INVENTORY MANAGEMENT SYSTEM FOR RETAILERS
Deploying to Kubernetes on IBM Cloud Overview

Date	**17 November 2022**
Team ID	**PNT2022TMID04841**
Project Name	**Inventory Management for Retailers**

Overview:

This tutorial shows how to deploy a look Back onto Kubernetes on the IBM Cloud.

Prerequisite:

You'll need the following:
Node.js 10 or higher Docker
18.06 or higher
Signup for an IBM Cloud Account if you don'thavd one alreadyIBM
Cloud CLI,Container registry CLI,etc

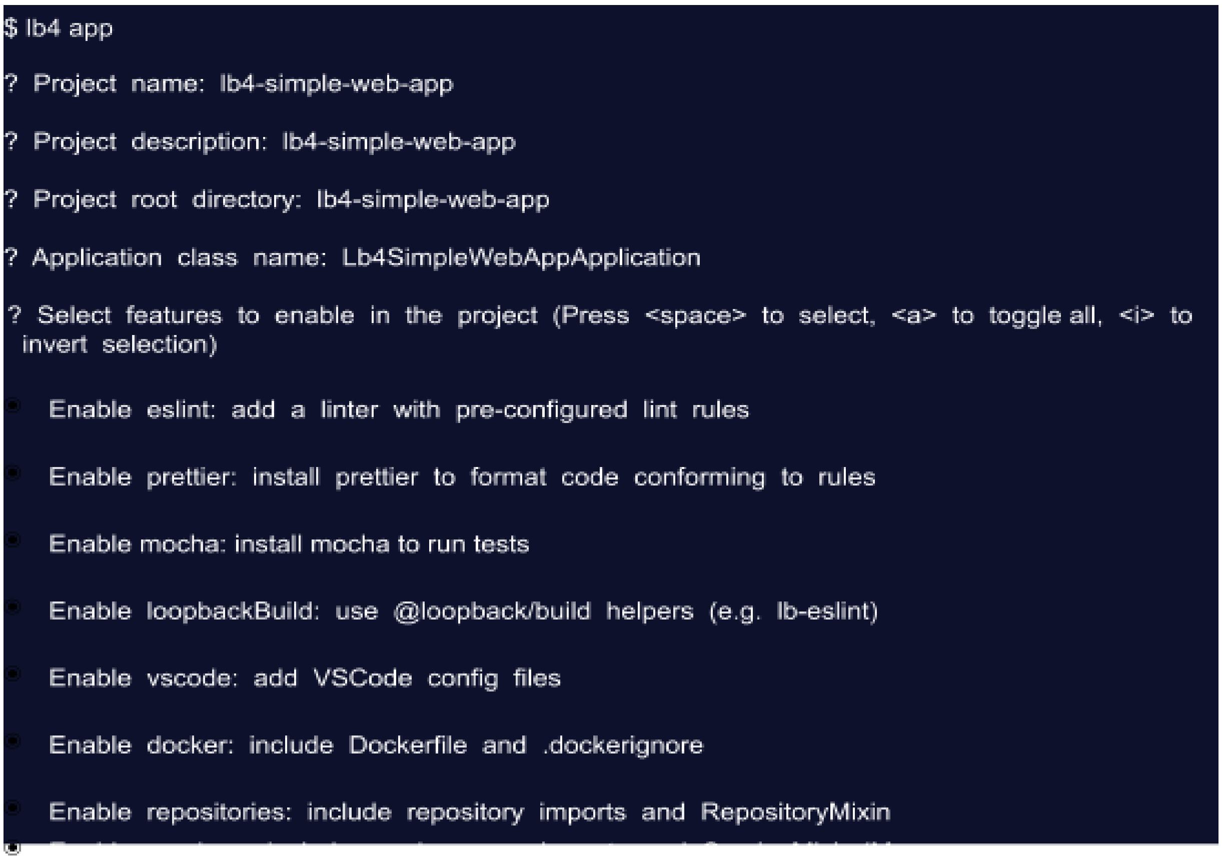
1. Kubermetes CLI (kubecti)
2. LoopBack4CLI

Let’s install the LoopBack 4 CLI :

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

Step 1: Scaffold LoopBack 4 app

Run the lb4 appcommand, and specify all the values provided below.



```
$ lb4 app
? Project name: lb4-simple-web-app
? Project description: lb4-simple-web-app
? Project root directory: lb4-simple-web-app
? Application class name: Lb4SimpleWebAppApplication
? Select features to enable in the project (Press <space> to select, <a> to toggle all, <i> to
  invert selection)
●  Enable eslint: add a linter with pre-configured lint rules
●  Enable prettier: install prettier to format code conforming to rules
●  Enable mocha: install mocha to run tests
●  Enable loopbackBuild: use @loopback/build helpers (e.g. lb-eslint)
●  Enable vscode: add VSCode config files
●  Enable docker: include Dockerfile and .dockerignore
●  Enable repositories: include repository imports and RepositoryMixin
●
```

The lb4-simple-web-appproject is created.

Navigate to the main directory of the project

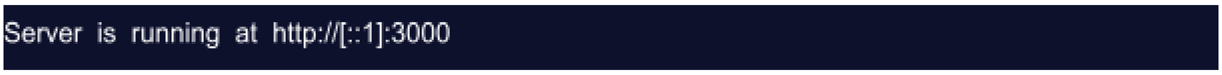
```
cd lb4-simple-web-app
```

Step 2: Run the application locally

In a command window in the main directory of your project, type:

```
npm start
```

The application will build, and then the server should start up successfully anddisplay



```
Server is running at http://[::1]:3000
```

```
Try http://[::1]:3000/ping
```

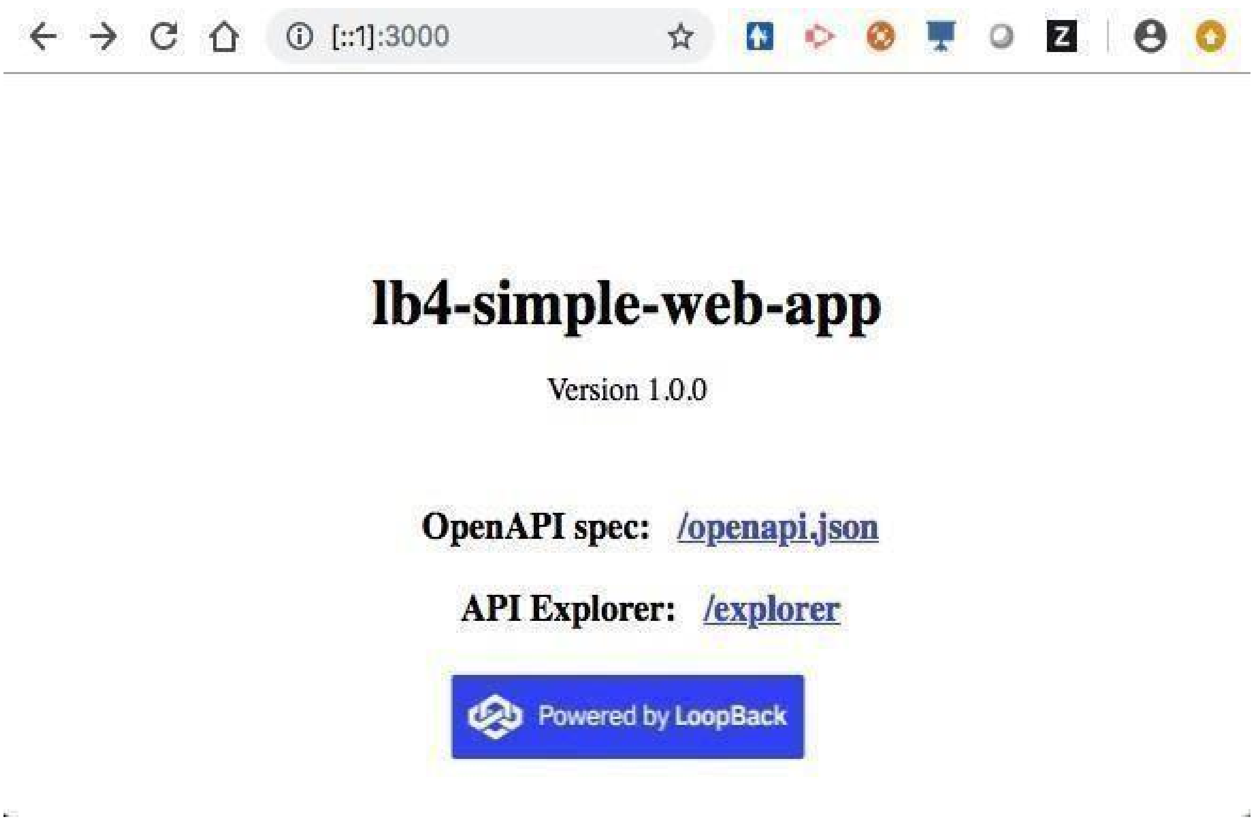
Open your browser and attempt to access all these url

http://[::1]:3000/

http://[::1]:3000/ping

http://[::1]:3000/explorer

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


[::1]:3000

lb4-simple-web-app

Version 1.0.0

OpenAPI spec: /openapi.json

API Explorer: /explorer

Make sure that the application runs well before continuing to the next step.In the command window, stop the application with

```
Ctrl + C
```

Step 3: Build a Docker image

Review the two Docker-related files that have been conveniently provided, .dockerignoreand Dockerfile, but leave them unchanged for this tutorial.Notice the HOSTand PORTenvironment variable values:

```
ENV HOST=0.0.0.0 PORT=3000
```

In the package.jsonfile, a docker:buildcommand has been provided.

```
"docker:build": "docker build -t lb4-simple-web-app ."
```

Run the command:

```
npm run docker:build
```

When it completes, you will see :

```
Successfully built 7d26df6c1561

Successfully tagged lb4-simple-web-app:latest
```

You can find your image by typing:

```
docker images | grep lb4-simple-web-app
```

It will display something like this :

```
lb4-simple-web-app       latest 7d26df6c1561
```

Step 4: Run the application in Docker

In the package.jsonfile, a docker:runcommand has been provided.

```
 "docker:run": "docker run -p 3000:3000 -d lb4-simple-web-app"
```

Run the command:

```
 npm run docker:run
```

Afterwards, type:

```
docker ps
```

You should see something like this:

```
                  AG                       MAN              ATED             TUS
          PO      E                        D

                                                             ago              seco
```

To see the log output of your container, you can type:

```
 docker  logs  <container_id>      For example : a9962339e863
```

You should see something like:

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

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

Open your browser and attempt to access all these urls

http://127.0.0.1:3000/

http://127.0.0.1:3000/ping

http://127.0.0.1:3000/explorer

http://127.0.0.1:3000/openapi.json

Step 5: Stop the application running in Docker

Find the container id

```
docker ps | grep lb4
```

You should see something like this:

```
a99623339e863     lb4-simple-web-app         "node ."
```

The leftmost value is the container id.

Type:

```
docker stop <container id>For example : a99623339e863
```

Step 6: Log into IBM Cloud using ibmcloud logincommand

Use ibmcloud logincommand to login.

After you've been successfully logged in, you'll see something like:

```
[redacted]             [redacted]
  endpoint:
[redacted]             [redacted]
  n:                     th
[redacted]             [redacted]
  er:                    com
[redacted]             [redacted]
  nt:                    Account
[redacted]             [redacted]
  group:                 lt
[redacted]
```

Step 7: Log into IBM Cloud Container Registry

```
ibmcloud cr login
```

You should see:

```
Logging in to 'registry.ng.bluemix.net'... Logged in to

'registry.ng.bluemix.net'.

OK
```

Step 8: Upload a docker image to the Container Registry

This requires several steps, let's quickly go through them.

Create a namespace

List your current namespace by typing:

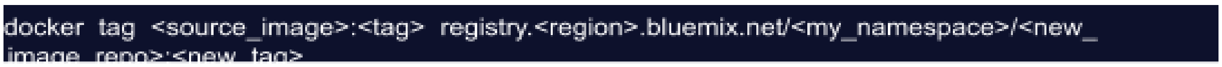
```
ibmcloud cr namespace-list
```

If you want to create a new namespace for yourself, you can do so with this command:

```
ibmcloud cr namespace-add <my_namespace>
```

Tag your local docker image with the IBM Cloud containerregistry

Here is the command:



```
docker  tag  <source_image>:<tag>  registry.<region>.bluemix.net/<my_namespace>/<new_
image_repo>:<new_tag>
```

<source_image>:<tag>is what you have on your machine that you created earlier.

For example : lb4-simple-web-app:latest

registry.<region>.bluemix.netis the container registry region you logged

intobefore. For example : registry.ng.bluemix.net

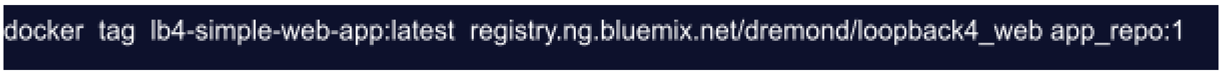
<my_namespace>is the namespace you created for yourselfFor

example : dremond

<new_image_repo>:<new_tag>can be whatever you want it to be; they don't have toexist yet

For example : loopback4_webapp_repo:1

So, putting these values together, my command will look like this:

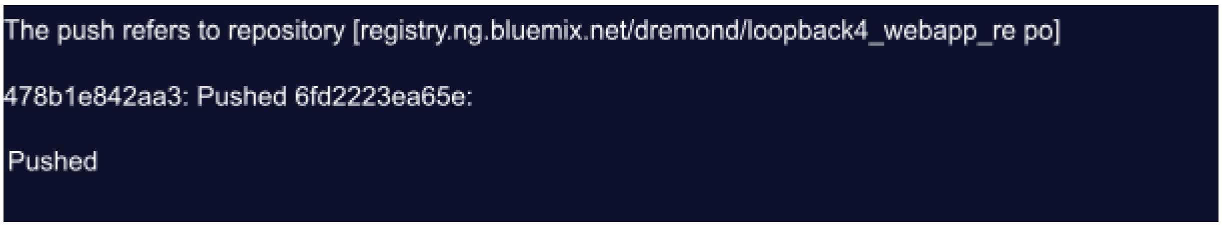


```
docker  tag  lb4-simple-web-app:latest  registry.ng.bluemix.net/dremond/loopback4_web app_repo:1
```

Push the local image to the container registry

```
docker push registry.ng.bluemix.net/dremond/loopback4_webapp_repo:1
```

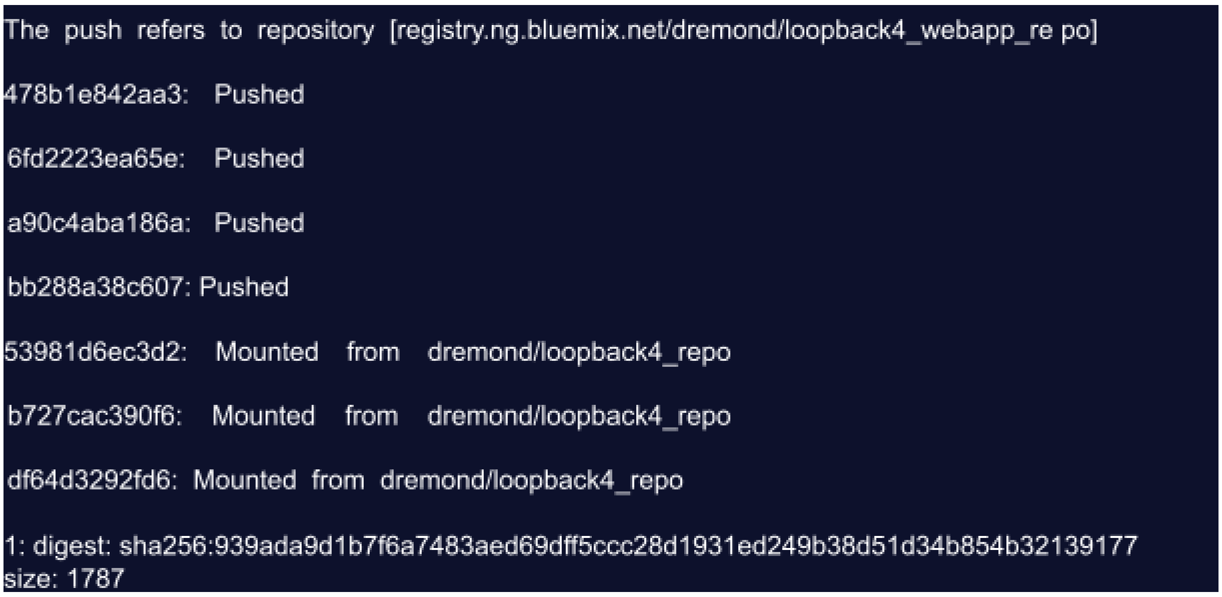
You will see a progress bar like this:



```
The push refers to repository [registry.ng.bluemix.net/dremond/loopback4_webapp_re po]

478b1e842aa3: Pushed 6fd2223ea65e:

 Pushed
```

```
[redacted]                                                    [redacted]
MB/207.9MB                                                     51.4
[redacted]
```

Wait until it is completed.



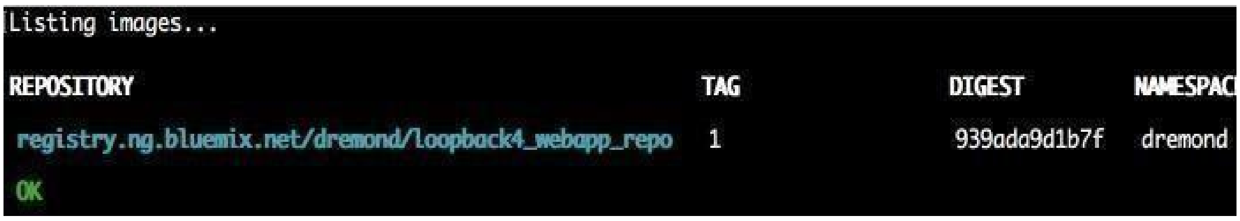
```
The push refers to repository [registry.ng.bluemix.net/dremond/loopback4_webapp_re po]
478b1e842aa3: Pushed
6fd2223ea65e: Pushed
a90c4aba186a: Pushed
bb288a38c607: Pushed
53981d6ec3d2: Mounted from dremond/loopback4_repo
b727cac390f6: Mounted from dremond/loopback4_repo
df64d3292fd6: Mounted from dremond/loopback4_repo
1: digest: sha256:939ada9d1b7f6a7483aed69dff5ccc28d1931ed249b38d51d34b854b32139177
size: 1787
```

Verify the image is in the container registry

Type the command :

```
ibmcloud cr image-list
```

You should see your image listed.



```
Listing images...

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

Perform a build for the container registry

Perform a build for the container registry.

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

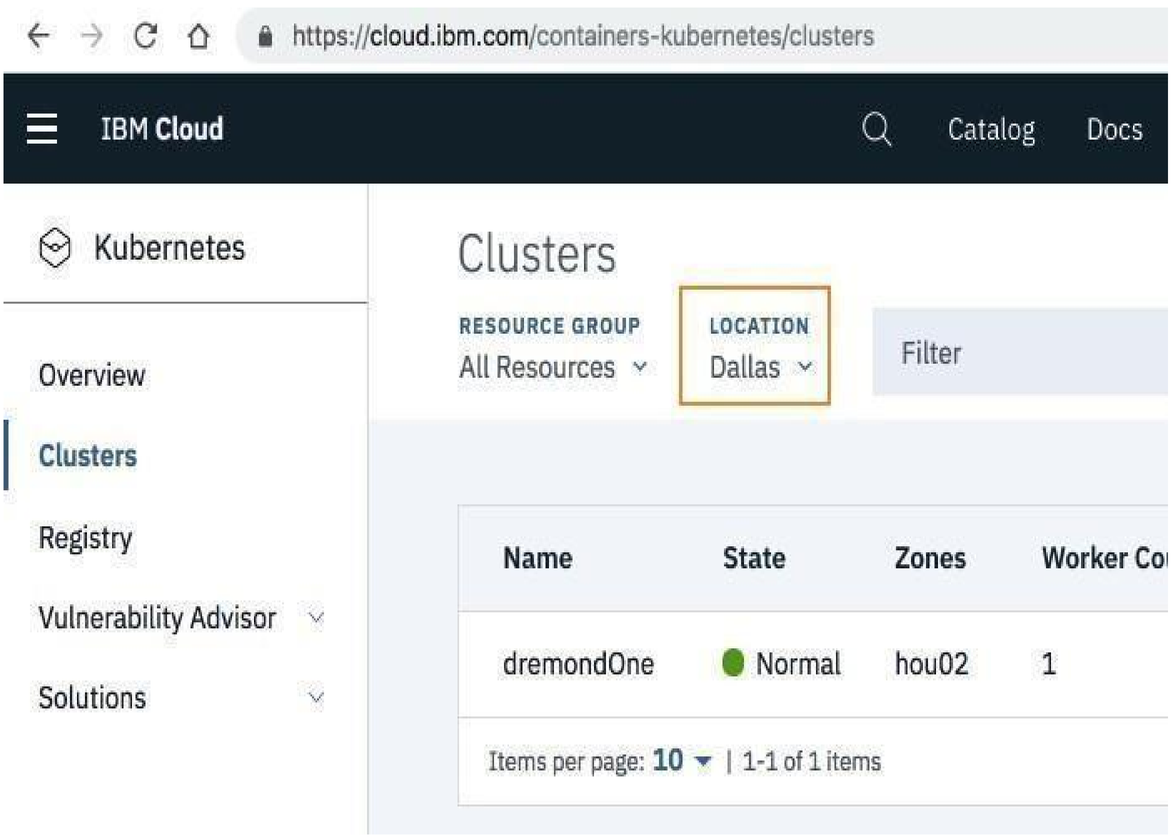
This step may fail if you have exceeded the QUOTA for images in your account. Inthat case clear up some room and try again.

Wait until it completes.

In your IBM Cloud account, you can view your images hereStep 9:

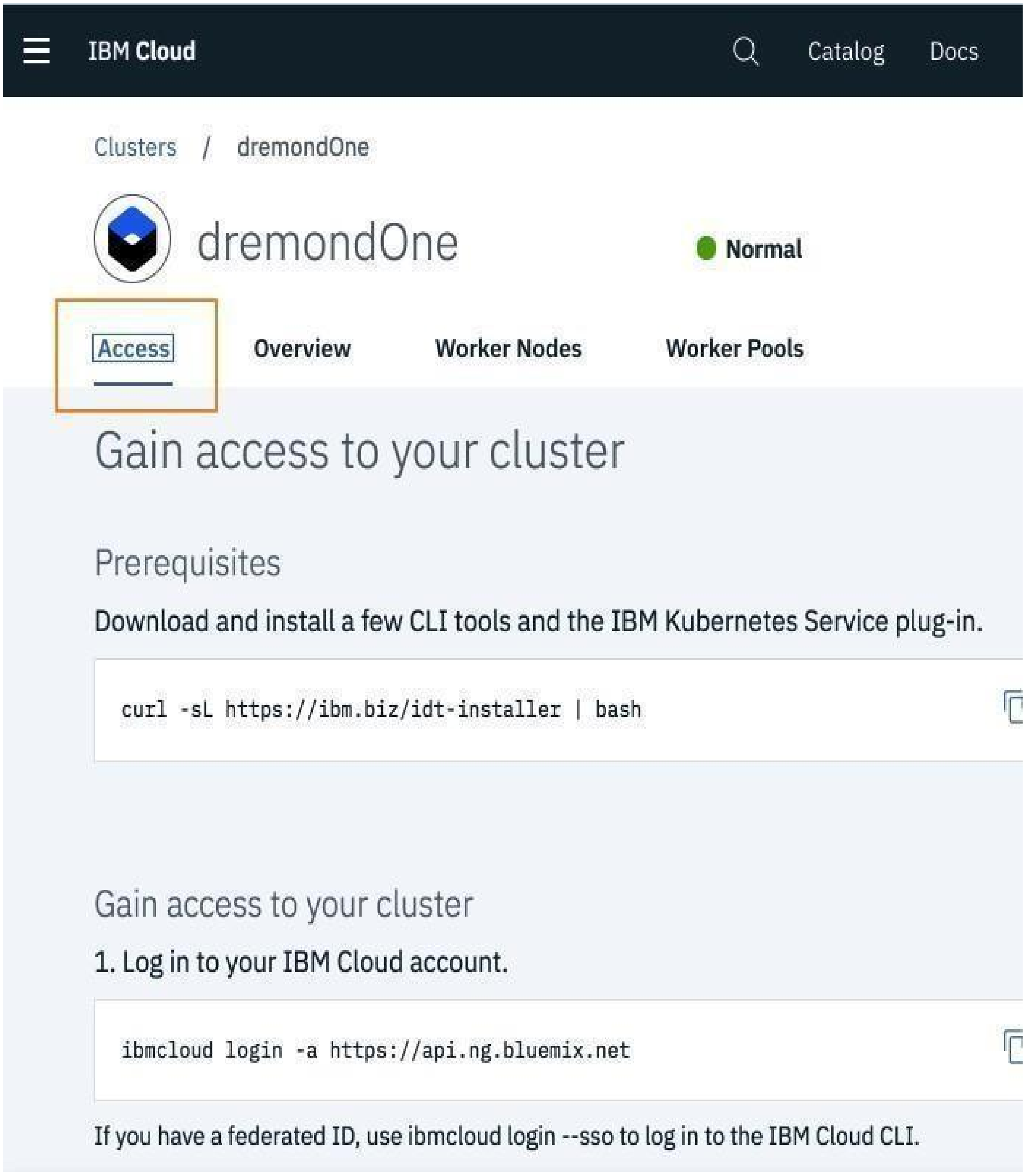
Point to your Kubernetes Cluster

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



Name	State	Zones	Worker Co
dremondOne	● Normal	hou02	1

I am choosing my cluster dremondOnein Dallas .



Gain access to your cluster

Prerequisites

Download and install a few CLI tools and the IBM Kubernetes Service plug-in.

```
curl -sL https://ibm.biz/idt-installer | bash
```

Gain access to your cluster

1. Log in to your IBM Cloud account.

```
ibmcloud login -a https://api.ng.bluemix.net
```

If you have a federated ID, use ibmcloud login --sso to log in to the IBM Cloud CLI.

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

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

```
ibmcloud cs cluster-config <Cluster Name>
```

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

```
OK

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

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

Take the 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 our Loopback4 application to Kubernetes!

Step 10: Deploy your Loopback4 application toKubernetes

Create a deployment

Create a deployment named : lb4-simple-web-app-deployment ; using the image weplaced 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          REST
ARTS    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

```
kubectl  expose  deployment/lb4-simple-web-app-deployment  --type=NodePort  --port=300 0
--name=lb4-simple-web-app-service  --target-port=3000
```

Obtain the NodePort of your service

Let’s determine the NodePort of your service.

```
kubectl  describe  service  lb4-simple-web-app-service
```

You will see output like this:

```
[illegible]                         [illegible]
 me
[illegible]                         [illegible]
 ace:                                lt
[illegible]                         [illegible]
 s:
[illegible]                         [illegible]
 :                                   ne>
[illegible]                         [illegible]
 :
[illegible]                         [illegible]
 pe:                                 ort
[illegible]                         [illegible]
 P                                   6
[illegible]                         [illegible]
 rt:                                 3000/TCP
[illegible]                         [illegible]
 :                                   CP
[illegible]                         [illegible]
 rt:                                 31701/TCP
[illegible]                         [illegible]
 s:                                  0
[illegible]                         [illegible]
                                     o
[illegible]
                                     e
[illegible]                         [illegible]
```

In this case, the NodePort is 31701 .

Obtain the public IP address of the cluster

Let’s determine the public IP address of the cluster

```
ibmcloud ks workers dremondOne
```

You should see something like this

In my case, the public IP is: 184.173.5.187

So now we can formulate the url of our loopback4 application using those two pieces
:

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

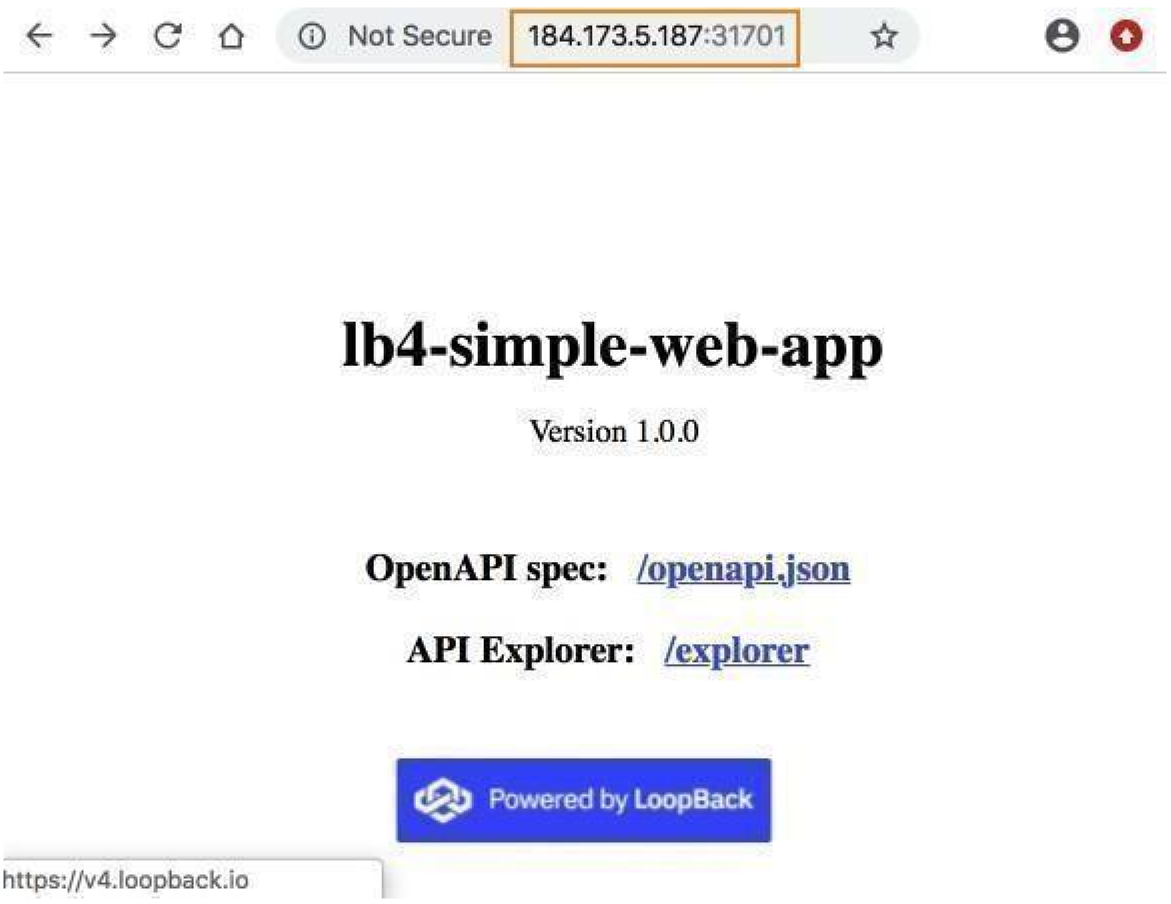
Open your browser and attempt to access all these urls

```
http://184.173.5.187:31701/


http://184.173.5.187:31701/ping


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

```
OK
ID                          Public IP        Private IP       Machine  Type   State     S
tatus
kube-hou02-pa45e...6-w1eady 184.173.5.187    10.76.193.58     free            normal    R
```

Step 11: View your app in the Kubernetes Dashboard

Let's go take a look at your application in the Kubernetes dashboard. Click the **'Kubernetes Dashboard'** button next to your cluster's name.

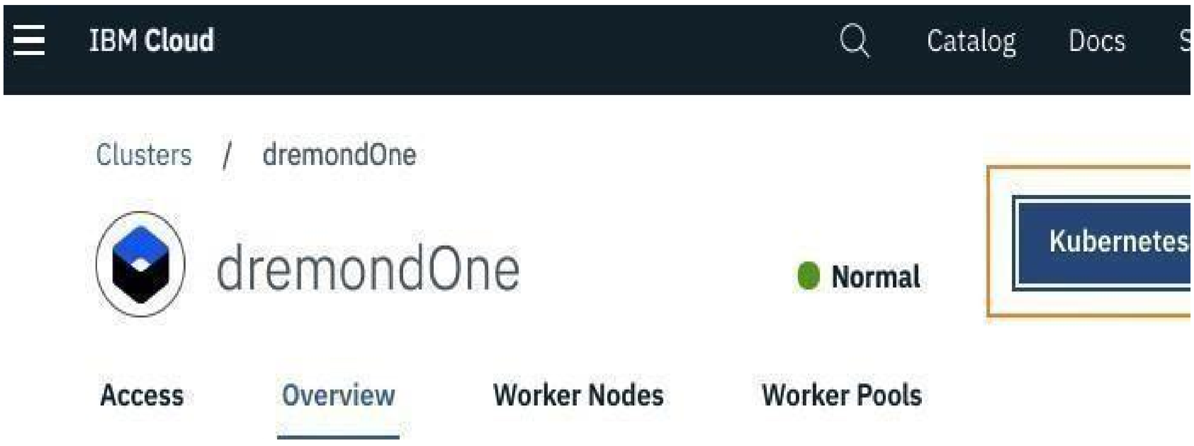


lb4-simple-web-app

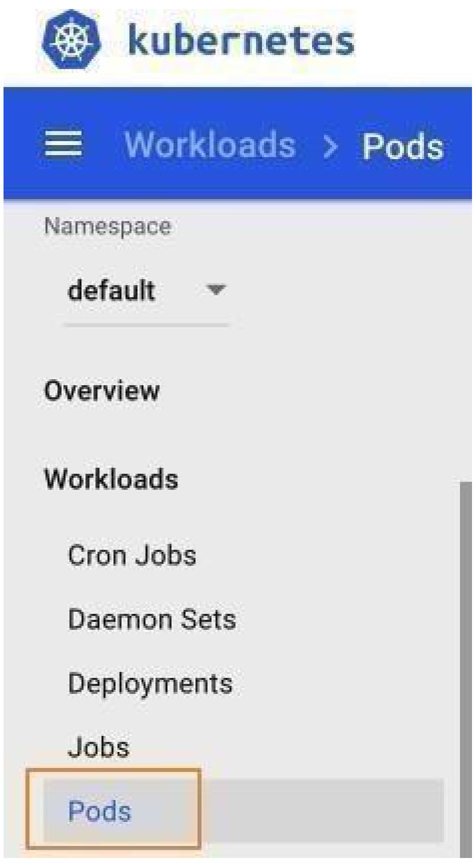
Version 1.0.0

OpenAPI spec: /openapi.json

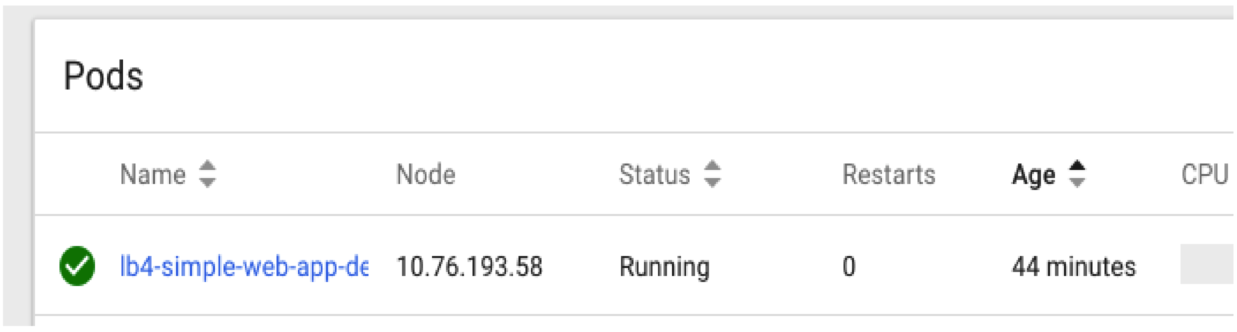
API Explorer: /explorer



Under **‘Workloads’**, select **‘Pods’**



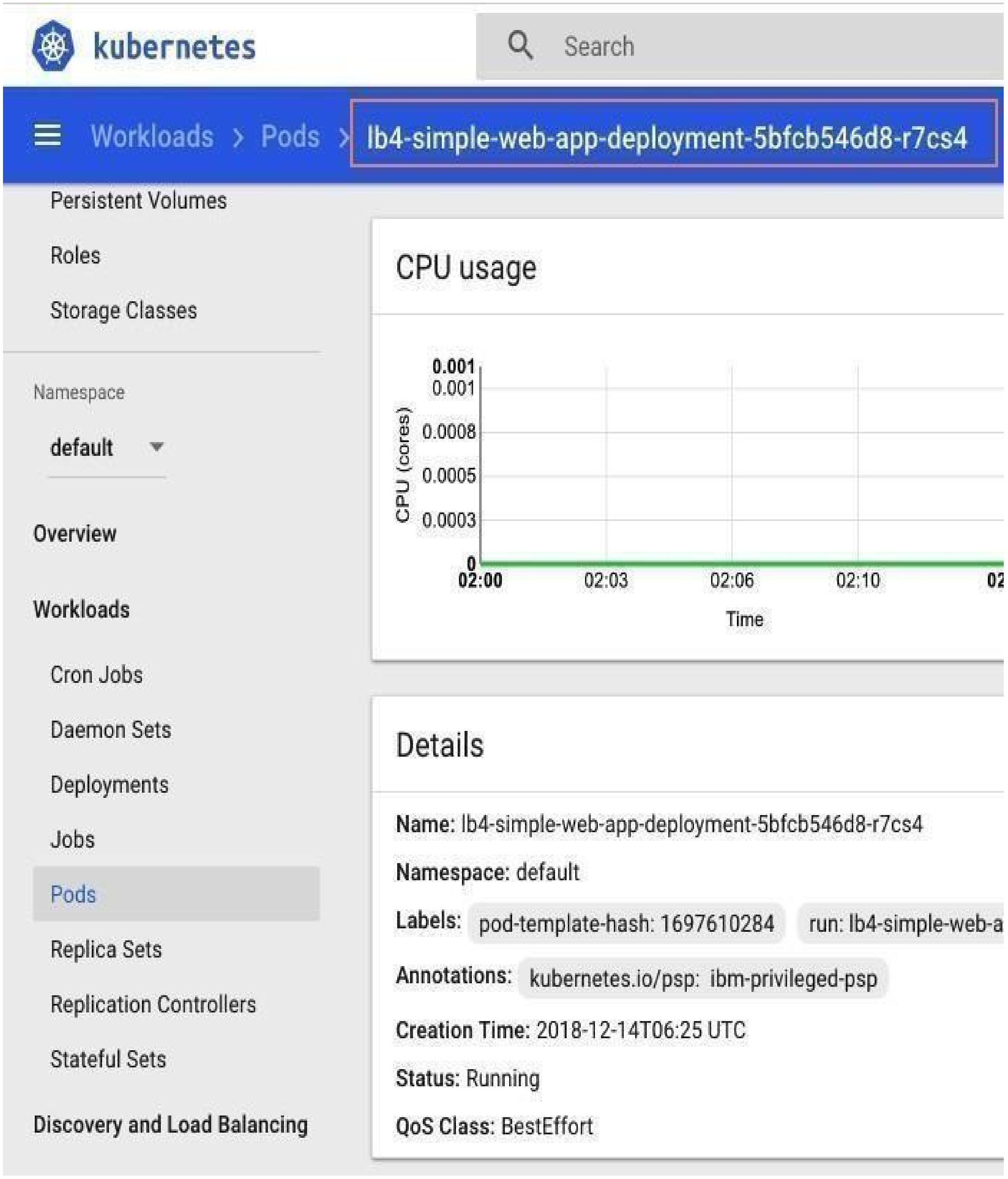
Locate your application, and click on its name



Name	Node	Status	Restarts	Age	CPU
✓ lb4-simple-web-app-de	10.76.193.58	Running	0	44 minutes	

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

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



Details

Name: lb4-simple-web-app-deployment-5bfcb546d8-r7cs4

Namespace: default

Labels: pod-template-hash: 1697610284 run: lb4-simple-web-a

Annotations: kubernetes.io/psp: ibm-privileged-psp

Creation Time: 2018-12-14T06:25 UTC

Status: Running

QoS Class: BestEffort

So there you have it! You have successfully deployed a Loopback4 application toKubernetes on the IBM Cloud.