УДК 004

Развертывание Java приложения в Kubernetes

Еровлева Регина Викторовна
Приамурский государственный университет имени Шолом-Алейхема
Студент

Еровлев Павел Андреевич
Приамурский государственный университет имени Шолом-Алейхема
Студент

Аннотация
В данной статье будут рассмотрены работы по выполнению развертывания приложения в Kubernetes. Будет произведена настройка связных компонентов для соединения развернутых модулей. Итоговым результатом будет являться развернутое приложение.
Ключевые слова: Kubernetes, Java, Deploying

Deploying your Java application in Kubernetes

Erovleva Regina Viktorovna
Sholom-Aleichem Priamursky State University
Student

Erovlev Pavel Andreevich
Sholom-Aleichem Priamursky State University
Student

Abstract
This article will cover the work of deploying an application to Kubernetes. Connected components will be configured to connect the deployed modules. The final result will be a deployed application.
Keywords: Kubernetes, Java, Deploying

В процесс разработки приложения и его запуска включен один из немаловажных аспектов – это развертывание. Обычно развертывание приложения производился ручным способом, когда сам программист, либо системный инженер создавали необходимую структуру для запуска приложения. Сейчас же технологии шагнули вперед и теперь нет необходимости создавать уже готовое. Можно воспользоваться услугами облачных провайдеров, у которых есть услуги под каждый определенный аспект.

Kubernetes – это открытое ПО для автоматического развертывания контейнеризированных приложений.

Цель данной статьи – развернуть Java приложение с использованием Kubernetes.

В своей статье А.И. Егунова, А.А. Аббакумов, М.А. Воропаева, Ю.С. Вечканова Предложили реализацию поисковой системы и информационного хранилища в виде J2EE-приложения с использованием фреймворка Spring [1]. В своей статье В.С. Жданова рассмотрела подход в создании бекэнда клиент-серверного приложения для мобильных устройств для просмотра расписания [2]. А.Д. Нарижный, Н.Е. Губенко произвели сравнительный анализ технологий, которые предназначены для одних и тех же задач и имеют схожую функциональность. Это стеки технологий Spring и JavaEE, которые предназначены для разработки Enterprise-приложений [3]. В.Л. Волушкова рассмотрела в своей работе технологии программирования на примере языка Java [4]. П.В. Прохоров, Н.В. Разговоров рассмотрели в своей работе современные подходы и технологии в разработке серверных приложений на примере онлайн-магазина с использованием Spring [5].

Для начала запустим кластер приложения в Kubernetes (k8s). Для работы с развертыванием приложений необходимо создать конфигурацию типа развертывания k8s. Это поможет создавать и обновлять экземпляры приложений и, кроме того, помогает в кластере приложений подготовки, включая модуль, службу, набор реплик и контроллер репликации.

В этой настройке необходимо настроить «Load balancer» поверх данного приложения, которое будет перенаправлять трафик на набор модулей, а позже они будут общаться с внутренними модулями. Связь между модулями происходит через объект службы, встроенный в Kubernetes (рис.1).

```
apiVersion: v1
kind: Service
metadata:
 name: dev-nginx
  labels:
    k8s-app: dev-api
spec:
 type: NodePort
 ports:
 - port: 8080
    nodePort: 31900 name: devnginx
 selector:
  k8s-app: appname
  component: nginx
  env: dev
```

Рисунок 1 – Файл Yaml балансировщика нагрузки Nginx

Первым шагом в переходе на Kubernetes является создание пода, в котором размещается контейнер приложения. Но так как поды эфемерны по своей природе, нужно создать контроллер более высокого уровня, который будет следить за подом: перезапустит, создаст связь с узлами (рис.2).

```
apiVersion:v1
kind:ReplicationController
 metadata:
   name: appname
spec:
 replicas:replica_count
 template:
   metadata:
    name: appname
    labels:
      k8s-app:   appname
      component:nginx
      env:env_name
spec:
  nodeSelector:
   resource-group: dev
 containers:
  - name: appname
  image:IMAGE_TEMPLATE
  imagePullPolicy:Always
  ports:
   -containerPort: 8080
   resources:
    requests:
     memory: "request_mem"
     cpu: "request_cpu"
    limits:
      memory: "limit_mem"
      cpu: "limit_cpu"
    env: - name: BACKEND_HOST
    value: dev-env_name-node:3000
```

Рисунок 2 – Контроллер репликации Nginx Yaml

Теперь, чтобы создать вход, нужно сначала создать службу интерфейса (рис.3).

```
apiVersion: v1
kind: Service
metadata:
 name: appname
  labels:
    k8s-app: appname
spec:
 type: NodePort
 ports
 - name:http
   port: 3000
   protocol:TCP
   targetPort: 3000
 selector:
  k8s-app: appname
  component: java
  env: dev
```

Рисунок 3 – Файл Yaml службы внешнего интерфейса

Так же создадим файл пода для настройки данного интерфейса (рис.4).

```
apiVersion:v1
kind:ReplicationController
 metadata:
   name: Frontend
spec:
 replicas:3
 template:
   metadata:
    name: Frontend
    labels:
      k8s-app: Frontend
      component:java
      env:dev
spec:
  nodeSelector:
   resource-group: dev
 containers:
  - name: appname
  image:IMAGE_TEMPLATE
  imagePullPolicy:Always
  ports:
   -containerPort:3000
   resources:
    requests:
     memory: "request_mem"
     cpu: "request_cpu"
    limits:
      memory: "limit_mem"
      cpu: "limit_cpu"
    env:
     -name: ENV
     valueFrom:
     configMapKeyRef:
     name:appname
     Key:congfig-env
```

Рисунок 4 – Yaml-файл внешнего контроллера репликации

Теперь осталось создать службу бекэнда и его под (рис.5-6).

```
apiVersion: v1
kind: Service
metadata:
 name: backend
  labels:
    k8s-app: backend
spec:
 type: NodePort
 ports
 - name:http
   port: 9010
   protocol:TCP
   targetPort: 9000
 selector:
  k8s-app: appname
  component: play
  env: dev
```

Рисунок 5 – Файл Yaml серверной службы

```
apiVersion:v1
kind:ReplicationController
 metadata:
   name: backend
spec:
 replicas:3
 template:
   metadata:
    name: backend
    labels:
      k8s-app: backend
      component:play
      env:dev
spec:
  nodeSelector:
   resource-group: dev
 containers:
  - name: appname
  image:IMAGE_TEMPLATE
  imagePullPolicy:Always
  ports:
   -containerPort:9000
   command: [ "./docker-entrypoint.sh" ]
   resources:
    requests:
     memory: "request_mem"
     cpu: "request_cpu"
    limits:
      memory: "limit_mem"
      cpu: "limit_cpu"
   volumeMounts:
     - name: config-volume
     mountPath: /app/vipin/play/conf
 volumes:
   -name:config-volume
  configMap:
  name: appname
```

Рисунок 6 – Файл Yaml поддерживаемого контроллера репликации

После сохранения всех настроек, приложение становится развернутым, а все данные можно получать через интерфейс Kubernetes.

В данной статье был рассмотрен процесс развертывания Java приложения в Kubernetes.

Библиографический список

1. Егунова А.И., Аббакумов А.А., Воропаева М.А., Вечканова Ю.С. Система управления хранилищем электронных образовательных ресурсов // Образовательные технологии и общество. 2019. №3. С. 145-154.
2. Жданова В.С. Серверный модуль системы уведомления об изменениях в расписании занятий // Молодость. Интеллект. Инициатива. 2017. С. 21-22.
3. Нарижный А.Д., Губенко Н.Е. Сравнительный анализ стеков технологий spring и javaee (jakartaee) для разработки enterprise приложений // Информатика, управляющие системы, математическое и компьютерное моделирование 2020. №3. С. 549-462.
4. Волушкова В.Л. Архитектурные решения java для доступа к данным // Теоретические основы программирования. Учебное пособие 2019. 137с.
5. Прохоров П.В., Разговоров Н.В. Современные подходы в backend разработке на примере онлайн-магазина // Прикладная математика и фундаментальная информатика. 2020. №2. С. 23-28.