

**MTender - system deployment documentation guide**

May 2020

Table of Contents

[1. Introduction 4](#_Toc39486762)

[2. Install Cassandra 5](#_Toc39486763)

[2.1 Prepare the nodes 5](#_Toc39486764)

[2.2 Install software 5](#_Toc39486765)

[2.3 Create and configure cluster 6](#_Toc39486766)

[2.4 Create roles 10](#_Toc39486767)

[2.5 Create test keyspace 10](#_Toc39486768)

[2.6 Autostart cassandra when the nodes start 12](#_Toc39486769)

[2.7 Create cassandra database 13](#_Toc39486770)

[3. DNS configuration 15](#_Toc39486771)

[4. Install Kakfa 17](#_Toc39486772)

[4.1 Prepare the nodes 17](#_Toc39486773)

[4.2 Install and configure software 17](#_Toc39486774)

[4.3 Launch zookeeper and kafka 19](#_Toc39486775)

[4.4 Autostart kafka when the nodes start 20](#_Toc39486776)

[4.5 Create topic and test producer and consumer 20](#_Toc39486777)

[5. Install Kafka in gitlab server 22](#_Toc39486778)

[5.1 Install and configure software 22](#_Toc39486779)

[5.2 Configure the properties files 22](#_Toc39486780)

[6. Install Postgresql 35](#_Toc39486781)

[7. Install docker swarm 40](#_Toc39486782)

[8. Install portainer 42](#_Toc39486783)

[9. Install gitLab 43](#_Toc39486784)

[10. Configuration server 49](#_Toc39486785)

[11. Deploy auth service 50](#_Toc39486786)

[12. Deploy chronograph service 53](#_Toc39486787)

[13. Deploy forms service 58](#_Toc39486788)

[14. Deploy gateway service 59](#_Toc39486789)

[15. Deploy mdm service 61](#_Toc39486790)

[16. Deploy public-point service 63](#_Toc39486791)

[17. Deploy storage service 65](#_Toc39486792)

[18. Deploy kafka notifier service 67](#_Toc39486793)

[19. Cassandra ocds keyspace creation 69](#_Toc39486794)

[20. Deploy BPE service 71](#_Toc39486795)

[21. Deploy e-Access service 74](#_Toc39486796)

[22. Deploy e-Auction service 77](#_Toc39486797)

[23. Deploy e-Budget service 87](#_Toc39486798)

[24. Deploy e-Clarification service 90](#_Toc39486799)

[25. Deploy e-Contracting service 93](#_Toc39486800)

[26. Deploy e-Evaluation service 95](#_Toc39486801)

[27. Deploy e-Notice service 98](#_Toc39486802)

[28. Deploy e-Regulation service 101](#_Toc39486803)

[29. Deploy e-Submission service 102](#_Toc39486804)

[30. Deploy Document Generator service 106](#_Toc39486805)

[31. Deploy Gateway Public service 108](#_Toc39486806)

[32. Deploy gateway-mconnect service 110](#_Toc39486807)

# Introduction

The objective of this document is to provide guidance on the deployment of the MTender system. The document reflects stept by step the whole process to be conducted for the successfly deployment of MTender.

The server’s configuration, security settings and some parameters will depend on the environtment where the system is being deployed.

The reader shall bear in mind that all information regarding IPs, Domains Names, servers, users or credentials are only fake data in order to provide a similar displaying of the code and information that shall be provided. Information regarding these parameters shall be adapted to the corresponding service of the server where the system is being deployed.

This system uses different types of databases and some specific technologies. It is recommended that the reader and deployer of the system bear significant knowledge, at least, on the majority of them. The list of main technologies used are:

* Cassandra
* PostgreSQL
* MySQL
* Phpmyadmin
* Kafka
* Docker & Docker Swarm
* GitLab
* Portainer

# Install Cassandra

## **Prepare the nodes**

*# apt-get update*

*# apt-get install install openjdk-8-jdk*

Sincronize the same local time for all nodes. In all Cassandra nodes the timezone must be the same:

*$ sudo –s*

*# ln -sf /usr/share/zoneinfo/Europe/****your\_time\_zone*** */etc/localtime*

*# hwclock --systohc --utc*

*(if too many arfguments error is returned, try this option # hwclock –w –u)*

*# apt-get install ntpdate –y*

*(if –y option is not recognize, try # apt-get install ntpdate)*

*# ntpdate -bs pool.ntp.org*

In all Cassandra nodes, set the host file with all the cassandra nodes:

*# vim /etc/hosts*

(add all cassandra nodes)

***IP\_NODE node\_name***

## **Install software**

Previous to cassandra, it must install open JDK version 8, because version 11 does not run for Cassandra. Check which version is installed in all cassandra nodes and remove version 11 if necessary.

*# dpkg -l | grep openj*

*# apt-get purge openjdk-11-jdk openjdk-11-jdk-hedless openjdk-11-jre-headless*

Install open JDK version 8 in all cassandra nodes:

*# apt-get update*

*# apt-get install -reinstall openjdk-8-jdk*

*(if reinstall option fails, try # apt-get install openjdk-8-jdk)*

Check that only jdk version 8 is installed.

*# dpkg -l | grep openj*

In all Cassandra nodes, remove previous version if it is installed or openjdk version changes.

*# apt-get purge datastax-ddc -y*

In all Cassandra nodes, create a Cassandra source’s list, manage key and install the datastax.

*# echo “deb* [*http://debian.datastax.com/datastax-ddc 3.9*](http://debian.datastax.com/datastax-ddc%203.9) *main” | tee -a /etc/apt/sources.list.d/cassandra.sources.list*

*# apt-key adv --keyserver keys.gnupg.net --recv-keys 350200F2B999A372*

*# apt-get update*

*# apt-get install datastax-ddc*

Check if it is installed correctly.

*# systemctl status cassandra*

*# nodetool status*

*Datacenter: datacenter1*

*=======================*

*Status=Up/Down*

*|/ State=Normal/Leaving/Joining/Moving*

*-- Address Load Tokens Owns (effective) Host ID Rack*

*UN 127.0.0.1 103.52 KiB 256 100.0% b56bdf9a-e516-4420-84ec-0803e79e345b rack1*

## **Create and configure cluster**

All cassandra nodes are the cluster and first node is the server.

Stop cassandra daemon in all nodes.

*# /etc/init.d/cassandra stop*

*# ps –ef | grep cassandra*

If it exists for a previous installation, remove the cassandra folder in all nodes.

*# rm -rf /var/lib/cassandra/\**

Casssandra is a distributed database and three tokens (one per node) have to be generated for the nodes, only in the main node.

Script to generate tokens:

# python -c ‘print [str(((2\*\*64 / number\_of\_tokens) \* i) - 2\*\*63) for i in range(number\_of\_tokens)]’

Example for 3 tokens from main node.

*# python -c 'print [str(((2\*\*64 / 3) \* i) - 2\*\*63) for i in range(3)]'*

*['-9223372036854775808', '-3074457345618258603', '3074457345618258602']*

**For all nodes, repeat the following steps.**

Stop Cassandra.

*#* /etc/init.d/cassandra stop

Configure cassandra.yaml file. First, copy cassandra.yaml file to backup.

*# cp /etc/cassandra/cassandra.yaml /etc/cassandra/cassandra.yaml.backup*

*# vim /etc/cassandra/cassandra.yaml*

Cassandra.yaml file:

N.B.: cassandra.yaml is very sentitive to spaces and format

configuration **/etc/cassandra/cassandra.yaml**for each node

N.B.: num\_tokens has to be commented

File content:

*cluster\_name:* ***'EBRD-Cluster'***

*initial\_token: [token\_of\_the\_node]*

*hinted\_handoff\_enabled: true*

*hinted\_handoff\_throttle\_in\_kb: 1024*

*max\_hints\_delivery\_threads: 2*

*hints\_flush\_period\_in\_ms: 10000*

*max\_hints\_file\_size\_in\_mb: 128*

*batchlog\_replay\_throttle\_in\_kb: 1024*

*authenticator:* ***PasswordAuthenticator***

*authorizer: AllowAllAuthorizer*

*role\_manager: CassandraRoleManager*

*roles\_validity\_in\_ms: 2000*

*permissions\_validity\_in\_ms: 2000*

*credentials\_validity\_in\_ms: 2000*

*partitioner: org.apache.cassandra.dht.Murmur3Partitioner*

*data\_file\_directories:*

*- /var/lib/cassandra/data*

*commitlog\_directory: /var/lib/cassandra/commitlog*

*cdc\_enabled: false*

*disk\_failure\_policy: stop*

*commit\_failure\_policy: stop*

*prepared\_statements\_cache\_size\_mb:*

*thrift\_prepared\_statements\_cache\_size\_mb:*

*key\_cache\_size\_in\_mb:*

*key\_cache\_save\_period: 14400*

*row\_cache\_size\_in\_mb: 0*

*row\_cache\_save\_period: 0*

*counter\_cache\_size\_in\_mb:*

*counter\_cache\_save\_period: 7200*

*saved\_caches\_directory: /var/lib/cassandra/saved\_caches*

*commitlog\_sync: periodic*

*commitlog\_sync\_period\_in\_ms: 10000*

*commitlog\_segment\_size\_in\_mb: 32*

*seed\_provider:*

*- class\_name: org.apache.cassandra.locator.SimpleSeedProvider*

*parameters:*

*- seeds:* ***"[node1\_IP]"***

*concurrent\_reads: 32*

*concurrent\_writes: 32*

*concurrent\_counter\_writes: 32*

*concurrent\_materialized\_view\_writes: 32*

*memtable\_allocation\_type: heap\_buffers*

*index\_summary\_capacity\_in\_mb:*

*index\_summary\_resize\_interval\_in\_minutes: 60*

*trickle\_fsync: false*

*trickle\_fsync\_interval\_in\_kb: 10240*

*storage\_port: 7000*

*ssl\_storage\_port: 7001*

*listen\_address:* ***node\_IP***

*start\_native\_transport: true*

*native\_transport\_port: 9042*

*start\_rpc: false*

*rpc\_interface:* ***eth0***

*rpc\_port: 9160*

*rpc\_keepalive: true*

*rpc\_server\_type: sync*

*thrift\_framed\_transport\_size\_in\_mb: 15*

*incremental\_backups: false*

*snapshot\_before\_compaction: false*

*auto\_snapshot: true*

*column\_index\_size\_in\_kb: 64*

*column\_index\_cache\_size\_in\_kb: 2*

*compaction\_throughput\_mb\_per\_sec: 16*

*sstable\_preemptive\_open\_interval\_in\_mb: 50*

*read\_request\_timeout\_in\_ms: 5000*

*range\_request\_timeout\_in\_ms: 10000*

*write\_request\_timeout\_in\_ms: 2000*

*counter\_write\_request\_timeout\_in\_ms: 5000*

*cas\_contention\_timeout\_in\_ms: 1000*

*truncate\_request\_timeout\_in\_ms: 60000*

*request\_timeout\_in\_ms: 10000*

*cross\_node\_timeout: false*

*endpoint\_snitch: RackInferringSnitch*

*endpoint\_snitch: SimpleSnitch*

*dynamic\_snitch\_update\_interval\_in\_ms: 100*

*dynamic\_snitch\_reset\_interval\_in\_ms: 600000*

*dynamic\_snitch\_badness\_threshold: 0.1*

*request\_scheduler: org.apache.cassandra.scheduler.NoScheduler*

*server\_encryption\_options:*

*internode\_encryption: none*

*keystore: conf/.keystore*

*keystore\_password: cassandra*

*truststore: conf/.truststore*

*truststore\_password: cassandra*

*client\_encryption\_options:*

*enabled: false*

*optional: false*

*keystore: conf/.keystore*

*keystore\_password: cassandra*

*internode\_compression: dc*

*inter\_dc\_tcp\_nodelay: false*

*tracetype\_query\_ttl: 86400*

*tracetype\_repair\_ttl: 604800*

*enable\_user\_defined\_functions: false*

*enable\_scripted\_user\_defined\_functions: false*

*windows\_timer\_interval: 1*

*transparent\_data\_encryption\_options:*

*enabled: false*

*chunk\_length\_kb: 64*

*cipher: AES/CBC/PKCS5Padding*

*key\_alias: testing:1*

*key\_provider:*

*- class\_name: org.apache.cassandra.security.JKSKeyProvider*

*parameters:*

*- keystore: conf/.keystore*

*keystore\_password: cassandra*

*store\_type: JCEKS*

*key\_password: cassandra*

*tombstone\_warn\_threshold: 1000*

*tombstone\_failure\_threshold: 100000*

*batch\_size\_warn\_threshold\_in\_kb: 5*

*batch\_size\_fail\_threshold\_in\_kb: 50*

*unlogged\_batch\_across\_partitions\_warn\_threshold: 10*

*compaction\_large\_partition\_warning\_threshold\_mb: 100*

*gc\_warn\_threshold\_in\_ms: 1000*

Run cassandra and check if it is ok:

*# cassandra -R*

*# nodetool status*

*Datacenter: datacenter1*

*=======================*

*Status=Up/Down*

*|/ State=Normal/Leaving/Joining/Moving*

*-- Address Load Tokens Owns (effective) Host ID Rack*

*UN 78.47.119.87 134.37 KiB 1 100.0% 814014de-c21a-4e12-a319-f60060119d3a rack1*

Last step to be repeated in all nodes.

## **Create roles**

Create role in cqlsh in cassandra main node.

Connect to cqlsh with cassandra user which is the default user for cassandra database and create new admin role caroot.

*# cqlsh -u cassandra -p cassandra* ***main\_node\_IP***

*> CREATE ROLE caroot WITH PASSWORD = ‘Y7NcDVDPen6ZC8fVMe’ AND SUPERUSER = true AND LOGIN = true;*

*> exit;*

Exit and connect again using caroot user to revoke role to user cassandra because it is the default user and is not secure. Set super user false for user cassandra and change the password and revoke role.

*# cqlsh -u caroot* ***main\_node\_IP***

*> ALTER ROLE cassandra WITH PASSWORD = ‘aA7TzD8s52y9gge6ZS’ AND SUPERUSER = false;*

*> exit;*

## **Create test keyspace**

Create new keyspace from main cassandra node.

Log in cassandra from main node and create test keyspace and database structure.

*# cqlsh -u caroot* ***main\_node\_IP***

*> CREATE KEYSPACE IF NOT EXISTS msg WITH REPLICATION = {‘class’ : ‘SimpleStrategy’, ‘replication\_factor’ : 3};*

*> use msg;*

*> CREATE TABLE IF NOT EXISTS pastes (*

*id timeuuid,*

*title text,*

*private boolean,*

*body text,*

*secret\_key text,*

*PRIMARY KEY (private, id)*

*) WITH default\_time\_to\_live = 120 and CLUSTERING ORDER BY (id DESC);*

*> INSERT INTO pastes (private, id, body, secret\_key, title) VALUES (true,3c68b5b4-b1b1-11e6-80f5-76304dec7eb7,'hellow','asdr22sd','This is title');*

*> exit;*

Go to the other cassandra nodes and read the table and rows.

First, check if nodes are running.

*# nodetool status*

*Console display the following information:*

*Datacenter: datacenter1*

*=======================*

*Status=Up/Down*

*|/ State=Normal/Leaving/Joining/Moving*

*-- Address Load Tokens Owns (effective) Host ID Rack*

*UN 94.130.172.193 138.88 KiB 1 100.0% 9eedf1c4-4d1d-4318-8f0d-cbbd6b5ed2e2 rack1*

*UN 78.47.119.87 185.82 KiB 1 100.0% 814014de-c21a-4e12-a319-f60060119d3a rack1*

*UN 116.203.46.5 184.15 KiB 1 100.0% 0711b4e7-732d-4d6a-87de-52d0bfc314c6 rack1*

Connect to cassandra using caroot user and read the table.

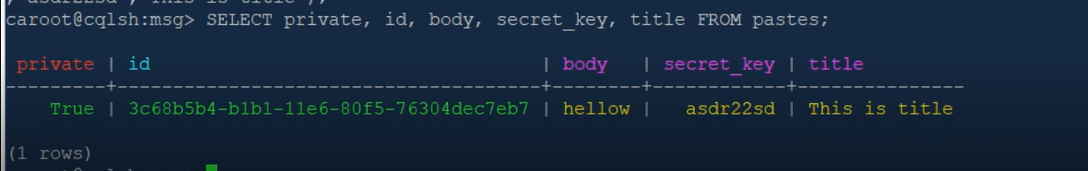
N.B.: time\_to\_live parameter, the rows only are inserted during the time\_to\_live

*# cqlsh –u caroot* ***node\_IP***

*> use msg;*

*> SELECT private, id, body, secret\_key, title FROM pastes;*

Console should display the stored information.



## **Autostart cassandra when the nodes start**

Add cassandra in rc.local to autostart in all nodes to execute automatically when the server starts.

*$ sudo –s*

*# vim /etc/rc.local*

Content:

*#!/bin/bash*

*/usr/sbin/cassandra -R 1>/dev/null 2>/dev/null*

*exit 0*

Grant permissions to the file.

*# chmod +x /etc/rc.local*

And restart the nodes.

From main node, the console can display how the other nodes status are changing.

If one node is shutdown can be verified from other one checking the UN (up normal) or DN (down normal).

*# nodetool status*

*Datacenter: datacenter1*

*=======================*

*Status=Up/Down*

*|/ State=Normal/Leaving/Joining/Moving*

*-- Address Load Tokens Owns (effective) Host ID Rack*

*UN 94.130.172.193 138.88 KiB 1 100.0% 9eedf1c4-4d1d-4318-8f0d-cbbd6b5ed2e2 rack1*

*UN 78.47.119.87 185.82 KiB 1 100.0% 814014de-c21a-4e12-a319-f60060119d3a rack1*

***DN*** *116.203.46.5 184.15 KiB 1 100.0% 0711b4e7-732d-4d6a-87de-52d0bfc314c6 rack1*

When all nodes are started:

*# nodetool status*

*Datacenter: datacenter1*

*=======================*

*Status=Up/Down*

*|/ State=Normal/Leaving/Joining/Moving*

*-- Address Load Tokens Owns (effective) Host ID Rack*

*UN 94.130.172.193 138.88 KiB 1 100.0% 9eedf1c4-4d1d-4318-8f0d-cbbd6b5ed2e2 rack1*

*UN 78.47.119.87 185.82 KiB 1 100.0% 814014de-c21a-4e12-a319-f60060119d3a rack1*

***UN*** *116.203.46.5 184.15 KiB 1 100.0% 0711b4e7-732d-4d6a-87de-52d0bfc314c6 rack1*

## **Create cassandra database**

Verify that cassandra is running in the three nodes.

*# nodetool status*

*Datacenter: datacenter1*

*=======================*

*Status=Up/Down*

*|/ State=Normal/Leaving/Joining/Moving*

*-- Address Load Tokens Owns (effective) Host ID Rack*

***UN*** *94.130.172.193 138.88 KiB 1 100.0% 9eedf1c4-4d1d-4318-8f0d-cbbd6b5ed2e2 rack1*

***UN*** *78.47.119.87 185.82 KiB 1 100.0% 814014de-c21a-4e12-a319-f60060119d3a rack1*

***UN*** *116.203.46.5 184.15 KiB 1 100.0% 0711b4e7-732d-4d6a-87de-52d0bfc314c6 rack1*

Create ocds keyspace and tables needed to the services.

Go to Cassandra main node and connect to Cassandra. Create the keyspace and the first table.

*# cqlsh -u caroot* ***main\_node\_IP***

*>CREATE KEYSPACE IF NOT EXISTS ocds WITH REPLICATION = { 'class' : 'SimpleStrategy', 'replication\_factor' : 3 };*

*> use ocds;*

*> CREATE TABLE ocds.notifier\_kafka\_topic (*

*platform\_id uuid,*

*topic\_id text,*

*PRIMARY KEY (platform\_id)*

*);*

*> insert into ocds.notifier\_kafka\_topic(platform\_id, topic\_id) values(b015f078-536f-4094-852b-9d55dcfcd1da, 'b015f078-536f-4094-852b-9d55dcfcd1da');*

*>CREATE ROL caclient WITH PASSWORD = ‘rMgdGszsgPv79Ts7CB’ AND SUPERUSER = false AND LOGIN = true;*

*>exit;*

Connect to Cassandra using caclient and read from ocds database.

*# cqlsh -u caclient* ***main\_node\_IP***

*> use ocds;*

*> list tables;*

*>exit;*

# DNS configuration

Configure the DNS services in the **Nginx server**.

This service resolves internal resources (nodes of Kafka, cassandra, gitlab, etc)

Go to Nginx server, update system and install bind package:

*# apt-get update*

*# apt-get install bind9 dnsutils*

Set kafka server name in the **/etc/resolv.conf**

Modify parameter: “nameserver 127.0.0.1”

Add the following configuration option to “**/etc/bind/named.conf**” file *(this file contains all the configuration options for the DNS server)*:

*# include "/etc/bind/named.conf.zones";*

Create the file “named.conf.zones” in the path “**/etc/bind/**”, and add the following content**:**

***# vim /etc/bind/named.conf.zones***

Content to add:

*zone "ebrd.system" {*

*type master;*

*file "/etc/bind/zones/master/ebrd.system";*

*};*

Create the standard file “ebrd.system” in the path “etc/bind/zones/master/”, and add the following content**:**

# etc/bind/zones/master/ebrd.system

Content to add:

*$TTL 3600*

*@ IN SOA ns.ebrd.system. ns2.ebrd.system. (*

*2018061000 ; Serial*

*900 ; Refresh*

*900 ; Retry*

*3600 ; Expire*

*3600 ; Minimum*

*)*

*@ IN NS ns.ebrd.system.*

*@ IN NS ns2.ebrd.system.*

*@ IN A [server\_internal\_ip]*

*www IN CNAME @*

*ns IN A [server\_internal\_ip]*

*ns2 IN A [server\_internal\_ip]*

*ac-node1 IN A [server\_internal\_ip]*

*ac-node2 IN A [server\_internal\_ip]*

*ac-node3 IN A [server\_internal\_ip]*

*ak-node1 IN A [server\_internal\_ip]*

*ak-node2 IN A [server\_internal\_ip]*

*ak-node3 IN A [server\_internal\_ip]*

*postgresql IN A [server\_internal\_ip]*

*feed IN A [server\_internal\_ip]*

*docker-node1 IN A [server\_internal\_ip]*

*docker-node2 IN A [server\_internal\_ip]*

*docker-node3 IN A [server\_internal\_ip]*

*gitlab IN A [server\_internal\_ip]*

*Note: nginx is required only for systems that uses VPN. In our example it is disable.*

Restart DNS service with the new configuration:

*# /etc/init.d/bind9 restart*

Check the port 53 is working in Nginx server:

*# netstat -nltp | grep 53*

Check if DNS server is reached from other servers.

Go to docker-node1 and use the following command:

*# dig @116.203.93.15 postgresql.ebrd.system*

# Install Kakfa

## **Prepare the nodes**

All Kafka nodes the timezone must be the same:

*$ sudo –s*

*# ln -sf /usr/share/zoneinfo/Europe/****your\_time\_zone*** */etc/localtime*

*# hwclock --systohc --utc*

*(if too many arfguments error is returned, try this option # hwclock –w –u)*

*# apt-get install ntpdate –y*

*(if –y option is not recognize, try # apt-get install ntpdate)*

*# ntpdate -bs pool.ntp.org*

In all Kafka nodes, set the host file with all the kafka nodes:

*# vim /etc/hosts*

*(add all kafka nodes)*

***IP\_NODE node\_name***

Check with if the hosts file is correct with the ping to other nodes. From one kafka node, try to connect to the others kafka nodes.

*# ping* ***node\_name***

## **Install and configure software**

Intall jdk and kafka for all nodes. Repeat following steps for all of them.

*# apt-get install default-jdk*

Producer kafka node:

*# wget http://apache.volia.net/kafka/2.4.0/kafka\_2.11-2.4.0.tgz*

*# tar xf kafka\_2.11-2.4.0.tgz*

*# mkdir –p /var/zookeeper/data*

*# mkdir /home/kafka*

*# cp -rp kafka\_2.11-2.4.0/\* /home/kafka/*

*# cd /home/kafka*

Configure the properties files in all kafka nodes. Repeat the following actions for all nodes.

Create a backup of the file “config/zookeeper.properties”.

*# cd /home/kafka*

*# cp config/zookeeper.properties config/zookeeper.properties.back*

Set the zookeeper.properties in the original.

*dataDir=/var/zookeeper/data*

*# the port at which the clients will connect*

*clientPort=2181*

*# disable the per-ip limit on the number of connections since this is a non-production config*

*maxClientCnxns=0*

*server.1=****kafka\_node1\_name****:2888:3888*

*server.2=****kafka\_node2\_name****:2888:3888*

*server.3=****kafka\_node3\_name****:2888:3888*

*# Disable the adminserver by default to avoid port conflicts.*

*# Set the port to something non-conflicting if choosing to enable this*

*admin.enableServer=false*

*# admin.serverPort=8080*

*initLimit=5*

*syncLimit=2*

Set the id in all kafka nodes.

*# echo "****id****" > /var/zookeeper/data/myid*

Where id parameter should be the id of the node. (Different for all nodes)

Example:

*ak-node1 🡪 root@kafka1-1: echo "01" > /var/zookeeper/data/myid*

*ak-node2 🡪 root@kafka1-2: echo "02" > /var/zookeeper/data/myid*

*ak-node3 🡪 root@kafka1-3: echo "03" > /var/zookeeper/data/myid*

Create a backup of the file “config/server.properties”.

*# cd /home/kafka*

*# cp config/server.properties config/server.properties.back*

Set the server.properties in the original.

*broker.id=****id\_number***

*advertised.listeners=PLAINTEXT://****ak-node1.ebrd.system****:9092 (each kafka node)*

*num.network.threads=3*

*num.io.threads=8*

*socket.send.buffer.bytes=102400*

*socket.receive.buffer.bytes=102400*

*socket.request.max.bytes=104857600*

*log.dirs=/var/log/kafka-logs*

*num.partitions=1*

*num.recovery.threads.per.data.dir=1*

*offsets.topic.replication.factor=1*

*transaction.state.log.replication.factor=1*

*transaction.state.log.min.isr=1*

*log.retention.hours=168*

*log.segment.bytes=1073741824*

*log.retention.check.interval.ms=300000*

*zookeeper.connect=****ak-node1.ebrd.system:2181,ak-node2.ebrd.system:2181,ak-node3.ebrd.system****:2181*

*zookeeper.connection.timeout.ms=6000*

*group.initial.rebalance.delay.ms=0*

*delete.topic.enable=true*

“broker.id” and “advertised.listeners” parameters will depends on the node’s properties.

id\_number 🡪 identifier in each node (for example: “01”).

## **Launch zookeeper and kafka**

Launch zookeeper in all nodes.

Start the zookeeper server and test if it runs correctly.

*# cd /home/kafka*

*# bin/zookeeper-server-start.sh config/zookeeper.properties*

Confirm all is executed without errors and stop it.

Start it in background mode and check with the port listener.

*# nohup bin/zookeeper-server-start.sh config/zookeeper.properties &*

*# netstat –ntlp | grep 2181*

Launch kafka server in all nodes and test if it runs ok.

*# cd /home/kafka*

*# bin/kafka-server-start.sh config/server.properties*

Confirm all is executed without errors and stop it.

Start it in background mode and check with the port listener.

*# nohup bin/kafka-server-start.sh config/server.properties &*

*# netstat –nltp | grep 9092*

## **Autostart kafka when the nodes start**

Configure “/etc/rc.local” file to start automatically. (Zookeper must start first).

*# vim /etc/rc.local*

*Content:*

*#!/bin/bash*

*cd /home/kafka*

*nohup bin/zookeeper-server-start.sh config/zookeeper.properties &*

*sleep 5*

*nohup bin/kafka-server-start.sh config/server.properties &*

*exit 0*

Grant permissions to the file to execute.

*# chmod +x /etc/local*

Note: Here ends the instruction to repeat for all nodes.

## **Create topic and test producer and consumer**

The first node is the producer node, create a topic and run the producer.

*# bin/kafka-topics.sh --create --zookeeper* ***ak-node1.ebrd.system****:2181,****ak-node2.ebrd.system****:2181,****ak-node3.ebrd.system****:2181 --replication-factor 3 --partitions 1 --topic EBRD-first-topic*

Console should display the following message:

*Created topic EBRD-first-topic*

Run producer and add message in topic.

Go to Kafka-node1.

*# bin/kafka-console-producer.sh --broker-list* ***ak-node1.ebrd.system****:9092,****ak-node2.ebrd.system****:9092,****ak-node3.ebrd.system****:9092 --topic EBRD-first-topic*

*> hello my first message!*

In the other nodes (consumers), run consumer and test if producer messages are received. (Do twice, one for node1 and other for node2).

*# bin/kafka-console-consumer.sh --topic EBRD-first-topic --from-beginning --bootstrap-server* ***ak-node1.ebrd.system****:9092*

*Console should display the following message:*

*hello my first message!*

Add configuration to the resolv file.

*# vim /etc/resolv.conf*

***Nameserver [ip\_of\_the\_node]***

*Nameserver 1.1.1.1*

*Options edns0*

# Install Kafka in gitlab server

## **Install and configure software**

Go to Gitlab server

Create kafka directory and go to /home/kafka.

Install Java Development Kit.

Jdk: version 11

[*#*](mailto:root@nginx1-1:/home/kafka/kafka_2.11-2.4.0) *apt-get install default-jdk*

Prepare de kafka folder.

*# mkdir /home/kafka*

*# cd /home/kafka*

Get kafka software.

*# wget* [*http://apache.volia.net/kafka/2.4.0/kafka\_2.11-2.4.0.tgz*](http://apache.volia.net/kafka/2.4.0/kafka_2.11-2.4.0.tgz)

Unzip kafka file and move all source files to /home/kafka directory.

*# tar xf kafka\_2.11-2.4.0.tgz*

*# cd kafka\_2.11-2.4.0*

*# mv ./\* ../*

Check that all files have been moved and delete the empty folder.

*# rm -r kafka\_2.11-2.4.0*

## **Configure the properties files**

Set the kafka node identifier.

[*#*](mailto:root@nginx1-1:/home/kafka/kafka_2.11-2.4.0) *mkdir –p /var/zookeeper/data/*

[*#*](mailto:root@nginx1-1:/home/kafka/kafka_2.11-2.4.0) *echo “01” > /var/zookeeper/data/myid*

Copy zookeeper properties into zookeeper sasl file properties. It is a policy file to configure permissions.

Sasl is a terminal object for apache kafka product.

[*#*](mailto:root@nginx1-1:/home/kafka/kafka_2.11-2.4.0) *cd config/*

[*#*](mailto:root@nginx1-1:/home/kafka/kafka_2.11-2.4.0) *cp zookeeper.properties zookeeper-sasl.properties*

[*#*](mailto:root@nginx1-1:/home/kafka/kafka_2.11-2.4.0) *vim zookeeper-sasl.properties*

Content:

*dataDir=/var/zookeeper/data*

*clientPort=2181*

*maxClientCnxns=0*

*admin.enableServer=false*

*auth.provider=org.apache.zookeper.server.auth.SASLAuthenticationProvider*

*requiredClientAuthScheme=sasl*

*jassLoginRenew=3600000*

Copy consumer.properties to consumer-sasl.properties and edit it.

[*#*](mailto:root@nginx1-1:/home/kafka/kafka_2.11-2.4.0) *cp consumer.properties consumer-sasl.properties*

*# vim consumer-sasl.properties*

Content:

*bootstrap.servers=localhost:9092*

*group.id=test-consumer-group*

*security.protocol=SASL\_PLAINTEXT*

*sasl.mechanism=PLAIN*

*group.id=console-consumer-1262*

Copy producer.properties to producer-sasl.properties and edit it.

# cp producer.properties producer-sasl.*properties*

# vim producer-sasl.properties

Content:

*bootstrap.servers=localhost:9092*

*compression.type=none*

*security.protocol=SASL\_PLAINTEXT*

*sasl.mechanism=PLAIN*

Copy server.properties to server-sasl.properties and edit it.

[*#*](mailto:root@nginx1-1:/home/kafka/kafka_2.11-2.4.0) *cp server.properties server-sasl.properties*

*# vim server-sasl.properties*

Content:

*security.inter.broker.protocol=SASL\_PLAINTEXT*

*sasl.mechanism.inter.broker.protocol=PLAIN*

*sasl.enabled.mechanisms=PLAIN*

*authorizer.class.name=kafka.security.auth.SimpleAclAuthorizer*

*allow.everyone.if.no.acl.found=false*

*auto.create.topics.enable=false*

*broker.id=0*

*num.network.threads=3*

*num.io.threads=8*

*socket.send.buffer.bytes=102400*

*socket.receive.buffer.bytes=102400*

*socket.request.max.bytes=104857600*

*log.dirs=/tmp/kafka-logs*

*num.partitions=1*

*num.recovery.threads.per.data.dir=1*

*offsets.topic.replication.factor=1*

*transaction.state.log.replication.factor=1*

*transaction.state.log.min.isr=1*

*log.retention.hours=168*

*log.segment.bytes=1073741824*

*log.retention.check.interval.ms=300000*

*zookeeper.connect=localhost:2181*

*zookeeper.connection.timeout.ms=6000*

*group.initial.rebalance.delay.ms=0*

*listeners=SASL\_PLAINTEXT://:9092*

*advertised.listeners=SASL\_PLAINTEXT://feed.ebrd.system:9092*

*super.users=User:admin*

Modify the Standalone properties file:

[*#*](mailto:root@nginx1-1:/home/kafka/kafka_2.11-2.4.0) *vim connect-standalone.properties*

Content:

*bootstrap.servers=localhost:9092*

*key.converter=org.apache.kafka.connect.json.JsonConverter*

*value.converter=org.apache.kafka.connect.json.JsonConverter*

*key.converter.schemas.enable=true*

*value.converter.schemas.enable=true*

*offset.storage.file.filename=/tmp/connect.offsets*

*offset.flush.interval.ms=10000*

Create the file “jaas-kafka-client.conf”:

[*#*](mailto:root@nginx1-1:/home/kafka/kafka_2.11-2.4.0) *vim jaas-kafka-client.conf*

Content:

*KafkaClient {*

*org.apache.kafka.common.security.plain.PlainLoginModule required*

*username="admin"*

*password=kZTzfu9C5J7zW6KmW5;*

*};*

*Client {*

*org.apache.kafka.common.security.plain.PlainLoginModule required*

*username="admin"*

*password=kZTzfu9C5J7zW6KmW5;*

*};*

Create the file “jaas-kafka-server.conf”:

[#](mailto:root@nginx1-1:/home/kafka/kafka_2.11-2.4.0) vim jaas-kafka-server.conf

Content:

*KafkaServer {*

*org.apache.kafka.common.security.plain.PlainLoginModule required*

*username="admin"*

*password="kZTzfu9C5J7zW6KmW5"*

*user\_admin="kZTzfu9C5J7zW6KmW5"*

*user\_YodaAdmin="Tc7Wxb7yY6HGRgF4WX"*

*user\_8c32f576-4ebc-4eb4-8654-cc6fd87f091c="XEJUtKsVLvaeq7PZUr";*

*};*

*Client {*

*org.apache.kafka.common.security.plain.PlainLoginModule required*

*username="admin"*

*password="kZTzfu9C5J7zW6KmW5";*

*};*

Create file “jaas-zookeper.conf”:

*# vim jaas-zookeper.conf*

Content:

*Server {*

*org.apache.kafka.common.security.plain.PlainLoginModule required*

*username="admin"*

*password="kZTzfu9C5J7zW6KmW5"*

*user\_admin="kZTzfu9C5J7zW6KmW5"*

*user\_8c32f576-4ebc-4eb4-8654-cc6fd87f091c="XEJUtKsVLvaeq7PZUr"*

*user\_YodaAdmin="Tc7Wxb7yY6HGRgF4WX";*

*};*

Go to bin directory (/home/kafka/bin)

These scripts control consumer and producer.

Prepare zookeeper sasl bin file.

# cp *zookeeper*-server-start.sh zookeeper-server-sasl-start.sh

# vim zookeeper-server-sasl-start.sh

Content:

*#!/bin/bash*

*# Licensed to the Apache Software Foundation (ASF) under one or more*

*# contributor license agreements. See the NOTICE file distributed with*

*# this work for additional information regarding copyright ownership.*

*# The ASF licenses this file to You under the Apache License, Version 2.0*

*# (the "License"); you may not use this file except in compliance with*

*# the License. You may obtain a copy of the License at*

*#*

*# http://www.apache.org/licenses/LICENSE-2.0*

*#*

*# Unless required by applicable law or agreed to in writing, software*

*# distributed under the License is distributed on an "AS IS" BASIS,*

*# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.*

*# See the License for the specific language governing permissions and*

*# limitations under the License.*

*if [ $# -lt 1 ];*

*then*

*echo "USAGE: $0 [-daemon] zookeeper.properties"*

*exit 1*

*fi*

*base\_dir=$(dirname $0)*

*if [ "x$KAFKA\_LOG4J\_OPTS" = "x" ]; then*

*export KAFKA\_LOG4J\_OPTS="-Dlog4j.configuration=file:$base\_dir/../config/log4j.properties"*

*fi*

*if [ "x$KAFKA\_HEAP\_OPTS" = "x" ]; then*

*export KAFKA\_HEAP\_OPTS="-Xmx512M -Xms512M"*

*fi*

*EXTRA\_ARGS=${EXTRA\_ARGS-'-name zookeeper -loggc'}*

*COMMAND=$1*

*case $COMMAND in*

*-daemon)*

*EXTRA\_ARGS="-daemon "$EXTRA\_ARGS*

*shift*

*;;*

*\*)*

*;;*

*esac*

*#exec $base\_dir/kafka-run-class.sh $EXTRA\_ARGS org.apache.zookeeper.server.quorum.QuorumPeerMain "$@"*

Add the following line at the bottom of the document:

*exec $base\_dir/kafka-run-class.sh $EXTRA\_ARGS -Djava.security.auth.login.config=$base\_dir/../config/jaas-zookeeper.conf org.apache.zookeeper.server.quorum.QuorumPeerMain "$@"*

Prepare kafka server sasl bin file:

*# cp kafka-server-start.sh kafka-server-sasl-start.sh*

*# vim kafka-server-sasl-start.sh*

Content:

*#!/bin/bash*

*# Licensed to the Apache Software Foundation (ASF) under one or more*

*# contributor license agreements. See the NOTICE file distributed with*

*# this work for additional information regarding copyright ownership.*

*# The ASF licenses this file to You under the Apache License, Version 2.0*

*# (the "License"); you may not use this file except in compliance with*

*# the License. You may obtain a copy of the License at*

*#*

*# http://www.apache.org/licenses/LICENSE-2.0*

*#*

*# Unless required by applicable law or agreed to in writing, software*

*# distributed under the License is distributed on an "AS IS" BASIS,*

*# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.*

*# See the License for the specific language governing permissions and*

*# limitations under the License.*

*if [ $# -lt 1 ];*

*then*

*echo "USAGE: $0 [-daemon] server.properties [--override property=value]\*"*

*exit 1*

*fi*

*base\_dir=$(dirname $0)*

*if [ "x$KAFKA\_LOG4J\_OPTS" = "x" ]; then*

*export KAFKA\_LOG4J\_OPTS="-Dlog4j.configuration=file:$base\_dir/../config/log4j.properties"*

*fi*

*if [ "x$KAFKA\_HEAP\_OPTS" = "x" ]; then*

*export KAFKA\_HEAP\_OPTS="-Xmx1G -Xms1G"*

*fi*

*EXTRA\_ARGS=${EXTRA\_ARGS-'-name kafkaServer -loggc'}*

*COMMAND=$1*

*case $COMMAND in*

*-daemon)*

*EXTRA\_ARGS="-daemon "$EXTRA\_ARGS*

*shift*

*;;*

*\*)*

*;;*

*esac*

*#exec $base\_dir/kafka-run-class.sh $EXTRA\_ARGS kafka.Kafka "$@"*

*exec $base\_dir/kafka-run-class.sh $EXTRA\_ARGS -Djava.security.auth.login.config=$base\_dir/../config/jaas-kafka-server.conf kafka.Kafka "$@"*

Prepare admin config file.

*# cd ..*

*# cd config*

*# vim jaas-kafka-YodaAdmin.conf*

Content:

*KafkaClient {*

*org.apache.kafka.common.security.plain.PlainLoginModule required*

*username="YodaAdmin"*

*password="Tc7Wxb7yY6HGRgF4WX";*

*};*

Prepare the producer script for admin kafka-console-producer-sasl-YodaAdmin.sh

*# cd ..*

*# cd bin*

*# cp kafka-console-producer.sh kafka-console-producer-sasl-YodaAmin.sh*

*# vim kafka-console-producer-sasl-YodaAmin.sh*

Content:

*#!/bin/bash*

*# Licensed to the Apache Software Foundation (ASF) under one or more*

*# contributor license agreements. See the NOTICE file distributed with*

*# this work for additional information regarding copyright ownership.*

*# The ASF licenses this file to You under the Apache License, Version 2.0*

*# (the "License"); you may not use this file except in compliance with*

*# the License. You may obtain a copy of the License at*

*#*

*# http://www.apache.org/licenses/LICENSE-2.0*

*#*

*# Unless required by applicable law or agreed to in writing, software*

*# distributed under the License is distributed on an "AS IS" BASIS,*

*# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.*

*# See the License for the specific language governing permissions and*

*# limitations under the License.*

*if [ "x$KAFKA\_HEAP\_OPTS" = "x" ]; then*

*export KAFKA\_HEAP\_OPTS="-Xmx512M"*

*fi*

*#exec $(dirname $0)/kafka-run-class.sh kafka.tools.ConsoleProducer "$@"*

*base\_dir=$(dirname $0)*

*exec $base\_dir/kafka-run-class.sh $EXTRA\_ARGS -Djava.security.auth.login.config=$base\_dir/../config/jaas-kafka-YodaAdmin.conf kafka.tools.ConsoleProducer "$@"*

Prepare the kafka consumer kafka-console-consumer-sasl-8c32f576-4ebc-4eb4-8654-cc6fd87f091c.sh

*# cp kafka-console-consumer.sh kafka-console-consumer-sasl-8c32f576-4ebc-4eb4-8654-cc6fd87f091c.sh*

*# vim kafka-console-consumer-sasl-8c32f576-4ebc-4eb4-8654-cc6fd87f091c.sh*

Content:

*#!/bin/bash*

*# Licensed to the Apache Software Foundation (ASF) under one or more*

*# contributor license agreements. See the NOTICE file distributed with*

*# this work for additional information regarding copyright ownership.*

*# The ASF licenses this file to You under the Apache License, Version 2.0*

*# (the "License"); you may not use this file except in compliance with*

*# the License. You may obtain a copy of the License at*

*#*

*# http://www.apache.org/licenses/LICENSE-2.0*

*#*

*# Unless required by applicable law or agreed to in writing, software*

*# distributed under the License is distributed on an "AS IS" BASIS,*

*# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.*

*# See the License for the specific language governing permissions and*

*# limitations under the License.*

*if [ "x$KAFKA\_HEAP\_OPTS" = "x" ]; then*

*export KAFKA\_HEAP\_OPTS="-Xmx512M"*

*fi*

*#exec $(dirname $0)/kafka-run-class.sh kafka.tools.ConsoleConsumer "$@"*

*base\_dir=$(dirname $0)*

*exec $base\_dir/kafka-run-class.sh $EXTRA\_ARGS -Djava.security.auth.login.config=$base\_dir/../config/jaas-kafka-client-8c32f576-4ebc-4eb4-8654-cc6fd87f091c.conf kafka.tools.ConsoleConsumer "$@"*

Prepare the kafka client for admin users creating the file “*jaas-kafka-client-8c32f576-4ebc-4eb4-8654-cc6fd87f091c.conf*”

*# vim config/jaas-kafka-client-8c32f576-4ebc-4eb4-8654-cc6fd87f091c.conf*

Content:

*KafkaClient {*

*org.apache.kafka.common.security.plain.PlainLoginModule required*

*username="8c32f576-4ebc-4eb4-8654-cc6fd87f091c"*

*password="XEJUtKsVLvaeq7PZUr";*

*};*

Prepare the consumer properties file for admin user creating “*consumer-sasl-8c32f576-4ebc-4eb4-8654-cc6fd87f091c.properties*” file:

*# vim config/consumer-sasl-8c32f576-4ebc-4eb4-8654-cc6fd87f091c.properties*

Content:

*security.protocol=SASL\_PLAINTEXT*

*sasl.mechanism=PLAIN*

*group.id=ebrdgroup*

**Start zookeeper sals server in background mode**

*# nohup bin/zookeeper-server-sasl-start.sh config/zookeeper-sasl.properties &*

*# netstat -nltp | grep 2181*

Console:

*tcp 0 0 0.0.0.0:2181 0.0.0.0:\* LISTEN 60833/java*

**Start kafka-server sasl server in background mode**

*# nohup bin/kafka-server-sasl-start.sh config/server-sasl.properties &*

*# netstat -nltp | grep 9092*

Console:

*tcp 0 0 0.0.0.0:9092 0.0.0.0:\* LISTEN 61400/java*

Add zookeeper and kafka server sasl in rc.local to autostart

Content:

*# vim /etc/rc.local*

cd /home/kafka

*nohup bin/zookeeper-server-sasl-start.sh config/zookeeper-sasl.properties &*

*sleep 5*

*nohup bin/kafka-server-sasl-start.sh config/server-sasl.properties &*

*exit 0*

Grant access to the file.

*# chmod +x /etc/rc.local*

Create topics needed to the services

*# bin/kafka-topics.sh --create --zookeeper localhost:2181 --replication-factor 1 --partitions 1 --topic 8c32f576-4ebc-4eb4-8654-cc6fd87f091c*

*# bin/kafka-topics.sh --create --zookeeper localhost:2181 --replication-factor 1 --partitions 1 --topic b015f078-536f-4094-852b-9d55dcfcd1da*

List kafka topics to verify if they have been installed correctly

*# bin/kafka-topics.sh --list --zookeeper localhost:2181*

Console display:

*8c32f576-4ebc-4eb4-8654-cc6fd87f091c*

*\_\_consumer\_offsets*

*b015f078-536f-4094-852b-9d55dcfcd1da*

Give permision to kafka producer user

*# bin/kafka-acls.sh --authorizer kafka.security.auth.SimpleAclAuthorizer --authorizer-properties zookeeper.connect=localhost:2181 --add --allow-principal User:YodaAdmin --producer --topic 8c32f576-4ebc-4eb4-8654-cc6fd87f091c*

*# bin/kafka-acls.sh --authorizer kafka.security.auth.SimpleAclAuthorizer --authorizer-properties zookeeper.connect=localhost:2181 --add --allow-principal User:YodaAdmin --producer --topic b015f078-536f-4094-852b-9d55dcfcd1da*

Give permition to write to kafka producer user.

*# bin/kafka-acls.sh --authorizer kafka.security.auth.SimpleAclAuthorizer --authorizer-properties zookeeper.connect=localhost:2181 --add --allow-principal User:YodaAdmin --operation Read --operation Write --topic 8c32f576-4ebc-4eb4-8654-cc6fd87f091c*

*# bin/kafka-acls.sh --authorizer kafka.security.auth.SimpleAclAuthorizer --authorizer-properties zookeeper.connect=localhost:2181 --add --allow-principal User:YodaAdmin --operation Read --operation Write --topic b015f078-536f-4094-852b-9d55dcfcd1da*

Show list permitions to check if they have been granted correctly:

*# bin/kafka-acls.sh --authorizer kafka.security.auth.SimpleAclAuthorizer --authorizer-properties zookeeper.connect=localhost:2181 --list*

Adding user to ebrd group

*# bin/kafka-acls.sh --authorizer kafka.security.auth.SimpleAclAuthorizer --authorizer-properties zookeeper.connect=localhost:2181 --add --allow-principal User:8c32f576-4ebc-4eb4-8654-cc6fd87f091c --operation Read --group ebrdgroup*

Read permision granted to users to verify if they are correct:

*# bin/kafka-acls.sh --authorizer kafka.security.auth.SimpleAclAuthorizer --authorizer-properties zookeeper.connect=localhost:2181 --add --allow-principal User:8c32f576-4ebc-4eb4-8654-cc6fd87f091c --operation Read --topic 8c32f576-4ebc-4eb4-8654-cc6fd87f091c*

*# bin/kafka-acls.sh --authorizer kafka.security.auth.SimpleAclAuthorizer --authorizer-properties zookeeper.connect=localhost:2181 --add --allow-principal User:8c32f576-4ebc-4eb4-8654-cc6fd87f091c --operation Read --topic b015f078-536f-4094-852b-9d55dcfcd1da*

Check if producer and consumer are correctly defined sending a message

Send

*# bin/kafka-console-producer-sasl-YodaAdmin.sh --broker-list localhost:9092 --topic 8c32f576-4ebc-4eb4-8654-cc6fd87f091c --producer.config=/home/kafka/config/producer-sasl.properties*

Receive

*Be sure that /kafka-console-consumer-sasl-8c32f576-4ebc-4eb4-8654-cc6fd87f091c.sh has +x permision*

*# bin/kafka-console-consumer-sasl-8c32f576-4ebc-4eb4-8654-cc6fd87f091c.sh --bootstrap-server localhost:9092 --topic 8c32f576-4ebc-4eb4-8654-cc6fd87f091c --from-beginning --consumer.config=config/consumer-sasl-8c32f576-4ebc-4eb4-8654-cc6fd87f091c.properties*

Create kafka topic, needed to deploy services.

*#bin/kafka-topics.sh \*

*--create \*

*--zookeeper ak-node1.ebrd.system:2181,ak-node2.ebrd.system:2181,ak-node3.ebrd.system:2181 \*

*--replication-factor 2 \*

*--partitions 1 \*

*--topic notification-kafka-channel*

# Install Postgresql

Connecta as root user.

Update the package manager

*# sudo –s*

*# apt-get update*

Add certificates

*# sudo apt-get install curl ca-certificates gnupg*

*# curl https://www.postgresql.org/media/keys/ACCC4CF8.asc | sudo apt-key add -*

Search the latest available version of the postgres

*# apt-cache search postges | grep server*

Add the postgress repository and dependencies.

*# add-apt-repository "deb http://apt.postgresql.org/pub/repos/apt/ $(lsb\_release -sc)-pgdg main"*

*# wget --quiet -O - https://www.postgresql.org/media/keys/ACCC4CF8.asc | sudo apt-key add -*

Update the system.

*# apt-get update*

Install postgres

*# apt-get install postgresql-10*

Check if the port is listening

*# netstat -ntlp | grep 5432*

The console displays the following message:

*tcp 0 0 0.0.0.0:5432 0.0.0.0:\* LISTEN 5310/postgres*

Configure files and users setting up config file and adding all addresses.

Create a backup of the config file.

*# cp /etc/postgresql/10/main/postgresql.conf /etc/postgresql/10/main/postgresql.conf.back*

Change the listen\_address in postgresql.conf file

*# vim /etc/postgresql/10/main/postgresql.conf*

Content to add:

*listen\_addresses = '\*'*

Restart postgres service

*# /etc/init.d/postgresql restart*

The console displays the following message:

*[ ok ] Restarting postgresql (via systemctl): postgresql.service.*

Check if listener changes

*# netstat -nltp | grep post*

The console displays the following message:

tcp 0 0 0.0.0.0:5432 0.0.0.0:\* LISTEN 5310/postgres

tcp6 0 0 :::5432 :::\* LISTEN 5310/postgres

Add the postgres user to the config file which manages the security policy to accept network traffic.

*# vim /etc/postgresql/10/main/pg\_hba.conf*

Content:

# TYPE DATABASE USER ADDRESS METHOD

# IPv4 local connections:

host all all 127.0.0.1/32 md5

host all postgres ***your\_ip/your\_subnet*** md5

Restart the service.

*# /etc/init.d/postgresql restart*

The console displays the following message:

*[ ok ] Restarting postgresql (via systemctl): postgresql.service.*

Create role.

*# su postgres -c psql postgres*

*# CREATE ROLE pgsql WITH SUPERUSER LOGIN PASSWORD '5P4vxpsW7PvCt4Rekk';*

*# \q*

Add new role to pg\_hba.conf file and disable the default user.

*# vim /etc/postgresql/10/main/pg\_hba.conf*

Content:

*# TYPE DATABASE USER ADDRESS METHOD*

*# "local" is for Unix domain socket connections only*

*#local all all peer*

*local all pgsql trust*

Restart the service

*#/etc/init.d/postgresql restart*

The console displays the following message:

*[ ok ] Restarting postgresql (via systemctl): postgresql.service.*

Change password to the pgsql user in template1 (default template for postgres databases, it is created automatically).

*# psql -Upgsql template1 -c "alter user pgsql with password '5P4vx9877PvC\_AKekk';"*

Connect to postgress with the *pgslq* user:

*# psql --host localhost -Upgsql -p5432 --dbname template1*

Insert postgresql information needed to services using other server:

Provide access to postgresql to the new server, adding the following line in the file *“/etc/postgresql/10/main/pg\_hba.conf*”:

*host all all* ***your\_ip/your\_subnet*** *md5*

Acces to postgress from docker-swarm-node1 VM

Install psql service

*# sudo -s*

*# apt install postgresql-client*

Access to postgresql and create new Role, Database, Grant pvivileges and insert data for users.

*CREATE ROLE authadmin WITH NOSUPERUSER LOGIN PASSWORD 'KtexY7FJGQMXzZEwzf';*

*CREATE DATABASE authdb WITH ENCODING='UTF8' OWNER='authadmin';*

*GRANT ALL PRIVILEGES ON DATABASE authdb TO authadmin;*

*GRANT CONNECT ON DATABASE authdb TO authadmin;*

Log out from postgresql, and log in again with the new user:

*# psql –host* ***ip\_postgres\_server*** *–U authadmin –dbname authdb*

Create Sequences, tables, and insert data.

**Note: Inserted information, are fake data and you should use your own data. Password should be a hash password.**

*CREATE SEQUENCE account\_seq;*

*CREATE TABLE accounts (*

*id BIGINT NOT NULL DEFAULT nextval('account\_seq'),*

*username VARCHAR(50) NOT NULL,*

*hash\_password TEXT NOT NULL,*

*platform\_id UUID NOT NULL,*

*enabled BOOLEAN NOT NULL*

*);*

*ALTER TABLE public.accounts ADD CONSTRAINT accounts\_pk PRIMARY KEY (id);*

*INSERT INTO public.accounts (username, hash\_password, platform\_id, enabled) VALUES ('test-place', '$2a$10$1girgkqQ/P8mhW6NK7IXl.koEWkM.LIOOPnJ6kU6wO50jFocKhw/O', '882752f8-9985-460b-8cdf-6a62f93319ac', true);*

*INSERT INTO public.accounts (username, hash\_password, platform\_id, enabled) VALUES ('b015f078536f4094852b9d55dcfcd1da', '$2a$10$1girgkqQ/P8mhW6NK7IXl.koEWkM.LIOOPnJ6kU6wO50jFocKhw/O', 'b015f078-536f-4094-852b-9d55dcfcd1da', true);*

*INSERT INTO public.accounts (username, hash\_password, platform\_id, enabled) VALUES ('882752c8-9985-460b-8cdf-6a6ef90319ac', '$2a$10$1girgkqQ/P8mhW6NK7IXl.koEWkM.LIOOPnJ6kU6wO50jFocKhw/O', 'b015f078-536f-4094-852b-9d55dcfcd1da', true);*

*INSERT INTO public.accounts (username, hash\_password, platform\_id, enabled) VALUES ('b015f078-536f-4094-852b-9d55dcfcd1da', '$2a$10$1girgkqQ/P8mhW6NK7IXl.koEWkM.LIOOPnJ6kU6wO50jFocKhw/O', 'b015f078-536f-4094-852b-9d55dcfcd1da', true);*

Check the added data.

*select \* from public.accounts;*

Modify security settings disabling the pgsql user.

*# vim /etc/postgresql/10/main/pg\_hba.conf*

Content:

*# TYPE DATABASE USER ADDRESS METHOD*

*# "local" is for Unix domain socket connections only*

*#local all all peer*

*#local all pgsql trust*

Restart service

*# /etc/init.d/postgresql restart*

The console displays the following message:

*[ ok ] Restarting postgresql (via systemctl): postgresql.service.*

# Install docker swarm

**Note:**

Docker-swarm-node1 is the manager and a worker.

Docker-swarm-node2 is worker only.

Docker-swarm-node3 is worker only.

Repeat the following steps for each node in order to install docker in all of them:

*# apt-get update*

*# apt-get install apt-transport-https ca-certificates curl software-properties-common*

*# echo deb [arch=amd64]* [*http://download.docker.com/linux/ubuntu*](http://download.docker.com/linux/ubuntu) *bionic stable >> /etc/apt/sources.list.d/docker.list*

*# wget* [*https://download.docker.com/linux/ubuntu/gpg*](https://download.docker.com/linux/ubuntu/gpg) *&& apt-key add gpg*

*# apt-get update && apt-get install docker-ce –*

The following step is only for the node-1 because it is the manager:

*# docker swarm init --advertise-addr* ***docker\_node1\_ip***

This instruction generate some information similar to this:

*Swarm initialized: current node (dxu44lekrn740rotjxipwovf1) is now a manager.*

*To add a worker to this swarm, run the following command:*

*docker swarm join --token SWMTKN-1-5m39oqlzpa76rz3ltri16e720sc4a0bpuxntz1wl4tipen55yw-dw6q42s8w1i3sc3anuvbf4loj* ***docker\_node1\_ip****:2377*

Use the information generated by docker avobe going to the other docker-swarn nodes and create it as workers.

*# docker swarm join --token SWMTKN-1-5m39oqlzpa76rz3ltri16e720sc4a0bpuxntz1wl4tipen55yw-dw6q42s8w1i3sc3anuvbf4loj* ***docker\_node1\_ip****:2377*

Go to Docker-swarm-node1 and check that all nodes are linked.

*# docker node ls*

Create ocds-network

*docker network create \*

*--driver overlay \*

*--subnet 192.168.100.0/24 \*

*--gateway 192.168.100.1 \*

*ocds-network*

# Install portainer

Open Docker-swarm-node-1.

Create docker network and volume:

*# docker network create --driver overlay --attachable portainer\_agent\_network*

It returns an ID similar to cxiy8i4z7ahsf3w34jf78lakh.

Create docker service.

*# docker service create \*

*--name portainer\_agent \*

*--network portainer\_agent\_network \*

*-e AGENT\_CLUSTER\_ADDR=tasks.portainer\_agent \*

*--mode global \*

*--constraint 'node.platform.os == linux' \*

*--mount type=bind,src=//var/run/docker.sock,dst=/var/run/docker.sock \*

*--mount type=bind,src=//var/lib/docker/volumes,dst=/var/lib/docker/volumes \*

*portainer/agent*

In this script “portainer\_agent” is the name of the service and “portainer\_agent\_network” is the network.

Create docker volume

*# docker volume create portainer\_data*

Create portainer service:

*# docker service create \*

*--name portainer \*

*--network portainer\_agent\_network \*

*--publish 9000:9000 \*

*--replicas=1 \*

*--constraint 'node.role == manager' \*

*--mount type=bind,src=//var/run/docker.sock,dst=/var/run/docker.sock \*

*--mount type=volume,src=portainer\_data,dst=/data \*

*portainer/portainer \*

*-H "tcp://tasks.portainer\_agent:9001" --tlsskipverify*

Once created we can acces it using our browser using the defined port (in this case 9000). The access URL should be similar to the following one:

[*http://[public\_server\_ip]:9000/#/init/admin*](http://[public_server_ip]:9000/#/init/admin)

*(The IP will depend on the network where the service was deployed).*

**Note**: *The first time you log in the application it asks the password for the default user (“admin”).*

# Install gitLab

Go to GitLab server.

*# apt-get update*

Install dependencies.

*# apt-get install ca-certificates curl openssh-server postfix*

Some interface will be displayed in the console requiring some parameters. Complete the process leaving default values.

Go into temporal folder.

*# cd /tmp/*

Get script from gitlab.com to install packages.

*# curl -LO https://packages.gitlab.com/install/repositories/gitlab/gitlab-ce/script.deb.sh*

Execute and install the downloaded script.

*# bash /tmp/script.deb.sh*

*# apt-get install gitlab-ce*

Modify main settings content to configure our system.

*# vim /etc/gitlab/gitlab.rb*

Add the following content.

*external\_url '****[server\_public\_ip]****'*

***Note****: you can use the domain name, for example: gitlab.ebrd.system.*

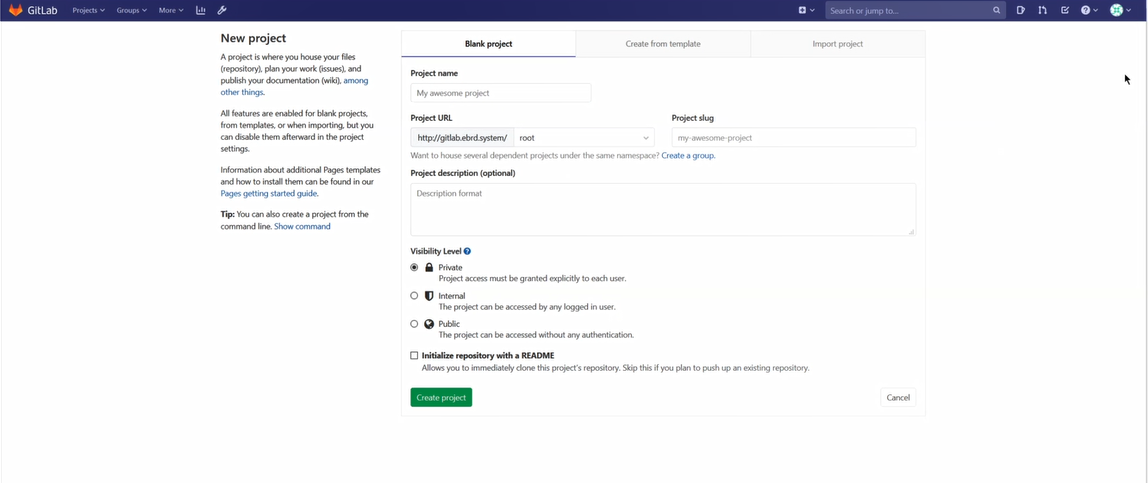
Reload the configuration of gitlab.

*# gitlab-ctl reconfigure*

Access to GitLab with the browser using the name or ip added to the configuration (externam\_url). In our example *'http://gitlab.ebrd.system'*

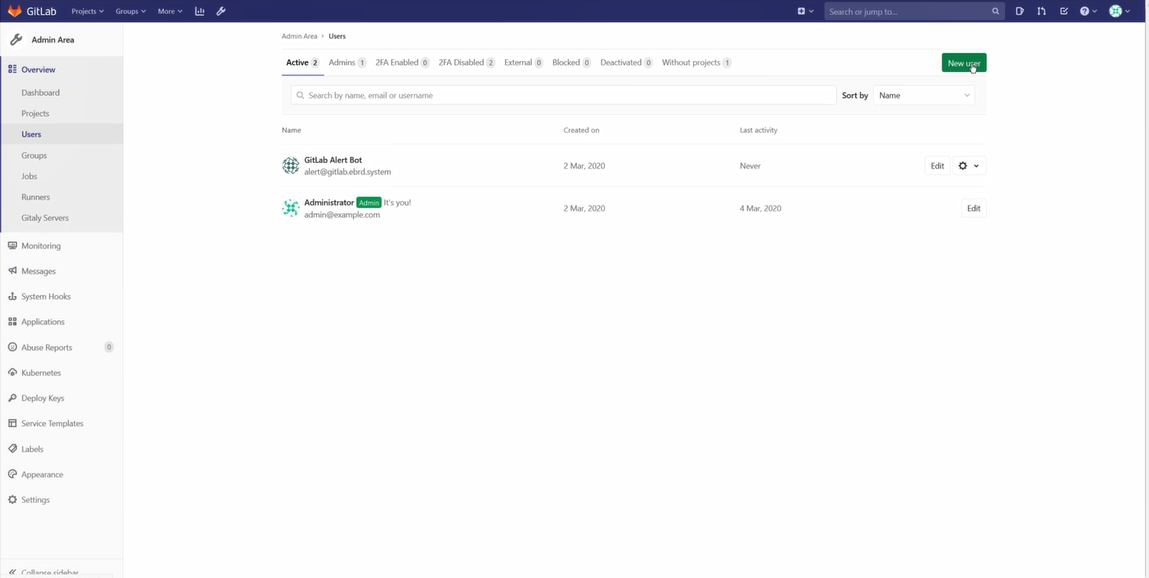
First step on GitLab is to create a new password for the predefined user in the navigator (“root”).

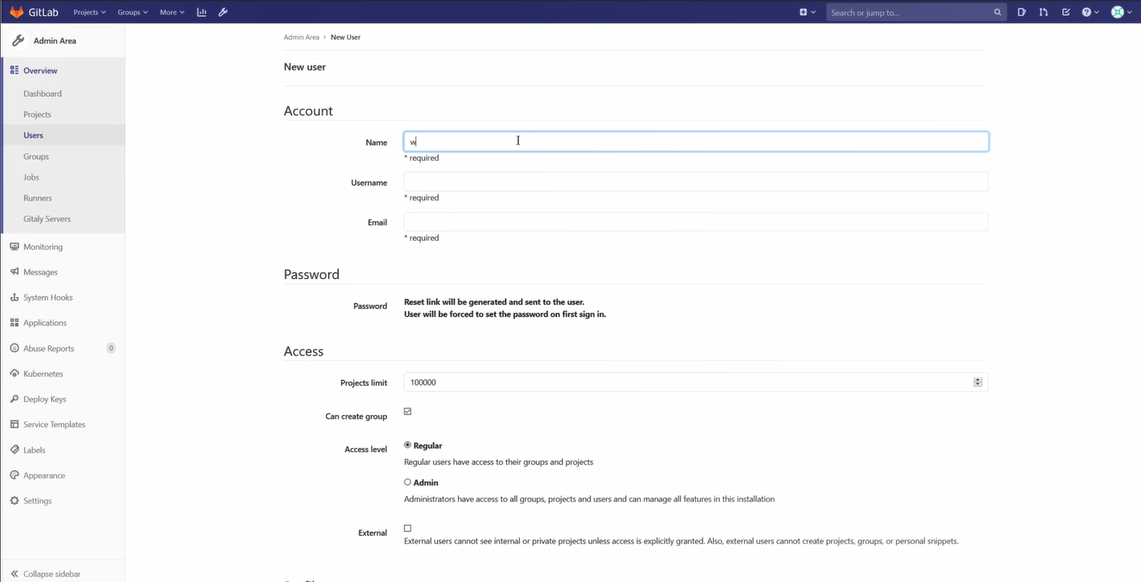
Create a new project in GitLab, using interface:



Add project name and push “Create project” button.

Create users, using the interface:





First user:

**Name**: writers

**Username**: writers

**Email**: [writers@ebrd.system](mailto:writers@ebrd.system)

**Password**: Will be add in a second step

**Can create group**: FALSE

**Access level**: Regular

Second user:

**Name**: conf-server

**Username**: conf-server

**Email**: [conf-server@ebrd.system](mailto:conf-server@ebrd.system)

**Password**: Will be add in a second step

**Can create group**: FALSE

**Access level**: Regular

Third user:

(This user is created to use as administrator user, instead of “Admin” user created by default).

**Name**: admins

**Username**: admins

**Email**: [admins@ebrd.system](mailto:admins@ebrd.system)

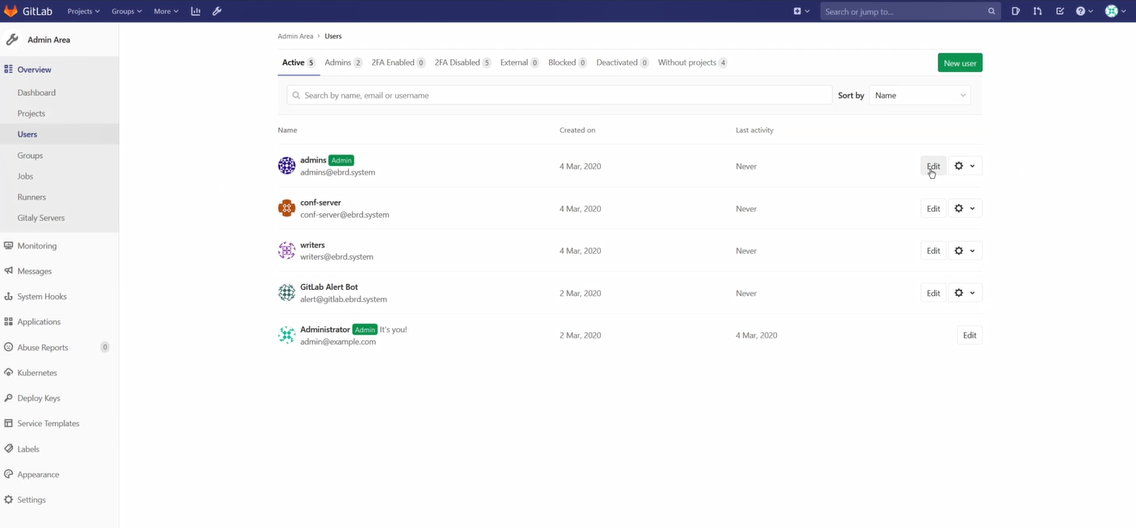
**Password**: Will be add in a second step

**Can create group**: TRUE

**Access level**: Admin

There are two more users created by default: “Administrator” user and “GitLab Alert Bot.

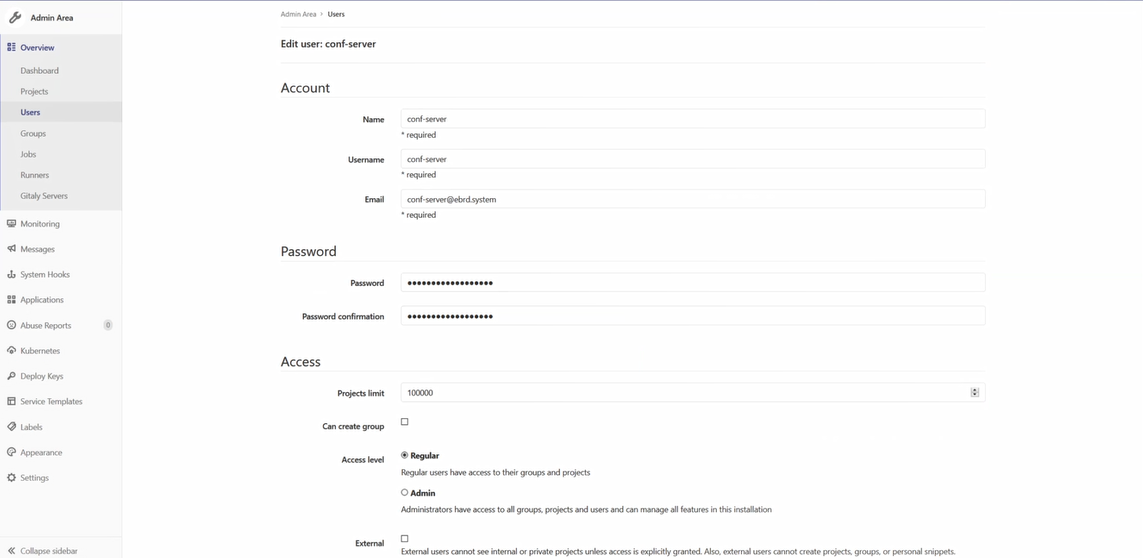
These are all existing users once created our users:



Modify password for the new (three) users:

The password is sent to the email address of the user automatically. But if we do not have access to the email account and we need to use that accounts, we can add a password directly editing the user.

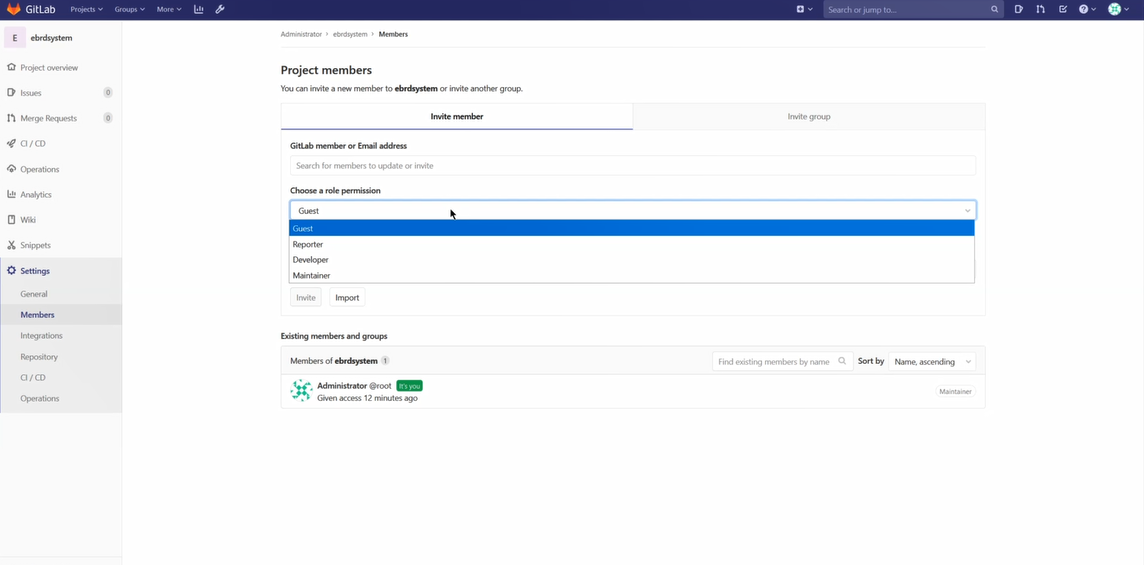
In this screen we can add the password and save the user’s profile.



Groups on GitLab:

By default exist roles “Guest”, “Reporter”, “Developer” and “Maintainer”.

We use that roles to grant permissions to the created users.



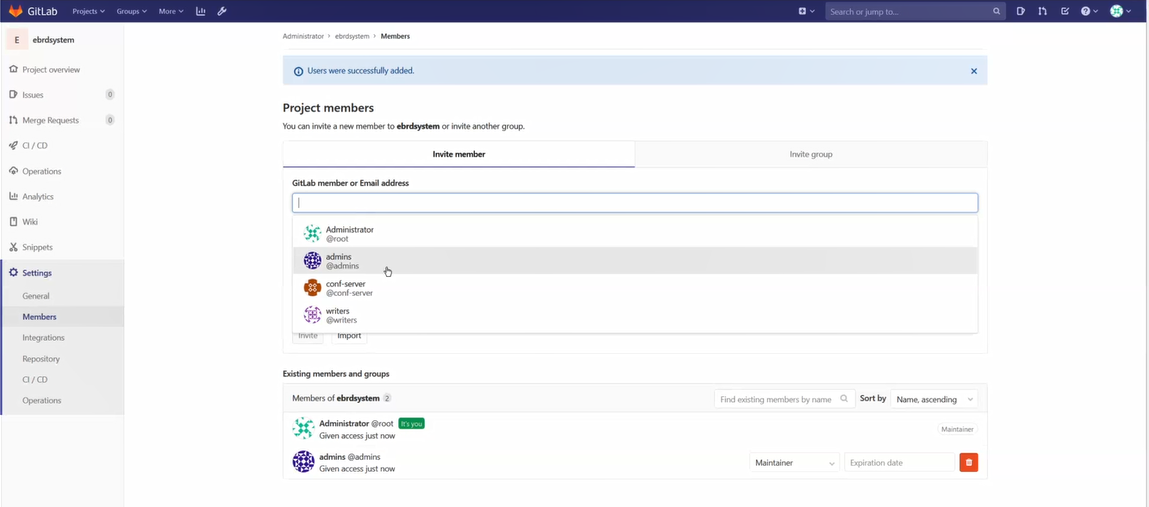
Invite members to the project:

Administrator is @root user.

Add Admins as “Maintainer”.

Add Writers as “Developer”.

Add Config-Server as “Guest”.



Log out as “Administrator” user.

Log in as “admins” user.

Chek the visibility of the project, and the general settings with the current user.

Creating master branch:

Step 1: Go to the project. (It will be empty)

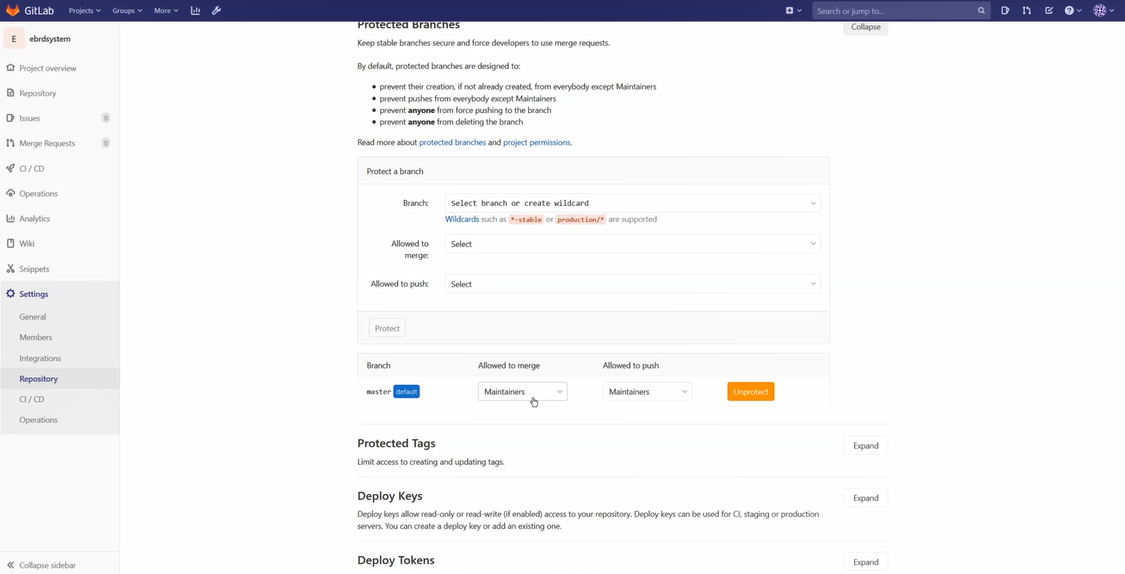
Step 2: Add license to the empty project.

Step 3: Select a template for the license.

Step 4: Commit the file.

File is created in the repository and the master branch was automatically created.

Add Developers + Maintainers profile to the branch in order to allow the merge and the push of source code.



*Branch: \**

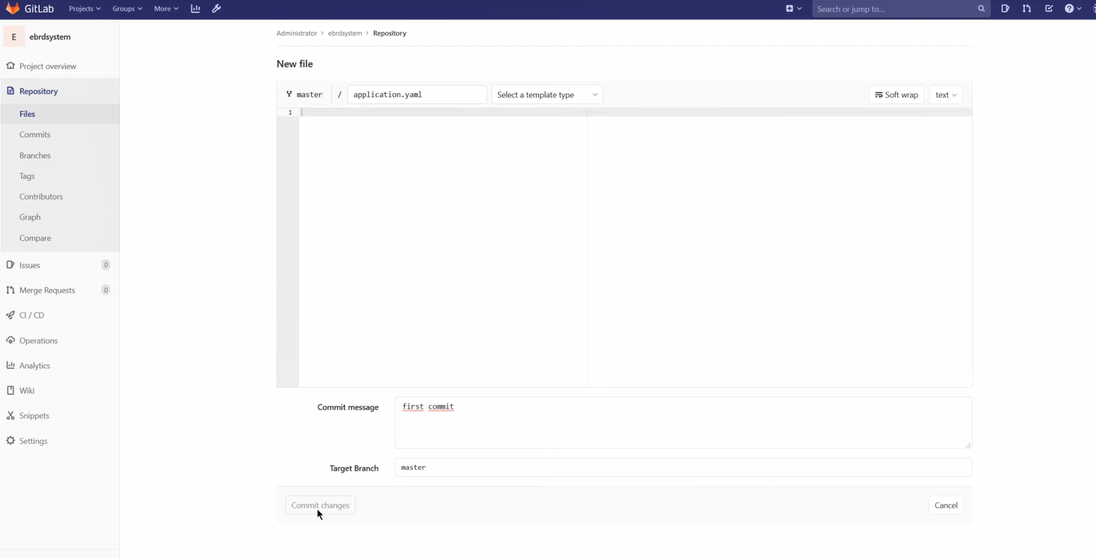
*Allowed to merge: Developers + Maintainers*

*Allowed to push: Developers + Maintainers*

(And push Protect button)

Go to the Repository.

Create a new *empty* file to the master branch named “application.yaml” and commit it.



Logout as “admin” user and login again as “writer” user.

Create new branch called “auth” from “master” branch. This branch should have application.yaml empty file be default.

# Configuration server

Open Docker-swarm-node1.

*# cd /root/*

Create directory for deploys.

*# mkdir deploys*

*# cd deploys*

Create the file “*config-server.sh*” for configuration in the current path and add the following content**:**

*# vim config-server.sh*

Content:

*docker service create --name="config-server" \*

*--env \_\_ PROP\_spring\_cloud\_config\_server\_git \_uri=*

*--env \_\_PROP\_spring\_cloud\_config\_server\_git\_username=conf-server \*

*--env \_\_PROP\_spring\_cloud\_config\_server\_git \_password= \*

*--publish published=8700,target=8080 \*

*--network ocds-network \*

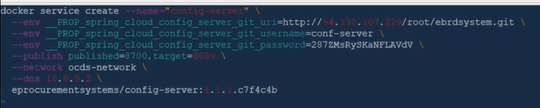
*--dns* ***[ip\_of\_dns\_server]****/ \*

*eprocurementsystems/config-server:1.1.1.c7f4c4b*

Copy the main repository url.

Get the user and password of the GitLab account.

Paste it in the already configuration file “config-server.sh” in the docker-swarp-node1.



*Note: Use the ip of the GitLab server, or the DNS name (for example: gitlab.ebrd.system).*

# Deploy auth service

Generate public and private keys.

Edit application.yaml from auth branch adding the following content (using GitLab interface):

**application.yaml**

spring:

cloud:

inetutils:

preferredNetworks:

- 192.168

datasource:

url: jdbc:postgresql://postgresql.ebrd.system:5432/authdb

driverClassName: org.postgresql.Driver

hikari:

maximum-pool-size: 10

pool-name: auth-pool

data-source-properties:

cachePrepStmts: true

prepStmtCacheSize: 250

prepStmtCacheSqlLimit: 2048

jwt:

life-time:

access: 300 # sec

refresh: 1800 # sec

public-key: |

-----BEGIN PUBLIC KEY-----

**<<add generated public key>>**

-----END PUBLIC KEY-----

private-key: |

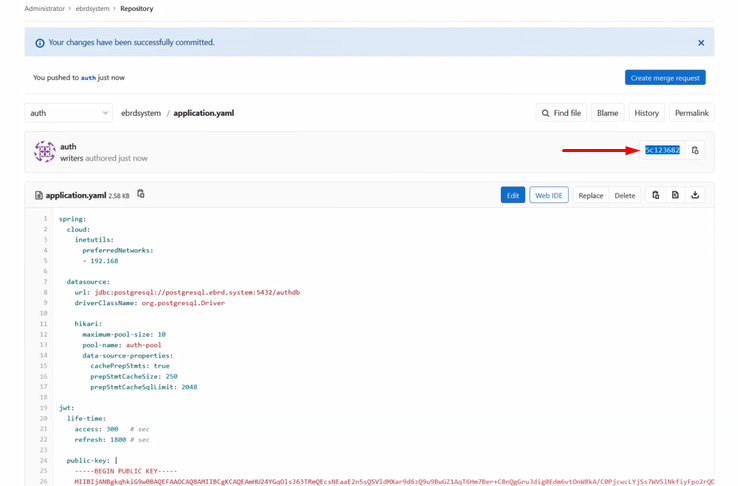
-----BEGIN PRIVATE KEY-----

**<<add generated private key>>**

-----END PRIVATE KEY-----

Commit added the information.

Get the commit number



Create new file “auth.sh” in the path “/root/deploys” of the docker-swarm-node1, with the following content:

*#!/bin/bash*

*docker service create --name="auth" \*

*--env \_\_PROP\_spring\_cloud\_config\_uri=http://config-server:8080 \*

*--env \_\_PROP\_spring\_profiles\_active=default \*

*--env \_\_PROP\_spring\_cloud\_config\_label=[commit\_number] \*

*--env \_\_PROP\_spring\_datasource\_username=[user of the postgress database] \*

*--env \_\_PROP\_spring\_datasource\_password=[password] \*

*--publish published=9011,target=8080 \*

*--network ocds-network \*

*--dns* ***[ip\_of\_dns\_server]*** *\*

*eprocurementsystems/auth:1.1.1.86d307b*

**Note**: User and password, are the credentials of the postgress database user.

Once created, we have to grant permission for the execution:

*# chmod +x auth.sh*

And now execute the service

*# ./auth.sh*

Check in portainer application the deployment way successful.

Check the connection from Docker-swarm-node1 to postgresql server. For this step, needs to install “*psql*” in the docker-swarm-node1, because this is required for connect to the postgresql service.

*# apt-cache search psql*

*# apt-get install postgresql-client-10*

*# psql –host* ***postgres\_server****\_****ip*** *–Uauthadmin –p5432 –dbname authdb*

System will ask the password, and when you insert it, it will access to the postgresql database.

Now we checked the service is reachable, we log out.

Test auth functionality using a HTTP GET call in:

*116.203.48.211:8700/auth/default/5c123682*

# Deploy chronograph service

Create user in the Postgres database going to Postgres server

Log in database:

*# psql --localhost –Upgsql –p5432 --dbname template1*

The system will ask the password, and the will be logged in.

*# su postgres -c psql postgres*

*CREATE ROLE chronoadmin WITH NOSUPERUSER LOGIN PASSWORD 'fD8D3AY8ykvpbeuv2r';*

*CREATE DATABASE chronodb WITH ENCODING='UTF8' OWNER='chronoadmin';*

*GRANT ALL PRIVILEGES ON DATABASE chronodb TO chronoadmin;*

*GRANT CONNECT ON DATABASE chronodb TO chronoadmin;*

Modify postgresql setting to provide access to the host adding the following lines to the file *“/etc/postgresql/10/main/pg\_hba.conf”*:

*host all chronoadmin* docker\_swarm\_node1/32 *md5*

*host all chronoadmin* docker\_swarm\_node2/32 *md5*

*host all chronoadmin* docker\_swarm\_node3/32 *md5*

Restart the service:

*# /etc/init.d/postgresql restart*

Check the connection from VM docker-swarm-node1

*# psql --host* ***postgresql\_server\_ip*** *-Uchronoadmin -p5432 --dbname chronodb*

Once connected, create the tables with the following scripts:

*CREATE SEQUENCE requests\_seq;*

*CREATE TABLE requests*

*(*

*id BIGINT NOT NULL DEFAULT nextval('requests\_seq'),*

*used BOOLEAN NOT NULL DEFAULT FALSE,*

*action VARCHAR(32) NOT NULL,*

*ocid VARCHAR(32) NOT NULL,*

*phase VARCHAR(32) NOT NULL,*

*launch\_time TIMESTAMP,*

*task\_data TEXT,*

*sent\_time TIMESTAMP NOT NULL,*

*received\_time TIMESTAMP NOT NULL*

*);*

*ALTER TABLE ONLY requests ADD CONSTRAINT requests\_pkey PRIMARY KEY (id);*

*CREATE INDEX id\_index ON requests (id) WHERE used is false;*

*CREATE TABLE tasks*

*(*

*request\_id BIGINT NOT NULL,*

*active BOOLEAN NOT NULL,*

*ocid VARCHAR(32) NOT NULL,*

*phase VARCHAR(32) NOT NULL,*

*launch\_time TIMESTAMP NOT NULL,*

*task\_data TEXT NOT NULL,*

*created\_time TIMESTAMP NOT NULL,*

*deactivate\_time TIMESTAMP,*

*canceled\_time TIMESTAMP*

*);*

*ALTER TABLE ONLY tasks ADD CONSTRAINT tasks\_pkey PRIMARY KEY (request\_id);*

*CREATE UNIQUE INDEX tasks\_key\_unq ON tasks (ocid, phase) WHERE active;*

*CREATE INDEX launch\_time\_index ON tasks (launch\_time) WHERE active is true;*

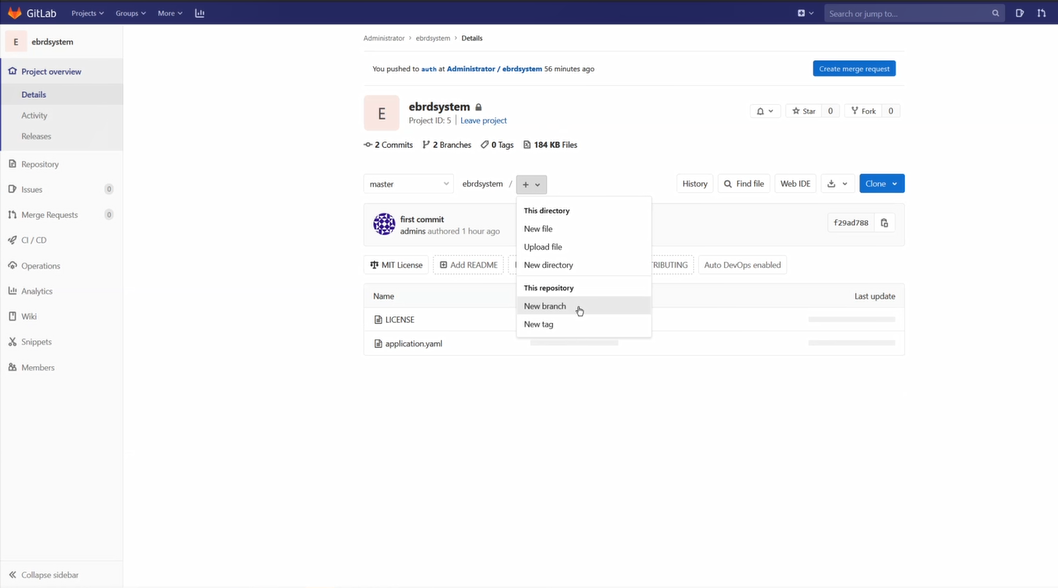
Log in as “writers” user.

Go to the repository.

Create a new branch using the user interface:

Branch name: chronograph

Create form: master



Create “application.yaml” file with the following content:

*spring:*

*datasource:*

*url: jdbc:postgresql://***postgres\_server\_address***:5432/chronodb*

*driverClassName: org.postgresql.Driver*

*hikari:*

*maximum-pool-size: 10*

*pool-name: chronograph-pool*

*data-source-properties:*

*cachePrepStmts: true*

*prepStmtCacheSize: 250*

*prepStmtCacheSqlLimit: 2048*

*kafka:*

*bootstrap-servers:* ***kafka\_node1\_address****:9092,* ***kafka\_node2\_address****:9092,ak-* ***kafka\_node3\_address****:9092*

*producer:*

*retries: 5*

*topic: chronograph-out*

*consumer:*

*topic: chronograph-in*

*group-id: chronograph-consumer-group*

*enable-auto-commit: false*

*ticker:*

*process:*

*repeat-time: 300000*

*advance-time: 120000*

*cache:*

*repeat-time: 1000*

Commit the updates.

*cd /home/kafka*

*bin/kafka-topics.sh \*

*--create \*

*--zookeeper ak-node1.ebrd.system:2181,ak-node2.ebrd.system:2181,ak-node3.ebrd.system:2181 \*

*--replication-factor 2 \*

*--partitions 1 \*

*--topic* ***chronograph-out***

*bin/kafka-topics.sh \*

*--create \*

*--zookeeper ak-node1.ebrd.system:2181,ak-node2.ebrd.system:2181,ak-node3.ebrd.system:21811 \*

*--replication-factor 2 \*

*--partitions 1 \*

*--topic* ***chronograph-in***

Create Kafka topics “chronograph-out” and “chronograph-in” for kafka-cluster as the yaml file defines:

Go to Kafka-node1 and create a new file “chronograph.sh” in the path “/root/deploys” of the Docker-swarm-node1, with the following content:

*#!/bin/bash*

*docker service create --name="chronograph" \*

*--env \_\_PROP\_spring\_cloud\_config\_uri=http://config-server:8080 \*

*--env \_\_PROP\_spring\_profiles\_active=default \*

*--env \_\_PROP\_spring\_cloud\_config\_label=[commit\_number] \*

*--env \_\_PROP\_spring\_datasource\_username=[user] \*

*--env \_\_PROP\_spring\_datasource\_password=[password] \*

*--network ocds-network \*

*--dns* ***[ip\_of\_dns\_server]*** *\*

*eprocurementsystems/chronograph:1.1.1.8725140*

Grant permission to file:

*# chmod +x chronograph.sh*

And deploy the service:

*# ./chronograph.sh*

# Deploy forms service

This service does not requires application.yaml configuration, but the branch and the file must exist.

Log in as “writers” user.

Go to the repository.

Create a new branch:

Branch name: *forms*

Create form: *master*

Create “application.yaml” file without content.

Edit the file.

Add an empty line.

Commit the modification in order to have a new commit number.

Create new file “forms.sh” in the path “/root/deploys” of the Docker-swarm-node, with the following content:

*#!/bin/bash*

*docker service create --name="forms" \*

*--env \_\_PROP\_spring\_cloud\_config\_uri=http://config-server:8080 \*

*--env \_\_PROP\_spring\_profiles\_active=default \*

*--env \_\_PROP\_spring\_cloud\_config\_label=[commit\_number] \*

*--publish published=9171,target=8080 \*

*--network ocds-network \*

*--dns* ***[ip\_of\_dns\_server]*** *\*

*eprocurementsystems/forms:1.1.1.562f10*

Grant permission to file:

*# chmod +x forms.sh*

And deploy the service:

*# ./forms.sh*

# Deploy gateway service

Log in in GitLab as “writers” user.

Go to the repository.

Create a new branch:

Branch name: *gateway*

Create form: *master*

Create “application.yaml” file with the following content:

*spring:*

*cloud:*

*inetutils:*

*preferredNetworks:*

*- 192.168*

*servlet:*

*multipart:*

*enabled: true*

*max-file-size: 50Mb*

*max-request-size: 50Mb*

*zuul:*

*rsa-filter:*

*exclude:*

*- auth-tokens*

*- auth-refresh*

*- auth-docs*

*- storage-public*

*sensitiveHeaders:*

*# sensitive-headers: Cookie,Set-Cookie,Authorization*

*# ignore-security-headers: false*

*prefix: /api/v1*

*ignoredServices: '\*'*

*routes:*

*auth-tokens:*

*path: /auth/signin*

*url: http://auth:8080*

*stripPrefix: false*

*auth-refresh:*

*path: /auth/refresh*

*url: http://auth:8080*

*stripPrefix: false*

*auth-docs:*

*path: /auth/docs/\*\**

*url: http://auth:8080*

*stripPrefix: false*

*operation:*

*path: /operations/\*\**

*url: http://operation:8080*

*stripPrefix: false*

*bpe:*

*path: /\*\**

*url: http://bpe:8080*

*stripPrefix: false*

*storage-public:*

*path: /storage/get/\*\**

*url: http://storage:8080*

*stripPrefix: false*

*storage-private:*

*path: /storage/\*\**

*url: http://storage:8080*

*stripPrefix: false*

*eureka:*

*client:*

*enabled: false*

*ribbon:*

*eureka:*

*enabled: false*

*feign:*

*hystrix:*

*enabled: false*

Edit the file.

Add an empty line.

Commit the modification in order to have a new commit number.

Create new file “gateway.sh” in the path “/root/deploys” of the docker-swarm-node1 VM, with the following content:

*#!/bin/bash*

*docker service create --name="gateway" \*

*--env \_\_PROP\_spring\_cloud\_config\_uri=http://config-server:8080 \*

*--env \_\_PROP\_spring\_profiles\_active=default \*

*--env \_\_PROP\_spring\_cloud\_config\_label=[commit\_number] \*

*--publish published=8900,target=8080 \*

*--network ocds-network \*

*--dns* ***[ip\_of\_dns\_server]*** *\*

*eprocurementsystems/gateway:1.1.1.f8f61e9*

Grant permission to file:

*# chmod +x gateway.sh*

And deploy the service:

*# ./ gateway.sh*

# Deploy mdm service

Go to Postgress.

Access to postgres.

*# psql --host localhost -Upgsql -p5432 --dbname template1*

Create role.

*CREATE ROLE mdmadmin WITH NOSUPERUSER LOGIN PASSWORD 'uxxBqaKFqtvDwd86B6';*

*CREATE DATABASE mdmdb WITH ENCODING='UTF8' OWNER='mdmadmin';*

*GRANT ALL PRIVILEGES ON DATABASE mdmdb TO mdmadmin;*

*GRANT CONNECT ON DATABASE mdmdb TO mdmadmin;*

*--liquibase formatted sql*

*--changeset msambulat:create\_old\_structure*

Logut as pgsql and login postgress to *mdmdb* database using *mdmadmin* users.

In order to create the database and the structure, go to the annex1.docx.

To add the data to the tables, go to the “dump.zip”, extract the sql files, and import the information.

Modify postgresql setting to provide access to the host adding the following lines to the file “/etc/postgresql/10/main/pg\_hba.conf”:

host all mdmadmin docker\_swarm\_node1/32 md5

host all mdmadmin docker\_swarm\_node2/32 md5

host all mdmadmin docker\_swarm\_node3/32 md5

Restart postgresql service

*# /etc/init.d/postgresql restart*

Check the connection from VM docker-swarm-node1

*# psql --host postgres\_server\_ip\_ -Umdmadmin -p5432 --dbname mdmdb*

Log in in GitLab as “writers” user.

Go to the repository and create a new branch:

Branch name: *mdm*

Create form: *master*

Create “application.yaml” file with the following content:

*spring:*

*datasource:*

*driver-class-name: org.postgresql.Driver*

*url: jdbc:postgresql://postgresql.ebrd.system:5432/mdmdb*

*jpa:*

*properties*

*hibernate:*

*jdbc:*

*lob:*

*non\_contextual\_creation: true*

Commit information.

Edit the file.

Add an empty line at the end.

Commit the modification in order to have a new commit number.

Create new file “mdm.sh” in the path “/root/deploys” of the Docker-swarm-node1, with the following content:

*#!/bin/bash*

*docker service create --name="mdm" \*

*--env \_\_PROP\_spring\_cloud\_config\_uri=http://config-server:8080 \*

*--env \_\_PROP\_spring\_profiles\_active=default \*

*--env \_\_PROP\_spring\_cloud\_config\_label=[commit\_number] \*

*--env \_\_PROP\_spring\_datasource\_username= [postgres\_database\_username] \*

*--env \_\_PROP\_spring\_datasource\_password= [postgres\_database\_password] \*

*--env \_\_PROP\_spring\_jpa\_hibernate\_dll-auto=update \*

*--publish published=8900,target=8080 \*

*--network ocds-network \*

*--dns* ***[ip\_of\_dns\_server]*** *\*

*eprocurementsystems/gateway:1.1.1.9a3ef38*

Grant permission to file:

*# chmod +x mdm.sh*

And deploy the service:

*# ./mdm.sh*

# Deploy public-point service

Log in GitLab as “writers” user.

Go to the repository.

Create a new branch:

Branch name: *public-point*

Create form: *master*

Create “application.yaml” file with the following content:

*spring:*

*profiles: default*

*cloud:*

*inetutils:*

*preferredNetworks:*

*- 192.168*

*data:*

*cassandra:*

*contact-points:* ***cassandra\_server\_node1****,* ***cassandra\_server\_node2****,* ***cassandra\_server\_node3***

*keyspace-name: ocds*

*port: 9042*

*ocds:*

*path: http://public.ebrd.system/*

*version: 1.1*

*license: http://opendefinition.org/licenses/*

*publicationPolicy: http://opendefinition.org/licenses/*

*extensions:*

*https://raw.githubusercontent.com/open-contracting/ocds\_bid\_extension/v1.1.1/extension.json,*

*https://raw.githubusercontent.com/open-contracting/ocds\_enquiry\_extension/v1.1.1/extension.json*

*publisherName: M-Tender*

*publisherUri: https://www.ebrd.system*

*defLimit: 5*

*maxLimit: 10*

*logging:*

*pattern:*

*console: "%d{yyyy-MM-dd HH:mm:ss} %-5level %logger{36} - %msg%n"*

Edit the file.

Add an empty line.

Commit the modification in order to have a new commit number.

Create new file “public-point.sh” in the path “/root/deploys” of the Docker-swarm-node1, with the following content:

*#!/bin/bash*

*docker service create --name="public-point" \*

*--env \_\_PROP\_spring\_cloud\_config\_uri=http://config-server:8080 \*

*--env \_\_PROP\_spring\_profiles\_active=default \*

*--env \_\_PROP\_spring\_cloud\_config\_label=27be359f \*

*--env \_\_PROP\_spring\_data\_cassandra\_username=caclient \*

*--env \_\_PROP\_spring\_data\_cassandra\_password=rMgdGszsgPv79Ts7CB \*

*--publish published=9111,target=8080 \*

*--network ocds-network \*

*eprocurementsystems/public-point:1.1.1.8f75880*

Grant permission to file:

*# chmod +x public-point.sh*

And deploy the service:

*# ./public-point.sh*

# Deploy storage service

Create volume in docker

Go to Docker-swarm-node1

*# docker volume create storage-vol*

Log in GitLab as “writers” user.

Go to the repository.

Create a new branch:

Branch name: *storage*

Create form: *master*

Create “application.yaml” file with the following content:

*spring:*

*profiles: default*

*cloud:*

*inetutils:*

*preferredNetworks:*

*- 192.168*

*servlet:*

*multipart:*

*enabled: true*

*max-file-size: 50Mb*

*max-request-size: 50Mb*

*cassandra:*

*contact-points: ac-node1.ebrd.system,ac-node2.ebrd.system,ac-node3.ebrd.system*

*keyspace-name: ocds*

*port: 9042*

*upload:*

*file:*

*max-weight: 52428800*

*path: http://storage.ebrd.system/get/*

*folder: /storage*

*extensions: jpg,png,gif,tif,rar,zip,7s,doc,docx,xls,xlsx,pdf,rtf,txt*

*logging:*

*pattern:*

*console: "%d{yyyy-MM-dd HH:mm:ss} %-5level %logger{36} - %msg%n"*

Edit the file.

Add an empty line.

Commit the modification in order to have a new commit number.

Create Cassandra table going to Cassandra-node1

*cqlsh -u caroot* ***Cassandra\_node\_IP***

*>use ocds;*

*> CREATE TABLE IF NOT EXISTS ocds.storage\_files (*

*file\_id text,*

*file\_is\_open boolean,*

*date\_modified timestamp,*

*date\_published timestamp,*

*file\_hash text,*

*file\_weight bigint,*

*file\_name text,*

*file\_on\_server text,*

*file\_owner text,*

*primary key(file\_id)*

*);*

Create new file “storage.sh” in the path “/root/deploys” of the docker-swarm-node1 VM, with the following content:

*#!/bin/bash*

*docker service create --name="storage" \*

*--env \_\_PROP\_spring\_cloud\_config\_uri=http://config-server:8080 \*

*--env \_\_PROP\_spring\_profiles\_active=default \*

*--env \_\_PROP\_spring\_cloud\_config\_label=[commit\_number] \*

*--env \_\_PROP\_cassandra\_username=caclient \*

*--env \_\_PROP\_cassandra\_password=[caclient\_password]caroot \*

*--mount src=storage-vol,dst=/storage \*

*--publish published=9131,target=8080 \*

*--network ocds-network \*

*--dns* ***[ip\_of\_dns\_server]*** *\*

*--constraint 'node.role == manager' \*

*eprocurementsystems/storage:1.1.1.82e489e*

Grant permission to file:

*# chmod +x storage.sh*

And deploy the service:

*# ./ storage.sh*

# Deploy kafka notifier service

Log in GitLab as “writers” user.

Go to the repository.

And create a new branch:

Branch name: *notifier-kafka*

Create form: *master*

Create “application.yaml” file with the following content:

*spring:*

*cloud:*

*inetutils:*

*preferredNetworks:*

*- 192.168*

*kafka:*

*consumer:*

*bootstrap-servers:* ***kafka\_node1\_address****:9092,* ***kafka\_node2\_address****:9092,ak-* ***kafka\_node3\_address****:9092*

*group-id: notification-kafka-consumer-group*

*enable-auto-commit: false*

*producer:*

*bootstrap-servers: feed.ebrd.system:9092*

*retries: 5*

*listener:*

*ack-mode: manual\_immediate*

*cassandra:*

*contact-points:* ***cassandra\_server\_node1****,* ***cassandra\_server\_node2****,* ***cassandra\_server\_node3***

*keyspace-name: ocds*

*port: 9042*

Edit the file.

Add an empty line.

Commit the modification in order to have a new commit number.

Create new file “notifier-kafka.sh” in the path “/root/deploys” of the docker-swarm-node1 VM, with the following content:

*docker service create --name="notifier-kafka" \*

*--env \_\_PROP\_spring\_cloud\_config\_uri=http://config-server:8080 \*

*--env \_\_PROP\_spring\_profiles\_active=default \*

*--env \_\_PROP\_spring\_cloud\_config\_label=[commit\_number] \*

*--env \_\_PROP\_spring\_kafka\_producer\_security\_username=[admin\_user\_kafkaProducer-] \*

*--env \_\_PROP\_spring\_kafka\_producer\_security\_password=[password\_of\_YodaAdmin] \*

*--env \_\_PROP\_cassandra\_username=caclient \*

*--env \_\_PROP\_cassandra\_password=[cassandra\_database\_password] \*

*--publish published=8900,target=8080 \*

*--network ocds-network \*

*eprocurementsystems/gateway:1.1.1.ab82f91*

Grant permission to file:

*# chmod +x notifier-kafka.sh*

And deploy the service:

*# ./ notifier-kafka.sh*

# Cassandra ocds keyspace creation

Log in GitLab as “writers” user.

Go to the repository.

Create a new branch:

Branch name: *operation*

Create form: *master*

Create “application.yaml” file with the following content:

*spring:*

*cloud:*

*inetutils:*

*preferredNetworks:*

* *192.168*

*cassandra:*

*contact-points: ac-node1.ebrd.system, ac-node2.ebrd.system, ac-node3.ebrd.system*

*keyspace-name: ocds*

*port: 9042*

Commit information

Create new file “operation.sh” in the path “/root/deploys” of the docker-swarm-node1 VM, with the following content:

*docker service create --name="operation" \*

*--env \_\_PROP\_spring\_cloud\_config\_uri=http://config-server:8080 \*

*--env \_\_PROP\_spring\_profiles\_active=default \*

*--env \_\_PROP\_spring\_cloud\_config\_label=[commit\_number] \*

*--env \_\_PROP\_cassandra\_username=caclient \*

*--env \_\_PROP\_cassandra\_password=[caclient\_password] \*

*--publish published=9021,target=8080 \*

*--network ocds-network \*

*--dns* ***[ip\_of\_dns\_server]*** *\*

*eprocurementsystems/operation:1.1.1.9c9a5be*

Grant permission to file:

*# chmod +x operation.sh*

Update the dependencies:

*# apt-get update*

Install python with setup tools:

*# apt-get install python3-pip*

*# pip3 install setuptools*

Instal cqlsh for client:

*#pip3 install cqlsh=4.0.1*

And deploy the service:

*# ./ operation.sh*

# Deploy BPE service

Go to Postgres server and access postgres.

*# psql --host localhost -Upgsql -p5432 --dbname template1*

Create role.

*CREATE ROLE bpeadmin WITH NOSUPERUSER LOGIN PASSWORD 'kExThkZU4Hz4pnTqFG';*

Create database.

*CREATE DATABASE bpedb WITH ENCODING='UTF8' OWNER='bpeadmin';*

Grant permissions.

*GRANT ALL PRIVILEGES ON DATABASE bpedb TO bpeadmin;*

*GRANT CONNECT ON DATABASE bpedb TO bpeadmin;*

Go to Cassandra-node1

*# cqlsh -u caroot* ***cassandra\_node1\_ip***

>use ocds;

In order to create the structure and insert the data, go to annex2.docx.

Log in GitLab as “writers” user.

Go to the repository and create a new branch:

Branch name: *bpe*

Create form: *master*

Create “application.yaml” file with the following content:

*spring:*

*profiles: default*

*datasource:*

*driver-class-name: org.postgresql.Driver*

*url: jdbc:postgresql://postgresql.ebrd.system:5432/bpedb*

*hikari:*

*maximum-pool-size: 10*

*pool-name: bpe-pool*

*kafka:*

*producer:*

*retries: 5*

*internal-bootstrap: ak-node1.ebrd.system:9092,ak-node2.ebrd.system:9092,ak-node3.ebrd.system:9092*

*consumer:*

*internal-bootstrap: ak-node1.ebrd.system:9092,ak-node2.ebrd.system:9092,ak-node3.ebrd.system:9092*

*group: orchestrator*

*cassandra:*

*contact-points: ac-node1.ebrd.system,ac-node2.ebrd.system,ac-node3.ebrd.system*

*keyspace-name: ocds*

*camunda:*

*bpm:*

*auto-deployment-enabled: true*

*history-level: none*

*feign:*

*client:*

*config:*

*default:*

*connectTimeout: 10000*

*readTimeout: 10000*

*uri:*

*budget: http://public.ebrd.system/budgets/*

*tender: http://public.ebrd.system/tenders/*

*ribbon:*

*eureka:*

*enabled: false*

*e-budget:*

*ribbon:*

*listOfServers: http://e-budget:8080*

*e-access:*

*ribbon:*

*listOfServers: http://e-access:8080*

*e-auction:*

*ribbon:*

*listOfServers: http://e-auction:8080*

*e-notice:*

*ribbon:*

*listOfServers: http://e-notice:8080*

*e-submission:*

*ribbon:*

*listOfServers: http://e-submission:8080*

*e-clarification:*

*ribbon:*

*listOfServers: http://e-clarification:8080*

*e-storage:*

*ribbon:*

*listOfServers: http://storage:8080*

*e-qualification:*

*ribbon:*

*listOfServers: http://e-qualification:8080*

*e-evaluation:*

*ribbon:*

*listOfServers: http://e-evaluation:8080*

*e-contracting:*

*ribbon:*

*listOfServers: http://e-contracting:8080*

*e-regulation:*

*ribbon:*

*listOfServers: http://e-regulation:8080*

*e-mdm:*

*ribbon:*

*listOfServers: http://mdm:8080*

*e-dossier:*

*ribbon:*

*listOfServers: http://e-dossier:8080*

*logging:*

*# level:*

*# root: WARN*

*pattern:*

*console: "%d{yyyy-MM-dd HH:mm:ss} %-5level %logger{36} - %msg%n"*

Edit the file.

Add an empty line.

Commit the modification in order to have a new commit number.

Create new file “bpe.sh” in the path “/root/deploys” of the Docker-swarm-node1, with the following content:

*#!/bin/bash*

*docker service create --name="bpe" \*

*--env \_\_PROP\_spring\_cloud\_config\_uri=http://config-server:8080 \*

*--env \_\_PROP\_spring\_profiles\_active=default \*

*--env \_\_PROP\_spring\_cloud\_config\_label=[commit\_number] \*

*--env \_\_PROP\_spring\_datasource\_username=[bpeadmin\_postgress\_user] \*

*--env \_\_PROP\_spring\_datasource\_password=[password\_of\_bpeadmin] \*

*--env \_\_PROP\_cassandra\_username=[caclient\_cassandra\_user] \*

*--env \_\_PROP\_cassandra\_password=[caclient\_password] \*

*--publish published=9101,target=8080 \*

*--network ocds-network \*

*--dns* ***[ip\_of\_dns\_server]*** *\*

*eprocurementsystems/bpe:1.1.1.f5f9207*

Grant permission to file:

*# chmod +x bpe.sh*

And deploy the service:

*# ./ bpe.sh*

# Deploy e-Access service

Go to Cassandra-node1

*# cqlsh -u caroot* ***cassandra\_node1\_ip***

*use ocds;*

*CREATE TABLE IF NOT EXISTS ocds.access\_tender (*

*cp\_id text,*

*stage text,*

*token\_entity uuid,*

*owner text,*

*created\_date timestamp,*

*json\_data text,*

*primary key(cp\_id, stage, token\_entity)*

*);*

*CREATE TABLE IF NOT EXISTS ocds.access\_history (*

*operation\_id text,*

*command text,*

*operation\_date timestamp,*

*json\_data text,*

*primary key(operation\_id, command)*

*);*

*CREATE TABLE IF NOT EXISTS ocds.access\_rules (*

*country text,*

*pmd text,*

*parameter text,*

*value text,*

*primary key(country, pmd, parameter)*

*);*

*INSERT INTO ocds.access\_rules (country, pmd, parameter, value) VALUES ('MD', 'OT', 'goods', 'false');*

*INSERT INTO ocds.access\_rules (country, pmd, parameter, value) VALUES ('MD', 'OT', 'services', 'false');*

*INSERT INTO ocds.access\_rules (country, pmd, parameter, value) VALUES ('MD', 'OT', 'works', 'false');*

*INSERT INTO ocds.access\_rules (country, pmd, parameter, value) VALUES ('MD', 'SV', 'goods', 'true');*

*INSERT INTO ocds.access\_rules (country, pmd, parameter, value) VALUES ('MD', 'SV', 'services', 'true');*

*INSERT INTO ocds.access\_rules (country, pmd, parameter, value) VALUES ('MD', 'SV', 'works', 'false');*

*INSERT INTO ocds.access\_rules (country, pmd, parameter, value) VALUES ('MD', 'TEST\_OT', 'goods', 'false');*

*INSERT INTO ocds.access\_rules (country, pmd, parameter, value) VALUES ('MD', 'TEST\_OT', 'services', 'false');*

*INSERT INTO ocds.access\_rules (country, pmd, parameter, value) VALUES ('MD', 'TEST\_OT', 'works', 'false');*

*INSERT INTO ocds.access\_rules (country, pmd, parameter, value) VALUES ('MD', 'TEST\_SV', 'goods', 'true');*

*INSERT INTO ocds.access\_rules (country, pmd, parameter, value) VALUES ('MD', 'TEST\_SV', 'services', 'true');*

*INSERT INTO ocds.access\_rules (country, pmd, parameter, value) VALUES ('MD', 'TEST\_SV', 'works', 'false');*

*INSERT INTO ocds.access\_rules (country,pmd,parameter,value) VALUES ('MD','MV','goods','false');*

*INSERT INTO ocds.access\_rules (country,pmd,parameter,value) VALUES ('MD','MV','services','false');*

*INSERT INTO ocds.access\_rules (country,pmd,parameter,value) VALUES ('MD','MV','works','false');*

*INSERT INTO ocds.access\_rules (country,pmd,parameter,value) VALUES ('MD','TEST\_MV','goods','false');*

*INSERT INTO ocds.access\_rules (country,pmd,parameter,value) VALUES ('MD','TEST\_MV','services','false');*

*INSERT INTO ocds.access\_rules (country,pmd,parameter,value) VALUES ('MD','TEST\_MV','works','false');*

Log in GitLab as “writers” user.

Go to the repository and create a new branch:

Branch name: *e-access*

Create form: *master*

Create “application.yaml” file with the following content:

*spring:*

*profiles: default*

*cloud:*

*inetutils:*

*preferredNetworks:*

*- 192.168*

*cassandra:*

*contact-points: ac-node1.ebrd.system,ac-node2.ebrd.system,ac-node3.ebrd.system*

*keyspace-name: ocds*

*port: 9042*

*# ocds.prefix: ocds-b3wdp1*

*ocds:*

*prefixes:*

*main: ocds-b3wdp1*

*test: test-b3wdp1*

*logging:*

*pattern:*

*console: "%d{yyyy-MM-dd HH:mm:ss} %-5level %logger{36} - %msg%n"*

Edit the file.

Add an empty line.

Commit the modification in order to have a new commit number.

Create new file “e-access.sh” in the path “/root/deploys” of the Docker-swarm-node1, with the following content:

*#!/bin/bash*

*docker service create --name="e-access" \*

*--env \_\_PROP\_spring\_cloud\_config\_uri=http://config-server:8080 \*

*--env \_\_PROP\_spring\_profiles\_active=default \*

*--env \_\_PROP\_spring\_cloud\_config\_label=[commit\_number] \*

*--env \_\_PROP\_cassandra\_username=caclient \*

*--env \_\_PROP\_cassandra\_password=[caclient\_password] \*

*--publish published=9031,target=8080 \*

*--network ocds-network \*

*--dns* ***[ip\_of\_dns\_server]*** *\*

*eprocurementsystems/e-access:1.1.1.c0cbc3b*

Grant permission to file:

*# chmod +x e-access.sh*

And deploy the service:

*# ./ e-access.sh*

# Deploy e-Auction service

Go to Cassandra-node1

*# cqlsh -u caroot* ***cassandra\_node1\_ip***

*CREATE KEYSPACE IF NOT EXISTS auctions WITH replication = { 'class' : 'SimpleStrategy', 'replication\_factor' : 3 };*

*>use auctions;*

*CREATE TABLE IF NOT EXISTS auctions.calendar (*

*country text,*

*year int,*

*month int,*

*work\_days set<int>,*

*PRIMARY KEY ((country, year, month))*

*);*

*CREATE TABLE IF NOT EXISTS auctions.buckets (*

*date date,*

*country text,*

*row\_version int,*

*api\_version text,*

*operation\_id text,*

*slots text,*

*occupancy text,*

*PRIMARY KEY ((date, country))*

*);*

*CREATE TABLE IF NOT EXISTS auctions.auctions (*

*cpid text,*

*row\_version int,*

*api\_version text,*

*operation\_id text,*

*country text,*

*status int,*

*data text,*

*PRIMARY KEY (cpid)*

*);*

*INSERT INTO auctions.calendar(country,year,month, work\_days) VALUES ('MD', 2018, 10, {1,2,3,4,5,8,9,10,11,12,15,16,17,18,19,22,23,24,25,26,29,30,31}) IF NOT EXISTS;*

*INSERT INTO auctions.calendar(country,year,month, work\_days) VALUES ('MD', 2018, 11, {1,2,5,6,7,8,9,12,13,14,15,16,19,20,21,22,23,26,27,28,29,30}) IF NOT EXISTS;*

*INSERT INTO auctions.calendar(country,year,month, work\_days) VALUES ('MD', 2018, 12, {3,4,5,6,7,10,11,12,13,14,17,18,19,20,21,24,25,26,27,28}) IF NOT EXISTS;*

*INSERT INTO auctions.calendar(country,year,month, work\_days) VALUES ('MD', 2019, 1, {2,3,4,9,10,11,14,15,16,17,18,21,22,23,24,25,28,29,30,31}) IF NOT EXISTS;*

*INSERT INTO auctions.calendar(country,year,month, work\_days) VALUES ('MD', 2019, 2, {1,4,5,6,7,8,11,12,13,14,15,18,19,20,21,22,25,26,27}) IF NOT EXISTS;*

*INSERT INTO auctions.calendar(country,year,month, work\_days) VALUES ('MD', 2019, 3, {4,5,6,11,12,13,14,15,18,19,20,21,22,25,26,27,28,29}) IF NOT EXISTS;*

*INSERT INTO auctions.calendar(country,year,month, work\_days) VALUES ('MD', 2019, 4, {1,2,3,4,5,8,9,10,11,12,15,16,17,18,19,22,23,24,25,26,29,30}) IF NOT EXISTS;*

*INSERT INTO auctions.calendar(country,year,month, work\_days) VALUES ('MD', 2019, 5, {1,2,3,6,7,10,13,14,15,16,17,20,21,22,23,24,27,28,29,30,31}) IF NOT EXISTS;*

*INSERT INTO auctions.calendar(country,year,month, work\_days) VALUES ('MD', 2019, 6, {3,4,5,6,7,10,11,12,13,14,17,18,19,20,21,24,25,26,27,28}) IF NOT EXISTS;*

*INSERT INTO auctions.calendar(country,year,month, work\_days) VALUES ('MD', 2019, 7, {1,2,3,4,5,8,9,10,11,12,15,16,17,18,19,22,23,24,25,26,29,30,31}) IF NOT EXISTS;*

*INSERT INTO auctions.calendar(country,year,month, work\_days) VALUES ('MD', 2019, 8, {1,2,5,6,7,8,9,12,13,14,15,16,19,20,21,22,23,28,29}) IF NOT EXISTS;*

*INSERT INTO auctions.calendar(country,year,month, work\_days) VALUES ('MD', 2019, 9, {3,4,5,6,9,10,11,12,13,16,17,18,19,20,23,24,25,26,27,30}) IF NOT EXISTS;*

*INSERT INTO auctions.calendar(country,year,month, work\_days) VALUES ('MD', 2019, 10, {1,2,3,4,7,8,9,10,11,14,15,16,17,18,21,22,23,24,25,28,29,30,31}) IF NOT EXISTS;*

*INSERT INTO auctions.calendar(country,year,month, work\_days) VALUES ('MD', 2019, 11, {1,4,5,6,7,8,11,12,13,14,15,18,19,20,21,22,25,26,27,28,29}) IF NOT EXISTS;*

*INSERT INTO auctions.calendar(country,year,month, work\_days) VALUES ('MD', 2019, 12, {2,3,4,5,6,9,10,11,12,13,16,17,18,19,20,23,24,25,26,27,30}) IF NOT EXISTS;*

*INSERT INTO auctions.calendar (country,year,month, work\_days) VALUES ('MD',2020,1,{2,3,6,9,10,13,14,15,16,17,20,21,22,23,24,27,28,29,30,31});*

*INSERT INTO auctions.calendar (country,year,month, work\_days) VALUES ('MD',2020,2,{3,4,5,6,7,10,11,12,13,14,17,18,19,20,21,24,25,26,27,28});*

*INSERT INTO auctions.calendar (country,year,month, work\_days) VALUES ('MD',2020,3,{2,3,4,5,6,9,10,11,12,13,16,17,18,19,20,23,24,25,26,27,30,31});*

*INSERT INTO auctions.calendar (country,year,month, work\_days) VALUES ('MD',2020,4,{1,2,3,6,7,8,9,10,13,14,15,16,17,21,22,23,24,28,29,30});*

*INSERT INTO auctions.calendar (country,year,month, work\_days) VALUES ('MD',2020,5,{4,5,6,7,8,11,12,13,14,15,18,19,20,21,22,25,26,27,28,29});*

*INSERT INTO auctions.calendar (country,year,month, work\_days) VALUES ('MD',2020,6,{2,3,4,5,8,9,10,11,12,15,16,17,18,19,22,23,24,25,26,29,30});*

*INSERT INTO auctions.calendar (country,year,month, work\_days) VALUES ('MD',2020,7,{1,2,3,6,7,8,9,10,13,14,15,16,17,20,21,22,23,24,27,28,29,30,31});*

*INSERT INTO auctions.calendar (country,year,month, work\_days) VALUES ('MD',2020,8,{3,4,5,6,7,10,11,12,13,14,17,18,19,20,21,24,25,26,28});*

*INSERT INTO auctions.calendar (country,year,month, work\_days) VALUES ('MD',2020,9,{1,2,3,7,8,9,10,11,14,15,16,17,18,21,22,23,24,25,28,29,30});*

*INSERT INTO auctions.calendar (country,year,month, work\_days) VALUES ('MD',2020,10,{1,2,5,6,7,8,9,12,13,14,15,16,19,20,21,22,23,26,27,28,29,30});*

*INSERT INTO auctions.calendar (country,year,month, work\_days) VALUES ('MD',2020,11,{2,3,4,5,6,9,10,11,12,13,16,17,18,19,20,23,24,25,26,27,30});*

*INSERT INTO auctions.calendar (country,year,month, work\_days) VALUES ('MD',2020,12,{1,2,3,4,7,8,9,10,11,14,15,16,17,18,21,22,23,24,28,29,30,31});*

Log in GitLab as “writers” user.

Go to the repository and create a new branch.

Branch name: *e-auction*

Create form: *master*

Create “application.yaml” file with the following content:

*spring:*

*cloud:*

*inetutils:*

*preferredNetworks:*

*- 192.168*

*cassandra:*

*contact-points: ac-node1.ebrd.system,ac-node2.ebrd.system,ac-node3.ebrd.system*

*keyspace-name: ocds*

*port: 9042*

*auctions:*

*url:*

*protocol: http*

*host: auction.ebrd.system*

*qty-rounds: 3*

*qty-participants: 4*

*duration-one-step: 180s*

*duration-pause-after-step: 30s*

*duration-pause-after-auction: 30s*

*scheduler:*

*slots:*

*- start-time: "06:00Z"*

*- start-time: "07:00Z"*

*- start-time: "08:00Z"*

*- start-time: "09:00Z"*

*- start-time: "10:00Z"*

*- start-time: "11:00Z"*

*- start-time: "12:00Z"*

*end-time-all-slots: "13:00Z"*

*qty-lines-per-slot: 2*

Edit the file.

Add an empty line.

Commit the modification in order to have a new commit number.

Create new file “e-auction.sh” in the path “/root/deploys” of the Docker-swarm-node1, with the following content:

*#!/bin/bash*

*docker service create --name="e-auction" \*

*--env \_\_PROP\_spring\_cloud\_config\_uri=http://config-server:8080 \*

*--env \_\_PROP\_spring\_profiles\_active=default \*

*--env \_\_PROP\_spring\_cloud\_config\_label=[commit\_number] \*

*--env \_\_PROP\_cassandra\_username=caclient \*

*--env \_\_PROP\_cassandra\_password=rMgdGszsgPv79Ts7CB \*

*--publish published=9191,target=8080 \*

*--network ocds-network \*

*--dns* ***[ip\_of\_dns\_server]*** *\*

*eprocurementsystems/e-auction:1.1.1.0863dc6*

Grant permission to file:

*# chmod +x e-auction.sh*

And deploy the service:

*# ./ e- auction.sh*

Go to CentOs server.

*# sudo –s*

*# vi /etc/selinux/config*

Change paramet “SELINUX” to “disabled” and save it.

Reboot the server and install php:

# shutdown –r now

# yum install epel-release

# yum install httpd.x86\_64

# systemctl restart httpd

Check httpd service staus:

# systemctl status httpd

**“active (running)”** text should appear in green in the **console**.

Check the port is listening:

# netstat –nltp | grep 80

# yum install <http://rpms.remirepo.net/enterprise/remi-release-7.rpm>

# yum install yum-utils

# yum-config-manager --enable remi-php70

# yum install httpd

# yum install php php-fpm

Once installed the software described above, check the following modules are already installed in your server:

*bz2 calendar ctype*

*curl dom exif*

*fileinfo ftp gettext*

*iconv json mbstring*

*mysqlnd pdo phar*

*posix shmop simplexml*

*sockets sqlite3 sysvmsg*

*sysvsem sysvshm tokenizer*

*xml xmlwriter xsl*

*mysqli pdo\_mysql pdo\_sqlite*

*wddx xmlreader xmlrpc*

*rdkafka*

You can use the following command to detect which of these modules are already installed or not

# php –m | grep **name\_of\_module**

If some of the are not installed, use the following command to install it

# yum install php- **name\_of\_module**

Configure apache using the file **/etc/httpd/conf/httpd.conf**

Content:

*ServerRoot "/etc/httpd"*

*Listen 80*

*Include conf.modules.d/\*.conf*

*User apache*

*Group apache*

*ServerAdmin root@localhost*

*ServerName 116.203.211.144:80*

*<Directory />*

*AllowOverride none*

*Require all denied*

*</Directory>*

*DocumentRoot "/var/www/auction.ebrd.system/web"*

*<Directory "/var/www">*

*AllowOverride All*

*</Directory>*

*<Directory "/var/www/auction.ebrd.system/web">*

*Options Indexes FollowSymLinks*

*AllowOverride All*

*Require all granted*

*</Directory>*

*<Location /ws>*

*ProxyPass ws://127.0.0.1:8000*

*ProxyPassReverse ws://127.0.0.1:8000*

*ProxyPreserveHost On*

*SetEnv force-proxy-request-1.0 1*

*SetEnv proxy-nokeepalive 1*

*</Location>*

*<Directory "/var/www/auction.ebrd.system/web/">*

*Options All*

*Order allow,deny*

*Allow from all*

*</Directory>*

*<IfModule dir\_module>*

*DirectoryIndex index.html*

*</IfModule>*

*<Files ".ht\*">*

*Require all denied*

*</Files>*

*ErrorLog "logs/error\_log"*

*LogLevel warn*

*<IfModule log\_config\_module>*

*LogFormat "%a %l %u %t \"%r\" %>s %O \"%{Referer}i\" \"%{User-Agent}i\" \"UI-Tab:%{X-Tab}i\" \"UI-Sign:%{X-Sign}i\" \"ServerSignKey:%{X-Sign}o\"" combined*

*LogFormat "%h %l %u %t \"%r\" %>s %b" common*

*<IfModule logio\_module>*

*LogFormat "%h %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-Agent}i\" %I %O" combinedio*

*</IfModule>*

*CustomLog "logs/access\_log" combined*

*</IfModule>*

*<IfModule alias\_module>*

*ScriptAlias /cgi-bin/ "/var/www/cgi-bin/"*

*</IfModule>*

*<Directory "/var/www/cgi-bin">*

*AllowOverride None*

*Options None*

*Require all granted*

*</Directory>*

*<IfModule mime\_module>*

*TypesConfig /etc/mime.types*

*AddType application/x-compress .Z*

*AddType application/x-gzip .gz .tgz*

*AddType text/html .shtml*

*AddOutputFilter INCLUDES .shtml*

*</IfModule>*

*AddDefaultCharset UTF-8*

*<IfModule mime\_magic\_module>*

*MIMEMagicFile conf/magic*

*</IfModule>*

*EnableSendfile on*

*IncludeOptional conf.d/\*.conf*

Upload the database and code in the following files to the server:

* *auction.ebrd.system.tar.gz*
* *auction.sql.tar.gz*

Unzip auction.ebrd.syste.tar.gz files in “/var/www/” folder.

*# tar xf auction.ebrd.system.tar.gz*

*# mv auction.ebrd.system /var/www/*

*# chown –R apache:apache auction.ebrd.system*

*# systemctl restart httpd*

*# netstat -nltp | grep 80*

Using the IP of the server, check if the deployed web is working in the browser.

Install MySQL

*# wget* [*http://dev.mysql.com/get/mysql57-community-release-el7-7.noarch.rpm*](http://dev.mysql.com/get/mysql57-community-release-el7-7.noarch.rpm)

*# yum localinstall mysql57-community-release-el7-7.noarch.rpm*

*# yum repolist enabled | grep "mysql.\*-community.\*"*

*# yum install mysql-community-server*

*# service mysqld start*

Get the temporary password using the following instruction:

*# grep 'temporary password' /var/log/mysqld.log*

*# mysql\_secure\_installation*

**Console will ask the temporary password and requires a new password.**

**And will display some questions:**

* *Change root password 🡪 no*
* *Remove anonymus users 🡪 yes*
* *Disallow root login remotely 🡪 yes*
* *Remove test database and access 🡪 yes*
* *Reload privilages tables now 🡪 yes*

Access to mysql:

*# mysql –u root –p*

Insert password and we are inside mysql

*>create database auction;*

*>quit*

Import dump database to our new database.

*# mysql –uroot –p auction < /home/centos/***path\_of\_the\_uploaded\_auction.sql\_file**

Download phpmyadmin:

# cd /var/www/auction.ebrd.system

*# wget* [*https://files.phpmyadmin.net/phpMyAdmin/4.9.5/phpMyAdmin-4.9.5-all-languages.zip*](https://files.phpmyadmin.net/phpMyAdmin/4.9.5/phpMyAdmin-4.9.5-all-languages.zip)

Install unzip service if id does not exist yet

# yum install unzip

*# unzip* **php\_my\_admin\_file**

# *mv* **php\_my\_admin\_file** *phpmyadmin*

*# chown –R apache:apache phpmyadmin/*

*# yum install php-mysqli*

*# systemctl restart httpd*

Access to phpmyadmin application using the browser, using the IP of the server adding “/phpmyadmin/”.

Use the “root” user and defined password during the installationto access.

Delete all data from tables.

Execute the following command and the add it to “/etc/rc.local” file for autostart

*/usr/bin/php /var/www/auction.ebrd.system/websocket/websocket.php start -d*

Provide execution permissions to rc.local file

*# chmod +x /etc/rc.local*

# vi /etc/hostname

Content:

*eAuction*

Check the required kafka libraries are correctly installed

# rpm -qa | grep rdkaf

Confirm the following modules and versión are installed in your server, if not, install them:

* librdkafka-0.11.5-1.el7.x86\_64
* librdkafka-devel-0.11.5-1.el7.x86\_64
* php-pecl-rdkafka-3.1.3-1.el7.remi.7.0.x86\_64

Add kafka servers to *hosts* file:

# vi /etc/hosts

Content:

*Ip\_kafka\_node\_1 ak-node1.ebrd.system*

*Ip\_kafka\_node\_2 ak-node2.ebrd.system*

*Ip\_kafka\_node\_3 ak-node3.ebrd.system*

Restart the server.

*# shutdown –r now*

# Deploy e-Budget service

Go to Cassandra-node1

*# cqlsh -u caroot* ***cassandra\_node1\_ip***

*>use ocds;*

*> CREATE TABLE IF NOT EXISTS ocds.budget\_ei (*

*cp\_id text,*

*token\_entity uuid,*

*owner text,*

*created\_date timestamp,*

*json\_data text,*

*primary key(cp\_id, token\_entity)*

*);*

*CREATE TABLE IF NOT EXISTS ocds.budget\_fs (*

*cp\_id text,*

*oc\_id text,*

*token\_entity uuid,*

*owner text,*

*amount decimal,*

*amount\_reserved decimal,*

*created\_date timestamp,*

*json\_data text,*

*primary key(cp\_id, token\_entity)*

*);*

*CREATE TABLE IF NOT EXISTS ocds.budget\_rules (*

*country text,*

*parameter text,*

*value text,*

*primary key(country, parameter)*

*);*

*CREATE TABLE IF NOT EXISTS ocds.budget\_history (*

*operation\_id text,*

*command text,*

*operation\_date timestamp,*

*json\_data text,*

*primary key(operation\_id, command)*

*);*

*INSERT INTO ocds.budget\_rules (country, parameter, value) VALUES ('MD', 'cpv', '^[0-9]{2}[1-9][0]{5}-[0-9]$');*

*INSERT INTO ocds.budget\_rules (country, parameter, value) VALUES ('TEST', 'cpv', '^[0-9]{2}[1-9][0]{5}-[0-9]$');*

Log in GitLab as “writers” user.

Go to the repository and create a new branch:

Branch name: *e-budget*

Create form: *master*

Create “application.yaml” file with the following content:

*spring:*

*profiles: default*

*cloud:*

*inetutils:*

*preferredNetworks:*

*- 192.168*

*cassandra:*

*contact-points: ac-node1.ebrd.system,ac-node2.ebrd.system,ac-node3.ebrd.system*

*port: 9042*

*# ocds.prefix: ocds-b3wdp1*

*ocds:*

*prefixes:*

*main: ocds-b3wdp1*

*test: test-b3wdp1*

*logging:*

*pattern:*

*console: "%d{yyyy-MM-dd HH:mm:ss} %-5level %logger{36} - %msg%n"*

Edit the file.

Add an empty line.

Commit the modification in order to have a new commit number.

Create new file “e-budget.sh” in the path “/root/deploys” of the Docker-swarm-node1, with the following content:

*#!/bin/bash*

*docker service create --name="e-budget" \*

*--env \_\_PROP\_spring\_cloud\_config\_uri=http://config-server:8080 \*

*--env \_\_PROP\_spring\_profiles\_active=default \*

*--env \_\_PROP\_spring\_cloud\_config\_label=[commit\_number \*

*--env \_\_PROP\_cassandra\_username=caclient \*

*--env \_\_PROP\_cassandra\_password=rMgdGszsgPv79Ts7CB \*

*--publish published=9051,target=8080 \*

*--network ocds-network \*

*--dns* ***[ip\_of\_dns\_server]*** *\*

*eprocurementsystems/e-budget:1.1.1.474c6e6*

Grant permission to file:

*# chmod +x e- budget.sh*

And deploy the service:

*# ./ e- budget.sh*

# Deploy e-Clarification service

Go to Cassandra-node1

*# cqlsh -u caroot* ***cassandra\_node1\_ip***

*>use ocds;*

*> CREATE TABLE IF NOT EXISTS ocds.clarification\_rules (*

*country text,*

*pmd text,*

*parameter text,*

*value text,*

*primary key(country, pmd, parameter)*

*);*

*CREATE TABLE IF NOT EXISTS ocds.clarification\_period (*

*cp\_id text,*

*stage text,*

*owner text,*

*start\_date timestamp,*

*end\_date timestamp,*

*tender\_end\_date timestamp,*

*primary key(cp\_id, stage)*

*);*

*CREATE TABLE IF NOT EXISTS ocds.clarification\_enquiry (*

*cp\_id text,*

*stage text,*

*token\_entity uuid,*

*json\_data text,*

*is\_answered boolean,*

*primary key(cp\_id, stage, token\_entity)*

*);*

*CREATE TABLE IF NOT EXISTS ocds.clarification\_history (*

*operation\_id text,*

*command text,*

*operation\_date timestamp,*

*json\_data text,*

*primary key(operation\_id, command)*

*);*

*INSERT INTO ocds.clarification\_rules (country, pmd, parameter, value) VALUES ('MD', 'OT', 'interval', '777600');*

*INSERT INTO ocds.clarification\_rules (country, pmd, parameter, value) VALUES ('MD', 'OT', 'offset', '777600');*

*INSERT INTO ocds.clarification\_rules (country, pmd, parameter, value) VALUES ('MD', 'OT', 'interval\_before', '172800');*

*INSERT INTO ocds.clarification\_rules (country, pmd, parameter, value) VALUES ('MD', 'OT', 'offsetExtended', '172800');*

*INSERT INTO ocds.clarification\_rules (country, pmd, parameter, value) VALUES ('MD', 'SV', 'interval', '345600');*

*INSERT INTO ocds.clarification\_rules (country, pmd, parameter, value) VALUES ('MD', 'SV', 'offset', '345600');*

*INSERT INTO ocds.clarification\_rules (country, pmd, parameter, value) VALUES ('MD', 'SV', 'interval\_before', '172800');*

*INSERT INTO ocds.clarification\_rules (country, pmd, parameter, value) VALUES ('MD', 'SV', 'offsetExtended', '172800');*

*INSERT INTO ocds.clarification\_rules (country, pmd, parameter, value) VALUES ('MD', 'TEST\_OT', 'interval', '90');*

*INSERT INTO ocds.clarification\_rules (country, pmd, parameter, value) VALUES ('MD', 'TEST\_OT', 'offset', '90');*

*INSERT INTO ocds.clarification\_rules (country, pmd, parameter, value) VALUES ('MD', 'TEST\_OT', 'interval\_before', '20');*

*INSERT INTO ocds.clarification\_rules (country, pmd, parameter, value) VALUES ('MD', 'TEST\_OT', 'offsetExtended', '20');*

*INSERT INTO ocds.clarification\_rules (country, pmd, parameter, value) VALUES ('MD', 'TEST\_SV', 'interval', '40');*

*INSERT INTO ocds.clarification\_rules (country, pmd, parameter, value) VALUES ('MD', 'TEST\_SV', 'offset', '40');*

*INSERT INTO ocds.clarification\_rules (country, pmd, parameter, value) VALUES ('MD', 'TEST\_SV', 'interval\_before', '20');*

*INSERT INTO ocds.clarification\_rules (country, pmd, parameter, value) VALUES ('MD', 'TEST\_SV', 'offsetExtended', '20');*

*INSERT INTO ocds.clarification\_rules (country,pmd,parameter,value) VALUES ('MD','MV','interval','86400');*

*INSERT INTO ocds.clarification\_rules (country,pmd,parameter,value) VALUES ('MD','MV','interval\_before','86400');*

*INSERT INTO ocds.clarification\_rules (country,pmd,parameter,value) VALUES ('MD','MV','offset','86400');*

*INSERT INTO ocds.clarification\_rules (country,pmd,parameter,value) VALUES ('MD','MV','offsetExtended','86400');*

*INSERT INTO ocds.clarification\_rules (country,pmd,parameter,value) VALUES ('MD','TEST\_MV','interval','10');*

*INSERT INTO ocds.clarification\_rules (country,pmd,parameter,value) VALUES ('MD','TEST\_MV','interval\_before','10');*

*INSERT INTO ocds.clarification\_rules (country,pmd,parameter,value) VALUES ('MD','TEST\_MV','offset','10');*

*INSERT INTO ocds.clarification\_rules (country,pmd,parameter,value) VALUES ('MD','TEST\_MV','offsetExtended','10');*

Log in GitLab as “writers” user.

Go to the repository and create a new branch:

Branch name: *e-clarification*

Create form: *master*

Create “application.yaml” file with the following content:

*spring:*

*profiles: default*

*cloud:*

*inetutils:*

*preferredNetworks:*

*- 192.168*

*cassandra:*

*contact-points: ac-node1.ebrd.system,ac-node2.ebrd.system,ac-node3.ebrd.system*

*keyspace-name: ocds*

*port: 9042*

*logging:*

*pattern:*

*console: "%d{yyyy-MM-dd HH:mm:ss} %-5level %logger{36} - %msg%n"*

Edit the file.

Add an empty line.

Commit the modification in order to have a new commit number.

Create new file “e-clarification.sh” in the path “/root/deploys” of the Docker-swarm-node1, with the following content:

*docker service create --name="e-clarification" \*

*--env \_\_PROP\_spring\_cloud\_config\_uri=http://config-server:8080 \*

*--env \_\_PROP\_spring\_profiles\_active=default \*

*--env \_\_PROP\_spring\_cloud\_config\_label=[commit\_number] \*

*--env \_\_PROP\_cassandra\_username=caclient \*

*--env \_\_PROP\_cassandra\_password=rMgdGszsgPv79Ts7CB \*

*--publish published=9071,target=8080 \*

*--network ocds-network \*

*--dns* ***[ip\_of\_dns\_server]*** *\*

*eprocurementsystems/e-clarification:1.1.1.6fa635c*

Grant permission to file:

*# chmod +x e- clarification.sh*

And deploy the service:

*# ./ e- clarification.sh*

# Deploy e-Contracting service

Go to Cassandra-node1 server

*# cqlsh -u caroot* ***cassandra\_node1\_ip***

>use ocds;

In order to create the structure and insert the data, go to annex3.docx.

Log in GitLab as “writers” user.

Go to the repository and create a new branch:

Branch name*: e-contracting*

Create form: *master*

Create “application.yaml” file with the following content:

*spring:*

*profiles: default*

*cloud:*

*inetutils:*

*preferredNetworks:*

*- 192.168*

*cassandra:*

*contact-points: ac-node1.ebrd.system,ac-node2.ebrd.system,ac-node3.ebrd.system*

*keyspace-name: ocds*

*port: 9042*

*logging:*

*pattern:*

*console: "%d{yyyy-MM-dd HH:mm:ss} %-5level %logger{36} - %msg%n"*

Edit the file.

Add an empty line.

Commit the modification in order to have a new commit number.

Create new file “e- contracting.sh” in the path “/root/deploys” of the Docker-swarm-node1, with the following content:

*docker service create --name="e-contracting" \*

*--env \_\_PROP\_spring\_cloud\_config\_uri=http://config-server:8080 \*

*--env \_\_PROP\_spring\_profiles\_active=default \*

*--env \_\_PROP\_spring\_cloud\_config\_label=[commit\_number] \*

*--env \_\_PROP\_cassandra\_username=caclient \*

*--env \_\_PROP\_cassandra\_password=rMgdGszsgPv79Ts7CB \*

*--publish published=9151,target=8080 \*

*--network ocds-network \*

*--dns* ***[ip\_of\_dns\_server]*** *\*

*eprocurementsystems/e-contracting:1.1.1.90e91c5*

Grant permission to file:

*# chmod +x e- contracting.sh*

And deploy the service:

*# ./ e- contracting.sh*

# Deploy e-Evaluation service

Go to Cassandra-node1

*# cqlsh -u caroot* ***cassandra\_node1\_ip***

>use ocds;

*> CREATE TABLE IF NOT EXISTS ocds.evaluation\_rules (*

*country text,*

*pmd text,*

*parameter text,*

*value text,*

*primary key(country, pmd, parameter)*

*);*

*CREATE TABLE IF NOT EXISTS ocds.evaluation\_period (*

*cp\_id text,*

*stage text,*

*award\_criteria text,*

*start\_date timestamp,*

*end\_date timestamp,*

*primary key(cp\_id, stage)*

*);*

*CREATE TABLE IF NOT EXISTS ocds.evaluation\_award (*

*cp\_id text,*

*stage text,*

*token\_entity uuid,*

*status text,*

*status\_details text,*

*owner text,*

*json\_data text,*

*primary key(cp\_id, stage, token\_entity)*

*);*

*CREATE TABLE IF NOT EXISTS ocds.evaluation\_history (*

*operation\_id text,*

*command text,*

*operation\_date timestamp,*

*json\_data text,*

*primary key(operation\_id, command)*

*);*

*INSERT INTO ocds.evaluation\_rules (country, pmd, parameter, value) VALUES ('MD', 'OT', 'minBids', '1');*

*INSERT INTO ocds.evaluation\_rules (country, pmd, parameter, value) VALUES ('MD', 'SV', 'minBids', '1');*

*INSERT INTO ocds.evaluation\_rules (country, pmd, parameter, value) VALUES ('MD', 'TEST\_OT', 'minBids', '1');*

*INSERT INTO ocds.evaluation\_rules (country, pmd, parameter, value) VALUES ('MD', 'TEST\_SV', 'minBids', '1');*

*INSERT INTO ocds.evaluation\_rules (country,pmd,parameter,value) VALUES ('MD','MV','minBids','1');*

*INSERT INTO ocds.evaluation\_rules (country,pmd,parameter,value) VALUES ('MD','TEST\_MV','minBids','1');*

*EST\_MV','works','false');*

Log in GitLab as “writers” user.

Go to the repository and create a new branch:

Branch name: *e-evaluation*

Create form: *master*

Create “application.yaml” file with the following content:

*spring:*

*profiles: default*

*cloud:*

*inetutils:*

*preferredNetworks:*

*- 192.168*

*cassandra:*

*contact-points: ac-node1.ebrd.system,ac-node2.ebrd.system,ac-node3.ebrd.system*

*keyspace-name: ocds*

*port: 9042*

*ocds.prefix: ocds-b3wdp1*

*logging:*

*pattern:*

*console: "%d{yyyy-MM-dd HH:mm:ss} %-5level %logger{36} - %msg%n"*

Edit the file.

Add an empty line.

Commit the modification in order to have a new commit number.

Create new file “e-evaluation.sh” in the path “/root/deploys” of the Docker-swarm-node1, with the following content:

*#!/bin/bash*

*docker service create --name="e-evaluation" \*

*--env \_\_PROP\_spring\_cloud\_config\_uri=http://config-server:8080 \*

*--env \_\_PROP\_spring\_profiles\_active=default \*

*--env \_\_PROP\_spring\_cloud\_config\_label=[commit\_number] \*

*--env \_\_PROP\_cassandra\_username=caclient \*

*--env \_\_PROP\_cassandra\_password=rMgdGszsgPv79Ts7CB \*

*--publish published=9081,target=8080 \*

*--network ocds-network \*

*--dns* ***[ip\_of\_dns\_server]*** *\*

*eprocurementsystems/e-evaluation:1.1.1.add850c*

Grant permission to file:

*# chmod +x e- evaluation.sh*

And deploy the service:

*# ./ e- evaluation.sh*

# Deploy e-Notice service

Go to Cassandra-node1

*# ./ e- evaluation.sh*

*# cqlsh -u caroot* ***cassandra\_node1\_ip***

*>use ocds;*

*> CREATE TABLE IF NOT EXISTS ocds.notice\_release (*

*cp\_id text,*

*oc\_id text,*

*release\_date timestamp,*

*release\_id text,*

*stage text,*

*json\_data text,*

*primary key(cp\_id, oc\_id, release\_id)*

*)WITH CLUSTERING ORDER BY (oc\_id ASC);*

*CREATE TABLE IF NOT EXISTS ocds.notice\_compiled\_release (*

*cp\_id text,*

*oc\_id text,*

*publish\_date timestamp,*

*release\_date timestamp,*

*release\_id text,*

*stage text,*

*status text,*

*json\_data text,*

*primary key(cp\_id, oc\_id)*

*)WITH CLUSTERING ORDER BY (oc\_id ASC);*

*CREATE TABLE IF NOT EXISTS ocds.notice\_offset (*

*cp\_id text,*

*release\_date timestamp,*

*status text,*

*stage text,*

*primary key(cp\_id)*

*);*

*CREATE TABLE IF NOT EXISTS ocds.notice\_budget\_release (*

*cp\_id text,*

*oc\_id text,*

*release\_date timestamp,*

*release\_id text,*

*stage text,*

*json\_data text,*

*primary key(cp\_id, oc\_id, release\_id)*

*)WITH CLUSTERING ORDER BY (oc\_id ASC);*

*CREATE TABLE IF NOT EXISTS ocds.notice\_budget\_compiled\_release (*

*cp\_id text,*

*oc\_id text,*

*publish\_date timestamp,*

*release\_date timestamp,*

*release\_id text,*

*stage text,*

*amount decimal,*

*json\_data text,*

*primary key(cp\_id, oc\_id)*

*)WITH CLUSTERING ORDER BY (oc\_id ASC);*

*CREATE TABLE IF NOT EXISTS ocds.notice\_budget\_offset (*

*cp\_id text,*

*release\_date timestamp,*

*primary key(cp\_id)*

*);*

*CREATE TABLE IF NOT EXISTS ocds.notice\_history (*

*operation\_id text,*

*command text,*

*operation\_date timestamp,*

*json\_data text,*

*primary key(operation\_id, command)*

*);*

Log in GitLab as “writers” user.

Go to the repository and create a new branch:

Branch name: *e-notice*

Create form: *master*

Create “application.yaml” file with the following content:

*spring:*

*profiles: default*

*cloud:*

*inetutils:*

*preferredNetworks:*

*- 192.168*

*cassandra:*

*contact-points: ac-node1.ebrd.system,ac-node2.ebrd.system,ac-node3.ebrd.system*

*keyspace-name: ocds*

*port: 9042*

*uri:*

*budget: http://public.ebrd.system/budgets/*

*tender: http://public.ebrd.system/tenders/*

*logging:*

*pattern:*

*console: "%d{yyyy-MM-dd HH:mm:ss} %-5level %logger{36} - %msg%n"*

Edit the file.

Add an empty line.

Commit the modification in order to have a new commit number.

Create new file “e-notice.sh” in the path “/root/deploys” of the Docker-swarm-node1, with the following content:

*#!/bin/bash*

*docker service create --name="e-notice" \*

*--env \_\_PROP\_spring\_cloud\_config\_uri=http://config-server:8080 \*

*--env \_\_PROP\_spring\_profiles\_active=default \*

*--env \_\_PROP\_spring\_cloud\_config\_label=ba65353d \*

*--env \_\_PROP\_cassandra\_username=caclient \*

*--env \_\_PROP\_cassandra\_password=rMgdGszsgPv79Ts7CB \*

*--publish published=9041,target=8080 \*

*--network ocds-network \*

*--dns* ***[ip\_of\_dns\_server]*** *\*

*eprocurementsystems/e-notice:1.1.1.87f6e9b*

Grant permission to file:

*# chmod +x e- notice.sh*

And deploy the service:

*# ./ e- notice.sh*

# Deploy e-Regulation service

Go to Cassandra-node1

# cqlsh -u caroot **cassandra\_node\_1\_ip**

>use ocds;

To create the structure and insert the data, go to annex4.docx

Log in GitLab as “writers” user.

Go to the repository and create a new branch:

Branch name: *e-regulation*

Create form: *master*

Create “application.yaml” file with the following content:

*spring:*

*profiles: default*

*cassandra:*

*contact-points: ac-node1.ebrd.system,ac-node2.ebrd.system,ac-node3.ebrd.system*

*keyspace-name: ocds*

*port: 9042*

Edit the file.

Add an empty line.

Commit the modification in order to have a new commit number.

Create new file “e-regulation.sh” in the path “/root/deploys” of the Docker-swarm-node1, with the following content:

*#!/bin/bash*

*docker service create --name="e-regulation" \*

*--env \_\_PROP\_spring\_cloud\_config\_uri=http://config-server:8080 \*

*--env \_\_PROP\_spring\_profiles\_active=default \*

*--env \_\_PROP\_spring\_cloud\_config\_label=[commit\_number] \*

*--env \_\_PROP\_cassandra\_username=caclient \*

*--env \_\_PROP\_cassandra\_password=rMgdGszsgPv79Ts7CB \*

*--publish published=9201,target=8080 \*

*--network ocds-network \*

*--dns* ***[ip\_of\_dns\_server]*** *\*

*eprocurementsystems/e-regulation:1.1.1.0eadbf3*

Grant permission to file:

*# chmod +x e- regulation.sh*

And deploy the service:

*# ./ e- regulation.sh*

# Deploy e-Submission service

Go to Cassandra-node1

*# cqlsh -u caroot* ***cassandra\_node1\_ip***

*>use ocds;*

*> CREATE TABLE IF NOT EXISTS ocds.submission\_rules (*

*country text,*

*pmd text,*

*parameter text,*

*value text,*

*primary key(country, pmd, parameter)*

*);*

*CREATE TABLE IF NOT EXISTS ocds.submission\_bid (*

*cp\_id text,*

*stage text,*

*owner text,*

*bid\_id uuid,*

*token\_entity uuid,*

*status text,*

*created\_date timestamp,*

*pending\_date timestamp,*

*json\_data text,*

*primary key(cp\_id, stage, bid\_id)*

*);*

*CREATE TABLE IF NOT EXISTS ocds.submission\_period (*

*cp\_id text,*

*stage text,*

*start\_date timestamp,*

*end\_date timestamp,*

*primary key(cp\_id, stage)*

*);*

*CREATE TABLE IF NOT EXISTS ocds.submission\_history (*

*operation\_id text,*

*command text,*

*operation\_date timestamp,*

*json\_data text,*

*primary key(operation\_id, command)*

*);*

*INSERT INTO ocds.submission\_rules (country, pmd, parameter, value) VALUES ('MD', 'OT', 'minBids', '1');*

*INSERT INTO ocds.submission\_rules (country, pmd, parameter, value) VALUES ('MD', 'OT', 'interval', '259200');*

*INSERT INTO ocds.submission\_rules (country, pmd, parameter, value) VALUES ('MD', 'OT', 'interval\_before', '172800');*

*INSERT INTO ocds.submission\_rules (country, pmd, parameter, value) VALUES ('MD', 'OT', 'unsuspend\_interval', '259200');*

*INSERT INTO ocds.submission\_rules (country, pmd, parameter, value) VALUES ('MD', 'TEST\_OT', 'minBids', '1');*

*INSERT INTO ocds.submission\_rules (country, pmd, parameter, value) VALUES ('MD', 'TEST\_OT', 'interval', '30');*

*INSERT INTO ocds.submission\_rules (country, pmd, parameter, value) VALUES ('MD', 'TEST\_OT', 'interval\_before', '20');*

*INSERT INTO ocds.submission\_rules (country, pmd, parameter, value) VALUES ('MD', 'TEST\_OT', 'unsuspend\_interval', '30');*

*INSERT INTO ocds.submission\_rules (country, pmd, parameter, value) VALUES ('MD', 'SV', 'minBids', '1');*

*INSERT INTO ocds.submission\_rules (country, pmd, parameter, value) VALUES ('MD', 'SV', 'interval', '259200');*

*INSERT INTO ocds.submission\_rules (country, pmd, parameter, value) VALUES ('MD', 'SV', 'interval\_before', '172800');*

*INSERT INTO ocds.submission\_rules (country, pmd, parameter, value) VALUES ('MD', 'SV', 'unsuspend\_interval', '259200');*

*INSERT INTO ocds.submission\_rules (country, pmd, parameter, value) VALUES ('MD', 'TEST\_SV', 'minBids', '1');*

*INSERT INTO ocds.submission\_rules (country, pmd, parameter, value) VALUES ('MD', 'TEST\_SV', 'interval', '30');*

*INSERT INTO ocds.submission\_rules (country, pmd, parameter, value) VALUES ('MD', 'TEST\_SV', 'interval\_before', '20');*

*INSERT INTO ocds.submission\_rules (country, pmd, parameter, value) VALUES ('MD', 'TEST\_SV', 'unsuspend\_interval', '30');*

*INSERT INTO ocds.submission\_rules (country,pmd,parameter,value) VALUES ('MD','MV','goods','false');*

*INSERT INTO ocds.submission\_rules (country,pmd,parameter,value) VALUES ('MD','MV','interval','172800');*

*INSERT INTO ocds.submission\_rules (country,pmd,parameter,value) VALUES ('MD','MV','interval\_before','86400');*

*INSERT INTO ocds.submission\_rules (country,pmd,parameter,value) VALUES ('MD','MV','minBids','1');*

*INSERT INTO ocds.submission\_rules (country,pmd,parameter,value) VALUES ('MD','MV','services','false');*

*INSERT INTO ocds.submission\_rules (country,pmd,parameter,value) VALUES ('MD','MV','unsuspend\_interval','172800');*

*INSERT INTO ocds.submission\_rules (country,pmd,parameter,value) VALUES ('MD','MV','works','false');*

*INSERT INTO ocds.submission\_rules (country,pmd,parameter,value) VALUES ('MD','TEST\_MV','goods','false');*

*INSERT INTO ocds.submission\_rules (country,pmd,parameter,value) VALUES ('MD','TEST\_MV','interval\_before','10');*

*INSERT INTO ocds.submission\_rules (country,pmd,parameter,value) VALUES ('MD','TEST\_MV','interval','20');*

*INSERT INTO ocds.submission\_rules (country,pmd,parameter,value) VALUES ('MD','TEST\_MV','minBids','1');*

*INSERT INTO ocds.submission\_rules (country,pmd,parameter,value) VALUES ('MD','TEST\_MV','services','false');*

*INSERT INTO ocds.submission\_rules (country,pmd,parameter,value) VALUES ('MD','TEST\_MV','unsuspend\_interval','20');*

*INSERT INTO ocds.submission\_rules (country,pmd,parameter,value) VALUES ('MD','TEST\_MV','works','false');*

Log in GitLab as “writers” user.

Go to the repository.

Create a new branch:

Branch name: *e-submission*

Create form: *master*

Create “application.yaml” file with the following content:

*spring:*

*profiles: default*

*cloud:*

*inetutils:*

*preferredNetworks:*

*- 192.168*

*cassandra:*

*contact-points: ac-node1.ebrd.system,ac-node2.ebrd.system,ac-node3.ebrd.system*

*keyspace-name: ocds*

*port: 9042*

*ocds.prefix: ocds-b3wdp1*

*logging:*

*pattern:*

*console: "%d{yyyy-MM-dd HH:mm:ss} %-5level %logger{36} - %msg%n"*

Edit the file.

Add an empty line.

Commit the modification in order to have a new commit number.

Create new file “e-submission.sh” in the path “/root/deploys” of the Docker-swarm-node1, with the following content:

*#!/bin/bash*

*docker service create --name="e-submission" \*

*--env \_\_PROP\_spring\_cloud\_config\_uri=http://config-server:8080 \*

*--env \_\_PROP\_spring\_profiles\_active=default \*

*--env \_\_PROP\_spring\_cloud\_config\_label=17d3842a \*

*--env \_\_PROP\_cassandra\_username=caclient \*

*--env \_\_PROP\_cassandra\_password=rMgdGszsgPv79Ts7CB \*

*--publish published=9061,target=8080 \*

*--network ocds-network \*

*--dns* ***[ip\_of\_dns\_server]*** *\*

*eprocurementsystems/e-submission:1.1.1.bb9cec9*

Grant permission to file:

*# chmod +x e- submission.sh*

And deploy the service:

*# ./ e- submission.sh*

# Deploy Document Generator service

Go to Cassandra-node1

*# cqlsh -u caroot* ***cassandra\_node1\_ip***

*>use ocds;*

*> CREATE KEYSPACE documents*

*WITH replication = {*

*'class' : 'SimpleStrategy',*

*'replication\_factor' : 3*

*};*

*CREATE TABLE IF NOT EXISTS documents.templates(*

*document\_id text,*

*document\_kind text,*

*lang text,*

*start\_date date,*

*format text,*

*engine text,*

*body blob,*

*PRIMARY KEY ((document\_id, document\_kind), lang, start\_date)*

*);*

*CREATE TABLE IF NOT EXISTS documents.descriptors(*

*command\_id text,*

*document\_id text,*

*document\_kind text,*

*lang text,*

*descriptor text,*

*PRIMARY KEY (command\_id)*

*);*

Log in GitLab as “writers” user.

Go to the repository and create a new branch:

Branch name: *document-generator*

Create form: *master*

Create “application.yaml” file with the following content:

*spring:*

*cloud:*

*inetutils:*

*preferredNetworks:*

*- 192.168*

*kafka:*

*bootstrap-servers: ak-node1.ebrd.system:9092,ak-node2.ebrd.system:9092,ak-node3.ebrd.system:9092*

*consumer:*

*group-id: document-generator-consumer-group*

*enable-auto-commit: false*

*producer:*

*retries: 5*

*cassandra:*

*contact-points: ac-node1.ebrd.system,ac-node2.ebrd.system,ac-node3.ebrd.system*

*keyspace-name: documents*

*port: 9042*

*external-endpoints:*

*public-point: http://public-point:8080*

*storage:*

*registration: http://storage:8080*

*upload: http://storage-service.ebrd.system:8080*

Edit the file.

Add an empty line.

Commit the modification in order to have a new commit number.

Create new file “e-document-generator.sh” in the path “/root/deploys” of the docker-swarm-node1 VM, with the following content:

*#!/bin/bash*

*docker service create --name="document-generator" \*

*--env \_\_PROP\_spring\_cloud\_config\_uri=http://config-server:8080 \*

*--env \_\_PROP\_spring\_profiles\_active=default \*

*--env \_\_PROP\_spring\_cloud\_config\_label=[commit\_number] \*

*--env \_\_PROP\_cassandra\_username=[caclient\_database\_client] \*

*--env \_\_PROP\_cassandra\_password=[caclient\_password] \*

*--publish published=9210,target=8080 \*

*--network ocds-network \*

*--dns* ***[ip\_of\_dns\_server]*** *\*

*eprocurementsystems/document-generator:1.1.1.78c6b14*

Grant permission to file:

*# chmod +x e- document-generator.sh*

And deploy the service:

*# ./ document-generator.sh*

# Deploy Gateway Public service

Log in GitLab as “writers” user.

Go to the repository and create a new branch:

Branch name: *gateway-public*

Create form: *master*

Create “application.yaml” file with the following content:

*spring:*

*cloud:*

*inetutils:*

*preferredNetworks:*

*- 192.168*

*servlet:*

*multipart:*

*enabled: true*

*max-file-size: 50Mb*

*max-request-size: 50Mb*

*zuul:*

*sensitiveHeaders:*

*ignoredServices: '\*'*

*routes:*

*public:*

*path: /\*\**

*url: http://public-point:8080*

*stripPrefix: true*

*mdm:*

*path: /mdm/\*\**

*url: http://mdm:8080*

*stripPrefix: true*

*eureka:*

*client:*

*enabled: false*

*ribbon:*

*eureka:*

*enabled: false*

Edit the file.

Add an empty line.

Commit the modification in order to have a new commit number.

Create new file “gateway-public.sh” in the path “/root/deploys” of the docker-swarm-node1 VM, with the following content:

*#!/bin/bash*

*docker service create --name="gateway-public" \*

*--env \_\_PROP\_spring\_cloud\_config\_uri=http://config-server:8080 \*

*--env \_\_PROP\_spring\_profiles\_active=default \*

*--env \_\_PROP\_spring\_cloud\_config\_label=[commit\_number] \*

*--publish published=8910,target=8080 \*

*--network ocds-network \*

*--dns* ***[ip\_of\_dns\_server]*** *\*

*eprocurementsystems/gateway-public:1.1.1.4a41c87*

Grant permission to file:

*# chmod +x gateway-public.sh*

And deploy the service:

*# ./ gateway-public.sh*

# Deploy gateway-mconnect service

Log in GitLab as “writers” user.

Go to the repository and create a new branch:

Branch name: *gateway-mconnect*

Create form: *master*

Create “application.yaml” file with the following content:

*spring:*

*cloud:*

*inetutils:*

*preferredNetworks:*

*- 192.168*

*zuul:*

*sensitiveHeaders:*

*# sensitive-headers: Cookie,Set-Cookie,Authorization*

*# ignore-security-headers: false*

*prefix: /api/v1*

*ignoredServices: '\*'*

*routes:*

*mconnect-budget:*

*path: /mconnect/budget/\*\**

*url: http://****[ip\_of\_dns\_server]****/api/v1/budget*

*stripPrefix: true*

*eureka:*

*client:*

*enabled: false*

*ribbon:*

*# eureka:*

*# enabled: false*

*MaxAutoRetries: 0*

*MaxAutoRetriesNextServer: 0*

*ReadTimeout: 10000*

*ConnectTimeout: 10000*

*#feign:*

*# hystrix:*

*# enabled: false*

*hystrix:*

*command:*

*default:*

*execution:*

*isolation:*

*thread:*

*timeoutInMilliseconds: 20000*

*timeout:*

*enabled: false*

*circuitBreaker:*

*enabled: false*

*fallback:*

*enabled: false*

Edit the file.

Add an empty line.

Commit the modification in order to have a new commit number.

Create new file “gateway-mconnect.sh” in the path “/root/deploys” of the docker-swarm-node1 VM, with the following content:

*#!/usr/bin/env bash*

*docker service create --name="gateway-mconnect" \*

*--env \_\_PROP\_spring\_application\_name=gateway-mconnect \*

*--env \_\_PROP\_spring\_cloud\_config\_name=gateway-mconnect \*

*--env \_\_PROP\_spring\_cloud\_config\_uri=http://config-server:8080 \*

*--env \_\_PROP\_spring\_profiles\_active=default \*

*--env \_\_PROP\_spring\_cloud\_config\_label=[commit\_number] \*

*--publish published=8920,target=8080 \*

*--network ocds-network \*

*--dns* ***[ip\_of\_dns\_server]***

*eprocurementsystems/gateway:1.1.1.f8f61e9*

Grant permission to file:

*# chmod +x gateway-mconnect.sh*

And deploy the service:

*# ./gateway-mconnect.sh*