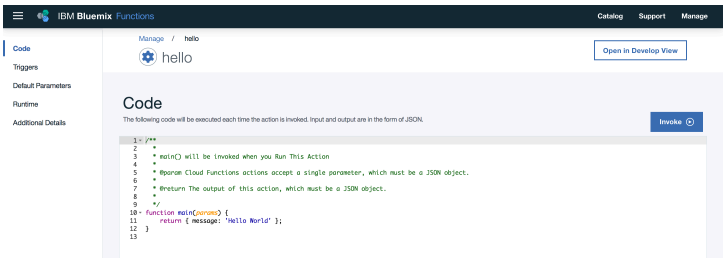
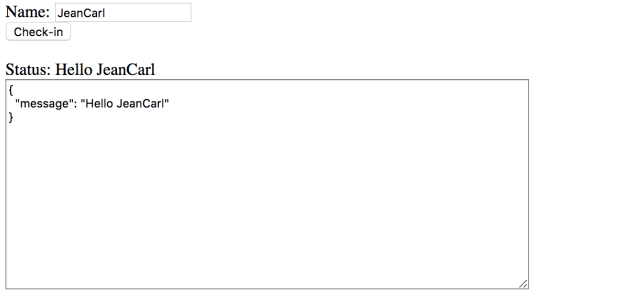
Introduction to Cloud Functions

Hands-On Lab

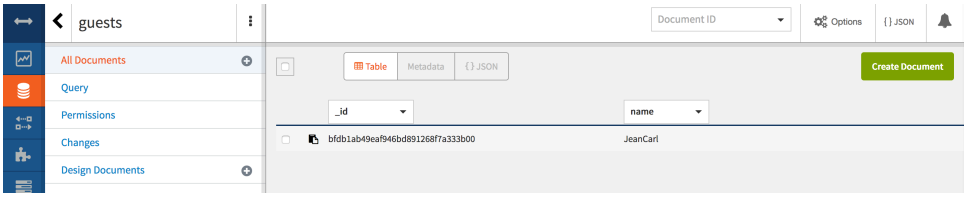
JeanCarl Bisson | jbisson@us.ibm.com | @dothewww



Create an application that takes form data and registers guests in a Cloudant database.



Create a Cloud Function to run code on demand.

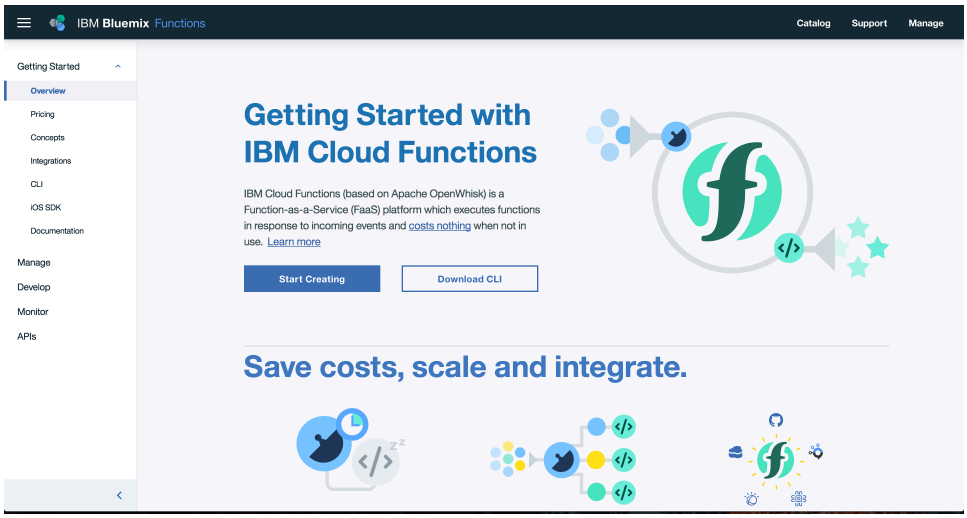


A digital copy of this lab and code snippets can be found at:
`http://ibm.biz/cloud-function-intro`

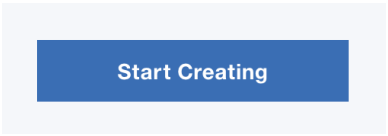
Create a Cloud Function

In this lab, we'll create a Cloud Function in IBM Bluemix. With a Cloud Function, you can write code that lives in the Cloud and can be executed on-demand and scale quickly. IBM Bluemix supports Python, Node.js, Java, Swift, PHP, and Docker environments. In this section, we'll create a basic Cloud Function.

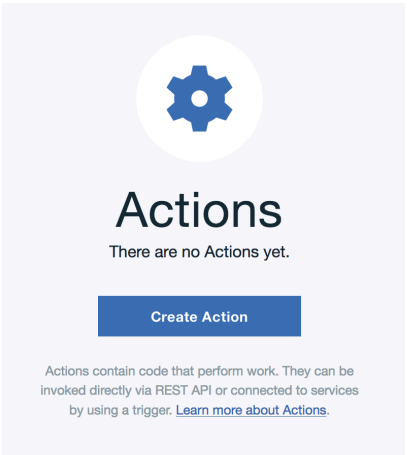
1. Sign up for an IBM Bluemix account at https://bluemix.net
2. Click on the menu in the top left corner of the IBM Bluemix console.
3. Select **Functions**.
4. This is the landing page where you can find the command line tool to deploy Cloud Functions via the command line, manage and monitor actions and triggers, and expose Cloud Functions as an API with this tooling. In this lab, we'll use the web-based tooling to create a Cloud Function.



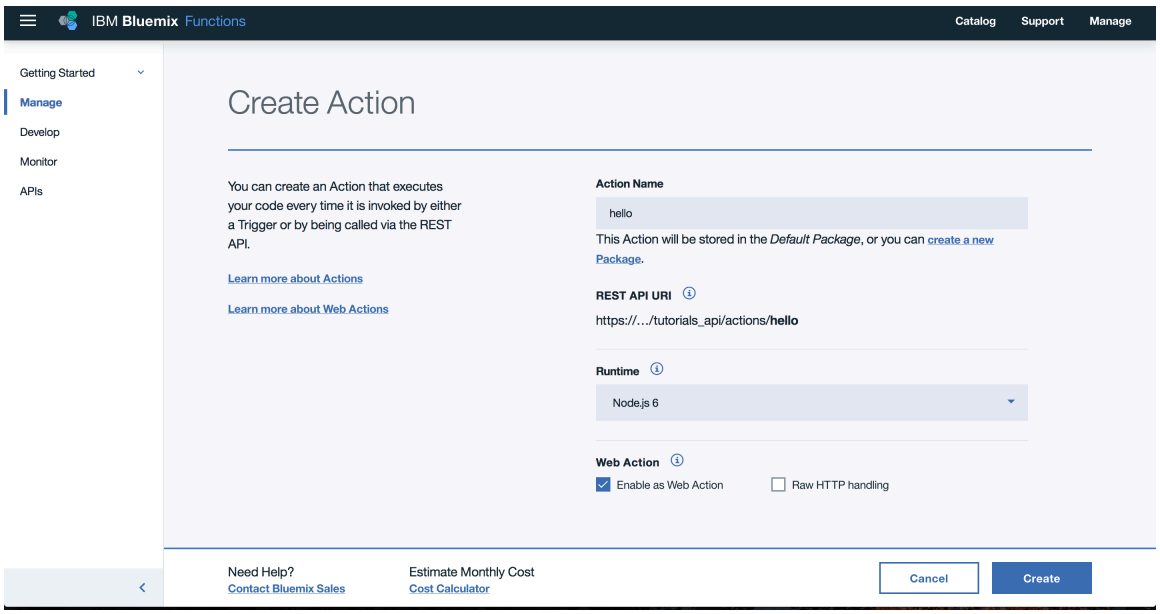
5. Click on **Start Creating**.



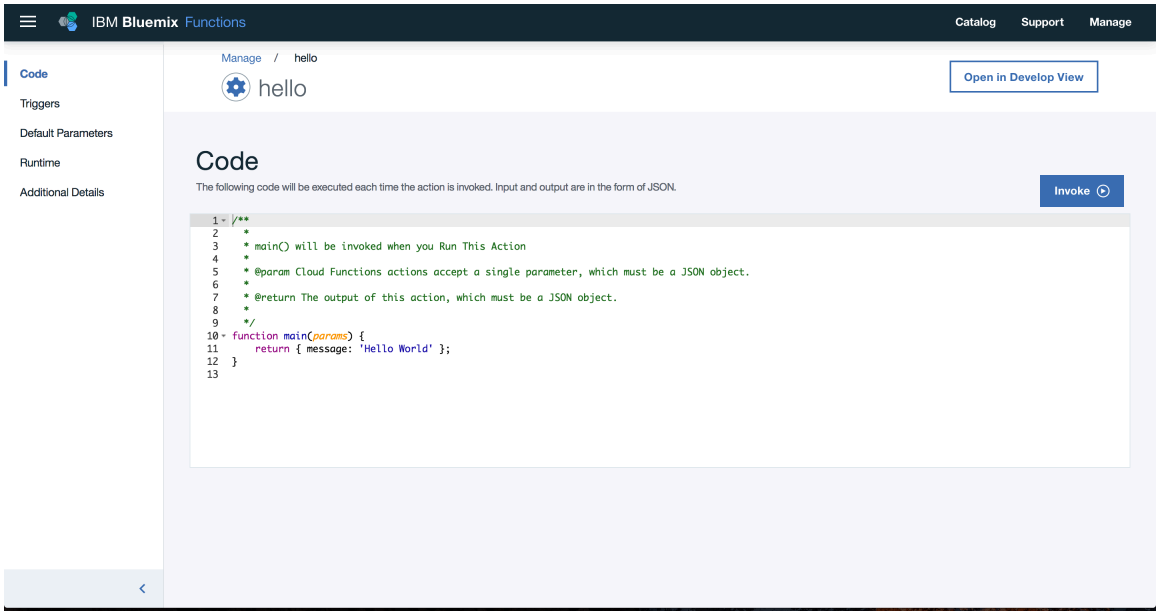
6. Since no Cloud Functions exist, click on **Create Action** to create a new action.



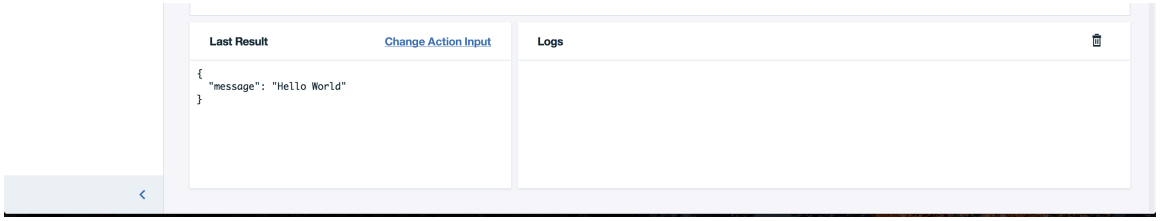
7. Enter an Action Name, `hello`, with a `Runtime Node.js 6`. Tick the checkbox next to **Enable as Web Action**. Click **Create** when finished.



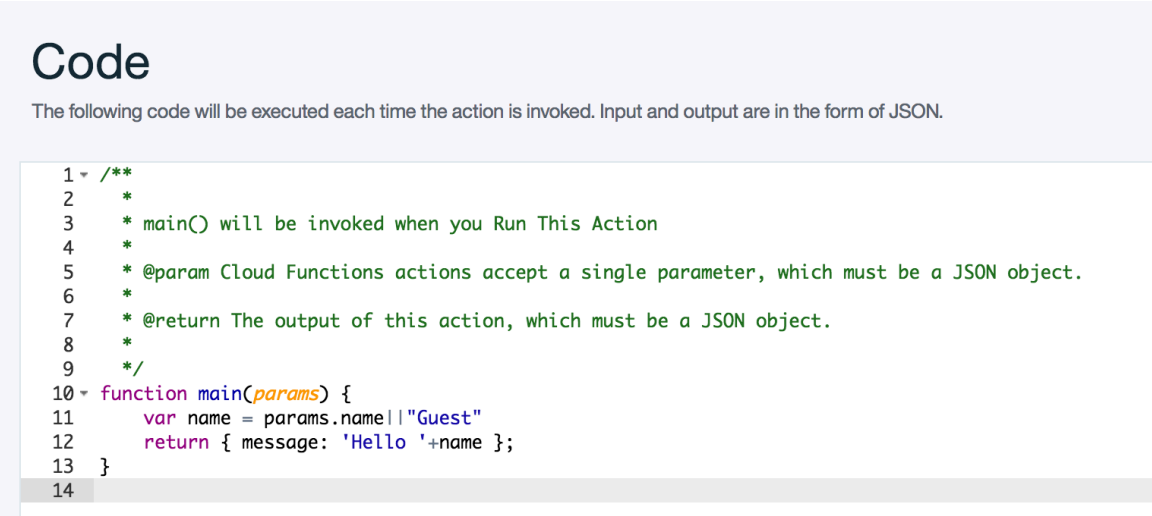
8. A Node.js template for a basic action is prepopulated into the textbox. This action returns a JSON object with a property named message and a value Hello World. Let's run this action. Click **Invoke**.



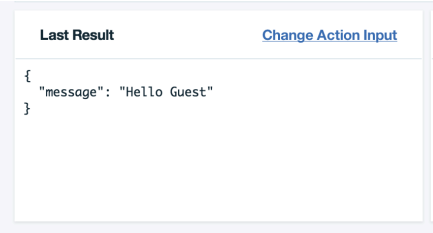
The code is executed and the last result is displayed below the code textbox.



9. Next, let's make this function more dynamic. If a **name** parameter is passed into the function, we'll greet the user by name. Change the code to use the value of the parameter labeled `name`.

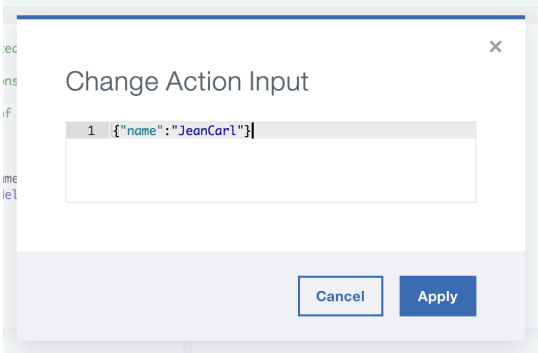


10. Click on the **Invoke** button on the right side of the page to run the code. The result includes the default name, `Guest`.

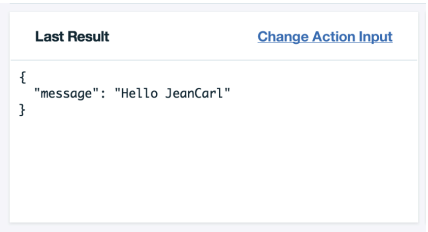


11. Make the name dynamic by changing the Action Input values. Click on the **Change Action Input** link in the box labeled **Last Result**.

12. Add a JSON property labeled `name` with a value of a name, such as your first name. Click **Apply** when finished.



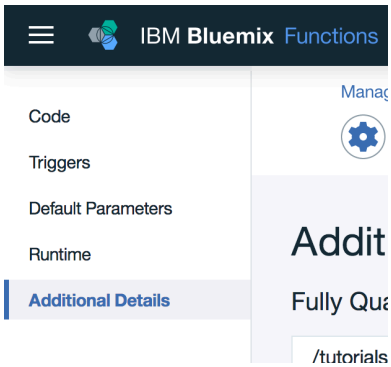
13. When the action is invoked with this name parameter, it is injected into the message value and the following result is returned.



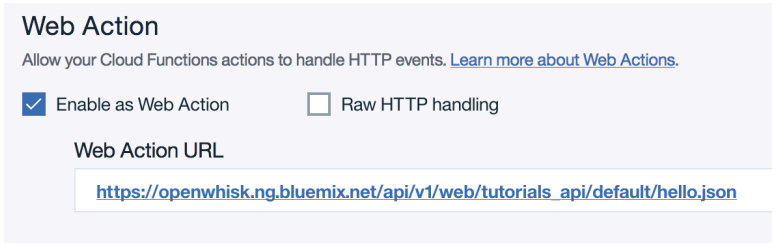
Create a Web Action

Cloud Functions can be made accessible as web actions, or in other words, given a URL that can be accessed by a browser or other application via an HTTP call. In this section, we'll access the Cloud Function we created in the previous section as a web action via a web browser.

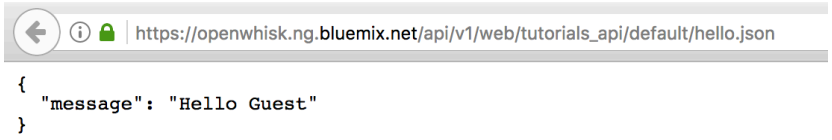
1. Select **Additional Details** in the menu in the left sidebar.



2. By ticking the checkbox next to **Enable Web Action** when the Cloud Function was created in the previous section, this Cloud function was given a URL. Copy this URL into the address bar of a new web browser tab.

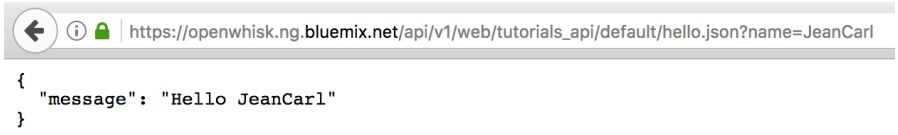


The JSON result is returned to the web browser.



3. Add a query **name** parameter to the end of the URL with a custom name as shown below. Query string parameters are passed into the Cloud Functions as an argument to the main function.

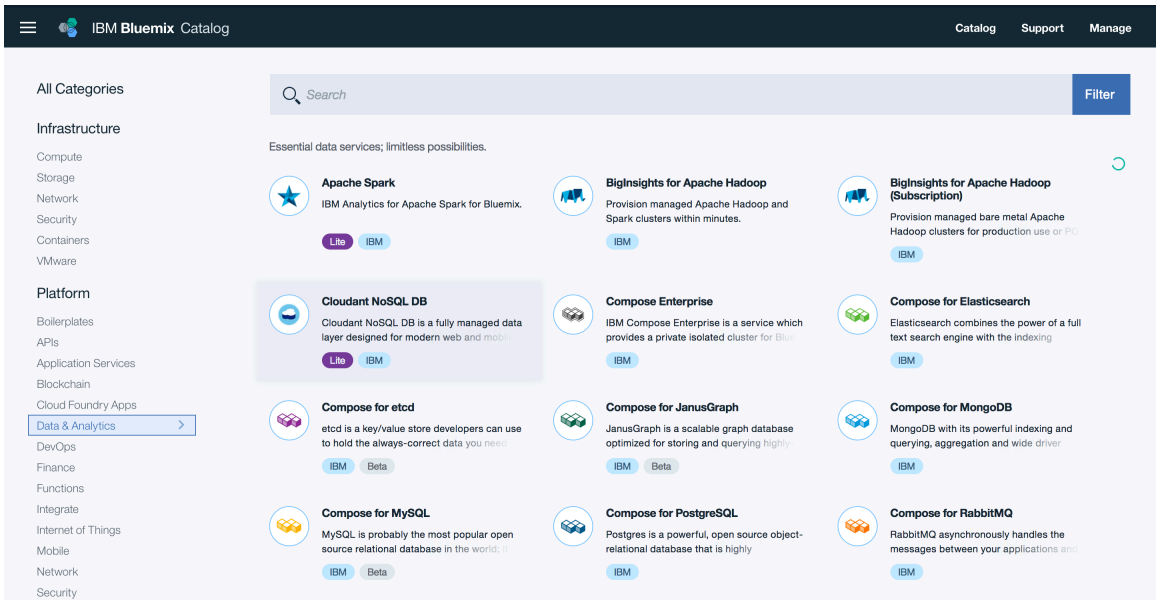
```
…/hello.json?name=JeanCarl
```



Build a Check-In Application

Cloud Functions are a great way to have a web service available on demand. It doesn't use any resources and you pay only for the time and resources the Cloud Function uses when you run the web action. In this section, we'll make a simple check-in application that records the name of a person into a database.

1. First, create a **Cloudant NoSQL DB** (database) in the **Data & Analytics** section of the IBM Bluemix catalog.



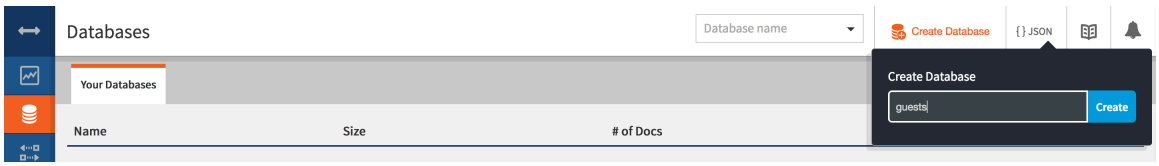
2. Click on **Service Credentials** in the sidebar. If the list of credentials are empty, click on **New credential**.

3. Click on **View Credentials** to display the service credentials. Copy the `url` property into the code in Step #7.

```
  "host": "8a1c48c1-3e55-45c5-9aad-b0bfd398a7cb-bluemix.cloudant.com",
  "port": 443,
  "url": "https://8a1c48c1-3e55-45c5-9aad-b0bfd398a7cb-bluemix:d2022e4e7f75d8badf5266e32a972fe52967
60dab83af321eb6a0e66c4141a63@8a1c48c1-3e55-45c5-9aad-b0bfd398a7cb-bluemix.cloudant.com"
}
```

4. Click on the green **Launch** button to launch the Cloudant dashboard.

5. Click on **Create Database** in the top right corner of the dashboard.
6. Give the database a name of `guests`.



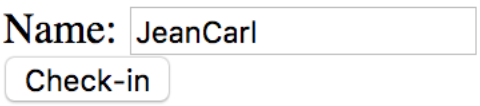
7. Keep the Cloudant dashboard open. Open another tab and return to the Cloud Functions section in the IBM Bluemix dashboard. Create a new Cloud Function, this time named `checkin`. To create a new Cloud Function, follow the steps on page two of this lab, and stop before you modify the code. Use the following function for this section.

```
function main(params) {
  return new Promise(function(reject, resolve) {

    var name = params.name||"Guest";
    var payload = {
      name: name
    }

    var Cloudant = require("cloudant");
    var cloudant = new Cloudant(____);
    var db = cloudant.db.use("guests");

    db.insert(payload, function(err, body, header) {
      if(err) {
        reject(err);
      } else {
        payload.message = "Hello "+payload.name;
        resolve(payload);
      }
    });
  });
}
```

8. This code returns a Promise function that inserts the name of the guest into a Cloudant NoSQL database, and eventually returns the response that is displayed on the webpage. Click on **Additional Details** in the sidebar to get the URL to the web action. Insert this URL in the placeholder (the following code uses the result from Step #3):

```
var cloudant = new Cloudant("https://8a1c48c1-3e55-45c5-9aad-b0bfd398a7cb-
bluemix:d2022e4e7f75d8badf5266e32a972fe5296760dab83af321eb6a0e66c4141a63@8a1c48c1-3e55-45c5-9aad-b0bfd398a7cb-
bluemix.cloudant.com");
```

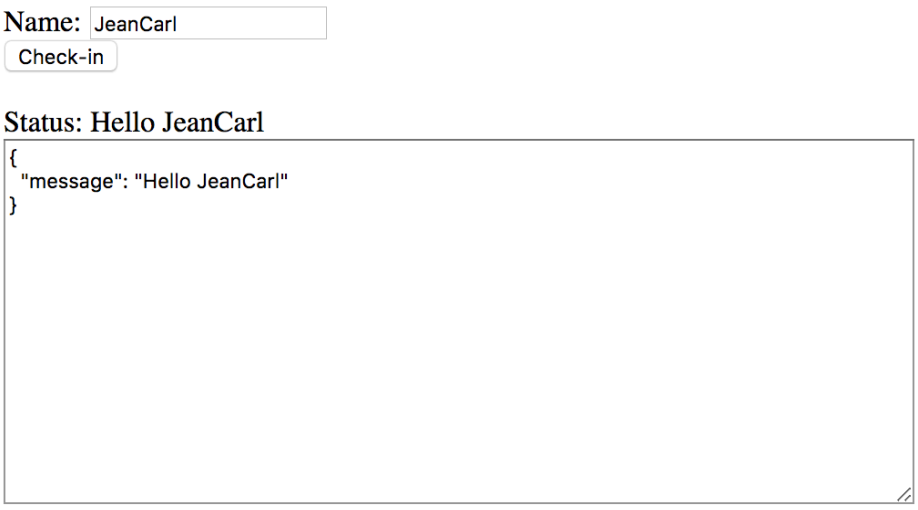
9. Clone the Git repo at `ibm.biz/cloud-function-intro-app`. This project contains a basic webpage with a form that collects a name. When the user submits the form, an AJAX call is made to the Cloud Function which adds the name to a Cloudant NoSQL database.



10. Edit the index.html file and replace the URL of the AJAX call to point to your Cloud Function web action.

```
$.ajax({
  type: "POST",
  url: "https://openwhisk.ng.bluemix.net/api/v1/web/___-___/default/checkin.json",
  data: {
    name: $("#name").val()
  },
```

11. Open the webpage in a bowser. Enter your name in the form and submit the form. The name value is posted to the web action. The JSON response is displayed in the textbox.



12. Return to the Cloudant dashboard and refresh the page. A new entry has been added to the database.