Kafka installation

Content

Change History

Version	Date	Comment
Current Version (v. 5)	**Dec 16, 2019 15:58**	**Vladimir Remenar**: Change uat to dev in create topics.
v. 4	Dec 16, 2019 15:57	**Vladimir Remenar**: Start Kafka, Zookeeper. Create all topics.
v. 3	Dec 16, 2019 15:45	**Vladimir Remenar**: Certificate, keystore and truststore creation
v. 2	Dec 16, 2019 15:16	**Vladimir Remenar**: User, group creation. Folder creation.
v. 1	Dec 16, 2019 14:59	**Vladimir Remenar**

Introduction

This document describes steps to install and configure Kafka instance. Operating system is CentOS8/RHEL8.

Set networking

```
$ vi /etc/sysconfig/network-scripts/ifcfg-eth0
```

IPv6 should be disabled, IP address should be sat as static and enabled on boot

```
TYPE=Ethernet
PROXY_METHOD=none
BROWSER_ONLY=no
BOOTPROTO=static
DEFROUTE=yes
IPV4_FAILURE_FATAL=no
IPV6INIT=no
IPV6_AUTOCONF=yes
IPV6_DEFROUTE=yes
IPV6_FAILURE_FATAL=no
IPV6_ADDR_GEN_MODE=stable-privacy
NAME=eth0
UUID=b10fd893-ec89-474a-9183-92e4066de240
DEVICE=eth0
ONBOOT=yes
IPADDR=10.10.10.81
PREFIX=24
GATEWAY=10.10.10.254
DNS1=10.10.10.254
DOMAIN=sberbank.hr
NM_CONTROLLED=yes
```

Set host name

Example of configuration of eth interface. Configuration has to be changed according to the environment.

Although DNS will be used it is recommended to set host name on the server itself. Replace "server.sberbank.hr" with actual server FQDN.

```
$ hostnamectl set-hostname server.sberbank.hr
```

Edit network file and add HOSTNAME line

```
$ vi /etc/sysconfig/network
```

Example of HOSTNAME line. Replace "server.sberbank.hr" with actual server FQDN

```
HOSTNAME=server.sberbank.hr
```

Edit hosts file and add IP, hostname, FQDN line

```
$ vi /etc/hosts
```

Example of IP, hostname, FQDN line. Replace with actual values.

```
10.10.10.81 lab-kafka01.lab.local lab-kafka01
```

Update and install necessary tools

Add Docker repo

```
$ dnf config-manager --add-repo https://download.docker.com/linux/centos/docker-ce.repo
```

Update operating system and packages

```
$ dnf -y update
```

Install necessary packages

```
$ dnf -y install nano net-tools wget git bind-utils bash-completion device-mapper-persistent-data lvm2 java-11-openjdk-devel
```

Install Docker

```
$ dnf -y install docker-ce docker-ce-cli containerd.io --nobest
```

Enable and start Docker service

Enable Docker service

```
$ systemctl enable docker
```

Start Docker service

```
$ systemctl start docker
```

Check status of Docker service

```
$ systemctl status docker
```

Configure firewall

Kafka and Kafka JMX

Kafka requires only one port to be exposed, Broker on TCP/9092. In order to access JMX endpoints another port has to be exposed, TCP/2628. Exact port has to be used when starting Docker container and setting JMX_PORT environment variable.

```
$ nano /etc/firewalld/services/kafka.xml
```

kafka.xml document example with Broker and JMX ports

```
<?xml version="1.0" encoding="utf-8"?>
<service>
    <short>kafka</short>
    <description>Kafka and Kafka JMX ports</description>
    <port protocol="tcp" port="9092"/>
    <port protocol="tcp" port="2628"/>
</service>
```

Load and apply rules

```
$ firewall-cmd --permanent --add-service=kafka
$ firewall-cmd --reload
```

Zookeeper

Zookeeper requires three ports: TCP/2888, TCP/3888 and TCP/2181

```
$ nano /etc/firewalld/services/zookeeper.xml
```

zookeeper.xml example with required ports

```
<?xml version="1.0" encoding="utf-8"?>
<service>
    <short>zookeeper</short>
    <description>Zokeeper ports</description>
    <port protocol="tcp" port="2888"/>
    <port protocol="tcp" port="3888"/>
    <port protocol="tcp" port="2181"/>
</service>
```

Load and apply rules

```
$ firewall-cmd --permanent --add-service=openshift-apps
$ firewall-cmd --reload
```

Group and user creation

Group id has to be 1001 and User id has to be 1001 due to hardcoded values. Bitnami images are not ran as root and requires specific user and group id.

Create group

```
$ groupadd kafka -g 1001
```

Create user

```
$ useradd svckafkad -u 1001 -g 1001
$ passwd svckafkad
$ usermod -aG wheel svckafkad
$ chage -I -1 -m 0 -M 99999 -E -1 svckafkad
```

Create folders

In order for Kafka and Zookeeper to have persistent storage, folder have to be created. Note the location of folders to be able to set correct folder locations in Docker run script. Also, a folder for keystore, truststore and scripts have to be created.

Data folders

These folders are used by Zookeeper and Kafka as a persistent storage.

```
$ mkdir -p /opt/data/kafka/
$ mkdir -p /opt/data/zookeeper/
```

Folder for misc scripts

```
$ mkdir -p /opt/scripts/
```

Folder for certificates, keystore and truststore

```
$ mkdir -p /opt/certificates/
```

Set folder permissions

```
$ chmod 775 -R /opt/data/
$ chmod 775 -R /opt/scripts/
$ chmod 775 -R /opt/certificates/
```

SSL configuration

In order to secure communication between Kafka Brokers and Brokers and Clients, certificates, keystore and truststore has to be created. We'll be using Confluent bash script to create everything necessary. This has to be performed only on first installed Kafka Broker. Keystore and truststore should be copied to other Kafka Brokers.

Download Confluent bash script

```
$ cd /opt/certificates/
$ wget https://raw.githubusercontent.com/confluentinc/confluent-platform-security-tools/master/kafka-generate-ssl.sh
```

Set executable bit on downloaded script

```
$ chmod +X kafka-generate-ssl.sh
```

Generate certificates, keystore and truststore

```
./kafka-generate-ssl.sh
```

Here is the sample output of certificate generation. Take care of passwords for truststore which will be used later on. Also when setting Common Name use wildcard on line 35 and 112 to be able to use same keystore on all Kafka brokers.

```
Welcome to the Kafka SSL keystore and truststore generator script.

First, do you need to generate a trust store and associated private key,
or do you already have a trust store file and private key?

Do you need to generate a trust store and associated private key? [yn] y

OK, we'll generate a trust store and associated private key.

First, the private key.

You will be prompted for:
 - A password for the private key. Remember this.
 - Information about you and your company.
 - NOTE that the Common Name (CN) is currently not important.
Generating a RSA private key
.....................................................+++++
......+++++
writing new private key to 'truststore/ca-key'
Enter PEM pass phrase:
Verifying - Enter PEM pass phrase:
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [XX]:HR
State or Province Name (full name) []:Grad Zagreb
Locality Name (eg, city) [Default City]:Zagreb
Organization Name (eg, company) [Default Company Ltd]:Sberbank d.d.
Organizational Unit Name (eg, section) []:OJ 030 Informatika
Common Name (eg, your name or your server's hostname) []:*.lab.local
Email Address []:030@sberbank.hr
```

```
Two files were created:
 - truststore/ca-key -- the private key used later to
   sign certificates
 - truststore/ca-cert -- the certificate that will be
   stored in the trust store in a moment and serve as the certificate
   authority (CA). Once this certificate has been stored in the trust
   store, it will be deleted. It can be retrieved from the trust store via:
   $ keytool -keystore <trust-store-file> -export -alias CARoot -rfc

Now the trust store will be generated from the certificate.

You will be prompted for:
 - the trust store's password (labeled 'keystore'). Remember this
 - a confirmation that you want to import the certificate
Enter keystore password:
Re-enter new password:
Owner: EMAILADDRESS=030@sberbank.hr, CN=*.lab.local, OU=OJ 030 Informatika, O=Sberbank d.d., L=Zagreb, ST=Grad
Zagreb, C=HR
Issuer: EMAILADDRESS=030@sberbank.hr, CN=*.lab.local, OU=OJ 030 Informatika, O=Sberbank d.d., L=Zagreb, ST=Grad
Zagreb, C=HR
Serial number: 14b598b3219192a7dc3788a48710610979e43330
Valid from: Mon Dec 16 15:39:07 CET 2019 until: Thu Dec 13 15:39:07 CET 2029
Certificate fingerprints:
         SHA1: 7E:C6:06:AE:8C:53:A6:E1:8D:64:71:C1:20:1B:F0:C5:76:1A:58:39
         SHA256: 53:33:CA:FD:D3:F2:AD:02:27:47:27:2A:24:E6:9E:E1:0F:A4:E1:A4:38:B8:5E:F7:DA:E2:B5:4F:86:25:85:0C
Signature algorithm name: SHA256withRSA
Subject Public Key Algorithm: 2048-bit RSA key
Version: 3

Extensions:

#1: ObjectId: 2.5.29.35 Criticality=false
AuthorityKeyIdentifier [
KeyIdentifier [
0000: 03 CA B1 B1 7D BB E6 35   B9 90 28 F4 D9 0A BA 23  .......5..(....#
0010: 1D BF C6 8E                                        ....
]
]

#2: ObjectId: 2.5.29.19 Criticality=true
BasicConstraints:[
  CA:true
  PathLen:2147483647
]

#3: ObjectId: 2.5.29.14 Criticality=false
SubjectKeyIdentifier [
KeyIdentifier [
0000: 03 CA B1 B1 7D BB E6 35   B9 90 28 F4 D9 0A BA 23  .......5..(....#
0010: 1D BF C6 8E                                        ....
]
]

Trust this certificate? [no]:  yes
Certificate was added to keystore

truststore/kafka.truststore.jks was created.

Continuing with:
 - trust store file:        truststore/kafka.truststore.jks
 - trust store private key: truststore/ca-key

Now, a keystore will be generated. Each broker and logical client needs its own
keystore. This script will create only one keystore. Run this script multiple
times for multiple keystores.

You will be prompted for the following:
 - A keystore password. Remember it.
 - Personal information, such as your name.
     NOTE: currently in Kafka, the Common Name (CN) does not need to be the FQDN of
```

```
          this host. However, at some point, this may change. As such, make the CN
          the FQDN. Some operating systems call the CN prompt 'first / last name'
 - A key password, for the key being generated within the keystore. Remember this.
Enter keystore password:
Re-enter new password:
What is your first and last name?
  [Unknown]:  *.lab.local
What is the name of your organizational unit?
  [Unknown]:  OJ 030 Informatika
What is the name of your organization?
  [Unknown]:  Sberbank d.d.
What is the name of your City or Locality?
  [Unknown]:  Zagreb
What is the name of your State or Province?
  [Unknown]:  Grad Zagreb
What is the two-letter country code for this unit?
  [Unknown]:  HR
Is CN=*.lab.local, OU=OJ 030 Informatika, O=Sberbank d.d., L=Zagreb, ST=Grad Zagreb, C=HR correct?
  [no]:  yes


'keystore/kafka.keystore.jks' now contains a key pair and a
self-signed certificate. Again, this keystore can only be used for one broker or
one logical client. Other brokers or clients need to generate their own keystores.

Fetching the certificate from the trust store and storing in ca-cert.

You will be prompted for the trust store's password (labeled 'keystore')
Enter keystore password:
Certificate stored in file <ca-cert>

Now a certificate signing request will be made to the keystore.

You will be prompted for the keystore's password.
Enter keystore password:

Now the trust store's private key (CA) will sign the keystore's certificate.

You will be prompted for the trust store's private key password.
Signature ok
subject=C = HR, ST = Grad Zagreb, L = Zagreb, O = Sberbank d.d., OU = OJ 030 Informatika, CN = *.lab.local
Getting CA Private Key
Enter pass phrase for truststore/ca-key:

Now the CA will be imported into the keystore.

You will be prompted for the keystore's password and a confirmation that you want to
import the certificate.
Enter keystore password:
Owner: EMAILADDRESS=030@sberbank.hr, CN=*.lab.local, OU=OJ 030 Informatika, O=Sberbank d.d., L=Zagreb, ST=Grad
Zagreb, C=HR
Issuer: EMAILADDRESS=030@sberbank.hr, CN=*.lab.local, OU=OJ 030 Informatika, O=Sberbank d.d., L=Zagreb, ST=Grad
Zagreb, C=HR
Serial number: 14b598b3219192a7dc3788a48710610979e43330
Valid from: Mon Dec 16 15:39:07 CET 2019 until: Thu Dec 13 15:39:07 CET 2029
Certificate fingerprints:
         SHA1: 7E:C6:06:AE:8C:53:A6:E1:8D:64:71:C1:20:1B:F0:C5:76:1A:58:39
         SHA256: 53:33:CA:FD:D3:F2:AD:02:27:47:27:2A:24:E6:9E:E1:0F:A4:E1:A4:38:B8:5E:F7:DA:E2:B5:4F:86:25:85:0C
Signature algorithm name: SHA256withRSA
Subject Public Key Algorithm: 2048-bit RSA key
Version: 3

Extensions:

#1: ObjectId: 2.5.29.35 Criticality=false
AuthorityKeyIdentifier [
KeyIdentifier [
0000: 03 CA B1 B1 7D BB E6 35   B9 90 28 F4 D9 0A BA 23  .......5..(....#
0010: 1D BF C6 8E                                        ....
]
]
```

```
#2: ObjectId: 2.5.29.19 Criticality=true
BasicConstraints:[
  CA:true
  PathLen:2147483647
]

#3: ObjectId: 2.5.29.14 Criticality=false
SubjectKeyIdentifier [
KeyIdentifier [
0000: 03 CA B1 B1 7D BB E6 35   B9 90 28 F4 D9 0A BA 23  .......5..(....#
0010: 1D BF C6 8E                                        ....
]
]

Trust this certificate? [no]:  yes
Certificate was added to keystore

Now the keystore's signed certificate will be imported back into the keystore.

You will be prompted for the keystore's password.
Enter keystore password:
Certificate reply was installed in keystore

All done!

Delete intermediate files? They are:
 - 'ca-cert.srl': CA serial number
 - 'cert-file': the keystore's certificate signing request
   (that was fulfilled)
 - 'cert-signed': the keystore's certificate, signed by the CA, and stored back
    into the keystore
Delete? [yn] y
```

Set folder ownership

Correct folder ownership has to be set in order for Kafka and Zookeeper to access and store data (hardcoded user and group id)

```
$ chown -R svckafkad:kafka /opt/data/
$ chown -R svckafkad:kafka /opt/scripts/
$ chown -R svckafkad:kafka /opt/certificates/
```

Start Kafka and Zookeeper

Pull Docker images

All Zookeeper and Kafka instances have to be on the same versions. To be able to use exact versions, specific tag has to be pulled.

```
$ docker pull bitnami/kafka:2.3.0-debian-9-r106
$ docker pull bitnami/zookeeper:3.4.14-debian-9-r31
```

Start Zookeeper

Create Zookeeper startup script

```
$ nano /opt/scripts/zookeeper.sh
```

Example of zookeeper.sh

```
#!/bin/bash
DATA_DIR=/opt/data/zookeeper
docker run -d --restart unless-stopped --network host \
         -e ZOO_SERVER_ID=1 \
         -e ZOO_SERVERS=0.0.0.0:2888:3888 \
         -e ZOO_ENABLE_AUTH=yes \
         -e ZOO_SERVER_USERS=user \
         -e ZOO_SERVER_PASSWORDS=password\
         -p 2181:2181 \
         -p 2888:2888 \
         -p 3888:3888 \
         -v $DATA_DIR:/bitnami/zookeeper \
         --name zookeeper_`date '+%Y_%m_%d__%H_%M_%S'` \
        bitnami/zookeeper:3.4.14-debian-9-r31
```

Set appropriate permissions and executable bit

```
$ chmod +X /opt/scripts/zookeeper.sh
$ chmod 755 /opt/scripts/zookeeper.sh
```

Start Zookeeper

```
$ /opt/scripts/zookeeper.sh
```

Check if Zookeeper started properly, Replace XXX with Docker container ID

```
$ docker ps
$ docker logs XXX -f
```

Start Kafka

Create Kafka startup script

```
$ nano /opt/scripts/kafka.sh
```

Example of kafka.sh

```
#!/bin/bash
DATA_DIR=/opt/data/kafka
docker run -d --restart unless-stopped --network host \
         -e ALLOW_PLAINTEXT_LISTENER=no \
         -e KAFKA_CERTIFICATE_PASSWORD=storepassword \
         -e KAFKA_INTER_BROKER_USER=admin \
         -e KAFKA_ZOOKEEPER_USER=user \
         -e KAFKA_ZOOKEEPER_PASSWORD=password \
         -e KAFKA_CFG_AUTO_CREATE_TOPICS_ENABLE=false \
         -e KAFKA_CFG_ADVERTISED_LISTENERS=SASL_SSL://:9092 \
         -e KAFKA_CFG_DELETE_TOPIC_ENABLE=true \
         -e KAFKA_CFG_DEFAULT_REPLICATION_FACTOR=1 \
         -e KAFKA_CFG_INTER_BROKER_PROTOCOL_VERSION=2.2.2 \
         -e KAFKA_CFG_LISTENERS=SASL_SSL://:9092 \
         -e KAFKA_CFG_ZOOKEEPER_CONNECT=lab-kafka01.lab.local:2181 \
         -e JMX_PORT=2628 \
         -p 9092:9092 \
         -p 2628:2628 \
         -v $DATA_DIR:/bitnami/kafka \
         -v '/opt/certificates/keystore/kafka.keystore.jks:/opt/bitnami/kafka/conf/certs/kafka.keystore.jks:ro'
\
         -v '/opt/certificates/truststore/kafka.truststore.jks:/opt/bitnami/kafka/conf/certs/kafka.truststore.
jks:ro' \
         --name kafka_`date '+%Y_%m_%d__%H_%M_%S'` \
         bitnami/kafka:2.3.0-debian-9-r106
```

Set appropriate permissions and executable bit

```
$ chmod +X /opt/scripts/kafka.sh
$ chmod 755 /opt/scripts/kafka.sh
```

Start Kafka

```
$ /opt/scripts/kafka.sh
```

Check if Kafka started properly, Replace XXX with Docker container ID

```
$ docker ps
$ docker logs XXX -f
```

Create all topics

Script will create all topics on the environment. This script should be ran from OSD001 with added dev folder, servers.list, jaas, keystore and truststore available in that folder.

```
#!/bin/bash

# Set Kafka tools path
KAFKA_FOLDER="/opt/kafka/kafka_2.12-2.3.0/"

# Set env
ENVIRONMENT='dev'
export KAFKA_OPTS="-Djava.security.auth.login.config=/opt/kafka/config/$ENVIRONMENT/kafka_jaas.conf"
source /opt/kafka/config/$ENVIRONMENT/servers.list
BOOTSTRAP_SERVERS=$BROKERS
ZOOKEEPER_SERVERS=$ZOOKEEPERS
PROPERTIES_CONFIG="/opt/kafka/config/$ENVIRONMENT/consumer.properties"

# error        6
KAFKA_FOLDER/bin/kafka-topics.sh $BOOTSTRAP_SERVERS --command-config $PROPERTIES_CONFIG --topic error --create
--partitions 6 --replication-factor 3 --config min.insync.replicas=2 --config segment.bytes=1073741824 --config
segment.ms=86400000 --config retention.ms=2678400000 --config retention.bytes=1073741824
```

```
# etl.banksoft.nks        12
KAFKA_FOLDER/bin/kafka-topics.sh $BOOTSTRAP_SERVERS --command-config $PROPERTIES_CONFIG --topic etl.banksoft.
nks --create --partitions 12 --replication-factor 3 --config min.insync.replicas=2 --config segment.
bytes=1073741824 --config segment.ms=86400000 --config retention.ms=2678400000 --config retention.
bytes=1073741824

# etl.cbs.archivedTransactions        12
KAFKA_FOLDER/bin/kafka-topics.sh $BOOTSTRAP_SERVERS --command-config $PROPERTIES_CONFIG --topic etl.cbs.
archivedTransactions --create --partitions 12 --replication-factor 3 --config min.insync.replicas=2 --config
segment.bytes=1073741824 --config segment.ms=86400000 --config retention.ms=2678400000 --config retention.
bytes=1073741824

# etl.cbs.dailyTransactions        6
KAFKA_FOLDER/bin/kafka-topics.sh $BOOTSTRAP_SERVERS --command-config $PROPERTIES_CONFIG --topic etl.cbs.
dailyTransactions --create --partitions 6 --replication-factor 3 --config min.insync.replicas=2 --config
segment.bytes=1073741824 --config segment.ms=86400000 --config retention.ms=2678400000 --config retention.
bytes=1073741824

# etl.fes.customers        6
KAFKA_FOLDER/bin/kafka-topics.sh $BOOTSTRAP_SERVERS --command-config $PROPERTIES_CONFIG --topic etl.fes.
customers --create --partitions 6 --replication-factor 3 --config min.insync.replicas=2 --config segment.
bytes=1073741824 --config segment.ms=86400000 --config retention.ms=2678400000 --config retention.
bytes=1073741824

# etl.karpo.authorizations        6
KAFKA_FOLDER/bin/kafka-topics.sh $BOOTSTRAP_SERVERS --command-config $PROPERTIES_CONFIG --topic etl.karpo.
authorizations --create --partitions 6 --replication-factor 3 --config min.insync.replicas=2 --config segment.
bytes=1073741824 --config segment.ms=86400000 --config retention.ms=2678400000 --config retention.
bytes=1073741824

# etl.karpo.cards        6
KAFKA_FOLDER/bin/kafka-topics.sh $BOOTSTRAP_SERVERS --command-config $PROPERTIES_CONFIG --topic etl.karpo.cards
--create --partitions 6 --replication-factor 3 --config min.insync.replicas=2 --config segment.bytes=1073741824
--config segment.ms=86400000 --config retention.ms=2678400000 --config retention.bytes=1073741824

# etl.karpo.transactions        12
KAFKA_FOLDER/bin/kafka-topics.sh $BOOTSTRAP_SERVERS --command-config $PROPERTIES_CONFIG --topic etl.karpo.
transactions --create --partitions 12 --replication-factor 3 --config min.insync.replicas=2 --config segment.
bytes=1073741824 --config segment.ms=86400000 --config retention.ms=2678400000 --config retention.
bytes=1073741824

# event.cbs.account        24
KAFKA_FOLDER/bin/kafka-topics.sh $BOOTSTRAP_SERVERS --command-config $PROPERTIES_CONFIG --topic event.cbs.
account --create --partitions 24 --replication-factor 3 --config min.insync.replicas=2 --config segment.
bytes=1073741824 --config segment.ms=86400000 --config retention.ms=2678400000 --config retention.
bytes=1073741824

# event.cbs.loan        6
KAFKA_FOLDER/bin/kafka-topics.sh $BOOTSTRAP_SERVERS --command-config $PROPERTIES_CONFIG --topic event.cbs.loan
--create --partitions 6 --replication-factor 3 --config min.insync.replicas=2 --config segment.bytes=1073741824
--config segment.ms=86400000 --config retention.ms=2678400000 --config retention.bytes=1073741824

# event.cbs.termDeposit        6
KAFKA_FOLDER/bin/kafka-topics.sh $BOOTSTRAP_SERVERS --command-config $PROPERTIES_CONFIG --topic event.cbs.
termDeposit --create --partitions 6 --replication-factor 3 --config min.insync.replicas=2 --config segment.
bytes=1073741824 --config segment.ms=86400000 --config retention.ms=2678400000 --config retention.
bytes=1073741824

# events.paymentOrders        12
KAFKA_FOLDER/bin/kafka-topics.sh $BOOTSTRAP_SERVERS --command-config $PROPERTIES_CONFIG --topic events.
paymentOrders --create --partitions 12 --replication-factor 3 --config min.insync.replicas=2 --config segment.
bytes=1073741824 --config segment.ms=86400000 --config retention.ms=2678400000 --config retention.
bytes=1073741824

# events.paymentOrders.requests        12
KAFKA_FOLDER/bin/kafka-topics.sh $BOOTSTRAP_SERVERS --command-config $PROPERTIES_CONFIG --topic events.
paymentOrders.requests --create --partitions 12 --replication-factor 3 --config min.insync.replicas=2 --config
segment.bytes=1073741824 --config segment.ms=86400000 --config retention.ms=2678400000 --config retention.
bytes=1073741824

# events.paymentOrders.bulk        6
```

```
KAFKA_FOLDER/bin/kafka-topics.sh $BOOTSTRAP_SERVERS --command-config $PROPERTIES_CONFIG --topic events.
paymentOrders.bulk --create --partitions 6 --replication-factor 3 --config min.insync.replicas=2 --config
segment.bytes=1073741824 --config segment.ms=86400000 --config retention.ms=2678400000 --config retention.
bytes=1073741824

# midasDataPublisher.internal        12
KAFKA_FOLDER/bin/kafka-topics.sh $BOOTSTRAP_SERVERS --command-config $PROPERTIES_CONFIG --topic
midasDataPublisher.internal --create --partitions 12 --replication-factor 3 --config min.insync.replicas=2 --
config segment.bytes=1073741824 --config segment.ms=86400000 --config retention.ms=2678400000 --config
retention.bytes=1073741824

# z.kafka.checkStatus        6
KAFKA_FOLDER/bin/kafka-topics.sh $BOOTSTRAP_SERVERS --command-config $PROPERTIES_CONFIG --topic z.kafka.
checkStatus --create --partitions 6 --replication-factor 3 --config min.insync.replicas=2 --config segment.
bytes=1073741824 --config segment.ms=86400000 --config retention.ms=2678400000 --config retention.
bytes=1073741824
```