How to deploy app code from Github onto your IBM Cloud Kubernetes Cluster

James Belton

Global IT Architect for IBM Cloud Customer Success

January 2020

(updated April 2020)

Introduction

In this 'how-to' guide, we're going to create an instance of the IBM Kubernetes Service (IKS) in the IBM Cloud, connect to it and then deploy an application to it, via a Container Registry, also in the IBM Cloud.

The application we're going to use is a simple Node.js application called "Example Health". This is a sample UI program for a patient health system and it is available in the following GitHub repository: https://github.com/IBM/node-s2i-openshift.

The steps we're going to take are broadly as follows:

1) Create the Kubernetes Cluster
2) Install the software and tools needed to complete the deployment from your local machine
3) Build a container
4) Push the container to a repository
5) Deploy and expose the application on your Kubernetes cluster

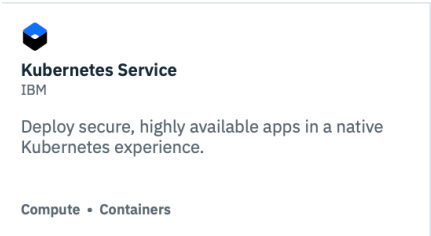
Depending on your level of expertise and what you already have deployed on your local machine, this can take a couple of hours or more to compete.

This guide may be updated from time to time, to reflect changes in the service as and when they arise.

PART 1. Create your Kubernetes cluster, download and install some software tools and connect to it.

The first step is to create a Kubernetes cluster in your IBM Cloud environment. Log you're your IBM Cloud account and then from the IBM Cloud Console, click:

Catalog -> Containers (from the left-hand menu) -> Kubernetes Service



On the next screen, click Create.

For this exercise, we'll use a free plan (so select the 'Free' tile). Note, if you don't use the Free plan and use a standard plan, these instructions currently work with clusters created using 'Classic' infrastructure and **not** VPC.

Give the cluster a suitable name (I called mine kubes-my-health - you'll see it referred to in some of the commands below), choose an IAM Resource Group and then click Create Cluster.

This will take you to a provisioning screen. It will take a little while for the cluster to provision, though it's normally done within 30 minutes.

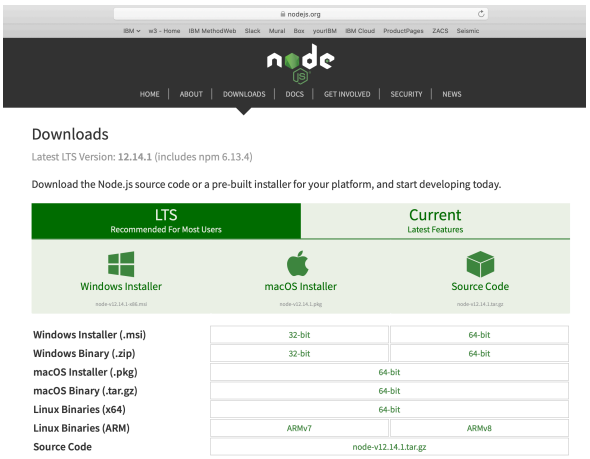
While you're waiting for the cluster to create, there are some other tools to install, if you don't already have them.

These are:

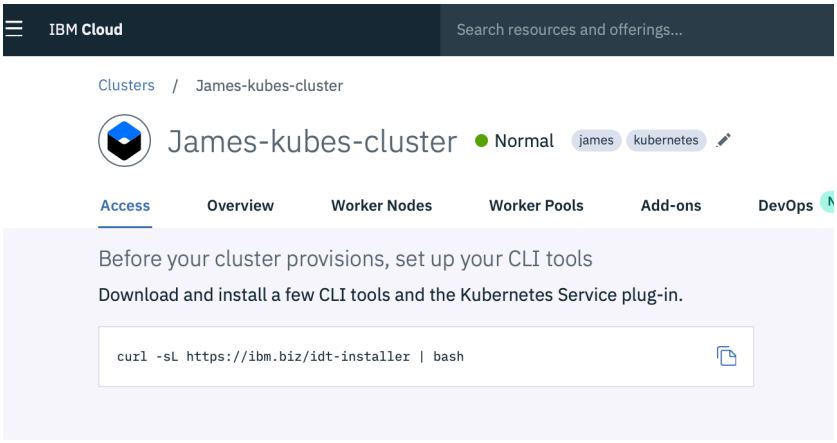
- Node.js
- The `ibmcloud` CLI
- Docker
- The `kubectl` CLI

These will allow us to build the My-health application on our local machine and to work with the Kubernetes cluster.

First, to install Node.js, open a browser and click through to https://nodejs.org. You will probably immediately see one or more 'download' buttons on your screen, but if not, then click on 'Downloads' and choose the appropriate download based on your operating system. The download and install are pretty simple, just follow the on-screen prompts.



Once Node.js is installed, the next step is to install the CLI's and Docker, which is done from a single command. The following is a breakdown of what you need to do but the 'Access' tab on your cluster's IBM Cloud page also provides these steps:



For Mac and Linux, type in the following from a Terminal window:

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

For Windows 10 Pro, run the following command in PowerShell. When starting the PowerShell window, you MUST choose run **as administrator** (note: if you see lots of errors when you run this, then you are most likely not running the PowerShell as an administrator):

```
[Net.ServicePointManager]::SecurityProtocol = "Tls12"; iex(New-Object
Net.WebClient).DownloadString('https://ibm.biz/idt-win-installer')
```

Installing the CLI and Docker may take a little while (I'd say 10 -20 minutes) but when it's done, make sure that the locations of the binaries 'ibmcloud' and 'kubectl' are in your PATH environment variable. On a Mac, these will be installed to /usr/local/bin. The easiest way to tell if the binaries are in your PATH is to open a command prompt / terminal window and type '`ibmcloud cr`'. If this returns some usage information, it's all working.

You will also need to start Docker up. On Windows systems, it tends to start automatically but, on a Mac, click Launchpad, find the Docker icon and double click it. If it's not running on Windows, click start, find Docker and double click the icon.

The next step is to log into you IBM Cloud account from the command line (if you are using Windows, switch from PowerShell to a regular 'command' window). Again, the 'Access' tab on you cluster page provides the CLI command, but you may need to define the resource group with value you set when you created the cluster.

```
ibmcloud login -a cloud.ibm.com -r <region> -g <resource-group>
```

Note: if you are using a federated account, then the login command is different and will be:

```
ibmcloud login -sso -r <region> -g <resource-group>
```

Once you are logged in, you then need to run the following command which will download some configuration information to your local machine to enable you to interact with your cluster (replace *kubes-my-health* with the name of your cluster, if it is different):

```
ibmcloud ks cluster config --cluster kubes-my-health
```

Note: In previous versions of this document that used an older version of the ibmcloud CLI, you will need to have set an environment variable KUBECONFIG. This seems to no longer be necessary.

Note: If running this ibmcloud command just gives an error message, then it's probably because your Kubernetes cluster is still building. Wait a few more minutes (it can sometimes take 30 mins or more for the cluster to build) and try again.

Then verify that you can connect to your cluster by running:

```
kubectl get nodes
```

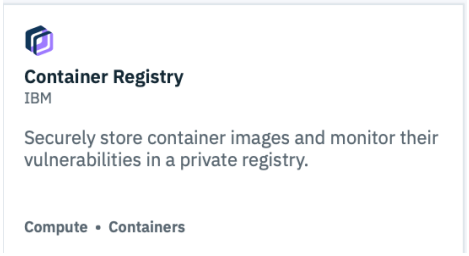
```
site — root@openface-684659594b-rch7t: ~/openface — -zsh — 84×41
jamesbelton@MacBeasty site % kubectl get nodes
NAME           STATUS   ROLES    AGE    VERSION
10.72.24.158   Ready    <none>   5d7h   v1.14.9+IKS
10.72.24.160   Ready    <none>   5d7h   v1.14.9+IKS
10.72.24.176   Ready    <none>   5d7h   v1.14.9+IKS
```

You should see output similar to that above. Note, this shows 3 nodes but if you created a free cluster, then you'll see just one.

PART 2: Create a container registry.

A container registry is a storage point in the cloud where you upload and store pre-built Docker containers that you can then run in your Kubernetes cluster. We'll use one to store the my-health container, once it's been created.

You will need to create a Container Registry in your IBM Cloud account. To do this, from the IBM Cloud console click Catalog -> Containers and then click the Container Registry tile.



Then click create. You'll notice you can't choose a free or standard plan for the Container Registry; however, the storage will be free of charge so long as you stay below the storage threshold shown. This demo will keep you well below that limit.

Back at the command line, ensure that the Container Registry plugin for the `ibmcloud` CLI is installed (it should have been already – if it isn't this command will install it, otherwise it will ask if you want to re-install – just decline):

```
ibmcloud plugin install container-registry
```

Great! We now have our tools installed, a registry configured and by now our Kubernetes cluster is probably also ready to go.

Part 3: Download the app from github, create a Docker Container and upload it to your container registry

At this point, you need to have Docker up and running on your local machine / laptop. If you've followed the steps above, you should be all set but if not and you don't have Docker on your local machine, download it from https://docker.com.

Run this command to log you Docker daemon into the IBM Cloud Container registry:

```
ibmcloud cr login
```

This will return something like:

```
Logging in to 'registry.eu-gb.bluemix.net'...
Logged in to 'registry.eu-gb.bluemix.net'.

Logging in to 'uk.icr.io'...
Logged in to 'uk.icr.io'.
```

Note: If your output is different (i.e. it does not say 'uk.icr.io') then use the values *you* see in the commands below.

Step 1 – Download the app code and run the app on your local machine:

First, pull the my-health application down to your laptop from github:

```
git clone https://github.com/jamesbeltonIBM/node-s2i-openshift
```

If this command errors with a message that implies git is not installed on your local machine, then simply open a browser, click to https://github.com/jamesbeltonIBM/node-s2i-openshift and click the green button to download a zip file that contains the code. After downloading, you'll need to move / copy the file to a working directory and unzip it.

Here, I'm going to assume that the file is in my $HOME directory (e.g. /Users/jamesbelton or C:\Users\jamesbelton) and the git command / zip download and unzip placed a directory called node-s2i-openshift into that directory.

From a command prompt, we need to run the application on the local machine to ensure it works (you can skip this step if you like but it does verify certain elements are working as they should and won't cause issues later):

```
cd node-s2i-openshift
cd site
npm install
npm start
```

In a browser, open http://localhost:8080/ - you'll see the app has started and you can play with it (enter anything for the username and password, it'll work).

If you got errors running the 'npm' commands, check that Node.js is installed on your local machine (see steps above).

Step 2 – Run it as a Docker Container on your local machine

To run the app on your local machine under Docker, we need to create a Docker Container.

First, we need a file called Dockerfile in the 'site' directory. To create the file using a Mac / Linux, type:

```
touch Dockerfile
```

On Windows, use a text editor such as Notepad or Textpad to create the file but make sure the file is saved without an extension (e.g. it appears as 'Dockerfile' and **not** 'Dockerfile.txt')

Edit (I used vi on my Mac) or create the file so that it contains the following text, and then save it.

```
####
FROM node:10-slim
```

```
USER node

RUN mkdir -p /home/node/app
WORKDIR /home/node/app

COPY --chown=node package*.json ./
RUN npm install
COPY --chown=node . .

ENV HOST=0.0.0.0 PORT=3000

EXPOSE ${PORT}
CMD [ "node", "app.js" ]
####
```

Next, build and run the Docker container on your local machine:

```
docker stop my-health
docker rm my-health
docker build --no-cache -t my-health .
docker run -d --restart always --name my-health -p 3000:3000 my-
health
```

Then, open a browser, enter http://localhost:3000 and you should see the application running.

Stop and remove the container:

```
docker stop my-healh
Docker rm my-health
```

We've now seen that the tools we have downloaded and installed are working correctly and we're ready to send our app to Kubernetes and run it there.

Step 3 – Push the Container to the Container Repository

At this point, make sure that you are still logged into the IBM Cloud from your command prompt window. If you're still using the same window, you should be OK but if not, just re-run:

```
ibmcloud login -a cloud.ibm.com -r <region> -g <resource-group>
```

or

```
ibmcloud login -sso -r <region> -g <resource-group>
```

as above.

Next, build and tag the container ready to push to the container repository

```
docker build -—no-cache -t my-health
```

Then, check that you are logged into the container registry and create a namespace:

```
Ibmcloud cr login

ibmcloud cr namespace-add my-health
```

The first command will show something like:

```
Logging in to 'registry.eu-gb.bluemix.net'...
Logged in to 'registry.eu-gb.bluemix.net'.

Logging in to 'uk.icr.io'...
Logged in to 'uk.icr.io'.
```

While the second that creates a namespace is essentially creating a logical 'directory' construct in your container registry into which we'll place the container image.

Now tag the container, using the 'Logged in to' info that *you* see (which will not necessarily be 'uk.icr.io'):

```
docker tag my-health:latest uk.icr.io/my-health/my-health:1.0.0
```

Tagging essentially gives the container some metadata info, describing it's namespace location and a version (in this case the version is 1.0.0)

Now, push the container to the repository:

```
docker push uk.icr.io/my-health/my-health:1.0.0
```

Check that the image got to the registry successfully:

```
ibmcloud cr image-list
```

You should see your image listed. You are now ready to get the application running on your Kubernetes Cluster.

Part 4: Get the container running on your Kubernetes cluster

The next commands actually take the container that you've uploaded and apply it to the cluster to run it:

```
kubectl run my-health --image=uk.icr.io/my-health/my-health:1.0.0
```

And next we expose it to the world:

```
kubectl expose deployment/my-health --type=NodePort --port=3000 --
name=my-health-service --target-port=3000
```

Check that that the container is running:

```
kubectl describe service my-health-service
```

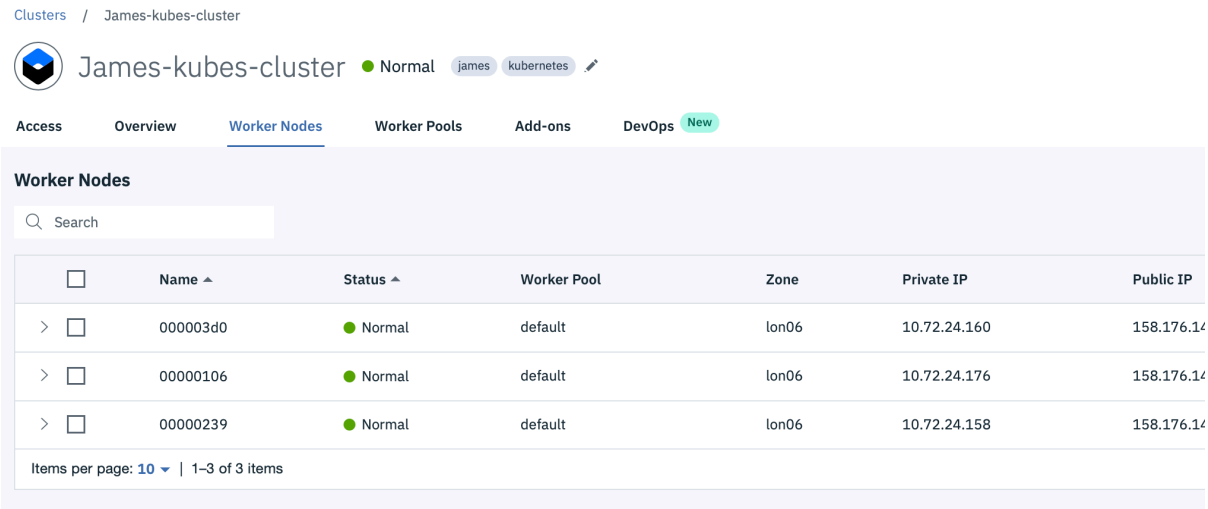
You should see output similar to (though not exactly the same as) this:

```
site — root@openface-684659594b-rch7t: ~/openface — -zsh — 84×41
jamesbelton@MacBeasty site % kubectl describe service example-health
Name:                     example-health-service
Namespace:                default
Labels:                   run=example-health
Annotations:              <none>
Selector:                 run=example-health
Type:                     NodePort
IP:                       172.21.225.116
Port:                     <unset>  3000/TCP
TargetPort:               3000/TCP
NodePort:                 <unset>  31948/TCP
Endpoints:                172.30.1.202:3000,172.30.251.134:3000,172.30.251.135:3000
+ 2 more...
Session Affinity:         None
External Traffic Policy:  Cluster
Events:                   <none>
```

Take a note of the port number on the NodePorts line as this is the port that the service is publicly listening on.

PART 5: How to find the Public IP address of your Kubernetes Cluster:

On the dashboard for your cluster, click Worker Nodes. If you are using a free cluster, only one node will be displayed. If you are running on a non-free cluster, then you'll see multiple nodes. For example:



Check the value listed under Public IP (I've deliberately not shown all of my Public IP address in the screen shot above). If you are running a free cluster, there will be only one Public IP address and that will be the node on which your service is running.

Next, enter the public IP address and the Node Ports port number (from the NodePorts line in Step 4 above) i.e. https://159.122.181.176:30896/ into a browser and the app should be seen (if you have multiple Worker Nodes, then you can either try each public IP address).

Congratulations! You have deployed the app to a Kubernetes cluster.

Conclusion

Congratulations! By following this how-to, you have been able to take a Node.js application from a Github repository, build a container from it and then take that container, push it to a container registry and then run and expose the application on an IBM Cloud Kubernetes cluster.

This is a rather simplistic example – if you wanted to deploy an application for production use, you would need to implement further steps to ensure that the application is secure and highly available. To find out more, check out the IBM Cloud Kubernetes service documentation at https://cloud.ibm.com/docs/containers?topic=containers-getting-started

IBM Cloud also offers OpenShift on its platform. OpenShift is a Red Hat product that makes Kubernetes easier to use, brings additional out of the box security and integrates features such as logging and monitoring so you don't have to. For more information about Red Hat OpenShift on IBM Cloud visit https://cloud.ibm.com/docs/containers?topic=containers-getting-started