Mikroservislerin Dağıtımı

Tahmini süre: 90 dakika

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

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

Öğrenme Hedefleri:

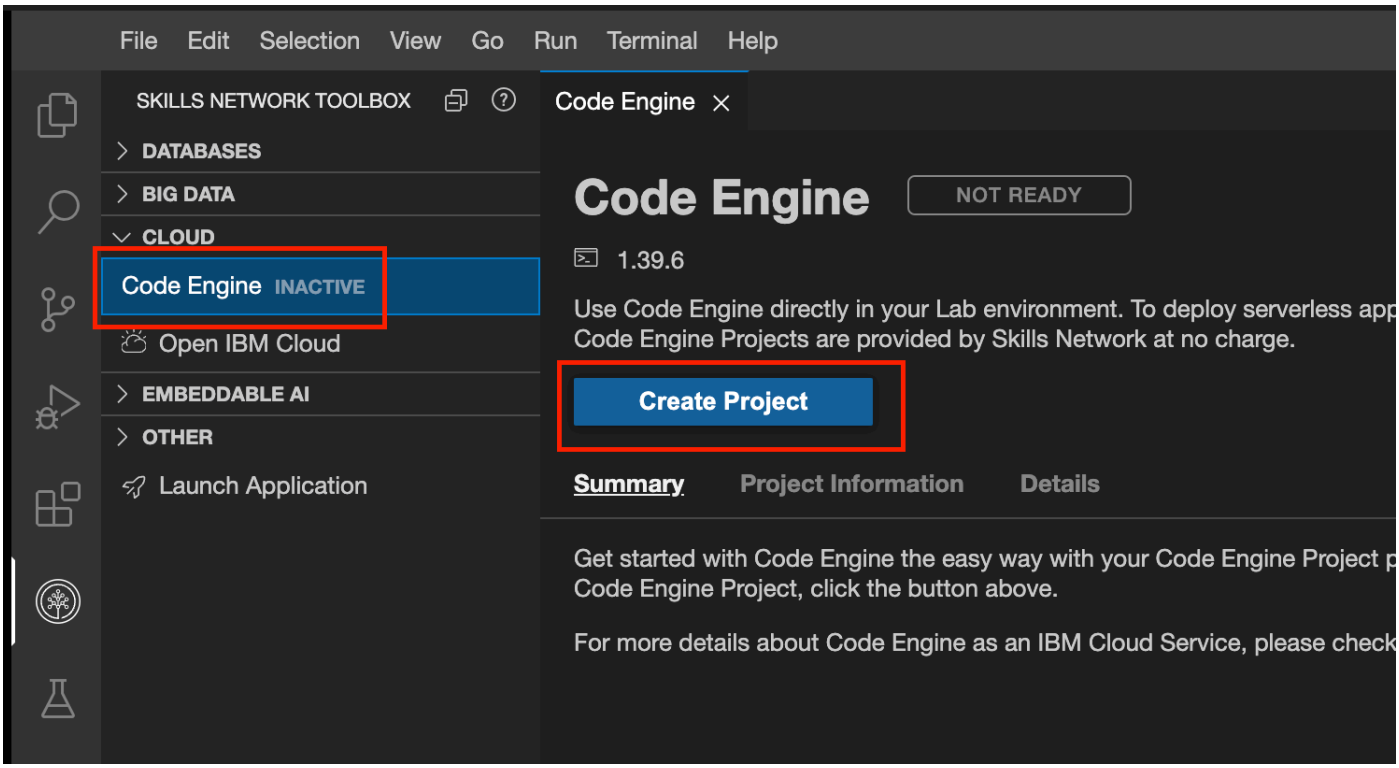
Bu laboratuvarı tamamladıktan sonra şunları yapabileceksiniz:

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

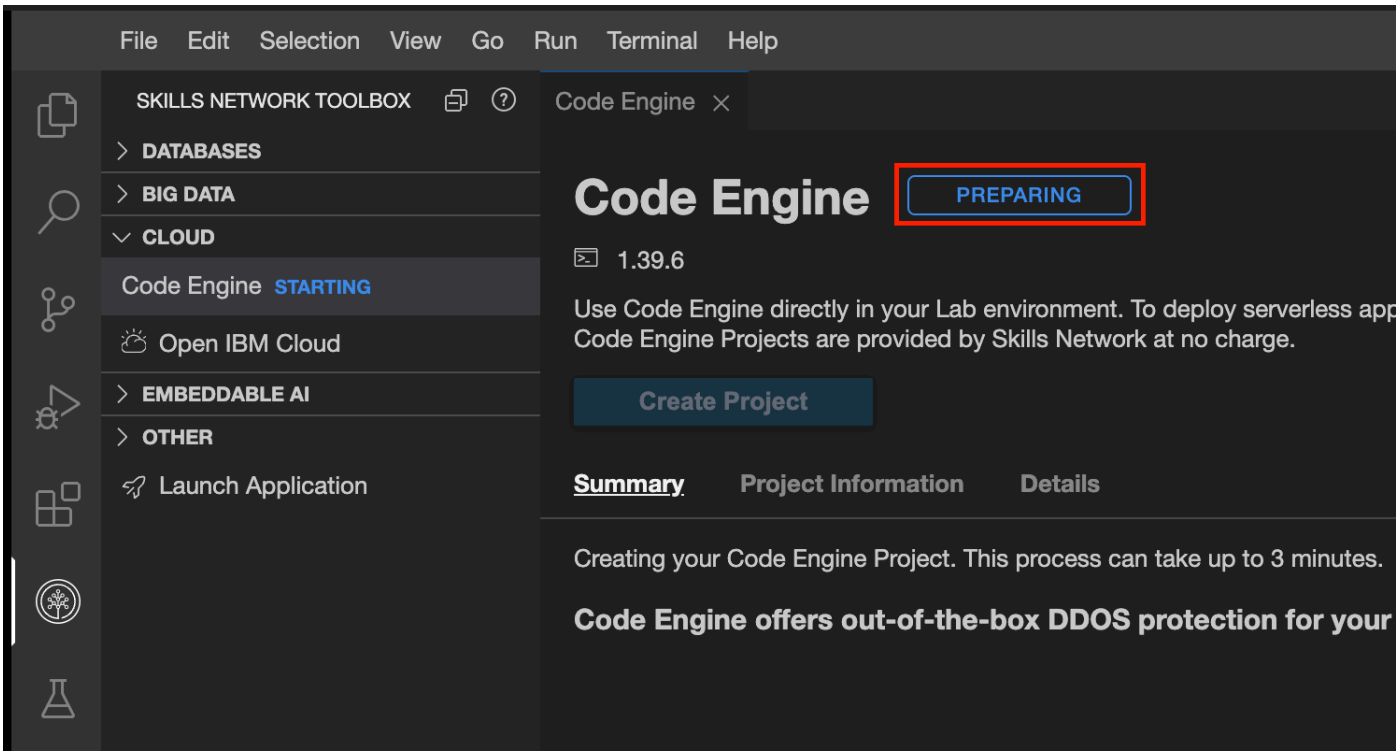
Egzersiz 1: Resimler - Başlangıç Kodu Motoru

Görevleriniz

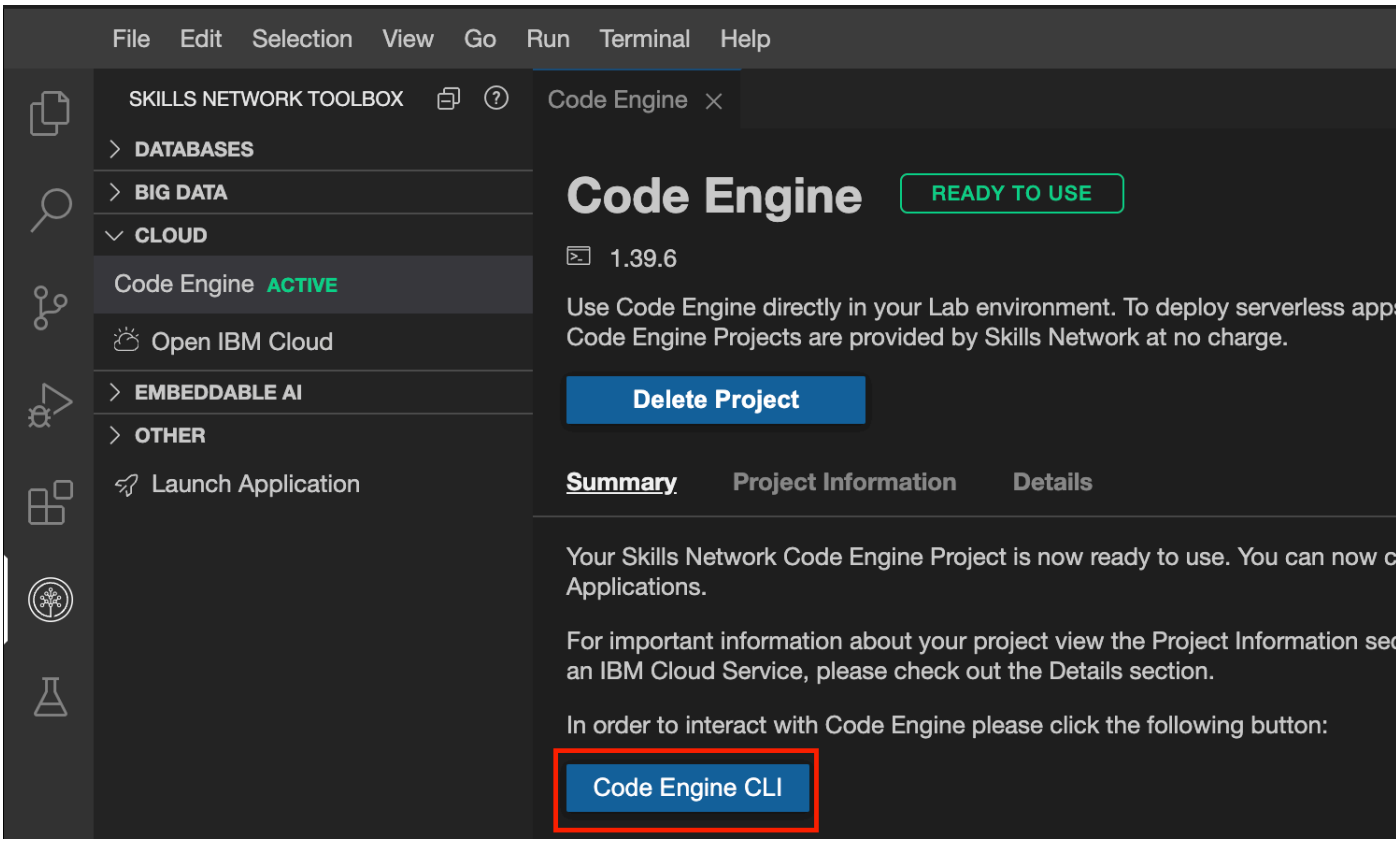
1. Laboratuvar ortamınızdaki menüden `Cloud` açılır menüsüne tıklayın ve `Code Engine` seçeneğini seçin. Kod motoru kurulum paneli görünecektir. Başlamak için `Create Project` butonuna tıklayın.



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

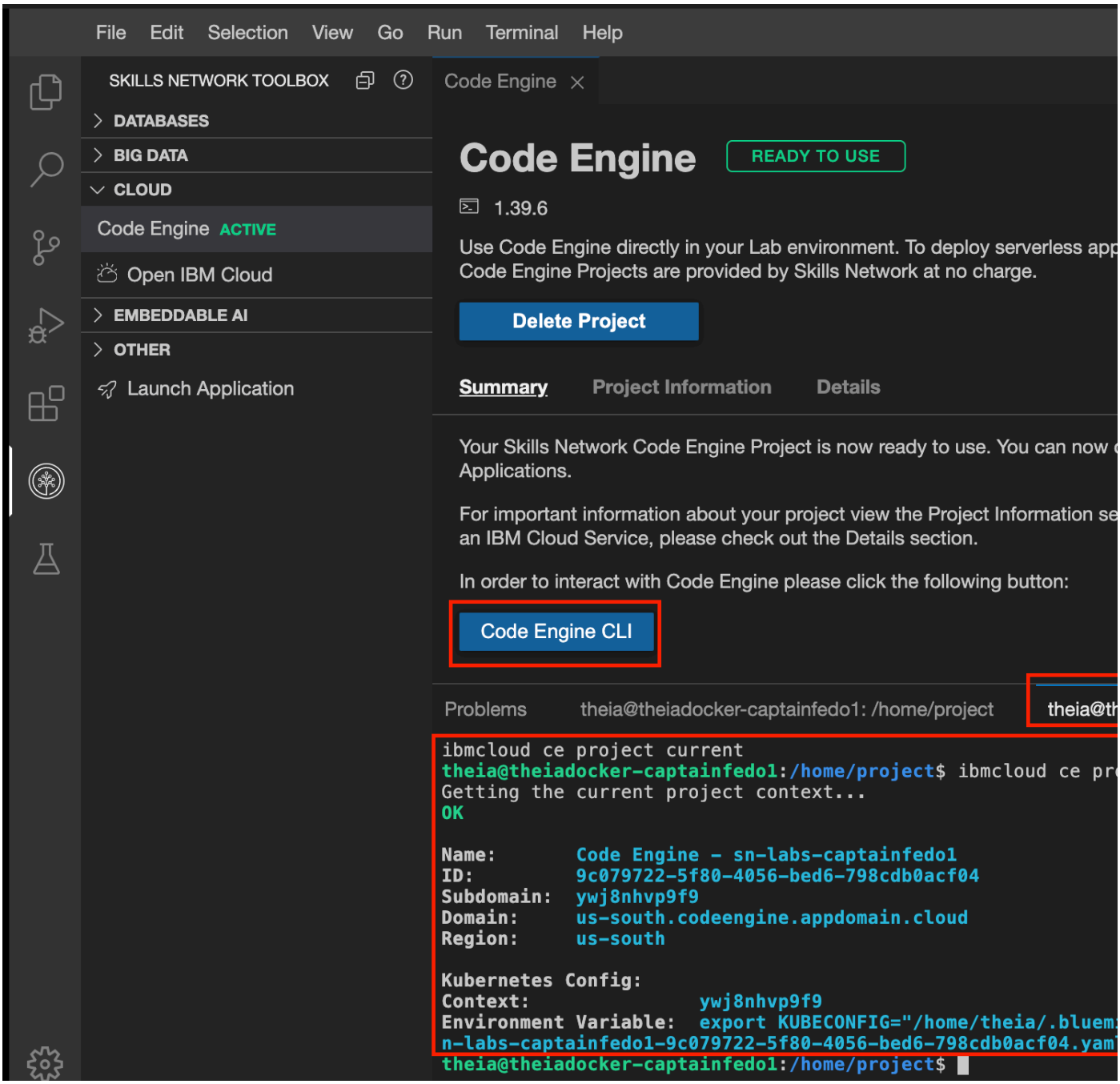


3. Kod motoru kurulumu tamamlandığında, aktif olduğunu göreceksiniz. Aşağıda gösterildiği gibi terminalde önceden yapılandırılmış CLI'yi başlatmak için `Code Engine CLI` butonuna tıklayın.



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

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



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 oluşturdunuz. Eğer oluşturmadıysanız, `Flask ile Resimleri Alma Servisi Oluştur` laboratuvarına geri dönün ve bu laboratuvarı 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 içerip içermediğini kontrol edin. Değişiklikleri görmüyorsanız, önceki laboratuvarı tekrar yapmanız ve Openshift konteynerinde dağıtımınızı yeniden oluşturmanız gerekir.

Open **routes.py** in IDE

Egzersiz 3: Resimler - Uygulamayı yerel olarak çalıştır

Görevleriniz

Artık tüm kodu yerel laboratuvar ortamında bulduğunuza göre, uygulamayı yerel olarak çalıştırarak beklendiği gibi çalıştığından emin olalım. Önceki laboratuvarda mikroservisi tamamlamış olmalısınız.

Görev 1: Uygulamayı yerel olarak çalıştır

1. Proje dizininize aşağıdaki gibi geçin.

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

2. Aşağıdaki komutu çalıştırarak yerel olarak bir kez test edin.

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

3. 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 beklendiği gibi çalıştığından emin olduktan sonra, bir sonraki adım uygulamayı Code Engine'a dağıtmaktır. Bu, zaten sağlanmış olan Dockerfile'ı gerektirecektir. Öncelikle uygulamanın bir görüntüsünü oluşturacak ve bunu hesabınız altında sağlanan laboratuvar görüntü kayıt defterine iteceksiniz.

Görevleriniz

1. 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 çıktıyı görmelisiniz:

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

2. Aşağıdaki komut ile görüntüyü oluşturun:

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

3. 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 görüntüleri görmek için:

```
docker images
```

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

```
$ docker images
REPOSITORY                                TAG           IMAGE ID       CREATED         SIZE
pictures                                  latest        7c230d5bcdbf   2 minutes ago   179MB
us.icr.io/sn-labs-captainfedo1/pictures   1             7c230d5bcdbf   2 minutes ago   179MB
```

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

5. Aşağıdaki komutu kullanarak görüntüyü kayıt defterine itin:

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

6. Aşağıdaki komutu kullanarak ad alanınızdaki tüm görüntüleri listeleyebilirsiniz.

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

Yeni itilen görüntüyü görmelisiniz:

```
$ ibmcloud cr images --restrict $SN_ICR_NAMESPACE
Listing images...
Repository                                Tag   Digest         Namespace              Created         Size    Security status
us.icr.io/sn-labs-captainfedo1/pictures   1     f3c8e1ef47f4   sn-labs-captainfedo1   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 sürer 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 bağlantıyı 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

1. 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.
2. Tarayıcınızda `/count` uç noktasındaki çalışan uygulamanın bir ekran görüntüsünü alın. Ekran görüntüsünü `deploy-getpic-3.png` (veya .jpg) olarak kaydedin.
3. 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 çalıştırdığınız için tebrikler. Bu son adım, uygulamayı terminal kullanarak doğrulayacaktır.

Görevleriniz

1. Uygulamanın detaylarını almak için aşağıdaki komutu çalıştırın:

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

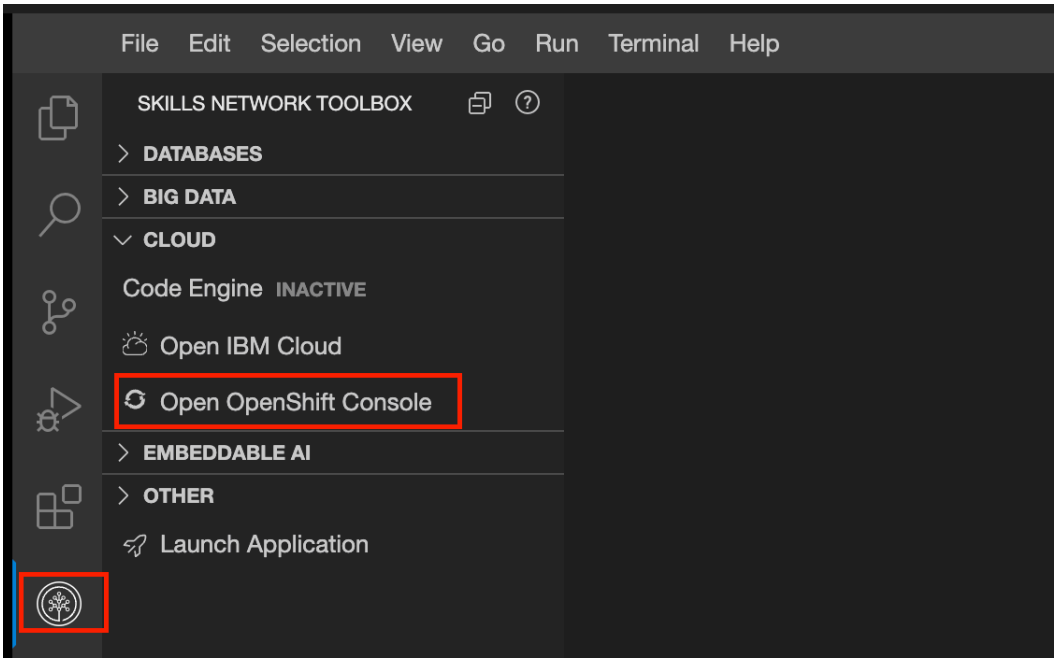
Uygulamanın oluşturulma zamanı, kullanılan kaynaklar, örnek sayısı ve diğer detaylar dahil olmak üzere ayrıntılı bilgileri görmelisiniz.

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

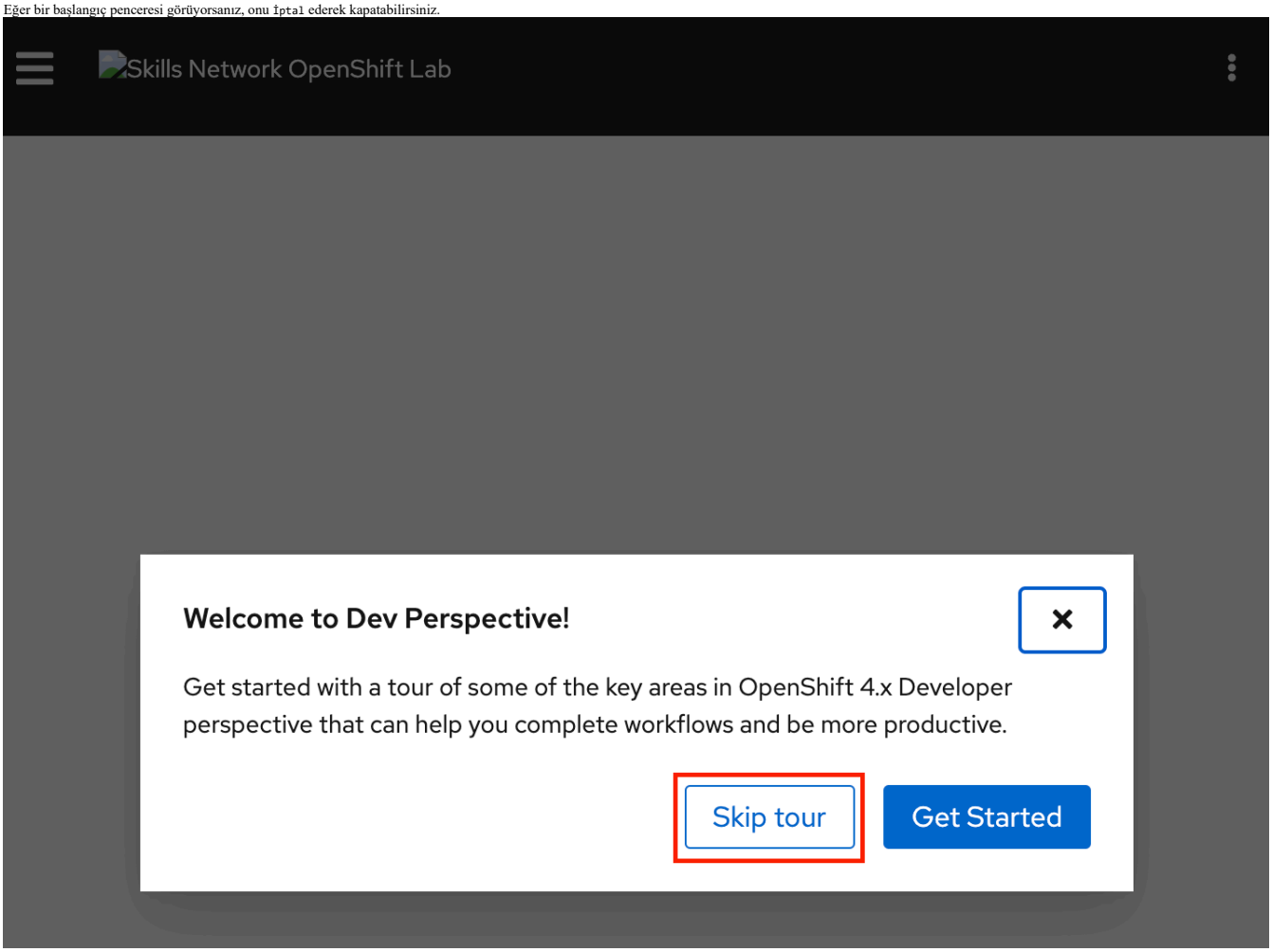
Önceki modüllerdeki alıştırmalar için laboratuvar ortamında sağlanan MongoDb sunucusunu kullandınız. Bu sunucu yalnızca laboratuvar ortamı ile sınırlıdır ve RedHat OpenShift gibi başka bir platformdan erişilemez. MongoDb'yi üretim ortamında veritabanı sunucusu olarak başarıyla kullanabilmek için bu alıştırmada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

Görevleriniz

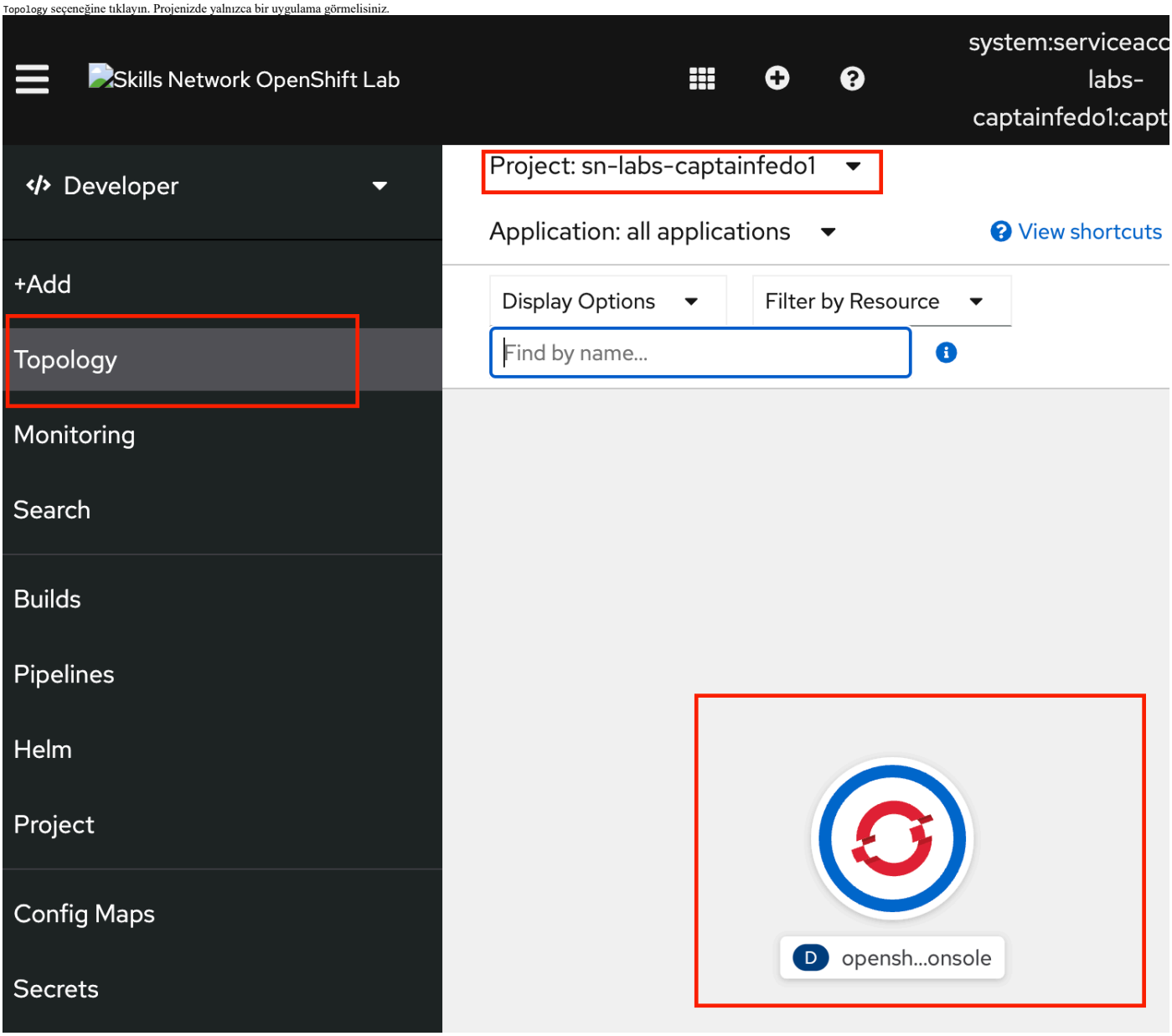
1. Laboratuvar ortamından OpenShift konsolunu başlatın. Bu, yeni bir sekme açacaktır.



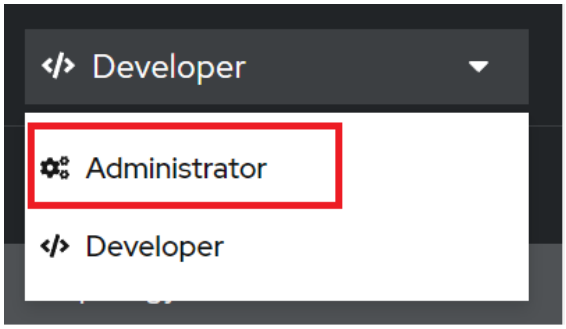
Eğer bir başlangıç penceresi görüyorsanız, onu `İptal` ederek kapatabilirsiniz.



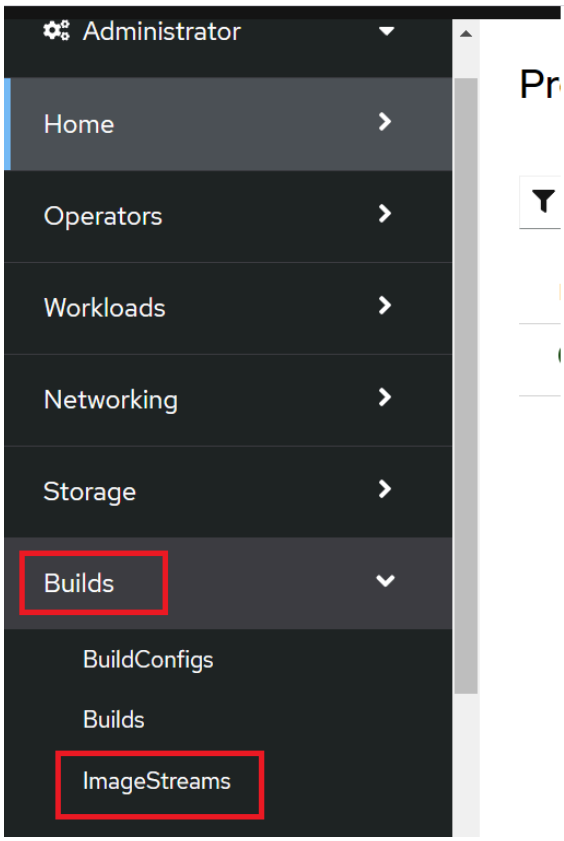
2. Topology seçeneğine tıklayın. Projenizde yalnızca bir uygulama görmelisiniz.



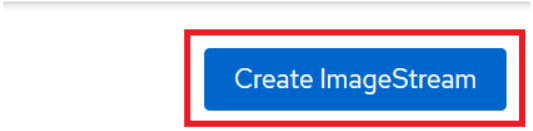
3. Sol menüden **Yönetici** seçeneğine geçin.



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

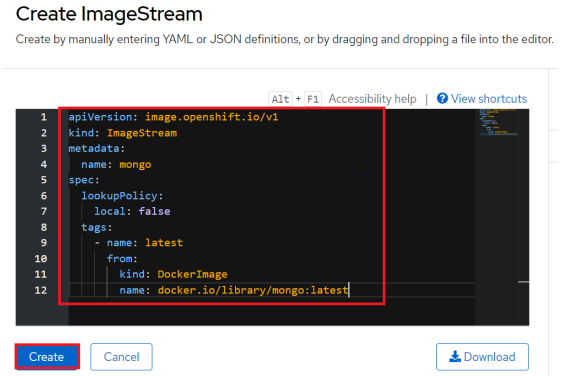


5. **Create ImageStream** butonuna tıklayarak bir ImageStream oluşturma seçeneğini seçin.

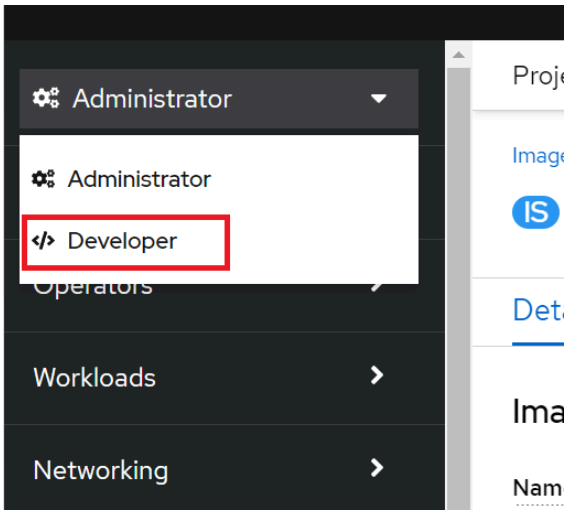


6. Sağlanan kodu kopyalayıp editöre yapıştırın, ardından oluşturma seçeneğini seçin.

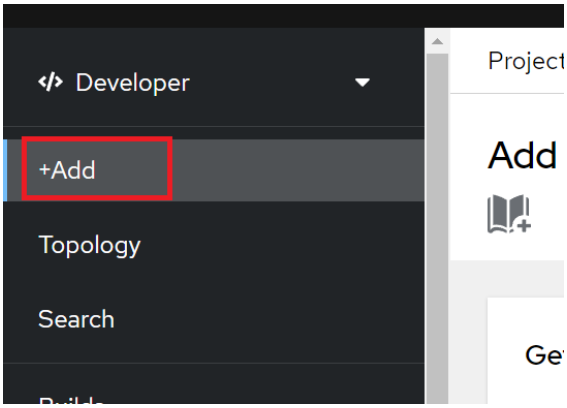
```
apiVersion: image.openshift.io/v1
kind: ImageStream
metadata:
  name: mongo
spec:
  lookupPolicy:
    local: false
  tags:
    - name: latest
      from:
        kind: DockerImage
        name: docker.io/library/mongo:latest
```



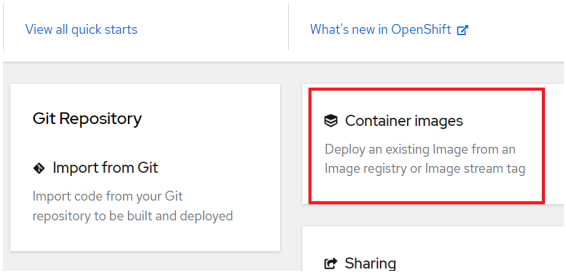
6. Sonraki adımda, sol menüdeki mevcut seçenekleri kullanarak **Geliştirici** moduna geçin.



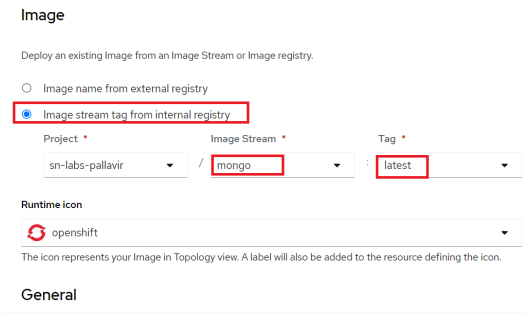
7. **“Ekle+”** seçeneğine tıklayarak seçin.



8. Önceki adımda oluşturulan görüntünün dağıtımını başlatmak için **“Konteyner görüntüleri”** seçeneğini seçin.

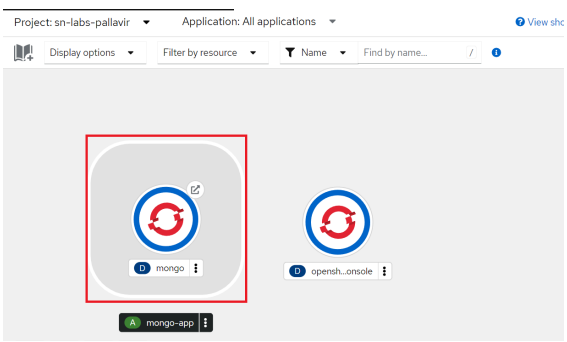


9. **“İç kayıt defterinden görüntü akışı etiketi”** seçeneğini seçin, ardından görüntü akışında **“mongo”** ve etiket olarak **“latest”** seçin. Kalan seçenekleri değiştirmeden bırakın ve **“oluştur”** butonuna tıklayarak devam edin.



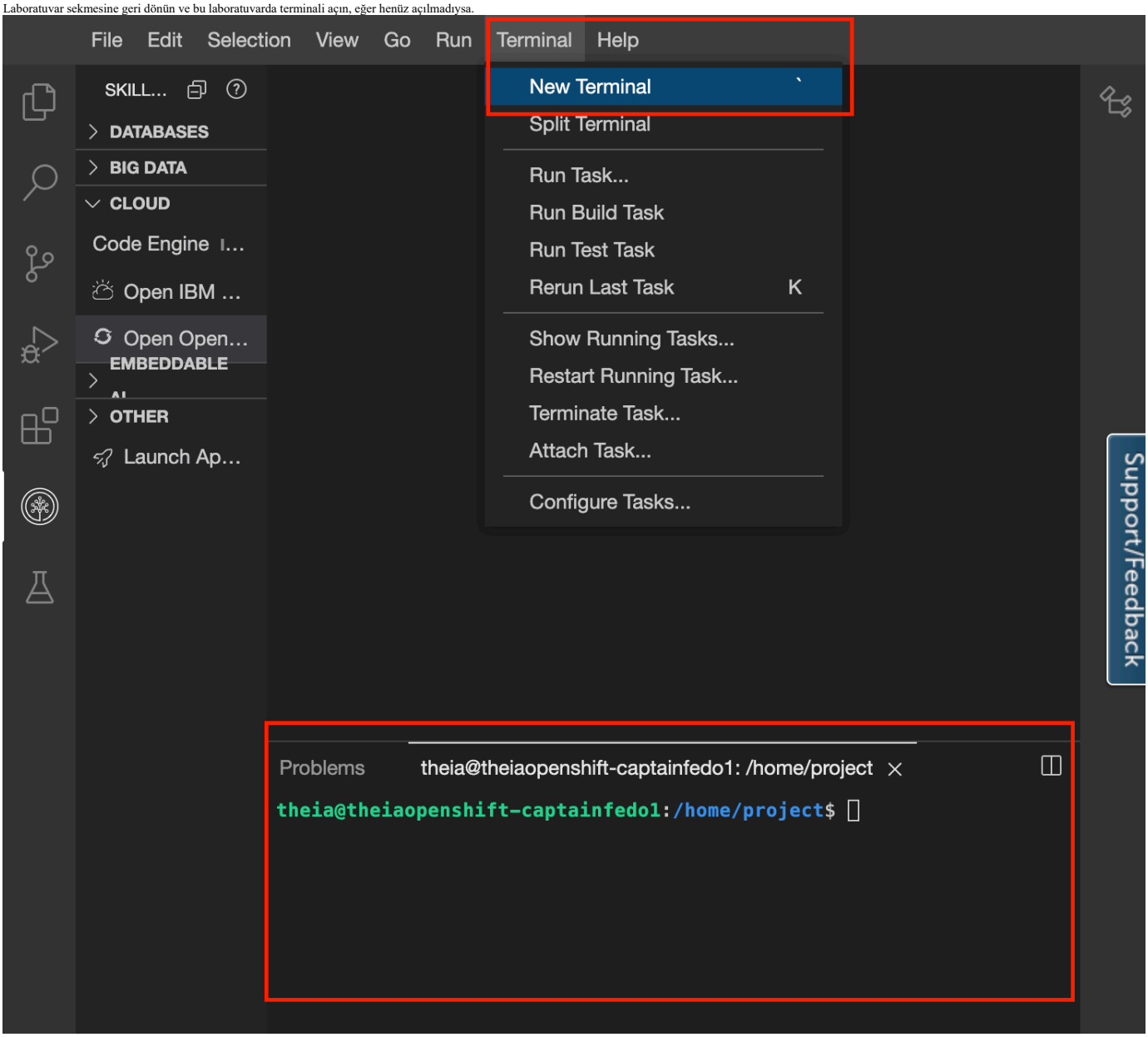
`Image` sayfasının doğru Konteyner Görüntüsü olarak `mongo` ve Etiket olarak `latest` gösterdiğini gösteren bir ekran görüntüsü alın. Bunu `deploy-getsong-1.png` (veya .jpg) olarak kaydedin.

10. OpenShift platformunda MongoDB sunucusunun dağıtımını gözlemlemek için Topoloji sayfasına gidin.



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

11. Laboratuvar sekmesine geri dönün ve bu laboratuvarda 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 çıktılar. 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 oluşturdunuz. Eğer oluşturmadıysanız, `Flask ile Şarkı Servisi Oluşturma` laboratuvarına geri dönün ve bu laboratuvarı tamamladığınızdan 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:

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 dalınıza kaydedildiğinden ve itildiğinden emin olun. Ayrıca, uygulamaya Mongo 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-captainfedo1                 Active
```

Proje adı `sn-labs-captainfedo1`'dir. Sizin proje adınız farklı görünebilir. Tam sunucu yolu `mongo.sn-labs-captainfedo1.svc.cluster.local` olur.

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

```
export OPENSHIFT_PROJECT=sn-labs-captainfedo1
```

Değeri OpenShift projenizle değiştirin.

3. Uygulamayı RedHat OpenShift'e aşağıdaki komutla itiniz:

```
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. Ekran 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:

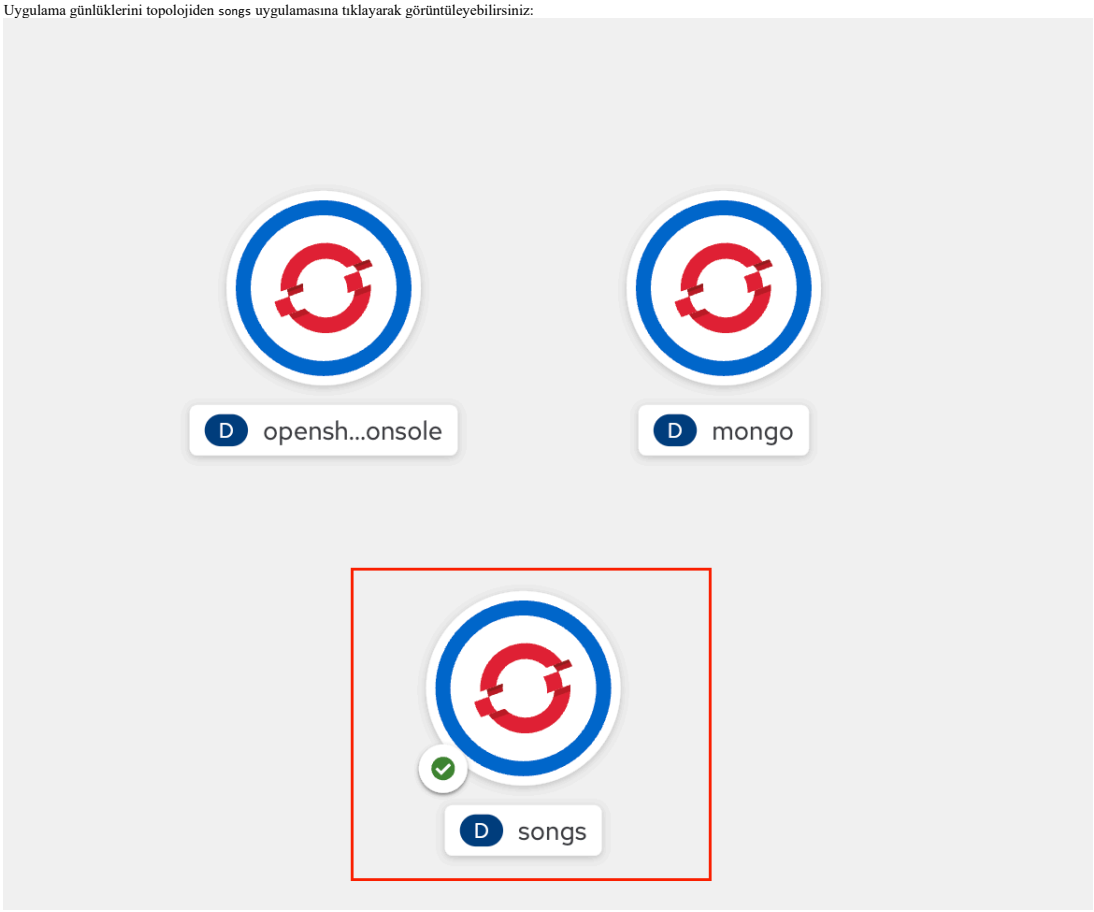
```
Cloning "https://github.com/captainfedoraskillup/private-get-songs" ...
    Commit: 4e2d6e08eaf34c97e4f01a418f90b6e398397c96 (Merge pull request #4 from captainfedoraskillup/openshift)
    Author: captainfedoraskillup <111002162+captainfedoraskillup@users.noreply.github.com>
    Date:   Wed Feb 15 18:35:16 2023 -0800
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 sha256:81051d76ef8269de00d9e37d9aa57e88998c055c18d9a970233ffdb62785c960
Copying blob sha256:bf83366e837954939693 67b9608afd8a893bc0719b952a3d190a107bb1b20d2d
Copying blob sha256:a30132716a0e734fb34506202fafa3d0c94b4ec8f4e7e71a73d78c3a117419f
Copying blob sha256:dae34b4e252ee42452b6bf54b0857049ca4115ef4dba7379e2cbd0815017fed8
Copying blob sha256:1593676b86e25227bc00ce535ec5dde8994eff3b65956654b7739043aa7d51ea
Copying config sha256:7caecf9bdc7e25f1928d716918c363637a4eaee8f404d7b5c3e9ef0258999cfe
Writing manifest to image destination
Storing signatures
```

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

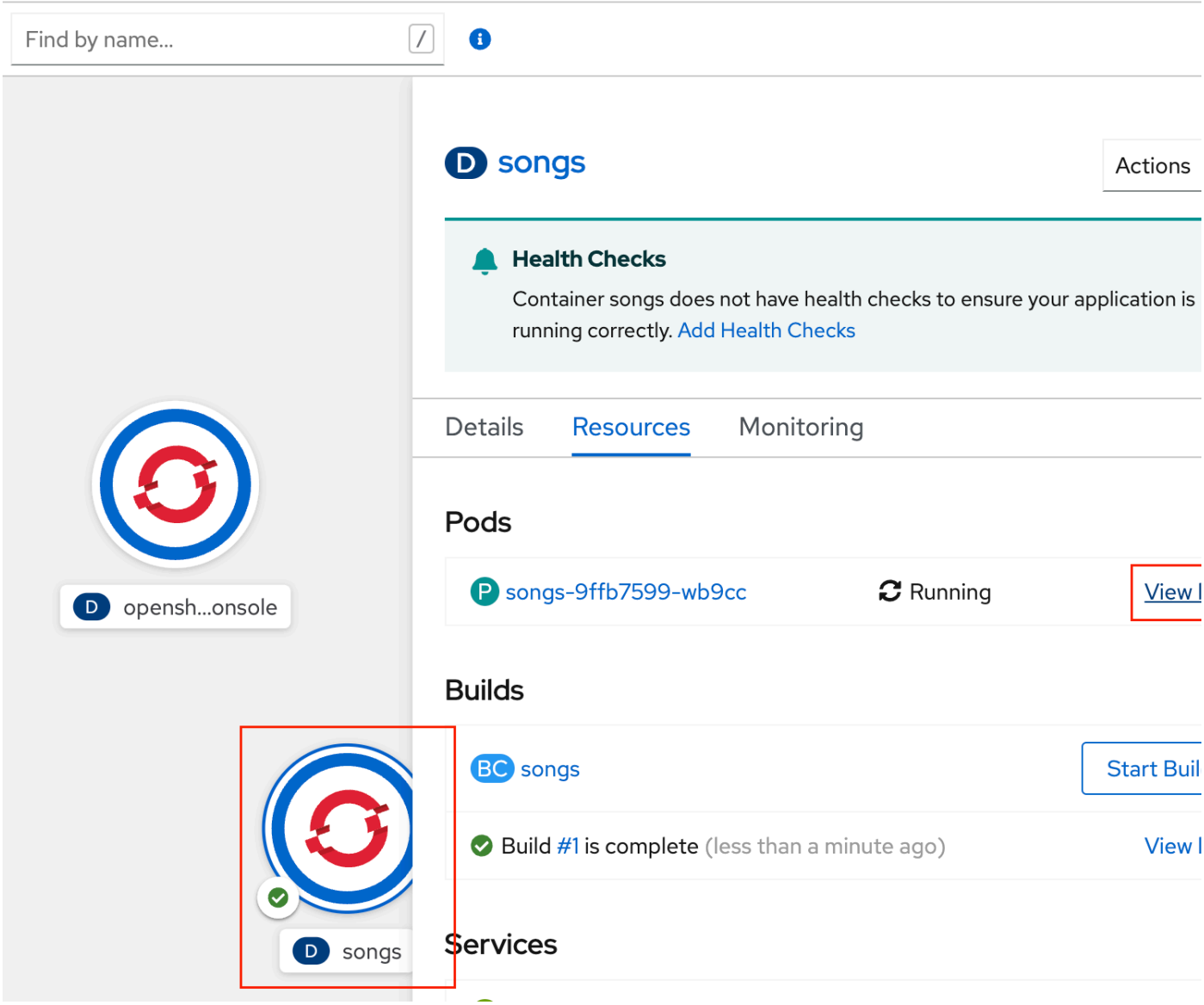
5. Günlüklerde `Push Successful` yazdığında, topolojiye geri dönebilir ve uygulamanın yeşil olmasını bekleyebilirsiniz. Bu birkaç dakika sürebilir.

```
Storing signatures
--> 59b748f50fa
Successfully tagged temp.builder.openshift.io/sn-labs-captainfedo1/songs-1:93ed970d
59b748f50fada2b6f31a5a55c9e0d1cc4c79ddd58f3901ccda0dc4eb243fa3d4
Pushing image image-registry.openshift-image-registry.svc:5000/sn-labs-         captainfedo1/songs:latest ...
Getting image source signatures
Storing signatures
Successfully pushed image-registry.openshift-image-registry.svc:5000/sn-labs-   captainfedo1/songs@sha256:f06d82d7440f550312b948eb51a4fc603c29e0fd7f556f6bb5b09d48113d4323
Push successful
```

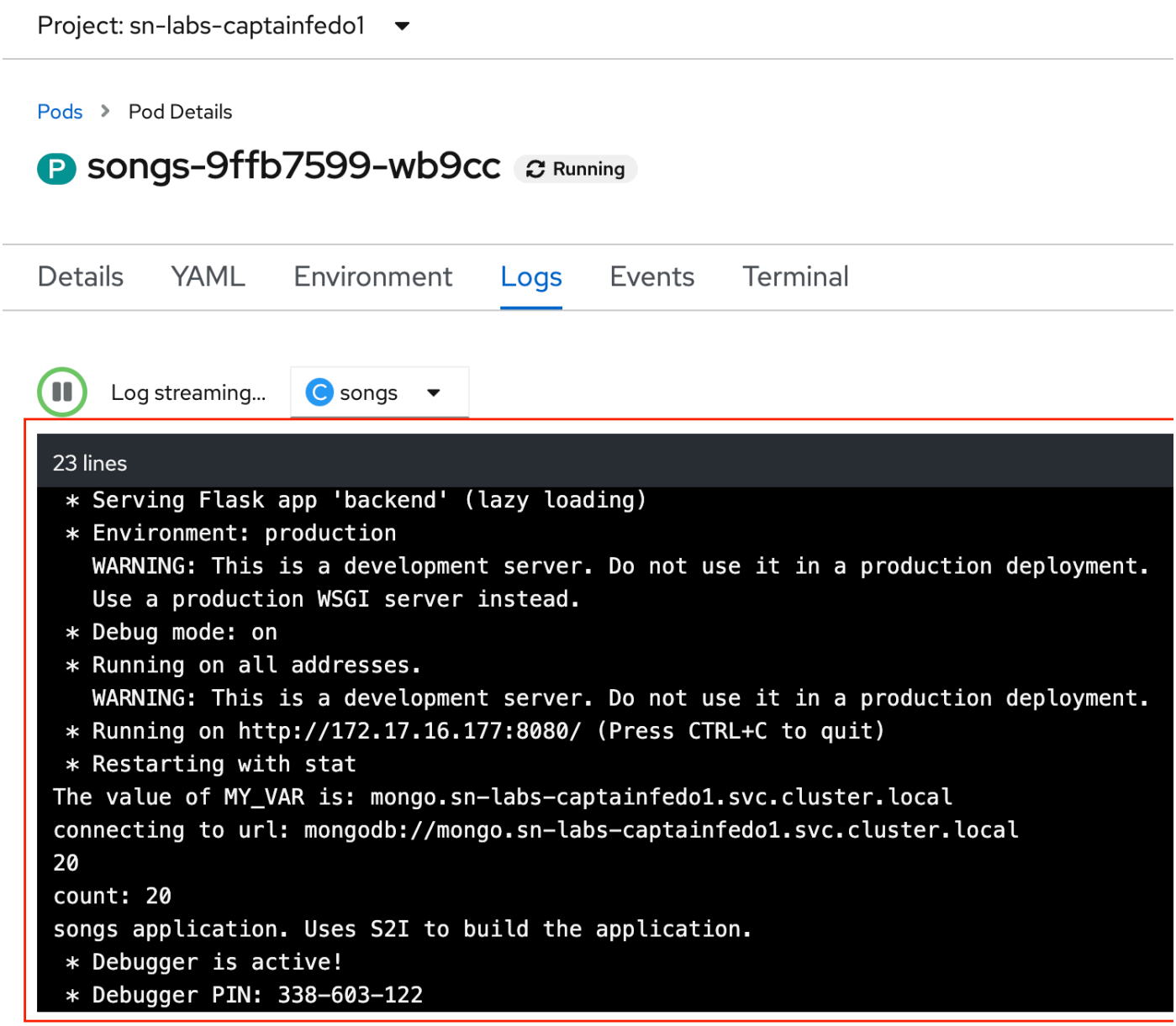
6. Uygulama günlüklerini topolojiden songs uygulamasına tıklayarak görüntüleyebilirsiniz:



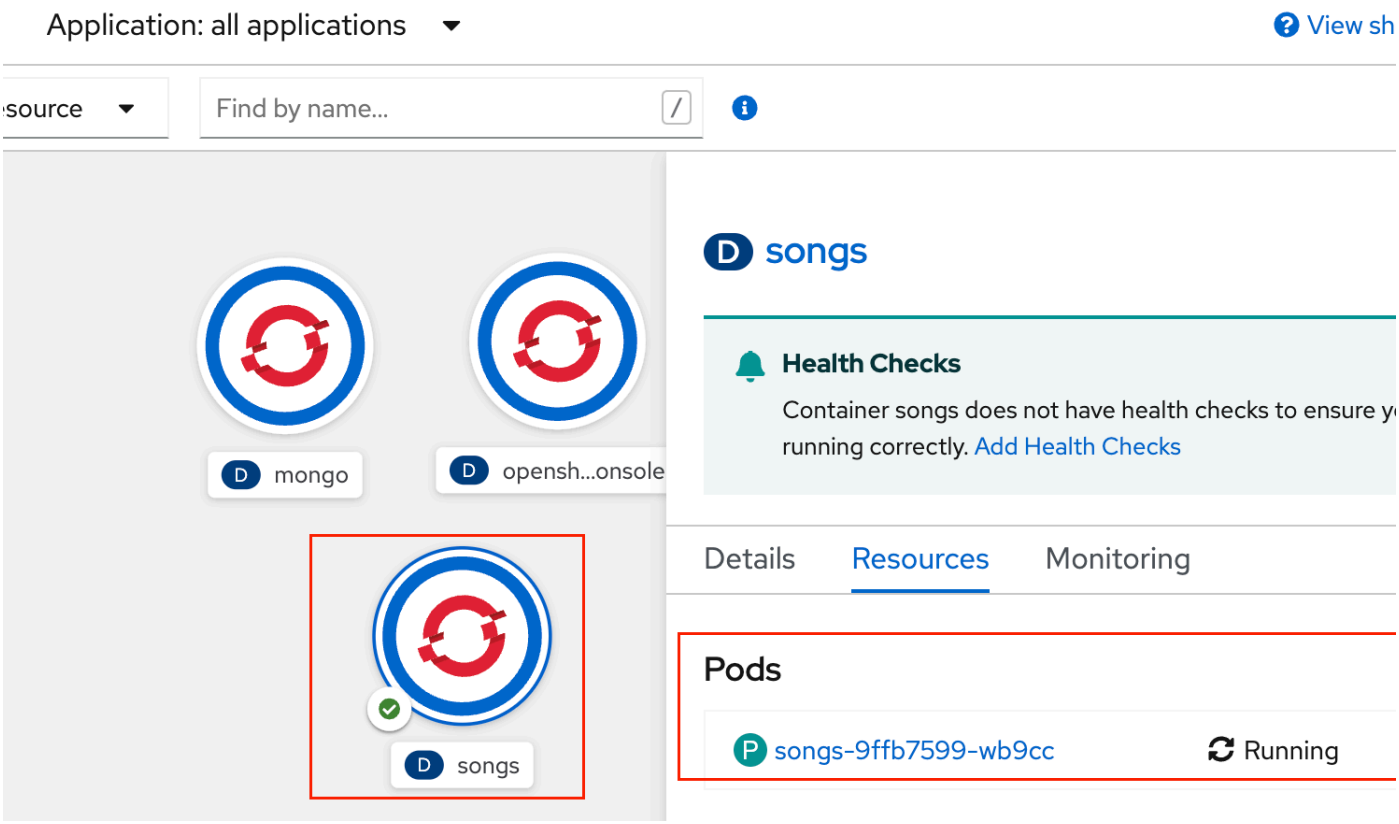
Günlükleri görmek için **Günlükleri Görüntüle**'ye tıklayın:



Aşağıdaki gibi bir çıktı görmelisiniz:



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



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

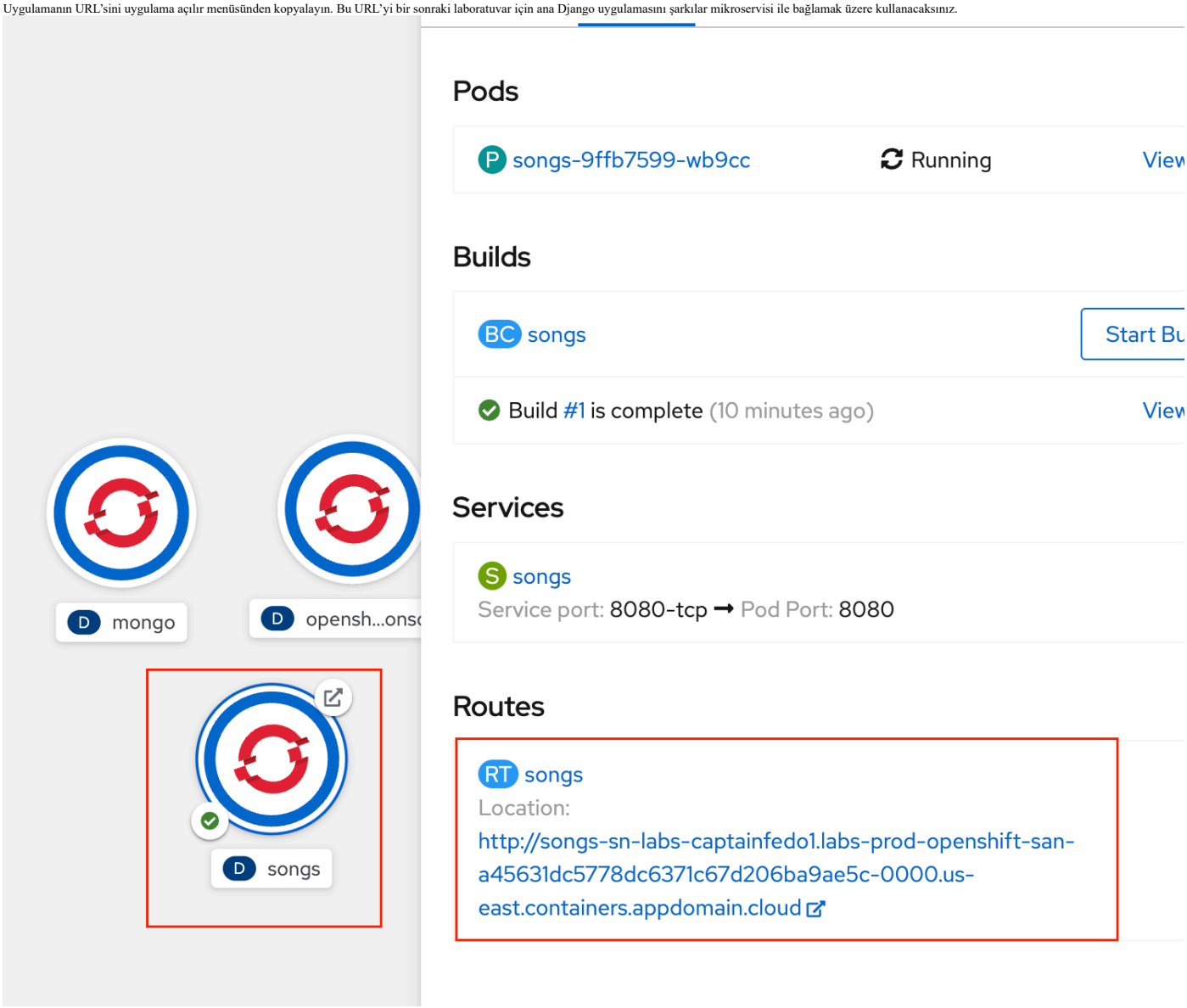
```
oc expose service/songs
```

Aşağıdaki çıktıyı 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.



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

```
oc get route songs
```

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

```
$ oc get route songs
NAME     HOST/PORT                                                                                                                PATH   SERVICES   PORT       TERMINATION   WILDCARD
songs    songs-sn-labs-captainfedo1.labs-prod-openshift-san-a45631dc5778dc6371c67d206ba9ae5c-0000.us-east.containers.appdomain.cloud          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 sizi `https://` güvenli sürümüne yönlendirebilir.

```
 http://songs-sn-labs-captainfedo1.labs-prod-openshift-san-a45631dc5778dc6371c67d206ba9ae5c-0000.us-east.containers.appdomain.cloud/health
{
  "status": "OK"
}
```

`/health` uç noktasına erişim çıktısının ekran görüntüsünü alın. Ekran görüntüsünü `deploy-getsong-5.jpg` (veya .png) olarak kaydedin.

Kanıt

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

2. `oc expose svc` komutunun çıktısının 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 gerekecek.
- Lütfen bu laboratuvardan çıkış yapmayın veya kapatmayın. Burada çalışan Code Engine örneği, bu capstone projesinin son laboratuvarında doğru çıktıyı almak için 2 Genel URL'ye ihtiyaç duymaktadır.

Yazar(lar)

Lavanya T S
CF