IBM Integration Bus, Kubernetes and the Bluemix Container Service

11

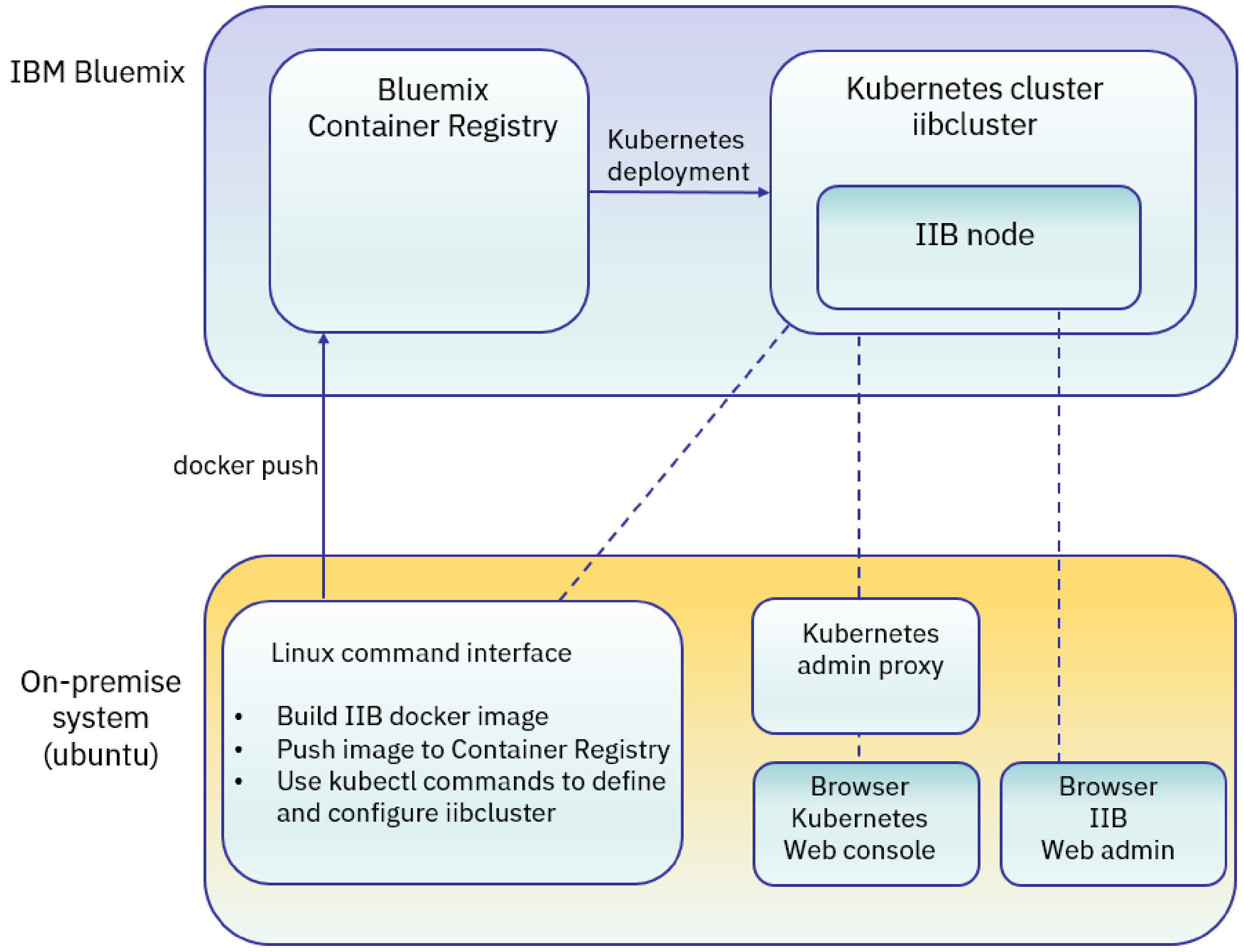
David Hardcastle
Published on August 23, 2017 / Updated on January 29, 2018

IBM is deploying the open-source software Kubernetes to manage container-based applications in the Bluemix environment. This article follows on from the first article in this series, and describes how to build and deploy a basic IBM Integration Bus (IIB) node in a Bluemix environment, using Kubernetes as a management tool for the IIB containers.

There is already a large and extensive set of Kubernetes documentation available. This article provides a basic implementation of IIB using Kubernetes, so if you are not familiar with Kubernetes, we recommend you review some of the many available articles. A good starting point would be the Bluemix Container Service website, at https://console.bluemix.net/docs/containers/container_index.html?pos=2, and specifically the paragraph “Tutorials for clusters” will be helpful.

Schematic for this article

This document describes the basic minimum set of tasks to configure a Kubernetes cluster to allow an IIB integration node to run. The schematic below shows the main components that will be used and built in this article.


IBM Bluemix
Bluemix Container Registry
Kubernetes deployment
Kubernetes cluster iibcluster
IIB node
docker push
On-premise system (ubuntu)
Linux command interface
• Build IIB docker image
• Push image to Container Registry
• Use kubectl commands to define and configure iibcluster
Kubernetes admin proxy
Browser Kubernetes Web console
Browser IIB Web admin

This article was generated on a Linux ubuntu system, so some of the commands will reflect this environment.

Linux commands are shown in italics, like this: cd /home/iibadmin.

Install the pre-requisite components on the ubuntu system

The ubuntu system that will be used to define and configure the Kubernetes and Container Registry require some software components to be available, as follows. Run the following commands in a Linux terminal:

Install curl. You may already have this installed, but if not:
sudo apt-get install -y curl

In a browser, download the Bluemix CLI from https://clis.ng.bluemix.net/ui/home.htmlThe current version for Linux is Bluemix_CLI_0.5.6_amd64.tar.gz

Unzip the downloaded Bluemix CLI installation file Bluemix_CLI_0.5.5_amd64.tar.gz.
For this article, we unzipped into /home/iibadmin, creating the Bluemix_CLI directory.

In the Linux terminal, navigate to /Bluemix_CLI, and then install the Bluemix CLI:

cd Bluemix_CLI

sudo ./install_bluemix_cli

Install the Kubernetes CLI (required to deploy apps to Kubernetes and run the local Kubernetes dashboard proxy). This is described for all platforms on this page: https://kubernetes.io/docs/tasks/tools/install-kubectl/, but for the Linux ubuntu system that we used for this article, all you need to do is run these commands:

curl -LO https://storage.googleapis.com/kubernetes-release/release/$(curl -s https://storage.googleapis.com/kubernetes-release/release/stable.txt)/bin/linux/amd64/kubectl

chmod +x ./kubectl

sudo mv ./kubectl /usr/local/bin/kubectl

Confirm the Kubernetes installation:
kubectl version
At the time of writing this article, the Kubernetes CLI was at version 1.7.3

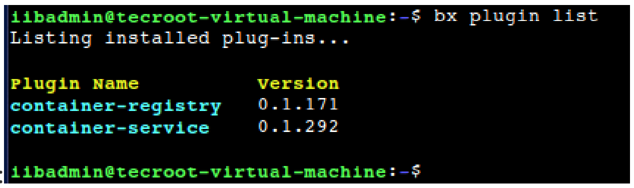
Install Bluemix Container Service plugin. Run the command:
bx plugin install container-service -r Bluemix

Confirm the Container Service plugin installation is OK:
bx plugin show container-service

Install the Container Registry plugin. Run the command:
bx plugin install container-registry -r Bluemix

Check the Bluemix Container Registry plugin is OK:
bx plugin show container-registry

Check Bluemix plugins
bx plugin list

This will show your installed Bluemix plugins: 

```
iibadmin@tecroot-virtual-machine:~$ bx plugin list
Listing installed plug-ins...

Plugin Name          Version
container-registry   0.1.171
container-service    0.1.292

iibadmin@tecroot-virtual-machine:~$
```

Define a Kubernetes Cluster

In this article, we have used the US-South region of Bluemix, so we will login to “api.ng.bluemix.net”, and push docker images to “registry.ng.bluemix.net”. If you are going to use other Bluemix regions, for example UK, you will need to adjust the commands to reflect those regions (for example “eu-gb.bluemix.net”).

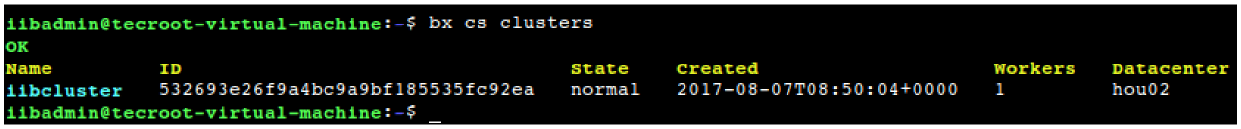
Login to Bluemix CLI:
bx login -a https://api.ng.bluemix.netNote – if you have a federated Bluemix ID, you have to include the -sso parameter, and obtain a one-time passcode, using this variation of the above command:
bx login -sso -a https://api.ng.bluemix.net
Note the cursor does not move when you type the passcode, so be sure you type (or paste) correctly.

Login to the Container Registry:
sudo bx cr loginNote – on my system, I required the “sudo” prefix, to ensure the correct local credentials were used for CR authentication.

Initialise the Bluemix Container Service plugin:
bx cs init

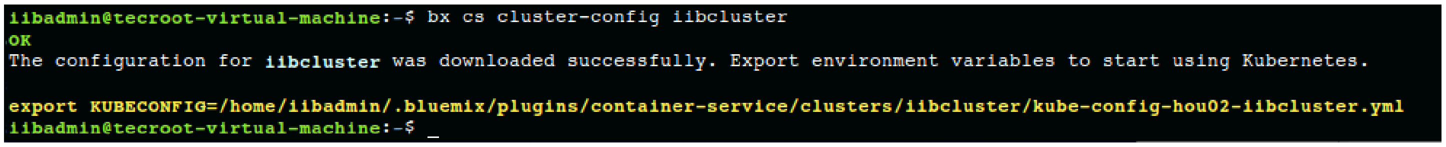
Create a Kubernetes cluster (provide a suitable name; this example uses “iibcluster”):
bx cs cluster-create –name iibcluster

Check the cluster has been created (this could take some time to complete – minimum 15 minutes, but possibly longer). This should show “normal” when fully available.bx cs clusters


```
iibadmin@tecroot-virtual-machine:~$ bx cs clusters
OK
Name         ID                                 State    Created                    Workers   Datacenter
iibcluster   532693e26f9a4bc9a9bf185535fc92ea   normal   2017-08-07T08:50:04+0000   1         hou02
iibadmin@tecroot-virtual-machine:~$ _
```

Check the cluster workers for the new cluster:
bx cs workers iibcluster

Get the cluster configuration details (required in order to set local configuration context):
bx cs cluster-config iibclusterThis will respond with a generated EXPORT command.


```
iibadmin@tecroot-virtual-machine:~$ bx cs cluster-config iibcluster
OK
The configuration for iibcluster was downloaded successfully. Export environment variables to start using Kubernetes.

export KUBECONFIG=/home/iibadmin/.bluemix/plugins/container-service/clusters/iibcluster/kube-config-hou02-iibcluster.yml
iibadmin@tecroot-virtual-machine:~$ _
```

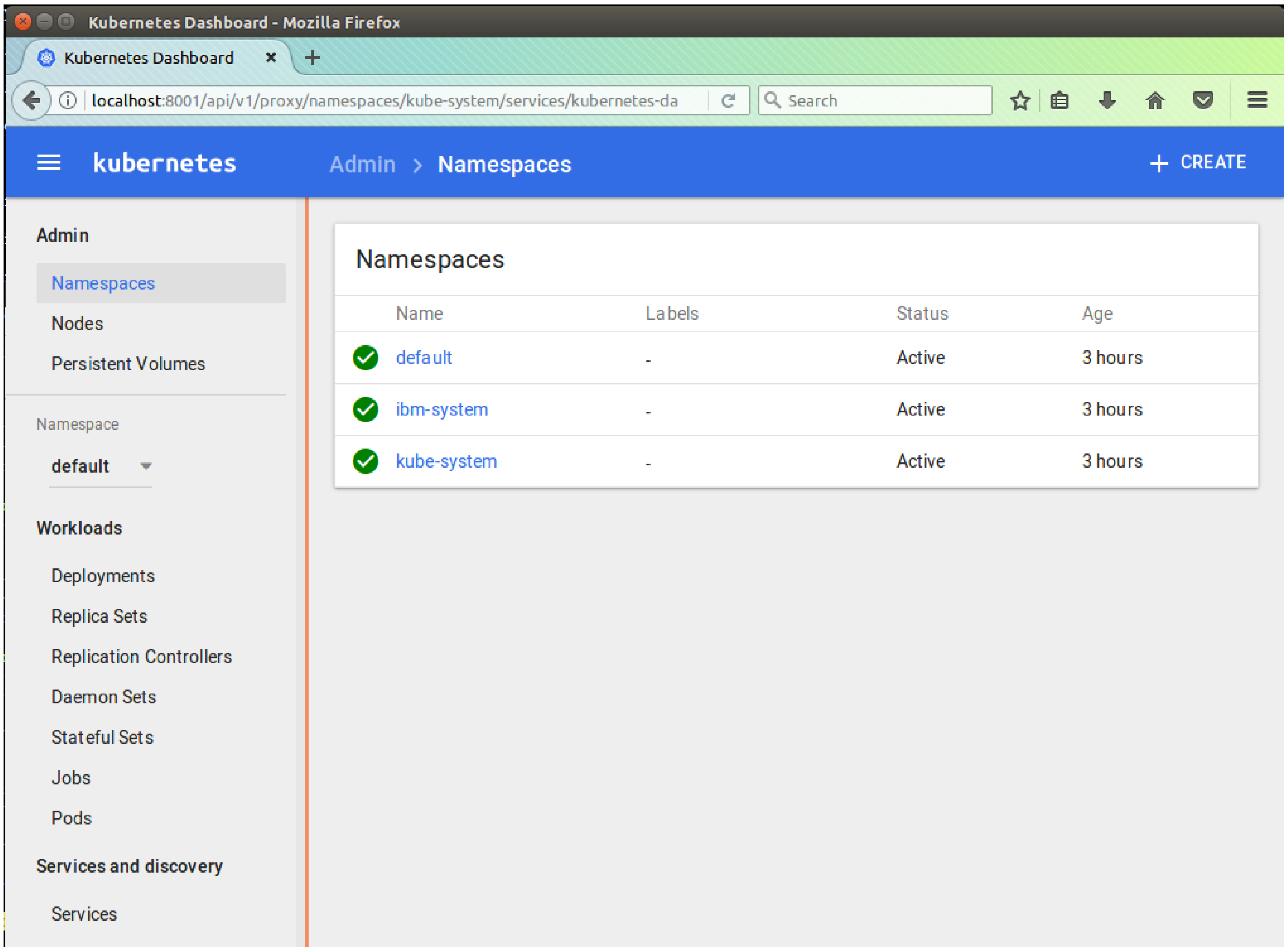
Copy this (highlight the entire EXPORT command, right-click, copy), and then paste into the command prompt. Then press return to execute. The EXPORT will have a format similar to this:
export KUBECONFIG=/home/iibadmin/.bluemix/plugins/container-service/clusters/iibcluster/kube-config-hou02-iibcluster.yml

Confirm the KUBECONFIG variable has been set correctly:
echo $KUBECONFIG

Start the kubectl web admin proxy. It’s a good idea to execute this proxy in a second Linux terminal (it performs a blocked wait in the terminal), so open a new terminal and run the following commands (the export can be copy/pasted from the previous step).
export KUBECONFIG= . . . as aboveThen run the command:
kubectl proxy
Tip – if you get the message that port 8001 is in use elsewhere, switch to port 8002, with this command:

kubectl proxy ––port=8002

Point a local browser at localhost:8001/ui. You should see the Kubernetes dashboard pages.



Namespaces

Name	Labels	Status	Age
✓ default	-	Active	3 hours
✓ ibm-system	-	Active	3 hours
✓ kube-system	-	Active	3 hours

Upload your IIB docker image to the Container Registry

The Kubernetes environment uses Docker containers to run your application. The use of IIB in Docker containers has been described elsewhere, so we will not cover this here.
Tip – if you get a “permission denied” response to these “bx cr” commands with the local config.json file, you may need to prefix the command with sudo.

Add a Container Registry namespace. This article will use the value “namespace”. Adjust each command as required.

```
bx cr namespace-add namespace
```

Confirm (or retrieve) your Bluemix namespace:

```
bx cr namespace-list
```

Obtain the IMAGE ID of your local docker image, by using this command:

```
sudo docker images
```

Tag your local docker image with details of your Bluemix container registry. This article will use a local docker image called iib10009.

Set the value of “namespace” as required.

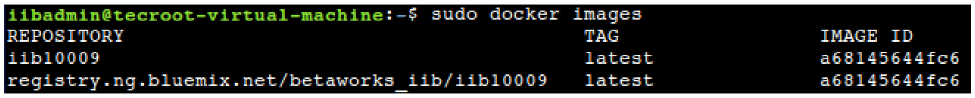
Set the Bluemix region to your own region (for example, replace “ng” with “eu-gb” … etc).

Set the value of “dockerImageID” to the ID of your own image (a68145644fc6 in the example show below).

```
sudo docker tag dockerImageID registry.ng.bluemix.net/namespace/iib10009
```

Check it worked:
sudo docker images

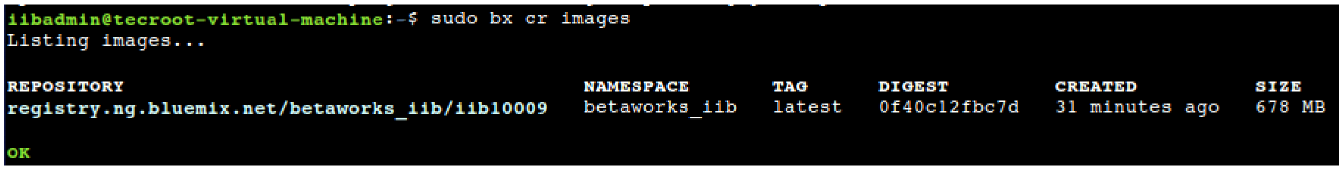
You should see a new image, with the tag that you just specified.



```
iibadmin@tecroot-virtual-machine:~$ sudo docker images
REPOSITORY                                      TAG                 IMAGE ID
iib10009                                        latest              a68145644fc6
registry.ng.bluemix.net/betaworks_iib/iib10009  latest              a68145644fc6
```

Push the image to Bluemix (again, set "namespace" and Bluemix region to your own values):
sudo docker push registry.ng.bluemix.net/namespace/iib10009Depending on your network bandwidth, this will take some time, typically 5-15 minutes. Future uploads of the docker images which contain incremental changes will be much quicker, due to the layered construction of docker images.

Check the uploaded docker images:
bx cr images



```
iibadmin@tecroot-virtual-machine:~$ sudo bx cr images
Listing images...

REPOSITORY                                      NAMESPACE      TAG      DIGEST         CREATED          SIZE
registry.ng.bluemix.net/betaworks_iib/iib10009  betaworks_iib  latest   0f40c12fbc7d   31 minutes ago   678 MB

OK
```

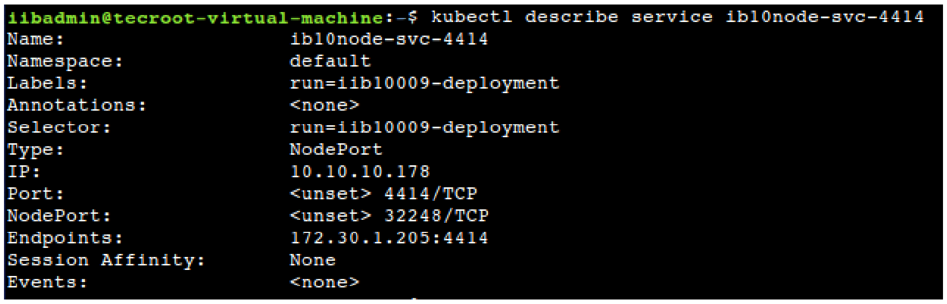
Create a Kubernetes deployment and service for the IIB node

To run an IIB integration node in a basic Kubernetes cluster, you define a Kubernetes deployment and a Kubernetes service:

Create a Kubernetes deployment for the IIB node by running a command similar to this (the example below assumes your Docker container is expecting environment variables LICENSE and NODENAME):
kubectl run iib10009-deployment
––image=registry.ng.bluemix.net/namespace/iib10009
––env="LICENSE=accept"
––env="NODENAME=IB10NODE"

Create a Kubernetes service for the IIB webadmin port, 4414:
kubectl expose deployment/iib10009-deployment
––type=NodePort
––port=4414
––target-port=4414
––name=ib10node-svc-4414In this example, for readability, we have named this service ib10node-svc-4414; this service only exposes port 4414 for IIB webadmin.
Using this command, Kubernetes will generate a value for the external port (known as the NodePort) in the range 30000-32767 that maps to the IIB internal port of 4414.

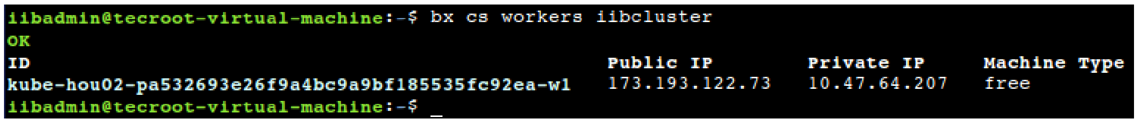
Retrieve service details (ports):
kubectl describe service ib10node-svc-4414The NodePort value is the external port on which the IIB web admin is available.



```
iibadmin@tecroot-virtual-machine:~$ kubectl describe service ib10node-svc-4414
Name:                   ib10node-svc-4414
Namespace:              default
Labels:                 run=iib10009-deployment
Annotations:            <none>
Selector:               run=iib10009-deployment
Type:                   NodePort
IP:                     10.10.10.178
Port:                   <unset> 4414/TCP
NodePort:               <unset> 32248/TCP
Endpoints:              172.30.1.205:4414
Session Affinity:       None
Events:                 <none>
```

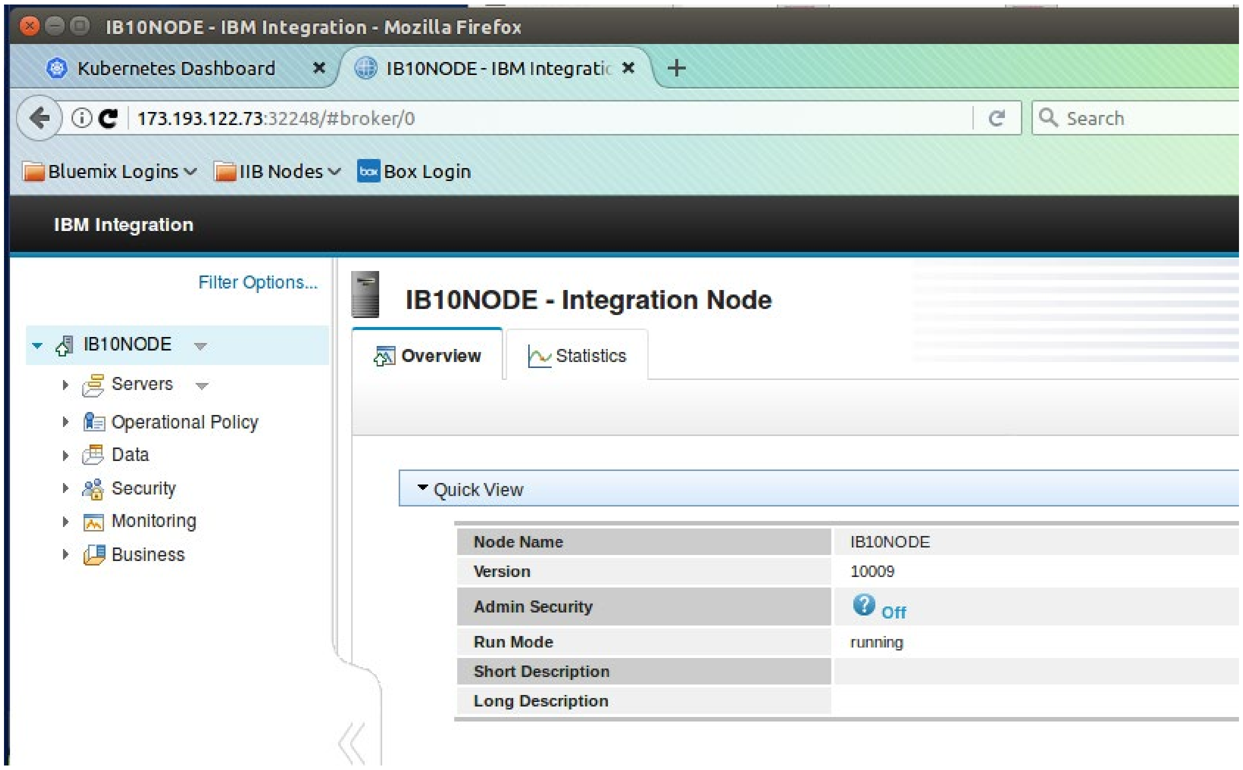
Get the Public IP address of your Kubernetes cluster:

bx cs workers iibcluster



```
iibadmin@tecroot-virtual-machine:~$ bx cs workers iibcluster
OK
ID                                                    Public IP        Private IP      Machine Type
kube-hou02-pa532693e26f9a4bc9a9bf185535fc92ea-w1     173.193.122.73   10.47.64.207    free
iibadmin@tecroot-virtual-machine:~$ _
```

In a browser, connect to the IIB web admin with “PublicIP”:”NodePort”.

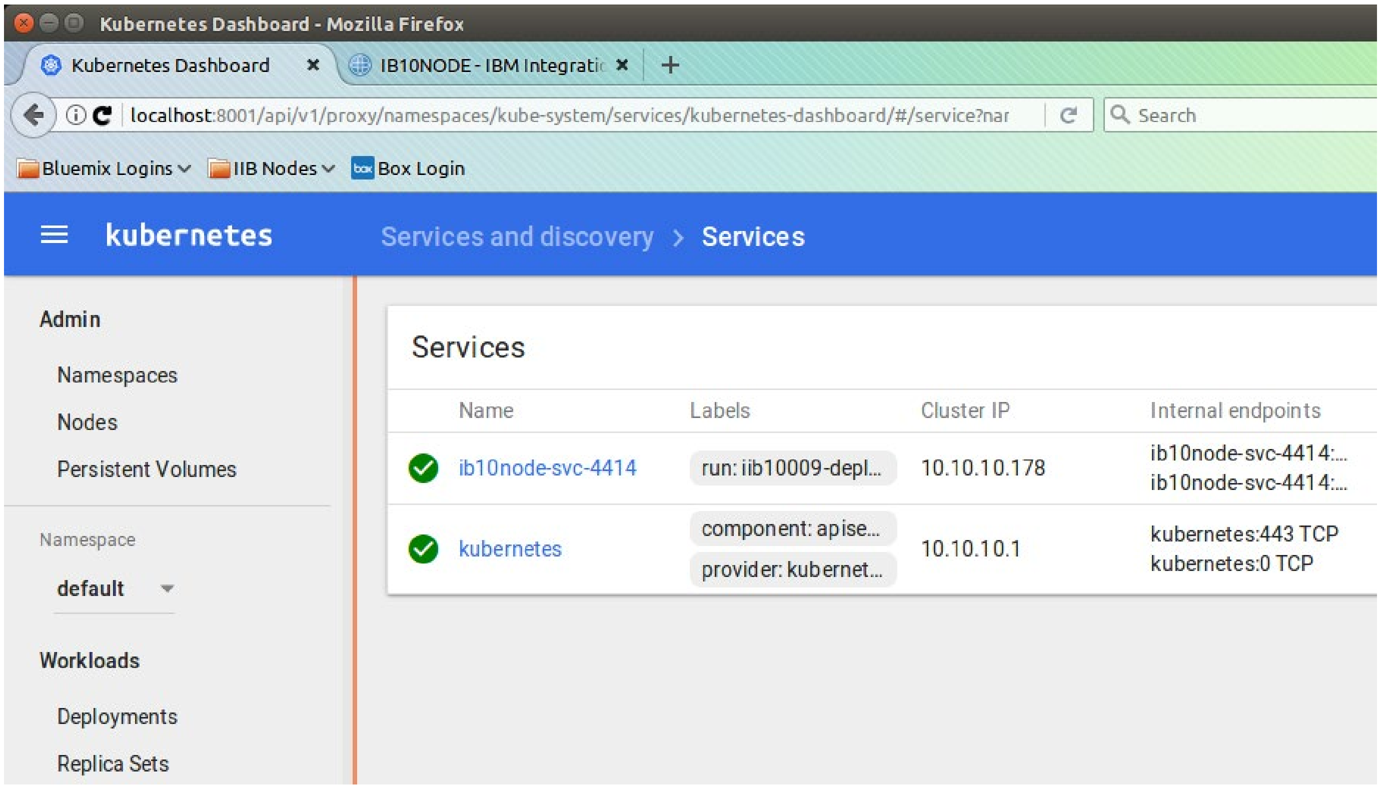


IB10NODE - Integration Node

Node Name	IB10NODE
Version	10009
Admin Security	Off
Run Mode	running
Short Description	
Long Description	

You can use the exposed web admin IP address and port to connect an Integration Toolkit or the Integration API to this IIB node, using the IIB facilities for connecting to remote nodes.

Finally, return to the Kubernetes dashboard. You can see details of the IIB deployment and service by clicking the appropriate links in the navigator. Clicking “ib10node-svc-4414” on the screen below will drill down to the service details.



Kubernetes Dashboard - Mozilla Firefox
Kubernetes Dashboard
IB10NODE - IBM Integratio
localhost:8001/api/v1/proxy/namespaces/kube-system/services/kubernetes-dashboard/#/service?nar
Search
Bluemix Logins
IIB Nodes
Box Login
kubernetes
Services and discovery > Services
Admin
Namespaces
Nodes
Persistent Volumes
Namespace
default
Workloads
Deployments
Replica Sets

Services

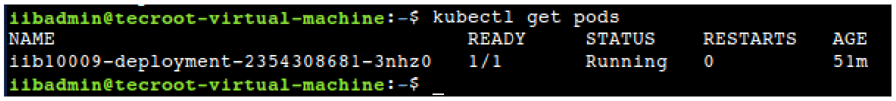
	Name	Labels	Cluster IP	Internal endpoints
✓	ib10node-svc-4414	run: iib10009-depl...	10.10.10.178	ib10node-svc-4414:... ib10node-svc-4414:...
✓	kubernetes	component: apise... provider: kubernet...	10.10.10.1	kubernetes:443 TCP kubernetes:0 TCP

Login to the bash shell of the IIB container

You can login directly to the command interface for the IIB node (ie. the bash shell for the Linux docker container). To do this, run the following commands in the Linux terminal:

Retrieve the Kubernetes pod ID of the IIB deployment:

kubectl get pods



```
iibadmin@tecroot-virtual-machine:~$ kubectl get pods
NAME                                READY     STATUS    RESTARTS   AGE
iib10009-deployment-2354308681-3nhz0   1/1       Running   0          51m
iibadmin@tecroot-virtual-machine:~$ _
```

Run the “kubectl exec” command to open a bash shell:

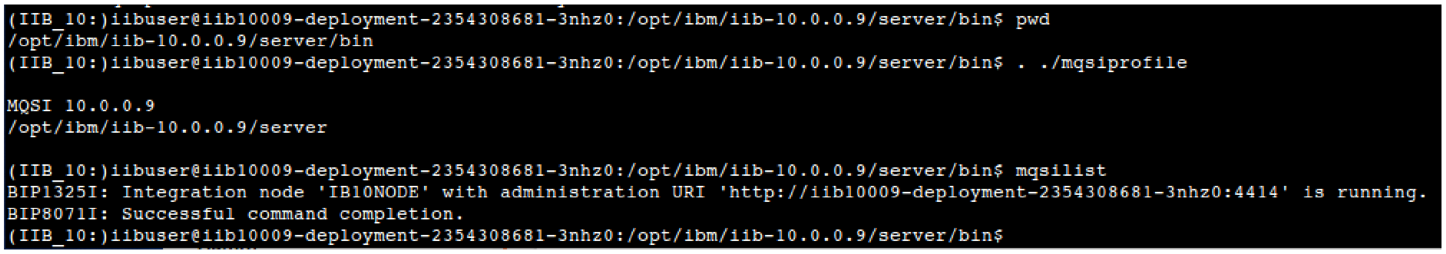
kubectl exec -it podID –– /bin/bash (where podID is the value of the NAME output of the above command – use copy/paste to get this value)

Now run your favourite IIB commands, for example:

cd /opt/ibm/iib-10.0.0.9/server/bin (adjust to your installation directory for IIB)

. ./mqsiprofile

mqsilist



```
(IIB_10:)iibuser@iib10009-deployment-2354308681-3nhz0:/opt/ibm/iib-10.0.0.9/server/bin$ pwd
/opt/ibm/iib-10.0.0.9/server/bin
(IIB_10:)iibuser@iib10009-deployment-2354308681-3nhz0:/opt/ibm/iib-10.0.0.9/server/bin$ . ./mqsiprofile

MQSI 10.0.0.9
/opt/ibm/iib-10.0.0.9/server

(IIB_10:)iibuser@iib10009-deployment-2354308681-3nhz0:/opt/ibm/iib-10.0.0.9/server/bin$ mqsilist
BIP1325I: Integration node 'IB10NODE' with administration URI 'http://iib10009-deployment-2354308681-3nhz0:4414' is running.
BIP8071I: Successful command completion.
(IIB_10:)iibuser@iib10009-deployment-2354308681-3nhz0:/opt/ibm/iib-10.0.0.9/server/bin$
```

Summary

This article has shown you how to define and configure a basic Kubernetes environment in the Bluemix system, how to upload a IIB docker image to the Container Registry, and use this image to start an IIB container, managed by Kubernetes. We used the basic Kubernetes command interface to define a deployment and a service, which allow IIB ports to be exposed and accessed by the IIB web admin browser and the integration toolkit.As you move to more sophisticated Kubernetes configurations, you would normally start to use file-based definitions for your deployments, services and other Kubernetes components. We plan to look at this, and the use of Helm charts, in future articles in this series.

TAGS IBM-INTEGRATION-BUS, KUBERNETES

David Hardcastle

11 comments on"IBM Integration Bus, Kubernetes and the Bluemix Container Service"

pankaj gupta September 16, 2017

these are the main 3 steps I used:

1. Pulled the IIB image from bx registery using docker
docker pull registry.ng.bluemix.net/ibm-integration-bus

2. kubectl run iibv10-deployment –image=registry.ng.bluemix.net/ibm-integration-bus –env=”LICENSE=accept” –env=”NODENAME=IIB10NODE”

3. kubectl expose deployment/iibv10-deployment –type=NodePort –port=4414 –target-port=4414 –name=iibservice

After this I see logs of kubernetes dashboard.

Here are some logs:

```
2017-09-16T05:27:58.568144019Z Sourcing profile
2017-09-16T05:27:58.606164291Z Set environment variable LICENSE=accept to indicate acceptance of license terms and conditions.
2017-09-16T05:27:58.606208805Z
2017-09-16T05:27:58.606216970Z License agreements and information can be viewed by running this image with the environment variable LICENSE=view. You can also set the LANG environment variable to view the license in a different language.
```

I can see these logs in loop on the pod:

1. Created container with docker id 9342a11d029a; Security:[seccomp=unconfined]
2. Started container with docker id 9342a11d029a
3. Back-off restarting failed docker container
4. Error syncing pod, skipping: failed to “StartContainer” for “iib-deployment” with CrashLoopBackOff: “Back-off 10s restarting failed container=iib-deployment pod=iib-deployment-2933641920-tf90k_default(51395781-9a9e-11e7-836d-325b12a74bc6)”

then logs 1,2,4 repeats…

Reply (Edit)

pankaj gupta September 10, 2017

Whenever I run
kubectl proxy
I get “I0909 22:35:46.776846 94318 logs.go:41] http: proxy error: dial tcp [::1]:8080: getsockopt: connection refused” and proxy failed to start..
Any particular reason you can think of…

Reply (Edit)

David Hardcastle September 11, 2017

Hi – not really I’m afraid… I haven’t seen problems like this with the Bluemix Kubernetes service, but i can’t speak for other K8S environments. Presumably you are able to issue other kubectl commands without problems. You could try using the non-default port for the proxy, although it looks like this is a more fundamental problems than simply ports. I might be inclined to reinstall the kubectl plugins ….

Reply (Edit)

pankaj gupta September 14, 2017

The problem I am facing is with kubectl run command.

```
bash-3.2$ kubectl run iibv10image-deployment –image=registry.ng.bluemix.net/pankajnamespace/iibv10image
Error from server (Forbidden): the server does not allow access to the requested resource (post deployments.extensions)
```

Also, when I start the kubectl dashboard and open http://127.0.0.1:8001/ui I get Forbidden: “/ui”

Whereas http://127.0.0.1:8001/api shows this response.

```
{
“kind”: “APIVersions”,
“versions”: [
“v1”
],
“serverAddressByClientCIDRs”: [
{
“clientCIDR”: “0.0.0.0/0”,
“serverAddress”: “169.47.104.210:31406”
}
```

]
}

Here is my kubectl version

bash-3.2$ kubectl version –short
Client Version: v1.5.6
Server Version: v1.5.6-4+abe34653415733

Reply (Edit)

René Paul September 14, 2017

Did you follow the above step: Get the cluster configuration details (required in order to set local configuration context)? I get the same error as you when I forget this step.

Reply (Edit)

pankaj gupta September 15, 2017

There was some issue with my access to bluemix account, once I fixed that I get passed that error. Now I have followed and completed all steps. i am able to open kubectl proxy but I can't get to web admin console of the broker.

When I look at the k8s dashboard I get this error for the pod:

Back-off restarting failed docker container
Error syncing pod, skipping: failed to "StartContainer" for "iibv10-deployment" with CrashLoopBackOff: "Back-off 5m0s restarting failed container=iibv10-deployment pod=iibv10-deployment-1930939146-0zcgx_default(6c48b4d7-998e-11e7-99d7-6e8d2d79eadc)"

(Edit)

David Hardcastle September 15, 2017

I'm speculating a bit here (from personal experience), but it's possible you may not have included the environment variable LICENSE=accept in your Kubernetes deployment. To view the IIB logs in the Kubernetes admin tool, select Pods, and then the Log viewer (three line menu, on the right hand side). That should show you where the IIB node has reached in its startup.

(Edit)

René Paul September 15, 2017

Thanks David! I got stuck with the same error as Pankai and I did include the License environment variable, however I pasted the text from your example above and the quotation marks did not translate properly. Examining the logs as you suggested brought this to light and after typing them manually it worked fine. Strangely an earlier error I got was with the image name, that error too was sloved when I typed it manually… and there were no quatation marks in that, so lesson learned is do not copy paste from webpage in bash shell

(Edit)

René Paul September 18, 2017

Hi David, I now have IIB running in Bluemix as you describe so many thanks for this article! I have also created a REST client application in IIB to play with, and I have deployed this application on the Integration Node in Bluemix. But I do not know what IP-address and Port I should use to talk to the REST client. When testing on a local Integration Node on my laptop I can send a POST request to my REST client at http://localhost:7800/REST/getForecast When I use the remote/Bluemix IP address I use for the IIB web admin with PORT 7800 the request times out. When I read the documentation for the HTTP Input node (https://ibm.co/2fvsDnj) then it states PORT 7800 is the default port for the embedded integration server listener, but I can change that by using the mqsichangeproperties command. However I do not know what port I should use and/or if I also have to expose that port in the container/kubernetes. I hope you can point me in the right direction!

(Edit)

Rene Paul September 20, 2017

Hi David, I was a bit quick with my question. Too many new things to learn… I just needed to expose port 7800 the same way as port 4414 as you explained in the article. So, you saved me once again!

(Edit)

pankaj gupta September 20, 2017

I got it resolved..
The problem was with the env variables while creating deployment. Somehow it was not getting properly set in YAML file when I add that to command line. I manually edit the yaml file and it worked.

Reply (Edit)