محمد فراهانی ۴۰۳۷۲۱۰۵۶

ابتدا توضیحی در مورد فایل‌های موجود می‌دهیم:

* پوشه cluster:  
  این پوشه شامل خدمت بر پایه Django می‌باشد که خود یک برنامه راخل دارد که دو سرویس ارائه می‌دهد:
  + سرویس check: این سرویس شناسه‌ی پردازه برنامه را برمی‌گرداند.
  + سرویس instance-id: این سرویس یک شناسه یکتای تصادفی به همراه نامی که در محیط برنامه تایین شده است را بر می‌گرداند.
* فایل Dockerfile:  
  این فایل برای ساخت docker image استفاده می‌شود:
  + خط اول (FROM): از تصویر پایه ی پایتون با نسخه مشخص شده برای اجرا استفاده می‌کند.
  + خط دوم (WORKDIR): پوشه که container با آن کار می‌کند را مشخص می‌کند.
  + خط سوم (COPY requirement.txt): فایل نیازمندی‌های پروژه را در پوشه کاری کپی می‌کند.
  + خط چهارم (RUN pip): نیازمندی‌ها را بر اساس فایل requirement.txt نصب می‌کند.
  + خط پنجم (COPY .): فایل‌های برنامه را در پوشه کاری کپی می‌کند.
  + خط ششم (CMD): برنامه را اجرا می‌کند.
* فایل deployment.yaml:  
  این فایل برای استقرار سرویس بر روی podها است و محتویات آن به صورت زیر است:
  + apiVersion: نسخه استقرار را مشخص می‌کند.
  + kind: نوع فایل را مشخص می‌کند که در اینجا استقرار اشت.
  + metadata:  
      name: نام برای آن می‌گذارد.
  + spec:  
      replicas: تعداد پادها را مشخص می‌کند.
  + selector:  
        matchLabels:  
          app: نام پاده‌ّایی که از آن‌ها استفاده می‌کند را مشخص می‌کند.
  + template:  
        metadata:  
          labels:  
            app: نام پادها را مشخص می‌کند
  + spec:  
          containers:  
          - name: نام کانتیرنر‌ها را مشخص می‌کند
  + image: نام تصویر را مشخص می‌کند
  + ports:  
            - containerPort:پورت داخلی کانتینر را مشخص می‌کند.
* فایل service.yaml:  
  این فایل برای تنظیمات سرویس که پادها را مدیریت می‌کند استفاده می‌شود
  + apiVersion: نسخه را مشخص می‌کند.
  + kind: نوع فایل را مشخص می‌کند که در اینجا سرویس است.
  + metadata:  
     name: نام سرویس را تعریف می‌کند.
  + spec:  
     selector:  
     app: نام استقرا مورد استفاده را مشخص می‌کند.
  + ports:  
     - port: پورت سرویس را مشخص می‌کند.
  + targetPort: پورت دسترسی به پادها را مشخص می‌کند.
* فایل ingress.yaml:  
  این فایل برای مدیریت درخواست‌هایی هست که توسط nginx مدیریت می‌شود.
  + apiVersion: نسخه را مشخص می‌کند.
  + kind: نوع فایل را مشخص می‌کند.
  + metadata:  
     name: اسم را مشخص می‌کند.
  + annotations:  
     nginx.ingress.kubernetes.io/rewrite-target: جایگزین پسوند آدرس را مشخص می‌کند.
  + nginx.ingress.kubernetes.io/load-balance: الگوریتم تقسیم بار را مشخص می‌کند.
  + spec:  
     rules:  
     - host: آدرس سرویس را مشخص می‌کند.
  + http:  
     paths:  
     - path: آدرس مورد قوبل سرویس را مشخص می‌کند.
  + pathType: نوع تطابق آدرس را مشخص می‌کند.
  + backend:  
     service:   
     name: نام سرویس را تعیین می‌کند.
  + port:
  + number: پورت دسترسی به سرویس را مشخص می‌کند.

برای اجرا باید از دستورات زیر استفاده کرد:

* ساخت و آمده سازی تصویر Docker:

docker build -t mfary636/k8s-instance:0.0.2 .

docker push mfary636/k8s-instance:0.0.2

* آماده سازی minikube:

minikube start

minikube addons enable ingress

* اعمال فایل‌ّهای تنظیمات:

kubectl apply -f deployment.yaml

kubectl apply -f service.yaml

kubectl apply -f ingress.yaml

* تنظیمات آدرس: این بخش در سیستم عامل ویندوز دردسرساز است و باید آدرس minikube با توجه به فیلد ingress.yaml در فایل etc/hosts اضافه شود.

minikube ip

minikube tunnel

* افزایش تعداد پادها به ۳:

kubectl scale deployment web-service --replicas=3

* بررسی وضعیت پادها:

kubectl get pods

* تست: آدرس می تواند دردسر ساز باشد. پس با دستورات زیر می‌توان سیستم را از درون خوشه با حفظ آدرس‌های بیرونی تست کرد:

kubectl run testpod --image=busybox --restart=Never -- sleep 3600

kubectl exec -it testpod -- wget -qO- http://django.local/instance-id