Hand Book for BlueMix

January 5, 2016

Chapter 1

Introduction

1.1 Install Cloud Foundry CLI

First yo need to install the

1.2

When you manage images or containers from the command line, you must have your private Bluemix repository URL. You can view your namespace and see how it is used to construct your private Bluemix repository URL. When you manage containers from the Bluemix user interface, you are not required to have this information to complete tasks.

When you manage containers, you should be familiar with namespaces and private repositories.

- Namespace: A unique name to identify your private repository within the Bluemix registry. The namespace is assigned one time for an organization and cannot be changed after it is created.

- Private Bluemix image repository: The Bluemix registry domain with the namespace of the organization's repository. When you run commands, use the full private Bluemix repository when you refer to an image. You can format your image paths like the following example.

```
cf ic run -p <port>–name <containername>registry.ng.bluemix.net/<namespace>/<image name>
```

1.2.1 Docker user

Important: By default, the ice commands must be run with root authentication and therefore, require the prefix sudo. If you want to run docker and ice commands without sudo, run the following command, where ¡CURRENTUSER¿ is the name of the current user, then, if you are on Linux, you must log out then log back in again.

```
sudo usermod -a -G docker <CURRENTUSER>
```

Chapter 2

Docker Files

2.1 The FROM instruction

The first instruction of every Dockerfile needs to be the FROM instruction

```
FROM image
```

It sets the base image for subsequent instructions. Example:

```
FROM ubuntu
```

2.2 The RUN Instruction

The RUN instruction will execute any commands on the current image and commit the results. The resulting committed image will be used for the next step in the Dockerfile. Layering RUN instructions and generating commits conforms to the core concepts of Docker where commits are cheap and containers can be created from any point in an images history, much like source control.

```
RUN command
```

RUN command is equivalent to the docker commands: docker run image command + docker commit container_id, Where image would be replaced automatically with the current image, and container_id would be the result of the previous RUN instruction. Example:

```
RUN apt-get install -y memcached
```

As we saw earlier, we need to have the FROM instruction before any RUN instruction, so that docker knows from which base image to start running the instruction on. For instance:

```
FROM ubuntu
RUN apt-get install -y memcached
```

Chapter 3

Bluemix Runtimes

3.1 Java Liberty

I need to login to my Bluemix account in order to upload my application.

```
cf login -a https://api.ng.bluemix.net -o organization -s space -u username -p password
```

Now to push the application, you have to use this command:

```
cf push application-name path-to-app -m 512
```

to access the app (which is web app), follow this link:

```
application-name.bluemix.net
```

cf push has the following options:

-d	:	Domain
-m	:	Memory limit (e.g. 256M, 1024M, 1G)
–health-check-type, -u	:	Application health check type (e.g. port or none)
-k	:	Disk limit (e.g. 256M, 1024M, 1G)
-s	:	Stack to use (a stack is a pre-built file system, including an operating system, that can apps)
-t	:	Maximum time (in seconds) for CLI to wait for application start, other server side timeouts may apply
–docker-image, -o	:	docker-image to be used (e.g. user/docker-image-name)
-b	:	Custom buildpack by name (e.g. my-buildpack) or GIT URL (e.g. 'https://github.com/heroku/heroku-buildpack-play.git') or GIT BRANCH URL (e.g. 'https://github.com/heroku/heroku-buildpack-play.git#develop' for 'develop' branch). Use built-in buildpacks only by setting value to 'null' or 'default'
-c	:	Startup command, set to null to reset to default start command
-f	:	Path to manifest
-i	:	Number of instances
-n	:	Hostname (e.g. my-subdomain)
-p	:	Path to app directory or to a zip file of the contents of the app directory
–random-route	:	Create a random route for this app
–no-manifest	:	Ignore manifest file
–no-hostname	:	Map the root domain to this app
–no-route	:	Do not map a route to this app and remove routes from previous pushes of this app.
–no-start	:	Do not start an app after pushing

To get the full list of build packs that are supported by Bluemix, use:

```
cf buildpacks
```

Java App: cf push appname -b liberty-for-java or cf push appname -b java_buildpack
Node.js: cf push appname -b sdk-for-nodejs or cf push appname -b nodejs_buildpack
Ruby: cf push appname -b ruby_buildpack
There are a bunch of other languages supported as well.
For a list head over to https://github.com/cloudfoundry-community/cf-docs-contrib/wiki/Buildpacks.
If for example you wanted to use PHP you would do the following. cf push -b https://github.com/cloudfoundry/php-buildpack.git
If you wanted to do Go you would do the following. cf push appname -b https://github.com/cloudfoundry/go-buildpack.git [http://stackoverflow.com/questions/27506058/not-able-to-push-my-local-app-to-bluemix]
to push/deploy/upload a project to Bluemix use:

```
cf push
```

To get the namespace you need first to login to cf:

```
cf login -u nawfalbluemix1@gmail.com -p Asdf_098 -a api.ng.bluemix.net -
```

then login to **ic** service using:

```
cf ic login
```

to build a container :

```
cf ic build -t registry.ng.bluemix.net/myorg8080/nawfal-node-centos .
```

To list all the available applications:

```
cf apps
```

To delete an application:

```
cf delete APP_NAME
```

To start an application:

```
cf start APP_NAME
```

to stop an application:

```
cf stop APP_NAME
```

to increase the number of instances of an application (i.e. horizontal scale), we need to get the application name first by:

```
cf apps
```

then we need to call this where 2 is the required number of instances:

```
cf scale APP_NAME -i 2
```

To do a vertical scale, increase the size of memory, for instance:

```
cf scale APP_NAME -m 512M
```

to increase the disk size:

```
cf scale APP_NAME -k 10G
```

3.2 SDK for Node.js

3.3 Go Community

3.4 PHP Community

3.5 Python Community

3.6 Ruby Community

3.7 Community builpacks Community

Chapter 4

Boilerplate

4.1 Internet of Things Foundation Starter

IBM Bluemix provides an Internet of Things (IoT) starter application for you to get up and running with an Internet of Things Foundation (IoTF) application, by using Node-RED in Bluemix. You can try the sample flow with a simulator and customize it for your own devices [1].

Node-RED provides a browser-based flow editor that makes it easy to wire together devices, APIs, and online services by using the wide range of nodes in the palette. Flows can then be deployed to the Node.js runtime environment with a single click. Included in this sample is a ready-made flow that can process temperature readings from a simulated device. The flow checks these temperature readings against a threshold, then indicates whether the temperature is safe. Use this sample as a starting point for creating your own IoT flows.

4.1.1 Usage

To use the Internet of Things starter from the Bluemix user interface, complete the following steps:

1. Create an Internet of Things starter application by taking the following steps:
    (a) Select the Internet of Things Foundation Starter from the boilerplates section of the catalog, and click View More.
    (b) Provide the application name, modify the host name, if required, and click Create.
2. After a short wait, your application starts.

When your application starts, click the Routes URL or enter this URL in a browser:

```
http://<yourhost>.mybluemix.net
```

You arrive at the Node-RED for Internet of Things landing page.

Click Go to your Node-RED flow editor.

You see a ready-made flow that can process temperature readings from a simulated device.

4.1.2 Set up the simulated device

Guidelines on how to set up the simulated device

1. To open a simulator, browse to IoT Sensor in a new browser window or on a mobile device.
   (a) Alternatively, enter this URL in a browser: `http://quickstart.internetofthings.ibmcloud.com/iotsensor`
2. Click the MAC address in the upper right hand corner of the simulator to visualize the simulator data in the Internet of Things application. The MAC address is something like AF:01:C7:BB:E5:72
   (a) Adjust the humidity, object temperature, and temperature on the simulator to see the chart update in the start screen.
3. Return to your Node-RED workspace and double-click the IBM IoT App In node to open the configuration dialog. In the Authentication type field, select start from the pull-down list. Remove the colons from the MAC address and enter it in the Device ID field and click OK. Make sure that the MAC address is entered in lowercase, and that there are no leading or trailing space characters.
4. Look for the deploy button in the upper right hand corner of your Node-RED workspace. The deploy button is now red; click it to deploy your flow.
5. Open the debug pane on the right. You will see that the flow is generating Temperature Status messages.
6. Increase the temperature value on the simulator to see the messages change in the debug pane. Note that a different message appears if the temperature exceeds 40 degrees.

4.1.3 Modifying the sample flow

Now you can modify your working flow to achieve business goals. Here are some examples:

1. Double-click the temp thresh node and you see the rules that are used to determine whether the temperature is in safe limits or not. You can modify the behavior of the flow by changing these rules, clicking OK, and then clicking the red Deploy button to change the running instance of the flow.

2. Double-click the temp node to see how it extracts the temperature value from the raw JSON data that is coming in from the device. You might modify this node to extract other values, such as humidity, or you might create a second function node to extract humidity and wire that up to the IBM IoT App In node in parallel to the existing temp node.

3. Wire in a node that notifies mobile users when the temperature threshold is exceeded. For more information see, Using the IBM Push service in Node-RED.

4.1.4 Using your own devices

You can create applications with devices registered to the IBM Internet of Things Foundation.

Sign up to the Internet of Things Foundation through Bluemix by using the Internet of Things Service.

To create applications with devices registered to the Internet of Things Foundation:

1. Bind an instance of the Internet of Things service to the application.

2. Register devices to your IoTF organization by starting IoTF from the dashboard that is provided by this service.

3. Configure the IBM IoT App In node with the identification of your device, then in the Authentication type field, select Bluemix service. You notice that this node offers more options now that your Node-RED flow is bound to the IBM Internet of Things Foundation Service. You can click the Any check box to select all your devices.

4. You can also use the IBM Iot App Out node to send commands to your registered device.

5. For more information, see Internet of Things Foundation docs.

4.1.5 Securing the flow editor

By default, the editor is open for anyone to access and modify flows. Follow these steps to password-protect the editor:

1. In the Bluemix dashboard, select the Environment Variables page for your application.

2. Add the following user-defined variables:
    - NODE_RED_USERNAME - the username to secure the editor with
    - NODE_RED_PASSWORD - the password to secure the editor with
3. Click Save.

4.1.6 Adding nodes and packages to Node-RED in IBM Bluemix

Adding more nodes or packages to your Node-RED instance within the Internet of Things Foundation boilerplate adds extra functions.

Packages contain nodes and any required dependencies. Adding nodes and packages to your Node-RED instance within IBM Bluemix increases the functionality of your Internet of Things Foundation boilerplate. The same procedure can be used to add either packages or nodes.

1. Open your Internet of Things Foundation application in the IBM Bluemix dashboard.
2. Click Start Coding in the menu bar on the left.
3. Follow the steps for Customizing your Node-RED intsance in the Start Coding page to download your application.
4. Open your local version of package.json and add the package or node name you wish to add to the Dependencies section, followed by the version number. ”node-red-contrib-mqttssl”:”latest”
5. Using the Cloud Foundry Command Line Interface, push your application to IBM Bluemix to update it. After you push the application to IBM Bluemix, your application will be restaged.
6. The new package or node is now available in the Node-RED palette.

You have added a node or package to your Node-RED instance in IBM Bluemix.

4.2 Node-RED

Bibliography

[1] Creating apps with the internet of things starter application. `https://www.ng.bluemix.net/docs/#starters/IoT/iot500.html#iot500`.