

Uygulamaları Ölçeklendirme ve Güncelleme



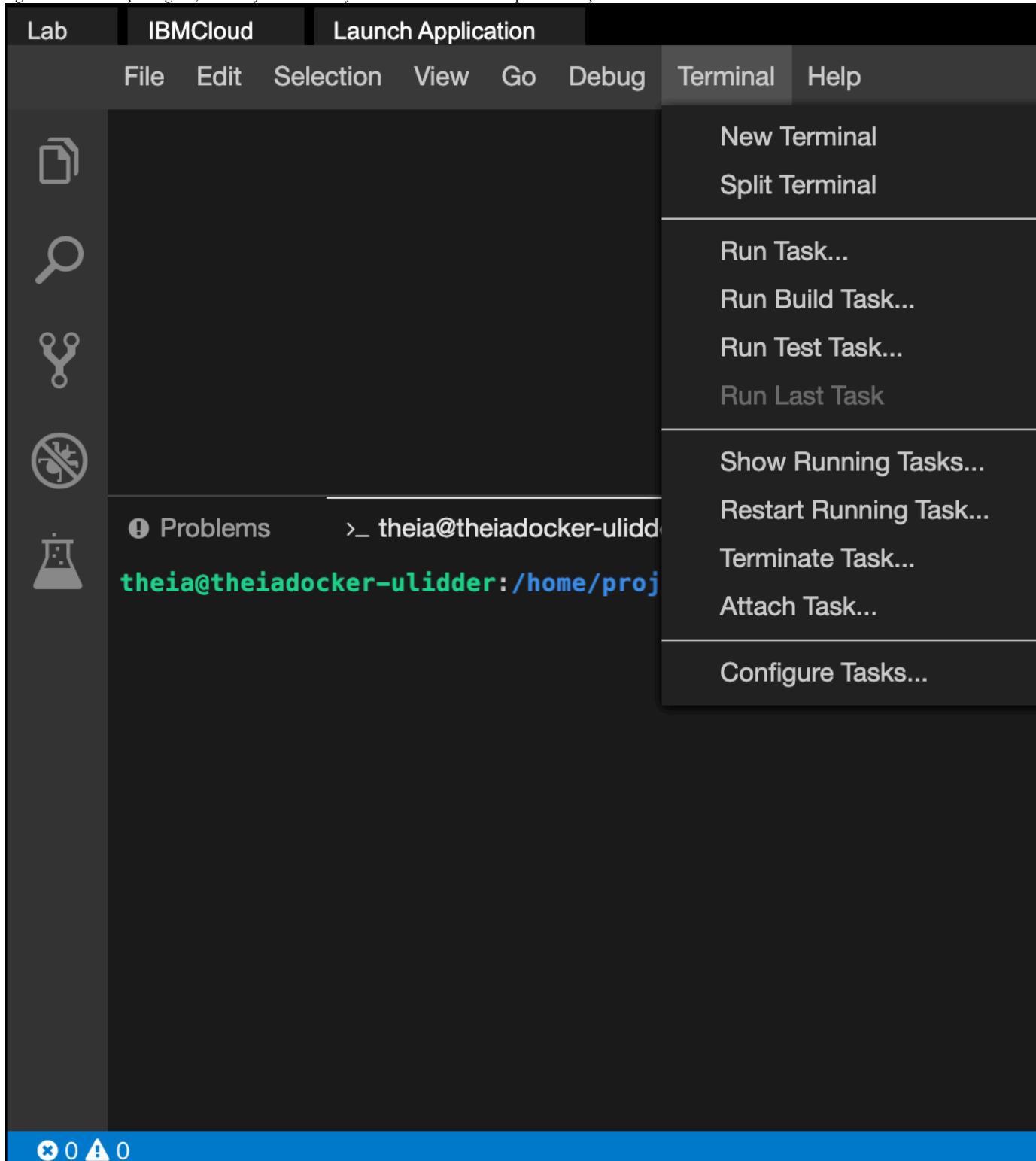
Hedefler

Bu laboratuvar çalışmasında şunları yapacaksınız:

- Bir ReplicaSet ile bir uygulamayı ölçeklendirin
- Bir uygulamaya kademeli güncellemler uygulayın
- Uygulama yapılandırmasını saklamak için bir ConfigMap kullanın
- Uygulamayı Yatay Pod Ölçekleyici kullanarak otomatik ölçeklendirin

Ortamı ve komut satırı araçlarını doğrulayın

1. Eğer bir terminal açık değilse, düzenleyicideki menüyü kullanarak bir terminal penceresi açın: Terminal > New Terminal.



NOT: Terminal İstemi'nin görünmesi biraz zaman alabilir. Eğer 5 dakika geçmesine rağmen terminal istemini göremiyorsanız, lütfen tarayıcı sekmesini kapatın ve laboratuvarı tekrar başlatın.

2. Proje klasörünüze geçin.

NOT: Eğer zaten /home/project dizimindeyseniz, bu adımı atlayabilirsiniz.

```
cd /home/project
```

3. Bu laboratuvar için gerekli olan varlıkları içeren git deposunu klonlayın, eğer zaten mevcut değilse.

```
[ ! -d 'CC201' ] && https://github.com/ibm-developer-skills-network/abahi-CC_201_labs.git CC201
```

```
theia@theiadocker- [ ~ :/home/project$ [ ! -d 'CC201' ] && git clone https://github.com/ibm-developer-skills-network/CC201.git
Cloning into 'CC201'...
remote: Enumerating objects: 20, done.
remote: Counting objects: 100% (20/20), done.
remote: Compressing objects: 100% (13/13), done.
remote: Total 20 (delta 6), reused 19 (delta 6), pack-reused 0
Unpacking objects: 100% (20/20), done.
```

4. Bu laboratuvar için dizine geçin.

```
cd CC201/labs/3_K8sScaleAndUpdate/
```

```
theia@theiadocker- [ ~ :/home/project$ cd CC201/labs/3_K8sScaleAndUpdate/
theia@theiadocker- [ ~ :/home/project/CC201/labs/3_K8sScaleAndUpdate$
```

5. Bu dizinin içeriğini listeleyerek bu laboratuvar için belgeleri görün.

```
ls
```

```
theia@theiadocker- [ ~ :/home/project/CC201/labs/3_K8sScaleAndUpdate$ ls
app.js deployment-configmap-env-var.yaml deployment.yaml Dockerfile package.json
theia@theiadocker- [ ~ :/home/project/CC201/labs/3_K8sScaleAndUpdate$
```

IBM Cloud Container Registry'ye uygulama görüntüsünü oluşturma ve yükleme

1. İlerleyen komutlarda kullanılabilmesi için ad alanınızı bir ortam değişkeni olarak dışa aktarın.

```
export MY_NAMESPACE=sn-labs-$USERNAME
```

```
theia@theiadocker- [ ~ :/home/project/CC201/labs/3_K8sScaleAndUpdate$ export MY_NAMESPACE=sn-labs-$USERNA
theia@theiadocker- [ ~ :/home/project/CC201/labs/3_K8sScaleAndUpdate$
```

2. Bir görüntü oluşturmak için kullanılacak Dockerfile'ı görüntülemek üzere Explorer'ı kullanın.

The screenshot shows a code editor interface with a dark theme. The top menu bar includes File, Edit, Selection, View, Go, Run, Terminal, and Help. On the left is a sidebar with various icons for file operations like Open Editors, Project, Search, and others. The main area has tabs for Welcome and Dockerfile M X. The Dockerfile tab is active, displaying the following code:

```
FROM node:9.4.0-alpine
COPY app.js .
COPY package.json .
RUN npm install &&
    apk update &&
    apk upgrade
EXPOSE 8080
CMD node app.js
USER 9000:9000
```

The project structure on the left shows a folder named CC201 containing a .theia folder with settings.json, a CC201 folder with labs (1_ContainersAndDocker, 2_IntroKubernetes, 3_K8sScaleAndUpdate), and a README.md file. The 3_K8sScaleAndUpdate folder contains app.js, deployment-configmap..., deployment.yaml, Dockerfile (which is selected and highlighted with a red border), package.json, .gitignore, LICENSE, and README.md.

3. Görüntüyü yeniden oluşturun ve gönderin, çünkü ilk laboratuvarı tamamladıktan sonra otomatik olarak silinmiş olabilir.

```
docker build -t us.icr.io/$MY_NAMESPACE/hello-world:1 . && docker push us.icr.io/$MY_NAMESPACE/hello-world:1
```

```

theia@theiadocker- [ ] :/home/project/CC201/labs/3_K8sScaleAndUpdate$ docker build -t us.icr.io/$MY_NAMESPACE/hello-
Sending build context to Docker daemon 6.144kB
Step 1/6 : FROM node:9.4.0-alpine
9.4.0-alpine: Pulling from library/node
605ce1bd3f31: Pull complete
fe58b30348fe: Pull complete
46ef8987ccbd: Pull complete
Digest: sha256:9cd67a00ed111285460a83847720132204185e9321ec35dacec0d8b9bf674adf
Status: Downloaded newer image for node:9.4.0-alpine
--> b5f94997f35f
Step 2/6 : COPY app.js .
--> 2f029424b7dc
Step 3/6 : COPY package.json .
--> d4f6f041bcfa
Step 4/6 : RUN npm install && apk update && apk upgrade
--> Running in eb1b0f41cbd7
npm notice created a lockfile as package-lock.json. You should commit this file.
npm WARN hello-world-armada@0.0.1 No repository field.
npm WARN hello-world-armada@0.0.1 No license field.

added 50 packages in 1.708s
fetch http://dl-cdn.alpinelinux.org/alpine/v3.6/main/x86_64/APKINDEX.tar.gz
fetch http://dl-cdn.alpinelinux.org/alpine/v3.6/community/x86_64/APKINDEX.tar.gz
v3.6.5-44-gda55e27396 [http://dl-cdn.alpinelinux.org/alpine/v3.6/main]
v3.6.5-34-gf0ba0b43d5 [http://dl-cdn.alpinelinux.org/alpine/v3.6/community]
OK: 8448 distinct packages available
Upgrading critical system libraries and apk-tools:
(1/1) Upgrading apk-tools (2.7.5-r0 -> 2.7.6-r0)
Executing busybox-1.26.2-r9.trigger
Continuing the upgrade transaction with new apk-tools:
(1/7) Upgrading musl (1.1.16-r14 -> 1.1.16-r15)
(2/7) Upgrading busybox (1.26.2-r9 -> 1.26.2-r11)
Executing busybox-1.26.2-r11.post-upgrade
(3/7) Upgrading libressl2.5-libcrypto (2.5.5-r0 -> 2.5.5-r2)
(4/7) Upgrading libressl2.5-libssl (2.5.5-r0 -> 2.5.5-r2)
(5/7) Installing libressl2.5-libtls (2.5.5-r2)
(6/7) Installing ssl_client (1.26.2-r11)
(7/7) Upgrading musl-utils (1.1.16-r14 -> 1.1.16-r15)
Executing busybox-1.26.2-r11.trigger
OK: 5 MiB in 15 packages
Removing intermediate container eb1b0f41cbd7
--> 8064e924ec74
Step 5/6 : EXPOSE 8080
--> Running in 06b2f40f50c1
Removing intermediate container 06b2f40f50c1
--> 74d97beb1311
Step 6/6 : CMD node app.js
--> Running in 8388f224b326
Removing intermediate container 8388f224b326
--> ca395ff2f872
Successfully built ca395ff2f872
Successfully tagged us.icr.io/sn-labs-[REDACTED]/hello-world:1
The push refers to repository [us.icr.io/sn-labs-[REDACTED]/hello-world]
fc8314e02b47: Pushed
2e7bcf63d006: Pushed
609d2e4acf9: Pushed
0804854a4553: Pushed
6bd4a62f5178: Pushed
9dfa40a0da3b: Pushed
1: digest: sha256:adb28bb0d3e133d2eb3563430dc41a7a35eb816331430bb601c6a5375fe351b size: 1576
theia@theiadocker- [ ] :/home/project/CC201/labs/3_K8sScaleAndUpdate$ █

```

NOT: Daha önce bu laboratuvarı denediyorsanız, önceki oturumun hala devam etme olasılığı olabilir. Bu durumda, yukarıdaki çıktıda ‘Pushed’ mesajı yerine ‘Layer already Exists’ mesajını göreceksiniz. Laboratuvarın sonraki adımlarına devam etmenizi öneririz.

Uygulamayı Kubernetes'e Dağıtma

1. Bu dizindeki deployment.yaml dosyasını düzenlemek için Explorer'ı kullanın. Bu dosyanın yolu cc201/labs/3_K8sScaleAndUpdate/’dir. <my_namespace> yazan yere kendi ad alanınızı eklemeniz gerekiyor. İşlemi tamamladıktan sonra dosyayı kaydetmeyi unutmayın.

NOT: Ad alanınızı öğrenmek için terminalde echo \$MY_NAMESPACE komutunu çalıştırın.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: hello-world
spec:
  selector:
    matchLabels:
      run: hello-world
  template:
    metadata:
      labels:
        run: hello-world
    spec:
      containers:
        - name: hello-world
          image: us.icr.io/sn-labs/hello-world:1
          imagePullPolicy: Always
          ports:
            - containerPort: 8080
          resources:
            limits:
              cpu: 2m
              memory: 30Mi
            requests:
              cpu: 1m
              memory: 10Mi
      securityContext:
        allowPrivilegeEscalation: false
        runAsNonRoot: true
        capabilities:
          drop: ["ALL"]
        seccompProfile:
          type: "RuntimeDefault"
        runAsUser: 999
      imagePullSecrets:
        - name: icr
```

2. Görüntünüzü bir Dağıtım (Deployment) olarak çalıştırın.

```
kubectl apply -f deployment.yaml
```

```
theia@theiadocker-:~/home/project/CC201/labs/3_K8sScaleAndUpdate$ kubectl apply -f deployment.yaml
deployment.apps/hello-world created
```

NOT: Eğer bu laboratuvarı daha önce denediyorsanız, önceki oturumun hala devam etme ihtimali olabilir. Bu durumda, yukarıdaki çıktıda ‘**Oluşturuldu**’ mesajı yerine ‘**Değişmedi**’ mesajını göreceksiniz. Laboratuvarın sonraki adımlarına devam etmenizi öneririz.

3. Durum ‘Çalışıyor’ olana kadar Pod’ları listeleyin.

```
kubectl get pods
```

```
theia@theiadocker-:~/home/project/CC201/labs/3_K8sScaleAndUpdate$ kubectl get pods
NAME           READY   STATUS    RESTARTS   AGE
hello-world-58985bb9fb-7nnqr   1/1     Running   0          4m52s
```

NOT: Lütfen pod durumunu ‘Çalışıyor’ olarak gördükten sonra bir sonraki adıma geçin. Eğer çıktıda ‘**Konteyner Oluşturuluyor**’ ifadesini görürorsanız, lütfen birkaç dakika sonra komutu tekrar çalıştırın.

4. Uygulamaya erişebilmek için, onu bir Kubernetes Servisi aracılığıyla internete açmamız gerekiyor.

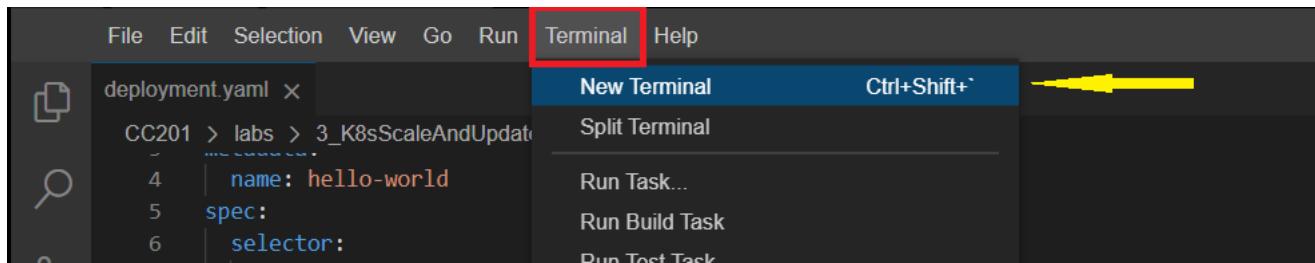
```
kubectl expose deployment/hello-world
```

Bu, ClusterIP türünde bir hizmet oluşturur.

```
theia@theiadocker-: /home/project/CC201/labs/3_K8sScaleAndUpdate$ kubectl expose deployment/hello-world
service/hello-world exposed
theia@theiadocker-: /home/project/CC201/labs/3_K8sScaleAndUpdate$
```

5. Terminal > New Terminal kullanarak yeni bir terminal penceresi açın.

NOT: Üzerinde çalışığınız terminal penceresini kapatmayın.



6. Cluster IP'ler yalnızca cluster içinde erişilebilir. Bunu dışarıdan erişilebilir hale getirmek için bir proxy oluşturacağız.

Not: Bu, bir uygulamayı üretim senaryosunda dışarıdan erişilebilir hale getirmenin yolu değildir.

Orijinal penceredeki ortam değişkenlerinin sonraki komutlar için erişilebilir olması gerekiğinden, bu komutu yeni terminal penceresinde çalıştırın.

```
kubectl proxy
```

```
theia@theiadocker-: /home/project/CC201/labs/3_K8sScaleAndUpdate theia@theiadocker-: /home/project X
theia@theiadocker-: /home/project$ kubectl proxy
Starting to serve on 127.0.0.1:8001
```

Bu komut çıkışa kadar çalışmaya devam edecek. Uygulamanıza erişmeye devam edebilmek için çalışır durumda tutun.

7. Orijinal terminal penceresine geri dönün, uygulamayı yanıt almak için ping'leyin.

NOT: proxy komutunun hala çalıştığı terminal penceresini kapatmayın.

```
curl -L localhost:8001/api/v1/namespaces/sn-labs-$USERNAME/services/hello-world/proxy
```

Mesajı gözlemleyin: "Merhaba dünya, hello-world-xxxxxxxx-xxxx'dan merhaba. Uygulamanız çalışıyor!"

```
theia@theiadocker-: /home/project/CC201/labs/3_K8sScaleAndUpdate theia@theiadocker-: /home/project
theia@theiadocker-: /home/project/CC201/labs/3_K8sScaleAndUpdate$ curl -L localhost:8001/api/v1/namespaces/sn-labs-$USERNAME/services/hello-world/proxy
Hello world from hello-world-58985bb9fb-7nnqr! Your app is up and running!
theia@theiadocker-: /home/project/CC201/labs/3_K8sScaleAndUpdate$
```

Uygulamayı ReplicaSet ile Ölçeklendirme

Gerçek dünya durumlarında, bir uygulamadaki yük zamanla değişebilir. Uygulamamız artan bir yükle karşılaşmaya başlarsa, bu yükü karşılamak için ölçeklendirmek isteriz. Ölçeklendirme için basit bir kubectl komutu vardır.

- Deployment’ınızı ölçeklendirmek için scale komutunu kullanın. Bunu proxy komutunun çalışmadığı bir terminal penceresinde çalıştırığınızdan emin olun.

```
kubectl scale deployment hello-world --replicas=3
```

```
theia@theiadocker-: /home/project/CC201/labs/3_K8sScaleAndUpdate$ kubectl scale deployment hello-world --replicas=3
```

- Pod’ları kontrol edin, böylece artık sadece bir tane yerine üç Pod olduğunu doğrulayın. Ayrıca, durumun sonunda üçü için de “Running” olarak güncellenmesi gereklidir.

```
kubectl get pods
```

```
theia@theiadocker-: /home/project/CC201/labs/3_K8sScaleAndUpdate$ kubectl get pods
NAME           READY   STATUS    RESTARTS   AGE
hello-world-58985bb9fb-7nnqr   1/1     Running   0          22m
hello-world-58985bb9fb-j9qkv   0/1     ContainerCreating   0          6s
hello-world-58985bb9fb-wg7nh   0/1     ContainerCreating   0          5s
```

- Son laboratuvarınızda yaptığınız gibi, Kubernetes’ın replikalar arasında yük dengelemesi yaptığından emin olmak için uygulamanızı birden fazla kez pingleyin.

```
for i in `seq 10`; do curl -L localhost:8001/api/v1/namespaces/sn-labs-$USERNAME/services/hello-world/proxy; done
```

```
theia@theiadocker-: /home/project/CC201/labs/3_K8sScaleAndUpdate$ for i in `seq 10`; do curl -L localhost:8001/api/v1/namespaces/sn-labs-$USERNAME/services/hello-world/proxy; done
Hello world from hello-world-58985bb9fb-7nnqr! Your app is up and running!
Hello world from hello-world-58985bb9fb-wg7nh! Your app is up and running!
Hello world from hello-world-58985bb9fb-wg7nh! Your app is up and running!
Hello world from hello-world-58985bb9fb-wg7nh! Your app is up and running!
Hello world from hello-world-58985bb9fb-7nnqr! Your app is up and running!
Hello world from hello-world-58985bb9fb-j9qkv! Your app is up and running!
Hello world from hello-world-58985bb9fb-wg7nh! Your app is up and running!
Hello world from hello-world-58985bb9fb-wg7nh! Your app is up and running!
Hello world from hello-world-58985bb9fb-7nnqr! Your app is up and running!
Hello world from hello-world-58985bb9fb-7nnqr! Your app is up and running!
```

Yük dengelemenin etkisi nedeniyle sorguların farklı Pod’lara gittiğini göremelisiniz.

- Benzer şekilde, Deployment’ınızı küçültmek için scale komutunu kullanabilirsiniz.

```
kubectl scale deployment hello-world --replicas=1
```

```
theia@theiadocker- [ ] :/home/project/CC201/labs/3_K8sScaleAndUpdate$ kubectl scale deployment hello-world deployment.apps/hello-world scaled
```

5. İki Pod'un silindiğini veya silinmeyeceğini kontrol edin.

```
kubectl get pods
```

```
theia@theiadocker- [ ] :/home/project/CC201/labs/3_K8sScaleAndUpdate$ kubectl get pods
NAME           READY   STATUS    RESTARTS   AGE
hello-world-58985bb9fb-7nnqr  1/1     Running   0          23m
hello-world-58985bb9fb-j9qkv  1/1     Terminating   0          44s
hello-world-58985bb9fb-wg7nh  1/1     Terminating   0          43s
theia@theiadocker- [ ] :/home/project/CC201/labs/3_K8sScaleAndUpdate$
```

6. Lütfen bir süre bekleyin ve yalnızca bir pod'un bulunduğundan emin olmak için aynı komutu tekrar çalıştırın.

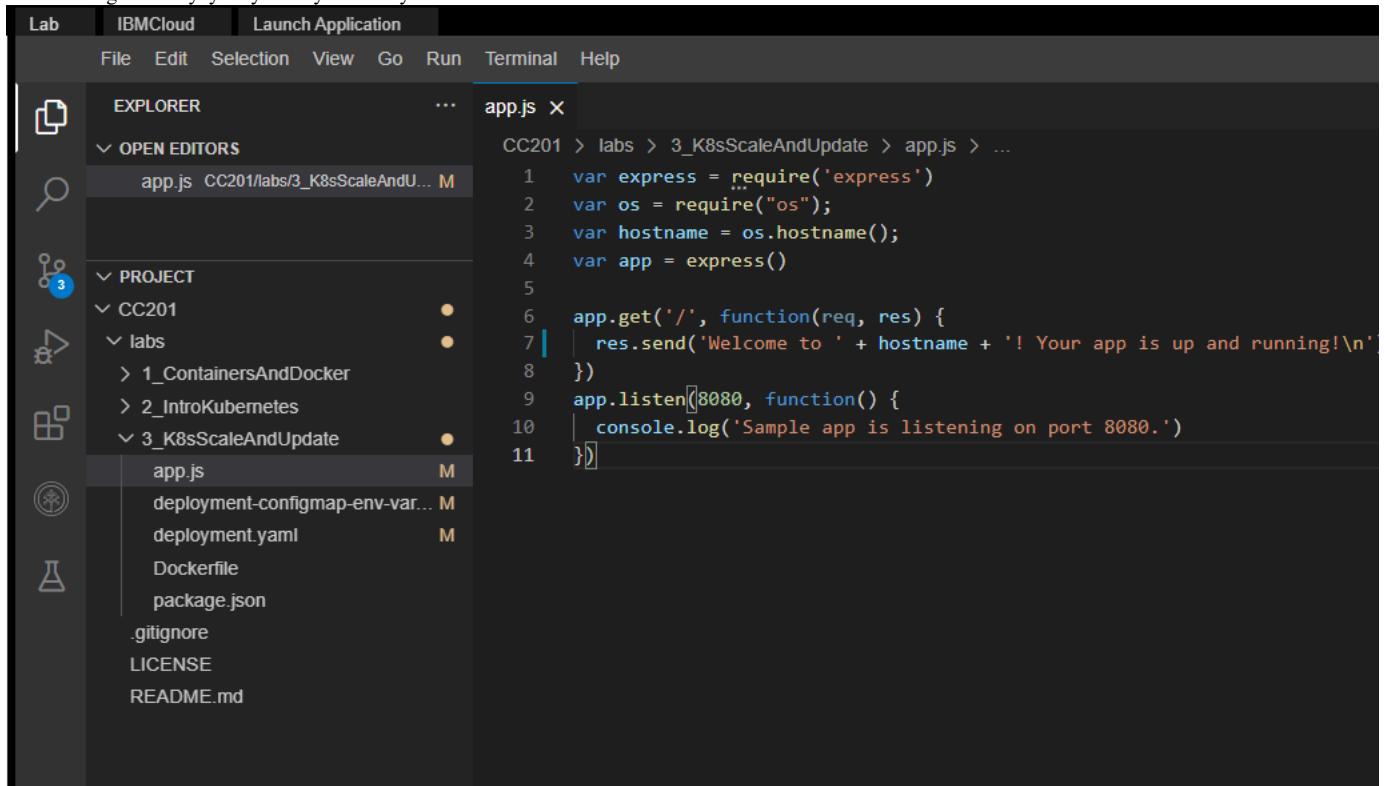
```
kubectl get pods
```

```
theia@theiadocker- [ ] :/home/project/CC201/labs/3_K8sScaleAndUpdate$ kubectl get pods
NAME           READY   STATUS    RESTARTS   AGE
hello-world-79c5684b95-5x4wp  1/1     Running   0          102s
theia@theiadocker- [ ] :/home/project/CC201/labs/3_K8sScaleAndUpdate$
```

Sürekli Güncellemeleri Gerçekleştirme

Sürekli güncellemeler, uygulamamızı otomatik ve kontrollü bir şekilde güncellemenin kolay bir yoludur. Bir güncellemeyi simüle etmek için, önce uygulamamızın yeni bir sürümünü oluşturalım ve bunu Container Registry'ye gönderelim.

1. app.js dosyasını düzenlemek için Explorer'i kullanın. Bu dosyanın yolu CC201/labs/3_K8sScaleAndUpdate/'dır. Hoş geldin mesajını 'Hello world from ' + hostname + '! Your app is up and running!\n' ifadesinden 'Welcome to ' + hostname + '! Your app is up and running!\n' ifadesine değiştirin. İşlem tamamlandıında dosyayı kaydetmeyi unutmayın.



```
app.js x
CC201 > labs > 3_K8sScaleAndUpdate > app.js > ...
1 var express = require('express')
2 var os = require("os");
3 var hostname = os.hostname();
4 var app = express()
5
6 app.get('/', function(req, res) {
7   res.send('Welcome to ' + hostname + '! Your app is up and running!\n')
8 }
9 app.listen(8080, function() {
10   console.log('Sample app is listening on port 8080.')
11 })
```

2. Bu yeni sürümü Container Registry'ye oluşturun ve gönderin. Bu uygulamanın ikinci sürümü olduğunu belirtmek için etiketi güncelleyin. proxy komutunu çalıştırmayan terminal penceresini kullandığınızdan emin olun.

NOT: proxy komutunu çalıştırın terminali kapatmayın.

```
docker build -t us.icr.io/$MY_NAMESPACE/hello-world:2 . && docker push us.icr.io/$MY_NAMESPACE/hello-world:2
```

```
theia@theiadocker- [ ] :/home/project/CC201/labs/3_K8sScaleAndUpdate$ docker build -t us.icr.io/$MY_NAMESPACE/hello-  
2 . && docker push us.icr.io/$MY_NAMESPACE/hello-world:2  
Sending build context to Docker daemon 6.144kB  
Step 1/6 : FROM node:9.4.0-alpine  
--> b5f94997f35f  
Step 2/6 : COPY app.js .  
--> f25f279213f5  
Step 3/6 : COPY package.json .  
--> 7d7357f01482  
Step 4/6 : RUN npm install && apk update && apk upgrade  
--> Running in cf3918a57f10  
npm notice created a lockfile as package-lock.json. You should commit this file.  
npm WARN hello-world-armada@0.0.1 No repository field.  
npm WARN hello-world-armada@0.0.1 No license field.  
  
added 50 packages in 1.662s  
fetch http://dl-cdn.alpinelinux.org/alpine/v3.6/main/x86_64/APKINDEX.tar.gz  
fetch http://dl-cdn.alpinelinux.org/alpine/v3.6/community/x86_64/APKINDEX.tar.gz  
v3.6.5-44-gda55e27396 [http://dl-cdn.alpinelinux.org/alpine/v3.6/main]  
v3.6.5-34-gf0ba0b43d5 [http://dl-cdn.alpinelinux.org/alpine/v3.6/community]  
OK: 8448 distinct packages available  
Upgrading critical system libraries and apk-tools:  
(1/1) Upgrading apk-tools (2.7.5-r0 -> 2.7.6-r0)  
Executing busybox-1.26.2-r9.trigger  
Continuing the upgrade transaction with new apk-tools:  
(1/7) Upgrading musl (1.1.16-r14 -> 1.1.16-r15)  
(2/7) Upgrading busybox (1.26.2-r9 -> 1.26.2-r11)  
Executing busybox-1.26.2-r11.post-upgrade  
(3/7) Upgrading libressl2.5-libcrypto (2.5.5-r0 -> 2.5.5-r2)  
(4/7) Upgrading libressl2.5-libssl (2.5.5-r0 -> 2.5.5-r2)  
(5/7) Installing libressl2.5-libtls (2.5.5-r2)  
(6/7) Installing ssl_client (1.26.2-r11)  
(7/7) Upgrading musl-utils (1.1.16-r14 -> 1.1.16-r15)  
Executing busybox-1.26.2-r11.trigger  
OK: 5 MiB in 15 packages  
Removing intermediate container cf3918a57f10  
--> 80a17e776942  
Step 5/6 : EXPOSE 8080  
--> Running in a868dd640957  
Removing intermediate container a868dd640957  
--> e2e4773f5ed3  
Step 6/6 : CMD node app.js  
--> Running in dad7dc244e00  
Removing intermediate container dad7dc244e00  
--> ce8704ad297f  
Successfully built ce8704ad297f  
Successfully tagged us.icr.io/sn-labs-[REDACTED]/hello-world:2  
The push refers to repository [us.icr.io/sn-labs-[REDACTED]/hello-world]  
237f3805cc80: Pushed  
2e7bcf63d006: Layer already exists  
ceb7ca869893: Pushed  
0804854a4553: Layer already exists  
6bd4a62f5178: Layer already exists  
9dfa40a0da3b: Layer already exists  
2: digest: sha256:839ba8ee302a5b4be4bcc4ad0c701b2c76627a592c9c7788f9e30674ab900748 size: 1576  
theia@theiadocker- [ ] :/home/project/CC201/labs/3_K8sScaleAndUpdate$
```

3. Şu ana kadar yüklediğiniz bu uygulamanın tüm farklı sürümlerini görmek için Container Registry'deki resimleri listeleyin.

```
ibmcloud cr images
```

Repository	Tag	Digest	Namespace
us.icr.io/sn-labs-snehar/hello-world	1	584c58ccc30f	sn-labs-s
us.icr.io/sn-labs-snehar/hello-world	2	0cddeed2656c	sn-labs-s
us.icr.io/sn-labsassets/categories-watson-nlp-runtime	latest	6b01b1e5527b	sn-labsas
us.icr.io/sn-labsassets/classification-watson-nlp-runtime	latest	dbd407898549	sn-labsas
us.icr.io/sn-labsassets/concepts-watson-nlp-runtime	latest	1e4741f10569	sn-labsas
us.icr.io/sn-labsassets/custom-watson-nlp-runtime	latest	f6513e19a33d	sn-labsas
us.icr.io/sn-labsassets/detag-watson-nlp-runtime	latest	38916c2119fc	sn-labsas
us.icr.io/sn-labsassets/emotion-watson-nlp-runtime	latest	1c9de1d27318	sn-labsas
us.icr.io/sn-labsassets/entity-mentions-bert-watson-nlp-runtime	latest	57d92957214f	sn-labsas
us.icr.io/sn-labsassets/entity-mentions-bilstm-watson-nlp-runtime	latest	76dbd3bdb12b	sn-labsas
us.icr.io/sn-labsassets/entity-mentions-rbr-multi-watson-nlp-runtime	latest	577399d7b4e7	sn-labsas
us.icr.io/sn-labsassets/entity-mentions-rbr-watson-nlp-runtime	latest	506cc92ecd3f	sn-labsas
us.icr.io/sn-labsassets/entity-mentions-sire-watson-nlp-runtime	latest	cd4e48efd3f6	sn-labsas
us.icr.io/sn-labsassets/entity-mentions-transformer-watson-nlp-runtime	latest	0584c56563ce	sn-labsas
us.icr.io/sn-labsassets/instructions-splitter	latest	2af122cfe4ee	sn-labsas
us.icr.io/sn-labsassets/keywords-watson-nlp-runtime	latest	e2b9dc471ae0	sn-labsas
us.icr.io/sn-labsassets/lang-detect-watson-nlp-runtime	latest	4d3b44e72af0	sn-labsas
us.icr.io/sn-labsassets/noun-phrases-watson-nlp-runtime	latest	c696f6af9797	sn-labsas
us.icr.io/sn-labsassets/pgadmin-theia	latest	0adf67ad81a3	sn-labsas
us.icr.io/sn-labsassets/phpmyadmin	latest	b66c30786353	sn-labsas
us.icr.io/sn-labsassets/relations-sire-watson-nlp-runtime	latest	65c2e74995d5	sn-labsas
us.icr.io/sn-labsassets/relations-transformer-watson-nlp-runtime	latest	18ffd6c35726	sn-labsas
us.icr.io/sn-labsassets/relations-watson-nlp-runtime	latest	3547dcc15c43	sn-labsas

Yeni görüntünün No Issues gösterdiğinde emin olun, aksi takdirde sorun kalmayana kadar görüntüyü birkaç kez yeniden çalıştırın.

4. Dağıtımını bu sürümü kullanacak şekilde güncelleyin.

```
kubectl set image deployment/hello-world hello-world=us.icr.io/$MY_NAMESPACE/hello-world:2
```

```
theia@theiadocker- [ ] :/home/project/CC201/labs/3_K8sScaleAndUpdate$ kubectl set image deployment/hello-world hello-world=us.icr.io/$MY_NAMESPACE/hello-world:2
deployment.apps/hello-world image updated
```

5. Aşağıdaki komutu kullanarak rolling güncellemenin durumunu alın:

```
kubectl rollout status deployment/hello-world
```

```
theia@theiadocker- [ ] :/home/project/CC201/labs/3_K8sScaleAndUpdate$ kubectl rollout status deployment/hello-world
deployment "hello-world" successfully rolled out
```

6. Yeni etiketin görüntü için kullanıldığını görmek üzere wide seçeneği ile Deployment'ı da alabilirsiniz.

```
kubectl get deployments -o wide
```

```
theia@theiadocker- [ ] :/home/project/CC201/labs/3_K8sScaleAndUpdate$ kubectl get deployments -o wide
NAME      READY   UP-TO-DATE   AVAILABLE   AGE     CONTAINERS   IMAGES
hello-world 1/1       1           1          39m    hello-world   us.icr.io/sn-labs- [REDACTED] /hello-world:2   SELECTOR
run=hello-world
```

IMAGES sütununu kontrol edin ve etiketin 2 olduğundan emin olun.

7. Uygulamanızı pingleyin ve yeni karşılama mesajının görüntülendiğinden emin olun.

```
curl -L localhost:8001/api/v1/namespaces/sn-labs-$USERNAME/services/hello-world/proxy
```

```
theia@theiadocker- [~]:/home/project/CC201/labs/3_K8sScaleAndUpdate$ curl -L localhost:8001/api/v1/namespaces/sn-lab
ERNAME/services/hello-world/proxy
Welcome to hello-world-5cc6f44c5-zhh96! Your app is up and running!
```

8. Bir uygulamanın yeni bir versiyonu bir hata içerebilir. Bu durumda, Kubernetes Deployment’ı şu şekilde geri alabilir:

```
kubectl rollout undo deployment/hello-world
```

```
theia@theiadocker- [~]:/home/project/CC201/labs/3_K8sScaleAndUpdate$ kubectl rollout undo deployment/hello-world
deployment.apps/hello-world rolled back
```

9. Aşağıdaki komutu kullanarak sürekli güncellemenin durumunu alın:

```
kubectl rollout status deployment/hello-world
```

```
theia@theiadocker- [~]:/home/project/CC201/labs/3_K8sScaleAndUpdate$ kubectl rollout status deployment/hello-world
deployment "hello-world" successfully rolled out
```

10. Eski etiketin kullanıldığını görmek için wide seçeneği ile Deployment’ı alın.

```
kubectl get deployments -o wide
```

```
theia@theiadocker- [~]:/home/project/CC201/labs/3_K8sScaleAndUpdate$ kubectl get deployments -o wide
NAME      READY   UP-TO-DATE   AVAILABLE   AGE   CONTAINERS   IMAGES   SELECTOR
hello-world  1/1     1           1          40m   hello-world   us.icr.io/sn-labs-   /hello-world:1   run=hello
theia@theiadocker- [~]:/home/project/CC201/labs/3_K8sScaleAndUpdate$
```

IMAGES sütununu kontrol edin ve etiketin 1 olduğundan emin olun.

11. Uygulamanızı pingleyin ve önceki ‘**Merhaba Dünya.. Uygulamanız çalışıyor!**‘ mesajının görüntülendiğinden emin olun.

```
curl -L localhost:8001/api/v1/namespaces/sn-labs-$USERNAME/services/hello-world/proxy
```

```
theia@theiadocker- [~]:/home/project/CC201/labs/3_K8sScaleAndUpdate$ curl -L localhost:8001/api/v1/namespaces/sn-labs-$USERNA
Hello world from hello-world-79c5684b95-6xr4l! Your app is up and running!
theia@theiadocker- [~]:/home/project/CC201/labs/3_K8sScaleAndUpdate$
```

Bir ConfigMap Kullanarak Konfigürasyon Saklama

ConfigMap'ler ve Secrets, konfigürasyon bilgilerini koddan ayırt tutmak için kullanılır, böylece hiçbir şey sabit kodlanmaz. Ayrıca, uygulamanın yeniden dağıtılmamasına gerek kalmadan konfigürasyon değişikliklerini almasına olanak tanır. Bunu göstermek için, uygulamanın mesajını bir ConfigMap'te saklayacağız, böylece mesajı yalnızca ConfigMap'i güncelleyerek güncellebiliriz.

1. Yeni bir mesaj içeren bir ConfigMap oluşturun.

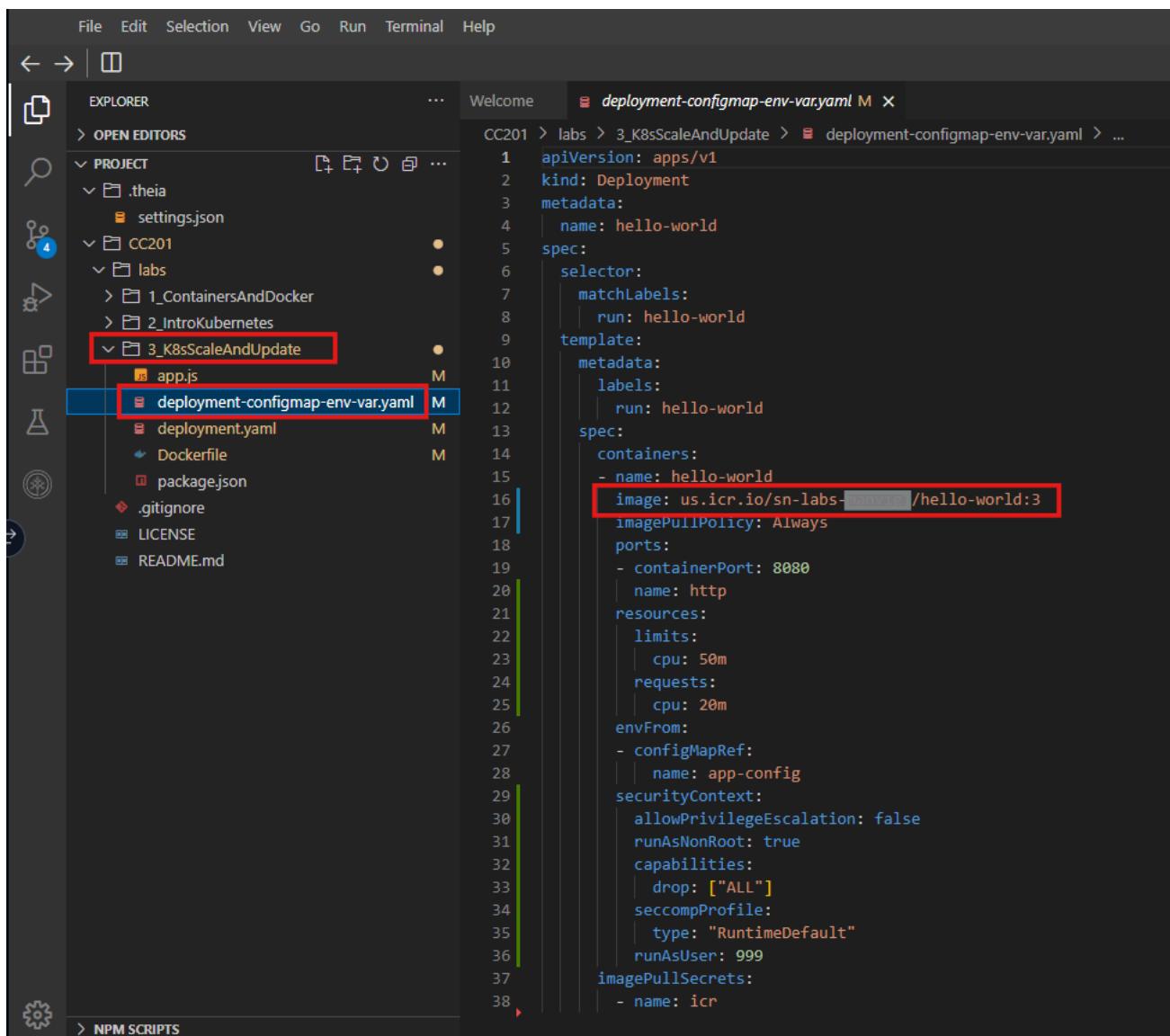
```
kubectl create configmap app-config --from-literal=MESSAGE="This message came from a ConfigMap!"
```

```
theia@theiadocker- [ ] :/home/project/CC201/labs/3_K8sScaleAndUpdate$ kubectl create configmap app-config --from-literal MESSAGE="This message came from a ConfigMap!"  
configmap/app-config created  
theia@theiadocker- [ ] :/home/project/CC201/labs/3_K8sScaleAndUpdate$
```

NOT: Eğer bu laboratuvarı daha önce denediyorsanız, önceki oturumun hala devam ediyor olma ihtimali vardır. Böyle bir durumda, aşağıdaki 'Created' mesajı yerine 'error: failed to create configmap: configmaps "app-config" already exists' mesajını göreceksiniz. Laboratuvarın ilerleyen adımlarına devam etmenizi öneririz.

```
theia@theiadocker- [ ] :/home/project/CC201/labs/3_K8sScaleAndUpdate$ kubectl create configmap app-config --from-literal=MESSAGE="This message came from a ConfigMap!"  
error: failed to create configmap: configmaps "app-config" already exists  
theia@theiadocker- [ ] :/home/project/CC201/labs/3_K8sScaleAndUpdate$
```

2. deployment-configmap-env-var.yaml dosyasını düzenlemek için Explorer'i kullanın. Bu dosyanın yolu CC201/labs/3_K8sScaleAndUpdate/'dir. <my_namespace> kısmına kendi ad alanınızı eklemeniz gerekiyor. İşiniz bittiğinde dosyayı kaydetmeye unutmayın.



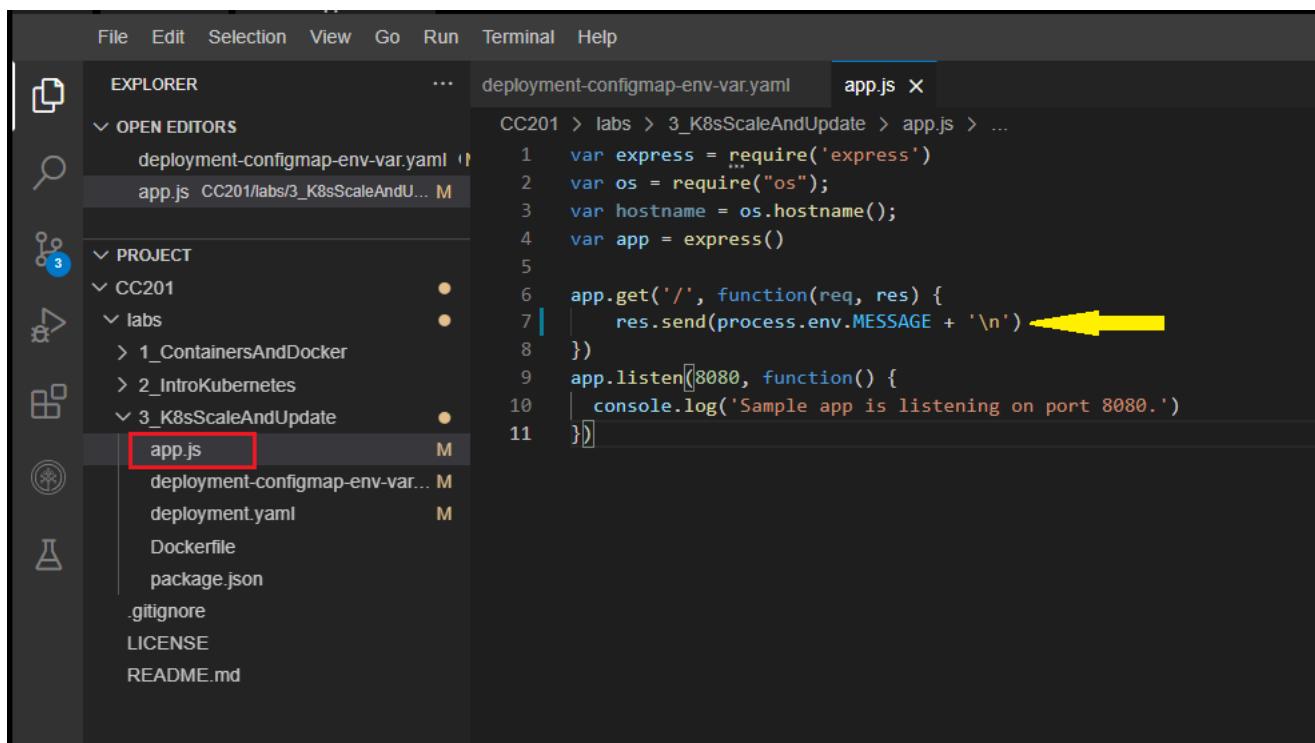
3. Aynı dosyada, aşağıda yeniden üretilen bölümü dikkatlice inceleyin. Alt kısım, ortam değişkenlerinin app-config adlı bir ConfigMap'ten alınan verilerle konteynerde tanımlanması gerektiğini belirtmektedir.

```
containers:
- name: hello-world
  image: us.icr.io/<my_namespace>/hello-world:3
  ports:
  - containerPort: 8080
  envFrom:
  - configMapRef:
    name: app-config
```

4. Explorer'ı kullanarak app.js dosyasını açın. Bu dosyanın yolu CC201/labs/3_K8sScaleAndUpdate/'dır. res.send('Welcome to ' + hostname + '! Your app is up and running!\n') diyen satırı bulun.

Bu satırı aşağıdaki gibi düzenleyin:

```
res.send(process.env.MESSAGE + '\n')
```



The screenshot shows the Visual Studio Code interface. On the left is the Explorer sidebar, which lists the project structure: CC201 > labs > 3_K8sScaleAndUpdate > app.js. The app.js file is currently selected and highlighted with a red border. In the main editor area, the code for app.js is displayed:

```
var express = require('express')
var os = require("os");
var hostname = os.hostname();
var app = express()

app.get('/', function(req, res) {
  res.send(process.env.MESSAGE + '\n')
})
app.listen(8080, function() {
  console.log('Sample app is listening on port 8080.')
})
```

A yellow arrow points to the line `res.send(process.env.MESSAGE + '\n')`, indicating where the user should edit the code.

İşlemi tamamladığınızda dosyayı kaydetmeyi unutmayın. Bu değişiklik, uygulamaya yapılan isteklerin ortam değişkeni MESSAGE'i döndürecekini belirtir.

5. Yeni uygulama kodunuza içeren yeni bir imaj oluşturun ve yükleyin.

```
docker build -t us.icr.io/$MY_NAMESPACE/hello-world:3 . && docker push us.icr.io/$MY_NAMESPACE/hello-world:3
```

```

theia@theiadocker- [ ] :/home/project/CC201/labs/3_K8sScaleAndUpdate$ docker build -t us.icr.io/$MY_NAMESPACE/hello-world:3 .
&& docker push us.icr.io/$MY_NAMESPACE/hello-world:3
Sending build context to Docker daemon 6.144kB
Step 1/6 : FROM node:9.4.0-alpine
--> b5f94997f35f
Step 2/6 : COPY app.js .
--> 3f0b66f4e16f
Step 3/6 : COPY package.json .
--> 8bcce318978a
Step 4/6 : RUN npm install && apk update && apk upgrade
--> Running in 7d432320817c
npm notice created a lockfile as package-lock.json. You should commit this file.
npm WARN hello-world-armada@0.0.1 No repository field.
npm WARN hello-world-armada@0.0.1 No license field.

added 50 packages in 1.615s
fetch http://dl-cdn.alpinelinux.org/alpine/v3.6/main/x86_64/APKINDEX.tar.gz
fetch http://dl-cdn.alpinelinux.org/alpine/v3.6/community/x86_64/APKINDEX.tar.gz
v3.6.5-44-gda55e27396 [http://dl-cdn.alpinelinux.org/alpine/v3.6/main]
v3.6.5-34-gf0ba0b43d5 [http://dl-cdn.alpinelinux.org/alpine/v3.6/community]
OK: 8448 distinct packages available
Upgrading critical system libraries and apk-tools:
(1/1) Upgrading apk-tools (2.7.5-r0 -> 2.7.6-r0)
Executing busybox-1.26.2-r9.trigger
Continuing the upgrade transaction with new apk-tools:
(1/7) Upgrading musl (1.1.16-r14 -> 1.1.16-r15)
(2/7) Upgrading busybox (1.26.2-r9 -> 1.26.2-r11)
Executing busybox-1.26.2-r11.post-upgrade
(3/7) Upgrading libressl2.5-libcrypto (2.5.5-r0 -> 2.5.5-r2)
(4/7) Upgrading libressl2.5-libssl (2.5.5-r0 -> 2.5.5-r2)
(5/7) Installing libressl2.5-libtls (2.5.5-r2)
(6/7) Installing ssl_client (1.26.2-r11)
(7/7) Upgrading musl-utils (1.1.16-r14 -> 1.1.16-r15)
Executing busybox-1.26.2-r11.trigger
OK: 5 MiB in 15 packages
Removing intermediate container 7d432320817c
--> ed77983749d5
Step 5/6 : EXPOSE 8080
--> Running in 5686c39353f8
Removing intermediate container 5686c39353f8
--> 529399efa32f
Step 6/6 : CMD node app.js
--> Running in 942b22038f71
Removing intermediate container 942b22038f71
--> 6e2bc34c6c21
Successfully built 6e2bc34c6c21
Successfully tagged us.icr.io/sn-labs-[REDACTED]/hello-world:3
The push refers to repository [us.icr.io/sn-labs-[REDACTED]/hello-world]
d4bcd81b0ba6: Pushed
2e7bcf63d006: Layer already exists
adf91d207735: Pushed
0804854a4553: Layer already exists
6bd4a62f5178: Layer already exists
9dfa40a0da3b: Layer already exists
3: digest: sha256:b9b9ee39218a0bc88a121fa60e6a1d1d4a5c5eae2d6122fc87b8d7f3911e5a8f size: 1576

```

deployment-configmap-env-var.yaml dosyası zaten 3 etiketini kullanacak şekilde yapılandırılmıştır.

6. Yeni Dağıtım yapılandırmasını uygulayın.

```
kubectl apply -f deployment-configmap-env-var.yaml
```

```

theia@theiadocker- [ ] :/home/project/CC201/labs/3_K8sScaleAndUpdate$ kubectl apply -f deployment-configmap-env-var.yaml
deployment.apps/hello-world configured

```

7. Uygulamanızı tekrar pingleyin ve ortam değişkeninden gelen mesajın döndürülüp döndürülmediğini kontrol edin.

NOT: Bu komutu tekrar çalıştırabilirsiniz. Hemen "This message came from a ConfigMap!" mesajını göstermeyebilir.

```
curl -L localhost:8001/api/v1/namespaces/sn-labs-$USERNAME/services/hello-world/proxy
```

```
theia@theiadocker- [ ] :/home/project/CC201/labs/3_K8sScaleAndUpdate$ curl -L localhost:8001/api/v1/namespaces/sn-lab-ERNAME/services/hello-world/proxy
This message came from a ConfigMap!
```

Eğer “Bu mesaj bir ConfigMap’ten geldi!” mesajını göründürseniz, harika bir iş çıkardınız!

NOT: Eğer önceki oturumunuz hala devam ediyorsa, aşağıdaki çıktıyı görebilirsiniz. Eğer öyleyse, laboratuvarın sonraki adımlarına geçmenizi öneririz.

```
theia@theiadocker- [ ] :/home/project/CC201/labs/3_K8sScaleAndUpdate$ curl -L localhost:8001/api/v1/namespaces/sn-labs-$USERNA-TERNAME/services/hello-world/proxy
This message is different, and you didn't have to rebuild the image!
theia@theiadocker- [ ] :/home/project/CC201/labs/3_K8sScaleAndUpdate$
```

8. Konfigürasyon koddu olduğu için, mesajı imajı yeniden inşa etmeden değiştirebilirsiniz. Aşağıdaki komutu kullanarak eski ConfigMap’i silin ve aynı isimle ancak farklı bir mesajla yeni bir tane oluşturun.

```
kubectl delete configmap app-config && kubectl create configmap app-config --from-literal=MESSAGE="This message is different, and you didn't have to rebuild the image!"
```

```
theia@theiadocker- [ ] :/home/project/CC201/labs/3_K8sScaleAndUpdate$ kubectl delete configmap app-config && kubectl e- configmap app-config --from-literal=MESSAGE="This message is different, and you didn't have to rebuild the image!"
configmap "app-config" deleted
configmap/app-config created
```

9. Deployment’ı yeniden başlatın, böylece konteynerler yeniden başlar. Bu, ortam değişkenlerinin başlangıçta ayarlandığı için gereklidir.

```
kubectl rollout restart deployment hello-world
```

```
theia@theiadocker- [ ] :/home/project/CC201/labs/3_K8sScaleAndUpdate$ kubectl rollout restart deployment hello-world
deployment.apps/hello-world restarted
```

10. Uygulamanızı tekrar pingleyin ve ortam değişkeninden yeni mesajın döndürülp döndürülmediğini kontrol edin.

```
curl -L localhost:8001/api/v1/namespaces/sn-labs-$USERNA-TERNAME/services/hello-world/proxy
```

```
theia@theiadocker- [ ] :/home/project/CC201/labs/3_K8sScaleAndUpdate$ curl -L localhost:8001/api/v1/namespaces/sn-lab-ERNAME/services/hello-world/proxy
This message is different, and you didn't have to rebuild the image!
```

hello-world uygulamasını Yatay Pod Autoscaler kullanarak otomatik ölçeklendirme

1. deployment.yaml dosyasındaki 20 ile 26. satırları, CPU kaynak tahsisini artırmak için aşağıdaki bölümle template.spec.containers altında değiştirin.

```
name: http
resources:
  limits:
    cpu: 50m
  requests:
    cpu: 20m
```

Not: Değişiklikleri yaptıktan sonra dosyayı kaydetmeyi unutmayın.

Güncellenmiş dosya aşağıdaki gibi olacaktır:

```
kind: Deployment
  apiVersion: apps/v1
  kind: Deployment
  metadata:
    name: hello-world
  spec:
    selector:
      matchLabels:
        run: hello-world
    template:
      metadata:
        labels:
          run: hello-world
      spec:
        containers:
          - name: hello-world
            image: us.icr.io/[REDACTED]/hello-world:1
            imagePullPolicy: Always
            ports:
              - containerPort: 8080
                name: http
            resources:
              limits:
                cpu: 50m
              requests:
                cpu: 20m
        securityContext:
          allowPrivilegeEscalation: false
          runAsNonRoot: true
          capabilities:
            drop: ["ALL"]
          seccompProfile:
            type: "RuntimeDefault"
          runAsUser: 999
        imagePullSecrets:
          - name: icr
```

2. Dağıtımı uygulayın:

```
kubectl apply -f deployment.yaml
```

```
theia@theiadocker-: /home/project/CC201/labs/3_K8sScaleAndUpdate$ kubectl apply -f deployment.yaml  
deployment.apps/hello-world configured
```

3. Aşağıdaki komutu kullanarak hello-world dağıtımını otomatik ölçeklendirin:

```
kubectl autoscale deployment hello-world --cpu-percent=5 --min=1 --max=10
```

```
theia@theiadocker- [ ] /home/project/CC201/labs/3_K8sScaleAndUpdate$ kubectl autoscale deployment hello-world --cpu-percent=100 horizontalpodautoscaler.autoscaling/hello-world autoscaled  
theia@theiadocker [ ] /home/project/CC201/labs/3_K8sScaleAndUpdate$ [ ]
```

4. Yeni olusturulan HorizontalPodAutoscaler'in mevcut durumunu kontrol etmek icin su komutu calistirabilirsiniz:

```
kubectl get hpa hello-world
```

```
^Ctheia@theiadocker- [~] /home/project/CC201/labs/3_K8sScaleAndUpdate$ kubectl get hpa hello-world
NAME          REFERENCE      TARGETS      MINPODS   MAXPODS   REPLICAS   AGE
hello-world   Deployment/hello-world  0%/5%       1          10        1          19m
theia@theiadocker- [~] /home/project/CC201/labs/3_K8sScaleAndUpdate$
```

5. Lütfen ikinci terminalde kubernetes proxy'sinin hala çalıştığından emin olun. Eğer çalışmıyorsa, lütfen tekrar başlatmak için su komutu çalıştırın:

kubectl proxy

6. Baska bir yeni terminal acın ve yükü artırmak için uygulamayı birden fazla istekle spamlamak için aşağıdaki komutu girin:

```
for i in `seq 1000000`; do curl -L localhost:8001/api/v1/namespaces/sn-labs-$USERNAME/services/hello-world/proxy; done
```

1. terminaldeki diğer komutlara devam edin

7. Aşağıdaki komutu çalıştırarak otomatik ölçeklendirmeye göre kopyaların arttığını gözlemleyin:

```
kubectl get hpa hello-world --watch
```

NAME	REFERENCE	TARGETS	MINPODS	MAXPODS	REPLICAS	AGE
hello-world	Deployment/hello-world	0%/5%	1	10	1	153m
hello-world	Deployment/hello-world	30%/5%	1	10	1	154m
hello-world	Deployment/hello-world	30%/5%	1	10	4	154m
hello-world	Deployment/hello-world	30%/5%	1	10	6	154m
hello-world	Deployment/hello-world	25%/5%	1	10	6	154m
hello-world	Deployment/hello-world	17%/5%	1	10	6	155m
hello-world	Deployment/hello-world	17%/5%	1	10	7	155m
hello-world	Deployment/hello-world	22%/5%	1	10	7	156m
hello-world	Deployment/hello-world	22%/5%	1	10	9	156m

Uygulamanızın otomatik ölçeklendirildiğini gösteren kopya sayısında bir artış göreceksiniz.

Bu komutu durdurmak için CTRL + C tuşlarına basın.

8. Aşağıdaki komutu çalıştırarak yatay pod otomatik ölçeklendiricisinin detaylarını gözlemleyin:

```
kubectl get hpa hello-world
```

NAME	REFERENCE	TARGETS	MINPODS	MAXPODS	REPLICAS	AGE
hello-world	Deployment/hello-world	5%/5%	1	10	9	160m

Şimdi replikaların sayısının arttığını göreceksiniz.

9. Diğer 2 terminalde çalışan proxy ve yük oluşturma komutlarını CTRL + C tuşlayarak durdurun.

10. Dağıtımını silin.

```
kubectl delete deployment hello-world
```

```
theia@theiadocker ~:/home/project/CC201/labs/3_K8sScaleAndUpdate$ kubectl delete deployment hello-world
deployment.apps "hello-world" deleted
theia@theiadocker ~:/home/project/CC201/labs/3_K8sScaleAndUpdate$
```

11. Servisi silin.

```
kubectl delete service hello-world
```

```
theia@theiadocker ~:/home/project/CC201/labs/3_K8sScaleAndUpdate$ kubectl delete service hello-world
service "hello-world" deleted
theia@theiadocker ~:/home/project/CC201/labs/3_K8sScaleAndUpdate$
```

Tebrikler! Bu kursun üçüncü modülüne ait laboratuvarı tamamladınız.

© IBM Corporation. Tüm hakları saklıdır.