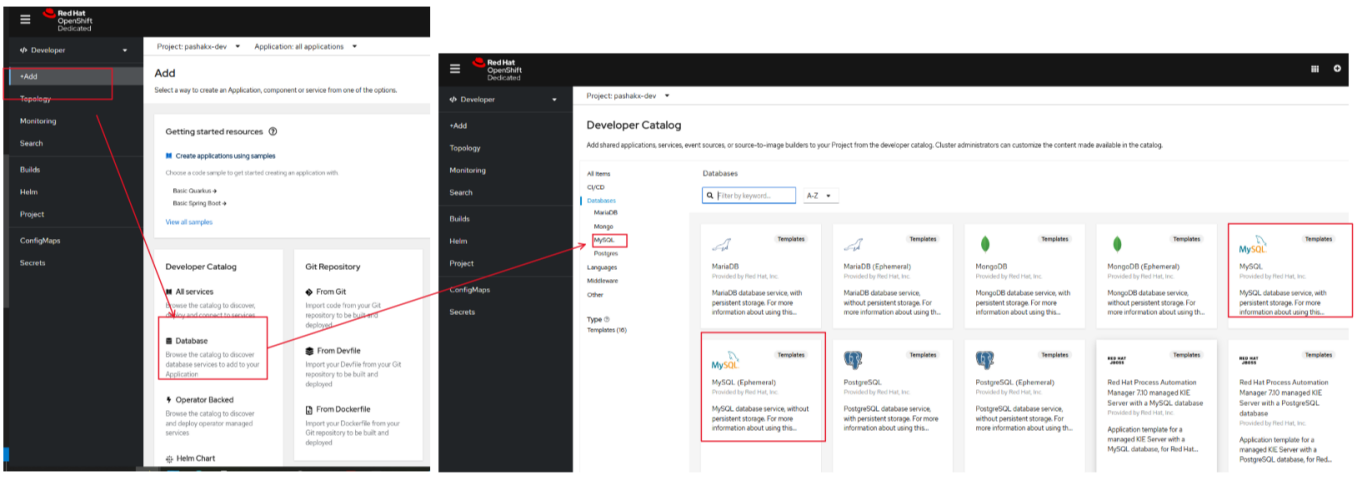
As an example, you can see on pic-02



pic-02

deploy into you senbox project MySQL server

Openshift have already had MySql database as an own template, pic-03.



pic-03

So MySQL server might be created from Openshift template using openshift CLI oc. The deployment script in filder [/openshift-deployment]](/openshift-deployment]).

- deploy database server: 1-create-mysql-db.cmd

```
rem
================================================================================
==============================================
rem Create MySql DB From OPenshift template
rem
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
rem get templates:          oc get templates -n openshift
rem get template's params:  oc process --parameters -n openshift mysql-persistent
rem
================================================================================
==============================================
rem      Parameters of mysql-persistent template
rem
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
rem NAME                    DESCRIPTION
GENERATOR           VALUE
rem MEMORY_LIMIT            Maximum amount of memory the container can use.
512Mi
rem NAMESPACE               The OpenShift Namespace where the ImageStream resides.
openshift
rem DATABASE_SERVICE_NAME   The name of the OpenShift Service exposed for the
database.                                 mysql
rem MYSQL_USER              Username for MySQL user that will be used for
accessing the database.   expression          user[A-Z0-9]{3}
rem MYSQL_PASSWORD          Password for the MySQL connection user.
expression          [a-zA-Z0-9]{16}
rem MYSQL_ROOT_PASSWORD     Password for the MySQL root user.
expression          [a-zA-Z0-9]{16}
rem MYSQL_DATABASE          Name of the MySQL database accessed.
sampledb
rem VOLUME_CAPACITY         Volume space available for data, e.g. 512Mi, 2Gi.
1Gi
rem MYSQL_VERSION           Version of MySQL image to be used (8.0-el7, 8.0-el8,
or latest).                           8.0-el8
rem
================================================================================
==============================================

call ..\login.cmd
oc project %APP_PROJ%

echo ============================================
echo Create MySQL DB
echo ============================================
```

```
pause

oc new-app --template=openshift/mysql-persistent --param=MEMORY_LIMIT=512Mi --
param=NAMESPACE=openshift --param=DATABASE_SERVICE_NAME=mysqldb --
param=MYSQL_USER=devadm --param=MYSQL_PASSWORD=22 --param=MYSQL_ROOT_PASSWORD=22 -
-param=VOLUME_CAPACITY=1Gi -l app=mysqldb

pause
```

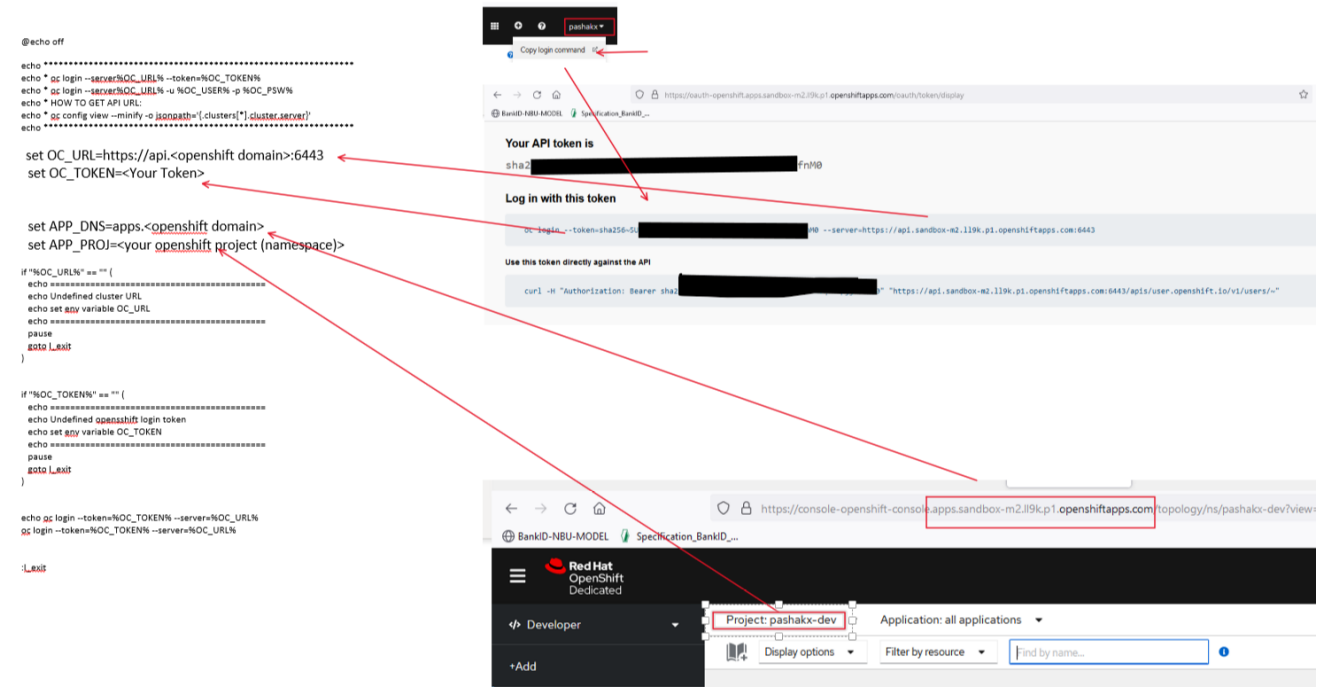
- delete deployment database server: 1-delete-mysql-db.cmd

```
rem
================================================================================
==============================================
rem  Delete MySql DB
rem
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
rem
================================================================================
==============================================

call ..\login.cmd
oc project %APP_PROJ%

oc delete all -l app=mysqldb
oc delete secret -l app=mysqldb

pause
```

Before running any deployments script you have to add some paramters in your **login.cmd**. Which parameters and from where is shown on pic-04.

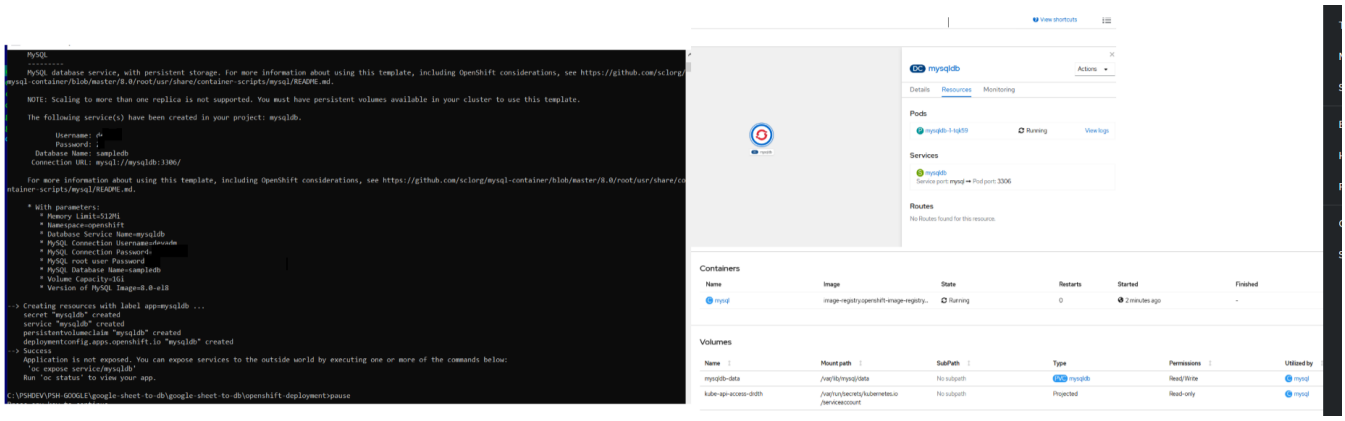
```
@echo off

echo ************************************************************
echo * oc login --server%OC_URL% --token=%OC_TOKEN%
echo * oc login --server%OC_URL% -u %OC_USER% -p %OC_PSW%
echo * HOW TO GET API URL:
echo * oc config view --minify -o jsonpath='{.clusters[*].cluster.server}'
echo ************************************************************

 set OC_URL=https://api.<openshift domain>:6443
 set OC_TOKEN=<Your Token>
```

```
 set APP_DNS=apps.<openshift domain>
 set APP_PROJ=<your openshift project (namespace)>

if "%OC_URL%" == "" (
  echo ============================================
  echo Undefined cluster URL
  echo set env variable OC_URL
  echo ============================================
  pause
  goto l_exit
)


if "%OC_TOKEN%" == "" (
  echo ============================================
  echo Undefined opensshift login token
  echo set env variable OC_TOKEN
  echo ============================================
  pause
  goto l_exit
)


echo oc login --token=%OC_TOKEN% --server=%OC_URL%
oc login --token=%OC_TOKEN% --server=%OC_URL%


:l_exit
```



pic-04

Deployment result is shown on pic-05.

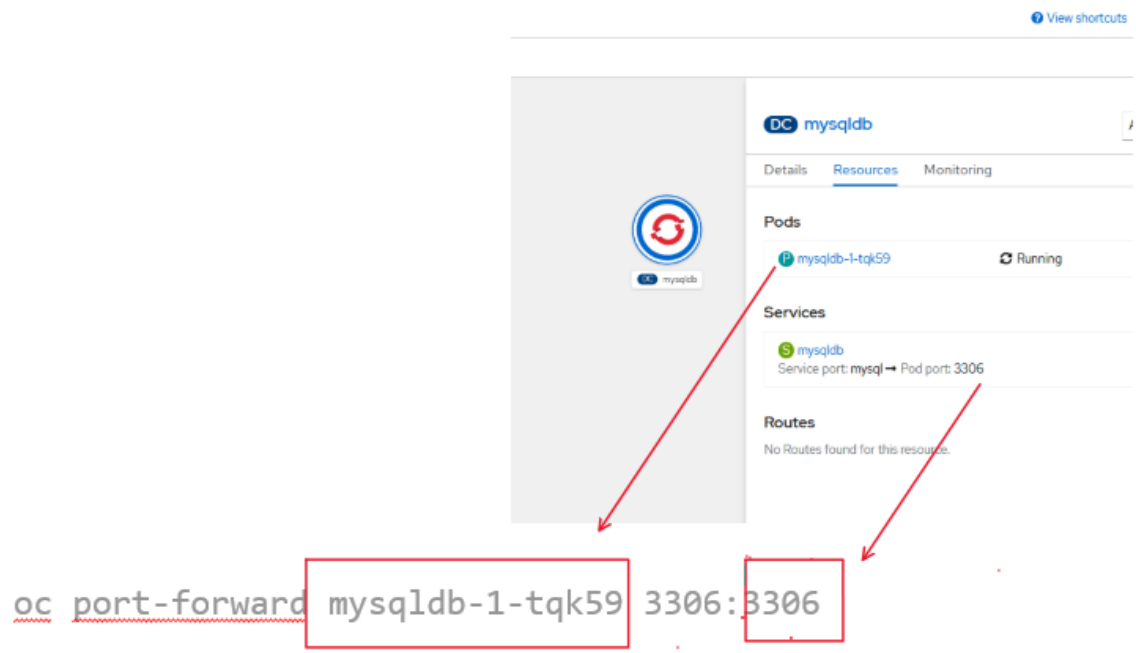


pic-05

Make conection from your laptop to MySQL server on openshift

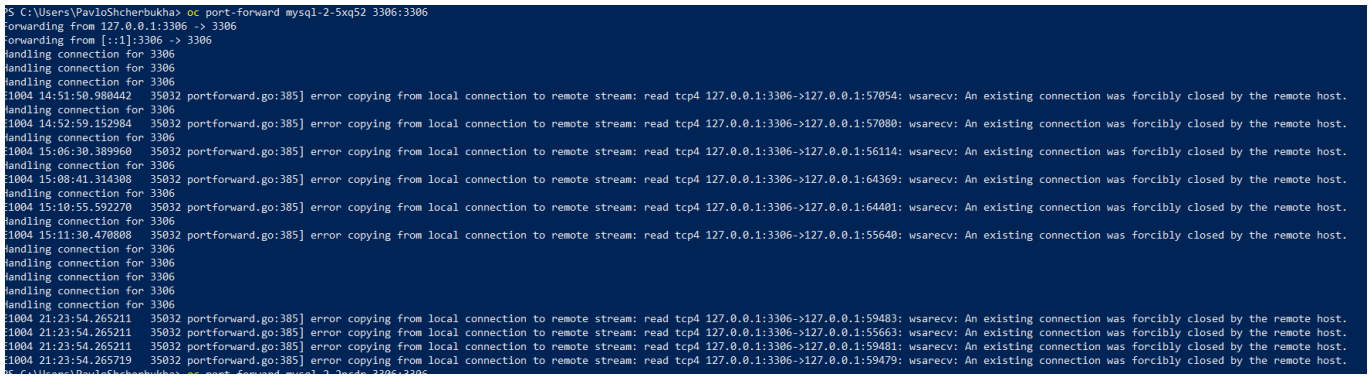
We can connect to db using port forward command pic-06.

```
# oc port-forward  <your pod>  <your local port> : <your remote port>
oc login --token=%OC_TOKEN% --server=%OC_URL%
oc project <your project>
oc port-forward mysqldb-1-tqk59 3306:3306
```



pic-06

If you run this commands from PowerShell you will see something like on pic-7.



pic-07

Let's check connection from your laptop to MySQL on openshift, using **mysql.exe**. I had stopped my local MySQL server before action which is described below. You can find in folder **ddl** file **mysqlrun2.cmd** which make connection to database server as a root. In case of successful connection I must run ddl **db-grn.sql** which grant permittion for user **devadm**.

```
C:\PSHDEV\PSH-GOOGLE\google-sheet-to-db\google-sheet-to-db\ddl>mysqlrun2.cmd
=========================================================
RUN  DDL, DML from  MySQL CLI
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

C:\PSHDEV\PSH-GOOGLE\google-sheet-to-db\google-sheet-to-db\ddl>mysql.exe  -uroot -
p22 --default-character-set=utf8mb4 -v --port 3306
mysql: [Warning] Using a password on the command line interface can be insecure.
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 383
Server version: 8.0.21 Source distribution

Copyright (c) 2000, 2021, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> source db-grn.sql;
--------------
grant super on *.* to 'devadm'@'%'
--------------

Query OK, 0 rows affected, 1 warning (0.18 sec)

--------------
show grants for 'devadm'@'%'
--------------

+-----------------------------------------------------+
| Grants for devadm@%                                 |
```

```
+-----------------------------------------------------+
| GRANT SUPER ON *.* TO `devadm`@`%`                  |
| GRANT ALL PRIVILEGES ON `sampledb`.* TO `devadm`@`%` |
+-----------------------------------------------------+
2 rows in set (0.18 sec)

mysql>
```

Then, you can connect under user 'DEVADM' and create database. Then, you can connect under user 'DEVADM' and create database. Connection under **DEVADM** will be created using **mysqlrun1.cmd**. Database structure stored in **db-build.sql**. Let's run it:

```
C:\PSHDEV\PSH-GOOGLE\google-sheet-to-db\google-sheet-to-db\ddl>mysqlrun1.cmd
=========================================================
RUN  DDL, DML from  MySQL CLI
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

C:\PSHDEV\PSH-GOOGLE\google-sheet-to-db\google-sheet-to-db\ddl>mysql.exe  -udevadm
-p22 --default-character-set=utf8mb4 -v --port 3306
mysql: [Warning] Using a password on the command line interface can be insecure.
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 485
Server version: 8.0.21 Source distribution

Copyright (c) 2000, 2021, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> source db-build.sql;


mysql> source db-build.sql;
Logging to file 'db-build.log'
--------------
drop database IF EXISTS test4
--------------

Query OK, 0 rows affected, 1 warning (0.18 sec)

--------------
create database test4
--------------

Query OK, 1 row affected (0.18 sec)
```

```
--------------
show databases
--------------

+--------------------+
| Database           |
+--------------------+
| information_schema |
| sampledb           |
| test4              |
+--------------------+
3 rows in set (0.17 sec)

Database changed
--------------
CREATE TABLE APP2$EMP
(
    IDREC    INT AUTO_INCREMENT,
.
.
.
.
.
.
```

Also, let's insert test data into the table **APP2$EMP**. Run script in the file **data-ins1.sql**:

```
mysql> source data-ins1.sql;
Logging to file 'data-ins1.log'
Database changed
--------------
delete from APP2$EMP
--------------

Query OK, 0 rows affected (0.17 sec)

--------------
insert into APP2$EMP( CODEBRN,
                      NAMEBRN,
                      TABNUM ,
                      FAM,
                      IM,
                      OTCH,
                      ADRESS,
                      MSTATUS,
                      COUNTRY,
                      DS,
                      DF)
VALUES (
```

```
                        '00',
                        'ГОЛОВНИЙ',
                        '00001',
                        'ВАСЕЧКИН',
                        'ПЕТРО',
                        'ПЕТРОВИЧ',
                        'На розі біля цирку',
                        'M',
                        'UA',
                        '2005-03-09',
.
.
.
.
.
.

--------------
select A.* from  APP2$EMP A
--------------

+-------+---------+------------------------+--------+------------------+----------
----------+----------------------+-----------------------------------+---------+--
-------+------------+------+--------------------+------------+------+--------+
| IDREC | CODEBRN | NAMEBRN                | TABNUM | FAM              | IM
| OTCH                | ADRESS                            | MSTATUS | COUNTRY |
DS          | DF    | IDT                 | IUSRNM     | MDT  | MUSRNM |
+-------+---------+------------------------+--------+------------------+----------
----------+----------------------+-----------------------------------+---------+--
-------+------------+------+--------------------+------------+------+--------+
|     1 | 00      | ГОЛОВНИЙ               | 00001  | ВАСЕЧКИН         | ПЕТРО
| ПЕТРОВИЧ            | На розі біля цирку                | M       | UA      |
2005-03-09 | NULL | 2021-10-04 21:54:26 | devadm@::1 | NULL | NULL   |
|     2 | 00      | ГОЛОВНИЙ               | 00002  | ПЕТРЕНКО         | СЕМЕН
| СЕМЕНОВИЧ           | БІЛЯ ПАРКУ                        | M       | UA      |
2007-02-03 | NULL | 2021-10-04 21:54:27 | devadm@::1 | NULL | NULL   |
|     3 | 01      | ЦЕНТРАЛЬНИЙ            | 00003  | САЄНКО           | МАРГАРИТА
| СЕРГІЇВНА           | БІЛЯ ТЕАТРУ                       | M       | UA      |
2007-02-03 | NULL | 2021-10-04 21:54:27 | devadm@::1 | NULL | NULL   |
|     4 | 01      | ЦЕНТРАЛЬНИЙ            | 00004  | ДУДКА            | АНАСТАСІЯ
| ВІКТОРІВНА          | БІЛЯ ТЕАТРУ                       | S       | UA      |
2019-07-23 | NULL | 2021-10-04 21:54:27 | devadm@::1 | NULL | NULL   |
+-------+---------+------------------------+--------+------------------+----------
----------+----------------------+-----------------------------------+---------+--
-------+------------+------+--------------------+------------+------+--------+
4 rows in set (0.17 sec)

mysql>
```

Finally, the database **test4** created. Test data inserted. So, this step is finished.

2. Write CRUD API server in node.js to manipulate data in created at step 1 DB via API

Node.js server developed as a simple Node.js express application. There are 2 routers:

- "/api/v1/emp" implemented CRUD operations on one record from table APP2$EMP;
- "/api/v1/emps" implemented get/delete for all records table APP2$EMP.

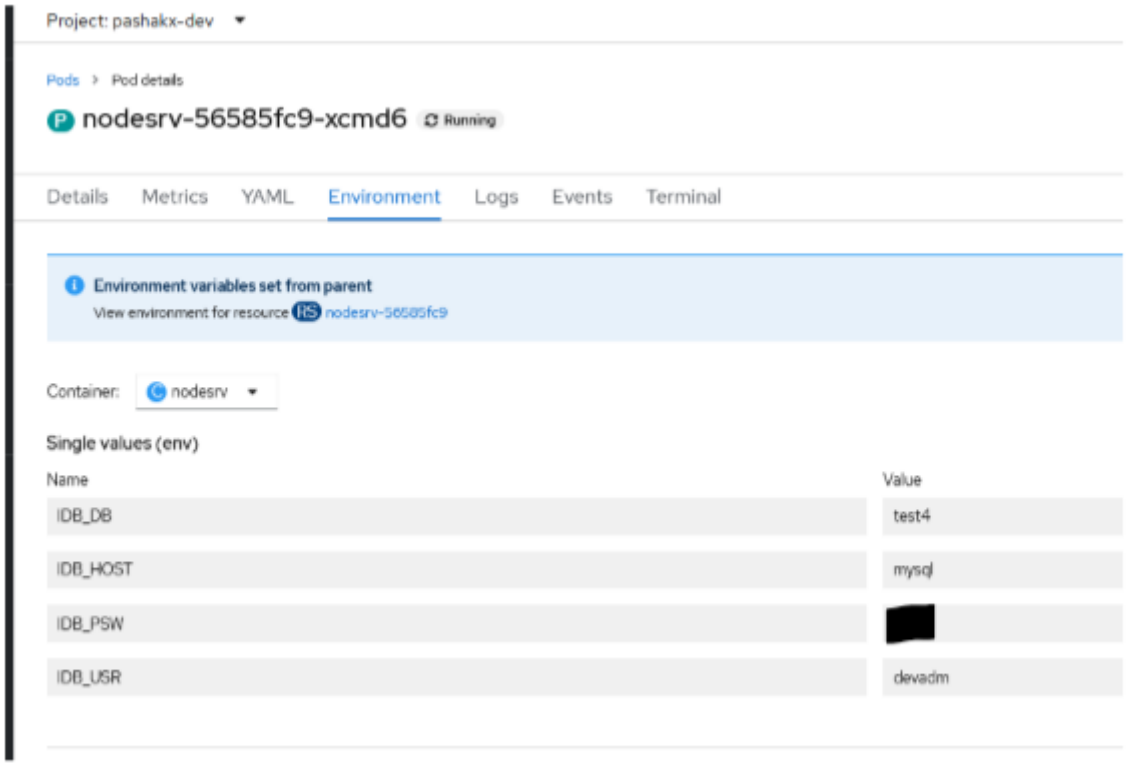
File ./config/local.json - contains number of local port which server will be listened.

```
{
  "port": 8080
}
```

File ./config/mapping.json contains the list of environment variables which are needed and search patterns for different environments. In case of running on your laptop all environment variables shuold be stored in the file **./localdev-config.json**.

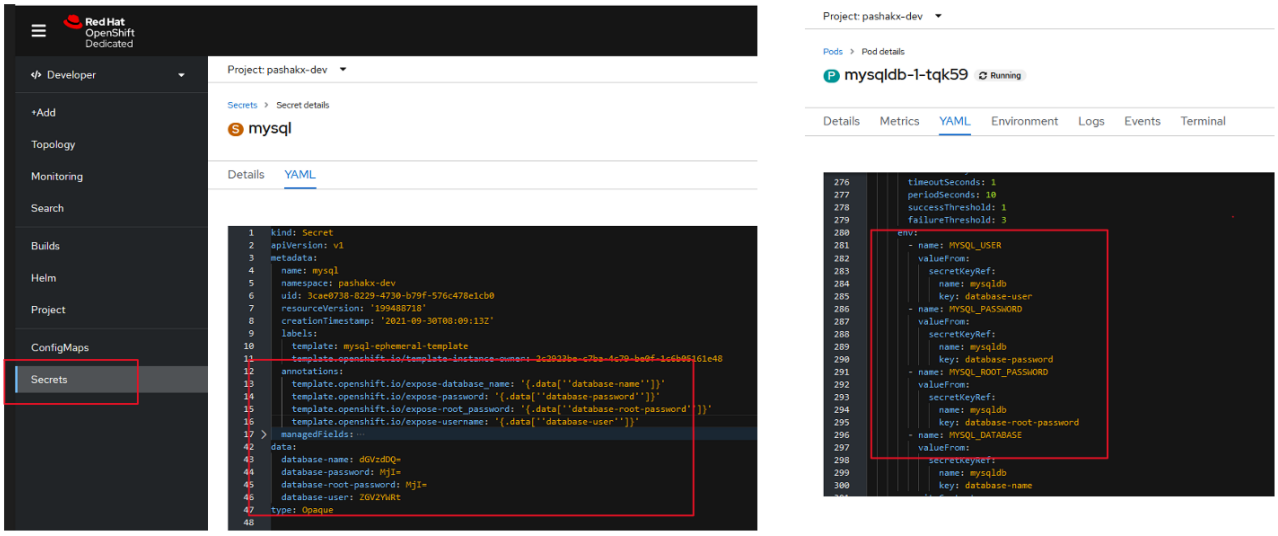
```
{
 "IDB_HOST": "localhost",
 "IDB_DB": "test4",
 "IDB_USR": "devadm",
 "IDB_PSW": "***********"
}
```

In case deployment on openshift parameters are stored in env variables .



pic-08

In addition, parameters of the database could be stored in openshift secrets sore pic-09.



pic-09

run server on your laptop

- clone repository from github usig

```
git clone https://github.com/pavlo-shcherbukha/google-sheet-to-db.git -b master
```

- install dependency

```
npm install
```

- correct config files: ./config/local.json and ./localdev-config.json.
- run **oc port-forward** command in order to establish connection from your localhost to database in Openshift.

```
# oc port-forward  <your pod>  <your local port> : <your remote port>
oc login --token=%OC_TOKEN% --server=%OC_URL%
oc project <your project>
oc port-forward mysqldb-1-tqk59 3306:3306
```

- run application using

```
npm start
```

run server on openshift

The application could be deployed in openshift using deployment script in file **openshift-deployment/2-crt_nodesrv-app.cmd** using openshift CLI directly from your github repository. The application could be deleted using script in file **openshift-deployment/2-del_nodesrv-app.cmd**. You can use the Red Hat Software Collections images as a foundation for applications that rely on specific runtime environments such as Node.js, Perl, or Python. Special versions of some of these runtime base images are referred to as Source-to-Image (S2I) images. With S2I images, you can insert your code into a base image environment that is ready to run that code.

Before deployment you have to correct **openshift-deployment/2-crt_nodesrv-app.cmd**:

- git repo url
- branch name
- environment variables
- in case using private git repo, you have to create secret in order to access it like this:

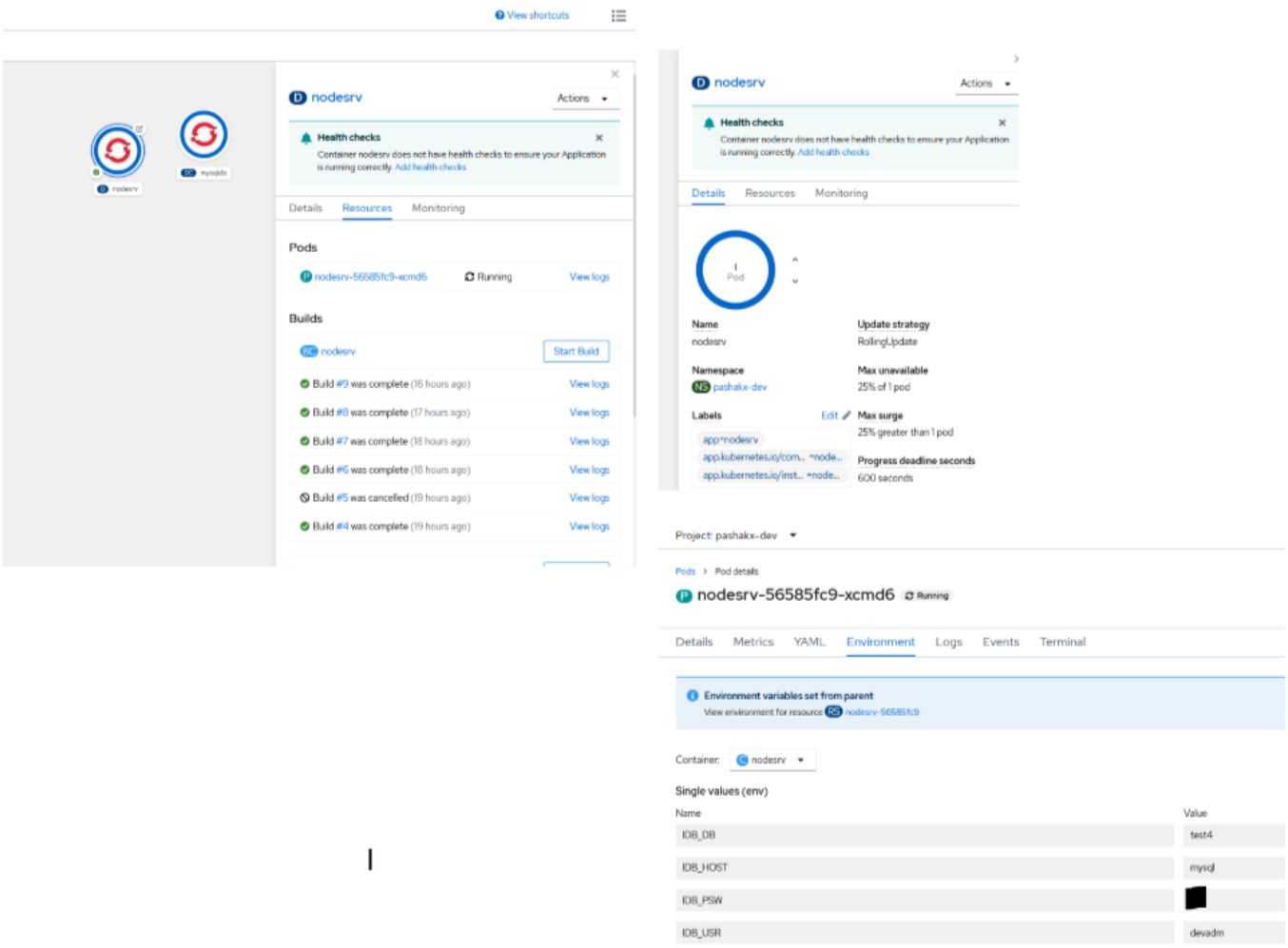
```
echo ****************************************
echo *    create secret for github
echo *
echo ****************************************

oc create secret generic sinc-gitlab-pvx-1 --from-literal=username=<your username>
--from-literal=password=<your githyb dev token>

oc secrets link deployer sinc-gitlab-pvx-1
oc secrets link builder sinc-gitlab-pvx-1

oc annotate secret sinc-gitlab-pvx-1 "build.openshift.io/source-secret-match-uri-
1=https://github.com/<your username>/*"
```

- correct route URL in file **openshift-deployment/2-crt_nodesrv-app.cmd**:

Openshift router is something like http balancer, if you run more then one instance of you application. The structure of the external URL like this: http:// - . "

- run **openshift-deployment/2-crt_nodesrv-app.cmd**

As the result, you will get something like pic-10.



pic-10

- test api using test cases in folder **./node-server/test/mysql-api**.

File **test-emp-api.js** for testing "/api/v1/emp". File **test-emps-api.js** for testing "/api/v1/emps".

Before run test cases set up correct base url:

```
//let i_baseurl = 'http://localhost:8080';
let i_baseurl = 'http://nodesrv-pashakx-dev.apps.sandbox-
m2.ll9k.p1.openshiftapps.com';
```

```
  Тестовые кейсы на сервис /api/v1/emps
Ответ:
{"status":200,"error":null,"response":[{"IDREC":1,"CODEBRN":"10","NAMEBRN":"Head
office","TABNUM":"10001","FAM":"Burleson","IM":"Janet","OTCH":"Jone","ADRESS":"For
est street","MSTATUS":"N","COUNTRY":"UA","DS":"2019-02-
21T00:00:00.000Z","DF":null,"IDT":"2021-10-
04T18:40:29.000Z","IUSRNM":"devadm@10.128.2.132","MDT":"2021-10-
04T19:18:28.000Z","MUSRNM":"devadm@10.128.2.132"},
{"IDREC":2,"CODEBRN":"10","NAMEBRN":"Head
```

office","TABNUM":"10002","FAM":"BlackCat","IM":"Michael","OTCH":"Laurent","ADRESS"
:"Yellow Hill","MSTATUS":"M","COUNTRY":"GB","DS":"2019-05-
21T00:00:00.000Z","DF":null,"IDT":"2021-10-
04T18:41:25.000Z","IUSRNM":"devadm@10.128.2.132","MDT":"2021-10-
04T19:18:28.000Z","MUSRNM":"devadm@10.128.2.132"},
{"IDREC":3,"CODEBRN":"12","NAMEBRN":"Summer
Branch","TABNUM":"10003","FAM":"Craft","IM":"Joann","OTCH":"Larries","ADRESS":"Kai
ro street","MSTATUS":"N","COUNTRY":"EG","DS":"2019-05-
21T00:00:00.000Z","DF":null,"IDT":"2021-10-
04T18:41:39.000Z","IUSRNM":"devadm@10.128.2.132","MDT":"2021-10-
04T19:18:28.000Z","MUSRNM":"devadm@10.128.2.132"},
{"IDREC":4,"CODEBRN":"11","NAMEBRN":"Winter
Branch","TABNUM":"10004","FAM":"Ellison","IM":"Larry","OTCH":"Joseph","ADRESS":"Ne
w York City","MSTATUS":"M","COUNTRY":"US","DS":"2021-05-
21T00:00:00.000Z","DF":null,"IDT":"2021-10-
04T18:41:44.000Z","IUSRNM":"devadm@10.128.2.132","MDT":"2021-10-
04T19:18:28.000Z","MUSRNM":"devadm@10.128.2.132"},
{"IDREC":6,"CODEBRN":"12","NAMEBRN":"Spring
Branch","TABNUM":"1006","FAM":"Green","IM":"Edvard","OTCH":"Jonson","ADRESS":"SanD
iego","MSTATUS":"M","COUNTRY":"US","DS":"2018-02-
01T00:00:00.000Z","DF":null,"IDT":"2021-10-
04T19:16:49.000Z","IUSRNM":"devadm@10.128.2.132","MDT":"2021-10-
04T19:18:28.000Z","MUSRNM":"devadm@10.128.2.132"},
{"IDREC":7,"CODEBRN":"12","NAMEBRN":"Spring
Branch","TABNUM":"1007","FAM":"EverGreen","IM":"Bob","OTCH":"Edvard","ADRESS":"San
Diego","MSTATUS":"M","COUNTRY":"US","DS":"2018-03-
01T00:00:00.000Z","DF":null,"IDT":"2021-10-
04T19:18:28.000Z","IUSRNM":"devadm@10.128.2.132","MDT":null,"MUSRNM":null},
{"IDREC":8,"CODEBRN":"03","NAMEBRN":"Чернигов","TABNUM":"000","FAM":"updatedfam","
IM":"Петро","OTCH":"Петролвич","ADRESS":"На
галявині","MSTATUS":"N","COUNTRY":"UA","DS":"2020-03-
19T00:00:00.000Z","DF":null,"IDT":"2021-10-
05T06:03:10.000Z","IUSRNM":"devadm@10.128.3.116","MDT":"2021-10-
05T06:03:10.000Z","MUSRNM":"devadm@10.128.3.116"}]}
    ✔ GET /api/v1/emps/ - Ожидаем ответ 200. Прочитать все записи из таблицы (395ms)
Ответ:
{"status":200,"error":null,"response":
{"fieldCount":0,"affectedRows":7,"insertId":0,"info":"","serverStatus":34,"warning
Status":0}}
    ✔ DELTE /api/v1/emps - Ожидаем ответ 200. Удалить все записи из таблицы (344ms)
Ответ:
{"status":200,"error":null,"response":[]}
    ✔ GET /api/v1/emps/ - Ожидаем ответ 200. Прочитать все записи из таблицы после удаления. Ождаем 0 записей === пустой массив (386ms)

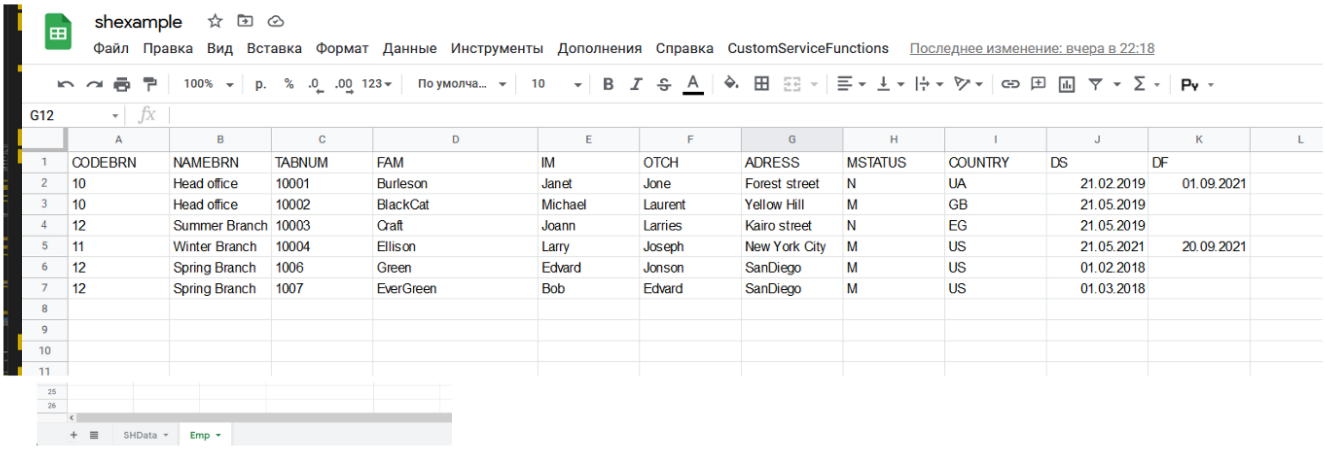
  3 passing (1s)

```
  Тестовые кейсы на сервис /api/v1/emp
Запрос:
{"CODEBRN":"03","NAMEBRN":"Чернигов","TABNUM":"000","FAM":"Петренко","IM":"Петро",
"OTCH":"Петролвич","ADRESS":"На галявині","MSTATUS":"N","COUNTRY":"UA","DS":"2020-
03-19"}
Ответ:
{"status":200,"error":null,"response":
{"fieldCount":0,"affectedRows":1,"insertId":10,"info":"","serverStatus":2,"warning
Status":0}}
    ✔ POST /api/v1/emp - Ожидаем ответ 200. Запись создана в БД (507ms)
Запрос: :id=000
Ответ:
{"status":200,"error":null,"response":
[{"IDREC":10,"CODEBRN":"03","NAMEBRN":"Чернигов","TABNUM":"000","FAM":"Петренко","
IM":"Петро","OTCH":"Петролвич","ADRESS":"На
галявині","MSTATUS":"N","COUNTRY":"UA","DS":"2020-03-
19T00:00:00.000Z","DF":null,"IDT":"2021-10-
05T06:05:42.000Z","IUSRNM":"devadm@10.128.3.116","MDT":null,"MUSRNM":null}]}
    ✔ GET /api/v1/emp/:id - Ожидаем ответ 200. Прочитать запись с tabnum=:id из
БД (345ms)
Запрос: :id=000
Запрос: upd body=
{"CODEBRN":"03","NAMEBRN":"Чернигов","FAM":"updatedfam","IM":"Петро","OTCH":"Петро
лвич","ADRESS":"На галявині","MSTATUS":"N","COUNTRY":"UA","DS":"2020-03-19"}
Ответ:
{"status":200,"error":null,"response":
{"fieldCount":0,"affectedRows":1,"insertId":0,"info":"Rows matched: 1  Changed: 1
Warnings: 0","serverStatus":34,"warningStatus":0,"changedRows":1}}
    ✔ POST /api/v1/emp/:id - Ожидаем ответ 200. Обновить запись с TABNUM=:id в
БД (376ms)
Запрос: :id=000
Ответ:
{"status":200,"error":null,"response":
[{"IDREC":10,"CODEBRN":"03","NAMEBRN":"Чернигов","TABNUM":"000","FAM":"updatedfam"
,"IM":"Петро","OTCH":"Петролвич","ADRESS":"На
галявині","MSTATUS":"N","COUNTRY":"UA","DS":"2020-03-
19T00:00:00.000Z","DF":null,"IDT":"2021-10-
05T06:05:42.000Z","IUSRNM":"devadm@10.128.3.116","MDT":"2021-10-
05T06:05:42.000Z","MUSRNM":"devadm@10.128.3.116"}]}
    ✔ GET /api/v1/emp/:id - Ожидаем ответ 200. Прочитать запись с TABNUM=:id
после обновления в БД. Значение  поля FAM  должно совпадать с полем  в обнолвении
(339ms)
Запрос: :id=000
Ответ:
{"status":200,"error":null,"response":
{"fieldCount":0,"affectedRows":1,"insertId":0,"info":"","serverStatus":34,"warning
Status":0}}
    ✔ DELETE /api/v1/emp/:id - Ожидаем ответ 200. Удалить запись с TABNUM=:id из
БД (345ms)


  5 passing (2s)
```

Base on this write CRUD API server in node.js to manipulate data in created at step 1 DB via API has done.

3. Create 10 columns table in Google sheet, fill 3 rows with dummy data

I have created spreadsheet with name "shexample" and sheet "Emp" with structure which is similar to table APP2$EMP pic-11.



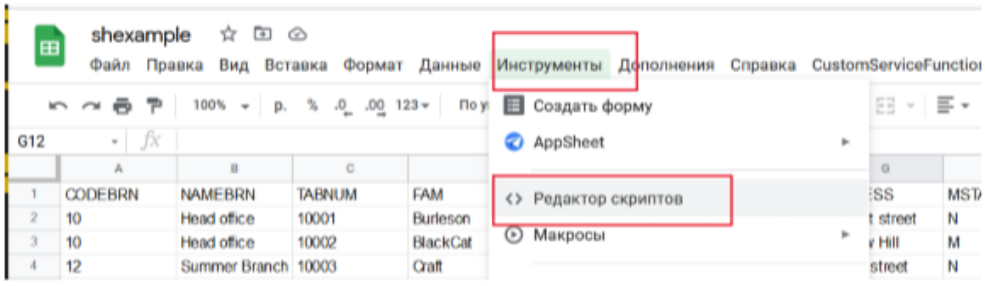
	A	B	C	D	E	F	G	H	I	J	K	L
1	CODEBRN	NAMEBRN	TABNUM	FAM	IM	OTCH	ADRESS	MSTATUS	COUNTRY	DS	DF	
2	10	Head office	10001	Burleson	Janet	Jone	Forest street	N	UA	21.02.2019	01.09.2021	
3	10	Head office	10002	BlackCat	Michael	Laurent	Yellow Hill	M	GB	21.05.2019		
4	12	Summer Branch	10003	Craft	Joann	Larries	Kairo street	N	EG	21.05.2019		
5	11	Winter Branch	10004	Ellison	Larry	Joseph	New York City	M	US	21.05.2021	20.09.2021	
6	12	Spring Branch	1006	Green	Edvard	Jonson	SanDiego	M	US	01.02.2018		
7	12	Spring Branch	1007	EverGreen	Bob	Edvard	SanDiego	M	US	01.03.2018		

pic-11

This step has done.

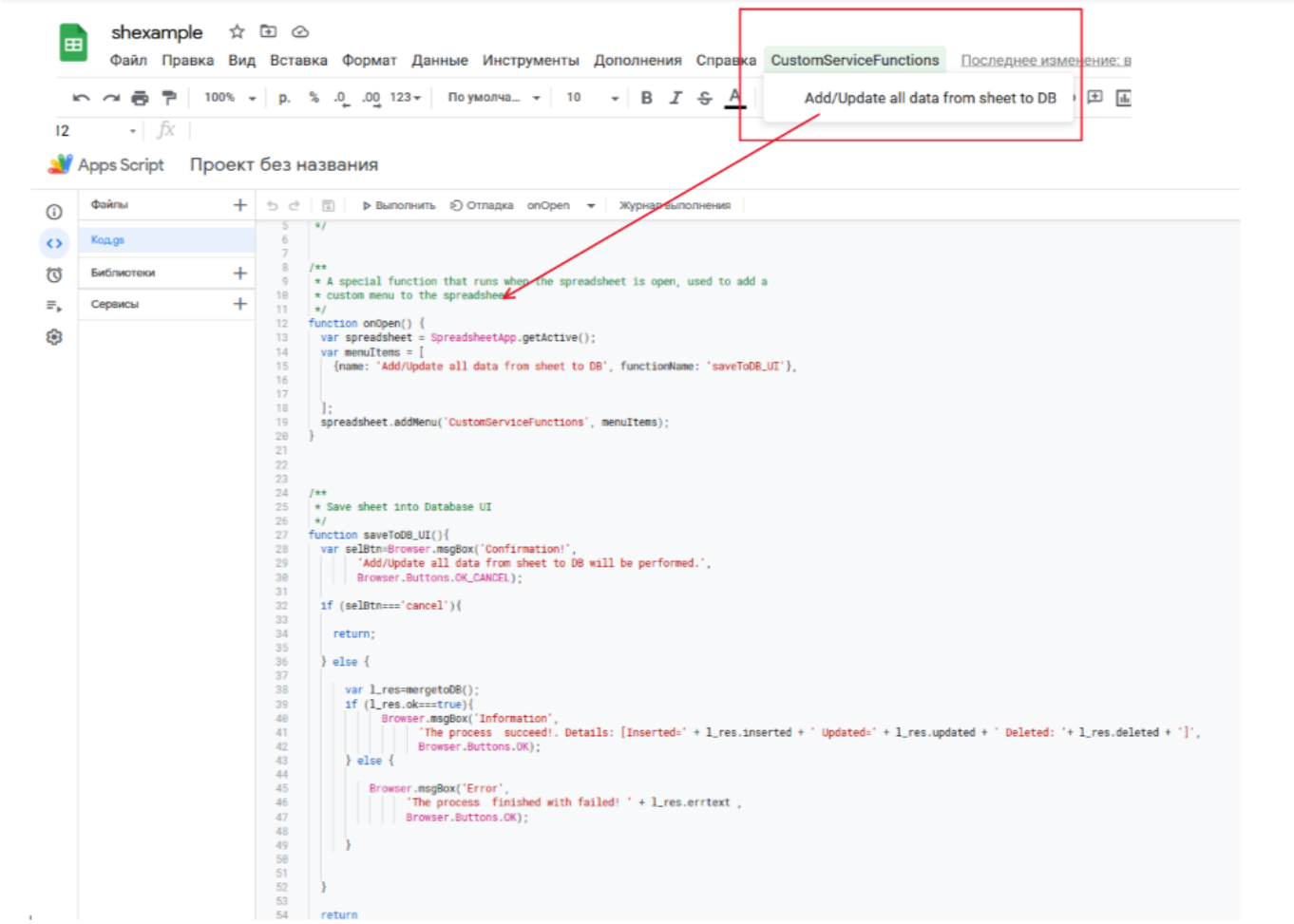
4. Write Google App Script application to connect spreadsheet with db via API created in step 2

Through menu on pic-12 you can create your Apps Script project and create all functions which you need.



pic-12

For UI implementation custom meny added to the main manu in spreadsheet pic-13.



pic-13

Connection to spreadsheet implemented in this part of code pic-14

pic-14

5. You should be able to add/update all data from table to DB

The full code for apps script in file *./appscript/addToDB.gs*. The main function is **function mergetoDB()**

Before run this script you jave to setup you spreadsheet ID and sheet name

```
        Logger.log('Open Spreadsheet');
        var ss = SpreadsheetApp.openById("************************************");
        SpreadsheetApp.setActiveSpreadsheet(ss);
        Logger.log('Select sheet');
        var sheet = ss.getSheetByName("*****");
        SpreadsheetApp.setActiveSheet(sheet);
```

6. How to provide api load test

For load test you may use the npm package loadtest. For demo purposes I have written two tests:

- test-emp-load.js

This module demonstrates how to write load-test with http post request and with generated body. You can run it by using command:

```
npm run ltest-emp
```

- test-emps-load.js

This module demonstrates how to write load-test with http get request. You can run it by using command:

```
npm run ltest-emps
```

If you run load tests, do not forget about pod scaling feature of kubernetes/Openshift. Try playing with it.

In addition, try to play with option parameters: maxRequests: 100 requestsPerSecond: 8 concurrency: 4 maxSeconds: 40