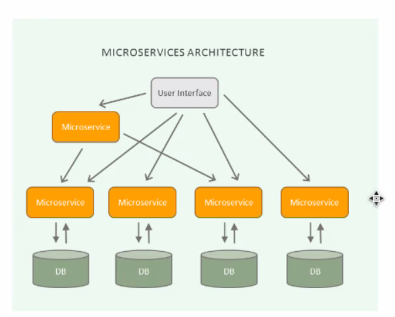
Bugünkü dersin ilk bölümünde microservis mimairisini görecez, kubernetes tekrarı gibi olacak.

Büyük bir projenin küçük parçalara bölünmüş halde olmasına microservis mimarisi denir.



Peki bu ayrı parçaları nasıl bütün halinde kullanacağız, aralarındaki bağlantı nasıl kurulacak.

Bugünkü örneğimizde db ve app kısmı var, ikisini nasıl bir arada kullanacağımızı göreceğiz.

mongo-db 
todoappl 
(node js) 

PVC 
ClusterlP 
port: 27017 
db-deployment 
(mongodb pod) 
targetPort:27017 
web-service 6 
portaooo 
web-deployment 
(nodejs pod) 
targetPort:3000 
web 

Bize interiewde şunu k8s ortamına yükleyin gibi taskler veriliyor, bunu iyi anlamamız lazım.

Önce pv ve pvc hazırlayıp sonra db deploymentını hazırlayacağız.

mkdir microservices 
cd microservices 
mkdir to-do 
cd to-do 

önce bu yapıyı oluşturuyoruz.

db-pv.yaml isimli pv manifest dosyamızı oluşturalım. gidip dokümantasyondan alıp değişiklikleri yapabiliriz:

apiVersion: VI 
kind: PersistentV01ume 
metadata : 
name: task-pv-volume 
labels : 
type: local 
spec : 
storageC1assName : 
capacity: 
storage: 18Gi 
accessModes : 
manual 
ReadWriteOnce 
host Path : 
path: "/mnt/data" 

daha sonra pvc için manifest dosyasını yazdık:

apiVersion: v1

kind: PersistentVolumeClaim

metadata:

name: database-persistent-volume-claim

spec:

accessModes:

- ReadWriteOnce

storageClassName: manual

resources:

requests:

storage: 1Gi

mongodb database için deployment dosyasını yazıyoruz:

apiVersion: apps/v1

kind: Deployment

metadata:

name: db-deployment

spec:

replicas: 1

selector:

matchLabels:

name: mongo

template:

metadata:

labels:

name: mongo

app: todoapp

spec:

containers:

- image: mongo:5.0

name: mongo

ports:

- containerPort: 27017

volumeMounts:

- name: mongo-storage

mountPath: /data/db

volumes:

#- name: mongo-storage

# hostPath:

# path: /home/ubuntu/pv-data

- name: mongo-storage

persistentVolumeClaim:

claimName: database-persistent-volume-claim

image olarak mongo:5.0 kullanıyoruz, latest ta çalışmıyor app, latest her zaman iyidir diye bir şey yok. mongo image i kullanıyorsak onu anlamamız gerekir. dockerhuba a gidiyoruz:

& dockerhub 
q mongo 
Explore 
Official Images 
mongo 
Overview 
Q DOCKER OFFICIAL IMAGE ± • 09.6K 
mongo 
MongoDB document databases provide high availability and easy scalability. 
Tags 

image in portunu da buradan alıyoruz o nedenle 27017 yazıyoruz.

Environment Variables 
When you start the mongo image, you can adjust the initialization of the MongoDB instance by passing one or more 
environment variables on the docker run command line. Do note that none of the variables below will have any 
effect if you start the container with a data directory that already contains a database: any pre-existing database will 
always be left untouched on container startup. 
These variables, used in conjunction, create a new user and set that user's password. This user is created in the 
admin authentication database and given the roleof root , which is a 'superuser' role. 
The following is an example of using these two variables to create a MongoD8 instance and then using the mongosh 
cli (use mongo with 4.x versions) to connect against the admin authentication database. 

ENV VAR da var anca zorunlu değiller, o nedenle kullanmıyoruz.

bizim db yi sabit bir volume e bağlamamız lazım, bunun için pv ve pvc mizi oluşturduk ancak mongo db nin hangi klasörüne volume bağlmamız gerektiğini de docs tan bakmamız lazım.

1. Create a data directory on a suitable volume on ycnrr host system, e.g. 
2. Start your mongo container like this: 
-name sone-mongo -v "y/oøn/datadlr 
S docker run 
/my/own/datadir 
-d mongo 
The -v /ny/own/datadir /data/db partofthe command mounts the /my/own/datadir directory from the 
underlying host system as 
inside the container. where MongoDB by default will write its data files. 
This image also defines a volume for /data/conflgdb for use with --configsvr (see docs.mongodb.com for 
more details). 
Creating database dumps 

volumeMounts:

- name: mongo-storage

mountPath: /data/db

bu kısımda mountPath te cont içinde nereye mount edeceğimizi yazıyoruz.

Volume un nereye bağlanacağını yazdık,

volumes:

- name: mongo-storage

persistentVolumeClaim:

claimName: database-persistent-volume-claim

bu kısımda da volumes altında volume u oluşturuyoruz ( aslında pvc yi poda bağlıyoruz)

şimdi db deployment, pv ve pvc mizi çalıştıralım.

• k apply -f 
deployment . apps/db-deployment created 
persistentvolume/db-pv-vol created 
persistentvolumeclaim/database-persistent-volume-claim created 

aşama aşama çalıştırıyoruz ki hata varsa şimdi ortaya çıksın.

apiVersion: v1

kind: Service

metadata:

name: db-service

labels:

name: mongo

app: todoapp

spec:

selector:

name: mongo

type: ClusterIP

ports:

- name: db

port: 27017

targetPort: 27017

service manisfestimizi yazdık.

en alttaki port service portu ve db portu ile aynı tutulur. daha önce farklı yazıyorduk aradaki farkı görelim diye ancak bu ikisi aynı tutulur.

pod un label i service ten fazla olabilirdi ancak service in daha fazla olamaz.

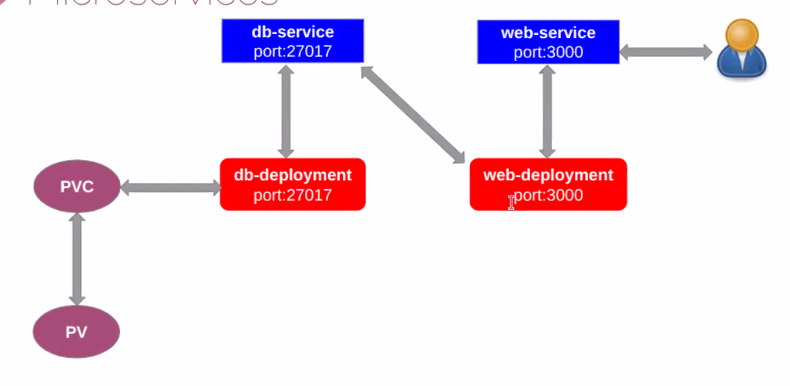
şimdi service e describe diyelim bağlanmış mı görelim:

• k describe svc db-service 
db-service 
Name : 
default 
Namespace: 
Labels : 
app=todoapp 
name—mongo 
Annotations : 
<none> 
Selector : 
name—mongo 
ClusterIP 
Type: 
IP Family Policy: 
SingleStack 
IP Families: 
IPv4 
10.98.180.99 
IPs: 
10.98.180.99 
db 27017/TCP 
Port : 
27017/TCP 
Target Port: 
Endpoints : 
10.244. 1.2: 27017 
Session Affinity: 
None 
Events : 
<none> 

endpoint adreslerini tutan bir obje vardı endpoint objesi, oradan da bakabiliriz ep ye:

• k get ep 
db-service 
kubernetes 
ENDPOINTS 
10.244. 1.2: 27017 
172.31.26.43:6443 
AGE 
2m8s 
48m 

mongo db yi oluşturduk şimdi to do app i hazırlayabiliriz.



web deploymetn ı kaldırdıktan sonra dn pod unu nasıl bulacak?

kendi kendine bulamaz conf yapmamız lazım.

bunu sağlamanın faklı yolları var ancak biz burada env ile yapacaz.

biz gidip app kodu içinde bulacağız.

O README.md 
JS db.js 
-06-microserv'ice-deployment-and-autos... > todo-app > app > JS db.js 
1 
2 
3 
5 
6 
7 
8 
9 
var 
var 
var 
var 
require( 'mongoose' ) ; 
mongoose 
Schema 
mongoose. Schema ; 
DBHost=process . env[" DBHOST"] I I "db" ; 
Todo 
user id 
content 
updated 
at 
Schema({ 
String, 
String, 
Date 

env DBHOST olanı bul demektir bu. bunu nşa da developerlar söyler, ancak interviewlerden sonra task verdiklerinde kendimizin bulmasını isteyebilirler.

spec:

containers:

- image: clarusway/todo

imagePullPolicy: Always

name: myweb

ports:

- containerPort: 3000

env:

- name: "DBHOST"

value: db-service

biz de burada env altında name olarak bunu yazıyoruz. value olarak da RDS kullansaydık endpointini yazacaktık. ancak burada servisten bulacak. service in ismini yazacaz.

DBHOST ta tırnak olmasa da oolur.

pod un ip sini yazsak pod çökene kadar bulur, ancak yeni pod gelince bulamaz.

web-deployment ın son hali:

apiVersion: apps/v1

kind: Deployment

metadata:

name: web-deployment

spec:

replicas: 1

selector:

matchLabels:

name: web

template:

metadata:

labels:

name: web

app: todoapp

spec:

containers:

- image: clarusway/todo

imagePullPolicy: Always

name: myweb

ports:

- containerPort: 3000

env:

- name: "DBHOST"

value: db-service

resources:

limits:

memory: 500Mi

cpu: 100m

requests:

memory: 250Mi

cpu: 80m

burada resources ı görüyoruz. İki kavramı var limit ve request. limit en fazla kullanacağı memory ve CPU. limit koymazsak pod şişer ve ec2 nun tüm kaynaklarını sömürür ve çökertir. bu nedenle limit verilir. request ise ;

Core Concepts 
kube controller 
manager 
e tcd 
cloud-controller 
manager 
kube•apł • server 
kubelet 
kubeehedulet 
kube•proxy 
Nedes 

kube scheduler ın görevi pod ları uygun node a yerleştirmekti, pod ların seviyesinin ne kadara ihtiyaç duyduğunu nereden bilecek, bu nedenle pod a request veriyoruz. yani requesti 2 Gi ise instance volume de 8 Gi ise en fazla 7 tane pod koyabilir o node a.

peki memory de koyduğumuz limiti pod aşmaya çalışırsa nolur OOMkill yani kill eder. restart eder.

CPU yu aşmaya çalışırsa nolur, throttle yani boğazlar. Orada tutar daha fazla kullanmasına müsaade etmez.

Burada namespace e göre de limitleyebiliriz. limit ve request değeri vermezsek default olarak ne olsun, bunlar için kuberneteste her biri için objeler var. bizim bununla ilgili 13. handsonumuz var.

bu handsonu kendmiz yapacaz, projeden sonra. dersi olmayacak.

bizim projeye bakalım:

11 
12 
13 
14 
15 
16 
18 
19 
Cr€late an object named app 
app 
configure mysql database 
app.config( = os.getenv( 
app.contig( ' os.getenv( 
app.contig[ ) 
app.config[ os.getenv( ) 
3306 
nysql = MysQL() 
mysql. Init _app(app) 
connection mysql.connect() 
connection. autocomit( True) 
cursor connection.cursor() 

database le ilgili env ler bu şekilde verilmiş. os.getenv işlerim sisteminin env lerinden çek demektir. kuberneteste ise os pod demektir. demek ki pod da bu değerleri tanımlamamız lazım.

MYSQL\_DATABASE değeri de default olarak phonebook tur.

MYSQL\_DATABASE\_HOST un db deployment ında olmasına gerek yok. bunn value si de service ismi olacak.

önce manuel olarak deployment altında env i yazıp sonra secret configmap yazın.

şimdi de app için web-service için oluşturuyoruz:

apiVersion: v1

kind: Service

metadata:

name: web-service

labels:

name: web

app: todoapp

spec:

selector:

name: web

type: NodePort

ports:

- name: http

port: 3000

targetPort: 3000

nodePort: 30001

protocol: TCP

şimdi apply edip localhost:30001 de görebiliriz.

minikube te çalışanlar için :

minikube service web-service

minikube service --url

ile minikbe un IP adresini alıp browserda görebiliriz.

O localhost:30001 
nent... Course: AWS 1 3-TU... 
Google Takvim 13.. 
O Your Repositories DeW3ps - Microsoft... (3) "devop 
Containers Todo Exarr 
proje bitecek 

Nodeport: 30001 
Service 
port:3000 
container 
Port: 
container 
Port! 
db-Service 
Port:27017 
db-container 
Port: 27017 

şimdi dersin ikinci kısmı horizontal pod autoscaling için yeni bir app oluşturacaz.

mkdir php- apache 
cd php-apache 

içinde php-apache.yaml

oluşturuyoruz.

php-apache kubernetes docs ten alınmış hpa dersi için kullanılacak hiçbir özelliği yok.

biz manifest dosyalarının arasına

---

koyarak aynı dosyada yazabiliriz objeleri.

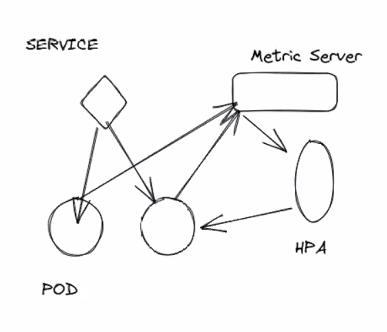
$ kubectl• apply• -f • 
deployment. apps/ php- apache 
created 
service/php-apache-service 
created 

bu sadece stres veren bir app. 30002 den app i görelim:

C O localhost:30002 
AWS Management... Course: AWS I 
OK! 

AWS de autoscaling nasıldı?

Kubernetes te nasıl?



Cloudwatch yerine Metric Server var, ASG yerine de HPA var.

Service de load balancing yapıyor.

HPA da bir objedir.

apiVersion: autoscaling/v2

kind: HorizontalPodAutoscaler

metadata:

name: php-apache

spec:

scaleTargetRef:

apiVersion: apps/v1

kind: Deployment

name: php-apache

minReplicas: 2

maxReplicas: 10

metrics:

- type: Resource

resource:

name: cpu

target:

type: Utilization

averageUtilization: 50

dameonset objesi her node da bir pod oluşturuyordu.

scaleTargetRef:

apiVersion: apps/v1

kind: Deployment

name: php-apache

şu kısım deployment ı belirlemek için yazılan kısımdır.

min ve max replicas ı belirtiyoruz.

resource olarak da cpu, memory ya da custom seçebiliriz. type utilization yazılınca girdiğimiz değer % olur.

apply ediyoruz.

• k get po 
db-deployment-76c458dd84-srxzk 
php-apache-667b4cf5bf-9k4nb 
web-deployment -7b74bfcf47-q2vjs 
READY 
1/1 
1/1 
1/1 
STATUS 
RESTARTS 
Running 
Running 
Running 
• k get po 
READY 
1/1 
1/1 
1/1 
1/1 
STATUS 
Running 
Running 
Running 
Running 
RESTARTS 
db-deployment -76c458dd84-srxzk 
php-apache-667b4cf5bf-99b44 
php-apache-667b4cf5bf-9k4nb 
web-deployment -7b74bfcf47-q2vjs 
AGE 
31m 
AGE 
95m 
16s 
31m 
55m 

pod sayısı sürekli artıyor.

şimdi de web için hpa manifest ini yazıyoruz.

apiVersion: autoscaling/v2

kind: HorizontalPodAutoscaler

metadata:

name: web-deployment

spec:

scaleTargetRef:

apiVersion: apps/v1

kind: Deployment

name: web-deployment

minReplicas: 3

maxReplicas: 5

metrics:

- type: Resource

resource:

name: cpu

target:

type: Utilization

averageUtilization: 50

onun da sayısı artıyor:

get po 
TATUS 
RESTARTS AGE 
db-deployment-76c458dd84-srxzk 
unning 
php-apache-667b4cf5bf-99b44 
READY 
1/1 
1/1 
1/1 
1/1 
1/1 
1/1 
s. 
unning 
4m43s 
php-apache-667b4cf5bf-9k4nb 
unning 
36m 
web-deployment- 7b 74bfcf47-d95c6 
unning 
web -deployment- 7b 74bfcf47-fvz6s 
unning 
web-deployment-7b74bfcf47-q2vjs 
unning 
59m 
ubuntu@kube-master : N/ microservices / autocali 

How to work Horizontal Pod Autoscaling (hpå 
3. Collect & 
Aggregate 
Metrics 
Metric Server 
6. Scale Replica 
8. Scale 
Replica 
K UBE LET 
2. Expose 
Metrics 
CAdvisor 
1. Collect 
Metrics 
DeploymentJReplica Set 
Horizontal Pod 
Autoscaler 
R9plica 
Coritroller 
Pod 
Pod 

metric server eksik, onu ekleyecez.

<https://github.com/kubernetes-sigs/metrics-server/releases/tag/v0.6.3>

wget <https://github.com/kubernetes-sigs/metrics-server/releases/download/v0.6.3/components.yaml>

komutuyla metric server ın components.yaml ını indirdik.

1. satıra
2. 135 
   136 
   137 
   138 
   139 
   lae 
   141 
   142 
   -- cert-dir=/tmp 
   - - -secure-port=4443 
   -kubelet-insecure -tlsl 
   - kubelet- preferred - address -types—Interna11P , External IP , Hostname 
   - kubelet- use- node -status - port 
   -metric-resolution=15s 
   image: registry . k8s . io/metrics-server/metrics-server :vø.6.3 
   imagePu11P01icy: IfNotPresent 

- --kubelet-insecure-tls

ekleme yaptık.

şimdi components.yaml dosyasını apply edelim ve bekleyelim.

kube-system namespace indeki podlara bakalım metric server running olmuş mu?:

-n kube-system get pods 
coredns -787d4945fb-b4tdb 
coredns -787d4945fb-qg66w 
etcd-kube-master 
kube-apiserver-kube-master 
kube-controller-manager-kube-master 
kube-proxy -ftm7v 
kube-proxy-rfpft 
kube-scheduler-kube-master 
metrics-server-7d4dc74cd9-szgfd 
READY 
1/1 
1/1 
1/1 
1/1 
1/1 
1/1 
1/1 
1/1 
1/1 
kubectl 
STATUS 
Running 
Running 
Running 
Running 
Running 
Running 
Running 
Running 
Running 
RESTARTS 
AGE 
143m 
143m 
144m 
144m 
144m 
143m 
143m 
144m 
93s 

algoritmasının detayları docs ta mevcut:

<https://kubernetes.io/docs/tasks/run-application/horizontal-pod-autoscale/>

• k get 
php-apache 
REFERENCE 
Deployment/php-apache 
TARGETS 
web-deployment Deployment/web-deployment 
hpa 
MINPODS 
2 
3 
MAXPODS 
10 
5 
REPLICAS 
2 
3 
AGE 
14m 
11m 

şu an target ın kullanım oranı %1 biz %50 yi aşarsa arttırmasını istedik.

$ kubectl top pods 
NAME 
db-deployment -8597967796 - 81wzr 
php -apache- 7869bd4fb -4q6mz 
php -apache-7869bd4fb -wmhtl 
web -deployment-6d8d8c777b-2fr9h 
web -deployment-6d8d8c777b-z5xd2 
CPU(cores) 
6m 
Im 
Im 
Im 
Im 
MEMORY(bytes) 
140Mi 
11Mi 
IOMi 
22Mi 
24Mi 

kubectl top pods komutu ile podların ne kadar kaynak tükettiğini görebiliriz.

$ kubectl top nodes 
NAME 
master 
nodel 
CPU (cores) 
188m 
108m 
CPU% 
9% 
5% 
MEMORY(bytes) 
1245Mi 
1035Mi 
MEMORY% 
32% 
27% 

k get nodes da node lar için bunu gösterir.

şimdi busybox image ından bir pod oluşturup podlara giden yük artıracaz.

What is BusyBox? The Swiss Army Knife of Embedded 
Linux 
Coming in somewhere between 1 and S Mb in on-disk size (depending on the variant), BusyBox is a very good 
ingredient to craft space-efficient distributions. 
BusyBox combines tiny versions of many common UNIX utilities into a single small executable. It provides 
replacements for most of the utilities you usually find in GNU fileutils, shellutils, etc. The utilities in BusyBox generally 
have fewer options than their full•featured GNU cousins; however. the options that are included provide the expected 
functionality and behave very much like their GNU counterparts. BusyBox provides a fairly complete environment for 
any small or embedded system. 
wikipedia.org/wiki/BusyBox 
How to use this image 
Run BusyBox shell 
S docker run -it 
-•rm busybox 

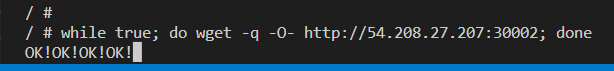
busybox sadece temel komutların olduğu bir image. komutu girelim:

kubectl run -it --rm load-generator --image=busybox /bin/sh

pod içinde

while true; do wget -q -O- [http://<puplic](about:blank) ip>:<port number of php-apache-service>; done

komutunu giriyoruz.



sürekli siteyi çağırıyor.

• ing$ kubectl 
TARGETS 
AGE 
php-apache 
65%/ 
20m 
get hpa 
REFERENCE 
MINPODS MAXPODS REPLICAS 
Deployment/php-apache 
2 
10 
2 
web-deployment 
17m 
Deployment/ web-deployment 
5 
3 
ubuntu@kube-master : N/ microservices / autoca 
o ling$ 

şu anda pod un kullanımı %62.

node:

ubuntu@kube-master : N/ microservices / autoca 
• ling$ kubectl top node 
CPU% MEMORY(b 
ytes) 
CPU(cores) 
MEMORY% 
kube-master 
57% 
kube-worker 
38% 
156m 
194m 
7% 
2167Mi 
IU5Mi 
ubuntu@kube-master : N/ microservices / autocal 

wweb-deployment için auto scaling yapalım

# while true; do wget -q -O- [http://54.208.27.207:30001](http://54.208.27.207:30001/) > /dev/null; done

bu komuttaki dev/null gelen bilgi çok fazla olacağı için linux un dışına atmak gibi bir işlevi var.

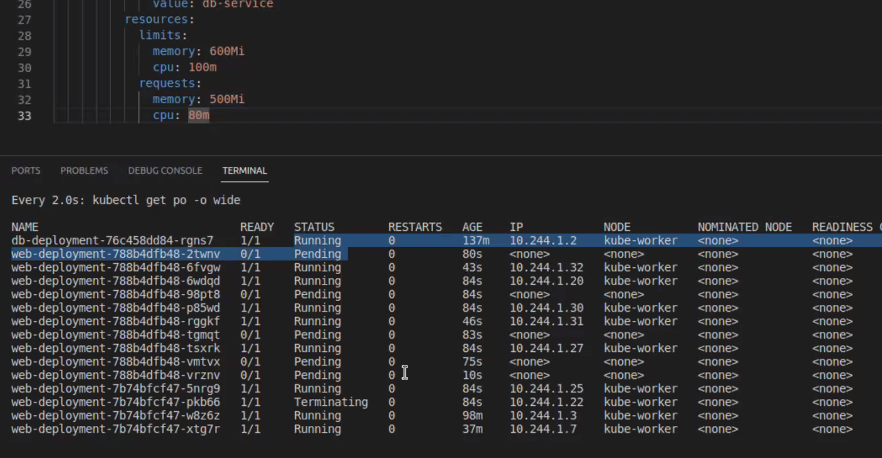
while true; do wget -q -O- [http://34.239.247.77:30001](http://34.239.247.77:30001/); done komutu ile stresi arttırdık ve podların arttığını gördük. 

Every 2. es: kubectl get hpa 
TARGETS 
MINPODS 
YAXPOOS 
10 
REPLICAS 
kube-mastt 
AGE 
php - apache 
web - dept oyænt 
REFERENCE 
Dep php apache 
Dep t / - dep oymen t 

yükü kesersek arttığı kadar hızlı düşmüyor ancak düşüyor.

metric server kaldırıp gözlem yaptık onun ayrıca dersi olacak.

Service te podları takip ediyordu, HPA da deployment ları takip ediyor.



resource ta limit leri görelim; bazı podlar pending yazıyor çünkü yer yok.