**Proof of Concept**

Using Spring Cloud Config to consolidate and serve runtime configuration from different sources simultaneously.

Author Aleksandrs Krivickis

[Alex.a.krivickis@hsbc.com.hk](mailto:Alex.a.krivickis@hsbc.com.hk)

Contents

[Scope 3](#_Toc526338046)

[Design 3](#_Toc526338047)

[Investigation 3](#_Toc526338048)

[Supported backends 3](#_Toc526338049)

[GitHub with Spring Cloud Config Server 4](#_Toc526338050)

[Features 4](#_Toc526338051)

[MySQL with Spring Cloud Config Server 6](#_Toc526338052)

[MySQL(MariaDB) with Spring Cloud Config Server 7](#_Toc526338053)

[MongoDB with Spring Cloud Config Server 9](#_Toc526338054)

[GitHub and MySQL with Spring Cloud Config Server 10](#_Toc526338055)

[GitHub and MySQL with Spring Cloud Config served through Event Queue 10](#_Toc526338056)

[Investigation summary 10](#_Toc526338057)

[Limitations 10](#_Toc526338058)

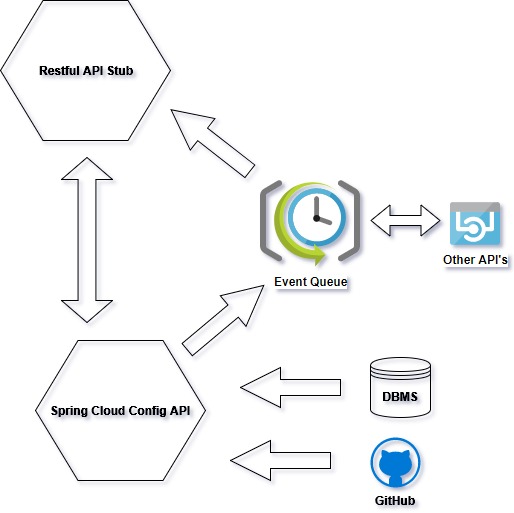
[Conclusions 11](#_Toc526338059)

[References 11](#_Toc526338060)

# Scope

* Find out if it is possible to reliably serve configuration from git and DBMS system simultaneously.
* Check if it is possible to Spring Cloud Config server in combination with Event Queue.
* Implement the system with different DBMS and evaluate them.
* Check if it is possible to store and serve configuration data in JSON format, explore limitations as there might be some in relational DBMS.

# Design



# Investigation

## Supported backends

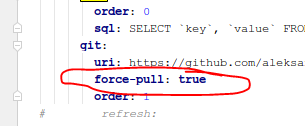
* Local filesystem
* Git
* MySQL
* MySQL(MariaDB)
* MongoDB
* PostgreSQL

## GitHub with Spring Cloud Config Server

<https://github.com/aleksandrskrivickis/spring_cloud_config_research_config>

It is served in a way which is different from JDBC. For every label on GitHub user has to create separate branch.

Also, there is a feature **force-pull,** that allows to pull git repo and serve fresh config data on-demand. Theoretically there is no need to create a callback/hook from GitHub.



### Features

#### Using multiple repositories

spring:

cloud:

config:

server:

git:

uri: https://git/common/config-repo.git

force-pull: **true**

repos:

team-a:

pattern: team-a-\*

uri: http://git/team-a/config-repo.git

force-pull: **true**

team-b:

pattern: team-b-\*

uri: http://git/team-b/config-repo.git

force-pull: **true**

team-c:

pattern: team-c-\*

uri: <http://git/team-a/config-repo.git>

local:

pattern: local\*

uri: file:/home/configsvc/config-repo

#### Git url placeholders:

Spring Cloud Config Server supports a git repository URL with placeholders for the {application} and {profile} (and {label} if you need it, but remember that the label is applied as a git label anyway). So you can easily support a "one repo per application" policy using (for example):

spring:

cloud:

config:

server:

git:

uri: https://github.com/myorg/{application**}**

#### Defining url pattern:

The pattern property in the repo is actually an array, so you can use a YAML array (or [0], [1], etc. suffixes in properties files) to bind to multiple patterns. You may need to do this if you are going to run apps with multiple profiles. Example:

spring:

cloud:

config:

server:

git:

uri: https://github.com/spring-cloud-samples/config-repo

repos:

development:

pattern:

- '\*/development'

- '\*/staging'

uri: https://github.com/development/config-repo

staging:

pattern:

- '\*/qa'

- '\*/production'

uri: https://github.com/staging/config-repo

#### Search path

Spring Cloud Config Server also supports a search path with placeholders for the {application} and {profile} (and {label} if you need it). Example:

spring:

cloud:

config:

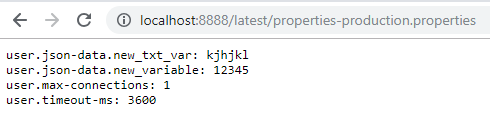
server:

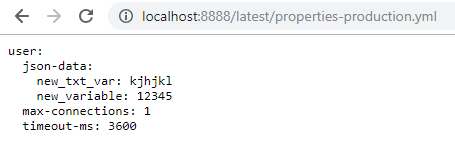
git:

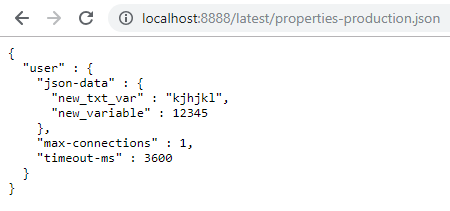
uri: https://github.com/spring-cloud-samples/config-repo

searchPaths: '{application}'

searches the repository for files in the same name as the directory (as well as the top level). Wildcards are also valid in a search path with placeholders (any matching directory is included in the search).



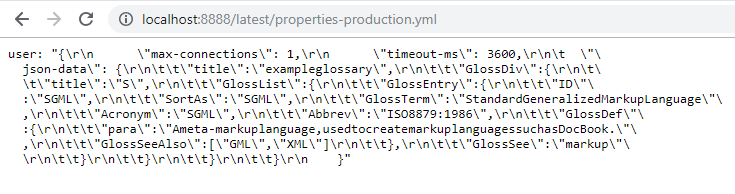


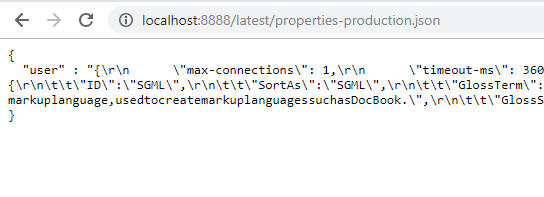


## MySQL with Spring Cloud Config Server

Same limitations and structure of response as with MariaDB. Different driver has to be used for establishing connection.







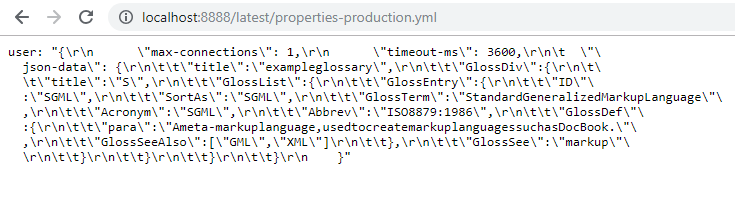
## MySQL(MariaDB) with Spring Cloud Config Server

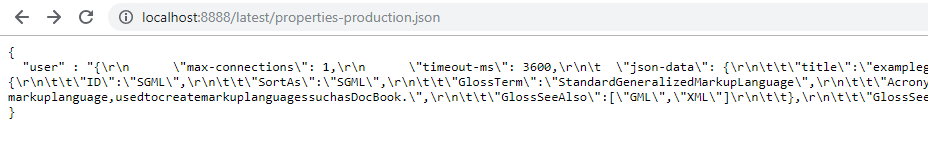
It is possible to serve MongoDB database in three different formats: properties, yml/yaml, json. It is possible to use exactly the same paths as in MongoDB. Unfortunately, I was unable to make the server display **indents** in the same way as with MongoDB but assume it does not make a difference as Client will ignore \n \t \r tags. **Must investigate further if it makes a difference in Spring Cloud Config Client.**

**Limitations**:

The effective maximum length of a **VARCHAR** is subject to the maximum row size (**65,535** bytes, which is shared among all columns). TINYTEXT is a string data type that can store up to to 255 characters. TEXT is a string data type that can store up to 65,535 characters. TEXT is commonly used for brief articles. **LONGTEXT** is a string data type with a maximum length of **4,294,967,295** **characters**.

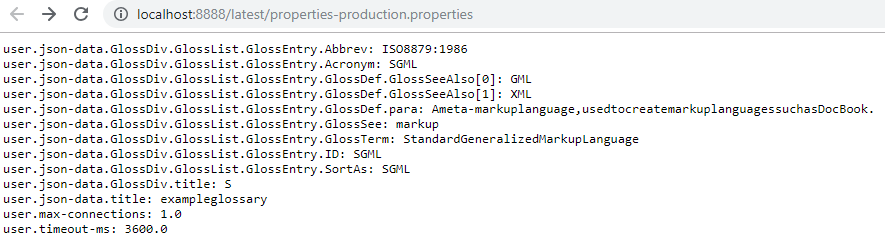


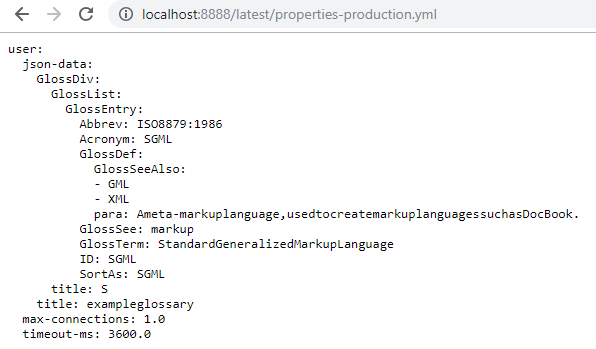




## MongoDB with Spring Cloud Config Server

It is possible to serve MongoDB database in three different formats: properties, yml/yaml, json. In order to display json with indents property “**spring.jackson.serialization.indent\_output=true**” has to be set.

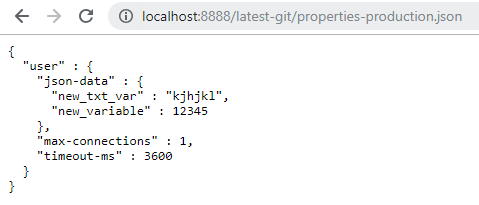


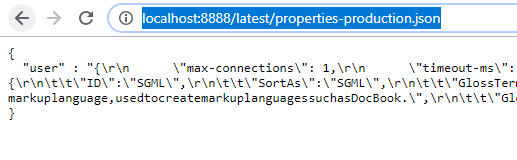




## GitHub and MySQL with Spring Cloud Config Server

It is possible to combine GitHub and MySQL data sources. Unfortunately, there is one limitation – GitHub and MySQL data can’t have the same label. For example: in case if we want to use label “**latest**” to fetch data from MySQL database we can’t create a branch with name “**latest**” as Spring Cloud application will throw an exception and none of data will be available. Unfortunately, to make both data sources work together we have to create an empty branch named “**latest**” and branch named “**latest-git**” in a Git repository. This is the only way for now.





In the next chapter I will try to set-up Spring Cloud Config Client.

## GitHub and MySQL with Spring Cloud Config served to an Spring Boot API through Event Queue

## Investigation summary

# Limitations

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Feature** | **MySQL** | **MySQL(MariaDB)** | **MongoDB** | **GitHub** |
| Value column character length | 4,294,967,295 | 4,294,967,295 | N/A | N/A |
| Max document size | N/A | N/A | 16 MB | 100 MB+ |
| Can use “-“ in database names | No | No | Yes | N/A |
| Serves JSON | With “\r\n\t” tags included, single line | With “\r\n\t” tags included, single line | Yes, pretty | Yes, data to be defined in \*.yml or \*.properties format |

# Conclusions

# References

* <https://github.com/spring-cloud-incubator/spring-cloud-config-server-mongodb>
* <https://medium.com/@nani2ratna/spring-cloud-config-server-with-jdbc-55de8f7ec86d>
* <https://cloud.spring.io/spring-cloud-static/Edgware.SR2/single/spring-cloud.html#_spring_cloud_config>
* <https://dev.mysql.com/doc/refman/8.0/en/data-types.html>
* <https://start.spring.io/>
* <http://cloud.spring.io/spring-cloud-config/single/spring-cloud-config.html>