Quick2Cloud for Cloud Systems

Built on IBM Cloud

Quick2Cloud College - Course 102

Running the Application Locally on your Desktop

Now that your sample application is running in the Cloud you may want the option to run it locally on your desktop so you can locally develop and test. Actually, IBM Cloud supports two types of development:

1. Locally on the Desktop
2. Directly in the IBM Cloud

In this Course we will continue with the local aspect with developing directly in the Cloud covered in **Course 103** on Continuous Delivery.

Before we can run the sample application locally, you will need to install Node.js which hopefully was accomplished when you were fulfilling the IBM Cloud requirements. If not, click on this link to download and install the latest version of Node.js from their website.

The *Quick2Clouds* Sample Web Application uses Port 8080 as the Listen Port for all incoming HTTPs requests. Although the Cloud, as well as running locally, allows you to pick Ports using environment variables, if you choose Port 8080 in both runtimes then the same code can handle both environments. This is accomplished by the use of the following Node.js function found in the sample application.

```
http.createServer .listen(8080);
```

This can be seen in the sample application around line 113.

IBM Cloud, after generating you a URL for your application, will route all requests to your URL to Port 8080 so things work smoothly.

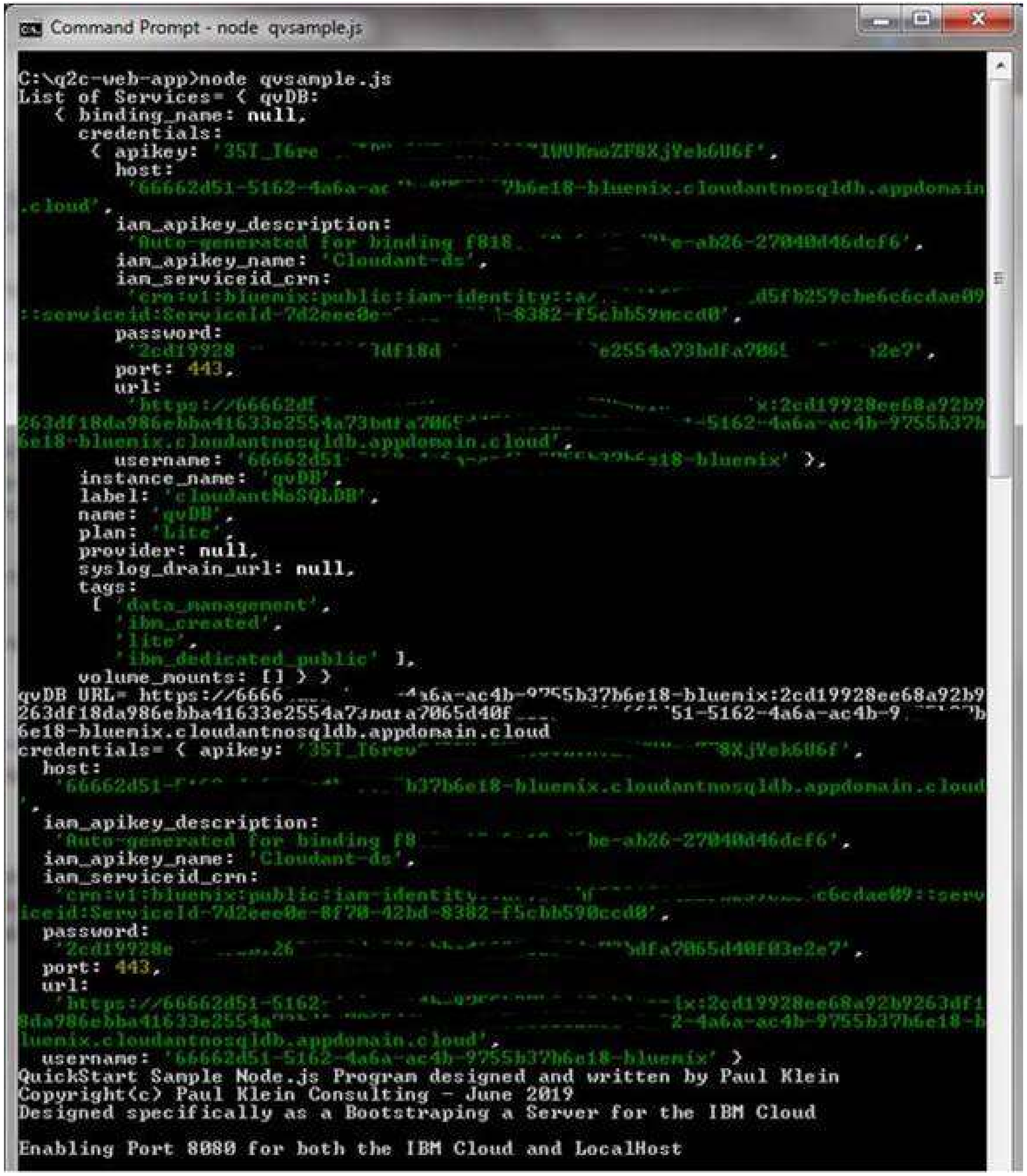
When preparing to run the application locally follow these commands and link:

1. cd q2c-web-app
2. node qvsample.js

3. http://localhost:8080

Your web application has a redirect that will display index.htm and now should be displaying the sample home page in your browser. Instead of communicating with the server running in the IBM Cloud, it is communicating with the server running locally that is executing as a process on your Desktop. **Localhost** is the key**,** it□s the desktop loopback address telling the desktop□s network the server resides locally.

Results after typing node qvsample.js

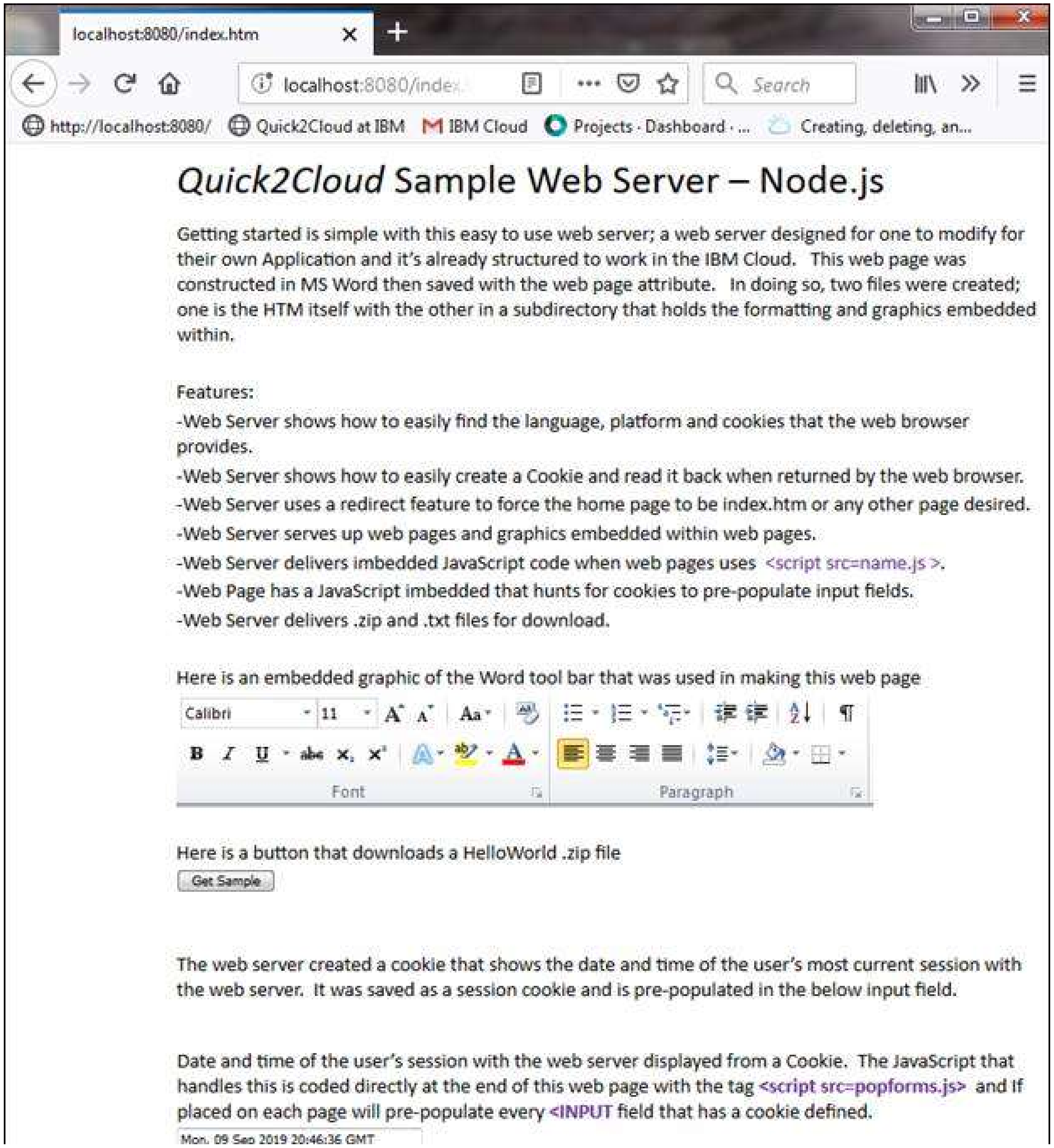

Command Prompt - node qvsample.js

```
C:\q2c-web-app>node qvsample.js
List of Services= { qvDB:
   { binding_name: null,
     credentials:
      { apikey: '35I_I6re [redacted]1WUKmoZF8XjYek6U6f',
        host:
         '66662d51-5162-4a6a-ac[redacted]7b6e18-bluemix.cloudantnosqldb.appdomain
.cloud',
        iam_apikey_description:
         'Auto-generated for binding f818[redacted]e-ab26-27040d46dcf6',
        iam_apikey_name: 'Cloudant-ds',
        iam_serviceid_crn:
         'crn:v1:bluemix:public:iam-identity::a/[redacted]d5fb259cbe6c6cdae09
::serviceid:ServiceId-7d2eee0e-[redacted]-8382-f5cbb59mccd0',
        password:
         '2cd19928[redacted]3df18d[redacted]e2554a73bdfa706[redacted]2e7',
        port: 443,
        url:
         'https://66662df[redacted]x:2cd19928ee68a92b9
263df18da986ebba41633e2554a73bdfa7065[redacted]-5162-4a6a-ac4b-9755b37b
6e18-bluemix.cloudantnosqldb.appdomain.cloud',
        username: '66662d51[redacted]e18-bluemix' },
     instance_name: 'qvDB',
     label: 'cloudantNoSQLDB',
     name: 'qvDB',
     plan: 'Lite',
     provider: null,
     syslog_drain_url: null,
     tags:
      [ 'data_management',
        'ibm_created',
        'lite',
        'ibm_dedicated_public' ],
     volume_mounts: [] } }
qvDB URL= https://6666[redacted]4a6a-ac4b-9755b37b6e18-bluemix:2cd19928ee68a92b9
263df18da986ebba41633e2554a73bdfa7065d40f[redacted]51-5162-4a6a-ac4b-9[redacted]b
6e18-bluemix.cloudantnosqldb.appdomain.cloud
credentials= { apikey: '35I_I6rev[redacted]8XjYek6U6f',
  host:
   '66662d51-[redacted]b37b6e18-bluemix.cloudantnosqldb.appdomain.cloud
',
  iam_apikey_description:
   'Auto-generated for binding f8[redacted]be-ab26-27040d46dcf6',
  iam_apikey_name: 'Cloudant-ds',
  iam_serviceid_crn:
   'crn:v1:bluemix:public:iam-identity[redacted]c6cdae09::serv
iceid:ServiceId-7d2eee0e-8f70-42bd-8382-f5cbb590ccd0',
  password:
   '2cd19928e[redacted]26[redacted]bdfa7065d40f03e2e7',
  port: 443,
  url:
   'https://66662d51-5162-[redacted]ix:2cd19928ee68a92b9263df1
8da986ebba41633e2554a[redacted]2-4a6a-ac4b-9755b37b6e18-b
luemix.cloudantnosqldb.appdomain.cloud',
  username: '66662d51-5162-4a6a-ac4b-9755b37b6e18-bluemix' }
QuickStart Sample Node.js Program designed and written by Paul Klein
Copyright(c) Paul Klein Consulting - June 2019
Designed specifically as a Bootstraping a Server for the IBM Cloud

Enabling Port 8080 for both the IBM Cloud and LocalHost
```

The above image is of the locally running sample application displaying the environment variables passed to the CFENV facility in the command shell.

By entering or clicking on http://localhost:8080 in a web browser you will render the sample application□s home page



Quick2Cloud Sample Web Server – Node.js

Getting started is simple with this easy to use web server; a web server designed for one to modify for their own Application and it's already structured to work in the IBM Cloud. This web page was constructed in MS Word then saved with the web page attribute. In doing so, two files were created; one is the HTM itself with the other in a subdirectory that holds the formatting and graphics embedded within.

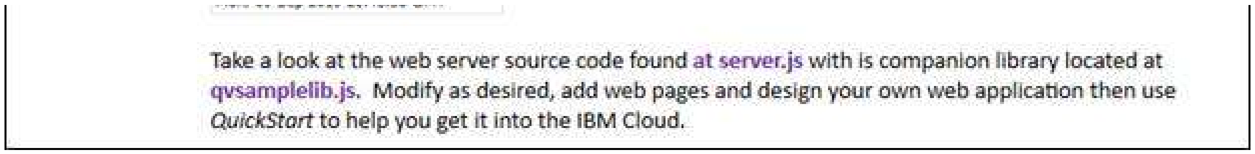
Features:
-Web Server shows how to easily find the language, platform and cookies that the web browser provides.
-Web Server shows how to easily create a Cookie and read it back when returned by the web browser.
-Web Server uses a redirect feature to force the home page to be index.htm or any other page desired.
-Web Server serves up web pages and graphics embedded within web pages.
-Web Server delivers imbedded JavaScript code when web pages uses <script src=name.js >.
-Web Page has a JavaScript imbedded that hunts for cookies to pre-populate input fields.
-Web Server delivers .zip and .txt files for download.

Here is an embedded graphic of the Word tool bar that was used in making this web page

Here is a button that downloads a HelloWorld .zip file

The web server created a cookie that shows the date and time of the user's most current session with the web server. It was saved as a session cookie and is pre-populated in the below input field.

Date and time of the user's session with the web server displayed from a Cookie. The JavaScript that handles this is coded directly at the end of this web page with the tag <script src=popforms.js> and If placed on each page will pre-populate every <INPUT field that has a cookie defined.


Take a look at the web server source code found at server.js with is companion library located at qvsamplelib.js. Modify as desired, add web pages and design your own web application then use *QuickStart* to help you get it into the IBM Cloud.

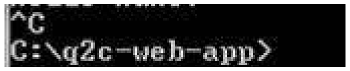
Running locally allows you to make changes to qvsample.js, or any of its support files and see the results immediately just by restarting the local web server. This is a much faster way to develop then having to push the new code up to the Cloud and restarting your Cloud application every time. It also does not affect your Customers who you don□t want to introduce changes to until you are ready.

Develop and test your sample application locally, and then when it is ready, push it up to the IBM Cloud using the script in **Course 100** that was generated for you.

fyi
Changes to web pages written with the extension .htm does not require a local server restart. The sample application, as written, always gets a new copy of the .htm pages when they are referenced. It makes development kind of easy.

However, any file that ends in .js is cached by Node.js which means any changes to any previously referenced .js file will require a local server restart for it to take effect. It□s also advisable to clear the browser cache from time-to-time, especially when you think you have an unexplainable error.

Best way to stop and restart the locally running web server is the use of Ctrl^c in the command line shell.


```
^C
C:\q2c-web-app>
```

Then just type node qvsample.js once again and the local server will restart.

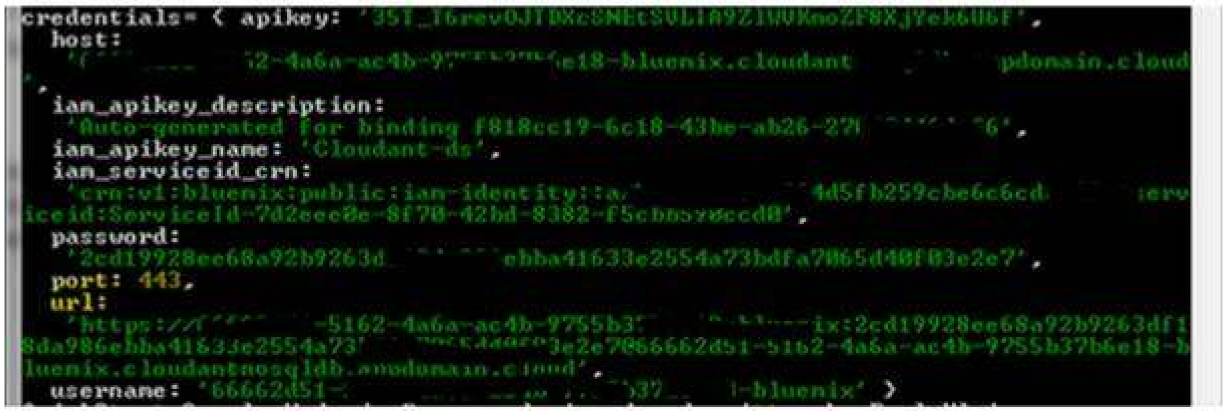
Once again, a last word on environment variables!

Let us briefly cover again how the locally running server gets environment variable information for it□s CFENV facility; variables that are typically only available when running in the Cloud. Locally on the desktop are two files, VCAP_APPLICATION.json and VCAP_Services.json which were both automatically generated for you by the q2cloud command during the application push and Service provisioning processes.

They contain json structures obtained from the Cloud that contains the credentials, userIDs and passwords needed to access the application and it□s Services.

Your sample web application has a facility inside, **CFENV**, which requires these two files so it can provide the local representation of the same environment variables that it would have if running in the Cloud. This is necessary because your locally running application can, and does, access the Services in the Cloud. These two json files are provided to you as an assist by *Quick2Cloud* to make running locally a lot easier.

Your sample web application is coded to put out a number of log messages when it starts up and these messages are seen in the command line shell window. Any messages logged from the application using the console.log() function will appear just as written and that includes the logging of the environment variables from the json files.



```
credentials= { apikey: '35T_I6revOJTDXcSNEtSULIA9Z1WUKnoZF8XjYek6U6F',
  host:
   '[redacted]2-4a6a-ac4b-9[redacted]e18-bluemix.cloudant[redacted]pdomain.cloud
',
  iam_apikey_description:
   'Auto-generated for binding f818cc19-6c18-43be-ab26-27[redacted]6',
  iam_apikey_name: 'Cloudant-ds',
  iam_serviceid_crn:
   'crn:v1:bluemix:public:iam-identity::a/[redacted]4d5fb259cbe6c6cd[redacted]:serv
iceid:ServiceId-7d2eee0e-8f70-42bd-8382-f5cbb59a0ccd0',
  password:
   '2cd19928ee68a92b9263d[redacted]ebba41633e2554a73bdfa7065d40f03e2e7',
  port: 443,
  url:
   'https://[redacted]-5162-4a6a-ac4b-9755b3[redacted]ix:2cd19928ee68a92b9263df1
8da986ebba41633e2554a73'[redacted]3e2e7066662d51-5162-4a6a-ac4b-9755b37b6e18-b
luemix.cloudantnosqldb.appdomain.cloud',
  username: '66662d51-[redacted]37[redacted]-bluemix' }
```

What appears in the shell image above will also appear in the Cloud under the Log link on the IBM Dashboard Application Detail display. Same code, same logged messages. Obviously, you would never allow anyone to see this information because they would then have your database login information. It is only rendered here to make a point, which is, the same application code running in the Cloud can run locally with the automatic assist of *Quick2Cloud*.

At this point you should have an understanding and examples of pushing an application into the Cloud and connecting up a Service. You have examples of how an application uses VCAP Environment Variables and how it interacts with a database Service. This may be all that is needed for you to be successful with the IBM Cloud. However, if you want to mature your application and get some commercial discipline around it - it is time to move forward with understanding Continuous Delivery and what it provides to you.

Repositories bypass

It is time for our Course to introduce the concept of a local repository in preparation for designing this application for Continuous Delivery. Entering the world of Continuous Delivery is not required for running in the Cloud but it does bring a certain discipline and maturity to what you just accomplished. If you are planning to have Customers or other programs use your application and want to provide ongoing capability, you will want a way to manage its development and deployment that is both practical and non-disruptive.

Let us continue the running locally concept by beginning a discussion on how a local repository brings benefits to your application. Select Create a local repositoy on the Main Menu to continue with your college courses.

Main Menu