IBM Predictive Modeling Service for Bluemix
General Discussion on Application Development

Prepare to Develop the Application

Bluemix makes it very easy to get a new application started. It is worth using the Bluemix 'create an app' at least once to get a good feel for what is involved.

1. Go to the Bluemix web site. We'll be using the 'Development instance' in this example which is at https://console.stage1.ng.bluemix.net/

2. Log in using your account details or sign up now for an IBM ID and access to Bluemix, it is free.

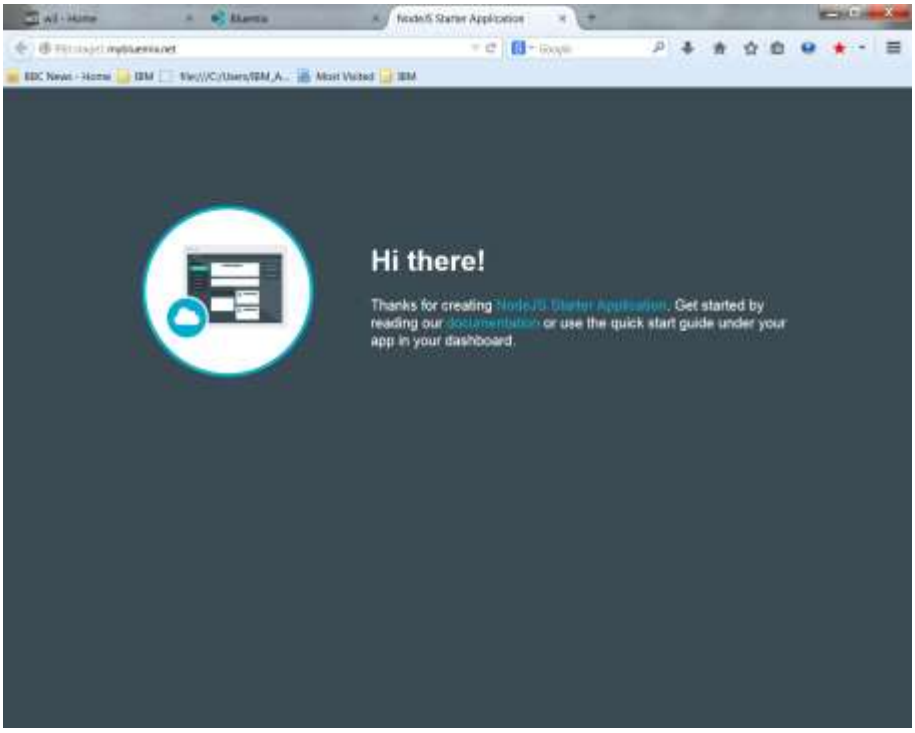
3. You may create an 'org' and 'space' for doing this sample development in, the names used are up to you.

4. Once on the dashboard for your development space push the 'Create an App' button

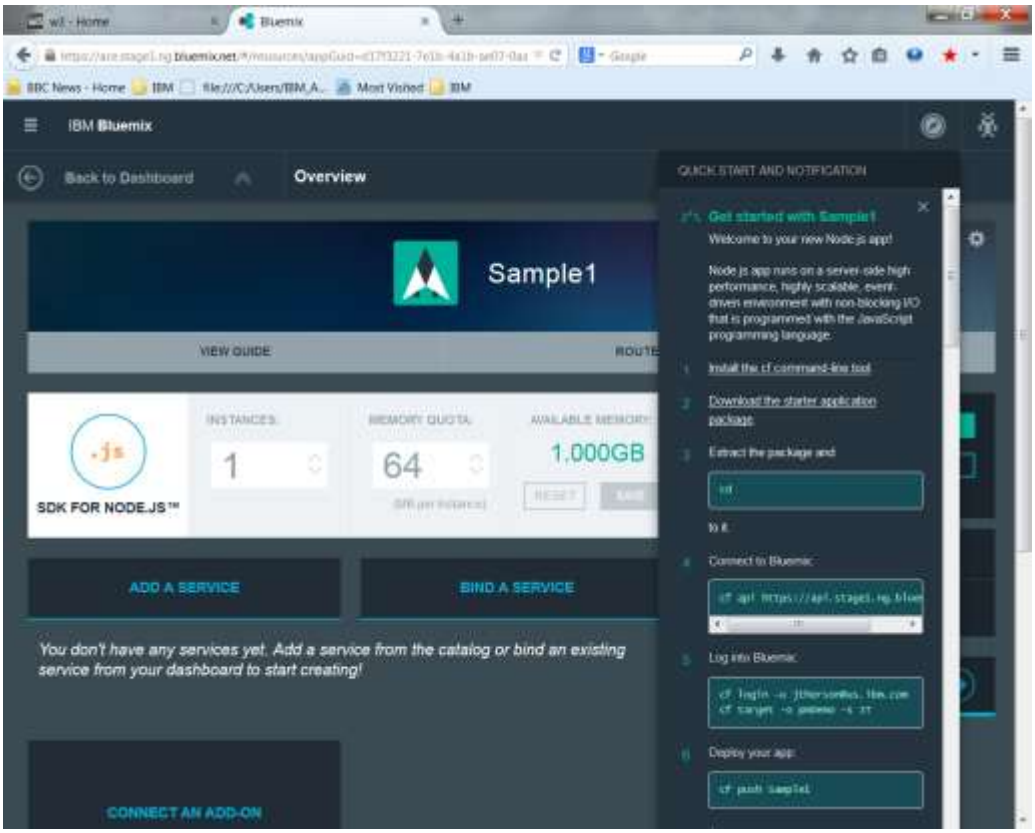
5. In this example we will create a NodeJS application but any language capable of making REST service calls will do.

6. The charges for the application in its simple ‘create’ form are listed on the panel where you give your application a name and name your ‘host’. Your total charges will vary based on the ‘plan’ you choose, the amount of memory your application needs, the number of instances of your application you need deployed and the services you bind to it.

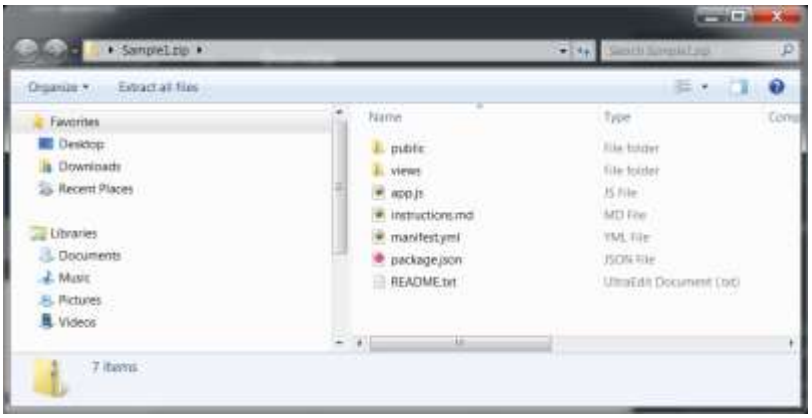
7. A number of things happen when you push ‘Create’ on your new application. A viable ‘shell’ application for NodeJS is created and ‘pushed’ out to Bluemix. The DNS routing entries that let you get to this application are entered. Then the application is started, a basic ‘Hello World’.



8. The fun begins when we download this application and work with in on our desktop. When we select ‘View Guide’ on our application we get a help in installing the CF command line tool and downloading our sample application by clicking on the ‘Download the starter application package’ which in our case is the Hello World shell for Sample1.



9. What is in this application bundle when we download it?



We get the basic framework for a NodeJS application (app.js and package.json) as well as the manifest.yml used when we ‘push’ this application up to Bluemix. There is also a ‘README.txt’ and ‘instructions.md’ to help you take your next steps in building a NodeJS application.

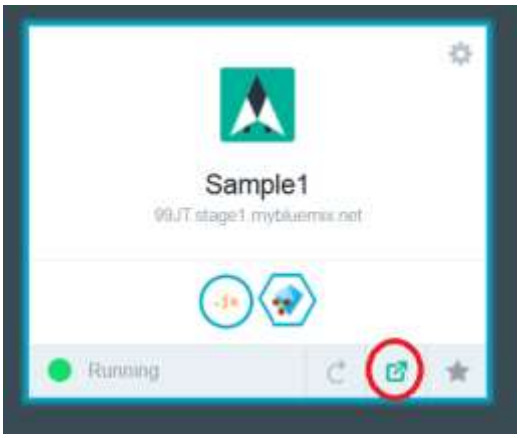
Push the Modified Application Back to Bluemix

Once your NodeJS application is ready to test push the update to Bluemix using the

'cf push <application name>'

command from your development directory. The entire bundle is transferred up to Bluemix and you will be notified as it goes through all phases of deployment and finally restarts.

Once restarted the application can be launched right from the Bluemix dashboard by pushing the 'open app in a new page' button.

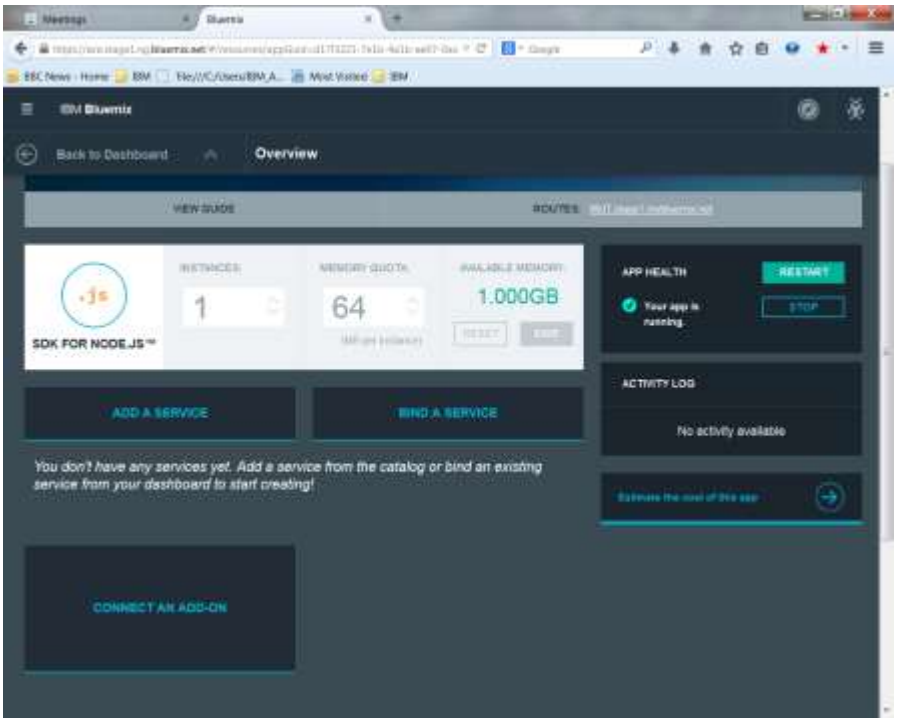


Testing Your Bluemix Application During Development

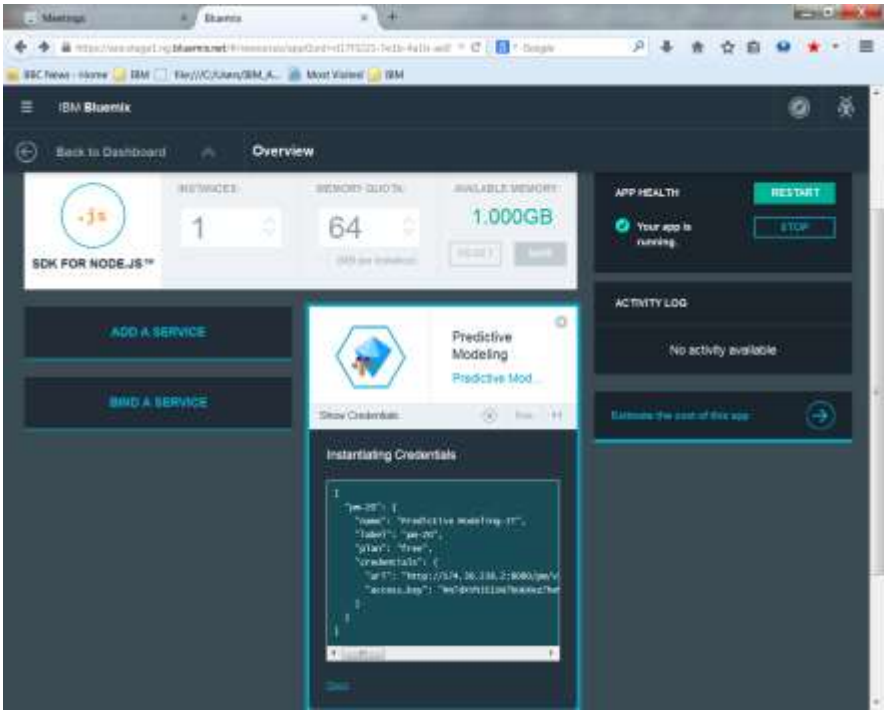
You may provision an instance of the IBM Predictive Modeling service at any point in time and get the connectivity information you need to test your application from your desktop. You may also 'push' your application back up to Bluemix at any point in time and test it there.

To test your application from your desktop you will need a compatible development environment. In the example we've been looking at here this would mean a NodeJS server install.

Use the “Bind a Service” button on your Bluemix application to connect it to your service instance.



Once the application is bound you can click on the “Show Credentials” button on the bound service icon to get the connectivity information you need to test your application from your desktop.



Modify the NodeJS example app.js file to use either the Bluemix provided environment vairables VCAP_APP_HOST, VCAP_APP_PORT and VCAP_SERVICES or values you set to test from your desktop.

```
var host = (process.env.VCAP_APP_HOST || 'localhost');
var port = (process.env.VCAP_APP_PORT || 3000);
var services = JSON.parse(process.env.VCAP_SERVICES || "{}");

// we are looking for the IBM Predictive Modeling service instance which has the ID of ‘pm-20’
var service = (services['pm-20'] || "{}");

// the credentials will give us the URL to the service instance as well as the ‘accesskey’ authentication
var credentials = service.credentials;
if (credentials != null) {
        env.baseURL = credentials.url;
        env.accessKey = credentials.access_key;
}
else {
        // set these values based on information gathered from bound service
        // in the ‘url’ and ‘access_key’ members of the credentials object
}
```