ПРОГРАММНЫЙ КОМПЛЕКС ДЛЯ УСКОРЕНИЯ РАЗРАБОТКИ, ИНТЕГРАЦИИ И ТЕСТИРОВАНИЯ СЛОЖНЫХ АВТОМАТИЗИРОВАННЫХ СИСТЕМ (ДЖАВАКС)

Инструкция по установке компонент

Листов 106

СОДЕРЖАНИЕ

1. ЗАПУСК КОМПОНЕНТОВ В ОРКЕСТРАТОРЕ

1.1. Общее описание

Процесс deploy основан на декларативном подходе, что несёт за собой использование заранее подготовленных шаблонов (template), регламентирующих состояние сущностей приложения необходимых для его работы в среде функционирования (оркестраторе).

Сам deploy приложения может осуществляться как в автоматическом (по средством pipeline) так и в ручном режиме с использованием специализированных консольных утилит, с помощью которых может осуществляться администрирование оркестратора.

Процесс написания pipeline в рамках настоящей инструкции не рассматривается по причине большого разнообразия продуктов, используемых в качестве систем доставки кода в оркестратор.

Штатная работа приложения возможна на базе оркестратора Kubernates v1.24.10 (k8s).

Таблица 1 — Перечень компонентов, входящих в состав программного комплекса «Джавакс»

№ п.п.	**Название компонента**	**Тип компонента**	**Краткое описание назначения**
1	emulator-service	Сервис	Универсальный единый эмулятор для интеграционного тестирования.
2	bpm-orchestration-service	Сервис	Оболочка для организации межсервисного взаимодействия и оркестрации процессов.
3	api-visualization-commons	Сервис	Конструктор экранных форм для визуализации API/Swagger, позволяет создавать удобный пользовательский интерфейс для взаимодействия с функционалом системы
4	pack-integration-svc-template	Сервис	Шаблон интеграционного сервиса для пакетной обработки данных через внешний REST-источник с возможностью выстраивания очереди сообщений, репроцессинга, ограничения по количеству запросов

№ п.п.	Название компонента	Тип компонента	Краткое описание назначения
5	kafka-template-service	Сервис	Микросервис, осуществляющий функционал разделения и объединения сообщений
6	BrOk (состоит из следующих компонентов: brok, brok-bpm, brok-auth, brok-ui)	Сервис	Система менеджмента брокеров сообщений различного уровня.
7	rest-validation-commons	Библиотека	Инструмент, предназначенный для проведения проверки интеграционных контрактов REST API смежных сервисов.
8	mask-commons	Библиотека	Инструмент, предназначенный для маскирования персональных данных в лог-файлах
9	kafka-starter	Библиотека	Библиотека, kafka-template-service, предоставляющая обеспечительные функции для выполнения его функционала.
10	Maestro	Набор скриптов	Инструмент управления жизненным циклом инфраструктурных компонентов.

В рамках настоящей инструкции рассматривается только установка компонентов, относящихся к типу «Сервис». Процедуры подключения библиотек и использование Maestro относится к инструкции по эксплуатации, что выходит за рамки настоящей инструкции.

1.2. Пререквизиты для запуска в оркестраторе

Необходимо использование к брокеру сообщений Kafka.

Поддерживаемыми версиями брокеров сообщений являются версии Kafka 2.12-3.5.2+.

Необходимо подключение к системе управления базами данных PostgreSQL.

Поддерживаемыми версиями системы управления базами данных являются версии PostgreSQL 11+.

Допускается использование базы данных/брокера сообщений, функционирующих в контейнере на виртуальной машине и/или в оркестраторе, однако в данной реализации необходимо обеспечить сохранность данных, хранящихся на носителях, монтируемых в контейнер, во избежание утраты хранящихся на них данных в случае перезапуска контейнера с базой данных приложения.

Процесс установки и настройки брокера сообщений/систем управления базами данными в настоящей инструкции не рассматривается, по причине большого количества различных требований к наличию или отсутствию той или иной настройки в различных учреждениях.

Для работы компонентов программного комплекса «Джавакс» требуется наличие ряда баз данных и схем, в которых автоматически, при подключении компонента к базе данных будут созданы все необходимые таблицы.

Чтобы создать схему и при необходимости отдельную базу данных, нужно выполнить следующие действия:

1) Зайти по ssh на сервер с PostgreSQL.

2) Создать файл скрипт ***create_db.sh*** с помощью штатного текстового редактора.

```
vi create_db.sh
```

3) Вставить в открывшийся файл текст самого скрипта:

   а) Для внесения изменений нажимаем латинскую «**i**».

   б) Вставляем текст скрипта:

```
#!/bin/bash

# Параметры подключения к PostgreSQL
PG_USER="postgres" #Имя учетной записи с полномочиями на создание ресурсов в PostgreSQL
DB_NAME_BROK="brok" #Имя создаваемой БД (если необходимо) либо имя существующей БД, в которой будет создана схема для brok-auth
DB_NAME_JAVAX="javax"#Имя создаваемой БД (если необходимо) либо имя существующей БД, в которой будет создана схема для компонентов JAVAX

SCHEMA_NAME_BROK="brok_auth" #Название создаваемой схемы. В названии схемы в качестве разделителя допускается использование только символа «_»
SCHEMAS_NAMES_JAVAX="emulator_service bpm_orchestration_service javax_rbg_adapter"#Название создаваемых схем. В названии схемы в качестве разделителя допускается использование только символа «_»

# Проверка наличия утилиты psql
if ! command -v psql &> /dev/null; then
```

```
            echo "Утилита psql не найдена. Установите PostgreSQL и попробуйте
снова."
            exit 1
        fi

        # Создание базы данных и схемы
        echo "Создание базы данных $DB_NAME и схемы $SCHEMA_NAME..."

        # Подключение к PostgreSQL и выполнение SQL-запросов
        sudo -u $PG_USER psql -c "CREATE DATABASE $DB_NAME_BROK;" # Если новую базу
данных создавать не нужно, то закомментируйте данную строчку, поставив в начало
данной строки символ «#»
        sudo -u $PG_USER psql -d $DB_NAME_BROK -c "CREATE SCHEMA
$SCHEMA_NAME_BROK;"

        sudo -u $PG_USER psql -c "CREATE DATABASE $DB_NAME_JAVAX;" # Если новую
базу данных создавать не нужно, то закомментируйте данную строчку, поставив в
начало данной строки символ «#»
        for SCHEMA_NAME in $SCHEMAS_NAMES_JAVAX; do
          sudo -u $PG_USER psql -d $DB_NAME_JAVAX -c "CREATE SCHEMA $SCHEMA_NAME;"
        done

        echo "База данных и схема успешно созданы."
```

в) Закрываем и закрываем файл, нажав ***Shift + :***, после чего вводим с клавиатуры ***wq*** и нажимаем ***Enter***, для выходы из файла.

4) Сделать файл скрипта исполняемым.

```
chmod 755 ./create_db.sh
```

5) Запустить файл скрипта, который создаст новую базу данных (при необходимости) и схему, необходимую для работы приложения brok-auth.

```
./create_db.sh
```

6) Если в консольном выводе вы видите сообщение **«База данных и схема успешно созданы»**, значит база данных (если Вы её создавали) и схема успешна создана.

2. ОПИСАНИЕ TEMPLATE И DEPLOY

2.1. Описание template используемых в процессе deploy компонентов в оркестратор Kubernetes(k8s)

При создании файлов template в качестве названия рекомендуется использование общепризнанного подхода в нейминге template, в частности:

- Deployment.yml.
- ConfigMap.yml
- Secret.yml
- Ingress.yml
- Service.yml

Допускается использование иного, отличного от указанного в рекомендациях подхода по неймингу template.

Для штатной работы приложения необходимо использовать одинаковые значения параметров с одинаковым названием, которые необходимо задать в template перед deploy.

2.1.1. Описание template Deployment.yml для каждого компонента

1) В качестве описания сущности Deployment для компонента **emulator-service/bpm-orchestration-service -service** используется следующий template:

```
kind: Deployment
apiVersion: apps/v1
metadata:
  name: < DC_NAME >
  namespace: < PROJECT >
  labels:
    app: < DC_NAME >
    name: < DC_NAME >
spec:
  replicas: 1
  selector:
    matchLabels:
      app: < DC_NAME >
  template:
    metadata:
      creationTimestamp: null
      labels:
        app: < DC_NAME >
        deployment: < DC_NAME >
    spec:
```

```
      triggers:
        - type: "ConfigChange"
      restartPolicy: Always
      containers:
        - name: < DC_NAME >
          image: <IMAGE_REGISTRY>/< DC_NAME >:< IMAGE_TAG >
          imagePullPolicy: Always
          ports:
            - containerPort: < APP_PORT >
              protocol: TCP
          env:
            - name: POD_NAME
              valueFrom:
                fieldRef:
                  fieldPath: metadata.name
            - name: NAMESPACE
              valueFrom:
                fieldRef:
                  fieldPath: metadata.namespace
            - name: LOG_LEVEL
              value: INFO
            - name: TZ
              value: 'Europe/Moscow'
            - name: JAVA_OPTS
              value: '-XX:+UseContainerSupport -XX:MaxRAMPercentage=75.0'
          envFrom:
            - configMapRef:
                name: < DC_NAME >
            - secretRef:
                name: < DC_NAME >
          resources:
            limits:
              cpu: <CPU_LIMIT>
              memory: <MEMORY_LIMIT>
            requests:
              cpu: < CPU_REQUEST >
              memory: < MEMORY_REQUEST >
          terminationMessagePath: /dev/termination-log
          terminationMessagePolicy: File
      terminationGracePeriodSeconds: 30
      dnsPolicy: ClusterFirst
      nodeSelector:
```

```
            node-role.kubernetes.io/worker: worker
          serviceAccountName: < SA >
          serviceAccount: < SA >
          securityContext: {}
          imagePullSecrets:
            - name: < IMAGE_PULL_SECRET >
          schedulerName: default-scheduler
      strategy:
        type: RollingUpdate
        rollingUpdate:
          maxUnavailable: 50%
          maxSurge: 50%
      revisionHistoryLimit: 10
      progressDeadlineSeconds: 600
```

2) В качестве описания сущности Deployment для компонента **api-visualization-commons/api-visualization-commons/pack-integration-svc-template** используется следующий template (концептуальных отличий от template, используемого для deploy **emulator-service/bpm-orchestration-service -service,** нет, за исключением того, что для данные сервисы не используют для хранения переменных сущность secret, в связи с чем нет необходимости её указывать в template):

```
kind: Deployment
apiVersion: apps/v1
metadata:
  name: < DC_NAME >
  namespace: < PROJECT >
  labels:
    app: < DC_NAME >
  name: < DC_NAME >
spec:
  replicas: 1
  selector:
    matchLabels:
      app: < DC_NAME >
  template:
    metadata:
      creationTimestamp: null
      labels:
        app: < DC_NAME >
        deployment: < DC_NAME >
```

```
spec:
  triggers:
    - type: "ConfigChange"
  restartPolicy: Always
  containers:
    - name: < DC_NAME >
      image: <IMAGE_REGISTRY>/< DC_NAME >:< IMAGE_TAG >
      imagePullPolicy: Always
      ports:
        - containerPort: < APP_PORT >
          protocol: TCP
      env:
        - name: POD_NAME
          valueFrom:
            fieldRef:
              fieldPath: metadata.name
        - name: NAMESPACE
          valueFrom:
            fieldRef:
              fieldPath: metadata.namespace
        - name: LOG_LEVEL
          value: INFO
        - name: TZ
          value: 'Europe/Moscow'
        - name: JAVA_OPTS
          value: '-XX:+UseContainerSupport -XX:MaxRAMPercentage=75.0'
      envFrom:
        - configMapRef:
            name: < DC_NAME >
      resources:
        limits:
          cpu: <CPU_LIMIT>
          memory: <MEMORY_LIMIT>
        requests:
          cpu: < CPU_REQUEST >
          memory: < MEMORY_REQUEST >
      terminationMessagePath: /dev/termination-log
      terminationMessagePolicy: File
  terminationGracePeriodSeconds: 30
  dnsPolicy: ClusterFirst
  nodeSelector:
    node-role.kubernetes.io/worker: worker
```

```
        serviceAccountName: < SA >
        serviceAccount: < SA >
        securityContext: {}
        imagePullSecrets:
          - name: < IMAGE_PULL_SECRET >
        schedulerName: default-scheduler
    strategy:
      type: RollingUpdate
      rollingUpdate:
        maxUnavailable: 50%
        maxSurge: 50%
    revisionHistoryLimit: 10
    progressDeadlineSeconds: 600
```

3) В качестве описания сущности Deployment для компонента kafka-template-service используется следующий template:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  namespace: < PROJECT >
  name: < DC_NAME >
  labels:
    app: < DC_NAME >
    name: < DC_NAME >
spec:
  replicas: 1
  strategy:
    type: RollingUpdate
    rollingUpdate:
      maxUnavailable: 50%
      maxSurge: 50%
  selector:
    matchLabels:
      deployment: < DC_NAME >
  strategy:
    rollingParams:
      maxSurge: 50%
      maxUnavailable: 50%
    type: RollingUpdate
  template:
    metadata:
```

```
      creationTimestamp: null
      annotations:
        app.kubernetes.io/name: < DC_NAME >
      labels:
        app: < DC_NAME >
        deployment: < DC_NAME >
    spec:
      nodeSelector:
        node-role.kubernetes.io/worker: worker
      imagePullSecrets:
        - name: < IMAGE_PULL_SECRET >
      securityContext: {}
      serviceAccountName: < SA >
      containers:
        - name: < DC_NAME >
          image: < IMAGE_REGISTRY >/< IMAGE_NAME >:< IMAGE_TAG >
          imagePullPolicy: Always
          env:
            - name: POD_NAME
              valueFrom:
                fieldRef:
                  fieldPath: metadata.name
            - name: NAMESPACE
              valueFrom:
                fieldRef:
                  fieldPath: metadata.namespace
            - name: LOG_LEVEL
              value: INFO
            - name: TZ
              value: 'Europe/Moscow'
          ports:
            - containerPort: < APP_PORT >
              protocol: TCP
          resources:
            limits:
              cpu: < CPU_LIMIT >
              memory: < MEMORY_LIMIT >
            requests:
              cpu: < CPU_REQUEST >
              memory: < MEMORY_REQUEST >
```

4) В качестве описания сущности Deployment для компонента brok и brok-bpm используется следующий template:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  namespace: < PROJECT >
  name: < DC_NAME >
  labels:
    app: < DC_NAME >
    name: < DC_NAME >
spec:
  replicas: < REPLICAS_COUNT >
  selector:
    matchLabels:
      app: < DC_NAME >
  template:
    metadata:
      creationTimestamp: null
      labels:
        app: < DC_NAME >
        deployment: < DC_NAME >
    spec:
      imagePullSecrets:
        - name: < IMAGE_PULL_SECRET >
      securityContext: {}
      nodeSelector:
        node-role.kubernetes.io/worker: worker
      serviceAccountName: < SA >
      serviceAccount: < SA >
      containers:
        - name: < DC_NAME >
          image: <IMAGE_REGISTRY>/< IMAGE_NAME >:<IMAGE_TAG>
          imagePullPolicy: Always
          env:
            - name: POD_NAME
              valueFrom:
                fieldRef:
                  fieldPath: metadata.name
            - name: NAMESPACE
              valueFrom:
                fieldRef:
                  fieldPath: metadata.namespace
```

```
      - name: LOG_LEVEL
        value: INFO
      - name: TZ
        value: "INFO"
    envFrom:
      - configMapRef:
          name: < DC_NAME >
      - secretRef:
          name: < DC_NAME >
    ports:
      - containerPort: < APP_PORT >
        protocol: TCP
    resources:
      limits:
        cpu: < CPU_LIMIT >
        memory: < MEMORY_LIMIT >
      requests:
        cpu: < CPU_REQUEST >
        memory: < MEMORY_REQUEST >
    livenessProbe:
      failureThreshold: 3
      httpGet:
        path: /actuator/health/liveness
        port: < APP_PORT >
        scheme: HTTP
      initialDelaySeconds: < LIVENESS_PROBE_INITIAL_DELAY_SECONDS >
      periodSeconds: < LIVENESS_PROBE_ PERIOD_SECONDS >
      successThreshold:  < LIVENESS_PROBE_ SUCCESS_TRESHHOLD >
      timeoutSeconds: < LIVENESS_PROBE_TIMEOUT_SECONDS >
    readinessProbe:
      httpGet:
        path: /actuator/health/readiness
        port: < APP_PORT >
        scheme: HTTP
      failureThreshold: < READINESS_PROBE_FAILURE_THRESHOLD >
      initialDelaySeconds: < READINESS_PROBE_INITIAL_DELAY_SECONDS >
      periodSeconds: < READINESS_PROBE_PERIOD_SECONDS >
```

5) В качестве описания сущности Deployment для компонента **brok-auth** используется следующий template (концептуальных отличий от template, используемого для deploy **brok,** нет, за исключением адресов, используемых для проверок **livenessProbe** и **readinessProbe**).

```
apiVersion: apps/v1
kind: Deployment
metadata:
  namespace: < PROJECT >
  name: < DC_NAME >
  labels:
    app: < DC_NAME >
    name: < DC_NAME >
spec:
  replicas: < REPLICAS_COUNT >
  selector:
    matchLabels:
      app: < DC_NAME >
  template:
    metadata:
      creationTimestamp: null
      labels:
        app: < DC_NAME >
        deployment: < DC_NAME >
    spec:
      imagePullSecrets:
        - name: < IMAGE_PULL_SECRET >
      securityContext: {}
      nodeSelector:
        node-role.kubernetes.io/worker: worker
      serviceAccountName: < SA >
      serviceAccount: < SA >
      containers:
        - name: < DC_NAME >
          image: <IMAGE_REGISTRY>/< IMAGE_NAME >:<IMAGE_TAG>
          imagePullPolicy: Always
          env:
            - name: POD_NAME
              valueFrom:
                fieldRef:
                  fieldPath: metadata.name
            - name: NAMESPACE
              valueFrom:
                fieldRef:
                  fieldPath: metadata.namespace
            - name: LOG_LEVEL
              value: INFO
```

```
    - name: TZ
      value: "INFO"
  envFrom:
    - configMapRef:
        name: < DC_NAME >
   - secretRef:
  name: < DC_NAME >
  ports:
    - containerPort: < APP_PORT >
      protocol: TCP
  resources:
    limits:
      cpu: < CPU_LIMIT >
      memory: < MEMORY_LIMIT >
    requests:
      cpu: < CPU_REQUEST >
      memory: < MEMORY_REQUEST >
  livenessProbe:
    failureThreshold: 3
    httpGet:
      path: /auth/
      port: < APP_PORT >
      scheme: HTTP
    initialDelaySeconds: < LIVENESS_PROBE_INITIAL_DELAY_SECONDS >
    periodSeconds: < LIVENESS_PROBE_ PERIOD_SECONDS >
    successThreshold:  < LIVENESS_PROBE_ SUCCESS_TRESHHOLD >
    timeoutSeconds: < LIVENESS_PROBE_TIMEOUT_SECONDS >
  readinessProbe:
    httpGet:
      path: /auth/realms/master
      port: < APP_PORT >
      scheme: HTTP
    failureThreshold: < READINESS_PROBE_FAILURE_THRESHOLD >
    initialDelaySeconds: < READINESS_PROBE_INITIAL_DELAY_SECONDS >
    periodSeconds: < READINESS_PROBE_PERIOD_SECONDS >
```

6) В качестве описания сущности Deployment для компонента **brok-ui** используется следующий template (концептуальных отличий от template используемого для deploy **brok** нет, за исключением монтирования volumes содержащих truststore, keystore используемые для подключения к брокеру кафка по протоколу SSL в случае необходимости):

```
apiVersion: apps/v1
kind: Deployment
metadata:
  namespace: < PROJECT >
  name: < DC_NAME >
  labels:
    app: < DC_NAME >
    name: < DC_NAME >
spec:
  replicas: < REPLICAS_COUNT >
  selector:
    matchLabels:
      app: < DC_NAME >
  template:
    metadata:
      creationTimestamp: null
      labels:
        app: < DC_NAME >
        deployment: < DC_NAME >
    spec:
      imagePullSecrets:
        - name: < IMAGE_PULL_SECRET >
      securityContext: {}
      nodeSelector:
        node-role.kubernetes.io/worker: worker
      serviceAccountName: < SA >
      serviceAccount: < SA >
      containers:
        - name: < DC_NAME >
          image: <IMAGE_REGISTRY>/< IMAGE_NAME >:<IMAGE_TAG>
          imagePullPolicy: Always
          env:
            - name: POD_NAME
              valueFrom:
                fieldRef:
                  fieldPath: metadata.name
            - name: NAMESPACE
              valueFrom:
                fieldRef:
                  fieldPath: metadata.namespace
            - name: LOG_LEVEL
              value: INFO
```

```
                - name: TZ
                  value: "INFO"
              envFrom:
                - configMapRef:
                    name: < DC_NAME >
                -     secretRef:
      name: < DC_NAME >
              volumeMounts:
                - name: brok-server
                  mountPath: /mnt/secrets/brok-server
                - name: brok-client
                  mountPath: /mnt/secrets/brok-client
              ports:
                - containerPort: < APP_PORT >
                  protocol: TCP
              resources:
                limits:
                  cpu: < CPU_LIMIT >
                  memory: < MEMORY_LIMIT >
                requests:
                  cpu: < CPU_REQUEST >
                  memory: < MEMORY_REQUEST >
              livenessProbe:
                failureThreshold: 3
                httpGet:
                  path: /auth/
                  port: < APP_PORT >
                  scheme: HTTP
                initialDelaySeconds: < LIVENESS_PROBE_INITIAL_DELAY_SECONDS >
                periodSeconds: < LIVENESS_PROBE_ PERIOD_SECONDS >
                successThreshold:  < LIVENESS_PROBE_ SUCCESS_TRESHHOLD >
                timeoutSeconds: < LIVENESS_PROBE_TIMEOUT_SECONDS >
              readinessProbe:
                httpGet:
                  path: /auth/realms/master
                  port: < APP_PORT >
                  scheme: HTTP
                failureThreshold: < READINESS_PROBE_FAILURE_THRESHOLD >
                initialDelaySeconds: < READINESS_PROBE_INITIAL_DELAY_SECONDS >
                periodSeconds: < READINESS_PROBE_PERIOD_SECONDS >
            volumes:
              - name: brok-server
```

```
      secret:
        secretName: "brok-server"
        items:
          - key: server.truststore.jks
            path: server.truststore.jks
          - key: server.keystore.jks
            path: server.keystore.jks
    - name: brok-client
      secret:
        secretName: "brok-client"
        items:
          - key: client.truststore.jks
            path: client.truststore.jks
          - key: client.keystore.jks
            path: client.keystore.jks
```

7) Параметры, которые необходимо задать в template перед deploy:

<DC_NAME> — имя компонента приложения. Рекомендуемые имена для компонентов приложения: brok/brok-bpm/brok-auth/brok-ui, однако может быть задано любое имя компонента приложения в соответствии с требованиями нейминга приложений внутри учреждения.

<PROJECT> — пространство имен оркестратора, в которое непосредственно будет осуществляться deploy приложения.

<IMAGE_PULL_SECRET> — названия секрета (сущность оркестратора), с использованием которого осуществляется процесс скачивания (pull) образа приложения из менеджера репозиториев для хранения артефактов.

<SA> — название технической учетной записи от имени, с которой осуществляется запуск контейнера с приложением внутри оркестратора.

<IMAGE_REGISTRY> — URL-адрес менеджера репозиториев для хранения артефактов, из которого осуществляется pull docker образа приложения.

<IMAGE_NAME> — названия docker образа приложения, хранящегося в менеджере репозиториев для хранения артефактов.

<IMAGE_TAG> — тег docker образа приложения, хранящегося в менеджере репозиториев для хранения артефактов.

<APP_PORT> — порт, по которому осуществляется доступ внутри оркестратора и по которому приложение отвечает по поступающие к нему запросы.

<CPU_LIMIT> — параметр, определяющий максимальное количество процессорного времени, которое контейнер может использовать. Штатная работа приложения протестирована на показателе данного параметра в 2 ядра. В случае если будут заданы более низкие показатели, работа приложения в штатном режиме не гарантирована.

<MEMORY_LIMIT> — максимальный объём оперативной памяти, доступный для работы данного компонента приложения. Штатная работа приложения протестирована на показателе данного параметра в 1Gi. В случае если будут заданы более низкие показатели, работа приложения в штатном режиме не гарантирована.

<CPU_REQUEST> — параметр, определяющий минимальное количество процессорного времени, которое контейнер запросит у оркестратора. Штатная работа приложения протестирована на показателе данного параметра в 150 milicore (150m). В случае если будут заданы более низкие показатели, работа приложения в штатном режиме не гарантирована.

<MEMORY_REQUEST> — объём оперативной памяти, зарезервированный контейнером для работы данного компонента приложения. Штатная работа приложения протестирована на показателе данного параметра в 1Gi. В случае если будут заданы более низкие показатели, работа приложения в штатном режиме не гарантирована.

<READINESS_PROBE_FAILURE_THRESHOLD> — параметр, определяющий количество последовательных неудачных проверок, после которого контейнер будет считаться неготовым. Рекомендуемое значение для данного параметра 1.

<READINESS_PROBE_INITIAL_DELAY_SECONDS> — параметр, определяющий количество секунд, которое оркестратор будет ожидать после запуска контейнера, прежде чем начнет проверять его параметр readiness. Рекомендуемое значение для данного параметра 30.

<READINESS_PROBE_PERIOD_SECONDS> — параметр устанавливающий интервал времени в секундах между последовательными проверками readiness. Рекомендуемое значение для данного параметра 30.

<LIVENESS_PROBE_INITIAL_DELAY_SECONDS> — параметр, определяющий количество секунд, которое оркестратор будет ожидать после запуска контейнера, прежде чем начнет проверять его liveness. Рекомендуемое значение для данного параметра 600.

< LIVENESS_PROBE_ SUCCESS_TRESHHOLD > — параметр, определяющий количество неуспешных попыток проверки доступности компонента приложения в контейнере.

<LIVENESS_PROBE_ PERIOD_SECONDS> — параметр, устанавливающий интервал времени в секундах между последовательными проверками liveness. Рекомендуемое значение для данного параметра 30.

<LIVENESS_PROBE_ SUCCESS_TRESHHOLD> — параметр, определяющий количество последовательных успешных проверок, необходимых для того, чтобы контейнер был считан живым. Рекомендуемое значение для данного параметра 1.

<LIVENESS_PROBE_TIMEOUT_SECONDS> — параметр, устанавливающий максимальное время в секундах, которое оркестратор будет ждать ответа от проверки liveness. Рекомендуемое значение для данного параметра 10.

8) Примеры template для каждого компонента программного комплекса «Джавакс».

a) Deployment emulator-service

```
apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
    app: emulator-service
    name: emulator-service
  name: emulator-service
  namespace: develop
spec:
  progressDeadlineSeconds: 600
  replicas: 1
  revisionHistoryLimit: 10
  selector:
    matchLabels:
      app: emulator-service
  strategy:
    rollingUpdate:
      maxSurge: 50%
      maxUnavailable: 50%
    type: RollingUpdate
  template:
    metadata:
      creationTimestamp: null
      labels:
        app: emulator-service
        deployment: emulator-service
    spec:
      containers:
        - env:
            - name: POD_NAME
              valueFrom:
```

```
                fieldRef:
                  apiVersion: v1
                  fieldPath: metadata.name
              - name: NAMESPACE
                valueFrom:
                  fieldRef:
                    apiVersion: v1
                    fieldPath: metadata.namespace
              - name: LOG_LEVEL
                value: INFO
              - name: TZ
                value: Europe/Moscow
              - name: JAVA_OPTS
                value: '-XX:+UseContainerSupport -XX:MaxRAMPercentage=75.0'
              - name: DATASOURCE_PASSWORD
            envFrom:
              - configMapRef:
                  name: emulator-service
              - secretRef:
                  name: emulator-service
            image: nexus.ru-central1.internal:5000/emulator-service:develop
            imagePullPolicy: Always
            name: emulator-service
            ports:
              - containerPort: 8080
                protocol: TCP
            resources:
              limits:
                cpu: '2'
                ephemeral-storage: 300Mi
                memory: 1Gi
              requests:
                cpu: 150m
                ephemeral-storage: 100Mi
                memory: 1Gi
            terminationMessagePath: /dev/termination-log
            terminationMessagePolicy: File
        dnsPolicy: ClusterFirst
        imagePullSecrets:
          - name: docker-registry-pull-secret
        nodeSelector:
          node-role.kubernetes.io/worker: worker
```

```
      restartPolicy: Always
      schedulerName: default-scheduler
      securityContext: {}
      serviceAccount: deployer-service
      serviceAccountName: deployer-service
      terminationGracePeriodSeconds: 30
```

б) Deployment bpm-orchestration-service

```
apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
    app: bpm-orchestration-service
    name: bpm-orchestration-service
  name: bpm-orchestration-service
  namespace: develop
spec:
  progressDeadlineSeconds: 600
  replicas: 0
  revisionHistoryLimit: 10
  selector:
    matchLabels:
      app: bpm-orchestration-service
  strategy:
    rollingUpdate:
      maxSurge: 50%
      maxUnavailable: 50%
    type: RollingUpdate
  template:
    metadata:
      creationTimestamp: null
      labels:
        app: bpm-orchestration-service
        deployment: bpm-orchestration-service
    spec:
      containers:
        - env:
            - name: POD_NAME
              valueFrom:
                fieldRef:
```

```
                        apiVersion: v1
                        fieldPath: metadata.name
                  - name: NAMESPACE
                    valueFrom:
                      fieldRef:
                        apiVersion: v1
                        fieldPath: metadata.namespace
                  - name: LOG_LEVEL
                    value: INFO
                  - name: TZ
                    value: Europe/Moscow
                  - name: JAVA_OPTS
                    value: '-XX:+UseContainerSupport -XX:MaxRAMPercentage=75.0'
                  - name: DATASOURCE_PASSWORD
                envFrom:
                  - configMapRef:
                      name: bpm-orchestration-service
                  - secretRef:
                      name: bpm-orchestration-service
                image: >-
                  nexus.ru-central1.internal:5000/bpm-orchestration-service:de-
velop
                imagePullPolicy: Always
                name: bpm-orchestration-service
                ports:
                  - containerPort: 8080
                    protocol: TCP
                resources:
                  limits:
                    cpu: '2'
                    ephemeral-storage: 300Mi
                    memory: 1Gi
                  requests:
                    cpu: 150m
                    ephemeral-storage: 100Mi
                    memory: 1Gi
                terminationMessagePath: /dev/termination-log
                terminationMessagePolicy: File
              dnsPolicy: ClusterFirst
              imagePullSecrets:
                - name: docker-registry-pull-secret
              nodeSelector:
```

```
            node-role.kubernetes.io/worker: worker
          restartPolicy: Always
          schedulerName: default-scheduler
          securityContext: {}
          serviceAccount: deployer-service
          serviceAccountName: deployer-service
          terminationGracePeriodSeconds: 30
```

в) Deployment api-visualization-commons

```
apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
    app: api-visualization-commons
    name: api-visualization-commons
  name: api-visualization-commons
  namespace: develop
spec:
  progressDeadlineSeconds: 600
  replicas: 1
  revisionHistoryLimit: 10
  selector:
    matchLabels:
      app: api-visualization-commons
  strategy:
    rollingUpdate:
      maxSurge: 50%
      maxUnavailable: 50%
    type: RollingUpdate
  template:
    metadata:
      creationTimestamp: null
      labels:
        app: api-visualization-commons
        deployment: api-visualization-commons
    spec:
      containers:
        - env:
            - name: POD_NAME
              valueFrom:
                fieldRef:
```

```
                    apiVersion: v1
                    fieldPath: metadata.name
              - name: NAMESPACE
                valueFrom:
                  fieldRef:
                    apiVersion: v1
                    fieldPath: metadata.namespace
              - name: LOG_LEVEL
                value: INFO
              - name: TZ
                value: Europe/Moscow
              - name: JAVA_OPTS
                value: '-XX:+UseContainerSupport -XX:MaxRAMPercentage=75.0'
            envFrom:
              - configMapRef:
                  name: api-visualization-commons
            image: >-
              nexusru-central1.internal:5000/api-visualization-commons:develop
            imagePullPolicy: Always
            name: api-visualization-commons
            ports:
              - containerPort: 8080
                protocol: TCP
            resources:
              limits:
                cpu: '2'
                ephemeral-storage: 300Mi
                memory: 1Gi
              requests:
                cpu: 150m
                ephemeral-storage: 100Mi
                memory: 1Gi
            terminationMessagePath: /dev/termination-log
            terminationMessagePolicy: File
          dnsPolicy: ClusterFirst
          imagePullSecrets:
            - name: docker-registry-pull-secret
          nodeSelector:
            node-role.kubernetes.io/worker: worker
          restartPolicy: Always
          schedulerName: default-scheduler
          securityContext: {}
```

```
      serviceAccount: deployer-service
      serviceAccountName: deployer-service
      terminationGracePeriodSeconds: 30
```

г) Deployment pack-integration-svc-template

```
apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
    app: pack-integration-svc-template
    name: pack-integration-svc-template
  name: pack-integration-svc-template
  namespace: develop
spec:
  progressDeadlineSeconds: 600
  replicas: 1
  revisionHistoryLimit: 10
  selector:
    matchLabels:
      app: pack-integration-svc-template
  strategy:
    rollingUpdate:
      maxSurge: 50%
      maxUnavailable: 50%
    type: RollingUpdate
  template:
    metadata:
      creationTimestamp: null
      labels:
        app: pack-integration-svc-template
        deployment: pack-integration-svc-template
    spec:
      containers:
        - env:
            - name: POD_NAME
              valueFrom:
                fieldRef:
                  apiVersion: v1
                  fieldPath: metadata.name
            - name: NAMESPACE
              valueFrom:
```

```
                        fieldRef:
                          apiVersion: v1
                          fieldPath: metadata.namespace
                    - name: LOG_LEVEL
                      value: INFO
                    - name: TZ
                      value: Europe/Moscow
                    - name: JAVA_OPTS
                      value: '-XX:+UseContainerSupport -XX:MaxRAMPercentage=75.0'
                  envFrom:
                    - configMapRef:
                        name: pack-integration-svc-template
                  image: >-
                    nexus.ru-central1.internal:5000/pack-integration-svc-tem-
plate:develop
                  imagePullPolicy: Always
                  name: pack-integration-svc-template
                  ports:
                    - containerPort: 8080
                      protocol: TCP
                  resources:
                    limits:
                      cpu: '2'
                      ephemeral-storage: 300Mi
                      memory: 1Gi
                    requests:
                      cpu: 150m
                      ephemeral-storage: 100Mi
                      memory: 1Gi
                  terminationMessagePath: /dev/termination-log
                  terminationMessagePolicy: File
              dnsPolicy: ClusterFirst
              imagePullSecrets:
                - name: docker-registry-pull-secret
              nodeSelector:
                node-role.kubernetes.io/worker: worker
              restartPolicy: Always
              schedulerName: default-scheduler
              securityContext: {}
              serviceAccount: deployer-service
              serviceAccountName: deployer-service
              terminationGracePeriodSeconds: 30
```

д) Deployment kafka-template-service

```
apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
    app: kafka-template-service
    name: kafka-template-service
  name: kafka-template-service
  namespace: develop
spec:
  progressDeadlineSeconds: 600
  replicas: 1
  revisionHistoryLimit: 10
  selector:
    matchLabels:
      deployment: kafka-template-service
  strategy:
    rollingUpdate:
      maxSurge: 25%
      maxUnavailable: 25%
    type: RollingUpdate
  template:
    metadata:
      annotations:
        app.kubernetes.io/name: kafka-template-service
        prometheus.io/scrape: 'true'
      creationTimestamp: null
      labels:
        app: kafka-template-service
        deployment: kafka-template-service
    spec:
      containers:
        - env:
            - name: POD_NAME
              valueFrom:
                fieldRef:
                  apiVersion: v1
                  fieldPath: metadata.name
            - name: NAMESPACE
              valueFrom:
                fieldRef:
```

```
                        apiVersion: v1
                        fieldPath: metadata.namespace
                  - name: LOG_LEVEL
                    value: INFO
                  - name: TZ
                    value: Europe/Moscow
                ports:
                  - containerPort: 8080
                    protocol: TCP
                resources:
                  limits:
                    cpu: '2'
                    ephemeral-storage: 300Mi
                    memory: 1Gi
                  requests:
                    cpu: 150m
                    ephemeral-storage: 100Mi
                    memory: 1Gi
                image: >-
                  nexus-dev-javax.ru-central1.internal:5000/kafka-template-ser-
vice:develop
                imagePullPolicy: Always
                name: kafka-template-service
                terminationMessagePath: /dev/termination-log
                terminationMessagePolicy: File
            dnsPolicy: ClusterFirst
            imagePullSecrets:
              - name: docker-registry-pull-secret
            nodeSelector:
              node-role.kubernetes.io/worker: worker
            restartPolicy: Always
            schedulerName: default-scheduler
            securityContext: {}
            serviceAccount: deployer-service
            serviceAccountName: deployer-service
            terminationGracePeriodSeconds: 30
```

e) Deployment brok

```
apiVersion: apps/v1
kind: Deployment
metadata:
```

```
    annotations:
      meta.helm.sh/release-name: brok
      meta.helm.sh/release-namespace: develop
    labels:
      app: brok
      app.kubernetes.io/managed-by: Helm
      name: brok
    name: brok
    namespace: develop
  spec:
    progressDeadlineSeconds: 600
    replicas: 1
    revisionHistoryLimit: 10
    selector:
      matchLabels:
        app: brok
    strategy:
      rollingUpdate:
        maxSurge: 25%
        maxUnavailable: 25%
      type: RollingUpdate
    template:
      metadata:
        labels:
          app: brok
          deployment: brok
      spec:
        containers:
          - env:
              - name: POD_NAME
                valueFrom:
                  fieldRef:
                    apiVersion: v1
                    fieldPath: metadata.name
              - name: NAMESPACE
                valueFrom:
                  fieldRef:
                    apiVersion: v1
                    fieldPath: metadata.namespace
              - name: LOG_LEVEL
                value: DEBUG
              - name: TZ
```

```
            value: Europe/Moscow
        envFrom:
          - configMapRef:
              name: brok
        image: nexus.ru-central1.internal:5000/brok:develop
        imagePullPolicy: Always
        livenessProbe:
          failureThreshold: 3
          httpGet:
            path: /actuator/health/liveness
            port: 8081
            scheme: HTTP
          initialDelaySeconds: 600
          periodSeconds: 30
          successThreshold: 1
          timeoutSeconds: 10
        name: brok
        ports:
          - containerPort: 8081
            protocol: TCP
        readinessProbe:
          failureThreshold: 1
          httpGet:
            path: /actuator/health/readiness
            port: 8081
            scheme: HTTP
          initialDelaySeconds: 30
          periodSeconds: 30
          successThreshold: 1
          timeoutSeconds: 1
        resources:
          limits:
            cpu: 600m
            memory: 4Gi
          requests:
            cpu: 300m
            memory: 2Gi
        terminationMessagePath: /dev/termination-log
        terminationMessagePolicy: File
      dnsPolicy: ClusterFirst
      imagePullSecrets:
        - name: docker-registry-pull-secret
```

```
    restartPolicy: Always
    schedulerName: default-scheduler
    securityContext: {}
    serviceAccount: < SA >
    serviceAccountName: < SA >
    terminationGracePeriodSeconds: 30
```

ж) Deployment brok-bpm

```
apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
    app: brok-bpm
    app.kubernetes.io/managed-by: Helm
    name: brok-bpm
  name: brok-bpm
  namespace: develop
spec:
  progressDeadlineSeconds: 600
  replicas: 1
  revisionHistoryLimit: 10
  selector:
    matchLabels:
      app: brok-bpm
  strategy:
    rollingUpdate:
      maxSurge: 25%
      maxUnavailable: 25%
    type: RollingUpdate
  template:
    metadata:
      creationTimestamp: null
      labels:
        app: brok-bpm
        deployment: brok-bpm
    spec:
      containers:
        - env:
            - name: POD_NAME
              valueFrom:
                fieldRef:
```

```
          apiVersion: v1
          fieldPath: metadata.name
    - name: NAMESPACE
      valueFrom:
        fieldRef:
          apiVersion: v1
          fieldPath: metadata.namespace
    - name: LOG_LEVEL
      value: DEBUG
    - name: TZ
      value: Europe/Moscow
  envFrom:
    - configMapRef:
        name: brok-bpm
  image: nexus.ru-central1.internal:5000/brok-bpm:develop
  imagePullPolicy: Always
  livenessProbe:
    failureThreshold: 3
    httpGet:
      path: /actuator/health/liveness
      port: 8082
      scheme: HTTP
    initialDelaySeconds: 600
    periodSeconds: 30
    successThreshold: 1
    timeoutSeconds: 10
  name: brok-bpm
  ports:
    - containerPort: 8082
      protocol: TCP
  readinessProbe:
    failureThreshold: 1
    httpGet:
      path: /actuator/health/readiness
      port: 8082
      scheme: HTTP
    initialDelaySeconds: 30
    periodSeconds: 30
    successThreshold: 1
    timeoutSeconds: 1
  resources:
    limits:
```

```
                cpu: 600m
                memory: 4Gi
              requests:
                cpu: 300m
                memory: 2Gi
            terminationMessagePath: /dev/termination-log
            terminationMessagePolicy: File
        dnsPolicy: ClusterFirst
        imagePullSecrets:
          - name: docker-registry-pull-secret
        nodeSelector:
          node-role.kubernetes.io/worker: worker
        restartPolicy: Always
        schedulerName: default-scheduler
        securityContext: {}
        serviceAccount: deployer-service
        serviceAccountName: deployer-service
        terminationGracePeriodSeconds: 30
```

з) Deployment brok-auth

```
apiVersion: apps/v1
kind: Deployment
metadata:
  annotations:
    meta.helm.sh/release-name: brok
    meta.helm.sh/release-namespace: develop
  generation: 2
  labels:
    app: brok-auth
    app.kubernetes.io/managed-by: Helm
    name: brok-auth
  name: brok-auth
  namespace: develop
spec:
  progressDeadlineSeconds: 600
  replicas: 1
  revisionHistoryLimit: 10
  selector:
    matchLabels:
      app: brok-auth
  strategy:
```

```
    rollingUpdate:
      maxSurge: 25%
      maxUnavailable: 25%
    type: RollingUpdate
  template:
    metadata:
      labels:
        app: brok-auth
        deployment: brok-auth
    spec:
      containers:
        - env:
            - name: POD_NAME
              valueFrom:
                fieldRef:
                  apiVersion: v1
                  fieldPath: metadata.name
            - name: NAMESPACE
              valueFrom:
                fieldRef:
                  apiVersion: v1
                  fieldPath: metadata.namespace
            - name: LOG_LEVEL
              value: DEBUG
            - name: TZ
              value: Europe/Moscow
          envFrom:
            - configMapRef:
                name: brok-auth
          image: nexus.ru-central1.internal:5000/brok-auth:develop
          imagePullPolicy: Always
          livenessProbe:
            failureThreshold: 3
            httpGet:
              path: /auth/
              port: 8080
              scheme: HTTP
            initialDelaySeconds: 600
            periodSeconds: 30
            successThreshold: 1
            timeoutSeconds: 10
          name: brok-auth
```

```
            ports:
              - containerPort: 8080
                protocol: TCP
            readinessProbe:
              failureThreshold: 1
              httpGet:
                path: /auth/realms/master
                port: 8080
                scheme: HTTP
              initialDelaySeconds: 30
              periodSeconds: 30
              successThreshold: 1
              timeoutSeconds: 1
            resources:
              limits:
                cpu: 600m
                memory: 4Gi
              requests:
                cpu: 300m
                memory: 2Gi
            terminationMessagePath: /dev/termination-log
            terminationMessagePolicy: File
        dnsPolicy: ClusterFirst
        imagePullSecrets:
          - name: docker-registry-pull-secret
        nodeSelector:
          node-role.kubernetes.io/worker: worker
        restartPolicy: Always
        schedulerName: default-scheduler
        securityContext: {}
        serviceAccount: deployer-service
        serviceAccountName: deployer-service
        terminationGracePeriodSeconds: 30
```

и) Deployment brok-ui

```
apiVersion: apps/v1
kind: Deployment
metadata:
  annotations:
    meta.helm.sh/release-name: brok
```

```
      meta.helm.sh/release-namespace: develop
    labels:
      app: brok-ui
      app.kubernetes.io/managed-by: Helm
      name: brok-ui
    name: brok-ui
    namespace: develop
  spec:
    progressDeadlineSeconds: 600
    replicas: 1
    revisionHistoryLimit: 10
    selector:
      matchLabels:
        app: brok-ui
    strategy:
      rollingUpdate:
        maxSurge: 25%
        maxUnavailable: 25%
      type: RollingUpdate
    template:
      metadata:
        labels:
          app: brok-ui
          deployment: brok-ui
      spec:
        containers:
          - env:
              - name: POD_NAME
                valueFrom:
                  fieldRef:
                    apiVersion: v1
                    fieldPath: metadata.name
              - name: NAMESPACE
                valueFrom:
                  fieldRef:
                    apiVersion: v1
                    fieldPath: metadata.namespace
              - name: LOG_LEVEL
                value: DEBUG
              - name: TZ
                value: Europe/Moscow
            envFrom:
```

```
        - configMapRef:
            name: brok-ui
      image: nexus.ru-central1.internal:5000/brok-ui:develop
      imagePullPolicy: Always
      livenessProbe:
        failureThreshold: 3
        httpGet:
          path: /actuator/health/liveness
          port: 8080
          scheme: HTTP
        initialDelaySeconds: 600
        periodSeconds: 30
        successThreshold: 1
        timeoutSeconds: 10
      name: brok-ui
      ports:
        - containerPort: 8080
          protocol: TCP
      readinessProbe:
        failureThreshold: 1
        httpGet:
          path: /
          port: 8080
          scheme: HTTP
        initialDelaySeconds: 30
        periodSeconds: 30
        successThreshold: 1
        timeoutSeconds: 1
      resources:
        limits:
          cpu: 600m
          memory: 4Gi
        requests:
          cpu: 300m
          memory: 2Gi
      terminationMessagePath: /dev/termination-log
      terminationMessagePolicy: File
    dnsPolicy: ClusterFirst
    imagePullSecrets:
      - name: docker-registry-pull-secret
    nodeSelector:
      node-role.kubernetes.io/worker: worker
```

```
    restartPolicy: Always
    schedulerName: default-scheduler
    securityContext: {}
    serviceAccount: deployer-service
    serviceAccountName: deployer-service
    terminationGracePeriodSeconds: 30
```

2.1.2. Описание template ConfigMap.yml для каждого компонента приложения

1) В качестве описания сущности ConfigMap для компонента **emulator-service** используется следующий template:

```
apiVersion: v1
kind: ConfigMap
metadata:
  labels:
    app: < DC_NAME >
  name: < DC_NAME >
  namespace: < PROJECT >
data:
  spring.kafka.bootstrapServers: < KAFKA_BOOTSTRAP_SERVERS >
  spring.kafka.consumer.groupId: < DC_NAME >
  spring.kafka.consumer.concurrency: '1'
  spring.kafka.consumer.pollTimeout: '5000'
  spring.kafka.listener.ackMode: 'MANUAL'

  spring.datasource.url: < DATASOURCE_URL >
  spring.datasource.username: < DATASOURCE_USERNAME >
  spring.datasource.schema: < DATASOURCE_SCHEMA >
  spring.datasource.maxPoolSize: '5'

  spring.liquibase.change-log: classpath:root-changelog.xml
  spring.liquibase.liquibase-schema: ${spring.datasource.schema}

  common.kafka.consumer.topic: < INPUT_KAFKA_TOPIC >
```

Параметры, которые необходимо задать в template перед deploy ConfigMap компонента **emulator-service**:

<DC_NAME> — имя компонента приложения. Рекомендуемые имена для компонентов приложения: **emulator-service**, однако может быть задано любое имя компонента приложения в соответствии с требованиями нейминга приложений внутри учреждения.

<PROJECT> — пространство имен оркестратора, в которое непосредственно будет осуществляться deploy приложения.

< KAFKA_BOOTSTRAP_SERVERS > — адрес сервера kafka, с которым осуществляется взаимодействие приложения. Адрес Kafka задаётся следующим образом: <SERVER_NAME>:<PORT> где SERVER_NAME — FQDN (доменное имя) виртуальной машины или короткое имя пода с запущенной на нём системой управления базами данных, <PORT> — номер порта по которому осуществляется взаимодействие с брокером сообщений.

< DATASOURCE_URL > — адрес базы данных, используемой приложением Brok в процессе своей штатной работы. Значение переменной формируется следующим образом: jdbc:postgresql://<DB_SERVER_NAME>/<DB_NAME>, где DB_SERVER_NAME — FQDN (доменное имя) виртуальной машины или короткое имя пода с запущенной на нём системой управления базами данных, DB_NAME — название базы данных к которой необходимо подключиться данно-му компоненту приложения.

< DATASOURCE_USERNAME > — имя учетной записи пользователя, с помощью которой необходимо подключаться к базе данных.

< DATASOURCE_SCHEMA > — название схемы в базе данных, которая используется данным компонентом приложения. Рекомендуемое значение: **emulator-service.**

< INPUT_KAFKA_TOPIC > — название топика в брокере сообщений Kafka с которым осуществляется взаимодействие приложения. Рекомендуемое значение: **emulator-service-group**.

2) В качестве описания сущности ConfigMap для компонента **pack-integration-svc-template** используется следующий template

```
        apiVersion: v1
        kind: ConfigMap
        metadata:
          labels:
      app: < DC_NAME >
    name: < DC_NAME >
    namespace: < PROJECT >
        data:
          kafka.bootstrapServers: < KAFKA_BOOTSTRAP_SERVERS >
          kafka.groupConsumer.groupId: pack-integration-svc-template
          kafka.groupConsumer.pollTimeout: '3000'
          kafka.groupConsumer.concurrency: '1'
          kafka.groupConsumer.topic: pack-integration-svc-template-group
          kafka.specificConsumer.groupId: pack-integration-svc-template
```

```
kafka.specificConsumer.pollTimeout: '3000'
kafka.specificConsumer.concurrency: '1'
kafka.specificConsumer.topic: pack-integration-svc-template-specific
kafka.uiConsumer.groupId: pack-integration-svc-template
kafka.uiConsumer.pollTimeout: '3000'
kafka.uiConsumer.concurrency: '1'
kafka.uiConsumer.topic: pack-integration-svc-template-ui
spring.datasource.url: < DATASOURCE_URL >
spring.datasource.username: < DATASOURCE_USERNAME >
spring.datasource.password: < DATASOURCE_PASSWORD >
spring.main.allow-bean-definition-overriding: 'true'
external.url: < EXTERNAL_URL >
external.delay: '< PA_DELAY >'
external.timeout: '< TIMEOUT >'
external.system.attempts.timeout: '10000'
external.system.attempts.count: '< MAX_RETRY_COUNT >'
feign.client.config.default.connectTimeout: '45000'
feign.client.config.default.readTimeout: '45000'
```

3) Параметры, которые необходимо задать в template перед deploy ConfigMap компонента **pack-integration-svc-template**:

<DC_NAME> — имя компонента приложения. Рекомендуемые имена для компонентов приложения: **pack-integration-svc-template**, однако может быть задано любое имя компонента приложения в соответствии с требованиями нейминга приложений внутри учреждения.

<PROJECT> — пространство имен оркестратора, в которое непосредственно будет осуществляться deploy приложения.

< KAFKA_BOOTSTRAP_SERVERS > — доменное имя и порт используемого брокера сообщений kafka.

<DATASOURCE_URL> — адрес базы данных, используемой приложением в процессе своей штатной работы. Значение переменной формируется следующим образом: jdbc:postgresql://<DB_SERVER_NAME>/<DB_NAME>, где DB_SERVER_NAME — FQDN виртуальной машины или короткое имя пода с запущенной на нём системой управления базами данных, DB_NAME – название базы данных к которой необходимо подключиться данному компоненту приложения.

<DATASOURCE_USERNAME> — имя учетной записи пользователя, с помощью которой необходимо подключаться к базе данных.

<DATASOURCE_PASSWORD> — пароль от учетной записи пользователя, с помощью которой необходимо подключаться к базе данных.

< EXTERNAL_URL > — внешний адрес приложения по которому к нему можно получить доступ.

< PA_DELAY > — время ожидания ответа от внешнего клиента в миллисекундах.

< TIMEOUT > — время ожидания до попытки повторного подключения внешнего клиента.

< MAX_RETRY_COUNT > — максимальное количество повторных подключений.

4) В качестве описания сущности ConfigMap для компонента brok используется следующий template:

```
apiVersion: v1
kind: ConfigMap
metadata:
  labels:
    app: < DC_NAME >
  name: < DC_NAME >
  namespace: < PROJECT >
data:
  DATASOURCE_URL: '< DATASOURCE_URL >'
  DATASOURCE_USERNAME: '< DATASOURCE_USERNAME >'
  DATASOURCE_PASSWORD: '< DATASOURCE_PASSWORD >'
  APPLICATION_PATHS_UI: '< APPLICATION_PATHS_UI >'
  APPLICATION_PATHS_CORE: '< APPLICATION_PATHS_CORE >'
  APPLICATION_PATHS_BPM: '< APPLICATION_PATHS_BPM >'
  APPLICATION_AUTHORIZATION_DATABASE_SCHEME: 'APPLICATION_AUTHORIZATION_DATA-
BASE_SCHEME'
  AUDIT_ENABLED: < AUDIT_ENABLED >
  CLEANER_CRONEXPRESSION: < CLEANER_CRONEXPRESSION >
  CLEANER_AUDIT_ENABLED: < CLEANER_AUDIT_ENABLED >
  CLEANER_AUDIT_COUNT: < CLEANER_AUDIT_COUNT >
  CACHE_ENABLED: < CACHE_ENABLED >
  CACHE_PROPERTIES_CLUSTERS_ENABLED: < CACHE_PROPERTIES_CLUSTERS_ENABLED >
  CACHE_PROPERTIES_FAVORITES_ENABLED: < CACHE_PROPERTIES_FAVORITES_ENABLED >
  CACHE_PROPERTIES_TEMPLATES_ENABLED: < CACHE_PROPERTIES_TEMPLATES_ENABLED >
  CACHE_PROPERTIES_TOPICS_ENABLED: < CACHE_PROPERTIES_TOPICS_ENABLED >
```

5) Параметры, которые необходимо задать в template перед deploy ConfigMap компонента **brok**:

<DC_NAME> — имя компонента приложения. Рекомендуемые имена для компонентов приложения: brok/brok-bpm/brok-auth/brok-ui, однако может быть задано любое имя компонента приложения в соответствии с требованиями нейминга приложений внутри учреждения.

<PROJECT> — пространство имен оркестратора, в которое непосредственно будет осуществляться deploy приложения.

<DATASOURCE_URL> — адрес базы данных, используемой приложением Brok в процессе своей штатной работы. Значение переменной формируется следующим образом: ***jdbc:postgresql://<DB_SERVER_NAME>/<DB_NAME>,*** где ***DB_SERVER_NAME*** — FQDN виртуальной машины или короткое имя пода с запущенной на нём системой управления базами данных, ***DB_NAME*** — название базы данных, к которой необходимо подключиться данному компоненту приложения.

<DATASOURCE_USERNAME> — имя учетной записи пользователя, с помощью которой необходимо подключаться к базе данных.

< APPLICATION_AUTHORIZATION_DATABASE_SCHEME > — название схемы в базе данных, к которой подключается сервис.

<APPLICATION_PATHS_UI> — переменная, определяющая путь до сущности service компонента UI. Значение переменной формируется следующим образом: ***http://<UI_SERVICE_NAME>,*** где ***UI_SERVICE_NAME*** — название сущности service, отвечающий за балансировку запросов, направленных к контейнеру, в котором функционирует компонент UI. Рекомендуемое значение данной переменной: ***brok-ui***, однако может быть задано любое иное название, которое используется для сущности service компонента UI.

< APPLICATION_PATHS_CORE > — переменная определяющая путь до сущности service компонента UI. Значение переменной формируется следующим образом: ***http://CORE_SERVICE_NAME>,*** где ***CORE_SERVICE_NAME*** — название сущности service, отвечающий за балансировку запросов, направленных к контейнеру, в котором функционирует компонент COREI. Рекомендуемое значение данной переменной: ***brok-core***, однако может быть задано любое иное название, которое используется для сущности service компонента CORE.

< APPLICATION_PATHS_BPM > — переменная определяющая путь до сущности service компонента UI. Значение переменной формируется следующим образом: ***http://<BPM_SERVICE_NAME>,*** где ***BPM_SERVICE_NAME*** — название сущности service, отвечающий за балансировку запросов, направленных к контейнеру, в котором функционирует компонент BPM. Рекомендуемое значение данной переменной: ***brok-BPM***, однако может быть задано любое иное название, которое используется для сущности service компонента BPM.

< CACHE_ENABLED > — параметр, отвечающий за включение/отключение функции кэширования используемой в приложении в целом. Принимает значение ***true/false.***

< CACHE_PROPERTIES_CLUSTERS_ENABLED > — параметр, отвечающий за включение/отключение кэширования брокеров. Принимает значение ***true/false.***

< CACHE_PROPERTIES_TOPICS_ENABLE > — параметр, отвечающий за включение/отключение кэширования топиков. Принимает значение ***true/false.***

< CACHE_PROPERTIES_FAVORITES_ENABLED > — параметр, отвечающий за включение/отключение кэширования избранных. Принимает значение ***true/false.***

< CACHE_PROPERTIES_TEMPLATES_ENABLED > — параметр, отвечающий за включение/отключение кэширования шаблонов. Принимает значение ***true/false.***

< AUDIT_ENABLED > — параметр, отвечающий за включение/отключение логирования действий пользователя в приложении. Принимает значение ***true/false.***

< CLEANER_AUDIT_ENABLED > — параметр, отвечающий за включение/отключение очистки базы данных от старых записей, содержащих информацию о действиях пользователей в приложении. Принимает значение ***true/false.***

< CLEANER_AUDIT_COUNT > — параметр, определяющий минимальное количество записей действий пользователей (для каждого пользователя), которое останется после очистки базы данных.

< CLEANER_CRONEXPRESSION > — параметр, определяющий расписание очистки базы данных от старых записей, содержащих информацию о действиях пользователей в приложении.

6) В качестве описания сущности ConfigMap для компонента brok-bpm используется следующий template:

```
apiVersion: v1
kind: ConfigMap
metadata:
  labels:
    app: < DC_NAME >
  name: < DC_NAME >
  namespace: < PROJECT >
data:
  SECRET_NAME: < DC_NAME >
  DATASOURCE_URL: < DATASOURCE_URL >
  DATASOURCE_USERNAME: < DATASOURCE_USERNAME >
  DATASOURCE_SCHEMA: < DATASOURCE_SCHEMA >
  APPLICATION_PATHS_UI: < APPLICATION_PATHS_UI >
  APPLICATION_PATHS_CORE: < APPLICATION_PATHS_CORE >
```

7) Параметры, которые необходимо задать в template перед deploy ConfigMap компонента **brok-bpm**

<DC_NAME> — имя компонента приложения. Рекомендуемые имена для компонентов приложения: brok/brok-bpm/brok-auth/brok-ui, однако может быть задано любое имя компонента приложения в соответствии с требованиями нейминга приложений внутри учреждения.

<PROJECT> — пространство имен оркестратора, в которое непосредственно будет осуществляться deploy приложения.

<APPLICATION_PATHS_UI> — переменная, определяющая путь до сущности service компонента UI. Значение переменной формируется следующим образом: ***http://<UI_SERVICE_NAME>,*** где ***UI_SERVICE_NAME*** — название сущности service, отвечающий за балансировку запросов, направленных к контейнеру, в котором функционирует компонент UI. Рекомендуемое значение данной переменной: ***brok-ui***, однако может быть задано любое иное название, которое используется для сущности service компонента UI.

< APPLICATION_PATHS_CORE > — переменная, определяющая путь до сущности service компонента UI. Значение переменной формируется следующим образом: ***http://CORE_SERVICE_NAME>,*** где ***CORE_SERVICE_NAME*** — название сущности service, отвечающий за балансировку запросов, направленных к контейнеру, в котором функционирует компонент COREI. Рекомендуемое значение данной переменной: ***brok-core***, однако может быть задано любое иное название, которое используется для сущности service компонента CORE.

<DATASOURCE_URL> — адрес базы данных, используемой приложением Brok в процессе своей штатной работы. Значение переменной формируется следующим образом: ***jdbc:postgresql://<DB_SERVER_NAME>/<DB_NAME>,*** где ***DB_SERVER_NAME*** — FQDN виртуальной машины или короткое имя пода с запущенной на нём системой управления базами данных, ***DB_NAME*** — название базы данных, к которой необходимо подключиться данному компоненту приложения.

<DATASOURCE_USERNAME> — имя учетной записи пользователя, с помощью которой необходимо подключаться к базе данных.

< DATASOURCE _SCHEMA> — название схемы в базе данных, которая используется данным компонентом приложения. Значение по умолчанию: ***brok_bpm***.

8) В качестве описания сущности ConfigMap для компонента **brok-auth** используется следующий template:

```
apiVersion: v1
kind: ConfigMap
metadata:
```

```
    labels:
      app: < DC_NAME >
    name: < DC_NAME >
    namespace: < PROJECT >
  data:
    SECRET_NAME: < DC_NAME >
    KEYCLOAK_DATABASE_HOST : '< DB_ADDR >'
    KEYCLOAK_DATABASE_PORT: '< DB_PORT >'
    KEYCLOAK_DATABASE_VENDOR: '< DB_VENDOR >'
    KEYCLOAK_DATABASE_NAME: '< DB_DATABASE >'
    KEYCLOAK_DATABASE_SCHEM: '< DB_SCHEMA >'
    KEYCLOAK_ADMIN: '< KEYCLOAK_USER >'
    KEYCLOAK_ENABLE_HEALTH_ENDPOINTS: < KEYCLOAK_ENABLE_HEALTH_ENDPOINTS >
    PROXY_ADDRESS_FORWARDING: '< PROXY_ADDRESS_FORWARDING >'
    KEYCLOAK_LOGLEVEL: '< KEYCLOAK_LOGLEVEL >'
    ROOT_LOGLEVEL: '< ROOT_LOGLEVEL >'
```

9) Параметры, которые необходимо задать в template перед deploy ConfigMap компонента **brok-auth**:

<DC_NAME> — имя компонента приложения. Рекомендуемые имена для компонентов приложения: brok/brok-bpm/brok-auth/brok-ui, однако может быть задано любое имя компонента приложения в соответствии с требованиями нейминга приложений внутри учреждения.

<PROJECT> — пространство имен оркестратора, в которое непосредственно будет осуществляться deploy приложения.

<DB_ADDR> — адрес базы данных, используемой приложением Brok в процессе своей штатной работы. В переменную указывается FQDN виртуальной машины или короткое имя сущности service, отвечающего за балансировку запросов, направленных к контейнеру с запущенной на нём системой управления базами данных.

<DB_PORT> — порт, по которому доступна база данных. Дефолтное значение: 5432

<DB_VENDOR> — параметр, определяющий систему управления базами данных, с которой взаимодействует данный компонент приложения. Значение по умолчанию: ***postgres***. Работа компонента приложения с иными отличными от значения по умолчанию системами управления базами данных не проверялась.

< DB_DATABASE > — параметр, определяющий название базы данных к которой необходимо подключиться данному компоненту приложения.

<DB_SCHEMA> — название схемы в базе данных, которая используется данным компонентом приложения. Значение по умолчанию: ***brok_auth***.

< KEYCLOAK_USER > — параметр, с помощью которого задаётся логин пользователя, используемый для авторизации в UI данного компонента приложения. Значение переменной определяется индивидуально для каждой интеграции приложения, поэтому значение по умолчанию отсутствует.

< PROXY_ADDRESS_FORWARDING > — параметр, с помощью которого позволяется/запрещается использовать информацию, предоставленную прокси, чтобы корректно определить IP-адрес клиента и другие параметры запроса. Значение по умолчанию: ***true.*** Работа компонента приложения в штатном режиме, с иными, отличными от значения по умолчанию значением переменной не гарантирована.

< KEYCLOAK_ENABLE_HEALTH_ENDPOINTS > — фиче-тоггл, отвечающий за включение/отключение специальных адресов, используемых для проверки liveness/readeneless probe.

< KEYCLOAK_LOGLEVEL > — параметр, определяющий уровень логирования данного компонента приложения. Значение по умолчанию: ***INFO.***

< ROOT_LOGLEVEL > — параметр, определяющий уровень логирования базового контейнера в котором запущен данного компонента приложения. Значение по умолчанию: ***INFO.***

10) В качестве описания сущности ConfigMap для компонента ***brok-ui*** используется следующий template:

```
apiVersion: v1
kind: ConfigMap
metadata:
  labels:
    app: < DC_NAME >
  name: < DC_NAME >
  namespace: < PROJECT >
data:
  AUTH_TYPE: < AUTH_TYPE >
  AUTH_OAUTH2_RESOURCESERVER_JWT_ISSUERURI: '< AUTH_OAUTH2_RESOURCE-
SERVER_JWT_ISSUERURI >'
  AUTH_OAUTH2_RESOURCESERVER_JWT_JWKSETURI: '< AUTH_OAUTH2_RESOURCE-
SERVER_JWT_JWKSETURI >'
```

11) Параметры, которые необходимо задать в template перед deploy ConfigMap компонента brok-ui:

<DC_NAME> — имя компонента приложения. Рекомендуемые имена для компонентов приложе-ния: brok/brok-bpm/brok-auth/brok-ui, однако может быть задано любое имя компонента приложения в соответствии с требованиями нейминга приложений внутри учреждения.

<PROJECT> — пространство имен оркестратора, в которое непосредственно будет осуществляться deploy приложения.

< AUTH_TYPE > — параметр, определяющий тип аутентификации, используемый данным компо-нентом приложения. Значение по умолчанию: KEYCLOAK. Работа компонента приложения в штатном режиме, с иными, отличными от значения по умолчанию значением переменной не проверялась.

< AUTH_OAUTH2_RESOURCESERVER_JWT_ISSUERURI > — параметр, позволяющий указать URI, который идентифицирует компонент приложения, отвечающий за выдачу токены JWT, в рамках процесса аутентификации и авторизации через OAuth 2.0. Значение по умолчанию: http://brok-auth/auth/realms/brok-realm

< AUTH_OAUTH2_RESOURCESERVER_JWT_JWKSETURI > — параметр, позволяющий указать URI, который идентифицирует компонент приложения, отвечающий за выдачу открытых ключей, используемых для проверки подписи JWT токенов. Значение по умолчанию: http://brok-auth/auth/realms/ brok -realm/protocol/openid-connect/certs.

12) Примеры template для каждого компонента приложения.

a) ConfigMap emulator-service-group

```
       apiVersion: v1
       data:
         common.kafka.consumer.topic: emulator-service-group
         spring.datasource.maxPoolSize: '5'
         spring.datasource.schema: emulator_service
         spring.datasource.url: jdbc:postgresql://postgres.ru-central1.inter-
nal:5432/javax_db1
         spring.datasource.username: emulator_service
         spring.kafka.bootstrapServers: kafka.ru-central1.internal:9092
         spring.kafka.consumer.concurrency: '1'
         spring.kafka.consumer.groupId: emulator-service
         spring.kafka.consumer.pollTimeout: '5000'
         spring.kafka.listener.ackMode: MANUAL
         spring.liquibase.change-log: classpath:root-changelog.xml
         spring.liquibase.liquibase-schema: ${spring.datasource.schema}
```

```
        kind: ConfigMap
        metadata:
          labels:
            app: emulator-service
          name: emulator-service
          namespace: develop
```

б) ConfigMap pack-integration-svc-template

```
        apiVersion: v1
        data:
          external.delay: '1000'
          external.system.attempts.count: '5'
          external.system.attempts.timeout: '10000'
          external.timeout: '45'
          external.url: http://emulator-service/test/pack-integration-svc-tem-
plate/common/external
          feign.client.config.default.connectTimeout: '45000'
          feign.client.config.default.readTimeout: '45000'
          kafka.bootstrapServers: kafka.ru-central1.internal:9092
          kafka.groupConsumer.concurrency: '1'
          kafka.groupConsumer.groupId: pack-integration-svc-template
          kafka.groupConsumer.pollTimeout: '3000'
          kafka.groupConsumer.topic: pack-integration-svc-template-group
          kafka.specificConsumer.concurrency: '1'
          kafka.specificConsumer.groupId: pack-integration-svc-template
          kafka.specificConsumer.pollTimeout: '3000'
          kafka.specificConsumer.topic: pack-integration-svc-template-specific
          kafka.uiConsumer.concurrency: '1'
          kafka.uiConsumer.groupId: pack-integration-svc-template
          kafka.uiConsumer.pollTimeout: '3000'
          kafka.uiConsumer.topic: pack-integration-svc-template-ui
          spring.datasource.password: akwdbkawbdwiuyeg@&@(I&(HG!BI-
HawjdbkawdlO@@!@jnK@))
          spring.datasource.url: jdbc:postgresql://postgres.ru-central1.inter-
nal:5432/javax_db1
          spring.datasource.username: javax_rbg_adapter
          spring.main.allow-bean-definition-overriding: 'true'
        kind: ConfigMap
        metadata:
          labels:
```

```
      app: pack-integration-svc-template
    name: pack-integration-svc-template
    namespace: develop
```

в) ConfigMap brok

```
- apiVersion: v1
  kind: ConfigMap
  metadata:
    labels:
      app: brok
    name: brok
    namespace: develop
  data:
    AUDIT_ENABLED: ‘true’
   CLEANER_AUDIT_COUNT: '100'
   CLEANER_AUDIT_ENABLED: 'true'
   CLEANER_CRONEXPRESSION: 0 0 0 * * ?
    DATASOURCE_URL: 'jdbc:postgresql://pgsql.ru-central1.internal/brok'
    APPLICATION_PATHS_UI: 'http://brok-ui/api'
    CACHE_ENABLED: 'true'
    CACHE_PROPERTIES_CLUSTERS_ENABLED: 'true'
    CACHE_PROPERTIES_TOPICS_ENABLED: 'true'
    CACHE_PROPERTIES_FAVORITES_ENABLED: 'true'
    CACHE_PROPERTIES_TEMPLATES_ENABLED: 'true'
    AUDIT_ENABLED: 'true'
    CLEANER_AUDIT_COUNT: '100'
    CLEANER_AUDIT_ENABLED: 'true'
    CLEANER_CRONEXPRESSION: 0 0 0 * * ?
```

г) ConfigMap brok-bpm

```
  apiVersion: v1
  data:
    APPLICATION_PATHS_CORE: http://brok-core
    APPLICATION_PATHS_UI: http://brok-ui
    CAMUNDA_BPM_ADMIN_USER_ID: admin
    CAMUNDA_BPM_ADMIN_USER_PASSWORD: WKLdnlwadajwdni72y3g3b@HO@bnjflscalscblec
    DATASOURCE_PASSWORD: EFNAEKLFNO£*H£$HFBIOUQJWENFQELFJN
    DATASOURCE_SCHEMA: brok_bpm
    DATASOURCE_URL: jdbc:postgresql://postgresx.ru-central1.internal/brok_demo
    DATASOURCE_USERNAME: brok_demo
```

```
    SECRET_NAME: brok-bpm
  kind: ConfigMap
  metadata:
    annotations:
      meta.helm.sh/release-name: brok
      meta.helm.sh/release-namespace: develop
    labels:
      app: brok-bpm
      app.kubernetes.io/managed-by: Helm
    name: brok-bpm
    namespace: develop
```

д) ConfigMap brok-auth

```
- apiVersion: v1
  kind: ConfigMap
  metadata:
    labels:
      app: brok-auth
    name: brok-auth
    namespace: develop
  data:
    SECRET_NAME: brok-auth
    DB_ADDR: 'pgsql.ru-central1.internal'
    DB_PORT: '5432'
    DB_VENDOR: 'postgres'
    DB_DATABASE: 'brok'
    DB_SCHEMA: 'brok_auth'
    DB_USER: 'brok'
    PROXY_ADDRESS_FORWARDING: 'True'
    START_PAGE_HOST: 'http://ui.develop.apps.dev.k8s /'
    KEYCLOAK_LOGLEVEL: 'INFO'
    ROOT_LOGLEVEL: 'INFO'
```

е) ConfigMap brok-ui

```
- apiVersion: v1
  kind: ConfigMap
  metadata:
    labels:
      app: brok-ui
    name: brok-ui
    namespace: develop
```

```
      data:
        AUTH_TYPE: KEYCLOAK
        AUTH_OAUTH2_RESOURCESERVER_JWT_ISSUERURI: 'http://brok-
  auth/auth/realms/brok-realm'
        AUTH_OAUTH2_RESOURCESERVER_JWT_JWKSETURI: 'http://brok-
  auth/auth/realms/brok-realm/protocol/openid-connect/certs'
```

2.1.3. Описание template Secret.yml для каждого компонента приложения

1) В качестве описания сущности Secret для компонента **emulator-service** используется следующий template:

```
      apiVersion: v1
      kind: Secret
      metadata:
        labels:
          app: < DC_NAME >
        name: < DC_NAME >
        namespace: < PROJECT >
      stringData:
        spring.datasource.password: {{ DATASOURCE_PASSWORD }}
```

2) Параметры, которые необходимо задать в template перед deploy Secret компонента **emulator-service**:

<DC_NAME> — имя компонента приложения. Рекомендуемое имя для приложения: **emulator-service**, однако может быть задано любое имя компонента приложения в соответствии с требованиями нейминга приложений внутри учреждения.

<PROJECT> — пространство имен оркестратора, в которое непосредственно будет осуществляться deploy приложения.

< DATASOURCE_PASSWORD > — пароль от учетной записи пользователя, с помощью которой необходимо подключаться к базе данных. Значение переменной определяется индивидуально для каждой интеграции приложения, поэтому значение по умолчанию отсутствует.

3) В качестве описания сущности Secret для компонента **bpm-orchestration-service** используется следующий template

```
      apiVersion: v1
      kind: Secret
      metadata:
```

```
    labels:
      app: < DC_NAME >
    name: < DC_NAME >
    namespace: < PROJECT >
  stringData:
    spring.main.allow-bean-definition-overriding: 'true'
    spring.datasource.password: < DATASOURCE_PASSWORD >
    spring.datasource.url: < DATASOURCE_URL >
    spring.datasource.username: < DATASOURCE_USERNAME >
    camunda.ldap.plugin.managerPassword: < CAMUNDA_LDAP_MANAGER_PASSWORD >
```

4) Параметры, которые необходимо задать в template перед deploy Secret компонента **bpm-orchestration-service**:

<DC_NAME> — имя компонента приложения. Рекомендуемое имя для приложения: **bpm-orchestration-service**, однако может быть задано любое имя компонента приложения в соответствии с требованиями нейминга приложений внутри учреждения.

<PROJECT> — пространство имен оркестратора, в которое непосредственно будет осуществляться deploy приложения.

< DATASOURCE_URL > — адрес базы данных, используемой приложением в процессе своей штатной работы. Значение переменной формируется следующим образом: jdbc:postgresql://<DB_SERVER_NAME>/<DB_NAME>, где DB_SERVER_NAME — FQDN (доменное имя) виртуальной машины или короткое имя пода с запущенной на нём системой управления базами данных, DB_NAME — название базы данных к которой необходимо подключиться данному компоненту приложения.

< DATASOURCE_USERNAME > — имя учетной записи пользователя с помощью которой необходимо подключаться к базе данных.

< DATASOURCE_PASSWORD > — пароль от учетной записи пользователя с помощью которой необходимо подключаться к базе данных. Значение переменной определяется индивидуально для каждой интеграции приложения, поэтому значение по умолчанию отсутствует.

< CAMUNDA_LDAP_MANAGER_PASSWORD > — пароль от учетной записи пользователя с помощью которой можно взаимодействовать с сервисом .

5) В качестве описания сущности Secret для компонента brok используется следующий template:

```
apiVersion: v1
kind: Secret
metadata:
```

```
    labels:
          app: < DC_NAME >
        name: < DC_NAME >
        namespace: < PROJECT >
  stringData:
    LICENSE_KEY: < LICENSE_KEY >
    LICENSE_TYPE: < LICENSE_TYPE >
    LICENSE_KUBERNETES_HOST: < LICENSE_TYPE >
    LICENSE_KUBERNETES_NAMESPACE:< LICENSE_KUBERNETES_NAMESPACE >
    LICENSE_KUBERNETES_API_VERSION: < LICENSE_KUBERNETES_API_VERSION >
    LICENSE_KUBERNETES_TOKEN: < LICENSE_KUBERNETES_TOKEN >
    DATASOURCE_PASSWORD: < DB_PASSWORD >
    DATASOURCE_USERNAME: < DB_USER >
    BOT_NAME: < BOT_NAME >
    BOT_TOKEN: < BOT_TOKEN>
    BOT_CHATID: <. BOT_CHATID >
    BOT_ENABLED: < BOT_ENABLED >
```

6) Параметры, которые необходимо задать в template перед deploy Secret компонента brok:

<DC_NAME> — имя компонента приложения. Рекомендуемые имена для компонентов приложе-ния: brok/brok-bpm/brok-auth/brok-ui, однако может быть задано любое имя компонента приложения в соответствии с требованиями нейминга приложений внутри учреждения.

<PROJECT> — пространство имен оркестратора, в которое непосредственно будет осуществляться deploy приложения.

<LICENSE_KEY> — ключ лицензии, в соответствии с которым определены правила использования приложения. Предаётся отдельным файлом, в момент передачи образов приложения. Обновление лицензии запрашивается у разработчиков приложения.

< LICENSE_TYPE > — тип лицензии. Возможные варианты: DEFAULT и KUBERNETES. Для запуска в оркестраторе используется лицензия типа KUBERNETES.

< LICENSE_KUBERNETES_NAMESPACE > — название целевого namespace в котором будет запущено приложения.

< LICENSE_KUBERNETES_API_VERSION > — apiVersion сущности Namespace в оркестраторе. При деплое в оркестратор k8s необходимо задать значение «v1».

< LICENSE_KUBERNETES_TOKEN > — токен доступа сервисного аккаунта используемый для проверки лицензии.

<DB_USER> — имя учетной записи пользователя, с помощью которой необходимо подключаться к базе данных. Значение переменной определяется индивидуально для каждой интеграции приложения, поэтому значение по умолчанию отсутствует.

<DB_PASSWORD> — пароль от учетной записи пользователя, с помощью которой необходимо подключаться к базе данных. Значение переменной определяется индивидуально для каждой интеграции приложения, поэтому значение по умолчанию отсутствует.

< BOT_NAME > — имя чат бота технической поддержки. Для данной переменной необходимо задать следующее значение: UiKafkaBot.

< BOT_TOKEN> — токен чат бота технической поддержки. Для данной переменной необходимо задать следующее значение: 6462748878:AAHNx2Cdb73MPRgL1uNd5hAzdPMoHM6w2Q.

<. BOT_CHATID > — ID чат бота технической поддержки. Для данной переменной необходимо задать следующее значение: -1001939060949.

< BOT_ENABLED > — фиче-тоггл, управляющий наличием подключения с телеграм-ботом. Переменная принимает булевы значения true/false.

7) В качестве описания сущности Secret для компонента brok-bpm используется следующий template:

```
apiVersion: v1
kind: Secret
metadata:
  labels:
        app: < DC_NAME >
      name: < DC_NAME >
      namespace: < PROJECT >
stringData:
  DATASOURCE_USERNAME: <DB_USER>
  DATASOURCE_PASSWORD: <DB_PASSWORD>
  CAMUNDA_BPM_ADMIN_USER_ID: < CAMUNDA_BPM_ADMIN_USER_ID >
  CAMUNDA_BPM_ADMIN_USER_PASSWORD: < CAMUNDA_BPM_ADMIN_USER_PASSWORD >
```

8) Параметры, которые необходимо задать в template перед deploy Secret компонента brok-bpm:

<DC_NAME> — имя компонента приложения. Рекомендуемые имена для компонентов приложе-ния: brok/brok-bpm/brok-auth/brok-ui, однако может быть задано любое имя компонента при-ложения в соответствии с требованиями нейминга приложений внутри учреждения.

<PROJECT> — пространство имен оркестратора, в которое непосредственно будет осуществляться deploy приложения.

<DB_USER> — имя учетной записи пользователя, с помощью которой необходимо подключаться к базе данных. Значение переменной определяется индивидуально для каждой интеграции при-ложения, поэтому значение по умолчанию отсутствует.

<DB_PASSWORD> — пароль от учетной записи пользователя, с помощью которой необходимо подключаться к базе данных. Значение переменной определяется индивидуально для каждой интеграции приложения, поэтому значение по умолчанию отсутствует.

<CAMUNDA_BPM_ADMIN_USER_ID> — техническая учетная запись, с помощью которой осуществляется взаимодействие с приложением.

< CAMUNDA_BPM_ADMIN_USER_PASSWORD > — пароль от технической учетной записи, с помощью которой осуществляется взаимодействие с приложением.

9) В качестве описания сущности Secret для компонента brok-auth используется следующий template:

```
- apiVersion: v1
  kind: Secret
  metadata:
    labels:
      app: < DC_NAME >
    name: < DC_NAME >
    namespace: < PROJECT >
  stringData:
    KEYCLOAK_DATABASE_USER: < DB_USER >
    KEYCLOAK_DATABASE_PASSWORD: < DB_PASSWORD >
    KEYCLOAK_ADMIN: < KEYCLOAK_USER >
    KEYCLOAK_ADMIN_PASSWORD: < KEYCLOAK_PASSWORD >
```

10) Параметры, которые необходимо задать в template перед deploy Secret компонента brok-auth:

<DC_NAME> — имя компонента приложения. Рекомендуемые имена для компонентов приложения: brok/brok-bpm/brok-auth/brok-ui, однако может быть задано любое имя компонента приложения в соответствии с требованиями нейминга приложений внутри учреждения.

<PROJECT> — пространство имен оркестратора, в которое непосредственно будет осуществляться deploy приложения.

<DB_USER> — имя учетной записи пользователя, с помощью которой необходимо подключаться к базе данных. Значение переменной определяется индивидуально для каждой интеграции приложения, поэтому значение по умолчанию отсутствует.

<DB_PASSWORD> — пароль от учетной записи пользователя, с помощью которой необходимо подключаться к базе данных. Значение переменной определяется индивидуально для каждой интеграции приложения, поэтому значение по умолчанию отсутствует.

< KEYCLOAK_USER > — параметр, с помощью которого задаётся логин пользователя, используемый для авторизации в UI данного компонента приложения. Значение переменной определяется индивидуально для каждой интеграции приложения, поэтому значение по умолчанию отсутствует.

< KEYCLOAK_PASSWORD > — параметр, с помощью которого задаётся пароль, используемый для авторизации в UI данного компонента приложения. Значение переменной определяется индивидуально для каждой интеграции приложения, поэтому значение по умолчанию отсутствует.

11) Так же для компонента brok-ui создаётся две сущности Secret в случае, если необходимо использовать защищенное TLS соединение от приложения до брокеров, к которым оно подключается. Для создания двух Secret используется следующие template:

```
- apiVersion: v1
  kind: Secret
  metadata:
    name: brok-client
    namespace: < PROJECT >
  data:
    client.truststore.jks: < BROK_CLIENT_TRUSTSTORE_JKS >
    client.keystore.jks: < BROK_CLIENT_KEYSTORE_JKS >

- apiVersion: v1
  kind: Secret
  metadata:
    name: brok-server
    namespace: < PROJECT >
  data:
    server.truststore.jks: < BROK_SERVER_TRUSTSTORE_JKS >
    server.keystore.jks: < BROK_SERVER_KEYSTORE_JKS >
```

12) Параметры, которые необходимо задать в template перед deploy Secret компонента brok-ui:

<DC_NAME> — имя компонента приложения. Рекомендуемые имена для компонентов приложения: brok/brok-bpm/brok-auth/brok-ui, однако может быть задано любое имя компонента приложения в соответствии с требованиями нейминга приложений внутри учреждения.

<PROJECT> — пространство имен оркестратора, в которое непосредственно будет осуществляться deploy приложения.

< BROK_CLIENT_TRUSTSTORE_JKS > — переменная, содержащая в себе информацию о доверенном хранилище клиентских сертификатов, которая записана в кодировке base64.

< BROK_CLIENT_KEYSTORE_JKS > — переменная содержащая в себе информацию о ключевыхх материалах (частные ключи и сертификаты) клиентов, которая записана в кодировке base64.

< BROK_SERVER_TRUSTSTORE_JKS > — переменная, содержащая в себе информацию о доверенном хранилище серверных сертификатов, которая записана в кодировке base64.

< BROK_SERVER_KEYSTORE_JKS > — переменная, содержащая в себе информацию о ключевыхх материалах (частные ключи и сертификаты) серверов, которая записана в кодировке base64.

<LICENSE_KEY> — ключ лицензии, в соответствии с которым определены правила использования приложения. Предаётся отдельным файлом, в момент передачи образов приложения. Обновление лицензии запрашивается у разработчиков приложения.

13) Примеры template для каждого компонента приложения.

а) Secret emulator-service

```
apiVersion: v1
kind: Secret
metadata:
  labels:
    app: emulator-service
  name: emulator-service
  namespace: develop
stringData:
  spring.datasource.password: lenfeL£R839qhfON@@jr0(*J$R)TQI)q2(@
```

б) Secret bpm-orchestration-service

```
apiVersion: v1
kind: Secret
metadata:
  labels:
```

```
            app: bpm-orchestration-service
          name: bpm-orchestration-service
          namespace: develop
        stringData:
          spring.main.allow-bean-definition-overriding: 'true'
          spring.datasource.password: awdLNwdlQ£UB£IQBFO@2bfqh2qkK@@@
          spring.datasource.url: jdbc:postgresql://postgres.ru-central1.inter-
nal:5432/javax_db1
          spring.datasource.username: bpm_orchestration_service
          camunda.ldap.plugin.managerPassword: awdklJN@*@@**@BFkebvskenv99221
```

в) Secret brok

```
          - apiVersion: v1
            kind: Secret
            metadata:
              labels:
                app: brok
              name: brok
              namespace: develop
            stringData:
              LICENSE_KEY: "PGxpY2Vuc2U+CjxuYW1lPkF4ZW5peC-
BCck9rPC9uYW1lPgo8ZGVzY3JpcHRpb24+wadawdawdclaenf:KEJAWBFpIBEF JHSLEv sljveb
lsejfbnpae;sjnfPAK:SEJVnsepjvbJSLEBHfvELVBejlv+MDEuMTIuMjAzMyAyMzo1OTo1OT-
wvZXhwaXJhdGlvbkRhdGU+Cjx0eXBlPktVQkVSTkVURVM8L3R5cGU+Cjxwcm9wZXJ0aWVzPgo8dWlkPmJlM
WU4YjNlLTFhZTItNDdjMy1iNmM4LTU1ZDU3NjU5ZmI4OT-
wvdWlkPgo8L3Byb3BlcnRpZXM+CjxzaWduYXR1cmU+MzU0QkVCNDgwQjJCRTQzRjMwMEYzQkQ4RjZFRjJBM
zM8L3NpZ25hdHVyZT4KPC9saWNlbnNlPgo=”
              LICENSE_TYPE: KUBERNETES
              LICENSE_KUBERNETES_HOST: https://11.222.33.444:6443
              LICENSE_KUBERNETES_NAMESPACE: develop
              LICENSE_KUBERNETES_API_VERSION: v1
              LICENSE_KUBERNETES_TOKEN: eyJhbGciOiJSUzI1NiI-
sImtpZCI6IlVmOHR0RTBKT3NDd2lTdWN0UFM1ZXFTbGZyV0hIelRVYkt2d2dhenM2LTAifQ.eyJpc3Mi-
OiJrdWJlcm5ldGVzL3NlcnZpY2VhY2NvdW50Iiwia3ViZXJuZXRlcy5pby9zZXJ2aWNlY-
WNjb3VudC9uYW1lc3BhY2UiOiJkZW1vIiwia3ViZXJuZXRlcy5pby9zZXJ2aWNlY-
WNjb3VudC9zZWNyZawd;akweflNL”EAf vmsdawf/eljv r;dksfmaw. C,svdkjwlad.m,scd mnd-
faeldkmsc.,d vdvnksekdmc vfndfdselmc vfnVjLTgxYWVlOTU1MThmMSIs-
InN1YiI6InN5c3RlbTpzZXJ2aWNlYWNjb3VudDpkZW1vOmJyb2stc2EifQ.3QTR-
d1ghiccf4Wi37fNPbk7XOKQI-
AAnZA2o73_1eGXCaayX_idt4W22Q7B_se3sSA5SlGW2pIVv4EYBRWieDAqFL_S_vPlf9N-8HBr-
vUa8sIWWDBQG6J-
```

```
8B5n2s4H7HsXXFC0W_I401SL1ZfkPIR6n1O8QvnYlzQB7VURBWITh7uRUZxKopX-
fOQdPdDR4JexZxq7PwDG-WmwgbuHq7vA9rR9oaGj3rfGdoI7lLQbP9beoRoGha4mrJXaC-
dAOcZdRs6yUS7JhSMZ_moysoCpNjyiOfcD8Qz14NpLldpYg7iM5oYDIchNYfYkQjOmO8dB5V80pC6j-
wGrJE9sfFlpkw
             DATASOURCE_PASSWORD: apeofml’wsegkn vberfd;kjh£I*HBIPFWJKEVSniu
             DATASOURCE_USERNAME: postgres
             BOT_NAME: UiKafkaBot
             BOT_TOKEN: 6462748878:AAHNx2Cdb73MPRgL1uNd5hAzdPMo-HM6w2Q
             BOT_CHATID: '-1001939060949'
             BOT_ENABLED: true
```

г) Secret brok-bpm

```
        - apiVersion: v1
          kind: Secret
          metadata:
            labels:
              app: brok-auth
            name: brok-auth
            namespace: develop
          stringData:
            DATASOURCE_USERNAME: brok_develop
            DATASOURCE_PASSWORD: alefjn:KQH£RUIF$BP(£*RHFBUIVCJVBKESF
            CAMUNDA_BPM_ADMIN_USER_ID: admin
           CAMUNDA_BPM_ADMIN_USER_PASSWORD: alkewfnKWJBFLIEAHv@@ERAcksnm
```

д) Secret brok-auth

```
          - apiVersion: v1
            kind: Secret
            metadata:
              labels:
                app: brok-auth
              name: brok-auth
              namespace: develop
            stringData:
              KEYCLOAK_DATABASE_USER: brok
              KEYCLOAK_DATABASE_PASSWORD: 'KJenjY3*kanLe)'
              KEYCLOAK_ADMIN: admin
              KEYCLOAK_ADMIN_PASSWORD: 'admin'
```

е) Secret brok-ui

```
  - apiVersion: v1
    kind: Secret
    metadata:
      name: brok-ui
      namespace: develop
    data:
      client.truststore.jks: MIIGLgIBAzCCBecGCSqGSIb3DQEHAaC-
CBdgEggXUMIIF0DCCBcwGCSqGSIb3DQEHBqCCBb0wggW5AgEAMIIFsgYJKoZIhvc-
NAQcBMCkGCiqGSIb3DQEMAQYwGwQUOyAHoMkvbj5oYdPCDq+xvV5Di9wCAwDDUIC-
CBXgCh509MdZNnEDPKVRzz++LSOWXiblXuRuFcEn-
xxn6/eZWr9xnBY1kvPaiyl0zBQi6Kt33+rgb7XwBFEhhH1QLYnlxtD4P7SPKQCLQda-
TUh/Vm+bVseR3v1WaO8EjtI95nxSKG6MGvq+j4+L76au/h17UJLR8OZRQW-
povqcfJd+F1oofQ6aZDI2c9DYwYeAQVsj+iM/PCLxXqpqnedvsaEi+SexeEWr+dd0c3lbDw-
CaDo/P8ZTGWAHc20pMIStMAEirPOu8ZvV9KvpJytrWoIVH3/ZoygCwic0zDr-
Nelw4cBcydpJPGWD3jOJVo1V1yf3eDWtgXZ3RrWebHfRYYBIgsSl2HU+FnQfOG9nJ1a+2bDNl-
vAgvpvfvE9xbNTAqW304iBcpWVESsRO9LJ9W73TlTynW0qGexC6lz5+KwP/+4lkVh5OGe-
OGTKaxJuv0SH6rrdCCS14FdyxOGQtHkiK6yGMpvTZId5uy1iOE-
qXQ4tgaLjY5BWip08HZrQC/MgmeIzhO0kRCSBkyS6/ZZZgLqXj95TC05wUSDw0OeWjwy-
vxQ1pzmaVGJN4pxgLl4yit4vh/s3B9I0QGPqXFd7rSrc84pUlrS/8wTHLQqYDThM0RkytheX-
USx7pCpv8f3mu1djldLmhYlRQZKcYIq5rZKNz1kiqsnE1Gx/wCO9S2AihAFuaa5X0ca-
FAvZ/ZGXrhhl3quLG8zXEZC5HdKW2ebOjcP1xUa2vJ2cAkzUN+lpc4+HOWa9693dGHlz-
nLu3lA2h/hu+GoJ7mq2crNtffIXIJWH2zgaSA0sqytfCJDsfEMdIz5yjWil
      client.keystore.jks: BROK_CLIENT_KEYSTORE_JKS: MIIU2QIBAzCCFJIGCSqG-
SIb3DQEHAaCCFIMEghR/MIIUezCCAwcGCSqGSIb3DQEHAaCCAvgEggL0MIIC8DCCAuwGCyqG-
SIb3DQEMCgECoIICmzCCApcwKQYKKoZIhvcNAQwBAzAbBBQDY8Bb/tnhtJhiMG9o6FujGv2YI-
QIDAMNQBIICaNzyhNw9GZ76YjmBlQAi4Wjp7QtVUgKmK3UBxVyt-
Tjzfoy+cA6/WlXT9WXQUYfeV6CtF5Hms6BjNZPEERP3F4VH0c9K+8n/mcxt08jjma-
iip0HkgmEuq7fgDDPSBVu6gX9WD8DaVG63Is6ehX0JoOOZGv+d5BlzBk-
suBSX3UDu3RwXGe5/MvXQhT4/JZ4QZeURJ/RixND5PYn3c1ofeA4IkR1J03SH8HHlgO9Wc27pr
Sx0eLxOc9on2GoR7RirGeUeQiHJqsqZdKF4qGoL8H1baT8eDZaYt5ejj3dNlRdi2OjBzVJkhoV
v2it-
PNy1DlXB4xyL19qEBXmKDN7yjFpMvCvTer/o/DwrBQkO/kpzcXXsMWCmkV1oRzjHjql19p0oQ1
rg4UW9fYFS1tQrAjTz6z65O3/yZi7yk2ZUtsWq587VJHnMSFNtmcH9fHrNg56B61fieHJkf0U3
tuA/URMawAl-
zaVkyeh5uU9+NvCPoJ1opx1dtNcZLHxYCai+DDnjNgk3tyxVoTukL3HAVfX0WdsrezunSpnG-
SIBTwHwnwmucmBLo7oxWri9lnZ5SFbOSuEW6Itde9T0N7rFV61tBkoj5XvgLC5KDgztoGERmvE
Atjqa32eMWxrIYUpZDGiYNmv6564z2E/MxuwbAglRKytJ/6W9+sImZulu1CllSgK-
PEHsWgb0eTM+

      - apiVersion: v1
        kind: Secret
        metadata:
```

```
                    name: brok-server
                    namespace: develop
                  data:
              server.truststore.jks: MIIGLgIBAzCCBecGCSqGSIb3DQEHAaC-
CBdgEggXUMIIF0DCCBcwGCSqGSIb3DQEHBqCCBb0wggW5AgEAMIIFsgYJKoZIhvcNAQcBMCkGCiqG-
SIb3DQEMAQYwGwQUOyAHoMkvbj5oYdPCDq+xvV5Di9wCAwDDUICCBXgCh509MdZNnEDPKVRzz++LSOWXi-
blXuRuFcEnxxn6/eZWr9xnBY1kvPaiyl0zBQi6Kt33+rgb7XwBFEhhH1QLYnlxtD4P7SPKQCLQda-
TUh/Vm+bVseR3v1WaO8EjtI95nxSKG6MGvq+j4+L76au/h17UJLR8OZRQW-
povqcfJd+F1oofQ6aZDI2c9DYwYeAQVsj+iM/PCLxXqpqnedvsaEi+SexeEWr+dd0c3lbDw-
CaDo/P8ZTGWAHc20pMIStMAEirPOu8ZvV9KvpJytrWoIVH3/ZoygCwic0zDr-
Nelw4cBcydpJPGWD3jOJVo1V1yf3eDWtgXZ3RrWebHfRYYBIgsSl2HU+FnQfOG9nJ1a+aweawfaswdwdEF-
SEFSEFGBSVEFGBRSDefsdsefEFq234r4rfAESFservGRs/8wTHLQqYDThM0RkytheX-
USx7pCpv8f3mu1djldLmhYlRQZKcYIq5rZKNz1kiqsnE1Gx/wCO9S2AihAFuaa5X0ca-
FAvZ/ZGXrhhl3quLG8zXEZC5HdKW2ebOjcP1xUa2vJ2cAkzUN+lpc4+HOWa9693dGHlz-
nLu3lA2h/hu+GoJ7mq2crNtffIXIJWH2zgaSA0sqytfCJDsfEMdIz5yjWil
                    server.keystore.jks: BROK_CLIENT_KEYSTORE_JKS: MIIU2QIBA-
      zCCFJIGCSqGSIb3DQEHAaCCFIMEghR/MIIUezCCAwcGCSqGSIb3DQEHAaCCA-
      vgEggL0MIIC8DCCAuwGCyqGSIb3DQEMCgECoIICmzCCApcwKQYKKoZIhvc-
      NAQwBAzAbBBQDY8Bb/tnhtJhiMG9o6FujGv2YIQIDAMNQBIICaN-
      zyhNw9GZ76YjmBlQAi4Wjp7QtVUgKmK3UBxVyt-
      Tjzfoy+cA6/WlXT9WXQUYfeV6CtF5Hms6BjNZPEERP3F4VH0c9K+8n/wadopijoiN-
      WODJNPE:KSFJb:UEBNFpUBEFSKJBEVlSKJGVN:SEGKVnb”JLKMAWSCnes;ubfSLJEBFLSJEF-
      BHLSJKBDNFCNBSEF vlJB
      XQhT4/JZ4QZeURJ/RixND5PYn3c1ofeA4IkR1J03SH8HHlgO9Wc27prSx0eLxOc9on2GoR7RirGeU
      eQiHJqsqZdKF4qGoL8H1baT8eDZaYt5ejj3dNlRdi2OjBzVJkhoVv2it-
      PNy1DlXB4xyL19qEBXmKDN7yjFpMvCvTer/o/DwrBQkO/kpzcXXsMWCmkV1oRzjHjql19p0oQ1rg4
      UW9fYFS1tQrAjTz6z65O3/yZi7yk2ZUtsWq587VJHnMSFNtmcH9fHrNg56B61fieHJkf0U3tuA/UR
      MawAlzaVkyeh5uU9+NvCPoJ1opx1dtNcZLHxYCai+DDnjNgk3tyxVoTukL3HAVfX0Wdsrezun-
      SpnG-
      SIBTwHwnwmucmBLo7oxWri9lnZ5SFbOSuEW6Itde9T0N7rFV61tBkoj5XvgLC5KDgztoGERmvEAt-
      jqa32eMWxrIYUpZDGiYNmv6564z2E/MxuwbAglRKytJ/6W9+sImZulu1CllSgKPEHsWgb0eTM+
```

2.1.4. Описание template Ingress.yml для каждого компонента приложения

1) В качестве описания сущности Ingress для компонентов emulator-service/ bpm-orchestration-service/api-visualization-commons/pack-integration-svc-template/kafka-template-service используются следующие template.

Без TLS:

```
      apiVersion: networking.k8s.io/v1
      kind: Ingress
      metadata:
```

```
  labels:
    app: < DC_NAME >
  name: < DC_NAME >
  namespace: < PROJECT >
spec:
  rules:
    - host: < DC_NAME >.< PROJECT >.< ROUTER_PREFIX >
      http:
        paths:
        - path: /
          pathType: Prefix
          backend:
            service:
              name: < DC_NAME >
              port:
                number: 8080
```

С активным TLS:

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  labels:
    app: < DC_NAME >
  name: < DC_NAME >
  namespace: < PROJECT >
spec:
  rules:
    - host: < DC_NAME >.< PROJECT >.<ROUTER_PREFIX >
      http:
        paths:
        - path: /
          pathType: Prefix
          backend:
            service:
              name: < DC_NAME >
              port:
                number: 8080
tls:
    - hosts:
        - < DC_NAME >.< PROJECT >.< ROUTER_PREFIX >
      secretName: < TLS_SECRET_NAME >
```

2) В качестве описания сущности Ingress для компонента ***brok*** используются следующие template:

Без TLS:

```
kind: Ingress
apiVersion: networking.k8s.io/v1
metadata:
  name: < DC_NAME >
  namespace: < PROJECT >
  labels:
    name: < DC_NAME >
  annotations:
    kubernetes.io/ingress.class: nginx
    nginx.ingress.kubernetes.io/backend-protocol: HTTP
    nginx.ingress.kubernetes.io/ssl-passthrough: 'true'
spec:
  rules:
    - host: < DC_NAME >.< PROJECT >.< ROUTER_PREFIX >
      http:
        paths:
          - path: /core/
            pathType: Prefix
            backend:
              service:
                name: < DC_NAME >
                port:
                  name: < APP_PORT >-tcp
```

С активным TLS:

```
kind: Ingress
apiVersion: networking.k8s.io/v1
metadata:
  name: < DC_NAME >
  namespace: < PROJECT >
  labels:
    name: < DC_NAME >
  annotations:
    kubernetes.io/ingress.class: nginx
    nginx.ingress.kubernetes.io/backend-protocol: HTTP
    nginx.ingress.kubernetes.io/ssl-passthrough: 'true'
spec:
```

```
  rules:
    - host: < DC_NAME >.< PROJECT >.< ROUTER_PREFIX >
      http:
        paths:
          - path: /core/
            pathType: Prefix
            backend:
              service:
                name: < DC_NAME >
                port:
                  name: < APP_PORT >-tcp
  tls:
    - hosts:
        - < DC_NAME >.< PROJECT >.< ROUTER_PREFIX >
      secretName: < TLS_SECRET_NAME >
```

3) В качестве описания сущности Ingress для компонента ***brok-ui*** и ***brok-auth*** используется следующие template:

Без TLS:

```
kind: Ingress
apiVersion: networking.k8s.io/v1
metadata:
  name: < DC_NAME >
  namespace: < PROJECT >
  labels:
    name: < DC_NAME >
  annotations:
    kubernetes.io/ingress.class: nginx
    nginx.ingress.kubernetes.io/backend-protocol: HTTP
    nginx.ingress.kubernetes.io/ssl-passthrough: 'true'
    nginx.ingress.kubernetes.io/rewrite-target: /
spec:
  rules:
    - host: < DC_NAME >.< PROJECT >.< ROUTER_PREFIX >
      http:
        paths:
          - path: /
            pathType: Prefix
            backend:
              service:
```

```
                    name: < DC_NAME >
                    port:
                      name: < APP_PORT >-tcp
```

С активным TLS:

```
kind: Ingress
apiVersion: networking.k8s.io/v1
metadata:
  name: < DC_NAME >
  namespace: < PROJECT >
  labels:
    name: < DC_NAME >
  annotations:
    kubernetes.io/ingress.class: nginx
    nginx.ingress.kubernetes.io/backend-protocol: HTTP
    nginx.ingress.kubernetes.io/ssl-passthrough: 'true'
    nginx.ingress.kubernetes.io/rewrite-target: /
spec:
  rules:
    - host: < DC_NAME >.< PROJECT >.< ROUTER_PREFIX >
      http:
        paths:
          - path: /
            pathType: Prefix
            backend:
              service:
                name: < DC_NAME >
                port:
                  name: < APP_PORT >-tcp
  tls:
    - hosts:
        - < DC_NAME >< PROJECT >.< ROUTER_PREFIX >
      secretName: < TLS_SECRET_NAME >
```

4) Параметры, которые необходимо задать в template перед deploy Ingress компонентов **brok**, ***brok-ui*** и ***brok-auth:***

<DC_NAME> — имя компонента приложения. Рекомендуемые имена для компонентов приложения: brok/brok-bpm/brok-auth/brok-ui, однако может быть задано любое имя компонента приложения в соответствии с требованиями нейминга приложений внутри учреждения.

<PROJECT> — пространство имен оркестратора, в которое непосредственно будет осуществляться deploy приложения.

<ROUTER_PREFIX> — параметр, определяющий доменное имя оркестратора, в который осуществляется Deploy приложения.

<APP_PORT> — порт, по которому осуществляется доступ внутри оркестратора и по которому приложение отвечает по поступающие к нему запросы.

< TLS_SECRET_NAME > — имя секрета (сущности оркестратора k8s), в котором хранится пользовательский сертификат и ключ к нему.

5) Примеры template для каждого компонента приложения.

a) Ingress emulator-service

С включённой защитой соединения по TLS:

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  labels:
    app: emulator-service
  name: emulator-service
  namespace: develop
spec:
  rules:
    - host: emulator-service.develop.apps.dev.k8s-javax
      http:
        paths:
          - backend:
              service:
                name: emulator-service
                port:
                  number: 8080
            path: /
            pathType: Prefix
  tls:
    - hosts:
        - emulator-service.develop.apps.dev.k8s-javax
      secretName: tls-secret
```

Без защиты соединения по TLS:

```
apiVersion: networking.k8s.io/v1
kind: Ingress
```

```
metadata:
  labels:
    app: emulator-service
  name: emulator-service
  namespace: develop
spec:
  rules:
    - host: emulator-service.develop.apps.dev.k8s-javax
      http:
        paths:
          - backend:
              service:
                name: emulator-service
                port:
                  number: 8080
            path: /
            pathType: Prefix
```

б) Ingress bpm-orchestration-service

С включённой защитой соединения по TLS:

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  labels:
    app: bpm-orchestration-service
  name: bpm-orchestration-service
  namespace: develop
spec:
  rules:
    - host: bpm-orchestration-service.develop.apps.dev.k8s-javax
      http:
        paths:
          - backend:
              service:
                name: bpm-orchestration-service
                port:
                  number: 8080
            path: /
            pathType: Prefix
  tls:
```

```
    - hosts:
        - bpm-orchestration-service.develop.apps.dev.k8s-javax
      secretName: tls-secret
```

Без защиты соединения по TLS:

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  labels:
    app: bpm-orchestration-service
  name: bpm-orchestration-service
  namespace: develop
spec:
  rules:
    - host: bpm-orchestration-service.develop.apps.dev.k8s-javax
      http:
        paths:
          - backend:
              service:
                name: bpm-orchestration-service
                port:
                  number: 8080
            path: /
            pathType: Prefix
```

в) Ingress api-visualization-commons

С включённой защитой соединения по TLS:

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  labels:
    app: api-visualization-commons
  name: api-visualization-commons
  namespace: develop
spec:
  rules:
    - host: api-visualization-commons.develop.apps.dev.k8s-javax
      http:
        paths:
          - backend:
```

```
                service:
                  name: api-visualization-commons
                  port:
                    number: 8080
              path: /
              pathType: Prefix
  tls:
    - hosts:
        - api-visualization-commons.develop.apps.dev.k8s-javax
      secretName: tls-secret
```

Без защиты соединения по TLS:

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  labels:
    app: api-visualization-commons
  name: api-visualization-commons
  namespace: develop
spec:
  rules:
    - host: api-visualization-commons.develop.apps.dev.k8s-javax
      http:
        paths:
          - backend:
              service:
                name: api-visualization-commons
                port:
                  number: 8080
            path: /
            pathType: Prefix
```

г) Ingress pack-integration-svc-template

С включённой защитой соединения по TLS:

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  labels:
    app: pack-integration-svc-template
  name: pack-integration-svc-template
  namespace: develop
```

```
spec:
  rules:
    - host: pack-integration-svc-template.develop.apps.dev.k8s-javax
      http:
        paths:
          - backend:
              service:
                name: pack-integration-svc-template
                port:
                  number: 8080
            path: /
            pathType: Prefix
  tls:
    - hosts:
        - pack-integration-svc-template.develop.apps.dev.k8s-javax
      secretName: tls-secret
```

Без защиты соединения по TLS:

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  labels:
    app: pack-integration-svc-template
  name: pack-integration-svc-template
  namespace: develop
spec:
  rules:
    - host: pack-integration-svc-template.develop.apps.dev.k8s-javax
      http:
        paths:
          - backend:
              service:
                name: pack-integration-svc-template
                port:
                  number: 8080
            path: /
            pathType: Prefix
```

д) Ingress kafka-template-service

С включённой защитой соединения по TLS:

```
apiVersion: networking.k8s.io/v1
```

```
kind: Ingress
metadata:
  labels:
    app: kafka-template-service
  name: kafka-template-service
  namespace: develop
spec:
  rules:
    - host: kafka-template-service.develop.apps.dev.k8s-javax
      http:
        paths:
          - backend:
              service:
                name: kafka-template-service
                port:
                  number: 8080
            path: /
            pathType: Prefix
  tls:
    - hosts:
        - kafka-template-service.develop.apps.dev.k8s-javax
      secretName: tls-secret
```

Без защиты соединения по TLS:

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  labels:
    app: kafka-template-service
  name: kafka-template-service
  namespace: develop
spec:
  rules:
    - host: kafka-template-service.develop.apps.dev.k8s-javax
      http:
        paths:
          - backend:
              service:
                name: kafka-template-service
                port:
                  number: 8080
```

```
              path: /
              pathType: Prefix
```

e) Ingress ***brok***

С включённой защитой соединения по TLS:

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  annotations:
    kubernetes.io/ingress.class: nginx
    meta.helm.sh/release-name: brok
    meta.helm.sh/release-namespace: develop
    nginx.ingress.kubernetes.io/backend-protocol: HTTP
    nginx.ingress.kubernetes.io/ssl-passthrough: 'true'
  labels:
    app.kubernetes.io/managed-by: Helm
    name: brok
  name: brok
  namespace: develop
spec:
  rules:
    - host: brok.develop.apps.dev.k8s
      http:
        paths:
          - backend:
              service:
                name: brok
                port:
                  name: 8081-tcp
            path: /core/
            pathType: Prefix
  tls:
    - hosts:
        - brok.develop.apps.dev.k8s
      secretName: tls-secret
```

Без защиты соединения по TLS:

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  annotations:
```

```
    kubernetes.io/ingress.class: nginx
    meta.helm.sh/release-name: brok
    meta.helm.sh/release-namespace: develop
    nginx.ingress.kubernetes.io/backend-protocol: HTTP
    nginx.ingress.kubernetes.io/ssl-passthrough: 'true'
  labels:
    app.kubernetes.io/managed-by: Helm
    name: brok
  name: brok
  namespace: develop
spec:
  rules:
    - host: brok.develop.apps.dev.k8s
      http:
        paths:
          - backend:
              service:
                name: brok
                port:
                  name: 8081-tcp
            path: /core/
            pathType: Prefix
```

ж) Ingress brok-auth

С включённой защитой соединения по TLS:

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  annotations:
    kubernetes.io/ingress.class: nginx
    meta.helm.sh/release-name: brok
    meta.helm.sh/release-namespace: develop
    nginx.ingress.kubernetes.io/backend-protocol: HTTP
    nginx.ingress.kubernetes.io/rewrite-target: /
    nginx.ingress.kubernetes.io/ssl-passthrough: 'true'
  labels:
    app: brok-auth
    app.kubernetes.io/managed-by: Helm
    name: brok-auth
  name: brok-auth
  namespace: develop
```

```
spec:
  rules:
    - host: brok-auth.develop.apps.dev.k8s
      http:
        paths:
          - backend:
              service:
                name: brok-auth
                port:
                  name: 8080-tcp
            path: /
            pathType: Prefix
  tls:
    - hosts:
        - brok-auth.develop.apps.dev.k8s
      secretName: tls-secret
```

Без защиты соединения по TLS:

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  annotations:
    kubernetes.io/ingress.class: nginx
    meta.helm.sh/release-name: brok
    meta.helm.sh/release-namespace: develop
    nginx.ingress.kubernetes.io/backend-protocol: HTTP
    nginx.ingress.kubernetes.io/rewrite-target: /
    nginx.ingress.kubernetes.io/ssl-passthrough: 'true'
  labels:
    app: brok-auth
    app.kubernetes.io/managed-by: Helm
    name: brok-auth
  name: brok-auth
  namespace: develop
spec:
  rules:
    - host: brok-auth.develop.apps.dev.k8s
      http:
        paths:
          - backend:
              service:
```

```
                    name: brok-auth
                    port:
                      name: 8080-tcp
                path: /
                pathType: Prefix
```

з) Ingress brok-ui

С включённой защитой соединения по TLS:

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  annotations:
    kubernetes.io/ingress.class: nginx
    meta.helm.sh/release-name: brok
    meta.helm.sh/release-namespace: develop
    nginx.ingress.kubernetes.io/backend-protocol: HTTP
    nginx.ingress.kubernetes.io/ssl-passthrough: 'true'
  labels:
    app.kubernetes.io/managed-by: Helm
    name: brok-ui
  name: brok-ui
  namespace: develop
spec:
  rules:
    - host: brok.develop.apps.dev.k8s
      http:
        paths:
          - backend:
              service:
                name: brok-ui
                port:
                  name: 8080-tcp
            path: /
            pathType: Prefix
  tls:
    - hosts:
        - brok.develop.apps.dev.k8s
      secretName: tls-secret
```

Без защиты соединения по TLS:

```
apiVersion: networking.k8s.io/v1
```

```
    kind: Ingress
    metadata:
      annotations:
        kubernetes.io/ingress.class: nginx
        meta.helm.sh/release-name: brok
        meta.helm.sh/release-namespace: develop
        nginx.ingress.kubernetes.io/backend-protocol: HTTP
        nginx.ingress.kubernetes.io/ssl-passthrough: 'true'
      labels:
        app.kubernetes.io/managed-by: Helm
        name: brok-ui
      name: brok-ui
      namespace: develop
    spec:
      rules:
        - host: brok.develop.apps.dev.k8s
          http:
            paths:
              - backend:
                  service:
                    name: brok-ui
                    port:
                      name: 8080-tcp
                path: /
                pathType: Prefix
```

2.1.5. Описание template Service.yml для каждого компонента приложения

1) В качестве описания сущности Service для компонентов программного комплекса «Джавакс» используется следующий template:

```
  - apiVersion: v1
    kind: Service
    metadata:
      labels:
        app: < DC_NAME >
      name: < DC_NAME >
      namespace: < PROJECT >
    spec:
      ports:
        - name: < APP_PORT >-tcp
          port: 80
```

```
      protocol: TCP
      targetPort: < APP_PORT >
  selector:
    deployment: < DC_NAME >
  sessionAffinity: None
  type: ClusterIP
```

2) Параметры, которые необходимо задать в template перед deploy Service программного комплекса «Джавакс»:

<DC_NAME> — имя компонента приложения. Рекомендуемые имена для компонентов приложения: brok/brok-bpm/brok-auth/brok-ui, однако может быть задано любое имя компонента приложения в соответствии с требованиями нейминга приложений внутри учреждения.

<PROJECT> — пространство имен оркестратора, в которое непосредственно будет осуществляться deploy приложения.

<APP_PORT> — порт, по которому осуществляется доступ внутри оркестратора и по которому приложение отвечает по поступающие к нему запросы.

3) Примеры template для каждого компонента приложения.

а) Service emulator-service

```
apiVersion: v1
kind: Service
metadata:
  labels:
    app: emulator-service
  name: emulator-service
  namespace: develop
spec:
  ports:
    - name: 8080-tcp
      port: 80
      protocol: TCP
      targetPort: 8080
  selector:
    deployment: emulator-service
  sessionAffinity: None
  type: ClusterIP
```

б) Service bpm-orchestration-service

```
apiVersion: v1
```

```
kind: Service
metadata:
  labels:
    app: bpm-orchestration-service
  name: bpm-orchestration-service
  namespace: demo
spec:
  ports:
    - name: 8080-tcp
      port: 80
      protocol: TCP
      targetPort: 8080
  selector:
    deployment: bpm-orchestration-service
  sessionAffinity: None
  type: ClusterIP
```

в) Service api-visualization-commons

```
apiVersion: v1
kind: Service
metadata:
  labels:
    app: api-visualization-commons
  name: api-visualization-commons
  namespace: demo
spec:
  ports:
    - name: 8080-tcp
      port: 80
      protocol: TCP
      targetPort: 8080
  selector:
    deployment: api-visualization-commons
  sessionAffinity: None
  type: ClusterIP
```

г) Service pack-integration-svc-template

```
apiVersion: v1
kind: Service
metadata:
  labels:
```

```
      app: pack-integration-svc-template
    name: pack-integration-svc-template
    namespace: demo
  spec:
    ports:
      - name: 8080-tcp
        port: 80
        protocol: TCP
        targetPort: 8080
    selector:
      deployment: pack-integration-svc-template
    sessionAffinity: None
    type: ClusterIP
```

д) Service kafka-template-service

```
apiVersion: v1
kind: Service
metadata:
  labels:
    app: kafka-template-service
  name: kafka-template-service
  namespace: demo
spec:
  ports:
    - name: 8080-tcp
      port: 80
      protocol: TCP
      targetPort: 8080
  selector:
    deployment: kafka-template-service
  sessionAffinity: None
  type: ClusterIP
```

е) Service brok

```
    - apiVersion: v1
      kind: Service
      metadata:
        labels:
          app: brok
        name: brok
        namespace: develop
```

```
    spec:
      ports:
        - name: 8081-tcp
          port: 80
          protocol: TCP
          targetPort: 8081
      selector:
        deployment: brok
      sessionAffinity: None
      type: ClusterIP
```

ж) Service brok-bpm

```
- apiVersion: v1
  kind: Service
  metadata:
    labels:
      app: brok-bpm
    name: brok-bpm
    namespace: develop
  spec:
    ports:
      - name: 8082-tcp
        port: 80
        protocol: TCP
        targetPort: 8082
    selector:
      deployment: brok-bpm
    sessionAffinity: None
    type: ClusterIP
```

з) Service brok-auth

```
- apiVersion: v1
  kind: Service
  metadata:
    labels:
      app: brok-auth
    name: brok-auth
    namespace: develop
  spec:
    ports:
      - name: 8080-tcp
```

```
      port: 80
      protocol: TCP
      targetPort: 8080
    selector:
      deployment: brok-auth
    sessionAffinity: None
    type: ClusterIP
```

и) Service brok-ui

```
- apiVersion: v1
  kind: Service
  metadata:
    labels:
      app: brok-ui
    name: brok-ui
    namespace: develop
  spec:
    ports:
      - name: 8080-tcp
        port: 80
        protocol: TCP
        targetPort: 8080
    selector:
      deployment: brok-ui
    sessionAffinity: None
    type: ClusterIP
```

2.2. Описание возможных вариантов Deploy в оркестратор Kubernetes

2.2.1. Ручной (не автоматизированный) вариант deploy компанентов

Deploy всех компонентов приложения, в ручном (не автоматизированном) режиме осуществляется с использованием специализированной консольной утилиты — kubectl.

Подробное описание о том, как установить данную консольную утилиту на Ваш компьютер, можно найти в официальной документации Kubernates по адресу: ***https://kubernetes.io/ru/docs/tasks/tools/install-kubectl/***

После установки утилиты необходимо выполнить последовательный вызов команды с указанием каждого заполненными template компонентов приложения:

```
kubectl apply -f <TEMPLATE_NAME>.yml
```

2.2.2. Автоматизированный вариант deploy BrOk

Как говорилось ранее процесс написания pipeline в рамках настоящей инструкции рассматриваться не будет по причине большого разнообразия продуктов, используемых в качестве систем доставки кода в оркестратор.

Однако в данном пункте будут рассмотрены основные сущности, участвующие в процессе автоматизации Deploy приложения BrOk в оркестратор.

Для автоматизации процесса Deploy рекомендуется использование средства установки с открытым исходным кодом, которое помогает установить приложения Kubernates и управлять их жизненным циклом — ***Helm.***

Автоматизированный деплой иных компонентов программного комплекса «Джавакс» в оркестратор с помощью ***Helm*** в рамках настоящей инструкции не рассматривается по причине отсутствия необходимых манифестов для иных компонентов программного комплекса «Джавакс».

1) Ниже приведены пример всех template и values, которые могут быть использованы в процессе Deploy всех компонентов приложения в оркестратор Kubernates с помощью Helm.

a) Helm template для компонента ***brok-core***

ConfigMap:

```
apiVersion: v1
kind: ConfigMap
metadata:
  labels:
    app: {{ .Values.CommonValues.MainProjectName }}
  name: {{ .Values.CommonValues.MainProjectName }}
  namespace: {{ .Values.CommonValues.Namespace }}
data:
```

```
  DATASOURCE_URL: {{ .Values.ConfigMap.common.DatasourceUrl }}
  DATASOURCE_USERNAME: {{ .Values.ConfigMap.common.DatasourceUsername }}
  APPLICATION_PATHS_UI: "{{ .Values.ConfigMap.common.application.paths.ui }}"
  APPLICATION_PATHS_CORE: "{{ .Values.ConfigMap.common.application.paths.core
}}"
  APPLICATION_PATHS_BPM: "{{ .Values.ConfigMap.common.application.paths.bpm }}"
  APPLICATION_AUTHORIZATION_DATABASE_SCHEME: "{{ .Values.CommonValues.KcDbSchema
}}"
  AUDIT_ENABLED: "{{ .Values.ConfigMap.common.audit.Enabled }}"
  CLEANER_CRONEXPRESSION: "{{ .Values.ConfigMap.common.cleaner.cronExpression
}}"
  CLEANER_AUDIT_ENABLED: "{{ .Values.ConfigMap.common.cleaner.audit.Enabled }}"
  CLEANER_AUDIT_COUNT: "{{ .Values.ConfigMap.common.cleaner.audit.count }}"
  CACHE_ENABLED: "{{ .Values.ConfigMap.common.cache.Enabled }}"
  CACHE_PROPERTIES_CLUSTERS_ENABLED: "{{ .Values.ConfigMap.common.cache.proper-
ties.clusters.Enabled }}"
  CACHE_PROPERTIES_TOPICS_ENABLED: "{{ .Values.ConfigMap.common.cache.proper-
ties.topics.Enabled }}"
  CACHE_PROPERTIES_FAVORITES_ENABLED: "{{ .Values.ConfigMap.common.cache.proper-
ties.favorites.Enabled }}"
  CACHE_PROPERTIES_TEMPLATES_ENABLED: "{{ .Values.ConfigMap.common.cache.proper-
ties.templates.Enabled }}"
  LOGGING_LEVEL_COM_AXENIX_BROK: {{ .Values.ConfigMap.common.LdapAutorisa-
tion.KafkaUiLogLevel }}
  {{ if .Values.ConfigMap.common.LdapAutorisation.enabled }}
  LOGGING_LEVEL_ROOT: {{ .Values.ConfigMap.common.LdapAutorisation.Kaf-
kaUiLogLevel }}
  AUTH_TYPE: {{ .Values.ConfigMap.common.LdapAutorisation.AuthType }}
  SPRING_LDAP_URLS: {{ .Values.ConfigMap.common.LdapAutorisation.SpringLdapUrls
}}
  SPRING_LDAP_DN_PATTERN: {{ .Values.ConfigMap.common.LdapAutorisa-
tion.SpringLdapDnPattern }}
  SPRING_LDAP_USERFILTER_SEARCHBASE: {{ .Values.ConfigMap.common.LdapAutorisa-
tion.SpringLdapUserfilterSearchbase }}
  SPRING_LDAP_USERFILTER_SEARCHFILTER: {{ .Values.ConfigMap.common.LdapAutorisa-
tion.SpringLdapUserFilterSearchfilter }}
  {{ else if .Values.ConfigMap.common.KeycloakAutorisation.enabled }}
  AUTH_TYPE: {{ .Values.ConfigMap.common.KeycloakAutorisation.AuthType }}
  AUTH_OAUTH2_RESOURCESERVER_JWT_ISSUERURI: {{ .Values.ConfigMap.common.Key-
cloakAutorisation.AuthOauth2ResourcesServerJwtIssUerUri }}
  AUTH_OAUTH2_RESOURCESERVER_JWT_JWKSETURI: {{ .Values.ConfigMap.common.Key-
cloakAutorisation.AuthOauth2ResourcesServerJwtJwkSetUei }}
```

```
    {{ end }}
```

Deployment:

```
      apiVersion: apps/v1
      kind: Deployment
      metadata:
        namespace: {{ .Values.CommonValues.Namespace }}
        name: {{ .Values.CommonValues.MainProjectName }}
        labels:
          commit: {{ .Values.CommonValues.commitTag}}
          app: {{ .Values.CommonValues.MainProjectName }}
          name: {{ .Values.CommonValues.MainProjectName }}
      spec:
        replicas: {{ .Values.Deployment.ReplicasCount }}
        selector:
          matchLabels:
            app: {{ .Values.CommonValues.MainProjectName }}
        template:
          metadata:
            creationTimestamp: null
            labels:
              app: {{ .Values.CommonValues.MainProjectName }}
              deployment: {{.Values.CommonValues.MainProjectName }}
          spec:
            imagePullSecrets:
              - name: {{ .Values.Deployment.ImagePullSecretsName }}
            securityContext: {}
      {{ if .Values.Deployment.enablednodeSelector }}
            nodeSelector:
              node-role.kubernetes.io/worker: {{ .Values.Deployment.nodeSelector
}}
      {{ end }}
            serviceAccountName: {{ .Values.Deployment.SA }}
            serviceAccount: {{ .Values.Deployment.SA }}
            containers:
              - name: {{ .Values.CommonValues.MainProjectName }}
                image: {{ .Values.Deployment.ImageNameBase }}
                imagePullPolicy: Always
                env:
                  - name: POD_NAME
                    valueFrom:
```

```
                fieldRef:
                  fieldPath: metadata.name
            - name: NAMESPACE
              valueFrom:
                fieldRef:
                  fieldPath: metadata.namespace
            - name: LOG_LEVEL
              value: {{ .Values.Deployment.LogLevel }}
            - name: TZ
              value: "{{ .Values.Deployment.TZ }}"
          envFrom:
            - configMapRef:
                name: {{ .Values.CommonValues.MainProjectName }}
            - secretRef:
                name: {{ .Values.CommonValues.MainProjectName }}
          ports:
            - containerPort: {{ .Values.CommonValues.AppPortApi }}
              protocol: TCP
          resources:
            limits:
              cpu: {{ .Values.Deployment.CpuLimit }}
              memory: {{ .Values.Deployment.MemoryLimit }}
            requests:
              cpu: {{ .Values.Deployment.CpuRequest  }}
              memory: {{ .Values.Deployment.MemoryRequest }}
          livenessProbe:
            failureThreshold: 3
            httpGet:
              path: /actuator/health/liveness
              port: {{ .Values.CommonValues.AppPortApi }}
              scheme: HTTP
            initialDelaySeconds: 600
            periodSeconds: 30
            successThreshold: 1
            timeoutSeconds: 10
          readinessProbe:
            httpGet:
              path: /actuator/health/readiness
              port: {{ .Values.CommonValues.AppPortApi }}
              scheme: HTTP
            failureThreshold: {{ .Values.Deployment.ReadinessProbeFailure-
Threshhold }}
```

```
                    initialDelaySeconds: {{ .Values.Deployment.ReadinessProbeI-
nitialDelaySeconds }}
                    periodSeconds: {{ .Values.Deployment.ReadinessProbePeriodSecond
}}
```

Ingress:

```
        kind: Ingress
        apiVersion: networking.k8s.io/v1
        metadata:
          name: {{ .Values.CommonValues.MainProjectName }}
          namespace: {{ .Values.CommonValues.Namespace }}
          labels:
            name: {{ .Values.CommonValues.MainProjectName }}
          annotations:
            kubernetes.io/ingress.class: nginx
            nginx.ingress.kubernetes.io/backend-protocol: HTTP
            nginx.ingress.kubernetes.io/ssl-passthrough: 'true'
        spec:
          rules:
            - host: {{ .Values.CommonValues.MainProjectName }}.{{ .Values.CommonVal-
ues.Namespace }}.{{ .Values.Ingress.IngressPrefix }}
              http:
                paths:
                  - path: /core/
                    pathType: Prefix
                    backend:
                      service:
                        name: {{ .Values.CommonValues.MainProjectName }}
                        port:
                          name: {{ .Values.CommonValues.AppPortApi }}-tcp
```

Secret:

```
        apiVersion: v1
        kind: Secret
        metadata:
          labels:
            app: {{ .Values.CommonValues.MainProjectName }}
          name: {{ .Values.CommonValues.MainProjectName }}
          namespace: {{ .Values.CommonValues.Namespace }}
        stringData:
          LICENSE_KEY: "{{ .Values.ConfigMap.common.license.key }}"
```

```
          LICENSE_TYPE: "{{ .Values.ConfigMap.common.license.type }}"
          LICENSE_KUBERNETES_HOST: "{{ .Values.ConfigMap.common.license.kuber-
netes.host }}"
          LICENSE_KUBERNETES_NAMESPACE: "{{ .Values.ConfigMap.common.license.kuber-
netes.namespace }}"
          LICENSE_KUBERNETES_API_VERSION: "{{ .Values.ConfigMap.common.license.ku-
bernetes.apiVersion }}"
          LICENSE_KUBERNETES_TOKEN: "{{ .Values.ConfigMap.common.license.kuber-
netes.token }}"
          DATASOURCE_PASSWORD:  {{ .Values.ConfigMap.common.DatasourcePassword }}
          DATASOURCE_USERNAME: {{ .Values.ConfigMap.common.DatasourceUsername }}
          BOT_NAME: "{{ .Values.ConfigMap.common.botName }}"
          BOT_TOKEN: "{{ .Values.ConfigMap.common.botToken }}"
          BOT_CHATID: "{{ .Values.ConfigMap.common.botChatID }}"
          BOT_ENABLED: "{{ .Values.ConfigMap.common.botEnabled }}"
          {{ if .Values.ConfigMap.common.LdapAutorisation.enabled }}
          SPRING_LDAP_ADMINUSER: {{ .Values.ConfigMap.common.LdapAutorisa-
tion.SpringLdapAdminuser }}
          SPRING_LDAP_ADMINPASSWORD: {{ .Values.ConfigMap.common.LdapAutorisa-
tion.SpringLdapAdminPassword }}
          {{ end }}
```

Service:

```
        apiVersion: v1
        kind: Service
        metadata:
          labels:
            app: {{ .Values.CommonValues.MainProjectName }}
          name: {{ .Values.CommonValues.MainProjectName }}
          namespace: {{ .Values.CommonValues.Namespace }}
        spec:
          ports:
            - name: {{ .Values.CommonValues.AppPortApi }}-tcp
              port: 80
              protocol: TCP
              targetPort: {{ .Values.CommonValues.AppPortApi }}
          selector:
            deployment: {{ .Values.CommonValues.MainProjectName }}
          sessionAffinity: None
          type: ClusterIP
```

б) Helm template для компонента ***brok-bpm***

ConfigMap

```
apiVersion: v1
kind: ConfigMap
metadata:
  labels:
    app: {{ .Values.CommonValues.MainProjectName }}-bpm
  name: {{ .Values.CommonValues.MainProjectName }}-bpm
  namespace: {{ .Values.CommonValues.Namespace }}
data:
  SECRET_NAME: "{{ .Values.CommonValues.MainProjectName }}-bpm"
  DATASOURCE_URL: "{{ .Values.ConfigMap.common.DatasourceUrl }}"
  DATASOURCE_USERNAME: "{{ .Values.ConfigMap.common.DatasourceUsername }}"
  DATASOURCE_SCHEMA: "{{ .Values.ConfigMap.bpm.datasource.schema }}"
  APPLICATION_PATHS_UI: "{{ .Values.ConfigMap.common.application.paths.ui
}}"
  APPLICATION_PATHS_CORE: "{{ .Values.ConfigMap.common.applica-
tion.paths.core }}"
```

Deployment:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  namespace: {{ .Values.CommonValues.Namespace }}
  name: {{ .Values.CommonValues.MainProjectName }}-bpm
  labels:
    commit: {{ .Values.CommonValues.commitTag}}
    app: {{ .Values.CommonValues.MainProjectName }}-bpm
    name: {{ .Values.CommonValues.MainProjectName }}-bpm
spec:
  replicas: {{ .Values.Deployment.ReplicasCount }}
  selector:
    matchLabels:
      app: {{ .Values.CommonValues.MainProjectName }}-bpm
  template:
    metadata:
      creationTimestamp: null
      labels:
        app: {{ .Values.CommonValues.MainProjectName }}-bpm
        deployment: {{.Values.CommonValues.MainProjectName }}-bpm
```

```
            spec:
              imagePullSecrets:
                - name: {{ .Values.Deployment.ImagePullSecretsName }}
              securityContext: {}
        {{ if .Values.Deployment.enablednodeSelector }}
              nodeSelector:
                node-role.kubernetes.io/worker: {{ .Values.Deployment.nodeSelector
}}
        {{ end }}
              serviceAccountName: {{ .Values.Deployment.SA }}
              serviceAccount: {{ .Values.Deployment.SA }}
              containers:
                - name: {{ .Values.CommonValues.MainProjectName }}-bpm
                  image: {{ .Values.Deployment.ImageNameBpm }}
                  imagePullPolicy: Always
                  env:
                    - name: POD_NAME
                      valueFrom:
                        fieldRef:
                          fieldPath: metadata.name
                    - name: NAMESPACE
                      valueFrom:
                        fieldRef:
                          fieldPath: metadata.namespace
                    - name: LOG_LEVEL
                      value: {{ .Values.Deployment.LogLevel }}
                    - name: TZ
                      value: "{{ .Values.Deployment.TZ }}"
                  envFrom:
                    - configMapRef:
                        name: {{ .Values.CommonValues.MainProjectName }}-bpm
                    - secretRef:
                        name: {{ .Values.CommonValues.MainProjectName }}-bpm
                  ports:
                    - containerPort: {{ .Values.CommonValues.AppPortBpm }}
                      protocol: TCP
                  resources:
                    limits:
                      cpu: {{ .Values.Deployment.CpuLimit }}
                      memory: {{ .Values.Deployment.MemoryLimit }}
                    requests:
                      cpu: {{ .Values.Deployment.CpuRequest  }}
```

```
                    memory: {{ .Values.Deployment.MemoryRequest }}
                livenessProbe:
                  failureThreshold: 3
                  httpGet:
                    path: /actuator/health/liveness
                    port: {{ .Values.CommonValues.AppPortBpm }}
                    scheme: HTTP
                  initialDelaySeconds: 600
                  periodSeconds: 30
                  successThreshold: 1
                  timeoutSeconds: 10
                readinessProbe:
                  httpGet:
                    path: /actuator/health/readiness
                    port: {{ .Values.CommonValues.AppPortBpm }}
                    scheme: HTTP
                  failureThreshold: {{ .Values.Deployment.ReadinessProbeFailure-
Threshhold }}
                  initialDelaySeconds: {{ .Values.Deployment.ReadinessProbeI-
nitialDelaySeconds }}
                  periodSeconds: {{ .Values.Deployment.ReadinessProbePeriodSecond
}}
```

Service:

```
      apiVersion: v1
      kind: Service
      metadata:
        labels:
          app: {{ .Values.CommonValues.MainProjectName }}-bpm
        name: {{ .Values.CommonValues.MainProjectName }}-bpm
        namespace: {{ .Values.CommonValues.Namespace }}
      spec:
        ports:
          - name: {{ .Values.CommonValues.AppPortBpm }}-tcp
            port: 80
            protocol: TCP
            targetPort: {{ .Values.CommonValues.AppPortBpm }}
        selector:
          deployment: {{ .Values.CommonValues.MainProjectName }}-bpm
        sessionAffinity: None
        type: ClusterIP
```

Secret:

```
apiVersion: v1
kind: Secret
metadata:
  labels:
    app: {{ .Values.CommonValues.MainProjectName }}-bpm
  name: {{ .Values.CommonValues.MainProjectName }}-bpm
  namespace: {{ .Values.CommonValues.Namespace }}
stringData:
  DATASOURCE_USERNAME: "{{ .Values.ConfigMap.bpm.datasource.username }}"
  DATASOURCE_PASSWORD: "{{ .Values.ConfigMap.bpm.datasource.password }}"
  CAMUNDA_BPM_ADMIN_USER_ID: "{{ .Values.ConfigMap.bpm.camunda.adminUser.id
}}"
  CAMUNDA_BPM_ADMIN_USER_PASSWORD: "{{ .Values.ConfigMap.bpm.camunda.ad-
minUser.password }}"
```

в) Helm template для компонента **brok-auth**

ConfigMap

```
apiVersion: v1
kind: ConfigMap
metadata:
  labels:
    app: {{ .Values.CommonValues.MainProjectName }}-auth
  name: {{ .Values.CommonValues.MainProjectName }}-auth
  namespace: {{ .Values.CommonValues.Namespace }}
data:
  SECRET_NAME: "{{ .Values.CommonValues.MainProjectName }}-auth"
  KEYCLOAK_DATABASE_HOST: "{{ .Values.CommonValues.KcDbHost }}"
  KEYCLOAK_DATABASE_PORT: "{{ .Values.CommonValues.KcDbPort }}"
  KEYCLOAK_DATABASE_VENDOR: "{{ .Values.CommonValues.DbVendor }}"
  KEYCLOAK_DATABASE_NAME: "{{ .Values.CommonValues.DbDatabase }}"
  KEYCLOAK_DATABASE_SCHEMA: "{{ .Values.CommonValues.KcDbSchema }}"
  KEYCLOAK_ADMIN: "{{ .Values.CommonValues.KcAdminUser }}"
  PROXY_ADDRESS_FORWARDING: "{{ .Values.ConfigMap.auth.ProxyAddressForward-
ing}}"
  START_PAGE_HOST: "{{ .Values.ConfigMap.auth.StartPageHost }}"
  KEYCLOAK_LOGLEVEL: "{{ .Values.ConfigMap.auth.KeycloakLoglvl }}"
  KEYCLOAK_ENABLE_HEALTH_ENDPOINTS: "{{ .Values.CommonValues.KcEnable-
HealthEndpoints }}"
```

Deployment:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  namespace: {{ .Values.CommonValues.Namespace }}
  name: {{ .Values.CommonValues.MainProjectName }}-auth
  labels:
    commit: {{ .Values.CommonValues.commitTag}}
    app: {{ .Values.CommonValues.MainProjectName }}-auth
    name: {{ .Values.CommonValues.MainProjectName }}-auth
spec:
  replicas: {{ .Values.Deployment.ReplicasCount }}
  selector:
    matchLabels:
      app: {{ .Values.CommonValues.MainProjectName }}-auth
  template:
    metadata:
      creationTimestamp: null
      labels:
        app: {{ .Values.CommonValues.MainProjectName }}-auth
        deployment: {{.Values.CommonValues.MainProjectName }}-auth
    spec:
      imagePullSecrets:
        - name: {{ .Values.Deployment.ImagePullSecretsName }}
      securityContext: {}
{{ if .Values.Deployment.enablednodeSelector }}
      nodeSelector:
        node-role.kubernetes.io/worker: {{ .Values.Deployment.nodeSelector
}}
{{ end }}
      serviceAccountName: {{ .Values.Deployment.SA }}
      serviceAccount: {{ .Values.Deployment.SA }}
      containers:
        - name: {{ .Values.CommonValues.MainProjectName }}-auth
          image: {{ .Values.Deployment.ImageNameAuth }}
          imagePullPolicy: Always
          env:
            - name: POD_NAME
              valueFrom:
                fieldRef:
                  fieldPath: metadata.name
            - name: NAMESPACE
```

```
                    valueFrom:
                      fieldRef:
                        fieldPath: metadata.namespace
                  - name: LOG_LEVEL
                    value: {{ .Values.Deployment.LogLevel }}
                  - name: TZ
                    value: "{{ .Values.Deployment.TZ }}"
                envFrom:
                  - configMapRef:
                      name: {{ .Values.CommonValues.MainProjectName }}-auth
                  - secretRef:
                      name: {{ .Values.CommonValues.MainProjectName }}-auth
                ports:
                  - containerPort: {{ .Values.CommonValues.AppPortUi }}
                    protocol: TCP
                resources:
                  limits:
                    cpu: {{ .Values.Deployment.CpuLimit }}
                    memory: {{ .Values.Deployment.MemoryLimit }}
                  requests:
                    cpu: {{ .Values.Deployment.CpuRequest  }}
                    memory: {{ .Values.Deployment.MemoryRequest }}
                livenessProbe:
                  failureThreshold: 3
                  httpGet:
                    path: /health/live
                    port: {{ .Values.CommonValues.AppPortUi }}
                    scheme: HTTP
                  initialDelaySeconds: 600
                  periodSeconds: 30
                  successThreshold: 1
                  timeoutSeconds: 10
                readinessProbe:
                  httpGet:
                    path: /health/ready
                    port: {{ .Values.CommonValues.AppPortUi }}
                    scheme: HTTP
                  failureThreshold: {{ .Values.Deployment.ReadinessProbeFailure-
Threshhold }}
                  initialDelaySeconds: {{ .Values.Deployment.ReadinessProbeI-
nitialDelaySeconds }}
```

```
                    periodSeconds: {{ .Values.Deployment.ReadinessProbePeriodSecond
}}
```

Ingress:

```
        kind: Ingress
        apiVersion: networking.k8s.io/v1
        metadata:
          name: {{ .Values.CommonValues.MainProjectName }}-auth
          namespace: {{ .Values.CommonValues.Namespace }}
          labels:
            app: {{ .Values.CommonValues.MainProjectName }}-auth
            name: {{ .Values.CommonValues.MainProjectName }}-auth
          annotations:
            kubernetes.io/ingress.class: "nginx"
            nginx.ingress.kubernetes.io/backend-protocol: "HTTP"
            nginx.ingress.kubernetes.io/ssl-passthrough: "true"
            nginx.ingress.kubernetes.io/rewrite-target: /
        spec:
          rules:
            - host: {{ .Values.CommonValues.MainProjectName }}-auth.{{ .Values.Com-
monValues.Namespace }}.{{ .Values.Ingress.IngressPrefix }}
              http:
                paths:
                  - path: /
                    pathType: Prefix
                    backend:
                      service:
                        name: {{ .Values.CommonValues.MainProjectName }}-auth
                        port:
                          name: {{ .Values.CommonValues.AppPortUi }}-tcp
          tls:
            - hosts:
                - {{ .Values.CommonValues.MainProjectName }}-auth.{{ .Values.Common-
Values.Namespace }}.{{ .Values.Ingress.IngressPrefix }}
              secretName: {{ .Values.CommonValues.TlsSecretName }}
```

Service:

```
        apiVersion: v1
        kind: Service
        metadata:
          labels:
```

```
    app: {{ .Values.CommonValues.MainProjectName }}-auth
  name: {{ .Values.CommonValues.MainProjectName }}-auth
  namespace: {{ .Values.CommonValues.Namespace }}
spec:
  ports:
    - name: {{ .Values.CommonValues.AppPortUi }}-tcp
      port: 80
      protocol: TCP
      targetPort: {{ .Values.CommonValues.AppPortUi }}
  selector:
    deployment: {{ .Values.CommonValues.MainProjectName }}-auth
  sessionAffinity: None
  type: ClusterIP
```

Secret:

```
apiVersion: v1
kind: Secret
metadata:
  labels:
    app: {{ .Values.CommonValues.MainProjectName }}-auth
  name: {{ .Values.CommonValues.MainProjectName }}-auth
  namespace: {{ .Values.CommonValues.Namespace }}
stringData:
  KEYCLOAK_DATABASE_USER: "{{ .Values.CommonValues.DbUser }}"
  KEYCLOAK_DATABASE_PASSWORD: "{{ .Values.CommonValues.DbPassword }}"
  KEYCLOAK_ADMIN: "{{ .Values.CommonValues.KcAdminUser }}"
  KEYCLOAK_ADMIN_PASSWORD: "{{ .Values.CommonValues.KcAdminPassword }}"
```

г) Helm template для компонента ***brok-ui***

ConfigMap:

```
apiVersion: v1
kind: ConfigMap
metadata:
  labels:
    app: {{ .Values.CommonValues.MainProjectName }}-ui
  name: {{ .Values.CommonValues.MainProjectName }}-ui
  namespace: {{ .Values.CommonValues.Namespace }}
data:
  {{ if .Values.ConfigMap.common.LdapAutorisation.enabled }}
  LOGGING_LEVEL_ROOT: {{ .Values.ConfigMap.common.LdapAutorisation.Kaf-
kaUiLogLevel }}
```

```
        AUTH_TYPE: {{ .Values.ConfigMap.common.LdapAutorisation.AuthType }}
        SPRING_LDAP_URLS: {{ .Values.ConfigMap.common.LdapAutorisa-
tion.SpringLdapUrls }}
        SPRING_LDAP_DN_PATTERN: {{ .Values.ConfigMap.common.LdapAutorisa-
tion.SpringLdapDnPattern }}
        SPRING_LDAP_USERFILTER_SEARCHBASE: {{ .Values.ConfigMap.common.LdapAutori-
sation.SpringLdapUserfilterSearchbase }}
        SPRING_LDAP_USERFILTER_SEARCHFILTER: {{ .Values.ConfigMap.common.LdapAuto-
risation.SpringLdapUserFilterSearchfilter }}
        {{ else if .Values.ConfigMap.common.KeycloakAutorisation.enabled }}
        AUTH_TYPE: {{ .Values.ConfigMap.common.KeycloakAutorisation.AuthType }}
        AUTH_OAUTH2_RESOURCESERVER_JWT_ISSUERURI: {{ .Values.ConfigMap.common.Key-
cloakAutorisation.AuthOauth2ResourcesServerJwtIssUerUri }}
        AUTH_OAUTH2_RESOURCESERVER_JWT_JWKSETURI: {{ .Values.ConfigMap.common.Key-
cloakAutorisation.AuthOauth2ResourcesServerJwtJwkSetUei }}
        {{ else }}
        DATASOURCE_URL: {{ .Values.ConfigMap.common.DatasourceUrl }}
        DATASOURCE_USERNAME: {{ .Values.ConfigMap.common.DatasourceUsername }}
        APPLICATION_PATHS_UI: {{ .Values.ConfigMap.common.ApplicationPathUi }}
        {{ end }}
```

Deployment:

```
      apiVersion: apps/v1
      kind: Deployment
      metadata:
        namespace: {{ .Values.CommonValues.Namespace }}
        name: {{ .Values.CommonValues.MainProjectName }}-ui
        labels:
          commit: {{ .Values.CommonValues.commitTag}}
          app: {{ .Values.CommonValues.MainProjectName }}-ui
          name: {{ .Values.CommonValues.MainProjectName }}-ui
      spec:
        replicas: {{ .Values.Deployment.ReplicasCount }}
        selector:
          matchLabels:
            app: {{ .Values.CommonValues.MainProjectName }}-ui
        template:
          metadata:
            creationTimestamp: null
            labels:
              app: {{ .Values.CommonValues.MainProjectName }}-ui
```

```
                deployment: {{.Values.CommonValues.MainProjectName }}-ui
            spec:
              imagePullSecrets:
                - name: {{ .Values.Deployment.ImagePullSecretsName }}
              securityContext: {}
        {{ if .Values.Deployment.enablednodeSelector }}
              nodeSelector:
                node-role.kubernetes.io/worker: {{ .Values.Deployment.nodeSelector
}}
        {{ end }}
              serviceAccountName: {{ .Values.Deployment.SA }}
              serviceAccount: {{ .Values.Deployment.SA }}
              containers:
                - name: {{ .Values.CommonValues.MainProjectName }}-ui
                  image: {{ .Values.Deployment.ImageNameUi }}
                  imagePullPolicy: Always
                  env:
                    - name: POD_NAME
                      valueFrom:
                        fieldRef:
                          fieldPath: metadata.name
                    - name: NAMESPACE
                      valueFrom:
                        fieldRef:
                          fieldPath: metadata.namespace
                    - name: LOG_LEVEL
                      value: {{ .Values.Deployment.LogLevel }}
                    - name: TZ
                      value: "{{ .Values.Deployment.TZ }}"
                  envFrom:
                    - configMapRef:
                        name: {{ .Values.CommonValues.MainProjectName }}-ui
                    - secretRef:
                        name: {{ .Values.CommonValues.MainProjectName }}-ui
                  ports:
                    - containerPort: {{ .Values.CommonValues.AppPortUi }}
                      protocol: TCP
                  resources:
                    limits:
                      cpu: {{ .Values.Deployment.CpuLimit }}
                      memory: {{ .Values.Deployment.MemoryLimit }}
                    requests:
```

```
                    cpu: {{ .Values.Deployment.CpuRequest  }}
                    memory: {{ .Values.Deployment.MemoryRequest }}
                livenessProbe:
                  failureThreshold: 3
                  httpGet:
                    path: /actuator/health/liveness
                    port: {{ .Values.CommonValues.AppPortUi }}
                    scheme: HTTP
                  initialDelaySeconds: 600
                  periodSeconds: 30
                  successThreshold: 1
                  timeoutSeconds: 10
                readinessProbe:
                  httpGet:
                    port: {{ .Values.CommonValues.AppPortUi }}
                    scheme: HTTP
                  failureThreshold: {{ .Values.Deployment.ReadinessProbeFailure-
Threshhold }}
                  initialDelaySeconds: {{ .Values.Deployment.ReadinessProbeI-
nitialDelaySeconds }}
                  periodSeconds: {{ .Values.Deployment.ReadinessProbePeriodSecond
}}
```

Ingress:

```
      kind: Ingress
      apiVersion: networking.k8s.io/v1
      metadata:
        name: {{ .Values.CommonValues.MainProjectName }}-ui
        namespace: {{ .Values.CommonValues.Namespace }}
        labels:
          name: {{ .Values.CommonValues.MainProjectName }}-ui
        annotations:
          kubernetes.io/ingress.class: nginx
          nginx.ingress.kubernetes.io/backend-protocol: HTTP
          nginx.ingress.kubernetes.io/ssl-passthrough: 'true'

      spec:
        rules:
          - host: {{ .Values.CommonValues.MainProjectName }}.{{ .Values.CommonVal-
ues.Namespace }}.{{ .Values.Ingress.IngressPrefix }}
            http:
```

```
              paths:
                - path: /
                  pathType: Prefix
                  backend:
                    service:
                      name: {{ .Values.CommonValues.MainProjectName }}-ui
                      port:
                        name: {{ .Values.CommonValues.AppPortUi }}-tcp
        tls:
          - hosts:
              - {{ .Values.CommonValues.MainProjectName }}.{{ .Values.CommonVal-
ues.Namespace }}.{{ .Values.Ingress.IngressPrefix }}
            secretName: {{ .Values.Ingress.TlsSecretName }}
```

Service:

```
      apiVersion: v1
      kind: Service
      metadata:
        labels:
          app: {{ .Values.CommonValues.MainProjectName }}-ui
        name: {{ .Values.CommonValues.MainProjectName }}-ui
        namespace: {{ .Values.CommonValues.Namespace }}
      spec:
        ports:
          - name: {{ .Values.CommonValues.AppPortUi }}-tcp
            port: 80
            protocol: TCP
            targetPort: {{ .Values.CommonValues.AppPortUi }}
        selector:
          deployment: {{ .Values.CommonValues.MainProjectName }}-ui
        sessionAffinity: None
        type: ClusterIP
```

2) Для Deploy с использованием всех template указанных выше рекомендуется следующий **values** файл:

CommonValues:

```
        MainProjectName: <APP_NAME> # Example: brok. Название приложения исполь-
зумое в лейблах манифестов
        commitTag: <COMIT_TAG>  # Example: lates.  Дефолтное значение лейбла ис-
пользуемого для автоматического редеплоя компонентов приложения
```

```
        Namespace: <NAMESPACE> # целевой namespace в котором будет развернуты поды brok
        AppPortApi: "8081" # Порт пода с приложением brok
        AppPortBpm: "8082" # Порт пода с приложением brok-bpm
        AppPortUi: "8080" # Порт пода с приложениями brok-ui и brok-auth
        KcDbHost: <POSTGRES_HOST> # Example: postgres.ru-central1.internal Адрес базы данных используемой brok-auth для хранения списка пользователей
        KcDbPort: "5432" # Порт базы данных используемой brok-auth для хранения списка пользователей
        DbVendor: "postgresql" # Тип используемой базы данных
        DbUser: <DB_USER> # Example: brok_demo - Учетная запись пользователя используемая для подключения к базе данных
        DbPassword: <DB_USER_PASSWORD> #Example: SupperP@$$w0rD Пароль от учетной записи пользователя используемой для подключения к базе данных
        DbDatabase: <DB_BROK> # Example: brok_demo  База данных используемая для хранения списка пользователей
        KcDbSchema: <BROK_AUTH_SCHEMA> # Example: brok_auth Схема в базе даннхы "DbDatabase" используемая для хранения списка пользователей
        KcAdminUser: <ADMIN_USER_BROK_AUTH> # Example: admin Адмнская учетная запись, используемая для входа в UI brok-auth
        KcAdminPassword: <ADMIN_USER_BROK_AUTH_PASSWORD> #Example: SupperP@$$w0rDR0K@uth Пароль от адмнской учетной записи, используемой для входа в UI brok-auth
        KcEnableHealthEndpoints: "true" # Включение эндпоинтов Readiness Checks, Liveness Check
```

Ingress:

```
        IngressPrefix: <K8S_PREFIX> # Example: apps.dev.k8s-javax Префикс кластера k8s
        TlsSecretName: <TLS_SECRET_NAME> # Example: tls-secret Название секрета с сертификатами
```

Deployment:

```
        enablednodeSelector: True
        nodeSelector: <NODESELECTOR> # Example: worker
        SA: <SERVICE_ACCOUNT_NAME> # Example: deployer-service Название сервис-аккаунта, используемого для запуска приложений в k8s
        ImagePullSecretsName: <IMAGE_PULL_SECRET> # Example: docker-registry-pull-secret Название секрета, используемого для пула Docker image из Registry
        ImageNameBase: <IMAGE_NAME_CORE> # Example: "nexus.ru-central1.internal:5000/brok-api:demo"  Docker image используемого в приложении brok
```

```
        ImageNameUi: <IMAGE_NAME_UI>  # Example: "nexus.ru-central1.inter-
nal:5000/brok-ui:demo"  Docker image используемого в приложении brok-ui
        ImageNameAuth: <IMAGE_NAME_AUTH>  # Example: "nexus.ru-central1.inter-
nal:5000/brok-auth:demo"  Docker image используемого в приложении brok-auth
        ImageNameBpm: <IMAGE_NAME_BPM>  # Example: "nexus.ru-central1.inter-
nal:5000/brok-bpm:demo" Docker image используемого в приложении brok-bpm

        LogLevel: DEBUG # Уровень логирования в приложениях brok-ui, brok.
        CpuLimit: "600m" # Рекомендуемый показатель CpuLimit
        CpuRequest: "300m" # Рекомендуемый показатель для CpuRequest
        MemoryLimit: "4Gi" # Рекомендуемый показатель для MemoryLimit
        MemoryRequest: "2Gi" # Рекомендуемый показатель для MemoryRequest
        TZ: "Europe/Moscow" # Time Zone используемая в приложениях
        ReplicasCount: "1" #  ReplicasCount приложений
        ReadinessProbeFailureThreshhold: "1" # Рекомендуемое время ответа для
ReadinessProbe
        ReadinessProbeInitialDelaySeconds: "30" # Рекомендуемое время начала про-
верки ReadinessProbe
        ReadinessProbePeriodSecond: "30" # Рекомендуемое время периодичности про-
верок ReadinessProbe
```

ConfigMap:

```
        common:
          DatasourceUrl: <POSTGRES_HOST> # Example: "jdbc:postgresql://post-
gres.ru-central1.internal/brok_demo" Адрес базы данных используемой brok
          DatasourceUsername:  <DB_USER> # Example: brok_demo Учетная запись поль-
зователя используемая для подключения к базе данных
          DatasourcePassword: <DB_USER_PASSWORD> #Example: SupperP@$$w0rD Пароль
от учетной записи пользователя используемой для подключения к базе данных
          application:
            paths: # Адреса балансировщиков компонентов приложения
              ui: http://brok-ui
              core: http://brok-core
              bpm: http://brok-bpm
          license:
            type: "KUBERNETES" # Тип используемой лицензии
            kubernetes:
              host: <EXTERNAL_IP_BALANCER> # Example: "https://11.222.33.444:6443"
Внешний адрес балансировщика оркестратора
              namespace: <BROK_NAMESPACE> # Example: demo название целевого
namespace
```

```
              apiVersion: # Example: "v1" apiVersion сущности Namespace в оркест-
раторе
              token: <SA_TOKEN> # Токен доступа сервисного аккаунта используемый
для проверки лицензии не в base64
            key: <BROK_LIC_KEY> # Ключ лицензии
          botName: "UiKafkaBot" # Имя бота технической поддержки
          botToken: "6462748878:AAHNx2Cdb73MPRgL1uNd5hAzdPMo-HM6w2Q" # Токен бота
технической поддержки
          botChatID: "-1001939060949" # ID чата с  ботом технической поддержки
          botEnabled: true # Фичетогл управляющий наличием подключения с телеграм
ботом
          camunda: # Фичетогл управляющий наличием компонента brok-bpm
            bpm:
              enabled: true
          cache:
            Enabled: true # Фичетогл управляющий использованием кеша
            properties: # Настройка кеширования
              clusters:
                Enabled: true
              topics:
                Enabled: true
              favorites:
                Enabled: true
              templates:
                Enabled: true
          audit: # Фичетогл управляющий аудитом в приложении
            Enabled: true
          cleaner:
            cronExpression: "0 0 0 * * ?" # Cron расписание очистки кеша
            audit:  # Фичетогл управляющий очисткой аудита
              Enabled: true
              count: 100
          LdapAutorisation:
            enabled: False # Тогл переключения на использование LDAP авторизации
true/false.
            KafkaUiLogLevel: "DEBUG" #  Уровень логирования в приложениях brok.
            AuthType: "LDAP" # Тип аутентификации пользователей.
            SpringLdapUrls: "ldap://<FQDN>:<PORT>" # Адрес сервера LDAP.
            SpringLdapDnPattern: "cn=accounts,dc=<DOMEN_P1>,dc=<DOMEN_P2>" # Цели
для поиска пользователей.
            SpringLdapUserfilterSearchbase: "dc=<DOMEN_P1>,dc=<DOMEN_P2>" # Фильтр
для поиска пользователей на сервере LDAP.
```

```
            SpringLdapUserFilterSearchfilter: "(&(uid={0})(objectClass=inetOrgPer-
son))" # Фильтр для поиска пользователей на сервере LDAP.
            SpringLdapAdminuser: "" # Учетная запись администратора LDAP использу-
емая для подключения к серверу LDAP.
            SpringLdapAdminPassword: "" # Пароль от учетной записи администратора
LDAP используемая для подключения к серверу LDAP.
          KeycloakAutorisation:
            enabled: True
            AuthType: "KEYCLOAK" # Тип аутентификации пользователей.
            AuthOauth2ResourcesServerJwtIssUerUri: "http://brok-auth/realms/brok-
realm" # Внутренние URL адреса используемые для получения информации о пользовате-
лях приложением brok-auth
            AuthOauth2ResourcesServerJwtJwkSetUei: "http://brok-auth/realms/brok-
realm/protocol/openid-connect/certs" # Внутренние URL адреса используемые для полу-
чения информации о пользователях приложением brok-auth

        auth:
          ProxyAddressForwarding: true
          StartPageHost: "http://brok-ui/" # # Внутрений url используемый приложе-
нием brok-auth для взайимодействия.
          KeycloakLoglvl: "INFO" #  Уровень логирования в приложениях brok-auth.
          RootLoglvl: "INFO" #  Уровень логирования в приложениях brok-auth.

        bpm:
          datasource:
            schema: <BPM_SCHEMA> # Example:  brok_bpm названиe схемы в БД
          camunda:
            adminUser:
              id: <BROK_BPM_USER> # admin логин тех пользователя для brok-bpm
              password: <BROK_BPM_PASSWORD> # Dvlre1pow3 пароль тех пользователя
для brok-bpm

       sa:
         saName: <SA_NAME> # Example: brok-sa. Имя сервисного аккаунта от которого
будет осуществляться проверка лицензии
         roleName: <ROLE_NAME> # Example: brok-sa-role Имя роли, с которой будет
рабоатать сервисный аккаунт
         roleBindingName: <ROLE_BINDING_NAME> #Example: brok-sa-role-binding Имя
RoleBinding для сервисного аккаунта
```

3) Для Deploy приложения в кластер k8s с использованием pipeline рекомендуется использовать систему управления конфигурациями — Ansible, в частности модуль ***kubernetes.core.k8s***.

4) Допускается использования всех template, указанных выше для не автоматического (ручного) Deploy приложения с помощью утилиты Helm.

5) Дополнительно существует кейс по автоматизации процесса получения актуальной версии приложения. Способ реализации данного кейса в рамках настоящей инструкции не рассматривается и обговаривается отдельно.