SpringCloud微服务容器云进阶之路

1. Springboot应用配合Actuator开启： `监控检查` ， `优雅停机` ， `监控metrics` 等endpoints
2. 根据 `Dockerfile` 定义制作Docker镜像并上传 `Harbor` 私有Docker Registry
3. 渲染K8S部署模板文件并完成应用部署，同时考虑快速回滚等保障机制

> **Tips:** 步骤2 - 可使用Maven Plugin dockerfile-maven集成到Maven流程中，命令如: `mvn dockerfile:build` , `mvn dockerfile:push` , 详见官方文档

1、Springboot配置示例

```yaml
spring:
  application:
    name: spring-produce
  profiles:
    active: ${ENVIRONMENT:pro}
server:
  port: 10080
management:
  endpoints:
    web:
      base-path: /actuator/
      exposure:
        include: health,shutdown,prometheus # 监控检查，优雅停机，监控metrics
  endpoint:
    shutdown:
      enabled: true
  metrics:
    tags:
      application: ${spring.application.name} # 监控metrics Tag
eureka:
  instance:
    preferIpAddress: true
```

```
  client:
    serviceUrl:
      defaultZone: http://eureka-0.eureka.micro-public.svc.cluster.local:8761/eureka,http://eureka-1.eureka.micro-public.svc.cluster.local
logging:
  file:
    max-size: 200MB
    max-history: 7
    path: /opt/logs
    name: /opt/logs/${HOSTNAME}.${spring.application.name}.log
```

注意：actuator开启prometheus需要添加依赖

```
<dependency>
  <groupId>io.micrometer</groupId>
  <artifactId>micrometer-registry-prometheus</artifactId>
  <version>${micrometer.version}</version>
</dependency>
```

2、Dockerfile

> 在项目根目录下创建 Dockerfile

```
# base镜像 - JDK发行版
FROM adoptopenjdk:8u252-b09-jdk-hotspot
# 工作目录 - 即jar包所在目录
WORKDIR /opt
# mvn clean package -Dmaven.test.skip=true
## 服务版本
COPY target/produce-1.0.1.jar .
# 更改镜像时区
RUN echo "Asia/Chongqing" > /etc/timezone
# 暴露服务端口
```

```
EXPOSE 10080
# 服务启动命令
## 1、${JVM_OPTS} - JVM配置，如：-Xms768m -Xmx768m -javaagent:/usr/skywalking/agent/skywalking-agent.jar
## 2、${APP_OPTS} - APP配置，如：--spring.profiles.active=dev --spring.kafka.consumer.group-id=xxx.group
CMD [ "sh", "-c", "java ${JVM_OPTS} -jar produce-1.0.1.jar ${APP_OPTS}" ]
```

3、K8S容器云部署文件模板

> 在项目根目录下创建 `manifests` 目录，在目录下创建文件 `k8s.yaml`

1. **服务名称** - 全局替换 `<change-me>` 为您的服务名称
2. **服务端口** - 全局替换 `10080` 为您的服务端口
3. **服务资源** - 注意 `resources` 字段服务所申请的资源
4. **服务域名** - 根据实际情况，选择是否需要对外暴露Ingress
5. **服务版本** - 根据项目pom文件 `version` 字段，同时修改Dockerfile中jar包版本

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: <change-me>-deployment # deployment名称
  annotations:
    kubernetes.io/change-cause: <CHANGE_CAUSE> # 版本说明 - 用于回滚等
spec:
  selector:
    matchLabels:
      app: <change-me> # 标签选择器，与下面[Flag_label]对应
  replicas: 1 # 多实例
  template:
    metadata:
      labels:
        app: <change-me> # [Flag_label]
    spec:
      initContainers: # skywalking-agent initContainer
```

```
  - image: registry.meitianiot.lo/public/skywalking-agent:8.1.0
    name: skywalking-agent
    imagePullPolicy: IfNotPresent
    command: ['sh']
    args: ['-c','cp -r /usr/skywalking/agent/* /skywalking/agent']
    volumeMounts:
      - mountPath: /skywalking/agent
        name: skywalking-agent
containers:
  - name: <change-me> # EFK日志系统日志查询tag
    image: <IMAGE>:<IMAGE_TAG> # 镜像地址:版本
    imagePullPolicy: IfNotPresent
    volumeMounts:
      - mountPath: /usr/skywalking/agent # 挂载skywalking-agent到pod
        name: skywalking-agent
      - mountPath: /opt/logs # 挂载app-logs到node
        name: app-logs
    ports:
      - containerPort: 10080 # 服务暴露端口
    resources: # 服务所需资源申请
      requests:
        memory: "512Mi"
        cpu: "200m"
      limits:
        memory: "1Gi"
        cpu: "600m"
    env: # 环境变量
      - name: ENVIRONMENT
        value: "pro"
      - name: APP_OPTS
        value: "--spring.kafka.consumer.group-id=xxx.group"
      - name: JVM_OPTS
        value: "-Xms512m -Xmx512m -javaagent:/usr/skywalking/agent/skywalking-agent.jar"
    livenessProbe: # 存活探针
      httpGet:
        path: /actuator/health
        port: 10080
```

```
            initialDelaySeconds: 60
            periodSeconds: 10
            timeoutSeconds: 5
          readinessProbe: # 就绪探针
            httpGet:
              path: /actuator/health
              port: 10080
            initialDelaySeconds: 30
            periodSeconds: 10
            timeoutSeconds: 5
          lifecycle:
            preStop: # pod停止前Hook - 优雅停机
              exec:
                command:
                  - "curl"
                  - "-XPOST"
                  - "http://127.0.0.1:10080/actuator/shutdown"
      imagePullSecrets:
        - name: regcred
      volumes:
        - name: skywalking-agent
          emptyDir: {}
        - name: app-logs
          hostPath:
            path: /opt/app-logs/<change-me>
            type: DirectoryOrCreate
---
apiVersion: v1
kind: Service
metadata:
  name: <change-me>-service # 服务名称
  annotations:
    prometheus.io/path: /actuator/prometheus # 应用监控metrics路径 - 对应配置文件开启prometheus
    prometheus.io/port: "10080"
    prometheus.io/scrape: "true"
  labels:
    app: <change-me>
```

```yaml
spec:
  type: ClusterIP
  ports:
    - port: 10080
      targetPort: 10080
  selector:
    app: <change-me>
---
# 请根据实际情况，选择是否需要对外暴露Ingress
## 在微服务架构中，一般通过Gateway网关统一对外提供服务，微服务单独对外暴露Ingress情况请谨慎选择
apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
  name: <change-me>-ingress
  annotations:
    nginx.ingress.kubernetes.io/rewrite-target: /
    nginx.ingress.kubernetes.io/load-balance: "ip_hash" # session保持
    nginx.ingress.kubernetes.io/upstream-hash-by: "$request_uri" # 配合ip_hash使用
spec:
  rules:
    - host: <change-me>.meitianiot.lo # Ingress 域名
      http:
        paths:
          - path: /
            backend:
              serviceName: <change-me>-service
              servicePort: 10080
```

4、K8S发布

> 此过程已通过Jenkins Pipeline自动化CICD方式实现

1. **Git clone/pull代码**： `git clone git://gitea.meitianiot.lo/spring-produce.git`
2. **Junit单元测试**： `mvn test` # 普通程序员都不需要写单元测试?
3. **Maven编译打包**： `mvn clean package -Dmaven.test.skip=true`

4. **Docker打包镜像**： `docker build -t ${image}:${imageTag} .`
5. **Harbor镜像推送**： `docker push ${image}:${imageTag}`
6. **Sed渲染模板**：

```
sed -i "s|<CHANGE_CAUSE>|${changeCause}|g" manifests/k8s.yaml
sed -i "s|<IMAGE>|${image}|g" manifests/k8s.yaml
sed -i "s|<IMAGE_TAG>|${imageTag}|g" manifests/k8s.yaml
```

7. **Kubectl部署应用**： `kubectl --kubeconfig $kubeconfig apply -f manifests/k8s.yaml -n ${NAMESPACE}`
8. **Rollout快速回滚**： `kubectl --kubeconfig ${kubeconfig} rollout undo deployment consume-deployment -n ${NAMESPACE}`

> thanks for you!