# Быстрая установка MyBPM через docker-compose - демонстрационный режим

Можно быстро установить систему с помощью docker-compose на одной машине.

## Требования

Необходимо иметь любою linux-систему.

Нужно установить docker с плагином docker-compose.

Нужно получить от MyBPM логин и пароль для доступа к репозиторию образов hub.mybpm.kz

И создать регистрационную запись, указав полученный логин и пароль в команде:

docker login hub.mybpm.kz

## Установка

Создайте пустую папку и войдите в неё:

mkdir mybpm
cd mybpm

Теперь подготовим окружение для файла docker-compose.yaml

Файл инициации MongoDB:

mkdir mongo-init
cat > mongo-init/rs-init.js

С текстом:

rs.initiate();

Ctrl+D - чтобы выйти из ввода.

Директория для Kafka:

mkdir kf\_work

Файл генерации идентификатора кластера брокера кафки:

cat > kf\_work/create-kafka-cluster-id.sh

С содержимым:

#!/usr/bin/env bash
set -e
cd "$(dirname "$0")" || exit 113
KAFKA\_CLUSTER\_ID="$(docker run --rm -i bitnami/kafka:3.5.1 kafka-storage.sh random-uuid)"
KAFKA\_CLUSTER\_ID="$(echo -n "$KAFKA\_CLUSTER\_ID" | tr -d '[:blank:]' | tr -d '\n')"
echo -n "$KAFKA\_CLUSTER\_ID" > kafka-cluster-id.txt

Ctrl+D - чтобы выйти из ввода. И запустим его

bash kf\_work/create-kafka-cluster-id.sh

Должен появиться файл:

kf\_work/kafka-cluster-id.txt

Файла настроек кафки:

cat > kf\_work/kf.server.properties

С содержимым:

process.roles=broker,controller
node.id=1
controller.quorum.voters=1@kf:9091

############################# Socket Server Settings #############################

listeners=CONTROLLER://0.0.0.0:9091,IN\_BROKER://0.0.0.0:9092,FROM\_LOCALHOST://0.0.0.0:9093
advertised.listeners=IN\_BROKER://kf:9092,FROM\_LOCALHOST://localhost:10011
inter.broker.listener.name=IN\_BROKER
controller.listener.names=CONTROLLER
listener.security.protocol.map=CONTROLLER:PLAINTEXT,IN\_BROKER:PLAINTEXT,FROM\_LOCALHOST:PLAINTEXT
num.network.threads=3
num.io.threads=8
socket.send.buffer.bytes=102400
socket.receive.buffer.bytes=102400
socket.request.max.bytes=104857600

############################# Log Basics #############################

# A comma separated list of directories under which to store log files
log.dirs=/kafka-data
num.partitions=4
offsets.topic.num.partitions=4
num.recovery.threads.per.data.dir=1

auto.create.topics.enable=true
offsets.topic.replication.factor=1
transaction.state.log.replication.factor=1
transaction.state.log.min.isr=1
default.replication.factor=1
unclean.leader.election=true
compression.type=gzip
log.roll.hours=1

log.flush.interval.messages=10000
log.flush.interval.ms=1000

############################# Log Retention Policy #############################

log.retention.hours=-1
log.segment.bytes=1073741824
log.retention.check.interval.ms=300000

Ctrl+D - чтобы выйти из ввода.

Файл запуска кафки:

cat > kf\_work/run.sh

С содержимым:

#!/usr/bin/env bash
set -e
cd /kafka-data
if [ ! -f "/kafka-data/bootstrap.checkpoint" ] || [ ! -f "/kafka-data/meta.properties" ]; then
 rm -f "/kafka-data/bootstrap.checkpoint"
 rm -f "/kafka-data/meta.properties"
 KAFKA\_CLUSTER\_ID="$(cat /kf\_work/kafka-cluster-id.txt)"
 kafka-storage.sh format -t "$KAFKA\_CLUSTER\_ID" -c /kf\_work/kf.server.properties
fi
exec kafka-server-start.sh /kf\_work/kf.server.properties

И ему нужно установить атрибут исполнения:

chmod +x kf\_work/run.sh

Файл инициации PostgreSQL:

mkdir pg-init
cat > pg-init/init.sql

С содержимым:

CREATE USER mybpm WITH ENCRYPTED PASSWORD 't30my2ayTWsGKC0lf7P0SfCFc421fF';
ALTER USER mybpm WITH CREATEROLE;
CREATE DATABASE mybpm\_aux1 WITH OWNER mybpm;
CREATE USER test\_register\_util WITH ENCRYPTED PASSWORD '25UsbGa7G76X01F30K09D7v1a96vYZUybWVsZWqN';
ALTER USER test\_register\_util WITH CREATEROLE;
CREATE DATABASE test\_register\_util\_db WITH OWNER test\_register\_util;
CREATE USER in\_migration WITH ENCRYPTED PASSWORD 'xObV19Du1pUB931H5pQ8BnlR3RU5iY2T6fu3Yk4Z';
CREATE DATABASE in\_migration WITH OWNER in\_migration;

Файл для эластика:

mkdir es
cat > es/log4j2.properties

С содержимым:

status=error
appender.console.type=Console
appender.console.name=console
appender.console.layout.type=PatternLayout
appender.console.layout.pattern=[%d{ISO8601}][%-5p][%-25c{1.}] [%node\_name]%marker %m%n
######## Server JSON ############################
appender.rolling.type=RollingFile
appender.rolling.name=rolling
appender.rolling.fileName=${sys:es.logs.base\_path}${sys:file.separator}${sys:es.logs.cluster\_name}\_server.json
appender.rolling.layout.type=ECSJsonLayout
appender.rolling.layout.dataset=elasticsearch.server
appender.rolling.filePattern=${sys:es.logs.base\_path}${sys:file.separator}${sys:es.logs.cluster\_name}-%d{yyyy-MM-dd}-%i.json.gz
appender.rolling.policies.type=Policies
appender.rolling.policies.time.type=TimeBasedTriggeringPolicy
appender.rolling.policies.time.interval=1
appender.rolling.policies.time.modulate=true
appender.rolling.policies.size.type=SizeBasedTriggeringPolicy
appender.rolling.policies.size.size=128MB
appender.rolling.strategy.type=DefaultRolloverStrategy
appender.rolling.strategy.fileIndex=nomax
appender.rolling.strategy.action.type=Delete
appender.rolling.strategy.action.basepath=${sys:es.logs.base\_path}
appender.rolling.strategy.action.condition.type=IfFileName
appender.rolling.strategy.action.condition.glob=${sys:es.logs.cluster\_name}-\*
appender.rolling.strategy.action.condition.nested\_condition.type=IfAccumulatedFileSize
appender.rolling.strategy.action.condition.nested\_condition.exceeds=2GB
################################################
######## Server - old style pattern ###########
appender.rolling\_old.type=RollingFile
appender.rolling\_old.name=rolling\_old
appender.rolling\_old.fileName=${sys:es.logs.base\_path}${sys:file.separator}${sys:es.logs.cluster\_name}.log
appender.rolling\_old.layout.type=ECSJsonLayout
appender.rolling\_old.layout.pattern=[%d{ISO8601}][%-5p][%-25c{1.}] [%node\_name]%marker %m%n
appender.rolling\_old.filePattern=${sys:es.logs.base\_path}${sys:file.separator}${sys:es.logs.cluster\_name}-%d{yyyy-MM-dd}-%i.log.gz
appender.rolling\_old.policies.type=Policies
appender.rolling\_old.policies.time.type=TimeBasedTriggeringPolicy
appender.rolling\_old.policies.time.interval=1
appender.rolling\_old.policies.time.modulate=true
appender.rolling\_old.policies.size.type=SizeBasedTriggeringPolicy
appender.rolling\_old.policies.size.size=128MB
appender.rolling\_old.strategy.type=DefaultRolloverStrategy
appender.rolling\_old.strategy.fileIndex=nomax
appender.rolling\_old.strategy.action.type=Delete
appender.rolling\_old.strategy.action.basepath=${sys:es.logs.base\_path}
appender.rolling\_old.strategy.action.condition.type=IfFileName
appender.rolling\_old.strategy.action.condition.glob=${sys:es.logs.cluster\_name}-\*
appender.rolling\_old.strategy.action.condition.nested\_condition.type=IfAccumulatedFileSize
appender.rolling\_old.strategy.action.condition.nested\_condition.exceeds=2GB
################################################
rootLogger.level=info
rootLogger.appenderRef.console.ref=console
rootLogger.appenderRef.rolling.ref=rolling
rootLogger.appenderRef.rolling\_old.ref=rolling\_old
######## Deprecation JSON #######################
appender.deprecation\_rolling.type=RollingFile
appender.deprecation\_rolling.name=deprecation\_rolling
appender.deprecation\_rolling.fileName=${sys:es.logs.base\_path}${sys:file.separator}${sys:es.logs.cluster\_name}\_deprecation.json
appender.deprecation\_rolling.layout.type=ECSJsonLayout
# Intentionally follows a different pattern to above
appender.deprecation\_rolling.layout.dataset=deprecation.elasticsearch
appender.deprecation\_rolling.filter.rate\_limit.type=RateLimitingFilter
appender.deprecation\_rolling.filePattern=${sys:es.logs.base\_path}${sys:file.separator}${sys:es.logs.cluster\_name}\_deprecation-%i.json.gz
appender.deprecation\_rolling.policies.type=Policies
appender.deprecation\_rolling.policies.size.type=SizeBasedTriggeringPolicy
appender.deprecation\_rolling.policies.size.size=1GB
appender.deprecation\_rolling.strategy.type=DefaultRolloverStrategy
appender.deprecation\_rolling.strategy.max=4
appender.header\_warning.type=HeaderWarningAppender
appender.header\_warning.name=header\_warning
#################################################
logger.deprecation.name=org.elasticsearch.deprecation
logger.deprecation.level=WARN
logger.deprecation.appenderRef.deprecation\_rolling.ref=deprecation\_rolling
logger.deprecation.appenderRef.header\_warning.ref=header\_warning
logger.deprecation.additivity=false
######## Search slowlog JSON ####################
appender.index\_search\_slowlog\_rolling.type=RollingFile
appender.index\_search\_slowlog\_rolling.name=index\_search\_slowlog\_rolling
appender.index\_search\_slowlog\_rolling.fileName=${sys:es.logs.base\_path}${sys:file.separator}${sys:es.logs\
.cluster\_name}\_index\_search\_slowlog.json
appender.index\_search\_slowlog\_rolling.layout.type=ECSJsonLayout
appender.index\_search\_slowlog\_rolling.layout.dataset=elasticsearch.index\_search\_slowlog
appender.index\_search\_slowlog\_rolling.filePattern=${sys:es.logs.base\_path}${sys:file.separator}${sys:es.logs\
.cluster\_name}\_index\_search\_slowlog-%i.json.gz
appender.index\_search\_slowlog\_rolling.policies.type=Policies
appender.index\_search\_slowlog\_rolling.policies.size.type=SizeBasedTriggeringPolicy
appender.index\_search\_slowlog\_rolling.policies.size.size=1GB
appender.index\_search\_slowlog\_rolling.strategy.type=DefaultRolloverStrategy
appender.index\_search\_slowlog\_rolling.strategy.max=4
#################################################
#################################################
logger.index\_search\_slowlog\_rolling.name=index.search.slowlog
logger.index\_search\_slowlog\_rolling.level=trace
logger.index\_search\_slowlog\_rolling.appenderRef.index\_search\_slowlog\_rolling.ref=index\_search\_slowlog\_rolling
logger.index\_search\_slowlog\_rolling.additivity=false
######## Indexing slowlog JSON ##################
appender.index\_indexing\_slowlog\_rolling.type=RollingFile
appender.index\_indexing\_slowlog\_rolling.name=index\_indexing\_slowlog\_rolling
appender.index\_indexing\_slowlog\_rolling.fileName=${sys:es.logs.base\_path}${sys:file.separator}${sys:es.logs.cluster\_name}\
\_index\_indexing\_slowlog.json
appender.index\_indexing\_slowlog\_rolling.layout.type=ECSJsonLayout
appender.index\_indexing\_slowlog\_rolling.layout.dataset=elasticsearch.index\_indexing\_slowlog
appender.index\_indexing\_slowlog\_rolling.filePattern=${sys:es.logs.base\_path}${sys:file.separator}${sys:es.logs.cluster\_name}\
\_index\_indexing\_slowlog-%i.json.gz
appender.index\_indexing\_slowlog\_rolling.policies.type=Policies
appender.index\_indexing\_slowlog\_rolling.policies.size.type=SizeBasedTriggeringPolicy
appender.index\_indexing\_slowlog\_rolling.policies.size.size=1GB
appender.index\_indexing\_slowlog\_rolling.strategy.type=DefaultRolloverStrategy
appender.index\_indexing\_slowlog\_rolling.strategy.max=4
#################################################
logger.index\_indexing\_slowlog.name=index.indexing.slowlog.index
logger.index\_indexing\_slowlog.level=trace
logger.index\_indexing\_slowlog.appenderRef.index\_indexing\_slowlog\_rolling.ref=index\_indexing\_slowlog\_rolling
logger.index\_indexing\_slowlog.additivity=false
logger.com\_amazonaws.name=com.amazonaws
logger.com\_amazonaws.level=warn
logger.com\_amazonaws\_jmx\_SdkMBeanRegistrySupport.name=com.amazonaws.jmx.SdkMBeanRegistrySupport
logger.com\_amazonaws\_jmx\_SdkMBeanRegistrySupport.level=error
logger.com\_amazonaws\_metrics\_AwsSdkMetrics.name=com.amazonaws.metrics.AwsSdkMetrics
logger.com\_amazonaws\_metrics\_AwsSdkMetrics.level=error
logger.com\_amazonaws\_auth\_profile\_internal\_BasicProfileConfigFileLoader.name=com.amazonaws.auth.profile.internal.BasicProfileConfigFileLoader
logger.com\_amazonaws\_auth\_profile\_internal\_BasicProfileConfigFileLoader.level=error
logger.com\_amazonaws\_services\_s3\_internal\_UseArnRegionResolver.name=com.amazonaws.services.s3.internal.UseArnRegionResolver
logger.com\_amazonaws\_services\_s3\_internal\_UseArnRegionResolver.level=error
appender.audit\_rolling.type=RollingFile
appender.audit\_rolling.name=audit\_rolling
appender.audit\_rolling.fileName=${sys:es.logs.base\_path}${sys:file.separator}${sys:es.logs.cluster\_name}\_audit.json
appender.audit\_rolling.layout.type=PatternLayout
appender.audit\_rolling.layout.pattern={\
"type":"audit", \
"timestamp":"%d{yyyy-MM-dd'T'HH:mm:ss,SSSZ}"\
%varsNotEmpty{, "cluster.name":"%enc{%map{cluster.name}}{JSON}"}\
%varsNotEmpty{, "cluster.uuid":"%enc{%map{cluster.uuid}}{JSON}"}\
%varsNotEmpty{, "node.name":"%enc{%map{node.name}}{JSON}"}\
%varsNotEmpty{, "node.id":"%enc{%map{node.id}}{JSON}"}\
%varsNotEmpty{, "host.name":"%enc{%map{host.name}}{JSON}"}\
%varsNotEmpty{, "host.ip":"%enc{%map{host.ip}}{JSON}"}\
%varsNotEmpty{, "event.type":"%enc{%map{event.type}}{JSON}"}\
%varsNotEmpty{, "event.action":"%enc{%map{event.action}}{JSON}"}\
%varsNotEmpty{, "authentication.type":"%enc{%map{authentication.type}}{JSON}"}\
%varsNotEmpty{, "user.name":"%enc{%map{user.name}}{JSON}"}\
%varsNotEmpty{, "user.run\_by.name":"%enc{%map{user.run\_by.name}}{JSON}"}\
%varsNotEmpty{, "user.run\_as.name":"%enc{%map{user.run\_as.name}}{JSON}"}\
%varsNotEmpty{, "user.realm":"%enc{%map{user.realm}}{JSON}"}\
%varsNotEmpty{, "user.run\_by.realm":"%enc{%map{user.run\_by.realm}}{JSON}"}\
%varsNotEmpty{, "user.run\_as.realm":"%enc{%map{user.run\_as.realm}}{JSON}"}\
%varsNotEmpty{, "user.roles":%map{user.roles}}\
%varsNotEmpty{, "apikey.id":"%enc{%map{apikey.id}}{JSON}"}\
%varsNotEmpty{, "apikey.name":"%enc{%map{apikey.name}}{JSON}"}\
%varsNotEmpty{, "authentication.token.name":"%enc{%map{authentication.token.name}}{JSON}"}\
%varsNotEmpty{, "authentication.token.type":"%enc{%map{authentication.token.type}}{JSON}"}\
%varsNotEmpty{, "origin.type":"%enc{%map{origin.type}}{JSON}"}\
%varsNotEmpty{, "origin.address":"%enc{%map{origin.address}}{JSON}"}\
%varsNotEmpty{, "realm":"%enc{%map{realm}}{JSON}"}\
%varsNotEmpty{, "url.path":"%enc{%map{url.path}}{JSON}"}\
%varsNotEmpty{, "url.query":"%enc{%map{url.query}}{JSON}"}\
%varsNotEmpty{, "request.method":"%enc{%map{request.method}}{JSON}"}\
%varsNotEmpty{, "request.body":"%enc{%map{request.body}}{JSON}"}\
%varsNotEmpty{, "request.id":"%enc{%map{request.id}}{JSON}"}\
%varsNotEmpty{, "action":"%enc{%map{action}}{JSON}"}\
%varsNotEmpty{, "request.name":"%enc{%map{request.name}}{JSON}"}\
%varsNotEmpty{, "indices":%map{indices}}\
%varsNotEmpty{, "opaque\_id":"%enc{%map{opaque\_id}}{JSON}"}\
%varsNotEmpty{, "trace.id":"%enc{%map{trace.id}}{JSON}"}\
%varsNotEmpty{, "x\_forwarded\_for":"%enc{%map{x\_forwarded\_for}}{JSON}"}\
%varsNotEmpty{, "transport.profile":"%enc{%map{transport.profile}}{JSON}"}\
%varsNotEmpty{, "rule":"%enc{%map{rule}}{JSON}"}\
%varsNotEmpty{, "put":%map{put}}\
%varsNotEmpty{, "delete":%map{delete}}\
%varsNotEmpty{, "change":%map{change}}\
%varsNotEmpty{, "create":%map{create}}\
%varsNotEmpty{, "invalidate":%map{invalidate}}\
}%n
# "node.name" node name from the `elasticsearch.yml` settings
# "node.id" node id which should not change between cluster restarts
# "host.name" unresolved hostname of the local node
# "host.ip" the local bound ip (i.e. the ip listening for connections)
# "origin.type" a received REST request is translated into one or more transport requests. This indicates which processing layer generated the event "rest" or "transport" (internal)
# "event.action" the name of the audited event, eg. "authentication\_failed", "access\_granted", "run\_as\_granted", etc.
# "authentication.type" one of "realm", "api\_key", "token", "anonymous" or "internal"
# "user.name" the subject name as authenticated by a realm
# "user.run\_by.name" the original authenticated subject name that is impersonating another one.
# "user.run\_as.name" if this "event.action" is of a run\_as type, this is the subject name to be impersonated as.
# "user.realm" the name of the realm that authenticated "user.name"
# "user.run\_by.realm" the realm name of the impersonating subject ("user.run\_by.name")
# "user.run\_as.realm" if this "event.action" is of a run\_as type, this is the realm name the impersonated user is looked up from
# "user.roles" the roles array of the user; these are the roles that are granting privileges
# "apikey.id" this field is present if and only if the "authentication.type" is "api\_key"
# "apikey.name" this field is present if and only if the "authentication.type" is "api\_key"
# "authentication.token.name" this field is present if and only if the authenticating credential is a service account token
# "authentication.token.type" this field is present if and only if the authenticating credential is a service account token
# "event.type" informs about what internal system generated the event; possible values are "rest", "transport", "ip\_filter" and "security\_config\_change"
# "origin.address" the remote address and port of the first network hop, i.e. a REST proxy or another cluster node
# "realm" name of a realm that has generated an "authentication\_failed" or an "authentication\_successful"; the subject is not yet authenticated
# "url.path" the URI component between the port and the query string; it is percent (URL) encoded
# "url.query" the URI component after the path and before the fragment; it is percent (URL) encoded
# "request.method" the method of the HTTP request, i.e. one of GET, POST, PUT, DELETE, OPTIONS, HEAD, PATCH, TRACE, CONNECT
# "request.body" the content of the request body entity, JSON escaped
# "request.id" a synthetic identifier for the incoming request, this is unique per incoming request, and consistent across all audit events generated by that request
# "action" an action is the most granular operation that is authorized and this identifies it in a namespaced way (internal)
# "request.name" if the event is in connection to a transport message this is the name of the request class, similar to how rest requests are identified by the url path (internal)
# "indices" the array of indices that the "action" is acting upon
# "opaque\_id" opaque value conveyed by the "X-Opaque-Id" request header
# "trace\_id" an identifier conveyed by the part of "traceparent" request header
# "x\_forwarded\_for" the addresses from the "X-Forwarded-For" request header, as a verbatim string value (not an array)
# "transport.profile" name of the transport profile in case this is a "connection\_granted" or "connection\_denied" event
# "rule" name of the applied rule if the "origin.type" is "ip\_filter"
# the "put", "delete", "change", "create", "invalidate" fields are only present
# when the "event.type" is "security\_config\_change" and contain the security config change (as an object) taking effect
appender.audit\_rolling.filePattern=${sys:es.logs.base\_path}${sys:file.separator}${sys:es.logs.cluster\_name}\_audit-%d{yyyy-MM-dd}-%i.json.gz
appender.audit\_rolling.policies.type=Policies
appender.audit\_rolling.policies.time.type=TimeBasedTriggeringPolicy
appender.audit\_rolling.policies.time.interval=1
appender.audit\_rolling.policies.time.modulate=true
appender.audit\_rolling.policies.size.type=SizeBasedTriggeringPolicy
appender.audit\_rolling.policies.size.size=1GB
appender.audit\_rolling.strategy.type=DefaultRolloverStrategy
appender.audit\_rolling.strategy.fileIndex=nomax
logger.xpack\_security\_audit\_logfile.name=org.elasticsearch.xpack.security.audit.logfile.LoggingAuditTrail
logger.xpack\_security\_audit\_logfile.level=info
logger.xpack\_security\_audit\_logfile.appenderRef.audit\_rolling.ref=audit\_rolling
logger.xpack\_security\_audit\_logfile.additivity=false
logger.xmlsig.name=org.apache.xml.security.signature.XMLSignature
logger.xmlsig.level=error
logger.samlxml\_decrypt.name=org.opensaml.xmlsec.encryption.support.Decrypter
logger.samlxml\_decrypt.level=fatal
logger.saml2\_decrypt.name=org.opensaml.saml.saml2.encryption.Decrypter
logger.saml2\_decrypt.level=fatal

Теперь сам файл docker-compose-а:

cat > docker-compose.yaml

С содержимым:

networks:
 default:
 name: mybpm-aio-network

services:

 web:
 container\_name: mybpm-aio-web
 image: hub.mybpm.kz/mybpm-web-release:4.24.12.0
 restart: always
 ports:
 - "10000:8000" # WEB http://localhost:10000
 depends\_on:
 - api
 environment:
 MYBPM\_API\_HOST: "api"
 MYBPM\_API\_PORT: "8080"

 api:
 container\_name: mybpm-aio-api
 image: hub.mybpm.kz/mybpm-api-release:4.24.12.0
 restart: always
 ports:
 - "10001:8080" # SERVER http://localhost:10001/web/v2/test/hello
 - "10002:5005" # DEBUG localhost 10002
 - "10003:10003" # JConsole localhost 10003
 volumes:
 - ~/volumes/mybpm-aio-debug/api-logs:/var/log/mybpm
 # command:
 # - tail
 # - -f
 # - /etc/hosts
 depends\_on: [kf, es, zoo, mongo, pg]
 environment:
 # MY1BPM\_PLUGINS: "test"
 MYBPM\_USE\_SHENANDOAH: "yes"
 MYBPM\_JAVA\_DEBUG: "yes"
 MYBPM\_JAVA\_CONSOLE: "yes"
 MYBPM\_JAVA\_RMI\_HOST: "192.168.111.111" # здесь укажите IP вашей машины
 MYBPM\_JAVA\_RMI\_PORT: "10003"
 MYBPM\_JAVA\_JMX\_PORT: "10003"
 MYBPM\_LOGS\_COLORED: "true"
 MYBPM\_COMPANY\_CODE: "greetgo"
 MYBPM\_MONGO\_SERVERS: "mongodb://mongo:27017"
 MYBPM\_ZOOKEEPER\_SERVERS: "zoo:2181"
 MYBPM\_KAFKA\_SERVERS: "kf:9092"
 MYBPM\_AUX1\_DB\_NAME: "mybpm\_aux1"
 MYBPM\_AUX1\_HOST: "pg"
 MYBPM\_AUX1\_PORT: "5432"
 MYBPM\_AUX1\_USER\_NAME: "mybpm"
 MYBPM\_AUX1\_PASSWORD: "t30my2ayTWsGKC0lf7P0SfCFc421fF"
 MYBPM\_ELASTIC\_SEARCH\_SERVERS: "es:9200"
 MYBPM\_FILES\_MONGO\_SERVERS: "mongodb://mongo:27017"
 healthcheck:
 test: [ "CMD", "curl", "-f", "http://localhost:8080/web/health" ]
 interval: 30s
 timeout: 10s
 retries: 30
 start\_period: 10s

 pg: # docker exec -it mybpm-aio-pg psql -U postgres
 image: postgres:13.4
 container\_name: mybpm-aio-pg
 restart: always
 mem\_limit: 700M
 environment:
 POSTGRES\_PASSWORD: "iWAKOy4uS3v04T7bWM3SHNLiR8WyBP"
 ports:
 - "10018:5432"
 volumes:
 - ~/volumes/mybpm-aio-debug/pg-data:/var/lib/postgresql/data
 - ./pg-init:/docker-entrypoint-initdb.d
 command:
 - "docker-entrypoint.sh"
 - "-c"
 - "max-connections=900"

 mongo: # docker exec -it mybpm-aio-mongo mongo
 image: mongo:4.4.9
 container\_name: mybpm-aio-mongo
 mem\_limit: 700M
 restart: always
 ports:
 - "10017:27017"
 volumes:
 - ~/volumes/mybpm-aio-debug/mongo:/data/db
 - ./mongo-init:/docker-entrypoint-initdb.d
 command:
 - docker-entrypoint.sh
 - --bind\_ip\_all
 - --replSet
 - main
 - --profileFilter
 - '{"command.$$db": "mybpm"}'
 - --profile
 - "1"
# - --slowms
# - "0"

 mongo-express:
 image: mongo-express:1.0.0-alpha.4
 container\_name: mybpm-aio-mongo-express
 mem\_limit: 200M
 restart: always
 depends\_on:
 - mongo
 ports:
 - "10013:8081" # MONGO http://localhost:10013
 environment:
 ME\_CONFIG\_OPTIONS\_EDITORTHEME: cobalt
 ME\_CONFIG\_BASICAUTH\_USERNAME: admin
 ME\_CONFIG\_BASICAUTH\_PASSWORD: 111
 ME\_CONFIG\_MONGODB\_SERVER: mongo

 zoo:
 container\_name: mybpm-aio-zoo
 image: confluentinc/cp-zookeeper:5.5.0
 user: "0:0"
 mem\_limit: 200M
 restart: always
 ports:
 - "10012:2181"
 volumes:
 - ~/volumes/mybpm-aio-debug/zookeeper/data:/var/lib/zookeeper/data
 - ~/volumes/mybpm-aio-debug/zookeeper/log:/var/lib/zookeeper/log
 environment:
 ZOOKEEPER\_SERVER\_ID: 1
 ZOOKEEPER\_CLIENT\_PORT: 2181
 ZOOKEEPER\_TICK\_TIME: 2000
 ZOOKEEPER\_INIT\_LIMIT: 11
 ZOOKEEPER\_SYNC\_LIMIT: 5

 zoo-navigator:
 container\_name: mybpm-aio-zoo-navigator
 # noinspection SpellCheckingInspection
 image: elkozmon/zoonavigator:1.1.2
 restart: always
 mem\_limit: 500M
 ports:
 - "10010:9001" # http://localhost:10010
 environment:
 HTTP\_PORT: "9001"
 AUTO\_CONNECT\_CONNECTION\_ID: "MAIN"
 CONNECTION\_MAIN\_NAME: "main"
 CONNECTION\_MAIN\_CONN: "zoo:2181"

 kf:
 container\_name: mybpm-aio-kf
 image: bitnami/kafka:3.5.1
 mem\_limit: 1G
 restart: always
 ports:
 - "10011:9093"
 - "10015:7071"
 volumes:
 - ~/volumes/mybpm-aio-debug/kafka:/kafka-data
 - ./kf\_work:/kf\_work
 user: "0:0"
 entrypoint: [ /kf\_work/run.sh ]
 environment:
 KAFKA\_HEAP\_OPTS: "-Xmx1G -Xms1G"

 kafdrop:
 # noinspection SpellCheckingInspection
 container\_name: mybpm-aio-kafdrop
 # noinspection SpellCheckingInspection
 image: obsidiandynamics/kafdrop:4.0.0-SNAPSHOT
 depends\_on:
 - kf
 mem\_limit: 500M
 restart: always
 ports:
 - "10014:9000" # http://localhost:10014
 environment:
 KAFKA\_BROKERCONNECT: "kf:9092"
 SERVER\_PORT: "9000"
 JVM\_OPTS: "-Xms700M -Xmx700M"
 SERVER\_SERVLET\_CONTEXTPATH: "/"

 es:
 container\_name: mybpm-aio-es
 image: elasticsearch:8.3.2
 # noinspection ComposeUnknownValues
 mem\_limit: "${MYBPM\_ES\_MEMORY\_MAIN:-3000M}"
 restart: always
 ports:
 - "10016:9200" # http://localhost:10016
 # noinspection SpellCheckingInspection
 environment:
 #- cluster.name=docker-cluster
 - discovery.type=single-node
 - node.name=from-plugin
 - bootstrap.memory\_lock=true
 - index.store.type=hybridfs
 - "ES\_JAVA\_OPTS=-Xms${MYBPM\_ES\_MEMORY\_JAVA:-1500M} -Xmx${MYBPM\_ES\_MEMORY\_JAVA:-1500M}"
 # - TAKE\_FILE\_OWNERSHIP=true
 - xpack.security.enabled=false
 ulimits:
 memlock:
 soft: -1
 hard: -1
 volumes:
 - ~/volumes/mybpm-aio-debug/elasticsearch/data:/usr/share/elasticsearch/data
 - ~/volumes/mybpm-aio-debug/elasticsearch/logs:/usr/share/elasticsearch/logs
 - ./es/log4j2.properties:/usr/share/elasticsearch/config/log4j2.properties:ro

Файла перезапуска:

cat > restart.bash

С содержимым:

#!/usr/bin/env bash
cd "$(dirname "$0")" || exit 131
sudo mkdir -p "$HOME/volumes/mybpm-aio-debug/elasticsearch/data"
sudo mkdir -p "$HOME/volumes/mybpm-aio-debug/elasticsearch/logs"
sudo chmod 777 -R "$HOME/volumes/mybpm-aio-debug/elasticsearch"

docker compose down
EXIT="$?"
# shellcheck disable=SC2181
if [ "$EXIT" != "0" ] ; then
echo "%%%"
echo "%%% ERROR of : docker compose down"
echo "%%%"
exit "$EXIT"
fi

docker compose up -d
EXIT="$?"
# shellcheck disable=SC2181
if [ "$EXIT" != "0" ] ; then
echo "%%%"
echo "%%% ERROR of : docker compose up -d"
echo "%%%"
exit "$EXIT"
fi

echo "%%%"
echo "%%% ГОТОВО (restart)"
echo "%%%"

И файл остановки:

cat > stop.bash

С содержимым:

#!/usr/bin/env bash
cd "$(dirname "$0")" || exit 131
mkdir -p ~/volumes/mybpm-aio-debug/elasticsearch
sudo chmod a+rwx ~/volumes/mybpm-aio-debug/elasticsearch
docker compose down
EXIT="$?"
# shellcheck disable=SC2181
if [ "$EXIT" != "0" ] ; then
 echo "%%%"
 echo "%%% ERROR of : docker compose down"
 echo "%%%"
 exit "$EXIT"
fi

После чего запускаем файла

bash restart.bash

И смотрим логи системы:

docker compose logs -f api

Может случиться, что эластик не хочет стартовать, потому что у него нет доступа к директориям:

~/volumes/mybpm-aio-debug/elasticsearch/data
~/volumes/mybpm-aio-debug/elasticsearch/logs

Дайте доступ к этим директориям

sudo mkdir -p "$HOME/volumes/mybpm-aio-debug/elasticsearch/data"
sudo mkdir -p "$HOME/volumes/mybpm-aio-debug/elasticsearch/logs"
sudo chmod 777 -R "$HOME/volumes/mybpm-aio-debug/elasticsearch"

Когда появиться строка

INFO Tomcat started on port(s): 8080 (http) with context path ''

Это означает, что сервер стартанул и готов к работе.

Можно зайти в браузере по адресу:

http://localhost:10000

И получиться страницу логина и пароля. Для первого входа необходим логин и пароль пользователя root - он генерируется системой случайным образом и фиксируется в БД. Чтобы его посмотреть выполните команду:

docker exec -it mybpm-aio-mongo mongo

Откроется приглашение MongoDB:

main:PRIMARY>

Теперь можно вводить команды для MongoDB. Активируйте нужную базу данных:

use mybpm\_aux

И посмотрите коллекцию:

db.PersonPassword.find()

Должно появиться много строк, из них одна такая:

{ "\_id" : ObjectId("3b430203076fff3236b508cf"), "initPassword" : "root : 0jAWS240G2uTQ" }

В ней виден пароль рута.

Войдите в систему с этим паролем и тут-же его поменяйте, чтобы этот стал неактуальный.

Всё можете смотреть платформу MyBPM в демонстрационном режиме.

# Общие положения

Платформа MyBPM состоит из трёх основных компонентов:

1. MyBPM WEB - предоставляет статический контент для работы Web-части (HTML, CSS, JavaScript, картинки, шрифты).
2. MyBPM API - производит обработки запросов других компонентов: MyBPM WEB и внешних сервисов. В этом компоненте реализована вся бизнес-логика платформы.
3. MyBPM Cons - обслуживает получатели кафки. Этих компонентов может быть несколько (например: cons1, cons2, cons3), чтобы каждый обслуживал какой-то конкретный набор получателей кафки.

Для работы компонента MyBPM WEB нужен компонент MyBPM API. Поэтому вначале нужно установить и настроить его. А потом будет описано как установить MyBPM WEB.

Далее [Установка компонента MyBPM API](020-install-API.md)

# Установка компонента MyBPM API

## Подготовка баз данных

Прежде чем установить компонент MyBPM API необходимо подготовить доступы к следующим базам данных:

1. MongoDB - в этой БД будет храниться оперативная информация. Так же в ней будут храниться файлы. Если количество файлов будет велико, то их стоит вынести в отдельный кластер. В этом случае необходимо подготовить два кластера MongoDB: один для оперативной информации, другой для хранилища файлов.
2. ElasticSearch - в этой БД будет тоже храниться оперативная информация - копия MongoDB. Но использоваться будет по другому. В MongoDB платформа отправляет запросы только по идентификаторам - никаких поисковых запросов туда не отправляется, чтобы не замедлить работу этой БД. Все запросы связанные с поиском отправляются в ElasticSearch.
3. PostgreSQL - в ней храниться различная служебная информация для решения внутренних задач. Также в ней хранятся данные для отчётов.
4. Apache Kafka - в эту БД выгружаются все изменения данных в системе. Их можно хранить, при этом можно хранить историю изменений. Также Apache Kafka используется как брокер сообщений, для обмена точечными сообщениями между различными компонентами платформы.
5. Apache Zookeeper - данная БД используется для хранения конфигурации системы (конфигурационных файлов). Тут настраиваются различные взаимодействия платформы как внутри между подсистемами, так и снаружи - с другими сторонними системами.

## Установка Kubernetes

После того как подготовлены все базы данных нужно установить кластер Kubernetes.

## Подбор имени пространства имён для компонентов платформы MyBPM внутри Kubernetes

После установки кластера kubernetes необходимо подобрать имя пространства имён. Его рекомендуется выбирать из двух частей:

1. первая часть - имя самой платформы в рамках её назначения, например, если Вы используете платформу как CRM, то назовём первую часть как crm.
2. Вторая часть соответствует типу среды: например у нас имеется три среды: test, stage, prod. (тестовая, подготовительная, продуктивная). Например, на если данном этапе мы устанавливаем продуктивную среду, тогда в качестве имени второй части выбираем: prod.

Дальше эти две части соединяем тире, в нашем случае получается следующее имя:

crm-prod

В данном руководстве везде для пространства имён будет использоваться crm-prod. Вам стоит везде заменять его на то, которое у Вас получилось.

Вы можете использовать свои подходы для придумывания данного имени, но тот подход, зарекомендовал себя как удобный при условии, если у Вас много различного программного обеспечения размещённого на разных кластерах Kubernetes. По такому имени сразу становиться понятным, что имеется ввиду, и можно ли там смело что-то менять. Например, если Вы все продуктивные среды называете с окончанием -prod, то увидев такое окончание, Вы будете работать аккуратно.

## Установка MyBPM API

Для установки MyBPM API можно воспользоваться yaml-файлом:

Файл: 10-mybpm-api.yaml

apiVersion: v1
kind: Service
metadata:
 name: mybpm-api
 labels:
 app: mybpm-api
 namespace: crm-prod
spec:
 #type: NodePort
 ports:
 - protocol: TCP
 port: 8080
 targetPort: 8080
 #nodePort: 30207
 selector:
 app: mybpm-api
---
apiVersion: apps/v1
kind: Deployment
metadata:
 name: mybpm-api
 labels:
 app: mybpm-api
 namespace: crm-prod
spec:
 selector:
 matchLabels:
 app: mybpm-api
 replicas: 1
 template:
 metadata:
 labels:
 app: mybpm-api
 spec:
 nodeSelector:
 mybpm-api: ok
 containers:
 - name: main
 image: hub.mybpm.kz/mybpm-api-release:4.24.18.7 # здесь нужно указать актуальную версию
 imagePullPolicy: IfNotPresent
 resources:
 requests:
 memory: "3Gi"
 limits:
 memory: "3Gi"
 ports:
 - containerPort: 8080
 env:
 - { name: MYBPM\_JAVA\_OPTS, value: "-Xmx2Gi -Xms2Gi" }
 - { name: MYBPM\_ALL\_CONSUMERS\_OFF, value: "true" }
 - { name: MYBPM\_CONSUMER\_DIR, value: "api" }
 - { name: MYBPM\_COMPANY\_CODE, value: "greetgo" }
 - { name: MYBPM\_LICENCE\_NUMBER, value: "6120b632-f91a-416d-a7d0-180ffd7ce380" } # поставьте сюда Ваш номер лицензии
 - { name: MYBPM\_MONGO\_SERVERS, value: "mongo:27017" } # поставьте сюда правильный доступ к MongoDB
 - { name: MYBPM\_ZOOKEEPER\_SERVERS, value: "zookeeper-1:2181" } # поставьте сюда правильный доступ к Zookeeper
 - { name: MYBPM\_KAFKA\_SERVERS, value: "kf-1:9092,kf-2:9092,kf-3:9092" } # поставьте сюда правильный доступ к Kafka
 - { name: MYBPM\_AUX1\_HOST, value: "pg-aux1-host" }# поставьте сюда правильный доступ к PG
 - { name: MYBPM\_AUX1\_PORT, value: "5432" }
 - { name: MYBPM\_AUX1\_DB\_NAME, value: "mybpm\_aux1" }
 - { name: MYBPM\_AUX1\_USER\_NAME, valueFrom: { secretKeyRef: { name: pg\_mybpm\_aux1, key: username } } }
 - { name: MYBPM\_AUX1\_PASSWORD, valueFrom: { secretKeyRef: { name: pg\_mybpm\_aux1, key: password } } }
 - { name: MYBPM\_ELASTIC\_SEARCH\_SERVERS, value: "elastic:9200" }
 - { name: MYBPM\_NODE\_NAME, valueFrom: { fieldRef: { fieldPath: spec.nodeName } } }
 - { name: MYBPM\_MEMORY\_LIMIT, valueFrom: { resourceFieldRef: { containerName: main, resource: limits.memory } } }
 - { name: MYBPM\_NODE\_NAME, valueFrom: { fieldRef: { fieldPath: spec.nodeName } } }
 startupProbe:
 httpGet: { path: /web/health, port: 8080 }
 initialDelaySeconds: 15
 failureThreshold: 175
 periodSeconds: 4
 readinessProbe:
 httpGet: { path: /web/health, port: 8080 }
 initialDelaySeconds: 10
 periodSeconds: 4
 livenessProbe:
 httpGet: { path: /web/health, port: 8080 }
 initialDelaySeconds: 13
 periodSeconds: 4
 volumeMounts:
 - { name: log-dir, mountPath: "/var/log/mybpm" }
 volumes:
 - name: log-dir
 #... определение этого volume

Применить это файл можно с помощью команды:

kubectl apply -f 10-mybpm-api.yaml

Для работы этого файла, нужно завести секрет pg\_mybpm\_aux1. Его можно создать с помощью yaml-файла:

apiVersion: v1
kind: Secret
metadata:
 name: pg\_mybpm\_aux1
data:
 username: mybpm # здесь нужно поставить правильного пользователя к postgres-у
 password: 4jk1v23k4j231j4b # а здесь пароль

Подредактируйте в этих файлах параметры, как указано в комментариях.

При запуске сервера в логах будет информация о запуске.

При старте программа проверит доступы ко всем базам данных. Если какого-то доступа не будет, то произойдёт ошибка и программа остановиться с ошибкой.

Если доступы все будут, то система активирует все указанные сервисы.

Инициирует конфиги значениями по умолчанию.

И выдаст сообщение об успешном запуске. Выглядеть оно будет примерно так:

2024-08-12T10:27:10.581 k.g.m.s.d.l.MybpmServer INFO Started MybpmServer in 19.982 seconds (process running for 24.588)

После этого активирует получатели кафки.

## Примечания

### Настройка доступа к базам данных

Смотрите здесь: [Настройка п.о., для доступа к базам данных](050-connect-to-DB.md)

### Параметр nodeSelector

Этот параметр позволяет контролировать на каких node-ах можно запускать данную программу, а на каких нет.

Обратите внимание, что в yaml-файле есть параметр:

 nodeSelector

Этот параметр определяет node-у, на которой будет запускаться данная программа.

В нашем случае у этой настройки указано значение: mybpm-api: ok. Это значение обозначает, что данная программа будет запускаться только на тех node-ах, на которых установлен атрибут: mybpm-api, и значение этого атрибута должно быть ok. На node-ах, где это не соблюдается, данная программа запускаться не будет.

Пожалуйста, установите данный атрибут на тех node-ах, на которых Вы планируете запускать данную программу.

Если Вы хотите, чтобы программа запускалась на любых node-ах, то просто удалите этот параметр из yaml-файла.

### Масштабирование в ширину - параметр replicas

Обратите внимание, что в yaml-файле есть параметр:

replicas

Он обозначает количество подов, которые надо запустить. Если один под не справляется с текущей нагрузкой, то можно этот параметр увеличить. Тогда Kubernetes запустит дополнительные поды, и, через встроенный балансировщик нагрузки, запросы в программу распределяться равномерно.

Компонент MyBPM API не содержит в себе состояния, а значит можно запускать несколько серверов этого компонента и случайным образом распределять запросы между ними, и логических ошибок не будет появляться.

## Заключение

После того как Вы установили компонент MyBPM API, можно [установить компонент MyBPM WEB](030-install-WEB.md), и получить доступ к Web-интерфейсу платформы.

Далее [Установка компонента MyBPM WEB](030-install-WEB.md)

# Установка компонента MyBPM WEB

После того как компонент MyBPM API установлен, можно устанавливать компонент MyBPM WEB. Для этого нужно использовать yaml-файл:

Файл: 20-mybpm-web.yaml

apiVersion: v1
kind: Service
metadata:
 name: mybpm-web
 labels:
 app: mybpm-web
 namespace: crm-prod
spec:
 type: NodePort
 ports:
 - protocol: TCP
 port: 8000
 targetPort: 8000
 nodePort: 30208
 selector:
 app: mybpm-web
---
apiVersion: apps/v1
kind: Deployment
metadata:
 name: mybpm-web
 labels:
 app: mybpm-web
 namespace: crm-prod
spec:
 selector:
 matchLabels:
 app: mybpm-web
 replicas: 1
 template:
 metadata:
 labels:
 app: mybpm-web
 spec:
 nodeSelector:
 mybpm-web: ok
 containers:
 - name: mybpm-web
 image: hub.mybpm.kz/mybpm-web-release:4.24.18.5
 imagePullPolicy: IfNotPresent
 ports:
 - containerPort: 8000
 env:
 - {name: MYBPM\_API\_HOST, value: "mybpm-api"}
 - {name: MYBPM\_API\_PORT, value: "8080"}

После применения данного yaml-файла развернётся сервис этого компонента и он станет доступным по порту:

30208

И теперь, зайдя в браузере по адресу:

http://any-node-host:30208

Вы должны получить страницу входа:



Теперь Вам нужно войти в систему. Как это сделать описано по этой ссылке: [Первый вход в систему](../050-First-Enter.md)

Далее [Установка компонента MyBPM Cons1](040-install-CONS.md)

# Установка компонента MyBPM Cons1

После установки компонентов API и WEB платформа ещё не функционирует в полной мере. Необходимо наладить синхронизацию различных её модулей между собой с помощью Apache Kafka. Для этого нужно поднять ещё несколько компонентов MyBPM Cons.

Файл: 30-mybpm-cons1.yaml

apiVersion: apps/v1
kind: Deployment
metadata:
 name: mybpm-cons1
 labels:
 app: mybpm-cons1
 namespace: crm-prod
spec:
 selector:
 matchLabels:
 app: mybpm-cons1
 replicas: 1
 template:
 metadata:
 labels:
 app: mybpm-cons1
 spec:
 nodeSelector:
 mybpm-cons1: ok
 containers:
 - name: main
 image: hub.mybpm.kz/mybpm-api-release:4.24.18.7 # здесь нужно указать актуальную версию
 imagePullPolicy: IfNotPresent
 resources:
 requests:
 memory: "3Gi"
 limits:
 memory: "3Gi"
 ports:
 - containerPort: 8080
 env:
 - { name: MYBPM\_JAVA\_OPTS, value: "-Xmx2Gi -Xms2Gi" }
 - { name: MYBPM\_ALL\_KAFKA\_NOTIFICATIONS\_OFF, value: "true" }
 - { name: MYBPM\_CONSUMER\_DIR, value: "cons1" }
 - { name: MYBPM\_COMPANY\_CODE, value: "greetgo" }
 - { name: MYBPM\_LICENCE\_NUMBER, value: "6120b632-f91a-416d-a7d0-180ffd7ce380" } # поставьте сюда Ваш номер лицензии
 - { name: MYBPM\_MONGO\_SERVERS, value: "mongo:27017" } # поставьте сюда правильный доступ к MongoDB
 - { name: MYBPM\_ZOOKEEPER\_SERVERS, value: "zookeeper-1:2181" } # поставьте сюда правильный доступ к Zookeeper
 - { name: MYBPM\_KAFKA\_SERVERS, value: "kf-1:9092,kf-2:9092,kf-3:9092" } # поставьте сюда правильный доступ к Kafka
 - { name: MYBPM\_AUX1\_HOST, value: "pg-aux1-host" }# поставьте сюда правильный доступ к PG
 - { name: MYBPM\_AUX1\_PORT, value: "5432" }
 - { name: MYBPM\_AUX1\_DB\_NAME, value: "mybpm\_aux1" }
 - { name: MYBPM\_AUX1\_USER\_NAME, valueFrom: { secretKeyRef: { name: pg\_mybpm\_aux1, key: username } } }
 - { name: MYBPM\_AUX1\_PASSWORD, valueFrom: { secretKeyRef: { name: pg\_mybpm\_aux1, key: password } } }
 - { name: MYBPM\_ELASTIC\_SEARCH\_SERVERS, value: "elastic:9200" }
 - { name: MYBPM\_NODE\_NAME, valueFrom: { fieldRef: { fieldPath: spec.nodeName } } }
 - { name: MYBPM\_MEMORY\_LIMIT, valueFrom: { resourceFieldRef: { containerName: main, resource: limits.memory } } }
 - { name: MYBPM\_NODE\_NAME, valueFrom: { fieldRef: { fieldPath: spec.nodeName } } }
 startupProbe:
 httpGet: { path: /web/health, port: 8080 }
 initialDelaySeconds: 15
 failureThreshold: 175
 periodSeconds: 4
 readinessProbe:
 httpGet: { path: /web/health, port: 8080 }
 initialDelaySeconds: 10
 periodSeconds: 4
 livenessProbe:
 httpGet: { path: /web/health, port: 8080 }
 initialDelaySeconds: 13
 periodSeconds: 4
 volumeMounts:
 - { name: log-dir, mountPath: "/var/log/mybpm" }
 volumes:
 - name: log-dir
 #... определение этого volume

Отличие данного файла от api (10-mybpm-api.yaml [тут](020-install-API.md)) в следующем:

* MYBPM\_ALL\_KAFKA\_NOTIFICATIONS\_OFF - здесь отключены оповещения - они нужны только для API, где обслуживаются пользователи, а к этому серверу пользователи не подключаются поэтому здесь это не нужно
* MYBPM\_ALL\_CONSUMERS\_OFF - данная переменная убрана, значит получатели кафки запускаются, что нам и нужно здесь
* MYBPM\_CONSUMER\_DIR=cons1 - эта переменная определяет директорию в конфигурационной среде, в которой настраиваются получатели кафки. Здесь подробнее описана настройка получателей кафки: [Настройка кафка-получателей](../010-10-config-consumers.md)

### Масштабирование в ширину - параметр replicas

Обратите внимание, что в yaml-файле есть параметр:

replicas

Он обозначает количество подов, которые надо запустить. Если один под не справляется с текущей нагрузкой, то можно этот параметр увеличить. Тогда Kubernetes запустит дополнительные поды с этим сервером, который будет отдельными потоками обслуживать получатели кафки.

Подробнее смотрите в разделе: [Настройка кафка-получателей](../010-10-config-consumers.md) пункт "Настройка масштабирования кафка получателей"

# Настройка п.о., для доступа к базам данных

Детальное описание переменных окружения смотрите в отдельном разделе: [Переменные окружения](../010-01-env-variables.md)

Здесь описаны лишь какие-то общие концептуальные сведения.

## Доступ к MongoDB

Если подготовлен один кластер MongoDB для хранения как данных, так и файлов, то для доступа к нему используется одно п.о.:

* MYBPM\_MONGO\_SERVERS - для доступов к данным и файлам

Если же подготовлено два кластера MongoDB - отдельно для данных, и отдельно для файлов, то нужно использовать две переменные окружения:

* MYBPM\_MONGO\_SERVERS - для доступа к данным
* MYBPM\_MONGO\_FILES\_SERVERS - для доступа к файлам

## Доступ к Apache Kafka

Основная переменная окружения MYBPM\_KAFKA\_SERVERS обеспечивает доступ к кафке. Эта переменная определяет параметр bootstrap.servers. И в случае если аутентификация и авторизация не нужна, то этого достаточно. Но если доступ к кафке защищён, то нужно иметь возможность передать механизму доступа к кафке дополнительные параметры через переменные окружения. Для этого создана дополнительная переменная окружения, точнее не одна, а множество, которое определяется определённым префиксом, а именно:

MYBPM\_KAFKA\_CON\_...

Эти переменные позволяют передать параметры в механизм доступа.

Ниже приведён пример аутентифицированного доступа к кафке:

 MYBPM\_KAFKA\_SERVERS: "kf:9094"
 MYBPM\_KAFKA\_CON\_001: 'sasl.jaas.config = TXT:org.apache.kafka.common.security.plain.PlainLoginModule required serviceName="Kafka" username="alice" password="alice-secret";'
 MYBPM\_KAFKA\_CON\_002: 'sasl.mechanism = TXT:PLAIN'
 MYBPM\_KAFKA\_CON\_003: 'ssl.truststore.location = Base64\_to\_FILE : MIIEcgIB...wICJxA='
 MYBPM\_KAFKA\_CON\_004: 'ssl.truststore.password = TXT:111222'
 MYBPM\_KAFKA\_CON\_005: 'security.protocol = TXT:SASL\_SSL'

Подробно смотрите в разделе: [Переменные окружения](../010-01-env-variables.md).

## Доступ к ElasticSearch

Основная переменная окружения для доступа к ElasticSearch - MYBPM\_ELASTIC\_SEARCH\_SERVERS. В ней перечислен список хостов и портов, где располагается кластер.

Если доступ к кластеру защищён, то есть дополнительные переменные, в которых можно указать параметры аутентификации.

Подробнее смотрите: [Переменные окружения](../010-01-env-variables.md) - MYBPM\_ELASTIC\_SEARCH\_SERVERS.

## Доступ к PostgreSQL

Смотрите здесь: [Настройка PostgreSQL](../010-20-config-postgres.md)

# Рекомендации по ресурсам

## Минимальные



## Без отказоустойчивости



## С отказоустойчивостью



# Особенности установки MyBPM

Платформа MyBPM состоит из двух компонентов:

1. [MyBPM API](020-MyBPM-API.md)
2. [MyBPM WEB](040-MyBPM-WEB.md)

Вначале необходимо установить MyBPM API и убедиться, что он подключился ко всем БД и имеет все необходимые ему права.

Потом установить MyBPM WEB и подключить его к API.

# Переменные окружения

Переменные окружения (далее п.о.) задаются для запуска API-сервера. В таблице перечислен минимальный набор п.о. с примерами значений

|  |  |
| --- | --- |
| Переменная окружения | Пример значения |
| MYBPM\_COMPANY\_CODE | greetgo |
| MYBPM\_ZOOKEEPER\_SERVERS | localhost:10012 |
| MYBPM\_KAFKA\_SERVERS | localhost:10011 |
| MYBPM\_MONGO\_SERVERS | mongodb://mybpm:123@192.168.111.1:11017, 192.168.111.2:11018, 192.168.111.3:11019/admin |
| MYBPM\_AUX1\_DB\_NAME | mybpm\_aux1 |
| MYBPM\_AUX1\_HOST | localhost |
| MYBPM\_AUX1\_PORT | 10018 |
| MYBPM\_AUX1\_USER\_NAME | mybpm |
| MYBPM\_AUX1\_PASSWORD | 4jk3hb2145hj23bv4123jkb4 |
| MYBPM\_ELASTIC\_SEARCH\_SERVERS | localhost:10016 |

В следующей таблице указан более расширенный пример доступа к кафке:

|  |  |
| --- | --- |
| Переменная окружения | Пример значения |
| MYBPM\_KAFKA\_SERVERS | localhost:10011 |
| MYBPM\_KAFKA\_CON\_001 | sasl.jaas.config = TXT:org.apache.kafka.common.security.plain.PlainLoginModule... |
| MYBPM\_KAFKA\_CON\_002 | sasl.mechanism = TXT:PLAIN |
| MYBPM\_KAFKA\_CON\_003 | security.protocol = TXT:SASL\_SSL |
| MYBPM\_KAFKA\_CON\_004 | ssl.truststore.location = Base64\_to\_file :4hJ1/v324gHkjY43bjk51b5v234v423v4... |
| MYBPM\_KAFKA\_CON\_005 | ssl.truststore.password = TXT :SuperSecret |

Ниже смотрите детальное описание п.о.:

### MYBPM\_APP\_PREFIX

Префикс для БД

##### Возможные значения:

[a-z][a-z1-9]\*

Данный префикс используется для определения баз данных, например если указать префикс tor, то в MongoDB будут использоваться базы данных tor\_mybpm, tor\_mybpm\_aux1 и так далее для всех БД. Если префикс не указать, то имена будут использоваться без префикса. Это нужно, чтобы можно было на одной среде развернуть несколько независимых друг от друга платформ MyBPM.

### MYBPM\_LICENCE\_NUMBER

Номер лицензии платформы MyBPM

##### Возможные значения:

6120b632-f91a-416d-a7d0-180ffd7ce380

Для активации функционирования платформы MyBPM в соответствии с закупленной лицензией, необходимо номер этой лицензии указать в данной п.о.

Так же необходимо обеспечить доступ платформы к лицензионному сервису платформы MyBPM. Или настроить прокси-сервер к одному или нескольким адресам лицензионного сервиса.

### MYBPM\_LICENCE\_SERVICE\_ADDRESSES

Альтернативные адреса лицензионного сервиса.

##### Возможные значения:

http://some-host:port,https://some-another-host

ИЛИ

X,http://some-host:port,https://some-another-host

Список адресов лицензионного сервиса разделённые запятой. Так же в начале можно поставить X (большая латинская буква "икс") с запятой - это отменит использование пред-настроенных адресов лицензионного сервиса.

Если не указать эту п.о., то будут использоваться пред-настроенные адреса лицензионного сервиса.

Платформа по очереди будет вызывать адреса лицензионного сервиса. И как только один адрес сработает, то произойдёт полноценная активация платформы в соответствии с лицензией, номер которой указан в п.о. MYBPM\_LICENCE\_NUMBER. Если ни один адрес не сработает, то произойдёт полная блокировка работы платформы.

Если необходимо иметь возможность просматривать данные, которые ходят на лицензионный сервис, то можно сделать прокси сервер перенаправляющий REST-запросы на один из адресов лицензионного сервиса. Прокси сервер может работать как на протоколе HTTP, так и на протоколе HTTPS.

Важно чтобы прокси сервер перебрасывал заголовок signature как в запросе, так и в ответе запроса - без него платформа не сможет убедиться в актуальности лицензионного сервиса, и будет заблокирована.

### MYBPM\_ALL\_CONSUMERS\_OFF

Отключение всех исторических consumer-ов

##### Возможные значения:

true, false

Исторический consumer - это такой consumer, который считывает данные из кафки с начала времён. Каждое сообщение в таком consumer-е обрабатывается только один раз на одном сервере, даже если настроено несколько серверов обрабатывающих данный consumer.

Данная п.о. позволяет отключить все исторические consumer-ы, вне зависимости от того, что установлено в конфигурации ZooKeeper. Чтобы это сделать, нужно установить данную п.о. в значение true (см. kz.greetgo.mybpm\_util\_light.etc.Env.envBool).

Если эта п.о. не установлена или имеет другое значения, то consumer-ы запускаются и настраиваются по конфигурации из ZooKeeper.

### MYBPM\_RUN\_PROCESS\_OFF

Отключение запуска процессов

##### Возможные значения:

true, false

Если эту переменную поставить в true, то на этом сервере отключится запуск процессов

### MYBPM\_IN\_PG\_MIGRATION\_OFF

Отключение запуска процессов

##### Возможные значения:

true, false

Если эту переменную поставить в true, то на этом сервере отключится запуск миграции из in-таблиц в PostgreSQL

### MYBPM\_ALL\_KAFKA\_NOTIFICATIONS\_OFF

Отключение всех consumer-ов-оповещений

##### Возможные значения:

true, false

Consumer-оповещение - это такой consumer, который считывает данные из кафки с момента запуска. Притом каждое сообщение считывается ВСЕМИ серверами, которые обрабатывают этот consumer-оповещение.

Данная п.о. позволяет отключить все consumer-оповещения, вне зависимости от того, что установлено в конфигурации ZooKeeper. Чтобы это сделать, нужно установить данную п.о. в значение true (см. kz.greetgo.mybpm\_util\_light.etc.Env.envBool).

Если эта п.о. не установлена или имеет другое значения, то consumer-ы-оповещения запускаются и настраиваются по конфигурации из ZooKeeper.

ВНИМАНИЕ: Серверы, которые обрабатывают WebSocket-ы, должны иметь все consumer-оповещения ВКЛЮЧЁННЫМИ.

Данная переменная окружения применяется для тех серверов, которые обрабатывают только исторические consumer-ы. На таких серверах, не нужны оповещения, так как они не работают с пользователями. И желательно их отключить данной п.о.

### MYBPM\_PLUGINS

Активация установленных плагинов

##### Возможные значения:

Идентификаторы плагинов через запятую без пробелов

Чтобы установить плагин, недостаточно его jar-файл поместить на место загружаемых библиотек, нужно ещё его активировать. Для этого нужно его идентификатор указать в списке активируемых идентификаторов в этой п.о.

### MYBPM\_DISABLE\_CACHE

Отключение всего кэша

##### Возможные значения:

true, false

Установка данной п.о. в значение true (см. kz.greetgo.mybpm\_util\_light.etc.Env.envBool) позволяет полностью отключить все кэши в системе.

### MYBPM\_USE\_SHENANDOAH

Использовать SHENANDOAH

##### Возможные значения:

yes, чё-то другое

SHENANDOAH - это прогрессивный сборщик мусора, но по-умолчанию в виртуальной Java-машине активирован другой. Чтобы активировать этот, нужно установить данную п.о. в значение yes. Другие значения и отсутствие данной п.о. активируют сборщик мусора по умолчанию.

### MYBPM\_CONSUMER\_DIR

Имя директории куда помещаются конфиги консьюмеров

##### Возможные значения:

строка из латинских символов, цифр и деления (/)

По умолчанию система располагает файлы конфигурации консьюмеров в ZooKeeper по пути:

/mybpm/consumers

Это расположение можно поменять. Например, если задать данной п.о. значение:

MYBPM\_CONSUMER\_DIR=hello/world

То расположение файлов конфигурации измениться на следующее:

/mybpm/consumers-hello/world

Т.е. к прежнему значению добавиться тире ("-") и после него то, что в п.о.

### MYBPM\_AD\_USER и MYBPM\_AD\_PASS

Параметры доступа к Active Directory (AD)

##### Возможные значения:

строка из латинских символов, цифр

Параметры доступа к AD находятся ZooKeeper-е в файле конфигурации:

/mybpm/configs/AdConfig.txt

Кроме всего прочего имя пользователя и пароль доступа задаются в параметрах: adUsername, adPassword в открытом виде. Но в этом файле их можно не задавать, а передать через указанные п.о. и они будут приоритетными по сравнению со значениями из файла конфигурации.

### MYBPM\_JAVA\_DEBUG

Активация Java-отладки

##### Возможные значения:

yes, чё-то ещё

По умолчанию Java-машина запускается без внешнего отладчика. Для его активации необходимо установить данную п.о. в значение yes. Это запустит отладчик на порту 5005.

После запуска с этой переменной к системе можно подключаться отладчиком и делать точки остановки и пошагово отлаживать программу.

Так же есть возможность запустить систему в остановленном режиме, т.е. система остановиться на первой команде и работать не будет, а будет ждать подключения отладчика, чтобы он запустил её дальше или посмотрел, что происходить в самом начале, для этого нужно установить п.о. MYBPM\_JAVA\_DEBUG\_SUSPEND=y

### MYBPM\_JAVA\_DEBUG\_SUSPEND

Остановки при запуске

##### Возможные значения:

y, n

Эта п.о. используется совместно с MYBPM\_JAVA\_DEBUG. Установив данную п.о. в значение 'y' система запуститься в остановленном режиме, т.е. система остановиться на первой команде и дальше работать не будет, а будет ждать подключения отладчика, чтобы он запустил её дальше или посмотрел, что происходить в самом начале.

### MYBPM\_JAVA\_CONSOLE

Активировать консоль jconsole на порту 1099

##### Возможные значения:

yes, пусто

Если установить данную п.о. в значение yes, то после запуска системы к ней можно будет подключиться с помощью утилиты jconsole по порту 1099.

### MYBPM\_MAIN\_CLASS

Сменить главный класс: kz.greetgo.mybpm.server.war.app.MybpmServer

##### Возможные значения:

Полное имя класса с фунцией main

По умолчанию система запускается с класса:

kz.greetgo.mybpm.server.war.app.MybpmServer

Который поднимает сервер приложения на порту 8080. Этот класс можно сменить, указав другой через эту п.о. - это позволит запустить какие-нибудь альтернативные сервера или задачи.

### MYBPM\_JAVA\_OPTS

Дополнительные опции к Java-машине

##### Возможные значения:

Опции к Java-машине, которые обычно начинаются с -X

В данной переменной можно указать дополнительные опции к Java-машине, например по ограничению занимаемой памяти.

Например:

MYBPM\_JAVA\_OPTS="-Xms16G -Xmx16G"

Позволяет ограничить кучу в 16-ю гигабайтами ОЗУ.

### MYBPM\_COMPANY\_CODE

Код главного аккаунта

##### Возможные значения:

greetgo

Эту переменную всегда нужно устанавливать и устанавливать в значение greetgo.

Ранее можно было выбирать главные аккаунты, но с какого-то момента, добавилась возможность добавлять аккаунты динамически и главный аккаунт навсегда закрепился за greetgo.

### MYBPM\_AUX1\_HOST, MYBPM\_AUX1\_PORT, MYBPM\_AUX1\_DB\_NAME, MYBPM\_AUX1\_USER\_NAME, MYBPM\_AUX1\_PASSWORD

Доступ к PostgreSQL

##### Возможные значения:

строка из латинских символов, цифр

Данные переменные окружения определяют параметры доступа к БО PostgreSQL AUX1. А именно:

* MYBPM\_AUX1\_HOST - хосты сервера AUX1 через запятую. Может один хост. Также через : можно указывать порты
* MYBPM\_AUX1\_PORT - номер порта AUX1 для тех хостов, у которых не указан порт
* MYBPM\_AUX1\_DB\_NAME - имя базы данных
* MYBPM\_AUX1\_USER\_NAME - имя пользователя, из-под которого система соединяется с сервером
* MYBPM\_AUX1\_PASSWORD - пароль доступа к серверу

Первые три переменные определяют строку доступа к БД. Она получается такой:

jdbc:postgresql://${HOSTS\_PORTS}/${MYBPM\_AUX1\_DB\_NAME}?ApplicationName=MyBPM

Тут параметр HOSTS\_PORTS вычисляется из переменных MYBPM\_AUX1\_HOST и MYBPM\_AUX1\_PORT по следующему алгоритму.

1. Берётся переменная MYBPM\_AUX1\_HOST и делиться по запятым - получается массив
2. в этом массиве в тех значениях, в которых нет двоеточия, в конце добавляется порт MYBPM\_AUX1\_PORT через двоеточие
3. Полученный новый массив обратно соединяется через запятые и получается переменная HOSTS\_PORTS

Таким образом получается, что переменная MYBPM\_AUX1\_PORT используется как порт по умолчанию для тех мест из MYBPM\_AUX1\_HOST, в которых порт не указан.

Пусть например у нас есть база данных mybpm на трех серверах кластера PostgreSQL:

1. Адрес **10.100.1.101** порт **5432**
2. Адрес **10.100.1.102** порт **5432**
3. Адрес **10.100.1.103** порт **5432**

Тогда нужно задать следующие переменные окружения:

* MYBPM\_AUX1\_HOST = 10.100.1.101,10.100.1.102,10.100.1.103
* MYBPM\_AUX1\_PORT = 5432
* MYBPM\_AUX1\_DB\_NAME = mybpm

И по этим переменным окружениям будет генерироваться следующая URI доступа к базе данных:

jdbc:postgresql://10.100.1.101:5432,10.100.1.102:5432,10.100.1.103:5432/mybpm?ApplicationName=MyBPM

Теперь допусти порты на серверах разные, например такие:

1. Адрес **10.100.1.101** порт **1234**
2. Адрес **10.100.1.102** порт **4567**
3. Адрес **10.100.1.103** порт **1001**

Тогда нужно задать следующие переменные окружения:

* MYBPM\_AUX1\_HOST = 10.100.1.101:1234,10.100.1.102:4567,10.100.1.103:1001
* MYBPM\_AUX1\_PORT = 0
* MYBPM\_AUX1\_DB\_NAME = mybpm

И по этим переменным окружениям будет генерироваться следующая URI доступа к базе данных:

jdbc:postgresql://10.100.1.101:1234,10.100.1.102:4567,10.100.1.103:1001/mybpm?ApplicationName=MyBPM

Подробнее про URI доступа к базе данных PostgreSQL смотрите по ссылке:

<https://www.postgresql.org/docs/current/libpq-connect.html#LIBPQ-CONNSTRING-URIS>

### MYBPM\_ELASTIC\_SEARCH\_SERVERS, MYBPM\_ELASTIC\_SEARCH\_USER, MYBPM\_ELASTIC\_SEARCH\_PASS, MYBPM\_ELASTIC\_SEARCH\_CA\_CRT

Параметры доступа к эластику

##### Возможные значения:

строка из латинских символов, цифр

Данные п.о. определяют параметры доступа к серверу эластика. Главным п.о. является MYBPM\_ELASTIC\_SEARCH\_SERVERS - в зависимости от его значения используются другие п.о.

П.о. MYBPM\_ELASTIC\_SEARCH\_SERVERS может иметь следующие варианты:

Вариант со схемой доступа http:

MYBPM\_ELASTIC\_SEARCH\_SERVERS=[host1]:[port1],[host2]:[port2],[host3]:[port3]

или

MYBPM\_ELASTIC\_SEARCH\_SERVERS=http://[host1]:[port1],[host2]:[port2],[host3]:[port3]

Вариант со схемой доступа https:

MYBPM\_ELASTIC\_SEARCH\_SERVERS=https://[host1]:[port1],[host2]:[port2],[host3]:[port3]

где:

* [hostNN] - доменное имя или IP-адрес одного сервера кластера эластика.
* [portNN] - порт, который слушает сервер кластера эластика.

П.о. MYBPM\_ELASTIC\_SEARCH\_USER и MYBPM\_ELASTIC\_SEARCH\_PASS используются для указания авторизации и аутентификации.

Если используется схема доступа https, то можно использовать п.о. MYBPM\_ELASTIC\_SEARCH\_CA\_CRT, в которой указать полный путь к файлу сертификата соединения. Обычно этот файл имеет имя ca.crt

### MYBPM\_CORS

Отключение проверки CORS

##### Возможные значения:

ALL, или чё-то другое

Если установить эту п.о. в значение ALL, то уберётся контроль CORS с запросов - это нужно для тестирования клиента в режиме отладки. Любые другие значение, как и отсутствие самой п.о. контроль CORS оставляет на высоком уровне.

### MYBPM\_LOG\_DIR

Базовый путь размещения файлов логирования системы

##### Значение по-умолчанию (если п.о. не указана или пустая):

/var/log/mybpm

##### Возможные значения:

абсолютный путь к сущестующей директории, в которую есть доступ на запись

Определяет пусть к существующей директории, в которой система будет создавать файлы логирования

### MYBPM\_ZOOKEEPER\_SERVERS, MYBPM\_ZOO\_AUTH\_..., MYBPM\_ZOO\_CONF\_...

Определяет параметры доступа к кластеру ZooKeeper.

На платформе MyBPM получение доступа к системе происходит с помощью CuratorFrameworkFactory. Ниже схематично показан псевдокод подключения к Zookeeper-у.

 //@formatter:off
 String connectStr = "some-zoo-host:4324,another-zoo-host:6453";

 List<AuthInfo> authInfoList = new ArrayList<>();
 authInfoList.add(new AuthInfo("Client", Base64.getDecoder().decode("U2hhcm9uLlN0b25lOmQ1Wmgxa3MwM3Q=")));
 authInfoList.add(new AuthInfo("Server", Base64.getDecoder().decode("RGVuLkJyb3duOkRaek41WE1qNEU=")));

 final ZKClientConfig zkClientConfig = new ZKClientConfig();
 zkClientConfig.setProperty("zookeeper.sasl.client", "true");
 zkClientConfig.setProperty("zookeeper.ssl.quorum.trustStore.location", "/path/to/client-truststore.jks");
 zkClientConfig.setProperty("zookeeper.ssl.quorum.trustStore.password", "very-secret");

 try (CuratorFramework client = CuratorFrameworkFactory.builder()
 .connectString(connectStr)
 .zkClientConfig(zkClientConfig)
 .authorization(authInfoList)
 .build()) {
 // working with client
 }
//@formatter:on

Где:

base64\_to\_text( "U2hhcm9uLlN0b25lOmQ1Wmgxa3MwM3Q=" ) равно "Sharon.Stone:d5Zh1ks03t"
base64\_to\_text( "RGVuLkJyb3duOkRaek41WE1qNEU=" ) равно "Den.Brown:DZzN5XMj4E"

Чтобы платформа MyBPM подобным образом подключалась к Zookeeper необходимо передать следующие п.о. (формат docker consumer):

environment:
 MYBPM\_ZOOKEEPER\_SERVERS: "some-zoo-host:4324,another-zoo-host:6453"
 MYBPM\_ZOO\_AUTH\_001: "Client : U2hhcm9uLlN0b25lOmQ1Wmgxa3MwM3Q="
 MYBPM\_ZOO\_AUTH\_002: "Server : RGVuLkJyb3duOkRaek41WE1qNEU="
 MYBPM\_ZOO\_CONF\_001: "zookeeper.sasl.client = TXT: true"
 MYBPM\_ZOO\_CONF\_002: "zookeeper.ssl.quorum.trustStore.location = Base64\_to\_File: h34b2j4b234=="
 MYBPM\_ZOO\_CONF\_003: "zookeeper.ssl.quorum.trustStore.password = TXT: very-secret"

Здесь представьте, что h34b2j4b234== - это Base64 код содержимого файла /path/to/client-truststore.jks.

Далее детальное описание п.о.:

##### MYBPM\_ZOOKEEPER\_SERVERS

Определяет список серверов кластера ZooKeeper.

Возможные значения п.о.:

Разделённый запятыми последовательности хост:порт

##### MYBPM\_ZOO\_AUTH\_...

Определяет параметры авторизации. Платформа MyBPM считывает все п.о., которые начинаются на MYBPM\_ZOO\_AUTH\_ и каждую превращает в объект класса AuthInfo (этот класс имеется в примере кода выше). Все эти объекты складывает в массив и передаёт коду обеспечения доступа к Zookeeper (он находиться внутри класса CuratorFramework - это показано на примере кода выше).

Класс авторизации содержит два поля:

public class AuthInfo {
 final String scheme; // схема авторизации
 final byte[] auth; // данные для авторизации
}

Эти поля заполняются из значения п.о., которое должно иметь конкретный формат, а именно:

<schema> : <base64-код>

Где:

<schema> - это текст содержащий схему авторизации, и он передаётся в поле AuthInfo.scheme.

<base64-код> - это текст, представляющий собой Base64-код массива байтов, который передаётся в поле AuthInfo.auth.

##### MYBPM\_ZOO\_CONF\_...:

Определяет конфигурацию подключения к Zookeeper-у. Платформа MyBPM считывает все п.о., начинающиеся с префикса MYBPM\_ZOO\_CONF\_, и каждое значение превращает в пару ключ-значение. Потом эти пары передаются объекту класса ZKClientConfig с помощью метода setProperty(...) - тем самым определяя конфигурацию доступа. На примере кода выше это показано.

Формат значения данных п.о. следующий:

<имя-параметра> = <ТИП> : <значение-параметра>

Где:

<имя-параметра> - текст, представляющий имя параметра, который передаётся в качестве ключа в метод setProperty(...) первым аргументом.

<ТИП> - текст, представляющий тип значения. Может иметь следующие варианты: TXT, Base64\_To\_File - ниже описаны их значения.

<значение-параметра> - тест, который используется в зависимости от типа. Пробелы всегда обрезаются с обеих сторон.

Если <ТИП> == 'TXT', то это обозначает, что <значение-параметра> нужно передать в качестве значения в метод setProperty(...) вторым аргументом.

Если <ТИП> == 'Base64\_To\_File', то это обозначает, что <значение-параметра> должно представлять собой Base64-код содержимого некого файла. Платформа MyBPM возьмёт этот Base64-код, превратит его в массив байт, и сохранит во временный файл. А в качестве значения в метод setProperty(...) вторым аргументом подставит абсолютный путь к этому файлу. После этого система авторизации внутри класса CuratorFramework сможет получить к нему доступ и использовать по назначению.

#### ПРИМЕЧАНИЕ

Иногда для доступа к Zookeeper-у необходимо при запуске Java-машины определить параметр:

-Djava.security.auth.login.config=/path/to/zookeeper-jaas.conf

Для этого можно использовать п.о. MYBPM\_SYSTEM\_PROPERTY\_ (его описание ищите на этой странице)

### MYBPM\_KAFKA\_SERVERS, MYBPM\_KAFKA\_CON\_...

Определяют параметры доступа к кафке.

##### Возможные значения для MYBPM\_KAFKA\_SERVERS:

Разделённый запятыми последовательности хост:порт

Определяет список серверов кластера Kafka. Эта переменная присваивается параметру

bootstrap.servers

В списке параметров доступа к Кафке как для поставщика данных (Producer) так и для потребителя данных (Consumer).

Например, для подключения к Кафке в виде поставщика данных можно воспользоваться кодом:

void someMethod() {
 Map<String, Object> prop = new HashMap<>();

 prop.put("bootstrap.servers", "localhost:13113"); // Вот сюда вставляется значение из п.о. MYBPM\_KAFKA\_SERVERS
 prop.put("key.serializer", StringSerializer.class.getName());
 prop.put("value.serializer", StringSerializer.class.getName());
 prop.put("request.timeout.ms", "5000");
 prop.put("max.block.ms", "5000");
 prop.put("metadata.fetch.timeout.ms", "5000");
 prop.put("enable.auto.commit", "false");
 prop.put("acks", "all");

 try (KafkaProducer<String, String> producer = new KafkaProducer<>(prop)) {
 // working with producer ...
 }
}

В этом коде в параметр bootstrap.servers вставляется значения из п.о. MYBPM\_KAFKA\_SERVERS. Остальные параметры платформа MyBPM генерирует по другой логике, в зависимости от ситуации.

Может случиться так, что доступ к Кафке предоставляется через безопасное соединение, и тогда Java-код должен выглядеть примерно так:

 void libraryMethod() {
 // ....
 Map<String, Object> prop = new HashMap<>();

 prop.put("key.... "); // (здесь параметры не связанные с доступом)

 prop.put("bootstrap.servers", "localhost:13113");

 prop.put("sasl.mechanism", "PLAIN");
 prop.put("security.protocol", "SASL\_SSL");
 prop.put("ssl.truststore.location", "/some/path/to/existing/file/client.truststore.jks");
 prop.put("ssl.truststore.password", "111222");
 prop.put("sasl.jaas.config", "org.apache.kafka.common.security.plain.PlainLoginModule" +
 " required serviceName=\"Kafka\" username=\"alice\" password=\"alice-secret\";");

 try (KafkaProducer<String, String> producer = new KafkaProducer<>(prop)) {
 // working with producer ...
 }
}

Т.е. нужно передать дополнительные параметры. Притом есть параметры, которые представляют путь к существующему файлу. Этот файл нужно подготовить, передать на сервер, и указать к нему путь.

##### Значения MYBPM\_KAFKA\_CON\_...

Кроме MYBPM\_KAFKA\_SERVERS можно ещё определить несколько дополнительных п.о. которые будут дополнительно добавляться к списку параметров доступа к Кафке.

Чтобы это сделать нужно указать несколько п.о., которые начинаются с MYBPM\_KAFKA\_CON\_ и имеют только цифры после этого префикса, которые нужно для того, чтобы можно было указать несколько параметров с одним смыслом.

Значения у этих п.о. должны иметь вид:

<имя-параметра> = <ТИП> : <значение-параметра>

Где:

<имя-параметра> - это строка, которая будет передаваться в качестве ключа в объект prop (см. Java-код выше).

<ТИП> - одно из значений: TXT или Base64\_to\_File

<значение-параметра> - строка, которая будет обрабатываться в зависимости от <ТИП>-а.

Если <ТИП> = TXT, то это обозначает, что <значение-параметра> будет переходить в prop как есть - один-к-одному.

Если <ТИП> = Base64\_to\_File, то это обозначает, что <значение-параметра> должно представлять собой код Base64. Система возьмёт этот код Base64, превратит его в последовательность байт, и сохранит его во временный файл. А в объект prop будет передана строка с абсолютным путём к этому файлу.

Таким образом Вы можете передать файл, не имея доступа к серверу.

Другими словами Вы можете сделать так:

Получите из Вашего файла код Base64, например командой:

base64 -w0 имя\_исходного\_файла > имя\_файла\_где\_будет\_сохранён\_base64

Допустим у Вас получился код (Ваш код может быть значительно длинней):

4kjh32b454/534b256k45b64l3b21434==

Этот код передайте в переменную окружения следующим образом (здесь использован синтаксис docker compose):

environment:
 MYBPM\_KAFKA\_CON\_004: "ssl.truststore.location = Base64\_to\_File:4kjh32b454/534b256k45b64l3b21434=="

И тогда в объект prop прилетит параметр с ключом ssl.truststore.location и значением некого абсолютного пути к файлу, содержимое которого будет соответствовать указанному Вами коду Base64.

Если у Вас есть доступ к серверу, то вы можете скопировать файл на сервер, например по пути:

/ssl/client.truststore.jks

И указать переменную окружения:

environment:
 MYBPM\_KAFKA\_CON\_004: "ssl.truststore.location = TXT:/ssl/client.truststore.jks"

Путь будет скопирован в объект prop как есть - один-к-одному.

### MYBPM\_MONGO\_SERVERS, MYBPM\_MONGO\_DB, MYBPM\_MONGO\_USER, MYBPM\_MONGO\_PASS

Параметры доступа к кластеру MongoDB

Данные переменные окружения определяют параметры доступа к кластеру MongoDB. Основной переменной окружения является MYBPM\_MONGO\_SERVERS - в зависимости от её значение определяется, будут ли использоваться другие переменные окружения.

Если MYBPM\_MONGO\_SERVERS НЕ содержит подстроку '://', то строка доступа к кластеру определяется по следующей схеме:

Если MYBPM\_MONGO\_USER определено и НЕ пустое, то получается следующая строка доступа:

mongodb://${MYBPM\_MONGO\_USER}:${MYBPM\_MONGO\_PASS}@${MYBPM\_MONGO\_SERVERS}/${MYBPM\_MONGO\_DB}

Если MYBPM\_MONGO\_USER НЕ определено или пустое, то получается такая строка доступа:

mongodb://${MYBPM\_MONGO\_SERVERS}/${MYBPM\_MONGO\_DB}

Притом /${MYBPM\_MONGO\_DB} добавляется только в том случае, если MYBPM\_MONGO\_DB определено и НЕ пустое.

В приведённых обозначениях подстановка делается по синтаксису интерпретатора bash, т.е. в место ${HI} подставляется значение переменной окружения HI

Если MYBPM\_MONGO\_SERVERS содержит подстроку '://', то для доступа к кластеру MongoDB используется только данное значение. Переменные окружения MYBPM\_MONGO\_DB, MYBPM\_MONGO\_USER, MYBPM\_MONGO\_PASS будут проигнорированы.

В этом случае значение переменной окружения MYBPM\_MONGO\_SERVERS будет использоваться как строка доступа к MongoDB. Подробное описание данной строки доступа приводиться в официальной документации по MongoDB, а именно по ссылке:

<https://www.mongodb.com/docs/manual/reference/connection-string/>

Если вкратце, то шаблон значения (взятый из приведённой ссылки) может быть такой:

MYBPM\_MONGO\_SERVERS = mongodb://[username:password@]host1[:port1][,...hostN[:portN]][/[defaultauthdb][?options]]

Тут квадратные скобки обозначают не обязательное значение.

### MYBPM\_MONGO\_FILES\_SERVERS, MYBPM\_MONGO\_FILES\_DB, MYBPM\_MONGO\_FILES\_USER, MYBPM\_MONGO\_FILES\_PASS

Параметры доступа к кластеру MongoDB для хранения файлов

Эти п.о. используются так же как и MYBPM\_MONGO\_[SERVERS,USER...], но для доступа к кластеру MongoDB, в котором хранятся файлы.

Если данные п.о. не заданны, то используются переменные MYBPM\_MONGO\_[SERVERS,USER...] для доступа к кластеру файлов. Т.е. в таком случае файлы хранятся там, же где и остальные данные.

### MYBPM\_SCHEDULER\_OFF

Если эта переменна установлена в yes, то шедулер не запускается

### MYBPM\_CACHE\_REDIS\_HOST\_PORT

В этой переменной нужно передать хост, порт, логин и пароль разделённые двоеточием, которые указывают на сервер redis. Он будет использоваться для кэширования. Если эта переменная окружения не указана, то будет использоваться внутренний кэш на основе библиотеки caffeine.

Переменная может содержать только хост, например:

192.168.11.23

В этом случае система будет обращаться по адресу **192.168.11.23** и порту **6379**

Переменная может содержать хост и порт, разделённые двоеточием, например:

redis.super.host.kz:40110

В этом случае система будет обращаться по адресу **redis.super.host.kz** и порту **40110**

Переменная может содержать хост, порт, логин и пароль, разделённые двоеточием, например:

redis.wood.com:34089:SharonStone:4gjv321gh5v:4v::23

В этом случае система будет обращаться по адресу **redis.wood.com** и порту **34089** под пользователем **SharonStone** и паролем: 4gjv321gh5v:4v::23

### MYBPM\_TOPIC\_SUFFIX\_REFRESH

Добавление суффикса топику REFRESH

##### Возможные значения:

строка из латинских символов, цифр

Данная переменная окружения позволяет добавить суффикс к топику REFRESH в кафке. Если эта переменная не определена или имеет пустое значение, то топик так и называется REFRESH. Но если эта переменная равна например:

MYBPM\_TOPIC\_SUFFIX\_REFRESH = 123

То топик REFRESH не используется, а вместо него используется топик REFRESH123.

### MYBPM\_ELASTIC\_NUMBER\_OF\_REPLICAS

Количество реплик, которые есть у каждого основного шарда в ElasticSearch.

##### Возможные значения:

целочисленные значения

Данная переменная окружения позволяет настройку количества репликационных шардов при создании основного шарда. Если эта переменная не указана, значение по умолчанию для данной переменной окружения является **1**. Если значение данной переменной окружения является:

MYBPM\_ELASTIC\_NUMBER\_OF\_REPLICAS = 2

Тогда при создании индекса, будет создано **2** репликационных шарда для основного шарда.

### MYBPM\_SCRIPTS\_SHOW\_PANEL\_WITH\_OLD\_ELEMENTS

Открывает панель с устаревшими компонентами в редакторе скриптов.

##### Возможные значения:

yes,no

Старыми компонентами в скриптах пользоваться нельзя, поэтому панель для них убрана. НО... если всё-таки сильно хочется, то можно установить эту переменную окружения со значением yes и панель станет доступна

### MYBPM\_SYSTEM\_PROPERTY\_...

Задаёт системные параметры внутри виртуальной Java-машины

##### Возможные значения:

<имя-параметра> = <ТИП> : <значение-параметра>

Где:

<имя-параметра> - это строка, которая будет передаваться в качестве имени системного параметра

<ТИП> - одно из значений: TXT или Base64\_to\_File

<значение-параметра> - строка, которая будет обрабатываться в зависимости от <ТИП>-а.

Если <ТИП> = TXT, то это обозначает, что <значение-параметра> будет переходить как есть - один-к-одному. Другими словами для значения:

<имя-параметра> = TXT : <значение-параметра>

Будет устанавливаться следующее значение системного параметра:

System.setProperty( "<имя-параметра>", "<значение-параметра>" );

Если <ТИП> = Base64\_to\_File, то это обозначает, что <значение-параметра> должно представлять собой код Base64. Система возьмёт этот код Base64, превратит его в последовательность байт, и сохранит её во временный файл. А в системный параметр будет передана строка с абсолютным путём к этому файлу. Другими словами для значения:

<имя-параметра> = Base64\_to\_File : jk34hb5hjg4352hv35vj==

Будет устанавливаться следующее значение системного параметра:

System.setProperty( "<имя-параметра>", "/абсолютный/путь/к/файлу/c/содержимым/jk34hb5hjg4352hv35vj==" );

С помощью переменных, которые начинаются на MYBPM\_SYSTEM\_PROPERTY\_ можно передать системные параметры в виртуальную машину Java - по одному параметру за одну переменную окружения. Системные параметры внутри виртуальной машины доступны с помощью функции:

System.getProperty("system.param.key")

Например, если в docker compose указать переменные окружения:

environment:
 MYBPM\_SYSTEM\_PROPERTY\_001: "some.key = TXT: some-value"
 MYBPM\_SYSTEM\_PROPERTY\_002: "test.config = Base64\_to\_File: jhkb45hj3kB254BV/24hgV355=="

То внутри Java-машины можно будет вызвать:

System.getProperty("some.key");

И он вернёт строку: "some-value". А если вызвать:

System.getProperty("test.config");

То он вернёт путь к файлу, в котором лежат байты из Base64-кода: jhkb45hj3kB254BV/24hgV355==

### MYBPM\_MEMORY\_LIMIT

Указание программе сервера ограничение на оперативную память, которое наложено внешней средой, например Kubernetes-ом.

##### Возможные значения:

10Gi

Значение представляет собой целое положительное число, за которым следует единица измерения информации. Между числом и единицей измерения можно поставить пробел. А можно не ставить - писать слитно.

Единицу измерения можно не указывать - оставить одно число. В этом случае это число будет указывать количество байт.

Ниже в таблице указаны доступные единицы измерения:

|  |  |  |  |
| --- | --- | --- | --- |
| Единица измерения | Масштабный коэффициент | Пример | Что пример обозначает |
| (пусто) | 1 | 3500 | 3 500 байтов |
| Ki | 1024 | 17Ki | 17 408 байтов |
| K | 1000 | 17K | 17 000 байтов |
| Mi | 1024 \* 1024 | 21Mi | 22 020 096 байтов |
| M | 1000 \* 1000 | 21M | 21 000 000 байтов |
| Gi | 1024 \* 1024 \* 1024 | 13Gi | 13 958 643 712 байтов |
| G | 1000 \* 1000 \* 1000 | 13G | 13 000 000 000 байтов |
| Ti | 1024 \* 1024 \* 1024 \* 1024 | 21Ti | 23 089 744 183 296 байтов |
| T | 1000 \* 1000 \* 1000 \* 1000 | 21T | 21 000 000 000 000 байтов |

Разделители разрядов (точка или запятая) использовать нельзя. Если Вы хотите указать: 1.3G, то пишите 1300M

В Kubernetes данную переменную рекомендуется задать следующим образом:

# .....
spec:
 containers:
 - name: main
 image: hub.mybpm.kz/mybpm-api
 resources:
 limits:
 memory: "1Gi"
 requests:
 memory: "1Gi"
 # .....
 env:
 - name: MYBPM\_MEMORY\_LIMIT,
 valueFrom: { resourceFieldRef: { containerName: main, resource: limits.memory } }
# ....

Что обозначает чтение её значение из самого ограничения. Это уберёт необходимость указания одного параметра в разных местах. Подробнее об использовании этой переменной смотрите в разделе [Генерация heap-dump-файлов](/040_for_developers/001_install/080-generate-heap-dump-files.md)

### MYBPM\_TELEMETRY\_ENDPOINT

Точка скидывания телеметрии в формате OpenTelemetry по протоколу gRPC

##### Возможные значения:

http://jaeger:4317

Значение представляет собой строку URL, которая указывает на сервер, который принимает телеметрию в формате OpenTelemetry по протоколу gRPC.

Если данная п.о. не указана, то механизм телеметрии будет отключён. В этом случае можно будет смотреть трассы в файлах.

Трассы в файлах можно отключить с помощью конфигурационного файла

/mybpm/configs/LogConfig.txt

параметр:

useLogFiles=true

Если его поставить в false, то файлы трасс перестанут заполняться.

# Настройка Kafka Consumer-ов (кафка-получателей)

Программе сервера можно указать переменную окружения MYBPM\_CONSUMER\_DIR например так:

MYBPM\_CONSUMER\_DIR=cons3

Тогда в среде конфигурирования будет создана директория:

/mybpm/consumers-cons3/

И в этой директории будут созданы файлы конфигурации для всех кафка-получателей, которые активированы на данном сервере.

Далее эта директория будет называться директорией конфигурации кафка-получателей.

Активацией тех или иных кафка-получателей занимаются переменные окружения:

MYBPM\_ALL\_CONSUMERS\_OFF, MYBPM\_ALL\_KAFKA\_NOTIFICATIONS\_OFF

Подробнее про них смотрите в разделе [Переменные окружения](010-01-env-variables.md).

Если переменная окружения MYBPM\_CONSUMER\_DIR не задана, то директорией конфигурации кафка-получателей будет директория:

/mybpm/consumers/

## Структура и значения конфигурационных файлов в директории конфигураций кафка-получателей

Откроем например файл:

BoiEvents.workerCount

В нём будет такое содержимое:

# Генерирует события инстанций БО по их изменениям и кидает их в кафку
boiKafkaToBoiEventKafka = 1

# Парсит события инстанций БО для отображения их на клиенте
applyBoiEventKafkaToMongo = 1

(некоторые строки не показаны).

В этом файле строки, которые начинаются с решётки (#) являются комментарием и компьютер из игнорирует. Они нужны для описания человеку.

Параметр:

boiKafkaToBoiEventKafka = 1

Обозначает, что нужно запустить один поток на каждом сервере, который будет обслуживать данный кафка-продюсер. Что это за продюсер описано в комментарии.

Если единицу поменять на тройку:

boiKafkaToBoiEventKafka = 3

То это обозначает, что на одном сервере для данного кафка-получателя запуститься три потока. Соответственно, если у нас два сервера, то обслуживать данный получатель будут шесть потоков.

Данный кафка-получатель boiKafkaToBoiEventKafka считывает данные с кафка-топика BOI. Если у этого топика имеется 48 партиций, то эти партиции поровну поделятся между всеми шести потоками, т.е. 48/6 = 8 патриций будет обслуживаться каждым потоком.

За соотношением количества партиций и общим количеством потоков нужно следить и не допускать, что количество потоков будет превышено количество партиций. Если это будет превышено, то некоторые потоки будут крутиться вхолостую, не обрабатывая данные, но занимая ресурсы процессора и ОЗУ.

Если случилось так, что количество потоков нужно увеличивать, а партиций уже не хватает, то нужно у кафка-топика увеличивать количество партиций.

## Распределение обязанностей между несколькими серверами MyBPM

Лучшей практикой считается распределение кафка-получателей между серверами MyBPM. По умолчанию сервер MyBPM обслуживает все кафка-получатели. Это может привести к повышенной нагрузке на один сервер, и её нужно распределить между несколькими серверами, для этого нужно уметь настраивать потоки кафка-получателей по-разному на разных серверах. Для этого предназначена переменная окружения MYBPM\_CONSUMER\_DIR, которая может задать индивидуальное расположение конфигурации кафка-получателей. Для этого нужно настроить распределение обязанностей между несколькими серверами MyBPM.

При высокой нагрузке рекомендуется иметь четыре сервера MyBPM, которые будут обслуживать разные обязанности. Давайте придумает им обозначения в соответствии с их обязанностями. А именно:

* api
	+ MYBPM\_CONSUMER\_DIR = api
	+ MYBPM\_ALL\_CONSUMERS\_OFF = true
* cons1
	+ MYBPM\_CONSUMER\_DIR = cons1
	+ MYBPM\_ALL\_KAFKA\_NOTIFICATIONS\_OFF = true
* cons2
	+ MYBPM\_CONSUMER\_DIR = cons2
	+ MYBPM\_ALL\_KAFKA\_NOTIFICATIONS\_OFF = true
* cons3
	+ MYBPM\_CONSUMER\_DIR = cons3
	+ MYBPM\_ALL\_KAFKA\_NOTIFICATIONS\_OFF = true

Здесь api обслуживает запросы пользователей, поэтому на ней отключены кафка-получатели вообще с помощью переменной окружения: MYBPM\_ALL\_CONSUMERS\_OFF = true.

А cons1, cons2, cons3 - обслуживают только кафка-получатели, а запросы пользователей не обслуживают. Поэтому на них отключены напоминания с помощью переменной окружения MYBPM\_ALL\_KAFKA\_NOTIFICATIONS\_OFF = true. Также средствами Kubernetes нужно запретить обращаться к этим серверам по сервисам, иначе предоставляемый функционал будет неполноценным.

В результате данных настроек будут созданы следующие директории в конфигурационной среде:

* /mybpm/consumers-api/
* /mybpm/consumers-cons1/
* /mybpm/consumers-cons2/
* /mybpm/consumers-cons3/

С одинаковыми иерархиями файлов конфигураций кафка-получателей, в которых всем получателям будет прописан один поток. Ну а в результате каждый получатель будет обслуживать три потока из серверов: cons1, cons2, cons3.

Для api тоже будет создана конфигурации, но там будут только оповещения. Их можно оставить без изменений. Оповещения обслуживают пользователей, работающие с платформой через браузер.

И теперь нужно пройтись по всем этим конфигурационным файлам и включить и выключить те или иные потоки.

Рекомендуется сделать следующие настройки (тут указываются настройки только в файлах с расширением .workerCount, остальные конфигурационные файлы будут расписаны ниже):

api - на нём нужно настроить оповещения, следующим образом:

* /mybpm/consumers-api/
	+ KafkaToWebsocket.workerCount
		- kafka\_\_to\_\_websocket = 1
		- kafkaToWebSocketBo = 1
		- kafkaToWebSocketNotification = 1
		- kafkaToWebsocketTelephony = 1
		- kafkaToWebsocketGeoMap = 1
	+ UserNotification.workerCount
		- chatMessageToUserNotification = 1
		- eventToUserNotification = 1

Если будут остальные в этой директории и в поддиректориях, то их нужно поставить в 0.

cons1 - выделить для миграции данных, для этого нужно активировать следующие настройки:

* /mybpm/consumers-cons1/
	+ InMigration.workerCount
		- in\_migration = 1
	+ InMigrationFromKafka.workerCount
		- in\_migration = 1
	+ InMigrationRunProcess.workerCount
		- in\_migration\_run\_process = 1
	+ OutMigration.workerCount
		- out\_migration = 1
	+ OutMigrationFromKafka.workerCount
		- comeFromKafka = 1
	+ OperativeReport.workerCount
		- report\_materialized\_view\_create = 1
	+ OperativeReportStandTables.workerCount
		- operative\_report\_bo = 1
		- report\_materialized\_view\_co = 1
		- operative\_report\_migration = 1
	+ ORRef.workerCount
		- operative\_report\_radio\_button\_ref = 1

Остальные в этой директории нужно поставить в 0.

cons2 - нужно выделить для запуска процессов, для этого нужно активировать следующие настройки:

* /mybpm/consumers-cons2/
	+ run\_process/RunProcess.workerCount
		- runProcess = 1
		- runProcessManualSave = 1
		- runProcessInMigration = 1

Остальные в этой директории нужно поставить в 0.

cons3 - нужно выделить для обработки остальных кафка-получателей. Нужно пройтись по директории

* /mybpm/consumers-cons3/

И во всех перечисленных выше файлах, но лежащих в этой директории, нужно поставить 0. А остальные поставить в 1.

## Другие файлы конфигурации кафка-получателей

Кроме файлов с расширением .workerCount есть ещё файлы с именем params.config - они служат для настройки других параметров кафка-получателей. Этот файл действует на кафка-получателей, которые находятся в той же директории - на другие под-директории и над-директории он не действует.

Например, файл:

/mybpm/consumers-cons3/params.config

Действует на кафка-получатель:

/mybpm/consumers-cons3/OutMigrationFromKafka.workerCount

И не действует на кафка-получатель:

/mybpm/consumers-cons2/run\_process/RunProcess.workerCount

В то же время на этого получателя действует файл:

/mybpm/consumers-cons2/run\_process/params.config

Вот пример файла params.config:

con.auto.commit.interval.ms = 1000
con.fetch.min.bytes = 1
con.max.partition.fetch.bytes = 141943040
con.connections.max.idle.ms = 540000
con.default.api.timeout.ms = 60000
con.fetch.max.bytes = 52428800
con.retry.backoff.ms = 2000
con.session.timeout.ms = 45000
con.heartbeat.interval.ms = 5000
con.max.poll.interval.ms = 3000000
con.max.poll.records = 500
con.receive.buffer.bytes = 65536
con.request.timeout.ms = 30000
con.send.buffer.bytes = 131072
con.fetch.max.wait.ms = 500
out.worker.count = 1
out.poll.duration.ms = 800

В нём параметр: out.worker.count - обозначает количество потоков по умолчанию. А параметр: out.poll.duration.ms - обозначает количество миллисекунд, которые тратятся на ожидание данных с брокера в реализации кафка-получателя на платформе. Значение 800 самое подходящее - вам его никогда, скорее всего, менять не придётся.

Параметры, которые начинаются с префикса: con. можно прочитать в документации к Apache Kafka, только этот префикс нужно убрать, а именно, например нам нужно узнать что значить параметр:

con.retry.backoff.ms = 2000

Убираем префикс con. и получаем параметр:

retry.backoff.ms = 2000

Теперь в документации к Apache Kafka ищем что обозначает этот параметр и находим следующее описание:

<https://kafka.apache.org/documentation/#producerconfigs_retry.backoff.ms>

Так же можно добавить и другие параметры, которых здесь нет, но, при этом, нужно не забыть добавить префикс con..

## Применение изменений файлов конфигурации кафка-получателей

Параметры применяются "на горячую" - это обозначает, что система постоянно отслеживает изменения этих файлов. И как только это произошло, она тут же их применяет - лаг составляет не больше пары секунд.

## Настройка масштабирования кафка получателей

Давайте разберёмся в масштабировании с одним кафка получателем.

Пусть имеется запущенный сервер компонента с помощью yaml-файла, который приведён здесь: [Установка компонента MyBPM Cons1](008-MyBPM-install/040-install-CONS.md)

И пусть мы настраиваем входящую миграцию через кафку. Для неё предусмотрен параметр:

* /mybpm/consumers-cons1/
	+ InMigration.workerCount
		- in\_migration = 1

Из пути конфига consumers-**cons1** мы видим, что с ним работает сервер, у которого определена переменная окружения:

MYBPM\_CONSUMER\_DIR=cons1

В yaml-файле, фрагмент которого изображён ниже.

Фрагмент файла: 30-mybpm-cons1.yaml

 env:
 - { name: MYBPM\_JAVA\_OPTS, value: "-Xmx2Gi -Xms2Gi" }
 - { name: MYBPM\_ALL\_CONSUMERS\_OFF, value: "true" }
 - { name: MYBPM\_CONSUMER\_DIR, value: "cons1" } # <------------------- ВОТ
 - { name: MYBPM\_COMPANY\_CODE, value: "greetgo" }

Так же в этом yaml-файле есть параметр replicas:

Фрагмент файла: 30-mybpm-cons1.yaml

spec:
 selector:
 matchLabels:
 app: mybpm-cons1
 replicas: 1 # <-------------------- ВОТ
 template:
 metadata:

На данный момент in\_migration = 1 и replicas: 1 - это означает, что запущен один сервер, и на этом сервере запущен один поток, который обслуживает данный кафка получатель (входящая миграция данных).

И пусть теперь Ваша задача ускорить эту миграцию. Вначале вы увеличиваете количество потоков, обслуживающие данный получатель, установив:

in\_migration = 16

Нужно поменять конфигурационный файл и сохранить его. Перезапускать сервер не нужно - система обнаружит изменения файла и автоматически запустит новые потоки. А кафка брокеры перестроятся и начнут в данном случае отдавать 16-ти потокам данного получателя.

При увеличении количества потоков, миграция должна ускориться.

Если увеличение количество потоков не приводит к ускорению миграции, например потому что упёрлись в производительность CPU, то нужно увеличить количество подов данного сервера. Для этого в yaml-файле увеличим параметр:

Фрагмент файла: 30-mybpm-cons1.yaml

 replicas: 3

Применив этот файл, Kubernetes запустит три сервера. Каждый сервер прочитает конфигурационный параметр in\_migration = 16 и запустит 16 потоков на обслуживание данного получателя. Итого количество потоков, которые обслуживают данный получатель, станет:

3 \* 16 = 48

Сорок восемь потоков будут обслуживать данный получатель.

Соответственно скорость обработки данного получателя должна тоже увеличиться.

***ВАЖНО***: Кафка устроена так, что одна партиция не может обслуживаться несколькими потоками одновременно. Соответственно, если Вы настроили 50 потоков, а у кафка-топика всего 48 партиций, то получается, что два потока будут крутиться бездействия.

Если нужно увеличивать количество потоков, а партиций мало, то количество партиций тоже нужно увеличивать. Это можно сделать в кафке.

По умолчанию каждому топику даётся 48 партиций. Если этого оказалось недостаточно, то сделайте настройку по умолчанию на 480 топиков и перезапустите кафка брокеры. Изменить значение по умолчанию недостаточно, нужно ещё пройтись по всем кафка топиками и индивидуально у каждого поменять это количество.

# Настройка PostgreSQL

Компоненты MyBPM API и MyBPM Cons должны иметь доступ к БД PostgreSQL. Параметры доступа настраиваются через переменные окружения:

|  |  |  |
| --- | --- | --- |
| П.о. | Пример значения | Описание |
| MYBPM\_AUX1\_HOST | localhost | Хост (домен или IP-адрес) машины, где находиться PostgreSQL |
| MYBPM\_AUX1\_PORT | 5432 | Порт, на котором работает сервер |
| MYBPM\_AUX1\_DB\_NAME | mybpm\_aux1 | Имя базы данных |
| MYBPM\_AUX1\_USER\_NAME | mybpm | Пользователь подключения |
| MYBPM\_AUX1\_PASSWORD | hj34b2j23b4jh4d | Пароль пользователя |

## 1. Дополнительные привилегии

### 1.2. Создание схем

Пользователь mybpm должен иметь возможность создавать схемы

create schema ...

Это нужно для пользователей плагинов - плагинам запрещено иметь доступ к системным данным, поэтому для каждого плагина готовиться своя схема и пользователь, который имеет доступ только к этой схеме.

### 1.3. Создание других пользователей

Пользователь mybpm должен иметь возможность создавать других пользователей для плагина, чтобы ограничить плагины только позволенным им данным.

## 2. Настройка пулов-соединений

Платформа для доступа к БД использует пул-соединений и параметрами этого пула можно управляет с помощью конфигурационного файла. В системе конфигурации он находится по адресу:

/mybpm/configs/Aux1DbDataSourceConfig.txt

Пример этого файла:

# Created at 2024-08-12 10:26:16
# Кофиг для настройки DataSource базы aux1

# Значение максимального времени когда коннекшн может быть в состоянии idle
idleTimeout=600000

# Значение максимального Pool size для коннекшна в базу aux
aux1MaxPoolSize=10

Сервер платформы при старте открывает сразу aux1MaxPoolSize соединений к БД и держит их в состоянии IDLE, чтобы можно было быстро использовать.

**ПРИМЕЧАНИЕ**: нужно быть внимательным, чтобы не превысить максимальное количество соединений. Например, если на сервере PostgreSQL настроено максимальное количество соединений 30, в файле настроено aux1MaxPoolSize=10, а количество запущенных серверов платформы 5 (один api и четыре cons). В результате каждый из пяти серверов пытается открыть 10 соединений, итого серверам нужно 5\*10 = 50 соединений, а в PostgreSQL настроено всего 30 соединений, из-за этого сервера не могут запуститься. В данном случает количество соединений нужно увеличить до 50\*2 = 100. При этом сервера займут 50 соединений, и 50 останется в резерве. В резерве всегда нужно оставлять, например для rollout-restart. Этот вид перезапуска серверов работает так, что вначале запускаются новые сервера, старые продолжают работать и обслуживать трафик. И только когда новые сервера запустились и получили все необходимые им соединения, старые сервера останавливаются.

Чтобы новые сервера могли запуститься нужно иметь резерв соединений в PostgreSQL.

# Компонент MyBPM API

Компонент MyBPM API распространяется как образ контейнера Docker. Примерное имя контейнера следующее:

hub.mybpm.kz/mybpm-api-...

Где вместо многоточия стоит ветка разработки и версия данного компонента. Вы его получите от разработчика.

При запуске, этот компонент вначале проверяет все соединения с базами данных. Если до какой-то базы данных он не может достучаться, то он падает и выдаёт ошибку в логи. По этой ошибке нужно понять причину. Исправить её и запустить компонент заново.

MyBPM API-сервис для своей работы требует следующие базы данных:

1. MongoDB - это основная база данных. Здесь платформа MyBPM хранит оперативную информацию.
2. MongoDB (Files) - здесь хранятся файлы. Можно эту базу-данных совместить с оперативной MongoDB. Но это не рекомендуется, так как характер работы с файлами сильно отличается.
3. Apache Kafka - здесь платформа MyBPM хранит историческую информацию. Также кафка используется как брокер сообщений для синхронизации данных между различными компонентами платформы MyBPM.
4. Apache Zookeeper - здесь платформа хранит свою конфигурацию. Желательно администраторам платформы иметь веб-интерфейс к Zookeeper. Например, ZooNavigator: <https://zoonavigator.elkozmon.com/en/latest/>
5. ElasticSearch - здесь платформа MyBPM хранит оперативную информацию для поиска по ней (в MongoDB для поиска ходить запрещено)
6. PostgreSQL - здесь обрабатываются различные алгоритмы с использованием реляционного механизма.

Требования к БД, к которым подключается компонент MyBPM API смотрите [здесь](030-MyBPM-API-DB-require.md)

### Конфигурация платформы MyBPM

Вся конфигурация платформы MyBPM находиться на сервере ZooKeeper поэтому рекомендуется иметь удобный интерфейс к нему, разработчики рекомендуют ZooNavigator: <https://zoonavigator.elkozmon.com/en/latest/>

Его можно быстро установить через докер: <https://hub.docker.com/r/elkozmon/zoonavigator>

В будущем планируется конфигурацию системы перенести полностью на MongoDB и отказаться от Zookeeper. Это будет сделано абсолютно бесшовно. А управление конфигурацией будет через Web-интерфейс, встроенный в платформу MyBPM.

Файлы конфигурации создаются автоматически со значениями по умолчанию. Эти значения уже настроены для оптимальной работы платформы MyBPM. Но иногда может возникнуть ситуации, что их можно поменять. После изменения этих параметров платформу MyBPM перезапускать НЕ нужно - они применяются на платформе НА ГОРЯЧУЮ.

### Жонглирование подсистемой MyBPM API

MyBPM API содержит в себе подсистемы:

1. REST-API: Подсистема обработки REST-сервисов и Web-сокетов
2. Kafka Consumers: Подсистема обработки сообщений кафки
3. Schedulers: Подсистема выполнения задач по расписанию.

Их можно включать и выключать на конкретном сервере. Тем самым можно распределять нагрузку между серверами, в случае, если один сервер не справляется со всеми задачами. Это напоминает микро-сервисную архитектуру, просто все сервисы сосредоточены в одной программе, и их можно активировать и деактивировать выборочно с помощью переменных окружения.

### Параллельная работа подсистем

Все эти подсистемы реализованы так, чтобы могли работать на нескольких экземплярах параллельно.

Подсистема REST-API реализована по принципу БЕЗ-СОСТОЯНИЯ (state-less), поэтому, если один сервер не справляется с нагрузкой, можно запустить несколько копий и случайным образом выбирать обработчик запросов. Для этого можно использовать балансировщик нагрузки, и он может работать в любых режимах (случайный выбор, выбор менее нагруженного сервере и др. - это зависит от имеющегося ПО по управлению нагрузкой).

Подсистема REST-API всегда активна - её отключение делается недоступностью данного сервера для REST-запросов.

В подсистеме Schedulers все задачи реализованы так, чтобы корректно отрабатывать даже если одновременно запущены на разных серверах. Следует знать, что эти задачи не нуждаются в большом количестве ресурсов, и поэтому их можно держать только на одном сервере. В случае например миграции, то миграция проходит через кафку, а сама задача просто инициирует миграцию заполняя кафку записями с идентификаторами, которые говорят куда смигрировать одну запись. Другие задачи решаются подобным образом.

Подсистему Schedulers на одном сервере можно отключить с помощью переменной окружения:

 MYBPM\_SCHEDULER\_OFF

[Описание переменных окружения здесь](010-01-env-variables.md).

Подсистема Consumers - это набор обработчиков сообщений из кафки из различных топиков. Её распараллеливание решается самой архитектурой кафки. Главное чтобы топики в кафке содержали большое количество партиций. Рекомендуется 48 для малых систем. И 480 - для больших.

Подсистему Consumers на одном сервере можно отключить с помощью переменной окружения:

MYBPM\_ALL\_CONSUMERS\_OFF

[Описание переменных окружения здесь](010-01-env-variables.md).

### Настройка обработчиков в подсистеме Consumers

Конфигурация управляющая обработчиками сообщений из кафки находиться по пути

/mybpm/consumers/...

Этот путь можно поменять с помощью переменной окружения

MYBPM\_CONSUMER\_DIR

[Описание переменных окружения здесь](010-01-env-variables.md).

Тем самым можно распределить настройки одних серверов-consumer-ов в одном месте, а других - в другом. Это позволяет распределять обработку различных ТИПОВ сообщений из кафки по разным серверам.

Рассмотрим пример. Есть файл конфигурации:

/mybpm/consumers/BoiEvents.workerCount

В нём содержится следующий текст:

#
# Created at 2023-10-23 16:26:40
#
# Генерирует события инстанций БО по их изменениям и кидает их в кафку
boiKafkaToBoiEventKafka = 1
#
# Appended at 2023-10-23 16:26:40
#
# Парсит события инстанций БО для отображения их на клиенте
applyBoiEventKafkaToMongo = 1

Параметр boiKafkaToBoiEventKafka соответствует одному обработчику, который "Генерирует события..." - об этом сказано в комментарии к параметру выше. Цифра у этого параметра обозначает количество потоков, которые обрабатывают данные сообщения.

Например, Вы хотите, чтобы на сервере S1 обрабатывались только boiKafkaToBoiEventKafka в три потока, а на сервере S2 обрабатывались только applyBoiEventKafkaToMongo в четыре потока.

Для этого на сервере S1 установите переменную окружения MYBPM\_CONSUMER\_DIR=s1, а на сервере S2 установите переменную окружения MYBPM\_CONSUMER\_DIR=s2. После запуска этих серверов платформа создаст следующие конфиги:

/mybpm/consumers-s1/BoiEvents.workerCount
boiKafkaToBoiEventKafka = 1
applyBoiEventKafkaToMongo = 1

И:

/mybpm/consumers-s2/BoiEvents.workerCount
boiKafkaToBoiEventKafka = 1
applyBoiEventKafkaToMongo = 1

Здесь комментарии убраны, а первая строка - это путь к файлу.

Вам нужно изменить эти конфиги следующим образом:

/mybpm/consumers-s1/BoiEvents.workerCount
boiKafkaToBoiEventKafka = 3
applyBoiEventKafkaToMongo = 0

И:

/mybpm/consumers-s2/BoiEvents.workerCount
boiKafkaToBoiEventKafka = 0
applyBoiEventKafkaToMongo = 4

После сохранения этих файлов перезапускать сервера НЕ НУЖНО - платформа автоматически узнает об их изменении и обновит своё поведение.

Теперь будет работать всё так, как Вы запланировали.

Учтите, что платформа MyBPM создаст не только этот конфигурационный файл, но и другие файлы тоже, и Вам нужно их тоже настроить соответствующим образом.

### Настройка расписания в подсистеме Schedulers

Расписания в подсистеме Schedulers тоже вынесены в конфигурационный файл. Конфигурация расписаний находиться по пути:

/mybpm/scheduler/core/...

Для ядра системы, и по пути:

/mybpm/scheduler/plugin/<plugin-id>/...

Для плагинов.

Эти файлы тоже создаются платформой автоматически и заполняются расписанием по умолчанию. Так же в эти конфиги добавлено описание как самих задач с расписанием, так и формата расписания.

Например, есть файл:

/mybpm/scheduler/core/BoProcessScheduler.scheduler-config.txt
executeProcesses = repeat every 10 min

(Здесь комментарии убраны, а первая строка - это путь к файлу.)

И Вам нужно увеличить интервал запуска этой задачи до тридцати минут. Тогда Вам нужно написать:

/mybpm/scheduler/core/BoProcessScheduler.scheduler-config.txt
executeProcesses = repeat every 30 min

И, после сохранения, платформа MyBPM автоматически идентифицирует изменение файла и обновит данное расписание. Если же Вы допустите ошибку в формате расписания, то платформа отключит запуск данной задачи, и создаст файл:

/mybpm/scheduler/core/BoProcessScheduler.scheduler-config.txt.error

В котором опишет данную ошибку. Вы её исправите, и после сохранения этот файл удалиться сам, что обозначает, что никаких ошибок нет, и расписание принялось к запуску задачи.

Так же Вам может понадобиться отключить вообще запуск данной задачи, для этого Вам нужно поставить символ решётки, вот так:

/mybpm/scheduler/core/BoProcessScheduler.scheduler-config.txt
executeProcesses = # repeat every 30 min

И тогда запуск этой задачи будет отменён.

УЧТИТЕ, решётка должна стоять ПОСЛЕ знака равно.

# Требования к настройке баз данных со стороны платформы MyBPM

## 1. Требования к настройке Apache Kafka

Необходимо чтобы пользователь, из-под которого платформа присоединяется к кластеру Apache Kafka, имел следующие возможности:

### 1.1. Автоматическое создание топиков

На каждом брокере кластера Apache Kafka нужно настроить автоматическое создание топиков.

Данную настройку нужно делать на каждом брокере с помощью параметра (нужно установить его в true):

auto.create.topics.enable=true

### 1.2. Множественное количество партиций

Необходимо настроить множественное количество партиций при создании топика. Количество партиций настраивается в зависимости от объёма данных, которые будут использоваться в системе.

Таблица рекомендаций количества партиций в автоматически создаваемом топике:

|  |  |
| --- | --- |
| Оценка количества данных | Рекомендуемая величина |
| Мало данных | 12 |
| Среднее кол-во данных | 48 |
| Большое кол-во данных | 480 |
| Огромное кол-во данных | 4800 |

Данную настройку нужно делать на каждом брокере с помощью параметра:

num.partitions=48

### 1.3. Нужно настроить топик \_\_offsets

Нужно настроить множественные партиции и фактор репликации на топик \_\_offsets.

Данную настройку нужно делать на каждом брокере с помощью параметров:

offsets.topic.num.partitions=48
offsets.topic.replication.factor=3

### 1.5. Фактор репликации на топики, создаваемые по-умолчанию

Нужно настроить фактор репликации на топики, создаваемые по-умолчанию. Для этого на каждый брокер нужно установить следующие настройки:

transaction.state.log.replication.factor=3
default.replication.factor=3
transaction.state.log.min.isr=2

### 1.6. Очистку топиков по-умолчанию необходимо выключить

Для этого необходимо установить следующие настройки на каждом брокере:

log.retention.hours=-1

Минус один обозначает, что топики, созданные автоматически, самостоятельно очищаться не будут

## 2. Требования к настройке Elastic Search

### 2.1. Разрешение на создание шаблонов индексов с шаблонами проекций полей (mapping)

Пользователь, который подключается базе данных Elastic Search со стороны платформы MyBPM, будет создавать шаблоны индексов. При том, эти шаблоны индексов могут содержать шаблоны проекций полей (mapping). Следовательно, у этого пользователя должны быть права для этого.

## 3. Требования к настройке PostgreSQL

Смотрите в другом разделе: [Настройка PostgreSQL](010-20-config-postgres.md)

## 4. Требования к настройке MongoDB

### 4.1. Автоматическое создание баз-данным

Платформа создаёт различное количество баз-данных, и их имена генерируются динамически, поэтому необходимо, чтобы пользователь, из-под которого платформа подключается к MongoDB имел возможность создавать новые базы данных и коллекции в них.

### 4.2. Создание индексов

Для ускорения работы некоторых аспектов системы, необходимо чтобы можно было создавать индексы на любые коллекции в любой базе данных.

# Компонент MyBPM WEB

Компонент MyBPM WEB распространяется как образ контейнера Docker. Примерное имя контейнера следующее:

out.mybpm.kz/mybpm-web-...

Где вместо многоточия стоит ветка разработки и версия данного компонента. Вы его получите от разработчика.

### Описание функционала

Компонент MyBPM WEB состоит из nginx-сервера, который отдаёт статический текст html, css и js. А также обеспечивает проксирование REST-запросов MyBPM API через себя.

Ниже приводиться Dockerfile:

FROM nginxinc/nginx-unprivileged:1.25
ARG projectName
ENV NGINX\_ENVSUBST\_OUTPUT\_DIR /etc/nginx
COPY ./src/nginx.conf.template /etc/nginx/templates/
COPY ./dist/ /app
WORKDIR /app
EXPOSE 8000

Ниже приводиться конфигурационный файл nginx.conf.template:

error\_log /var/log/nginx/error.log error;
pid /var/cache/nginx/nginx.pid;

worker\_processes 6;

events {
 worker\_connections 1024;
}

http {
 proxy\_read\_timeout 9500s;
 include mime.types;
 default\_type application/octet-stream;
 sendfile on;
 keepalive\_timeout 65;
 charset utf-8;
 underscores\_in\_headers on;

 log\_format main '$remote\_addr - $remote\_user [$time\_local] "$request" '
 '$status $body\_bytes\_sent "$http\_referer" '
 '"$http\_user\_agent" "$http\_x\_forwarded\_for"';

 access\_log /var/log/nginx/access.log main;

 server {
 client\_max\_body\_size 2000m;
 listen 8000 default\_server;

 root /app/;
 index index.html;

 server\_tokens off;

 location /web/ {
 proxy\_pass http://${MYBPM\_API\_HOST}:${MYBPM\_API\_PORT}/web/;
 proxy\_set\_header X-Real-IP $remote\_addr;
 proxy\_set\_header X-Forwarded-For $proxy\_add\_x\_forwarded\_for;
 proxy\_set\_header X-Forwarded-Proto https;
 proxy\_set\_header Authorization $http\_authorization;
 }

 location /api/ {
 proxy\_pass http://${MYBPM\_API\_HOST}:${MYBPM\_API\_PORT}/api/;
 proxy\_set\_header X-Real-IP $remote\_addr;
 proxy\_set\_header X-Forwarded-For $proxy\_add\_x\_forwarded\_for;
 proxy\_set\_header X-Forwarded-Proto https;
 proxy\_set\_header Authorization $http\_authorization;
 }

 location /websocket {
 proxy\_pass http://${MYBPM\_API\_HOST}:${MYBPM\_API\_PORT}/websocket;
 proxy\_set\_header Upgrade $http\_upgrade;
 proxy\_set\_header Connection "upgrade";
 proxy\_pass\_header Sec-Websocket-Extensions;

 proxy\_buffering off;

 proxy\_set\_header X-Real-IP $remote\_addr;
 proxy\_set\_header X-Forwarded-For $proxy\_add\_x\_forwarded\_for;
 proxy\_set\_header X-Forwarded-Proto https;
 }

 location / {
 alias /app/;

 expires -1;
 add\_header Pragma "no-cache";
 add\_header Cache-Control "no-store, no-cache, must-revalidate, post-check=0 pre-check=0";

 location ~ ^/(assets|bower\_components|scripts|styles|views) {
 expires 31d;
 add\_header Pragma "";
 add\_header Cache-Control "public";
 }

 location ~ \.(css|js|svg|woff|ico)$ {
 expires 31d;
 add\_header Pragma "";
 add\_header Cache-Control "public";
 }

 try\_files $uri $uri/ /index.html =404;
 }

 location /health {
 access\_log off;
 return 200 "OK";
 add\_header Content-Type "text/plain";
 }
 }

}

Как видите из конфигурационного файла, для того, чтобы подсоединить MyBPM WEB к MyBPM API необходимо определить две переменные окружения:

MYBPM\_API\_HOST - хост, на котором находиться MyBPM API

И:

MYBPM\_API\_PORT - порт, который обслуживает MyBPM API на указанном выше хосту

### Первый вход в систему

Когда платформа только установлена, то, чтобы с ней начать работать, в неё нужно войти. Подробнее смотрите: [Первый вход в систему](050-First-Enter.md)

# Первый вход в систему

Первый раз в систему нужно войти под пользователем с именем:

root

Пароль, нужно посмотреть в базе данных MongoDB.

Он находится в поле:

mybpm\_aux.PersonPassword.initPassword

(**ПРИМЕЧАНИЕ**: база данных mybpm\_aux может иметь префикс, который указан в переменной окружения MYBPM\_APP\_PREFIX)

Чтобы посмотреть эту таблицу, нужно подключиться к базе данных MongoDB и выполнить следующие команды:

use mybpm\_aux
db.PersonPassword.find({initPassword:{$exists:1}})

В терминал распечатается что-то типа:

{ "\_id" : ObjectId("86aad77ffa0bb982d56e4623"), "initPassword" : "root : SohLlsMS5q" }

SohLlsMS5q - это и есть пароль пользователя root.

После использования этого пароля, его необходимо тут же поменять, чтобы этот пароль стал не рабочим. Это нужно для информационной безопасности.

Все пароли платформа MyBPM хранит в зашифрованном виде, без возможности подбора. Поле initPassword используется однократно для возможности первого входа в систему.

Так же следует знать, что когда в платформе MyBPM создают новый аккаунт, то автоматически создаётся корневой пользователь в этом аккаунте. Имя этого пользователя создаётся следующее:

root@<код-аккаунта>

Например, Вы создали аккаунт с кодом hello, то в нём создастся корневой пользователь с именем root@hello.

После создания этого пользователя платформа MyBPM автоматически создаст для него случайный пароль, и поместит его в поле mybpm\_aux.PersonPassword.initPassword с соответствующей меткой. Администратор системы может посмотреть этот пароль и зайти под ним. После этого этот пароль нужно поменять на другой, чтобы в БД не осталось открытого действующего пароля.

#### LiquidMongo застряла

Платформа использует LiquidMongo для настройки структуры базы данных. Данная библиотека работает так: вначале она блокирует свой доступ к изменениям. Потом вносит изменения. Потом сбрасывает блокировку.

Если же при старте блокировка уже установлена, то LiquidMongo будет ждать, когда эта блокировка будет снята. Может случиться, что эту блокировку ни кто не снимет и тогда её нужно будет снять вручную. Для этого необходимо зайти на сервер, где установлена MongoDB и запустить команду:

mongo

Тем самым запуститься интерактивный терминал для работы с MongoDB. Вам нужно воспользоваться следующими командами.

show databases - посмотреть все базы данных
use <база\_данных> - выбрать текущую базу данных
show table - посмотреть все таблицы в текущей базе данных

Для снятия блокировки, Вам нужно выполнить следующие команды:

use mybpm

db.LiquidMongoUpdate.update({"\_id": "LOCKER"}, {$unset: {lockId: 1}});

Т.е. очистить колонку lockId в таблице LiquidMongoUpdate в строке с идентификатором LOCKER в базе данных mybpm.

База данных может иметь префикс, если установлена переменная окружения: MYBPM\_APP\_PREFIX.

# Восстановление реплики MongoDB из директории БД

### Введение

Логика восстановления следующая:

01 - Помещаем спасённую папку БД в отдельное место;

02 - Запускаем в этом месте простой процесс mongod указывающий на спасённую папку;

03 - Подключаем к этому процессу клиент mongo;

04 - Смотрим, что там всё есть (есть необходимые базы, в них есть нужные таблицы с нужным содержимым);

05 - И если всё ок, удаляем от туда информацию о связанности с бывшим кластером (несколько команд);

06 - Корректно останавливаем mongod;

07 - Полученную директорию помещаем на место первой реплики шарда;

08 - Запускаем первую реплику шарда;

09 - Инициируем её;

10 - Запускаем другие реплики с пустыми БД;

11 - Подсоединяем их к первой;

12 - Наблюдаем процесс репликации данных из первой реплики в остальные;

13 - подключаем шард к кластеру в mongos кластера.

### Подготовка спасённой папки

Пусть спасённая папка называется **mongo-data-1-0** - и пусть она раньше была подсоединена к шарду с именем **rs1** .

01 - Кладём её в текущую папку и запускаем команду

docker run --name=asd --rm -it -v $PWD/mongo-data-1-0:/data/db mongo:4.2.0 mongod --dbpath /data/db

Должны появиться логи запущенного сервера.

\*Примечание: asd - замените на другое имя, если это уже где-то используется и далее вместо asd используется то, которое вы исользовали здесь

02 - Подключаемся к этому серверу из другой консоли:

02.1 - Ищем докер-контэйнер командой

docker ps | grep asd

02.2 - Теперь заходим в контейнер:

docker exec -it ИД\_КОНТЭЙНЕРА mongo

Должно появиться приглашение от mongo.

03 - Смотрим содержимое директории и удостоверяемся что там всё есть - полезные команды для этого:

show databases;
use ИМЯ\_БД
show tables
db.ИМЯ\_ТАБЛИЦЫ.count()

04.1 - Теперь удаляем БД local командами

use local
db.dropDatabase()

Должно появиться сообщение об удачном удалении:

{ "dropped" : "local", "ok" : 1 }

04.2 - Теперь заходим в БД admin командой:

use admin

04.3 - И смотрим содержимое таблицы system.version командой:

db.system.version.find({})

Если появился документ с идентификатором "\_id" : "shardIdentity", то его надо удалить. Если такого документа нет, то пропускаем следующий шаг.

04.4 - Удалите shardIdentity командой:

db.system.version.remove({"\_id":"shardIdentity"});

Должно появиться сообщение об успешном удалении:

WriteResult({ "nRemoved" : 1 })

05 - Теперь останавливаем сервер командой

use admin
db.shutdownServer()

### Поднятие нового шарда

01 - Далее копируем подготовленную спасённую папку на место реплики шарда и запускаем только один сервер реплики - другие пока не запускаем.

Команды для запуска первой реплики должны быть примерно такими

mongod --replSet=rs1 --shardsvr --dbdata=/data/db

dbdata - можно не указывать. Здесь важен параметр replSet.

02 - Теперь надо подключиться к этой реплике

Например, можно использовать команду:

kubectl -n mybpm exec -it mongo-data-1-0-0 -- mongo

Ваша команда может быть другой.

Должно появиться приглашение от mongo c одним пустым знаком 'больше' - > . Это обозначает, что данная реплика не инициирована.

03 - Теперь надо инициировать реплику командой

var cfg = {
 "\_id": "rs1",
 "protocolVersion": 1,
 "members": [
 {
 "\_id": 1,
 "host": "mongo-data-1-0-0.mongo-data-1-0:27017"
 }
 ]
};
rs.initiate(cfg, {force: true});
rs.reconfig(cfg, {force: true});

Теперь нужно выйти и зайти снова - эта реплика должна стать PRIMARY, что свидетельствует приглашение:

rs1:PRIMARY>

Далее запускаем другие пустые реплики

И подсоединяем их к этой реплики командами:

rs.add( { host: "mongo-data-1-1-0.mongo-data-1-1:27017", priority: 0, votes: 0 } )

rs.add( { host: "mongo-data-1-2-0.mongo-data-1-2:27017", priority: 0, votes: 0 } )

Посмотреть состояние шарда можно командой

rs.status()

Эта команда должна показать одну реплику в статусе PRIMARY а остальные в статусе SECONDARY. По началу некоторое время ранее пустые реплики могут быть в статусе STARTUP2 - это означает, что на них идёт копирование данных. При завершении копирования статус этой реплики смениться на SECONDARY.

После того как добавленные реплики перейдут в состояние SECONDARY им нужно поменять priority:=1, votes:=1 для этого нужно в переменную скопировать текущую конфигурацию.

asd = rs.conf()

Потом в этой переменной поменять эти значения:

asd.members[1].priority = 1;
asd.members[2].priority = 1;
asd.members[1].votes = 1;
asd.members[2].votes = 1;

Ну и посмотреть значение этой переменной:

asd

Если всё нормально (у всех members значения для priority и votes должно быть равно 1).

Теперь нужно применить настройки из этой переменной:

rs.reconfig(asd)

Данные действия нужно делать на реплике с состоянием PRIMARY

# Преобразование идентификаторов

На платформе MyBPM используется два представления одного и того же идентификатора:

(1. В верблюжьем регистре (camel-case), например: KNSgGaAa1eV6mKkY

(2. В шестнадцатеричном представлении, например: 28d4a019a01ad5e57a98a918

Они оба формируют одно и тоже бинарное представление идентификатора. Для их преобразования существует специальная форма, которая находиться в интерфейсе в "Настройки" -> "IN Миграция" -> "Конвертация":



В поле загружает одно из представлений, нажимаем кнопку, получаем сразу два.



# Пере-индексация эластика для конкретного БО (update all boi)

Чтобы переиндексировать эластик для конкретного БО, (синхронизировать MongoDB с эластиком для конкретного БО) нужно

(1. узнать его идентификатор в camel-case

KNSgGaAa1eV6mKkY

(2. Вызвать REST-запрос

POST {MYBPM-API}/inner/elastic/update-all-boi-batch?boId=KNSgGaAa1eV6mKkY

Пример вызова через curl

curl -X POST http://localhost:1313/inner/elastic/update-all-boi-batch?boId=KNSgGaAa1eV6mKkY

Тут рассказано как можно преобразовывать идентификаторы [Преобразование идентификаторов](030-id-converting.md)

Примечание: {MYBPM-API} - это доступ к самому API-компоненту, а НЕ к Web-компоненту - через него закрыт.

# Удаление записи в MongoDB по идентификатору

Для этого нужно знать шестнадцатеричные идентификаторы для бизнес-объекта (БО или BO) и его инстанции BOI.

Например boId = 67511932d4f50b455ef1b4d2
 boiId = 5516b84fa2242a344ea1b2b3

Идентификатор БО можно посмотреть в студии, как указано на рисунке ниже:



Здесь можно посмотреть как преобразовывать идентификаторы [Преобразование идентификаторов](030-id-converting.md)

Для удаление записи нужно:

1] зайти в утилиту mongo (по старому) или mongosh (по новому)

2] выбрать базу данных mongo\_boi

use mongo\_boi

Если установлен префикс, то нужно его указать тоже

use v1\_1\_mongo\_boi

Можно посмотреть какие есть базы данных командой

show dbs

Потом можно посмотреть какие таблицы есть в этой БД командой:

show tables

Для удаления записи нужно выполнить команду:

db.boi\_67511932d4f50b455ef1b4d2.deleteOne({ \_id: ObjectId("5516b84fa2242a344ea1b2b3") })

Эта команда соответствует примеру выше - в Вашем случае будут другие идентификаторы БО и БОИ.

В это команде:

db - это ссылка на текущую базу данных, выбранную командой use ранее.

db.boi\_67511932d4f50b455ef1b4d2 - ссылка на коллекцию соответствующую БО.

deleteOne - функция удаления по условию

{ \_id: ObjectId("5516b84fa2242a344ea1b2b3") } - условие, которое соответствует записи с указанным идентификатором.

## Kubernetes

Как можно запустить эту команду, если MongoDB установлена в кластере Kubernetes.

CMD='b.boi\_67511932d4f50b455ef1b4d2.deleteOne({ \_id: ObjectId("5516b84fa2242a344ea1b2b3") })'
echo $CMD > \_\_\_cmd.txt
kubectl -n tst-ns exec -i tst-mongo-pod-0 -- mongo < \_\_\_cmd.txt
#or
# kubectl -n tst-ns exec -i tst-mongo-pod -- mongosh < \_\_\_cmd.txt

Здесь:

tst-ns - это имя пространства имён (namespace) в котором установлен сервер MongoDB - подставьте своё

tst-mongo-pod-0 - это имя пода в котором запущен сервер MongoDB - подставьте своё

mongo или для новой mongosh - терминал MongoDB для управления базой данных

# Удаление записи в ElasticSearch по идентификатору

Для этого нужно знать шестнадцатеричные идентификаторы для бизнес-объекта (БО или BO) и его инстанции BOI.

Например boId = 67511932d4f50b455ef1b4d2
 boiId = 5516b84fa2242a344ea1b2b3

Идентификатор БО можно посмотреть в студии, как указано на рисунке ниже:



Для удаления записи нужно вызвать Rest-сервис:

DELETE http://{ip:port}/boi18\_67511932d4f50b455ef1b4d2/\_doc/5516b84fa2242a344ea1b2b3

В этом запросе:

{ip:port} - хост и порт сервера эластика.

boi18\_67511932d4f50b455ef1b4d2 - имя индекса в эластике: 'boi18\_' - всегда такой префикс, '67511932d4f50b455ef1b4d2' - идентификатор БО

/\_doc/ - это тоже всегда так - обозначает, что мы обращаемся к записи по идентификатору

5516b84fa2242a344ea1b2b3 - идентификатор инстанции данного БО

Вызвать этот сервис можно с помощью curl:

curl -X DELETE http://elastic:9200/boi18\_67511932d4f50b455ef1b4d2/\_doc/5516b84fa2242a344ea1b2b3

# Физическое удаление БОИ

В интерфейсе можно настроить физическое удаление инстанций БО



Сам процесс удаления будет запускаться по расписанию, которое можно настроить в конфиге:

scheduler/core/BoiDeleteScheduler.scheduler-config.txt
 executeDeletion = 23:00

По умолчанию, настроено удаление на 23:00

# Перекинуть данные из MongoDB в кафку для БО по казанным полям

Чтобы перекинуть данные из MongoDB в кафку необходимо вызвать сервис

POST {MYBPM-API}/inner/elastic/all-boi-to-kafka?boId=KNSgGaAa1eV6mKkY&field-codes=CODE1,CODE2

В этом сервисе нужно указать идентификатор БО. В этом примере это KNSgGaAa1eV6mKkY также можно передавать расширенный идентификатор БО, например

POST {MYBPM-API}/inner/elastic/all-boi-to-kafka?boId=28d4a019a01ad5e57a98a918&field-codes=CODE1,CODE2

В параметре: field-codes указываются кода полей, значения которых будут уходить в кафку.

Так же можно указать идентификаторы полей с помощью параметра field-ids, например так:

POST {MYBPM-API}/inner/elastic/all-boi-to-kafka?boId=28d4a019a01ad5e57a98a918&field-ids=Imm2CtGcwmCRvVNE,~v0vbu3ouDBC6mej

Сервисы можно вызывать с помощью curl например так:

curl -X POST http://localhost:1313/inner/elastic/all-boi-to-kafka?boId=28d4a019a01ad5e57a98a918&field-ids=Imm2CtGcwmCRvVNE,~v0vbu3ouDBC6mej

Тут рассказано как можно преобразовывать идентификаторы [Преобразование идентификаторов](030-id-converting.md)

Примечание: {MYBPM-API} - это доступ к самому API-компоненту, а НЕ к Web-компоненту - через него закрыт.

## Генерация API-токена без интерфейса

Если через интерфейс не доступна возможность генерации API\_TOKEN, то это можно сделать с помощью REST-API вызовов.

### Поиск идентификатора пользователя

Вначале нужно узнать идентификатор пользователя, для которого необходимо сгенерировать API\_TOKEN.

Вариант, через реестр пользователей:

1. Зайдите в реестр пользователей, и найдите того пользователя, для которого нужно получить API\_TOKEN.
2. Откройте окно разработчика (F12) в браузере. И перейдите в список запросов. Установите фильтр Fetch/XHR.
3. Откройте карточку данного пользователя.
4. В списке появятся различные запросы на сервер.
5. Найдете запрос create-draft и выделите его
6. Откройте исходящие параметры (Payload)
7. В исходящих параметрах будет JSON и по пути params\_Lr1oSgwPR8.boiId - вы увидите идентификатор пользователя.

### Создание API\_TOKEN по идентификатору пользователя

Данное действие делать нужно только в том случае, если для данного пользователя ранее не создавался API\_TOKEN.

Также, если вы хотите стереть прежний API\_TOKEN и сгенерировать новый (например потому что старому вы уже не доверяете).

При генерации нового API\_TOKEN-а старый перестаёт аутентифицировать, и если у Вас он используется для каких-либо API вызовов, то они все перестанут работать. Будьте внимательным.

Для создания нового API\_TOKEN нужно вызвать сервис:

POST /web/v2/person/generate-api-token

{
 "useParamsFromBody": true,
 "params\_Lr1oSgwPR8": {
 "personId": "bgLQ8psUtTWbSty2"
 },
 "body\_o1nhHUG480": {}
}

В заголовок этого запроса нужно положить параметр с именем Token. Значение его можете взять из вызова create-draft из такого, же параметра. А вместо "bgLQ8psUtTWbSty2" подставьте нужный идентификатор пользователя.

Данный сервис сгенерирует API\_TOKEN для указанного пользователя и вернёт его:

"a27d599b4b2064b6580d0431"

Кавычки надо убрать.

### Получение текущего API\_TOKEN по идентификатору пользователя

Если пользователь уже имеет API\_TOKEN, то его можно получить с помощью вызова сервиса:

POST /web/v2/person/load-api-token

{
 "useParamsFromBody": true,
 "params\_Lr1oSgwPR8": {
 "personId": "bgLQ8psUtTWbSty2"
 },
 "body\_o1nhHUG480": {}
}

Данный сервис загрузит API\_TOKEN для указанного пользователя и вернёт его:

"a27d599b4b2064b6580d0431"

Кавычки надо убрать.

Если вернётся пусто, то у данного пользователя нет API\_TOKEN - его нужно сгенерировать сервисом описанным выше.

### Удаление API\_TOKEN по идентификатору пользователя

Если API\_TOKEN для данного пользователя создан по ошибке, или уже не нужен, то его можно удалить вызвав сервис:

POST /web/v2/person/remove-api-token

{
 "useParamsFromBody": true,
 "params\_Lr1oSgwPR8": {
 "personId": "bgLQ8psUtTWbSty2"
 },
 "body\_o1nhHUG480": {}
}

Код 200 обозначит факт удаления.

# Генерация heap-dump-файлов

На платформе предусмотрена возможность генерации heap-dump-файлов при условии, если программа превысила потребляемую память больше определённого процента от предоставленного ей. Это может потребоваться для анализа утечек по памяти, а так же других данных.

Для настройки генерации heap-dump-файлов необходимо:

1. подготовить директорию, куда будут выгружаться эти файлы. Эта директория должна быть доступна программе платформы MyBPM на запись.
2. настроить параметры генерации в конфиге /mybpm/configs/HeapDumpConfig.txt
3. прописать переменную окружения MYBPM\_MEMORY\_LIMIT

После этого можно будет просматривать сгенерированные файлы и выгружать их с помощью REST-сервисов.

## 1. Подготовка директории для выгрузки

Так как программа размещается в поде Kubernetes, то директорию нужно подготовить внутри пода. Директория должна быть подсоединяемым volume-ом. Иначе при падении программы все файлы будут стёрты и станут недоступны для анализа.

Если директория логов уже имеет свой volume, то можно использовать этот же volume.

Платформа, по умолчанию, логи выгружает в директорию /var/log/mybpm. Эту директорию можно сменить переменной окружения: MYBPM\_LOG\_DIR. Если эта переменная окружения не определена, то директория будет по умолчанию.

Рекомендуется создать volume и подключить его по пути /var/log/mybpm внутри пода.

Платформа будет создавать логи внутри этой директории, и параллельно с ними можно разместить директорию для выгрузки файлов heap-dump-ов. Пусть это будет директория:

/var/log/mybpm/heap-dumps

Эту директорию нужно прописать в файле конфигурации генерации heap-dump-файлов.

## 2. Настройка конфига

В системе конфигурации по пути:

/mybpm/configs/HeapDumpConfig.txt

Настраивается конфиг генерации heap-dump-файлов.

### 2.1. Настройка имени файла выгрузки

В файле конфигурации нужно прописать префикс heap-dump-файлов следующим образом:

heapDumpFilePrefix=/var/log/mybpm/heap-dumps/DreamSkies-cons3

Префикс состоит из директории, которую мы выбрали в предыдущем разделе этого документа. А дальше идёт префикс имени будущего сгенерированного файла.

Вендор настойчиво рекомендует использовать префикс состоящий из двух частей:

имя Клиента - тире - имя сервера программы

В данном примере имя Клиента - это DreamSkies, а cons3 - это имя сервера. Судя по имени сервера на нём запускаются Kafka Consumer-ы.

Данный префикс имени необходим вендеру, чтобы быстро понять от куда файл и с какого сервера. Иначе сопровождение может затянуться.

### 2.2. Активация процесса выгрузки

Также необходимо активировать сам процесс выгрузки, для этого в конфигурационном файле нужно прописать параметр:

enabled=true

После чего система поймёт, что всё настроена и будет создавать файлы. Но если не настроена переменная окружения MYBPM\_MEMORY\_LIMIT, то файлы создаваться не будут. В следующем разделе это описано.

### 2.3. Остальные параметры конфига

Остальные параметры конфига можно оставить без изменений.

Дополнительно лишь уточним, что в параметре overflowRamPercent указывается процент занимаемой памяти программы от ограничения наложенные на программу. Файлы будут генерировать только в том случае, если этот процент будет превышен.

## 3. Настройка переменной окружения MYBPM\_MEMORY\_LIMIT

Чтобы сгенерировать файл программа должна понять, что это нужно сделать - просто так генерировать файл расточительно. Для этого ей нужно узнать превышен ли установленный процент занимаемой системой памяти. Для того чтобы рассчитать этот процент, ей нужно знать ограничение по памяти наложенные kubernetes на программу. Для этого программе нужно сообщить с помощью переменной окружения MYBPM\_MEMORY\_LIMIT.

Пример yaml-файла настройки deployment для MyBPM-API:

apiVersion: apps/v1
kind: Deployment
metadata:
 name: mybpm-api
 labels:
 app: mybpm-api
 namespace: test
spec:
 selector:
 matchLabels:
 app: mybpm-api
 replicas: 1
 template:
 metadata:
 labels:
 app: mybpm-api
 app-group: mybpm-api
 spec:
 containers:
 - name: main
 image: hub.mybpm.kz/mybpm-api
 imagePullPolicy: Always
 resources:
 limits:
 memory: "1Gi"
 requests:
 memory: "1Gi"
 ports:
 - containerPort: 8080
 env:
 - name: MYBPM\_MEMORY\_LIMIT,
 valueFrom: { resourceFieldRef: { containerName: main, resource: limits.memory } }
 volumeMounts:
 - { name: log-dir, mountPath: "/var/log/mybpm" }
 volumes:
 - name: log-dir
 hostPath:
 path: /native/path/to/volume

В этом примере указана MYBPM\_MEMORY\_LIMIT, которая считывает значение limit.memory у контейнера с именем main, который определён чуть выше по коду.

Тем самым программе передаётся наложенное на неё ограничение по памяти.

После того как переменная окружения определена программа начнёт генерировать файлы (в случае превышение процента).

В нашем случае автоматически будет создана директория:

 /var/log/mybpm/heap-dumps/

И в ней будут появляться файлы с именем, которе начинается на DreamSkies-cons3- и заканчивается разрешением .hprof. Так же в имя файла будет добавлена текущая дата и время.

## 4. Предоставление файлов

Сгенерированные файлы нужно передать вендеру для анализа не меняя его имя, так как в нём содержится важная информация.

Это можно сделать средствами администрирования volume-ов, которые Вы используете в своей инфраструктуре.

Если же это сделать затруднительно, то для этого подготовлены REST-сервисы получения сгенерированных файлов.

Вообще есть команда

kubectl cp

Которая предназначена для копирования файлов с пода, но для больших файлов она почему-то не работает. Поэтому есть два REST-сервиса для просмотра и скачивания файлов.

Эти сервисы не доступны из-вне поэтому вначале нужно получить доступ к самим контейнерам. Это можно сделать с помощью командны:

 kubectl post-forward -n NS mybpm-api-7f765bb5c5-6dxwf 13000:8080

В этой команде:

* NS замените на имя namespace, которое Вы используете.
* mybpm-api-7f765bb5c5-6dxwf замените на имя пода, которое у вас. Посмотреть доступные под-ы можно командой:
* kubectl get pod --all-namespaces
* 13000 - это свободный порт на вашей терминальной машине. Можете его поменять на любой свободный.
* 8080 - не меняйте - это порт, который обслуживает сервер программы.

После того как эта команда запустилась, сервер MyBPM-API становиться доступен по

 localhost:13000

И можно вызывать REST-сервисы.

Просмотр доступных файлов можно сделать командой:

 curl http://localhost:13000/inner/hot-spot/list-available-heap-dump-files | jq

На выходе можно получить что-то вроде

 % Total % Received % Xferd Average Speed Time Time Time Current
 Dload Upload Total Spent Left Speed
100 238 0 238 0 0 39403 0 --:--:-- --:--:-- --:--:-- 47600
[
 {
 "name": "DreamSkies-cons3-heap-dump-20240808-041412-916.hprof",
 "sizeBytes": 204425284
 },
 {
 "name": "DreamSkies-cons3-heap-dump-20240808-041428-817.hprof",
 "sizeBytes": 204424631
 },
 {
 "name": "DreamSkies-cons3-heap-dump-20240808-041421-141.hprof",
 "sizeBytes": 204424551
 }
]

Т.е. для скачивания доступны три файла - первый самый большой.

Чтобы его скачать можно выполнить команду:

NAME=DreamSkies-cons3-heap-dump-20240808-041412-916.hprof
wget -O $NAME http://localhost:13000/inner/hot-spot/download-available-heap-dump-file?name=$NAME

Убедитесь, что имя файла, которое Вы отправите вендеру, останется исходным.

# Метрики для prometheus

Платформа генерирует метрики для Prometheus

## Точка доступа к метрикам

Приложение платформы использует Spring-boot, в котором из коробки предоставляются метрики для Prometheus.

Для доступа к этим метрикам нужно использовать URL:

http://HOST:PORT/actuator/prometheus

Где HOST:PORT - хост и порт серверной части платформы - она называется MyBPM API

Если используется kubernetes, и в нём установлен prometheus-operator, то для доступа к метрикам платформы MyBPM необходимо настроить ServiceMonitor:

apiVersion: monitoring.coreos.com/v1
kind: ServiceMonitor
metadata:
 name: mybpm-api-monitor
 namespace: monitoring
 labels:
 release: prometheus
spec:
 selector:
 matchLabels:
 app-group: mybpm-server
 namespaceSelector:
 matchNames:
 - perf-test
 endpoints:
 - targetPort: 8080
 scheme: http
 path: /actuator/prometheus
 interval: 1s

В этом yaml-файле необходимо поменять некоторые параметры, которые соответствуют Вашей установке. А именно:

|  |  |
| --- | --- |
| параметр | описание |
| metadata.namespace | здесь нужно указать ваш namespace, который Вы используете для мониторинга |
| spec.selector.matchLabels | здесь нужно указать фильтр для приложения платформы MyBPM, который у Вас настроен |
| spec.selector.matchLabels | здесь нужно указать правильный namespace, где находиться платформа |

Остальные параметры можно оставить без изменений.

## Описание метрик

application\_started\_time\_seconds

Указывает время старта приложения

mybpm\_go\_process\_threads\_fast\_count - Count of threads going fast processes
mybpm\_go\_process\_threads\_slow\_count - Count of threads going slow processes
mybpm\_go\_process\_threads\_fast\_limit - Limit of threads going fast processes - config RunProcessConfig.poolSizeFast
mybpm\_go\_process\_threads\_slow\_limit - Limit of threads going slow processes - config RunProcessConfig.poolSizeSlow

Эти метрики показывают состояние механизма исполнения процессов

http\_server\_requests\_seconds\_count
http\_server\_requests\_seconds\_sum

По каждому сервису в системе показываются такие метрики.

**\*\_count** - указывает количество запусков этого сервиса

**\*\_sum** - указывает суммарное время работы этого сервиса

Обе эти метрики растут, поэтому нужно использовать производные по ним, чтобы определить количество запросов в секунду, и время работы тех или иных запросов.

mybpm\_controller\_calls\_seconds\_count
mybpm\_controller\_calls\_seconds\_sum

По каждому методу в каждом контроллере на платформе формируется две метрики описывающие количественную работу этих методов.

**\*\_count** - указывает количество запусков этого метода контроллера

**\*\_sum** - указывает суммарное время работы этого метода в секундах

Контроллер - это класс в Java предназначенный для обработки REST-сервисов. В нём есть методы, которы определённым образом помечены, и они исполняют конкретные запросы по сервисам.

mongodb\_driver\_pool\_size

Указывает текущий размер пула соединений к MongoDB. Этот параметр позволяет отслеживать нагрузку на MongoDB.

logback\_events\_total{application="mybpm",level="info",} 4317071.0
logback\_events\_total{application="mybpm",level="debug",} 5.7446569E7
logback\_events\_total{application="mybpm",level="error",} 308.0
logback\_events\_total{application="mybpm",level="warn",} 4.0
logback\_events\_total{application="mybpm",level="trace",} 1.0376059E8

Показывают активность системы журналирования (логов).

jvm\_memory\_used\_bytes{application="mybpm",area="heap",id="G1 Eden Space",} 1.74063616E8
jvm\_memory\_used\_bytes{application="mybpm",area="nonheap",id="Metaspace",} 1.52550256E8
jvm\_memory\_used\_bytes{application="mybpm",area="heap",id="G1 Survivor Space",} 1848448.0
jvm\_memory\_used\_bytes{application="mybpm",area="nonheap",id="Compressed Class Space",} 1.8272688E7
jvm\_memory\_used\_bytes{application="mybpm",area="heap",id="G1 Old Gen",} 1.65742592E8
jvm\_memory\_used\_bytes{application="mybpm",area="nonheap",id="CodeCache",} 4.4950016E7
jvm\_memory\_committed\_bytes{application="mybpm",area="heap",id="G1 Eden Space",} 4.1418752E8
jvm\_memory\_committed\_bytes{application="mybpm",area="nonheap",id="Metaspace",} 1.54271744E8
jvm\_memory\_committed\_bytes{application="mybpm",area="heap",id="G1 Survivor Space",} 2097152.0
jvm\_memory\_committed\_bytes{application="mybpm",area="nonheap",id="Compressed Class Space",} 1.9202048E7
jvm\_memory\_committed\_bytes{application="mybpm",area="heap",id="G1 Old Gen",} 2.44318208E8
jvm\_memory\_committed\_bytes{application="mybpm",area="nonheap",id="CodeCache",} 5.9834368E7

Показывают количество используемой на данный момент памяти со стороны JVM.

process\_files\_open\_files{application="mybpm",} 620.0

Количество открытых дескрипторов файлов на данный момент со стороны JVM.

process\_cpu\_usage{application="mybpm",} 0.07142857142857142

Показывает количество используемого времени платформой у процессора.