

Mikroservislerin Dağıtımı



Tahmini süre: 90 dakika

Mikroservislerin Dağıtımı laboratuvarına hoş geldiniz. Bu laboratuvara **Resimler** ve **Şarkılar** hizmetlerini buluta dağıtaraksizez. Bu capstone'daki önceki laboratuvarlardan tüm kodun hazır olması gerekmektedir. Laboratuvar, çalışan kodu dağıtmaya odaklımaktadır.

- **Resimler** hizmeti IBM Code Engine'a dağıtılacaktır.
- **Şarkılar** hizmeti RedHat OpenShift'e dağıtılacaktır.

Öğrenme Hedefleri:

Bu laboratuvari tamamlandıktan sonra şunları yapabileceksiniz:

1. Laboratuvar ortamında Code Engine hizmetini başlatın.
2. Flask hizmetini Code Engine'a dağıtnın.
3. Laboratuvar ortamında RedHat OpenShift platformuna erişin.
4. Flask hizmetini RedHat OpenShift'e dağıtnın.

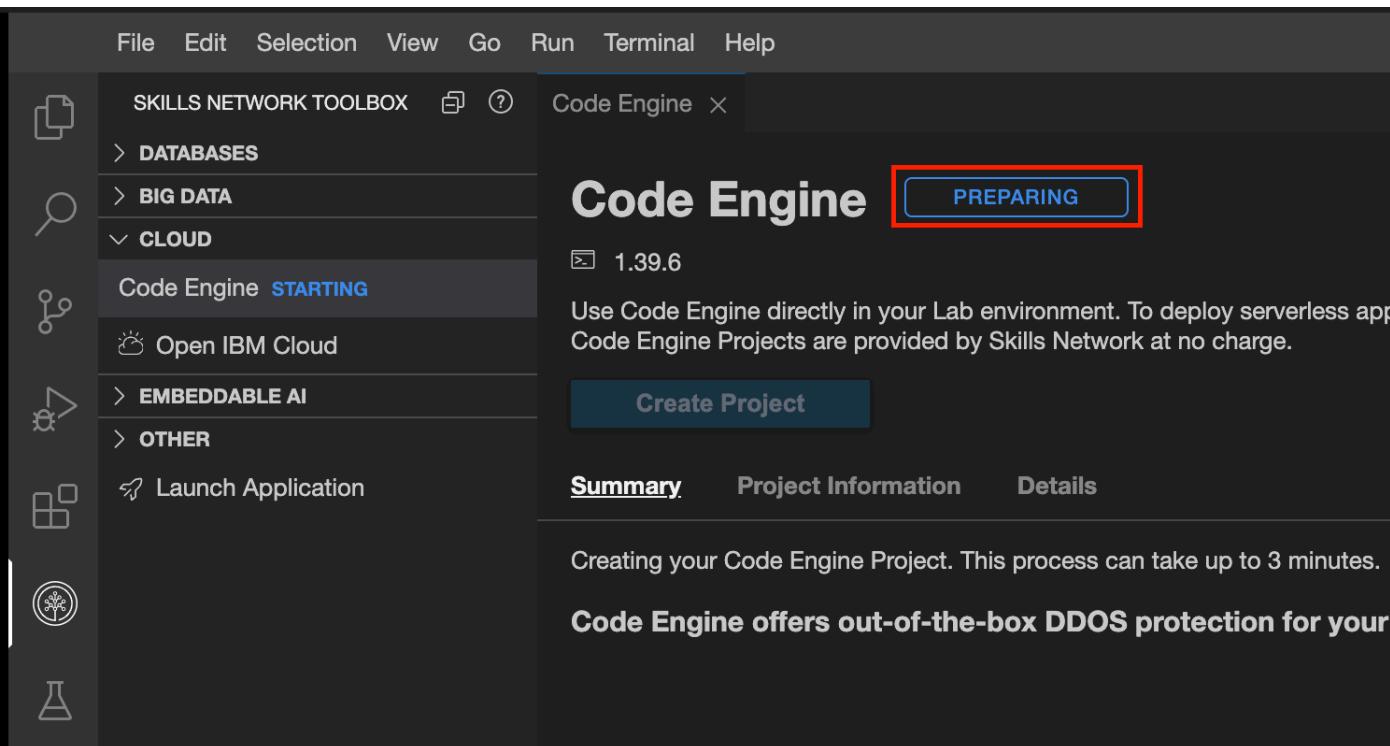
Egzersiz 1: Resimler - Başlangıç Kodu Motoru

Görevleriniz

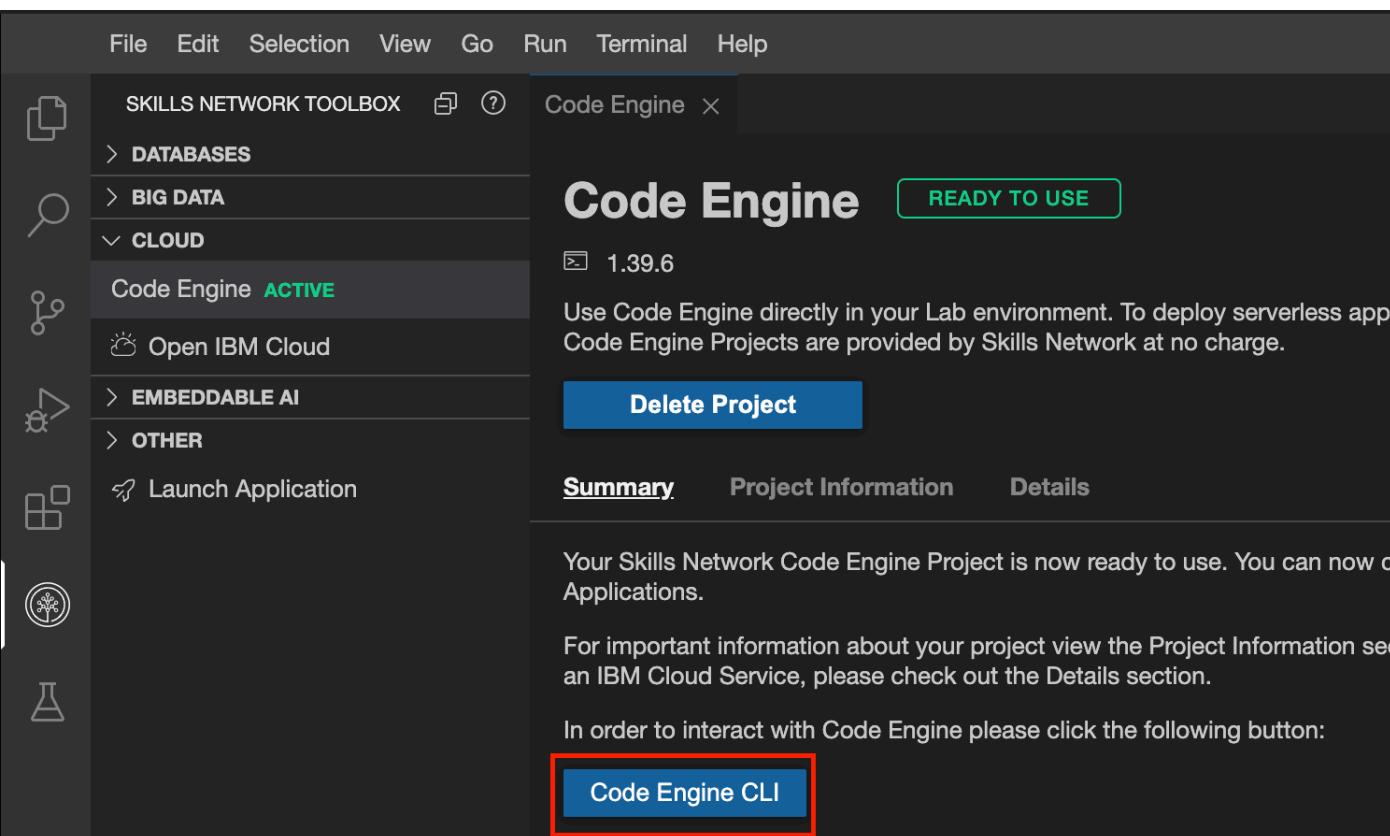
1. Laboratuvar ortamınızdaki menüden **Cloud** açılır menüsünü tıklayın ve **Code Engine** seçeneğini seçin. Kod motoru kurulum paneli görünecektir. Başlamak için **Create Project** butonuna tıklayın.

The screenshot shows the Skills Network Toolbox interface. On the left, there's a sidebar with various icons and sections like Databases, Big Data, Cloud, Embeddable AI, and Others. The 'Cloud' section is expanded, and the 'Code Engine' item is selected, highlighted with a red box. On the right, the main panel has a title 'Code Engine' with a status indicator 'NOT READY'. Below it, the version '1.39.6' is shown. A large button labeled 'Create Project' is also highlighted with a red box. At the bottom, there are tabs for 'Summary', 'Project Information', and 'Details', with 'Summary' being the active tab. The summary area contains text about getting started with Code Engine and deploying serverless apps.

2. Kod motoru ortamının hazırlanması biraz zaman alır. Kurulum panelinde ilerleme durumu gösterilecektir.



3. Kod motoru kurulumu tamamlandığında, aktif olduğunu göreceksiniz. Aşağıda gösterildiği gibi terminalde önceden yapılandırılmış CLI'yi başlatmak için `Code Engine CLI` butonuna tıklayın.



Terminalin ekran görüntüsünü alın ve `deploy-getpic-1.jpg` olarak kaydedin.

4. Önceden yapılandırılmış CLI'nin başlatıldığını ve ana dizinin mevcut dizine ayarlandığını gözlemyeceksiniz. Ön yapılandırmının bir parçası olarak proje kurulmuş ve Kubeconfig ayarlanmıştır. Detaylar terminalde aşağıdaki gibi gösterilmektedir.

The screenshot shows the Skills Network Toolbox interface with the 'Code Engine' project selected. The main panel displays the 'Code Engine' project details, including version 1.39.6 and a message stating 'READY TO USE'. Below this, there are tabs for 'Delete Project', 'Summary', 'Project Information', and 'Details'. The 'Summary' tab is active, showing a message: 'Your Skills Network Code Engine Project is now ready to use. You can now deploy your applications.' It also mentions that Code Engine Projects are provided by Skills Network at no charge. A large blue button labeled 'Code Engine CLI' is highlighted with a red box. The terminal window below shows the command 'ibmcloud ce project current' being run, followed by the output: 'ibmcloud ce project current theia@theiadocker-captainfedor1:/home/project\$ ibmcloud ce project current Getting the current project context... OK'. The output then lists project details: 'Name: Code Engine - sn-labs-captainfedor1', 'ID: 9c079722-5f80-4056-bed6-798cdb0acf04', 'Subdomain: ywj8nhvp9f9', 'Domain: us-south.codeengine.appdomain.cloud', and 'Region: us-south'. The 'Kubernetes Config' section shows the context and environment variable: 'Context: ywj8nhvp9f9' and 'Environment Variable: export KUBECONFIG="/home/theia/.bluemix-project/.config/kubeconfig"'.

Kanıt

1. `ibmcloud ce project current` komutunun çıktısını gösteren terminalin ekran görüntüsünü alın.
2. Ekran görüntüsünü `deploy-getpic-1.jpg` (veya .png) olarak kaydedin.

Egzersiz 2: Resimler - GitHub Deposu Klonlama

Sonraki adım, **Resimler** mikroservis deposunu yerel laboratuvar ortamına klonlamaktır.

Görevleriniz

Daha önceki laboratuvarınızda sağlanan şablondan resimler servisi için yeni bir depo oluşturduğunuz. Eğer oluşturmadıysanız, [Flask ile Resimleri Alma Servisi Oluştur](#) laboratuvarına geri dönün ve bu laboratuvari tamamladığınızdan emin olun.

1. Açık bir terminal yoksa Terminal -> Yeni Terminal ile bir terminal açın.
2. Sonra, GitHub hesabınızın adını içeren bir ortam değişkeni dışa aktarmak için `export GITHUB_ACCOUNT` komutunu kullanın.

Not: Aşağıdaki `{your_github_account}` yer tutucusunu gerçek GitHub hesabınızla değiştirin:

```
export GITHUB_ACCOUNT={your_github_account}
```

3. Ardından, deponuzu klonlamak için aşağıdaki komutları kullanın.

```
git clone https://github.com/$GITHUB_ACCOUNT/Back-End-Development-Pictures.git
```

4. Aşağıdaki butona tıklayın ve `routes.py` dosyasının, önceki laboratuvarınızda yaptığınız ve gönderdiğiniz tüm değişiklikleri içeriip içermediğini kontrol edin. Değişiklikleri görmüyorsanız, [önceki laboratuvar](#) tekrar yapmanız ve Openshift konteynerinde dağıtımınızı yeniden oluşturmanız gereklidir.

[Open routes.py in IDE](#)

Egzersiz 3: Resimler - Uygulamayı yerel olarak çalıştır

Görevleriniz

Artık tüm kodu yerel laboratuvar ortamında bulduğunuzu göre, uygulamayı yerel olarak çalıştırarak beklediği gibi çalıştığından emin olalım. Önceli laboratuvara mikroservisi tamamlamış olmalısınız.

Görev 1: Uygulamayı yerel olarak çalıştır

- Proje dizinimize aşağıdaki gibi geçin.

```
cd Back-End-Development-Pictures
```

- Aşağıdaki komutu çalıştırarak yerel olarak bir kez test edin.

```
flask run --debugger --reload
```

- Yeni bir terminal açın ve uygulama sağlığını kontrol etmek için aşağıdaki komutu çalıştırın.

```
curl localhost:5000/health
```

Eğer uygulama yerel olarak çalışmazsa, modül 1'e geri dönmeli ve [Create Get Pictures Service with Flask](#) laboratuvarını tamamladığınızdan emin olmalısınız. Yerel olarak test etmeyi tamamladıktan sonra, sunucudan çıkmak için klavyede `ctrl+c` tuşlarına basabilirsiniz. Terminalden çıkmak için `exit` komutunu kullanabilirsiniz.

Alıştırma 4: Resimler - Code Engine'a Dağıtım

Uygulamanın beklediği gibi çalıştığından emin olduktan sonra, bir sonraki adım uygulamayı Code Engine'a dağıtmaktır. Bu, zaten sağlanmış olan Dockerfile'i gerektirecektir. Öncelikle uygulamanın bir görüntüsünü oluşturacak ve bunu hesabınız altında sağlanan laboratuvar görüntüyü kayıt defterine iteceksizez.

Görevleriniz

- Laboratuvar ortamı bir IBM Container Registry ile birlikte gelir. Görüntüleri yalnızca bir ad alanına `SN_ICR_NAMESPACE` itebilirsiniz. Terminali kullanarak bu ad alanının laboratuvar ortamınız için ne olduğunu bulun:

```
echo ${SN_ICR_NAMESPACE}
```

Şu çıktıyi görmelisiniz:

```
theia@theiaopenshift-captainfedo1:/home/project$ echo ${SN_ICR_NAMESPACE}
sn-labs-captainfedo1
```

- Aşağıdaki komut ile görüntütüyü oluşturun:

```
docker build -t pictures .
```

- Görüntüyü `us.icr.io/$SN_ICR_NAMESPACE/pictures:1` olarak etiketleyin.

```
docker tag pictures us.icr.io/$SN_ICR_NAMESPACE/pictures:1
```

4. Hem orijinal hem de etiketli görüntülerini görmek için:

```
docker images
```

Cıktınız, laboratuvar ortamı tarafından sağlanan herhangi bir ek görüntü ile birlikte iki görüntüyü içermelidir:

```
$ docker images
REPOSITORY          TAG      IMAGE ID      CREATED       SIZE
pictures            latest   7c230d5bcd9f  2 minutes ago  179MB
us.icr.io/sn-labs-captainfed01/pictures  1        7c230d5bcd9f  2 minutes ago  179MB
```

Orijinal görüntü ile etiketli görüntülerin aynı IMAGE ID'ye sahip olduğunu, farkın yalnızca etiketlerde olduğunu unutmayın.

5. Aşağıdaki komutu kullanarak görüntütü kayit defterine itin:

```
docker push us.icr.io/$SN_ICR_NAMESPACE/pictures:1
```

6. Aşağıdaki komutu kullanarak ad alanınızda tüm görüntülerini listeleyebilirsiniz.

```
ibmcloud cr images --restrict $SN_ICR_NAMESPACE
```

Yeni itilen görüntütü görelisiniz:

```
$ ibmcloud cr images --restrict $SN_ICR_NAMESPACE
Listing images...
Repository          Tag      Digest      Namespace      Created      Size      Security status
us.icr.io/sn-labs-captainfed01/pictures  1        f3c8e1ef47f4  sn-labs-captainfed01  7 minutes ago  68 MB      -
OK
```

7. Uygulamayı Code Engine'a dağıtmak için aşağıdaki komutu kullanın:

```
ibmcloud ce app create --name pictures --image us.icr.io/${SN_ICR_NAMESPACE}/pictures:1 --registry-secret icr-secret --port 3000
```

Bu komutun uygulamayı oluşturduğunu ve ayrıca gerekli altyapıyı içten yapılandırdığını göreceksiniz. Birkaç dakika süre ve sonunda URL ile birlikte bir onay verir. URL'niz farklı görünecek:

```
$ ibmcloud ce app create --name pictures --image us.icr.io/${SN_ICR_NAMESPACE}/pictures:1 --registry-secret icr-secret --port 3000
Creating application 'pictures'...
The Route is still working to reflect the latest desired specification.
Configuration 'pictures' is waiting for a Revision to become ready.
Ingress has not yet been reconciled.
Waiting for load balancer to be ready.
Run 'ibmcloud ce application get -n pictures' to check the application status.
OK
https://pictures.zcx38jqwqmu.us-south.codeengine.appdomain.cloud
```

Terminalde `ibm ce app create --` komutunun çıktısını gösteren bir ekran görüntüsü alın ve bunu `deploy-getpic-2.png` (veya jpg) olarak kaydedin.

8. Uygulama URL'sini kopyalayın, yeni bir sekmeye yapıştırın ve **Enter** tuşuna basın. Sonuç görmek için geçerli bir yol eklemeniz gerekecek. Mikroservisi test etmek için URL/`count` veya URL/`health`'i deneyebilirsiniz. `/count` uç noktasını gösteren tarayıcı sekmesinin bir ekran görüntüsünü alın ve bunu `deploy-getpic-3.png` (veya jpg) olarak kaydedin.

9. Uygulama URL'sini kaybederseniz, aşağıdaki komutu kullanarak onu bulabilirsiniz:

```
ibmcloud ce application list
```

URL sütununda bağlantıyi görmelisiniz:

```
$ ibmcloud ce application list  
Listing all applications...  
OK  
Name      Status     URL          Latest    Age     Conditions  Reason  
pictures   Ready     https://pictures.zcx38jqwqmu.us-south.codeengine.appdomain.cloud pictures-00001 4m55s 3 OK / 3
```

Kanıt

- Uygulamanın son URL'sini içeren `ibmcloud ce create ...` komutunun çıktısını gösteren terminalin bir ekran görüntüsünü alın ve bunu `deploy-getpic-2.png` (veya `.jpg`) olarak kaydedin.
- Tarayıcınızda /count uc noktasındaki çalışan uygulamanın bir ekran görüntüsünü alın. Ekran görüntüsünü `deploy-getpic-3.png` (veya `.jpg`) olarak kaydedin.
- Resimler hizmetinin URL'sini kaydedin. Bunu bir sonraki laboratuvar çalışmasında ana Django uygulamasını mikroservisle bağlamak için kullanacaksınız.

Egzersiz 5: Resimler - Kod Motoru Uygulamasını Doğrula

Uygulamanızı Kod Motoru'nda çalıştırığınız için tebrikler. Bu son adım, uygulamayı terminal kullanarak doğrulayacaktır.

Görevleriniz

- Uygulamanın detaylarını almak için aşağıdaki komutu çalıştırın:

```
ibmcloud ce app get --name pictures
```

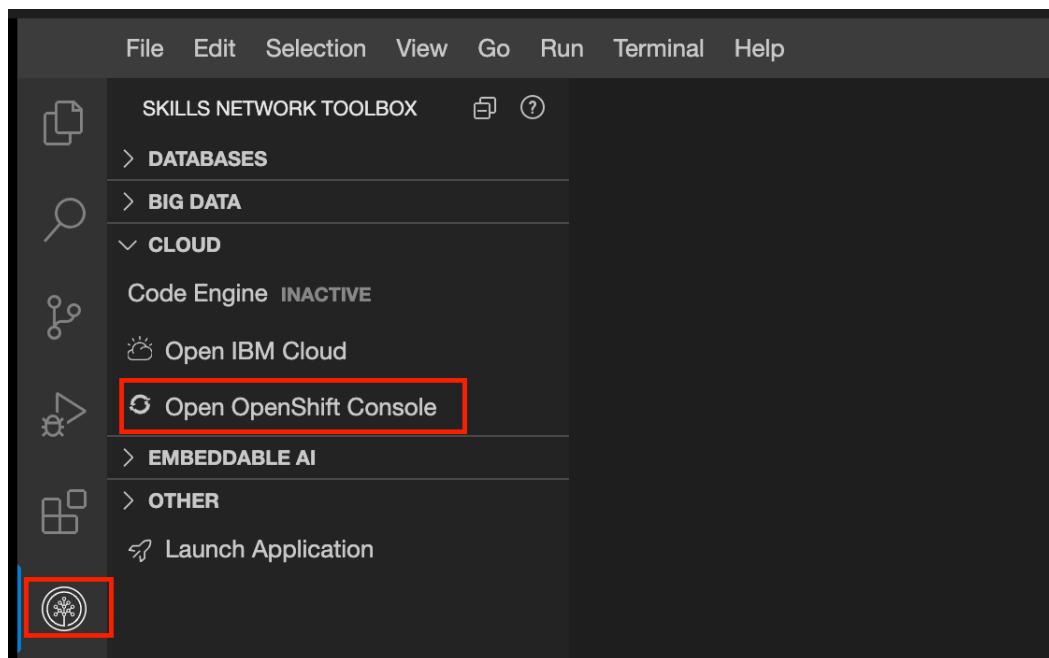
Uygulamanın oluşturulma zamanı, kullanılan kaynaklar, örnek sayısı ve diğer detaylar dahil olmak üzere ayrıntılı bilgileri görmelisiniz.

Alıştırma 6: Şarkılar - OpenShift'te MongoDB'yi Kurun

Önceki modüllerdeki alıştırmalar için laboratuvar ortamında sağlanan MongoDB sunucusunu kullandınız. Bu sunucu yalnızca laboratuvar ortamı ile sınırlıdır ve RedHat OpenShift gibi başka bir platformdan erişilemez. MongoDB'yi üretim ortamında veritabanı sunucusu olarak başarıyla kullanabilmek için bu alıştırmada RedHat OpenShift'e kuracaksınız.

Görevleriniz

- Laboratuvar ortamından OpenShift konsolunu başlatın. Bu, yeni bir sekme açacaktır.





Welcome to Dev Perspective!



Get started with a tour of some of the key areas in OpenShift 4.x Developer perspective that can help you complete workflows and be more productive.

[Skip tour](#)

[Get Started](#)

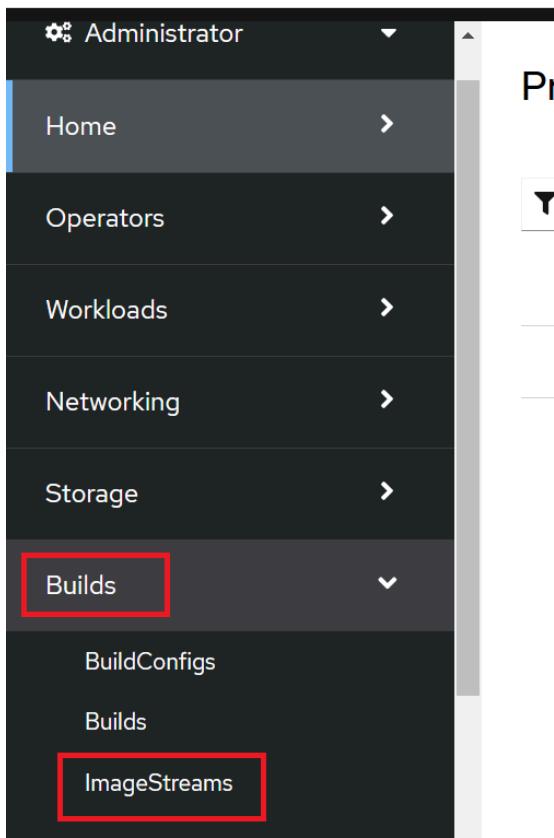
2. Topology seçeneğine tıklayın. Projenizde yalnızca bir uygulama görmelisiniz.

The screenshot shows the Skills Network OpenShift Lab interface. The top navigation bar includes a menu icon, the title "Skills Network OpenShift Lab", and icons for system:serviceaccount, labs-captainfedor1:captainfedor1, and a question mark. On the left sidebar, under the "Developer" category, the "Topology" option is highlighted with a red box. The main content area shows a dropdown for "Project: sn-labs-captainfedor1" (also with a red box), an "Application: all applications" dropdown, and a "View shortcuts" link. Below these are "Display Options" and "Filter by Resource" buttons, and a search bar with placeholder text "Find by name...". To the right, there is a large button with a circular logo containing a red and blue target-like symbol, labeled "D opensh...onsole".

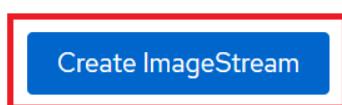
3. Sol menüden Yönetici seçeneğine geçin.

The screenshot shows the Skills Network OpenShift Lab interface. The left sidebar has a "Developer" dropdown at the top. Below it, the "Administrator" option is highlighted with a red box. Underneath, another "Developer" dropdown is visible.

4. Sol taraftaki menüdeki seçenekleri takip ederek Builds bölümüne erişin, ardından imageStreams kısmını tıklayıp açın.



5. Create ImageStream butonuna tıklayarak bir ImageStream oluşturma seçeneğini seçin.



6. Sağlanan kodu kopyalayıp editöre yapıştırın, ardından oluşturma seçeneğini seçin.

```
apiVersion: image.openshift.io/v1
kind: ImageStream
metadata:
  name: mongo
spec:
  lookupPolicy:
    local: false
  tags:
    - name: latest
      from:
        kind: DockerImage
        name: docker.io/library/mongo:latest
```

Create ImageStream

Create by manually entering YAML or JSON definitions, or by dragging and dropping a file into the editor.

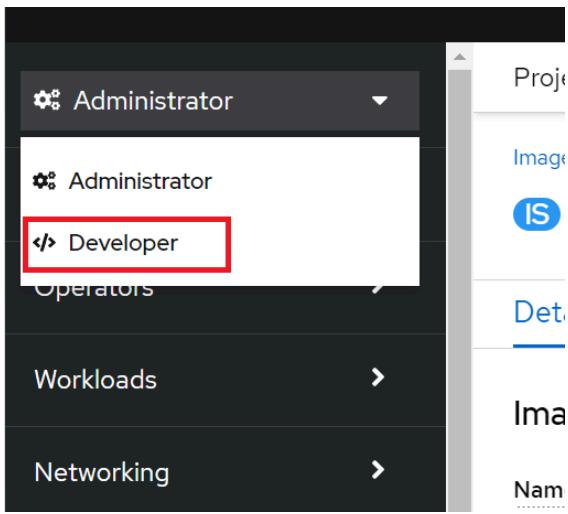
```
1 apiVersion: image.openshift.io/v1
2 kind: ImageStream
3 metadata:
4   name: mongo
5 spec:
6   lookupPolicy:
7     local: false
8   tags:
9     - name: latest
10    from:
11      kind: DockerImage
12      name: docker.io/library/mongo:latest
```

[Create](#)

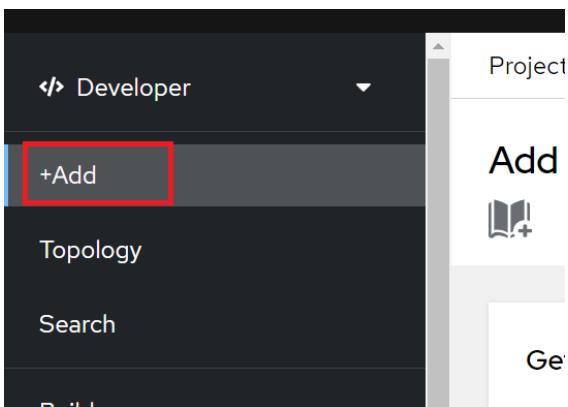
[Cancel](#)

[Download](#)

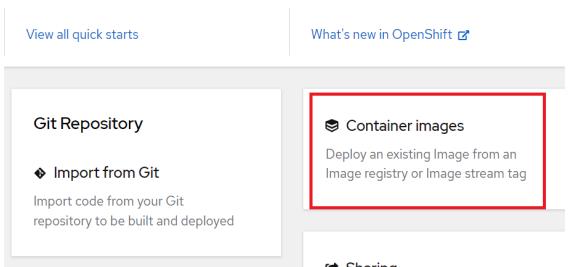
6. Sonraki adımda, sol menüdeki mevcut seçenekleri kullanarak Geliştirici moduna geçin.



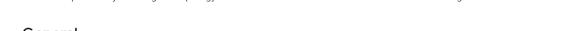
7. "Ekle+" seçeneğine tıklayarak seçin.



8. Önceki adımda oluşturulan görüntünün dağıtmak için "Konteyner görüntüleri" seçeneğini seçin.



9. "İç kayıt defterinden görüntü akışı etiketi" seçeneğini seçin, ardından görüntü akışında "mongo" ve etiket olarak "latest" seçin. Kalan seçenekleri bırakın ve "oluştur" butonuna tıklayarak devam edin.

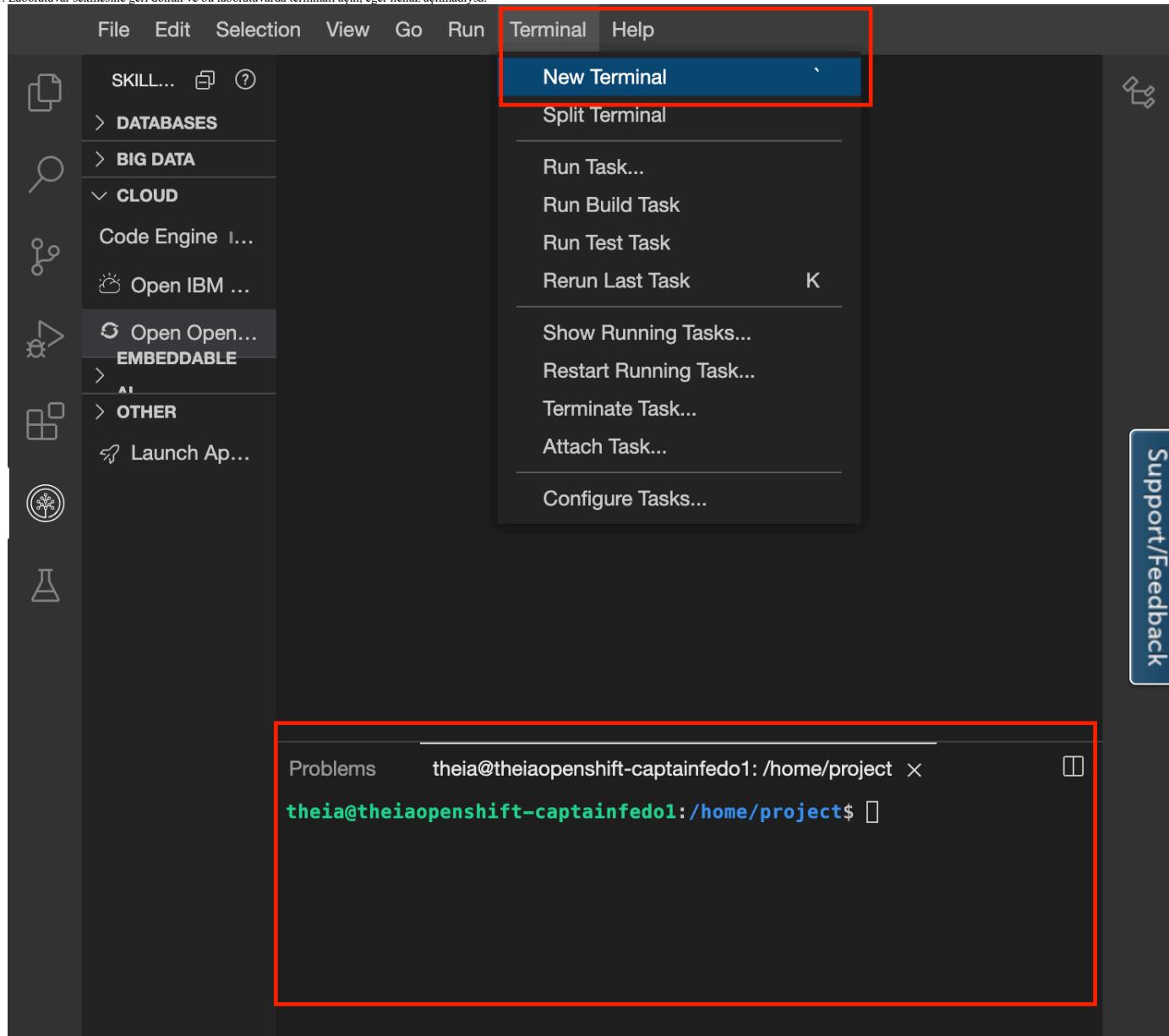


The icon represents your Image in Topology view. A label will also be added to the resource defining the icon.



Mongo'nun bir uygulama olarak çalıştığını gösteren topoloji görüntüsünü ekran görüntüsünü alın ve bunu deploy-getsong-2.png (veya .jpg) olarak kaydedin.

11. Laboratuvar sekmesine geri dönün ve bu laboratuvara terminali açın, eğer henüz açılmadıysa.



Kanıt

1. Image sayfasının doğru Konteyner Görüntüsü olarak mongo ve Etiket olarak latest gösterdiğini gösteren bir ekran görüntüsü alın. Bunu deploy-getsong-1.png (veya .jpg) olarak kaydedin.
2. Mongo'nun bir uygulama olarak çalıştığını gösteren topoloji görünümünün ekran görüntüsünü alın. Bunu deploy-getsong-2.png (veya .jpg) olarak kaydedin.

Opsiyonel Görevler

1. mongo uygulamasını internete açmadık. Ancak, bu sunucuya podun kendisinden erişebilirsiniz. Mongo sürümünü yazdırmak için terminalde aşağıdaki komutu çalıştırın:

```
oc get po
```

Aşağıdaki gibi bir çıktı görmelisiniz:

```
oc get po
NAME           READY   STATUS    RESTARTS   AGE
mongo-79cbc7d7b9-5x5p9   1/1     Running   0          18m
openshift-web-console-78f49566d7-4bs8m  2/2     Running   0          26m
openshift-web-console-78f49566d7-68dkp  2/2     Running   0          26m
```

2. Mongo podu ile aşağıdaki komutu çalıştırın. Podunuza atanan isim burada gösterilenlerden farklı olacaktır:

```
oc exec mongo-79cbc7d7b9-5x5p9 -- mongosh --quiet --eval "db.version()"
```

Bu komut, podunuzda kurulu olan Mongo sürümünü çekerler. Bu, burada gösterilenden farklı olabilir.

```
oc exec mongo-79cbc7d7b9-5x5p9 -- mongosh --quiet --eval "db.version()"  
7.0.8
```

Tebrikler! OpenShift'te MongoDB sunucusunu başarıyla kurdunuz. Ayrıca, `oc exec` komutunu kullanarak sunucuya erişebilirsiniz.

Egzersiz 7: Şarkılar - GitHub Deposu Kopyala

Görevleriniz

Daha önce sağlanan şablondan şarkılar hizmeti için yeni bir depo oluşturduğunuz. Eğer oluşturmadıysanız, [Flask ile Şarkı Servisi Oluşturma](#) laboratuvarına geri dönün ve bu laboratuvarı tamamlandıktan emin olun.

1. Eğer açık değilse, Terminal -> Yeni Terminal ile yeni bir terminal açın.
2. Proje dizinine geçin. Şarkılar hizmetini resim hizmeti dizininde kopyalamak istemiyorsunuz.

```
cd /home/project
```

3. Ardından, GitHub hesabınızın adını içeren bir ortam değişkeni dışa aktarmak için `export GITHUB_ACCOUNT` komutunu kullanın.

Not: Aşağıdaki `{your_github_account}` yer tutucusunu gerçek GitHub hesabınızla değiştirin:

```
export GITHUB_ACCOUNT={your_github_account}
```

4. Değeri kontrol etmek için terminalde değişkeni echo ile kullanabilirsiniz:

```
echo $GITHUB_ACCOUNT
```

5. Sonra, deponuzu kopyalamak için aşağıdaki komutları kullanın.

```
git clone https://github.com/$GITHUB_ACCOUNT/Back-End-Development-Songs.git
```

6. `devops-capstone-project` dizinine geçin ve `./bin/setup.sh` komutunu çalıştırın.

```
cd Back-End-Development-Songs  
bash ./bin/setup.sh  
exit
```

7. Kurulum yürütmesi sonunda aşağıdakileri görmelisiniz:

```
*****  
Capstone Environment Setup Complete  
*****  
  
Use 'exit' to close this terminal and open a new one to initialize the environment  
  
theia@theia-captainfed01:/home/project$
```

Alıştırma 8: Şarkılar - OpenShift'e Dağıtım

Şarkılar mikroservisini OpenShift'e yükleyelim. Bunu yapmak için, OpenShift'i GitHub deposuna yönlendireceksiniz. Eğer kaynak kodda herhangi bir değişiklik yaptıysanız, bunların ana dalmına kaydedildiğinden ve itildiğinden emin olun. Ayrıca, uygulamaya MongoDB sunucusunu nerede bulacağını belirtmeniz gerekecek.

Görevleriniz

1. Aynı OpenShift projesindeki uygulamalar, diğer uygulamalara `servicename.openshift_project.svc.cluster.local` adıyla atıfta bulunabilir. MongoDB Sunucu uygulaması `mongo` olarak adlandırılır. OpenShift projenizi almak için aşağıdaki komutu çalıştırın:

```
oc get project
```

Sonuç şöyle görünebilir:

NAME	DISPLAY NAME	STATUS
sn-labs-captainfedor1		Active

Proje adı sn-labs-captainfedor1'dır. Sizin proje adınız farklı görünebilir. Tam sunucu yolu mongo.sn-labs-captainfedor1.svc.cluster.local olur.

2. OpenShift proje adınızı OPENSHIFT_PROJECT değişkeni olarak aşağıdaki gibi dışa aktarın:

```
export OPENSHIFT_PROJECT=sn-labs-captainfedor1
```

Değeri OpenShift projenizle değiştirin.

3. Uygulamayı RedHat OpenShift'e aşağıdaki komutla itinize:

```
oc new-app https://github.com/${GITHUB_ACCOUNT}/Back-End-Development-Songs --strategy=source --name=songs --env MONGODB_SERVICE=mongo.${OPENSHIFT_PROJECT}.svc.cluster.local --name songs
```

- strategy=source: doğrudan kaynak kodundan inşa eder
- -name: uygulamaya songs adı verir
- -env: mongo servisi için iç adresi ayarlar

Şarkı mikroservisi için oc new-app çıktısının ekran görüntüsünü alın. Ekrana görüntüsünü deploy-getsong-3.jpg (veya .png) olarak kaydedin.

4. Önceki adım uygulama için bir inşa tetikler. Çıktıda belirtildiği gibi, inşa için günlükleri takip etmek için bu komutu kullanabilirsiniz:

```
oc logs -f buildconfig/songs
```

Aşağıdaki gibi bir çıktı görmelisiniz:

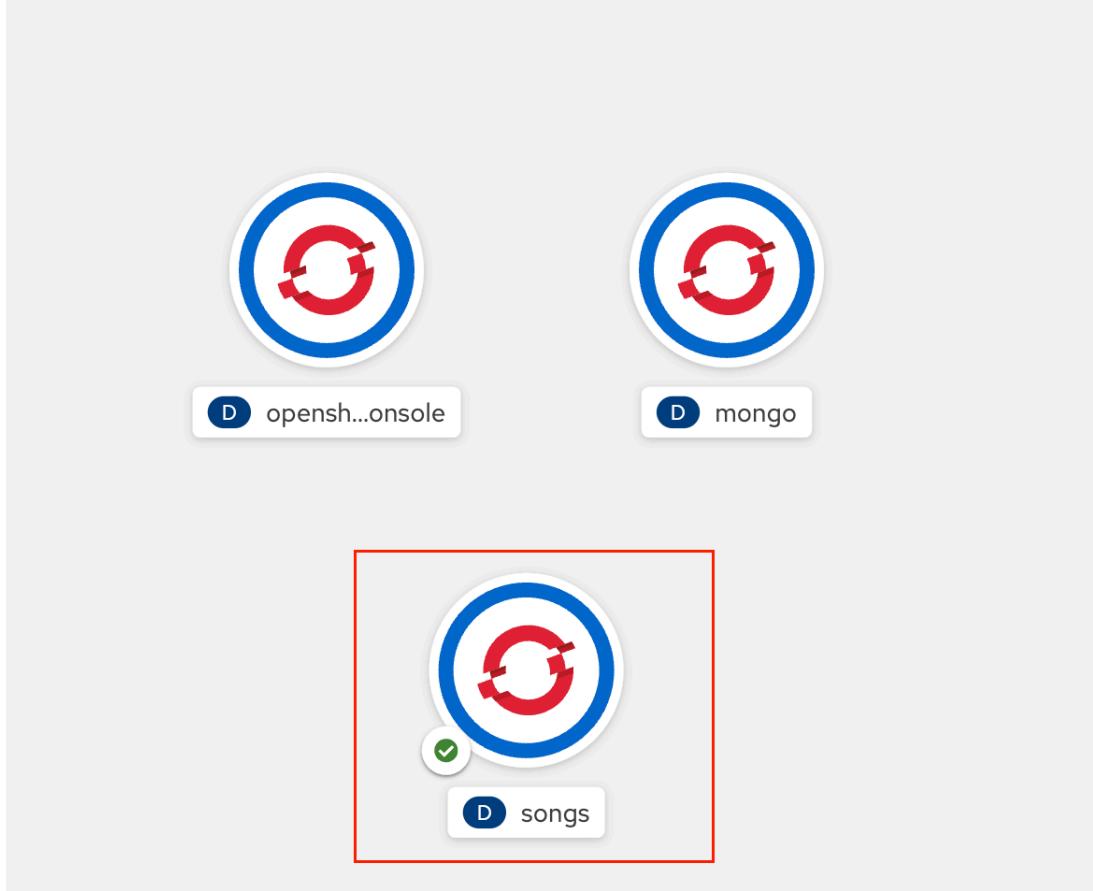
```
Cloning "https://github.com/captainfedoraskillup/private-get-songs" ...
Commit: 4e2d6e08eaef34c97e4ef01a418f90b6e398397c96 (Merge pull request #4 from captainfedoraskillup/openshift)
Author: captainfedoraskillup <111002162@captainfedoraskillup@users.noreply.github.com>
Date:   Wed Feb 15 18:35:16 2023 +0000
time="2023-02-23T02:20:19Z" level=info msg="Not using native diff for overlay, this may cause degraded performance for building images: kernel has CONFIG_OVERLAY_FS_REDIRECT_DIR enabled"
I0223 02:20:19.474218      1 defaults.go:102] Defaulting to storage driver "overlay" with options [mountopt=metacopy=on].
Caching blobs under "/var/cache/blobs".
Trying to pull image-registry.openshift-image-registry.svc:5000/openshift/python@sha256:7885a4366f211adff41a3ab6b3145ca8fd6e8857c8f6b6a30e43ab3e263498c0...
Getting image source signatures
Copying blob sha256:81051d76ef8269de00d9e37d9aa57e88998c05c18d9a970233ffd62785c960
Copying blob sha256:b83366e83795493969367b9608af8a893bc0719b952a3d190a107b1b120d2d
Copying blob sha256:a301327116a0e734fb34506202fafa3d0c94b4ec8f4e7e71a73d78c3a117419f
Copying blob sha256:dae34b4e252ee42452b6bf54b0857049ca4115e4f4db7379b2cb0815017fed8
Copying blob sha256:1593676b86e25227bc08c0e535ec5dde8994eff3b65956654b07739043aa7d51ea
Copying config sha256:7caecf9bcd7e25f1928d716918c363637a4eaee8f404d7b5c3e9ef0258999cfe
Writing manifest to image destination
Storing signatures
```

-f bayrağının günlükleri takip ettiğini unutmeyin. Terminal otomatik olarak çıkmayacaktır. Günlüklerden çıkmak için Ctrl+C kullanabilirsiniz.

5. Günlüklerde Push Successful yazdırında, topolojiye geri dönübilir ve uygulamanın yeşil olmasını bekleyebilirsiniz. Bu birkaç dakika sürebilir.

```
Storing signatures
--> 59b748f50fa
Successfully tagged temp.builder.openshift.io/sn-labs-captainfedor1/songs:1:93ed970d
59b748f50fa@da2b6f31a5a55c9e0d1c4c79dd58f3901cd00dc4eb243fa3d4
Pushing image image-registry.openshift-image-registry.svc:5000/sn-labs- captainfedor1/songs:latest ...
Getting image source signatures
Storing signatures
Successfully pushed image-registry.openshift-image-registry.svc:5000/sn-labs- captainfedor1/songs@sha256:f06d82d7440f550312b948eb51a4fc603c29e0fd7f556f6bb5b09d48113d4323
Push successful
```

6. Uygulama günlüklerini topolojiden songs uygulamasına tıklayarak görüntüleyebilirsiniz:



/ i

D songs

Actions

Health Checks Container songs does not have health checks to ensure your application is running correctly. [Add Health Checks](#)

Details Resources Monitoring

Pods

P songs-9ffb7599-wb9cc	Running	View
-------------------------------------	----------------------	-------------------

Builds

BC songs	Start Build
✓ Build #1 is complete (less than a minute ago)	View

Services



Aşağıdaki gibi bir çıktı görmelisiniz:

Project: sn-labs-captainfedo1 ▾

[Pods](#) > Pod Details

P songs-9ffb7599-wb9cc  Running

[Details](#) [YAML](#) [Environment](#) [Logs](#) [Events](#) [Terminal](#)



Log streaming...



songs ▾

23 lines

```
* Serving Flask app 'backend' (lazy loading)
* Environment: production
  WARNING: This is a development server. Do not use it in a production deployment.
  Use a production WSGI server instead.
* Debug mode: on
* Running on all addresses.
  WARNING: This is a development server. Do not use it in a production deployment.
* Running on http://172.17.16.177:8080/ (Press CTRL+C to quit)
* Restarting with stat
The value of MY_VAR is: mongo.sn-labs-captainfedo1.svc.cluster.local
connecting to url: mongodb://mongo.sn-labs-captainfedo1.svc.cluster.local
20
count: 20
songs application. Uses S2I to build the application.
* Debugger is active!
* Debugger PIN: 338-603-122
```

7. Uygulamaya tıkladığınızda başarılı bir şekilde çalışan bir pod görmelisiniz:

Application: all applications ▼

View sh

source ▼ Find by name... / ⓘ

The screenshot shows the OpenShift application dashboard. In the top left, there's a dropdown menu set to 'source' and a search bar with placeholder text 'Find by name...'. On the right, there's a help icon. Below this, there are three pods listed: 'D mongo' (status: green), 'D openshift-console' (status: green), and 'D songs' (status: green). The 'songs' pod is highlighted with a red border. A modal window for the 'songs' pod is open on the right. The title of the modal is 'D songs'. Inside, there's a 'Health Checks' section with a warning message: 'Container songs does not have health checks to ensure your application is running correctly.' It includes a link to 'Add Health Checks'. Below this are tabs for 'Details', 'Resources' (which is selected and underlined in blue), and 'Monitoring'. Under the 'Pods' heading, the 'songs' pod is listed again with its status as 'Running'.

D songs

Health Checks

Container songs does not have health checks to ensure your application is running correctly. [Add Health Checks](#)

Details Resources Monitoring

Pods

P songs-9ffb7599-wb9cc Running

8. Uygulamayı dışarıdan erişilebilir hale getirmek için açmanız gerekiyor. Laboratuvar IDE'sini çalıştırın sekmeye geri dönün ve yeni bir terminal açın. expose komutunu aşağıdaki gibi kullanın:

```
oc expose service/songs
```

Aşağıdaki çıktıyu görmelisiniz:

```
$ oc expose service/songs
route.route.openshift.io/songs exposed
```

oc expose svc komutunun çıktısının ekran görüntüsünü alın. Ekran görüntüsünü deploy-getsong-4.jpg (veya .png) olarak kaydedin.

9. Uygulamanın URL'sini uygulama açılır menüsünden kopyalayın. Bu URL'yi bir sonraki laboratuvar için ana Django uygulamasını şarkılar mikroservisi ile bağlamak üzere kullanacaksınız.

The screenshot shows the OpenShift web interface for the 'songs' application. It includes:

- Pods**: A table with one row: 'P songs-9ffb7599-wb9cc' status 'Running' with a 'View' link.
- Builds**: A table with one row: 'BC songs' status 'Build #1 is complete (10 minutes ago)' with a 'View' link and a 'Start Bu' button.
- Services**: A table with one row: 'S songs' Service port: 8080-tcp → Pod Port: 8080.
- Routes**: A table with one row: 'RT songs' Location: <http://songs-sn-labs-captainfedo1.labs-prod-openshift-san-a45631dc5778dc6371c67d206ba9ae5c-0000.us-east.containers.appdomain.cloud>.

Bunun yerine terminalde `oc get route` komutunu da kullanabilirsiniz:

```
oc get route songs
```

Aşağıdaki gibi bir çıktı görelisiniz; URL'nin yol sütununda listelendiğini göreceksiniz:

```
$ oc get route songs
NAME      HOST/PORT
songs     songs-sn-labs-captainfedo1.labs-prod-openshift-san-a45631dc5778dc6371c67d206ba9ae5c-0000.us-east.containers.appdomain.cloud
          PATH  SERVICES   PORT    TERMINATION   WILDCARD
          songs      8080-tcp
          None
```

10. Önceki çıktıda URL'yi alıp yeni bir tarayıcıda `http://$URL/health` ile yapıştırın. Aşağıdaki gibi bir sonuç almalısınız:

Not
Eğer Uygulama mevcut değil hatası alıyorsanız, URL'nin `http://` ile başladığından emin olun, çünkü tarayıcınız siz `https://` güvenli sürümüne yönlendirebilir.

```
...
http://songs-sn-labs-captainfedo1.labs-prod-openshift-san-a45631dc5778dc6371c67d206ba9ae5c-0000.us-east.containers.appdomain.cloud/health
{
  "status": "OK"
}..
'/health' uc noktasına erişim çıktısının ekran görüntüsünü alın. Ekran görüntüsünü 'deploy-getsong-5.jpg' (veya .png) olarak kaydedin.
```

Kanıt

- Şarkı mikroservisi için `oc new-app` çıktısının ekran görüntüsünü alm. Ekran görüntüsünü `deploy-getsong-3.jpg` (veya .png) olarak kaydedin.

2. `oc expose svc` komutunun ekran görüntüsünü alın. Ekran görüntüsünü `deploy-getsong-4.jpg` (veya .png) olarak kaydedin.

3. Tarayıcınızdaki `/health` uç noktasıyla çalışan uygulamanın ekran görüntüsünü alın. Ekran görüntüsünü `deploy-getsong-5.jpg` (veya .png) olarak kaydedin.

Eğer bir sonuç alamazsanız veya bir hata ile karşılaşırsanız, bu adımları tekrar takip edebilir veya kurs forumunda yardım isteyebilirsiniz.

Tebrikler! Şarkılar mikroservisini RedHat OpenShift'e başarıyla dağıttınız.

- Her iki servisin de genel URL'lerini kopyaladığınızdan emin olun. Bu, bu capstone projesinin son laboratuvarı için gerecek.
- Lütfen bu laboratuvardan çıkış yapmayın veya kapatmayın. Burada çalışan Code Engine örneği, bu capstone projesinin son laboratuvarında doğru çıktıyalmak için 2 Genel URL'ye ihtiyaç duymaktadır.

Yazar(lar)

Lavanya T S
CF