Getting Started with IBM Bluemix

Hands-On Workshop

Exercise 5d: Adding a REST API and Deploying to Bluemix

Adding the REST API and deploying to Bluemix

Now that the FizzBuzz functionality is created, it needs to be exposed as a REST API. The REST API endpoint will respond to an HTTP GET request that is sent to:

```
http(s)://<host>/fizzbuzz_range/:from/:to
```

1. Implement a Node.js server by using the Express.js framework:
   a. Create a new file named `server.js` in the project directory and add the following code to the file:

```
var express = require("express");
var app = express();

var server_port = 3000;
var server_host = "localhost";

var server = app.listen(server_port, server_host, function () {

  var host = server.address().address;
  var port = server.address().port;

  console.log("Example app listening at http://%s:%s", host, port);

});
```

2. Because the application requires the Express.js framework, install it and add it to the `packages.json` file. Use the command `npm install`:
   a. In the Terminals view, enter Ctrl+C to stop the Mocha tests.

   b. Enter the following command:

```
npm install express –save
```

   c. Refresh the Eclipse content by selecting the project name in Project Explorer view and then pressing F5.

3. Implement the REST API:
   a. Enter the following code after the `var app = express();` line:

```
var FizzBuzz = require("./fizzbuzz");

app.get("/fizzbuzz_range/:from/:to", function (req, res) {
  var fizzbuzz = new FizzBuzz();
  var from = req.params.from;
  var to = req.params.to;

  res.send({
    from: from,
    to: to,
    result: fizzbuzz.convertRangeToFizzBuzz(from, to)
  });
});
```

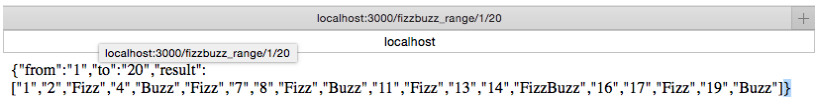
4. Test the API:
   a. Start the NodeJS server by entering the following command in the Terminals view:

```
npm start
```

   You should see a response similar to this output:

```
npm start

> fizzbuzz-w3@0.0.0 start /Users/binnes/git/fizzbuzz-w3
> node server.js

Example app listening at http://127.0.0.1:3000
```

5. Test the API in a browser or command line:
   a. Enter `http://localhost:3000/fizzbuzz_range/1/20` into a browser. You should see the following text in the browser:



   Tip: Instead of testing the API with a browser, you can use the command-line tool cURL:

```
curl -i http://localhost:3000/fizzbuzz_range/1/20
HTTP/1.1 200 OK
X-Powered-By: Express
Content-Type: application/json; charset=utf-8
Content-Length: 150
ETag: W/"96-VBtwqT0jYW1L/R6kdGqqKg"
Date: Mon, 22 Jun 2015 17:43:01 GMT
Connection: keep-alive

{"from":"1","to":"20","result":["1","2","Fizz","4","Buzz","Fizz","7","8",
"Fizz","Buzz","11","Fizz","13","14","FizzBuzz","16","17","Fizz","19","Buz
z"]}
```

   b. Enter Ctrl+C to terminate the server.

6. Modify code for Bluemix and Cloud Foundry:
   a. Before pushing the code to Bluemix, modify the `server.js` file to ensure that the server listens on the host name and port that is specified in the environment variables. Modify the definitions of the server_port and server_host variables to use the VCAP_APP_PORT and VCAP_APP_HOST environment variables:

```
var server_port = process.env.VCAP_APP_PORT || 3000;
var server_host = process.env.VCAP_APP_HOST || "localhost";
```

   b. In the project, create a new file named `manifest.yml`. Then, enter the following content and change the name so that it's unique, such as adding your initials at the start of the name:

```
applications:
```

```
- name: bi-FizzBuzz
```

c. Save the file.

d. In the project, create a new file named `.cfignore`. Then, enter the following content:

```
launchConfigurations/
.git/
node_modules/
npm-debug.log
```

e. Modify the `.gitignore` file so that it has the following content. Use the Navigator view to see files that start with a dot.

```
launchConfigurations/
.project
.settings/
node_modules/
npm-debug.log
```

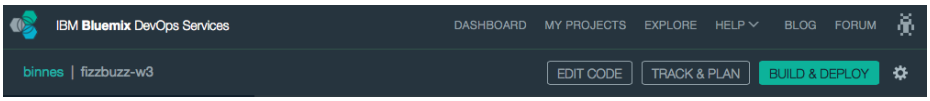
f. Commit the change to the master repo by using the Git Staging view.

Configure a DevOps pipeline to automatically test and deploy code

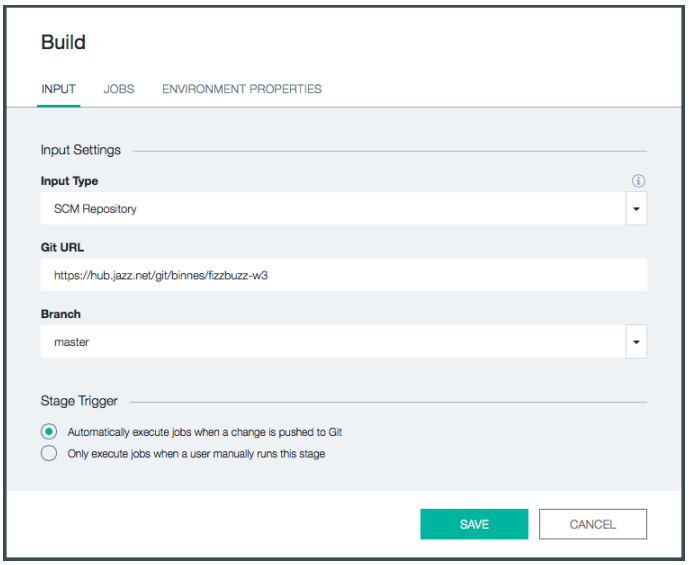
Now that the REST API is working, you must deploy it to IBM Bluemix. Use the Build and Deploy capability in IBM Bluemix DevOps services.

1. Log in to Bluemix DevOps services (https://hub.jazz.net) and from the MY PROJECTS page, select the FizzBuzz project that you created in the previous sections.

2. Build a DevOps pipeline for the project:
   a. Select the **BUILD & DEPLOY** section of the project.



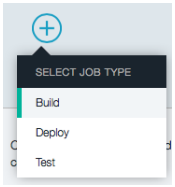
   b. Click **ADD STAGE**.



   c. On the INPUT page, name the stage `Build`. Leave the rest of the fields as default values.

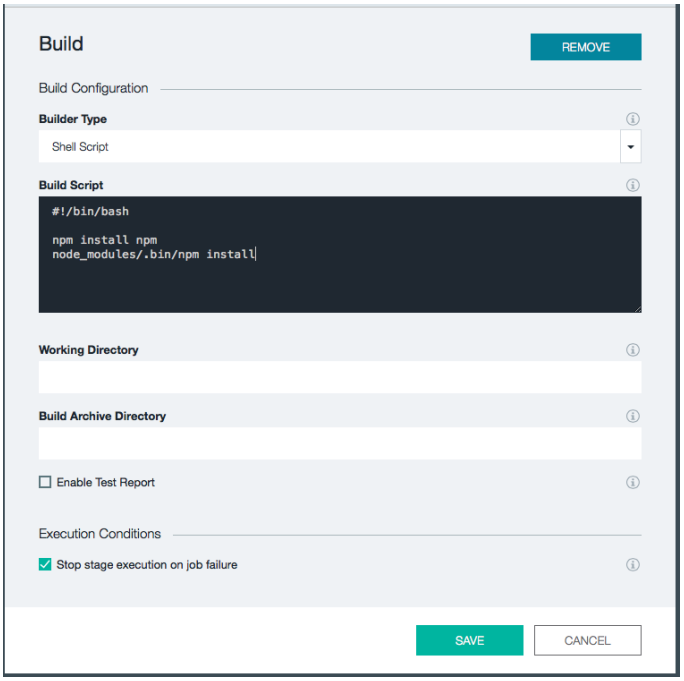
   d. Click the **JOBS** tab.

e. Click **ADD JOB** and then click **Build** as the job type.



f. For the **Builder Type**, select **Shell Script** and enter the script as shown in the screen capture below. Leave the other fields as default values.

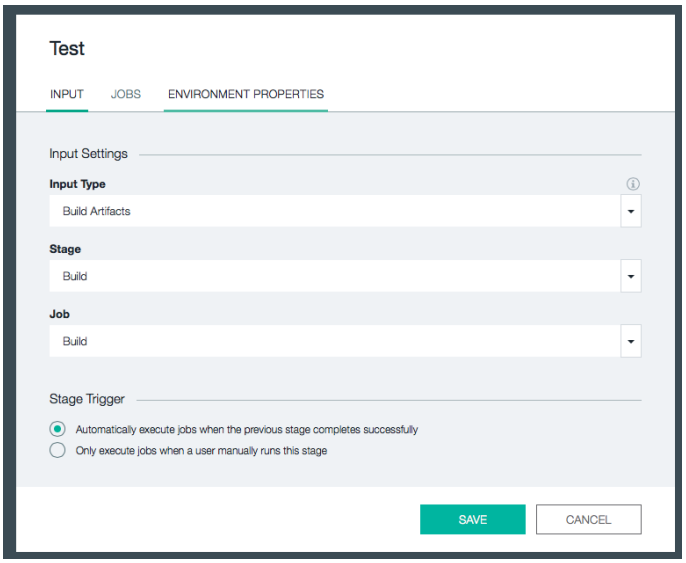
```
#!/bin/bash

npm install npm
node_modules/.bin/npm install
```



g. Click **Save** to save the stage.

h. Click **ADD STAGE** to add another stage in the DevOps pipeline.

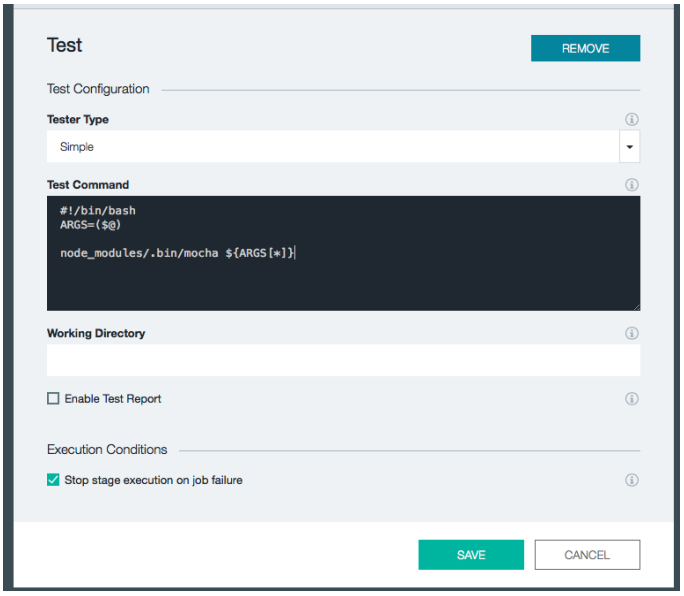
i. On the **Input** tab, name the stage `Test`. Then, ensure that the **Input Type** is **Build Artifacts**, the **Stage** is **Build**, and the **Job** is **Build**.



j. On the **JOBS** tab, click **ADD JOB** and click **Test** as the **Job Type**.

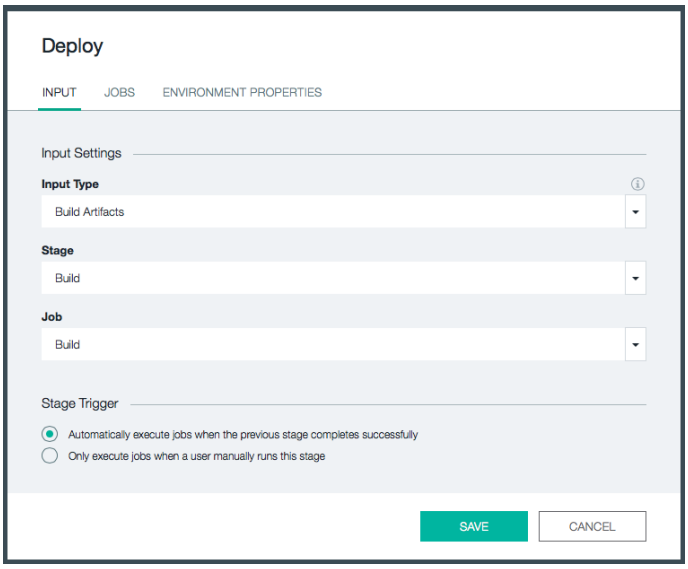
k. In the Test Configuration page, ensure that the type is **Simple** and enter the script as shown in the screen capture.

```
#!/bin/bash
ARGS=($@)

node_modules/.bin/mocha ${ARGS[*]}
```



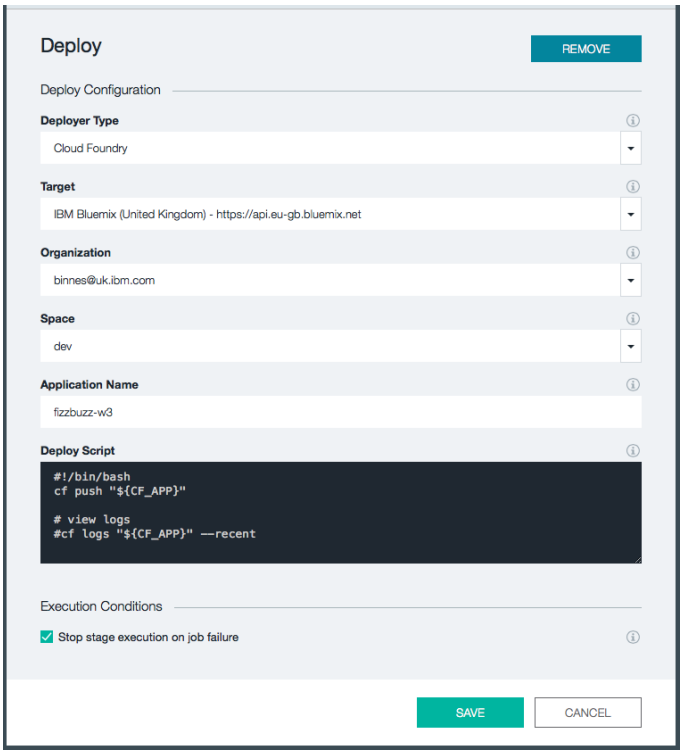
l. Save the stage.

m. Add a third stage and name it `Deploy`. Then, ensure that the **Input Type** is **Build Artifacts**, the **Stage** is **Build**, and the **Job** is **Build**.



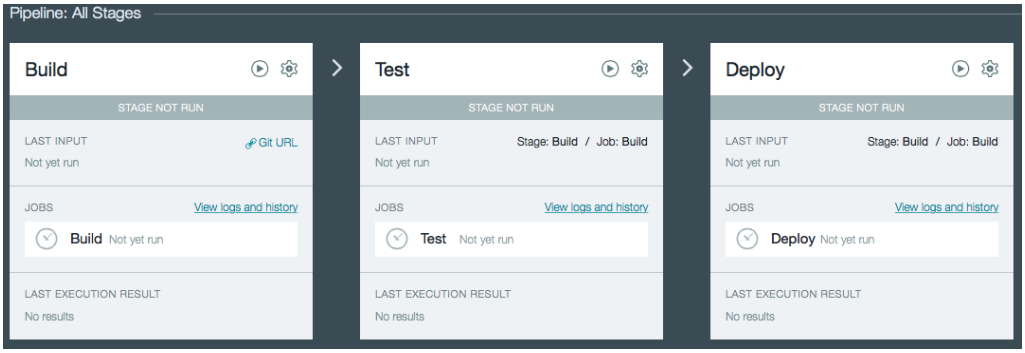
n. On the **JOBS** tab, add a job of type Deploy. Use the default values.

```
#!/bin/bash
cf push "${CF_APP}"

# view logs
#cf logs "${CF_APP}" -recent
```

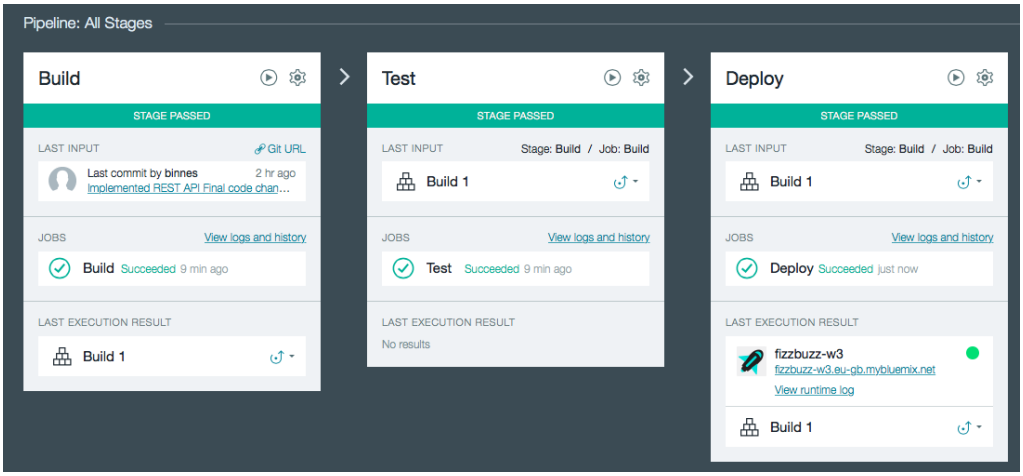


o. Save the stage.

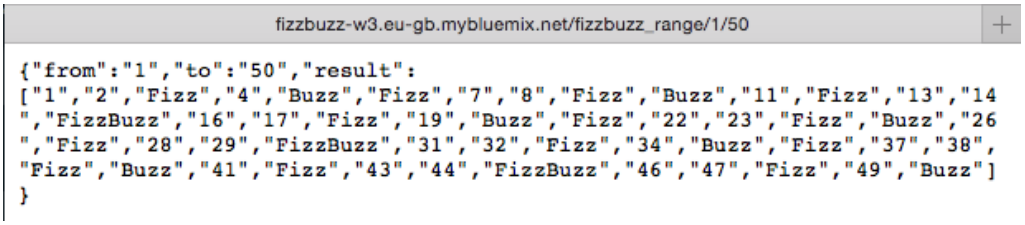
p. Start a new build by clicking **Run** on the Build stage.



You should see the Build stage start to run. After it completes, the Test stage should automatically run and then finally the Deploy stage will run.



q. Click the application URL to launch the application. There is no home page set. However, modify the URL by adding `/fizzbuzz_range/1/50` to request the range from 1 to 50.



The code is not yet complete.

A bug in the code was just delivered. Part of the skill of testing is to know what to test, and in this exercise, a common JavaScript error was not tested for.

If you test for the range 9 – 50, you get no results.

This occurs because of the difference between comparing numbers and comparing strings, for example:

- 9 < 50 is true for numbers
- "9" < "50" is false for strings

As part of the tests, ensure the correct type of comparison is taking place.

Some of the tests are not passing the same parameter types as the REST API call. The test is passing a number, but the API call is passing a string.

r. Rewrite the test for convertRangeToFizzBuzz to use string parameters. Also use a range that will fail a string comparison but pass a number comparison:

```
  describe("convertRangeToFizzBuzz()", function() {
          it("returns in correct order", function() {
            expect(f.convertRangeToFizzBuzz("1", "3")).to.be.eql(["1",
"2", "Fizz"]);
          });

          it("applies FizzBuzz to every number in the range", function()
{
              var spy = sinon.spy(f, "convertToFizzBuzz");

              f.convertRangeToFizzBuzz("9", "50");

              for (var i = 9; i <= 50; i++) {
                expect(spy.withArgs(i).calledOnce).to.be.eql(true,
"Expected convertToFizzBuzz to be called with " + i);
              }
              f.convertToFizzBuzz.restore();
            });
       });
```

The test called “applies FizzBuzz to every number in the range test” is now failing.

s. To fix the problem, ensure that the convertToFizzBuzz function is using numbers not strings:

```
FizzBuzz.prototype.convertRangeToFizzBuzz = function(start, end) {
      var result = [];
      var from = parseInt(start);
      var to = parseInt(end);

      for (var i = from; i <= to; i++) {
        result.push(this.convertToFizzBuzz(i));
      }

      return result;
};
```

t. From the Git Staging view, stage, comment, and commit and push the changes. Then, quickly switch to the IBM Bluemix DevOps Services console for your project in the Build & Deploy settings. You should see a build, test, and deploy being automatically run.