Herkese merhaba,

Bu benim ilk açıklamalı GitHub serüvenim.

Eksiklikler veya yanlış anlatımımdan dolayı şimdiden kusura bakmayın.

Eklemek istediğiniz noktalar varsa ya da yanlışımı bana söylemek isterseniz seve seve dinlemek isterim;

<https://www.linkedin.com/in/mehmetyasti/>

Umarım faydalı olur. :)

Bugün sizlere Spring Boot ile yazılan bir HelloWorld web uygulamasının DockerFile ile nasıl build edilip dockerize edildikten sonra k8s ortamına nasıl deploy edileceğini anlatmaya çalışacağım.

Required Tools;

**-VS Code**

**-Java jdk**

**-Docker Desktop**

**-k8s (minikube)**

**-mvn**

**-DockerHub account**

Öncelikle ilgili repoyu localimize çekelim.

Bunun için ihtiyacımız olacak olan komut; **git clone**

Ben bazı işlemlerde WSL kullanıyorum (Windows için ayarlamalar bazen uzun geliyor :))

**git clone https://gitlab.com/bcfmkubilay/dummy-spring-boot.git**

Java dosyası build etmemiz için Java jdk'sına ihtiyacımız olacak onun için Ubuntu (WSL) komutumuz;

**sudo apt-get install default-jdk**

Yüklenip yüklenmediğini kontrol etmek için;

**java -version**

İlgili repoyu locale çektikten sonra build işlemi için "Maven" aracına ihtiyacımız olacak.

Windows için Maven kurulumunu aşağıdaki linkten takip edebilirsiniz;

https://www.javatpoint.com/how-to-install-maven

WSL için;

**sudo apt-get install maven**

Yüklenip yüklenmediğini kontrol etmek için;

**mvn –version**

İlgili klasör içerisinde src dosyasının olduğu kökte terminalimizi açıp maven ile çalışmaya başlıyoruz.

**mvn clean install -DskipTests**

Başarılı bir şekilde çalıştıktan sonra klasör içerisinde “**target**” adına bir klasör ve o klasör altında

“**jar**” dosyası oluşacaktır. Oluşan klasör ve ilgili dosya oluşması mvn ‘ ı başarılı bir şekilde gerçekleştirdiğimizi gösteriyor. (Bu işlemi WSL’de başarılı şekilde çalıştığını görünce WSL’den devam ettim 😊)

Build işleminden önce aynı source kodu indirip tag:blue ve tag:green olarak code içerisinde mapping kısmını ayarlarsanız ilerde yapacağımız tag bazlı deployment için faydalı olur.(Tabi benim gibi Java bilmiyorsanız 😊) Eğer koda içerisinde dinamik bir şekilde prefixi alıp mapping string kısmına ekleyebilirseniz buna hiç gerek kalmaz.

VS Code’ta projenin root directory’sinde bir Dockerfile oluşturup, onun içerisinde uygulamamızı containerize edeceğiz.

Build için ilgili Dockerfile komutları aşağıda belirtilmiştir.

FROM java:8-jdk-alpine

COPY ./target/helloworld-0.0.1-SNAPSHOT.jar /usr/app/

WORKDIR /usr/app

RUN sh -c 'touch helloworld-0.0.1-SNAPSHOT.jar.jar'

ARG JAR\_FILE=target/helloworld-0.0.1-SNAPSHOT.jar

ENTRYPOINT ["java","-jar","helloworld-0.0.1-SNAPSHOT.jar"]

**Dockerfile** dosyasını oluşturduktan sonra tekrardan dosyanın olduğu klasörde terminale geçip

Uygulamamız docker imajı haline getirmemiz gerekir ve imaj oluşturma komutumuz;

**docker build -t istediginizismiverebilirsiniz .**

(sondaki noktayı unutmamalısınız 😊 )

**docker images** diyerek oluşan image’ı local image repo’nuzda görebilirsiniz.

Oluşan imajın container olarak çalışıp çalışmadığını anlamak için deployment işlemlerinden önce single container oluşturup deneyebilirsiniz.

Şimdi bu imajı dockerhub hesabımıza gönderelim.

Local repomuzda bulunan imagı dockerhub hesabımıza göndermek için aşağıdaki linkten detaylı bilgi alabilirsiniz.

<https://www.section.io/engineering-education/docker-push-for-publishing-images-to-docker-hub/>

\*\*\*shell’den docker’a login olmayı unutmayınız.

**docker login**

Hub’a imajı attıktan sonra yapacağımız işlem, k8s ortamında deployment’lar oluşturarak uygulamamıza içerden ve daha sonrasında dışardan ulaşmak.

K8s ortamında node oluşturmak için minikube kullanıyorum.

**minikube start**

diyerek docker-driver üstünde bir tane master node üzerinde containerımız oluşuyor.

(Evet, k8s de aslında bir container olarak çalışıyor 😊 )

**kubectl get nodes -A**

dersek k8s’in ayakta olup olmadığını anlarız.

Uygulamamızın container halinde pod’larda çalışabilmesi için **yaml** dosyalarında ilgili tanımlamaları yapıp **kubectl** komutu ile çalıştırmamız gerekiyor.

k8s uygulamamıza çalıştıktan sonra localimizden ulaşmak için ihtiyacımız olan servisler şunlardır;

**deployment ve service**

minikube start dedikten sonra master node oluşur ve yaml dosyalarımızın olduğu klasör içerisinde VS Code çalıştırırız ve bu yaml dosyalarını çalıştırıp k8s’de deployment ve servislerimizin oluşmasını sağlarız.

Localimizden ulaşacağımız ya da herhangi bir bulut servis sağlayıcısından erişmek istediğimizi uygulamamız için deployment dosyamız aynıdır. Ancak çalıştıracağımız servis dosyamızın servis tipi özelliğinde bazı değişiklikler olacaktır.

Önce lokalimizden ulaşacağımız senaryo için hazırlanalım.

Deployment dosyamız aşağıdaki gibidir.

apiVersion: apps/v1

kind: Deployment

metadata:

  name: frontend

spec:

  replicas: 1

  selector:

    matchLabels:

      app: frontend

  template:

    metadata:

      labels:

        app: frontend

    spec:

      containers:

      - name: frontend

        image: memoliyasti/dummy-spring-boot:blue

        ports:

        - containerPort: 80

Servis dosyamız ise aşağıdaki gibidir.

apiVersion: v1

kind: Service

metadata:

  name: frontend

spec:

  type: NodePort

  selector:

    app: frontend

  ports:

    - protocol: TCP

      port: 80

      targetPort: 8080

**type: NodePort** olarak servis edilirse podlarımıza node’un dış bacağından (şimdilik cluster içinden) erişilebilmesi için port açıyor.

Ancak minikube buna direkt olarak izin vermediği için tünel açmak zorunda kalıyoruz.

Bu tünelin açılması için gerekli olan komut;

minikube service –url $oluşanservisinadı

komutun output’unda <http://127.0.0.1:$yazanportnumber>/blue adresine browser üzerinden giderseniz blue tagli uygulamanın çalıştığını görebilirsiniz.

Aynı şekilde aynı işlemleri green tagli uygulama için yaparak aynı sonuçları alabilirsiniz.

Bu şekilde localimizde oluşturduğumuz uygulamalara aynı şekilde localimizden de ulaşabildik.

AWS ortamı içinde yapacağımız işlemlerin çoğu aynı sadece birkaç authentication işlemi var AWS CLI ile. Onları da sırayla yapalım.

Powershell ya da Linux terminalinden oluşturmak istediğimiz AWS Cluster’ı için **eksctl** aracına ihtiyacımız var.

Aşağıdaki linkten nasıl yüklenebileceğine ulaşabilirsiniz.

<https://docs.aws.amazon.com/eks/latest/userguide/eksctl.html>

Yükleme ve login işlemlerini yaptıktan sonra powershell üzerinden cluster oluşturma adımına geçerlim.

**eksctl create cluster --name** *<my-cluster>* **--version** *<1.21>*

<https://docs.aws.amazon.com/eks/latest/userguide/create-cluster.html>

Bu işlem biraz zaman almaktadır. Bilginize.

AWS’deki cluster oluşunca ordaki contexti yani ortamı kontrol etmek için contex değiştirmemiz gerekiyor. Localimizde olan yaml dosyalarını o ortamda çalıştırmak için geçiş yapacağımız yerde çalışmamız gerekli.Bunun için gerekli komut;

**kubectl config use-context $aws’deki context’inizin adı.**

Context adınızı öğrenmek için gereken komut;

**kubectl config view** yazarak **name:** kısmının karşısında aws ile olan yeri seçip yukardaki komuta ekleyebilirsiniz.

Geçiş yaptıktan sonra tekrardan blue deployment için yaml dosyamızı çalıştırıyoruz.

Yalnız AWS ortamındaki uygulamamızı dış dünayaya açacağımız için bu sefer service tipini LoadBalancer olarak seçiyoruz.

Service yaml dosyasını aşağıda bulabilirsiniz.

apiVersion: v1

kind: Service

metadata:

  name: frontendlb

spec:

  type: LoadBalancer

  selector:

    app: frontend

  ports:

    - protocol: TCP

      port: 80

      targetPort: 8080

Oluşturulan tüm servislerin ayakta olup olmadığını anlamak için tekrardan **kubectl get all** diyoruz istenilen servislerin oluşup oluşmadığını kontrol ediyoruz.

Bu LoadBalancer tipi servis bize AWS’de uygulamamızı dış dünyaya açmak için bir load balancer oluşturuyor ve bize bir external ip veriyor.

Bunu görmek için gereken komut;

**kubectl get svc**

Oradaki uzantıyı alıp browserımıza /blue prefixi ile girersek, uygulamamızın dış dünyaya açıldığını da görmüş oluruz.

Benim anlatacaklarım şimdilik bu kadar.

Hatalar ve eksiklikler için tekrardan kusura bakmayın lütfen.

Gönül isterdi CI/CD süreci ile de bütünleşmiş bir proje gerçekleştireyim ama maalesef beceremedim.

İlerleyen zamanlarda Jenkins ile daha çok çalışıp onu da yapacağım.

Buraya kadar okuduysanız teşekkür ederim.

Herkese sağlıklı günler dilerim. 😊