ИНСТРУКЦИЯ ПО УСТАНОВКЕ ЭКЗЕМПЛЯРА ПО

Программное обеспечение Джим («Jim»)

Аннотация

В документе приведена инструкция по установке экземпляра программного обеспечения «Джим (Jim)» (далее – Система).

Содержание

Аннотация 2
1. Общий порядок развертывания ПО «Джим (Jim)» 4
1.1. Подготовка PostgreSQL 4
1.2. Настройка доступа к PostgreSQL 5
1.3. Подготовка среды оркестрации Kubernetes 5
1.4. Создание Deployment-ресурсов модулей приложения 7
1.5. Создание Service-объектов 11

1. Общий порядок развертывания ПО «Джим (Jim)»

Программное обеспечение «Джим» (далее – ПО) предназначено для работы в среде оркестрации Kubernetes. Корректная установка включает последовательность действий, необходимых для подготовки инфраструктуры и последующего развертывания всех модулей системы.

Общий порядок развертывания содержит следующие действия:

– Развернуть СУБД PostgreSQL и создать необходимые базы данных, пользователей и их права доступа;

– Применить Liquibase миграции для создания схемы, таблиц и заполнения первичными данными;

– Создать Deployment-ресурсы в Kubernetes для модулей:

  – JIM Core,

  – JIM UI,

  – Camunda,

  – Connector Server;

– Настроить таблицу system_configuration, обеспечивающую работу всех модулей приложения;

– Настроить таблицу connection_properties для конфигурации коннекторов и подключения к целевым системам.

1.1. Подготовка PostgreSQL

ПО «Джим» использует две отдельные базы данных:

– jim – основная база ядра приложения;

– jimcamunda – база процессов Camunda.

Для доступа к ним необходимо создать две роли с правами логина, например: jim и camunda. Роли получают привилегированный доступ к своим соответствующим базам.

Пример SQL-команд:

```
CREATE ROLE jim WITH LOGIN PASSWORD ‘password’;
CREATE DATABASE jim;
GRANT ALL PRIVILEGES ON DATABASE jim TO jim;
```

```
CREATE ROLE camunda WITH LOGIN PASSWORD ‘password’;
CREATE DATABASE jimcamunda;
GRANT ALL PRIVILEGES ON DATABASE jimcamunda TO camunda;
```

1.2. Настройка доступа к PostgreSQL

Для обеспечения подключения модулей ПО к PostgreSQL необходимо отредактировать файл конфигурации pg_hba.conf, разрешив подключение ролей jim и camunda по паролю из диапазона адресов, используемых Kubernetes-кластером.

1.3. Подготовка среды оркестрации Kubernetes

Для обеспечения безопасной передачи конфиденциальных параметров в Deployment-ресурсы, а также для корректной загрузки контейнерных образов из внешних и внутренних репозиториев, необходимо создать ряд Secret-объектов в Kubernetes.

Следующие Secret-ресурсы являются обязательными:

– docker-secret – используется для аутентификации в внешнем Docker Hub при загрузке образов;

– registry-secret – предназначен для аутентификации во внутреннем корпоративном Docker-репозитории;

– jim-core-secret – хранит чувствительные параметры, необходимые для работы ядра приложения (JIM Core)

  – JIM_AES_KEYSTORE_PASSWORD
  – JIM_DB_PASSWORD

– jim-camunda-secret - используется модулем Camunda

  – JIM_API_PASSWORD
  – DB_PASSWORD

– jim-ui-secret - содержит параметры доступа пользовательского интерфейса

  – JIM_API_PASSWORD

Также для осуществления проксирования внутреннего трафика

приложения модулем nginx необходимо создать следующий ресурс типа ConfigMap:

- nginx-config
    - nginx.conf

Пример содержимого этого ConfigMap:

```
worker_processes 1;

events {
  worker_connections 1024;
}

http {
  include      mime.types;
  default_type  application/octet-stream;

  map $http_x_role $target_upstream {
    default    default_upstream;
    # при наличии других экземпляров ядра ПО здесь можно разместить маппинг на их апстримы
  }

  # здесь можно разместить апстримы на другие экземпляры ядра ПО

  upstream default_upstream {
    server jim-core-svc.core.svc.cluster.local:8085;
  }

  upstream camunda_upstream {
    server jim-camunda-svc.camunda.svc.cluster.local:8080;
  }

  server {
    listen 80;

    location ^~ /engine-rest/ {
      proxy_pass http://camunda_upstream;
      proxy_set_header Host $host;
      proxy_set_header X-Real-IP $remote_addr;
      proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
      proxy_set_header X-Forwarded-Proto $scheme;
    }

    location / {
      proxy_pass https://$target_upstream;
      proxy_set_header Host $host;
      proxy_set_header X-Real-IP $remote_addr;
      proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
      proxy_set_header X-Forwarded-Proto $scheme;

      if ($target_upstream = "") {
        return 400 "Missing or invalid X-Role header";
      }
    }
  }
```

```
  }
}
```

1.4. Создание Deployment-ресурсов модулей приложения

После подготовки баз данных, применения миграций и создания необходимых Secret-объектов следующим шагом является развертывание модулей ПО в кластере Kubernetes. Для этого создаются Deployment-ресурсы, определяющие контейнерные образы, параметры запуска, конфигурацию переменных окружения, а также ссылки на ранее созданные Secrets и ConfigMap-объекты. Каждый Deployment отвечает за запуск и масштабирование отдельных компонентов системы, обеспечивая их изолированную работу и возможность гибкого управления в рамках оркестрации.

Должен быть создан следующий набор Deployment-ресурсов:

– jim-core – ядро ПО «Джим»;

– nginx-proxy – модуль nginx, осуществляющий проксирования трафика между инстансами ядра приложения и Camunda;

– jim-ui – пользовательский интерфейс ПО «Джим»;

– jim-camunda – модуль Camunda;

– jim-connector-server – сервер коннекторов для подключения к целевым системам.

Пример Deployment-ресурса ядра приложения с ролью default:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: jim-core
  namespace: core
  labels:
    app.kubernetes.io/name: jim-core
spec:
  replicas: 1
  selector:
    matchLabels:
      app.kubernetes.io/name: jim-core
  template:
    metadata:
      labels:
        app.kubernetes.io/name: jim-core
    spec:
```

```
    imagePullSecrets:
      - name: dit-registry-secret
    containers:
      - name: jim-core
        image: registry.example.ru/docker/jim-core:2025.11.26-1941
        imagePullPolicy: IfNotPresent
        env:
          - name: APP_ROLE
            value: default
          - name: JIM_DB_SCHEMA
            value: jim
          - name: JIM_DB_DRIVER
            value: org.postgresql.Driver
          - name: JIM_DB_URL
            value: jdbc:postgresql://jim-db-svc.db.svc.cluster.local:5432/jim
          - name: JIM_DB_USER
            value: viadmin
          - name: JIM_DB_PASSWORD
            valueFrom:
              secretKeyRef:
                name: jim-core-secret
                key: JIM_DB_PASSWORD
          - name: JIM_JOOQ_SQL_DIALECT
            value: Postgres
          - name: JIM_PARALLEL_POOLSIZE
            value: "10"
          - name: JIM_AES_KEYSTORE_PASSWORD
            valueFrom:
              secretKeyRef:
                name: jim-core-secret
                key: JIM_AES_KEYSTORE_PASSWORD
        ports:
          - containerPort: 8085
        livenessProbe:
          httpGet:
            path: /check/liveness
            port: 8085
            scheme: HTTPS
        readinessProbe:
          httpGet:
            path: /check/readiness
            port: 8085
            scheme: HTTPSc
```

При развертывании большего количества экземпляров ядра ПО «Джим» манифест Deployment-ресурса будет отличаться только в двух параметрах:

– Параметр name Deployment-ресурса

– Передаваемый в контейнер Environment Variable APP_ROLE, который определяет, с какой ролью будет запускаться экземпляр приложения

Пример Deployment-ресурса пользовательского интерфейса ПО:

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

```
metadata:
  name: jim-ui
  namespace: ui
  labels:
    app.kubernetes.io/name: jim-ui
spec:
  replicas: 1
  selector:
    matchLabels:
      app.kubernetes.io/name: jim-ui
  template:
    metadata:
      labels:
        app.kubernetes.io/name: jim-ui
    spec:
      imagePullSecrets:
        - name: dit-registry-secret
      containers:
        - name: jim-ui
          image: registry.example.ru/docker/jim-ui:2025.11.26-1941
          imagePullPolicy: IfNotPresent
          env:
            - name: JIM_API_URL
              value: http://nginx-proxy-svc.core.svc.cluster.local
            - name: JIM_API_PORT
              value: "80"
            - name: JIM_API_USER
              value: admin
            - name: JIM_API_PASSWORD
              valueFrom:
                secretKeyRef:
                  name: jim-ui-secret
                  key: JIM_API_PASSWORD
          ports:
            - containerPort: 8080
```

Пример Deployment-ресурса модуля Camunda:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: jim-camunda
  namespace: camunda
  labels:
    app.kubernetes.io/name: jim-camunda
spec:
  replicas: 1
  selector:
    matchLabels:
      app.kubernetes.io/name: jim-camunda
  template:
    metadata:
      labels:
        app.kubernetes.io/name: jim-camunda
    spec:
      imagePullSecrets:
        - name: dit-registry-secret
      containers:
```

```
      - name: jim-camunda
        image: registry.example.ru/docker/jim-camunda:2025.11.26-1941
        imagePullPolicy: IfNotPresent
        env:
          - name: DB_DRIVER
            value: org.postgresql.Driver
          - name: DB_URL
            value: jdbc:postgresql://jim-db-svc.db.svc.cluster.local:5432/camunda
          - name: DB_USERNAME
            value: camunda
          - name: DB_PASSWORD
            valueFrom:
              secretKeyRef:
                name: jim-camunda-secret
                key: DB_PASSWORD
          - name: JIM_API_URL
            value: http://nginx-proxy-svc.core.svc.cluster.local
          - name: JIM_API_PORT
            value: "80"
          - name: JIM_API_USER
            value: admin
          - name: JIM_API_PASSWORD
            valueFrom:
              secretKeyRef:
                name: jim-camunda-secret
                key: JIM_API_PASSWORD
        ports:
          - containerPort: 8080
```

Пример Deployment-ресурса модуля сервера коннекторов:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: jim-connector-server
  namespace: connectorserver
  labels:
    app.kubernetes.io/name: jim-connector-server
spec:
  replicas: 1
  selector:
    matchLabels:
      app.kubernetes.io/name: jim-connector-server
  template:
    metadata:
      labels:
        app.kubernetes.io/name: jim-connector-server
    spec:
      imagePullSecrets:
        - name: dit-registry-secret
      containers:
        - name: jim-connector-server
          image: registry.example.ru/docker/jim-connector-server:2025.11.26-1941
          imagePullPolicy: IfNotPresent
          ports:
            - containerPort: 8759
```

1.5. Создание Service-объектов

Для каждого Deployment-ресурса необходимо создавать отдельный Service-объект, который обеспечивает сетевой доступ к запущенным контейнерам и позволяет другим модулям корректно взаимодействовать с приложением внутри кластера.

Ниже приведён пример Service-ресурса для компонента JIM Core:

```
apiVersion: v1
kind: Service
metadata:
  name: jim-core-svc
  namespace: core
spec:
  type: ClusterIP
  selector:
    app.kubernetes.io/name: jim-core
  ports:
    - name: http
      port: 8085
      targetPort: 8085
```

Пример Service-ресурса для компонента JIM UI:

```
apiVersion: v1
kind: Service
metadata:
  name: jim-ui-svc
  namespace: ui
spec:
  type: ClusterIP
  selector:
    app.kubernetes.io/name: jim-ui
  ports:
    - name: http
      port: 8080
      targetPort: 8080
```

Пример Service-ресурса для компонента Nginx:

```
apiVersion: v1
kind: Service
metadata:
  name: nginx-proxy-svc
  namespace: core
spec:
  type: ClusterIP
  selector:
    app: nginx-proxy
  ports:
    - name: http
      port: 80
      targetPort: 80
```

```
  - name: http2
    port: 8080
    targetPort: 8080
```

Пример Service-ресурса для компонента Camunda:

```
apiVersion: v1
kind: Service
metadata:
  name: jim-camunda-svc
  namespace: camunda
spec:
  type: ClusterIP
  selector:
    app.kubernetes.io/name: jim-camunda
  ports:
    - name: http
      port: 8080
      targetPort: 8080
```

Пример Service-ресурса для компонента Connector Server:

```
apiVersion: v1
kind: Service
metadata:
  name: jim-connector-server-svc
  namespace: connectorserver
spec:
  type: ClusterIP
  selector:
    app.kubernetes.io/name: jim-connector-server
  ports:
    - name: openicf
      port: 8759
      targetPort: 8759
```

Для обеспечения внешнего доступа к компонентам приложения, развернутого в кластере Kubernetes, необходимо создать соответствующие ресурсы Ingress. Эти ресурсы определяют правила маршрутизации HTTP(S)-трафика и направляют запросы к нужным Service-объектам приложения. Требуется создать следующие Ingress-объекты:

– jim-core-ingress – обеспечивает доступ к Nginx Proxy, который, в зависимости от URL и переданных заголовков X-Role, выполняет маршрутизацию трафика либо к модулю Camunda, либо к одному из экземпляров ядра ПО «Джим».

– jim-ui-ingress – предназначен для предоставления доступа к

пользовательскому интерфейсу системы.

- camunda-ingress – для доступа к веб-интерфейсу модуля Camunda.

Пример манифеста для ingress-контроллера jim-core-ingress:

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: jim-core-ingress
  namespace: core
spec:
  defaultBackend:
    service:
      name: nginx-proxy-svc
      port:
        number: 80
  ingressClassName: nginx
  rules:
    - host: jim-core.apps.idm-stend-k8s.jet.msk.su
      http:
        paths:
          - backend:
              service:
                name: nginx-proxy-svc
                port:
                  number: 80
            path: /
            pathType: Prefix
```

Пример манифеста для ingress-контроллера jim-ui-ingress:

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: jim-ui-ingress
  namespace: ui
spec:
  defaultBackend:
    service:
      name: jim-ui-svc
      port:
        number: 8080
  ingressClassName: nginx
  rules:
    - host: jimui.apps.idm-stend-k8s.jet.msk.su
      http:
        paths:
          - backend:
              service:
                name: jim-ui-svc
                port:
                  number: 8080
            path: /
            pathType: Prefix
```

Пример манифеста для ingress-контроллера camunda-ingress:

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: camunda-ingress
  namespace: camunda
spec:
  defaultBackend:
    service:
      name: jim-camunda-svc
      port:
        number: 8080
  ingressClassName: nginx
  rules:
    - host: camunda.apps.idm-stend-k8s.jet.msk.su
      http:
        paths:
          - backend:
              service:
                name: jim-camunda-svc
                port:
                  number: 8080
            path: /
            pathType: Prefix
```

После создания всех перечисленных ресурсов – баз данных, Secret-объектов, Deployment- и Service-ресурсов, а также соответствующих Ingress-настроек – приложение будет полностью готово к работе.

Если конфигурация выполнена корректно и все предварительные условия соблюдены, можно открыть адрес, указанный в Ingress-ресурсе пользовательского интерфейса. После успешной авторизации станут доступны основные функции системы.