INVENTORYMANAGEMENTSYSTEMFORRETAILERS

Domain:CloudApplicationDevelopment

TeamID	PNT2022TMID31378
ProjectName	InventoryManagementSystemforRetailers

DeployingtoKubernetesonIBMCloudOverview

Overview:

Thistutorialshowshowtodeployalookbackontokubernetesontheibmcloud.

Prerequisite:

You'llneedthefollowing:
Node.js 10 or
higherDocker18.06orhigher
Signup for an IBM Cloud Account if you don'thavd one
alreadyIBMCloudCLI,Containerregistry CLI,etc

1. KubernetesCLI(kubectl)
2. LoopBack4CLI

Let'sinstalltheLoopBack4CLI:

```
npm i -g @loopback/cli
```

Step1:ScaffoldLoopBack4app

Runthelb4appcommand,andspecifyallthevaluesprovided below.

```
$lb4app

?Projectname:lb4-simple-web-app

?Projectdescription:lb4-simple-web-app

?Projectrootdirectory:lb4-simple-web-app

?Applicationclassname:Lb4SimpleWebAppApplication

? Select features to enable in the project (Press <space> to select, <a> to toggleall,<i>toinvertselection)

◉ Enableeslint:addalinterwithpre-configuredlintrules

◉ Enableprettier:installprettiertoformatcodeconformingtorules

◉ Enablemocha:installmochatoruntests

◉ EnableloopbackBuild:use@loopback/buildhelpers(e.g.lb-eslint)

◉ Enablevscode:addVSCodeconfigfiles

◉ Enabledocker:includeDockerfileand.dockerignore

◉ Enablerepositories:includerepositoryimportsandRepositoryMixin

◉ Enableservices:includeservice-proxyimportsandServiceMixin(Moveupanddownto revealmore choices)
```

Thelb4-simple-web-appprojectiscreated.

Navigatetothemaindirectoryoftheproject

```
cdlb4-simple-web-app
```

Step2:Runtheapplicationlocally

Inacommandwindowinthemain directory ofyourproject,type:

```
npmstart
```

Theapplicationwillbuild,andthentheserversheuldstartupsuccessfullyanddisplay

```
Serverisrunningathttp://[::1]:3000
```

```
Tryhttp://[::1]:3000/ping
```

Openyourbrowserandattempttoaccessalltheseurl

http://[::1]:3000/http://[::1]:3000/pinghttp://[::1]:3000/explorerhttp://[::1]:3000/openapi.json

Makesurethattheapplicationrunswellbeforecontinuingtothenextstep.Inthecommandwindow,stoptheapplicatio

nwith

```
Ctrl+C
```

Step3:BuildaDockerimage

ReviewthetwoDocker-relatedfilesthathavebeenconveniently provided,.dockerignoreandDockerfile,butleavethem unchanged forthistutorial.NoticetheHOSTandPORTenvironmentvariablevalues:

```
ENVHOST=0.0.0.0PORT=3000
```

Inthepackage.jsonfile,adocker:buildcommandhasbeenprovided.

```
"docker:build":"dockerbuild-tlb4-simple-web-app."
```

Runthecommand:

```
npmrundocker:build
```

Whenitcompletes,youwillsee:

```
Successfullybuilt7d26df6e1561

Successfullytaggedlb4-simple-web-app:latest
```

Youcanfindyourimagebytyping:

```
dockerimages|greplb4-simple-web-app
```

Itwilldisplaysomethinglikethis:

```
lb4-simple-web-app                  latest7d26df6e1561
```

Step4:Runtheapplication inDocker

Inthe`package.json`file,a`docker:run`commandhasbeen provided.

```
"docker:run":"dockerrun-p3000:3000-dlb4-simple-web-app"
```

Runthecommand:

```
npmrundocker:run
```

Afterwards,type:

```
dockerps
```

Youshouldseesomethinglikethis:

```
CONTAINERID        IMAGE                  COMMAND                  CREATED                  STATUS
        PORTS

a9962339e863       lb4-simple-web-app     "node."                  8secondsago              Up7seco
nds     0.0.0.0:3000->3000/tcp
```

Toseethelogoutputofyourcontainer,youcantype:Youshouldseesomethinglike:

```
Serverisrunningathttp://127.0.0.1:3000

Tryhttp://127.0.0.1:3000/ping
```

Openyourbrowserandattempttoaccessallthesewrls

http://127.0.0.1:3000/ http://127.0.0.1:3000/

pinghttp://127.0.0.1:3000/explorerhttp://127.

0.0.1:3000/openapi.json

Step5:StoptheapplicationrunninginDocker

Findthecontainerid

```
dockerps|greplb4
```

Youshouldseesomethinglikethis:

```
a9962339e863                    lb4-simple-web-app                    "node."
```

Theleftmostvalueisthecontainerid.Type:

```
dockerstop<containerid>                    Forexample:a9962339e863
```

Step6:LogintoIBMCloudusingibmcloudlogincommand

Useibmcloudlogincommandtologin.

Afteryou'vebeen successfullyloggedin,you'llseesomethinglike:

```
API endpoint:      https://api.ng.bluemix.net
Region:            us-south
User:              dremond@ca.ibm.com
Account:           DominiqueEmond'sAccount
Resourcegroup:     default
CF API endpoint:
Org:
Space:
```

Step7:LogintoIBMCloudContainer Registry

```
ibmclouderlogin
```

Youshouldsee:

```
Logging in to 'registry.ng.bluemix.net'...Loggedinto'registry.ng.bluemix.net'.
OK
```

Step8:UploadadockerimagetotheContainerRegistry

Thisrequiresseveralsteps,let'squicklygothroughthem.

Createanamespace

Lístyour currentnamespacebytypíng:

```
ibmcloudcrnamespace-líst
```

If you want to create a new namespace for yourself, you can do so wíth thíscommand:

```
ibmcloudcrnamespace-add<my_namespace>
```

Tagyourlocaldockerímagewíththe IBMCloudcontaínerregístry

Hereísthecommand:

```
docker tag <source_ímage>:<tag> regístry.<regíon>.bluemíx.net/<my_namespace>/<new_ímage_repo>:<new_tag>
```

<source_ímage>:<tag>íswhatyou have on yourmachíne thatyoucreatedearlíer.

Forexample:lb4-símple-web-app:latest

regístry.<regíon>.bluemíx.netístheconteínerregístryregíonyouloggedíntobefore.

Forexample:regístry.ng.bluemíx.net

<my_namespace>ísthenamespaceyoucreatedforyourselfForexample:dremond

<new_ímage_repo>:<new_tag>canbewhateveryou wantíttobe;theydon'thave toexístyet

Forexample:loopback4_webapp_repo:1

So,puttíngthesevaluestogether,mycommandwílllooklíke thís:

```
docker tag lb4-símple-web-app:latest regístry.ng.bluemíx.net/dremond/loopback4_webapp_repo:1
```

Pushthelocalímagetotheconteínerregístry

```
dockerpushregístry.ng.bluemíx.net/dremond/loopback4_webapp_repo:1
```

Youwíllseeaprogressbarlíkethís:

```
The push refers to repository [regístry.ng.bluemíx.net/dremond/loopback4_webapp_repo]

478b1e842aa3:

Pushed6fd2223ea65e:Pushed
```

```
a90c4aba186a:Pushing[==========>                                                    ]     51.4
MB/207.9MB

bb288a38e607:Pushed

53981d6ee3d2:Mountedfromdremond/loopback4_repo

b727eae390f6:Mountedfro
```

Waituntilitiscompleted.

```
The push refers to repository [registry.ng.bluemix.net/dremond/loopback4_webapp_repo]

478b1e842aa3:

Pushed6fd2223ea65e:

Pusheda90c4aba186a:

Pushedbb288a38e607:Pushed

53981d6ee3d2: Mounted from dremond/loopback4_repob727eae390f6: Mounted from

dremond/loopback4_repodf64d3292fd6:Mountedfromdremond/loopback4_repo

1:digest:sha256:939ada9d1b7f6a7483aed69dff5eee28d1931ed249b38d51d34b854b32139177
  size:1787
```

Verifytheimageisinthecontainerregistry

Typethecommand:

```
ibmcloudcrimage-list
```

Youshouldseeyourimagelisted.

```
Listing images...

REPOSITORY                                          TAG        DIGEST        NAMESPAC
registry.ng.bluemix.net/dremond/loopback4_webapp_repo   1          939ada9d1b7f   dremond
OK
```

Performabuildforthecontainerregistry

Performabuild forthecontainerregistry.

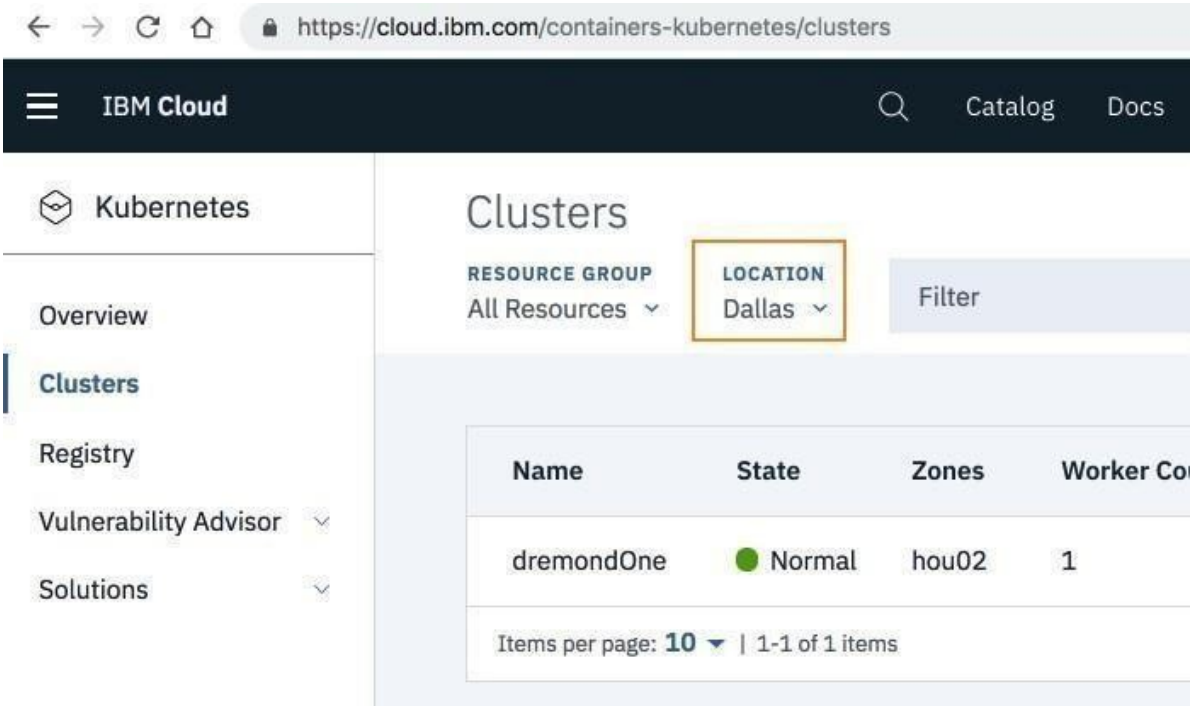
```
ibmcloud cr build -t registry.ng.bluemix.net/dremond/loopback4_webapp_repo:1.
```

This step may fail if you have exceeded the QUOTA for images in your account. In that case clear up some room and try again.

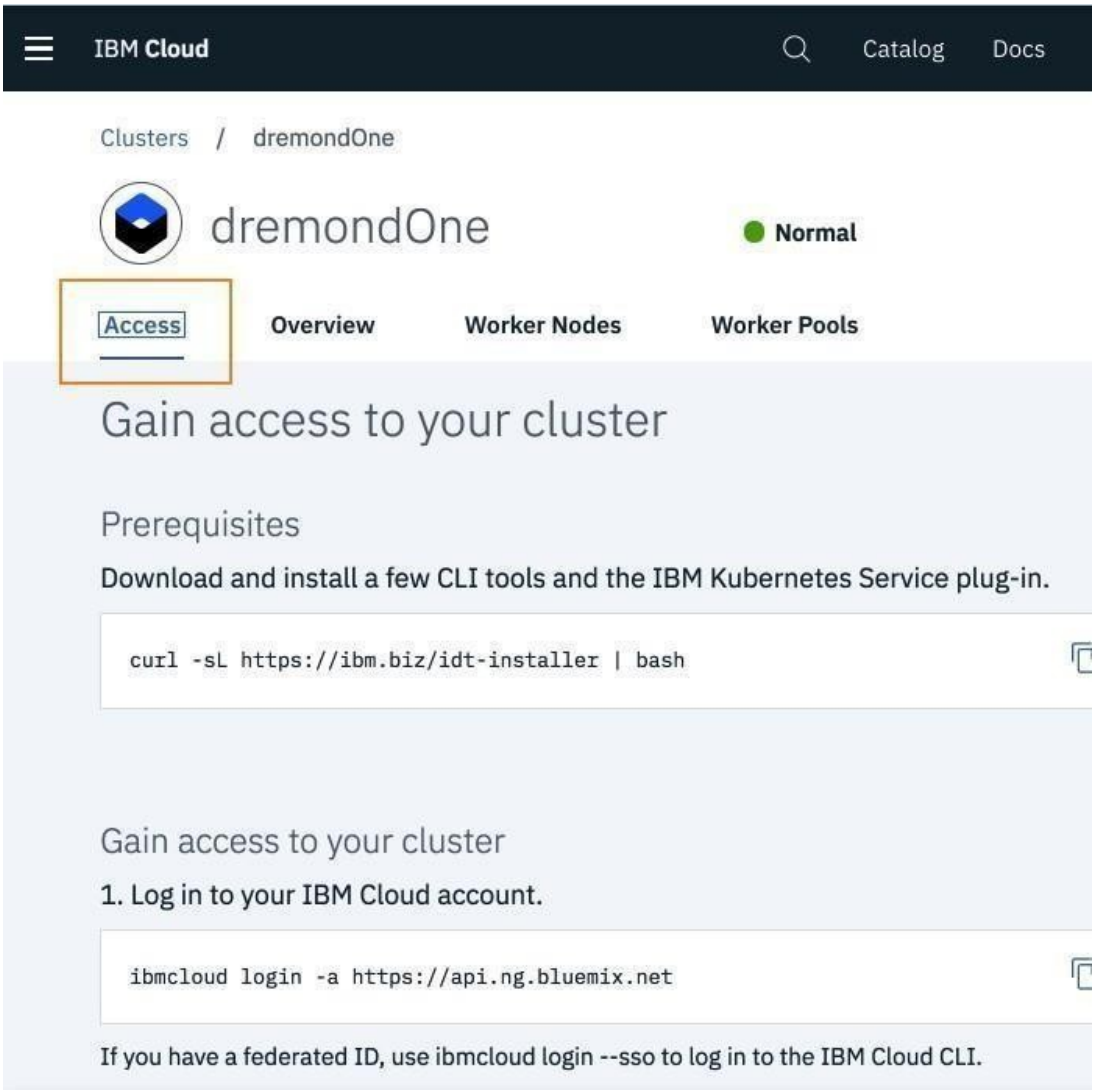
Wait until it completes.

In your IBM Cloud account, you can view your images here Step 9: Point to your Kubernetes Cluster

In a browser, log into your IBM Cloud account, and navigate to **Kubernetes > Clusters.**



I am choosing my cluster dremondOne in Dallas.



We already logged into the IBM Cloud in an earlier step, so we only need to point to the cluster.

```
ibmcloud cs region-set us-south
```

```
ibmcloud cs cluster-config <ClusterName>
```

My cluster name is **'dremondOne'** so I see this output:

```
OK

The configuration for dremondOne was downloaded successfully. Export environment variables to start using Kubernetes.
```

```
export KUBECONFIG=/Users/dremond/.bluemix/plugins/container-service/clusters/dremondOne/kube-config-hou02-dremondOne.yml
```

Takethe entire **'export'** line above, and paste it into your command window. Now you should be able to perform Kubernetes commands like:

```
kubectl get nodes
```

You will see output like this:

```
NAME            STATUS    ROLES     AGE       VERSION

10.76.193.58    Ready     <none>    13d       v1.10.8+IKS
```

Ok, so now we are ready to deploy your Loopback4 application to Kubernetes!

Step 10: Deploy your Loopback4 application to Kubernetes

Create a deployment

Create a deployment named: lb4-simple-web-app-deployment; using the image we placed in the container registry.

```
kubectl run lb4-simple-web-app-deployment --image=registry.ng.bluemix.net/dremond/loopback4_webapp_repo:1
```

Verify that the pods are running

```
kubectl get pods
```

You should see

```
NAME                                                READY     STATUS    RESTARTS   AGE

lb4-simple-web-app-deployment-5bfcb546d8-r7cs47m    1/1       Running   0
```

A status of **'Running'** is a good sign. If you have anything other than this, then there may be something wrong with your docker image, or it may have vulnerability issues you need to address.

To see the logs of your pod, type:

```
kubectl logs lb4-simple-web-app-deployment-5bfcb546d8-r7cs4
```

and you will see something like this:

```
Server is running at http://127.0.0.1:3000

Try http://127.0.0.1:3000/ping
```

Create a service

Expose your deployment with a service named: lb4-simple-web-app-service

```
kubectlexposedeployment/lb4-simple-web-app-deployment--type=NodePort--port=3000--name=lb4-simple-web-app-service--target-port=3000
```

ObtaintheNodePortofyourservice

Let'sdeterminetheNodePortofyourservice.

```
kubectldescribeservicelb4-simple-web-app-service
```

Youwillseeoutputlikethis:

```
Name:                     lb4-simple-web-app-service

Namespace:                default

Labels:                   run=lb4-simple-web-app-deployment

Annotations:              <none>

Selector:                 run=lb4-simple-web-app-deployment

Type:                     NodePort

IP:                       172.21.103.26

Port:                     <unset>3000/TCP

TargetPort:               3000/TCP

NodePort:                 <unset>31701/TCP

Endpoints:                172.30.78.136:3000

SessionAffinity:          None

ExternalTrafficPolicy:Cluster

Events:                   <none>
```

Inthiscase,theNodePortis31701.

ObtainthepublicIPaddressofthecluster

Let'sdeterminethepublicIPaddressofthecluster

```
ibmcloudksworkersdremondOne
```

Youshouldseesomethinglikethis

```
OK

ID                          PublicIP        PrivateIP       Machine   Type   State    S
tatus

kube-hou02-pa45e...6-w1ready 184.173.5.187  10.76.193.58    free      normal   R
```

Inmycase,thepublicIPis:184.173.5.187

Sonowwecanformulate theurlofourloopback4applicationusingthose twopieces
:

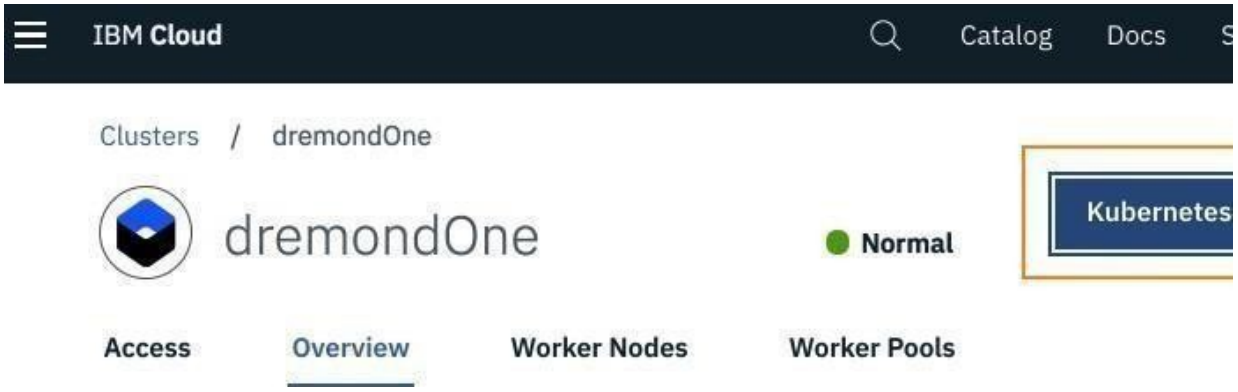
```
http://184.173.5.187:31701
```

Openyourbrowserandattempttoaccessalltheseurls

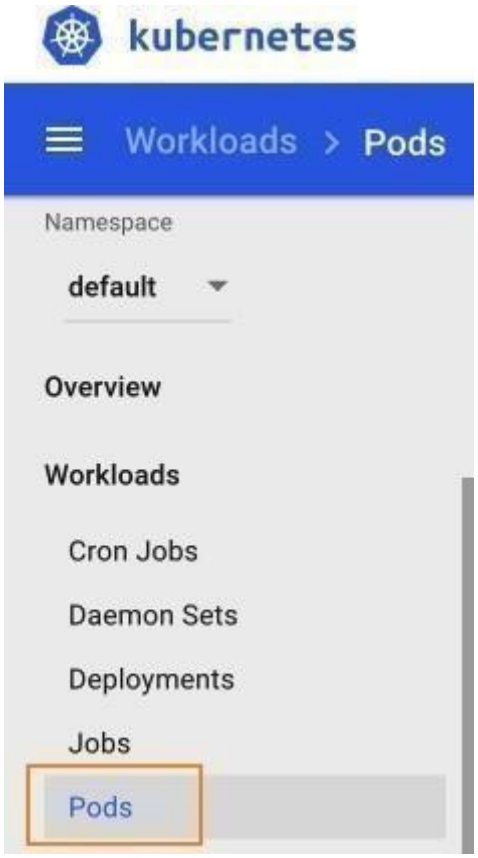
```
http://184.173.5.187:31701/

http://184.173.5.187:31701/ping

http://184.173.5.187:31701/explorer

http://184.173.5.187:31701/openapi.json
```

Step11:ViewyourappintheKubernetesDashboard

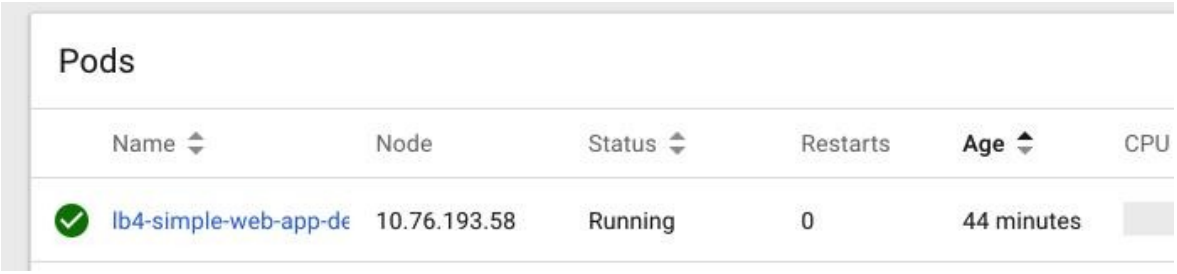
Let's go take a look at your application in the Kubernetes dashboard.Clickthe'**KubernetesDashboard**'buttonnexttoyourcluster'sname.



Under 'Workloads', select 'Pods'

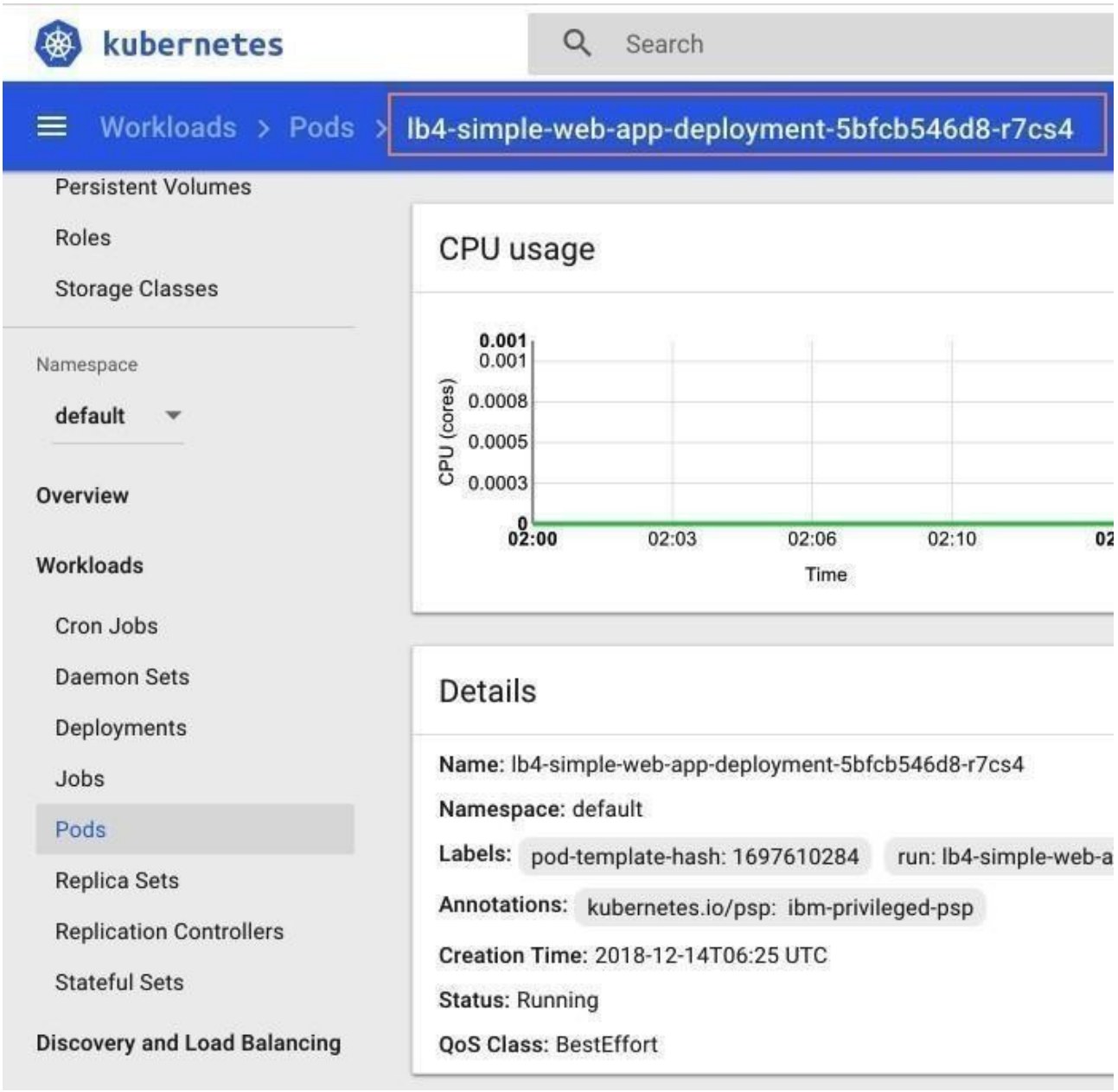


Locate your application, and click on its name



If you want to open a shell into the container in the pod, click on the EXEC button.

If you want to view the logs of the container in the pod, click on the LOGS button.



So there you have it! You have successfully deployed a Loopback 4 application to Kubernetes on the IBM Cloud.