



Red Hat Mobile Application Platform 4.7

Getting Started

For Red Hat Mobile Application Platform 4.7

Red Hat Mobile Application Platform 4.7 Getting Started

For Red Hat Mobile Application Platform 4.7

Legal Notice

Copyright © 2018 Red Hat, Inc.

The text of and illustrations in this document are licensed by Red Hat under a Creative Commons Attribution–Share Alike 3.0 Unported license ("CC-BY-SA"). An explanation of CC-BY-SA is available at

<http://creativecommons.org/licenses/by-sa/3.0/>

. In accordance with CC-BY-SA, if you distribute this document or an adaptation of it, you must provide the URL for the original version.

Red Hat, as the licensor of this document, waives the right to enforce, and agrees not to assert, Section 4d of CC-BY-SA to the fullest extent permitted by applicable law.

Red Hat, Red Hat Enterprise Linux, the Shadowman logo, JBoss, OpenShift, Fedora, the Infinity logo, and RHCE are trademarks of Red Hat, Inc., registered in the United States and other countries.

Linux ® is the registered trademark of Linus Torvalds in the United States and other countries.

Java ® is a registered trademark of Oracle and/or its affiliates.

XFS ® is a trademark of Silicon Graphics International Corp. or its subsidiaries in the United States and/or other countries.

MySQL ® is a registered trademark of MySQL AB in the United States, the European Union and other countries.

Node.js ® is an official trademark of Joyent. Red Hat Software Collections is not formally related to or endorsed by the official Joyent Node.js open source or commercial project.

The OpenStack ® Word Mark and OpenStack logo are either registered trademarks/service marks or trademarks/service marks of the OpenStack Foundation, in the United States and other countries and are used with the OpenStack Foundation's permission. We are not affiliated with, endorsed or sponsored by the OpenStack Foundation, or the OpenStack community.

All other trademarks are the property of their respective owners.

Abstract

This tutorial guides you through the core features of Red Hat Mobile Application Platform 4.7.

Table of Contents

PREFACE	3
CHAPTER 1. CREATE A PROJECT	4
1.1. EXPLORE THE PROJECT	5
CHAPTER 2. DEPLOY THE CLOUD APP	6
2.1. CLOUD APP DEPLOYMENTS	6
2.2. BUILD QUEUE POLICY	7
CHAPTER 3. PREVIEW THE CLIENT APP	8
CHAPTER 4. CUSTOMIZE THE CLOUD APP	10
CHAPTER 5. MODIFY THE CLIENT APP	12
CHAPTER 6. CONCLUSION	14

PREFACE

Overview

To get started with Red Hat Mobile Application Platform (RHMAP) quickly, the first step is to understand the project life cycle. This involves creating a project, creating a client and a Cloud App from templates, deploying the Cloud App to the MBaaS, building the Client App, and deploying it to a mobile device.

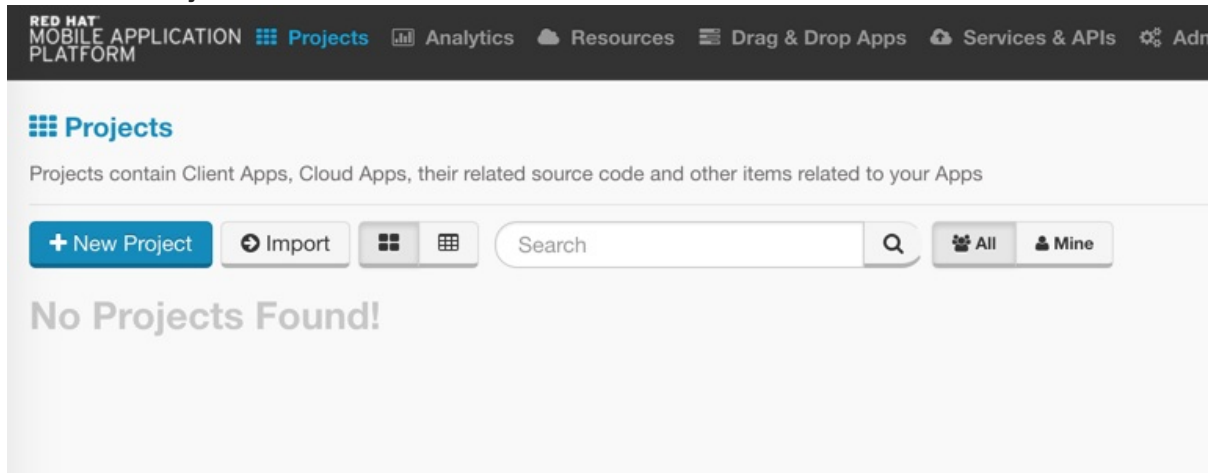
This guide uses the Studio — the web interface of RHMAP — for all operations. However, you can also use the [FHC command line tool](#) to access most functions of RHMAP.

CHAPTER 1. CREATE A PROJECT

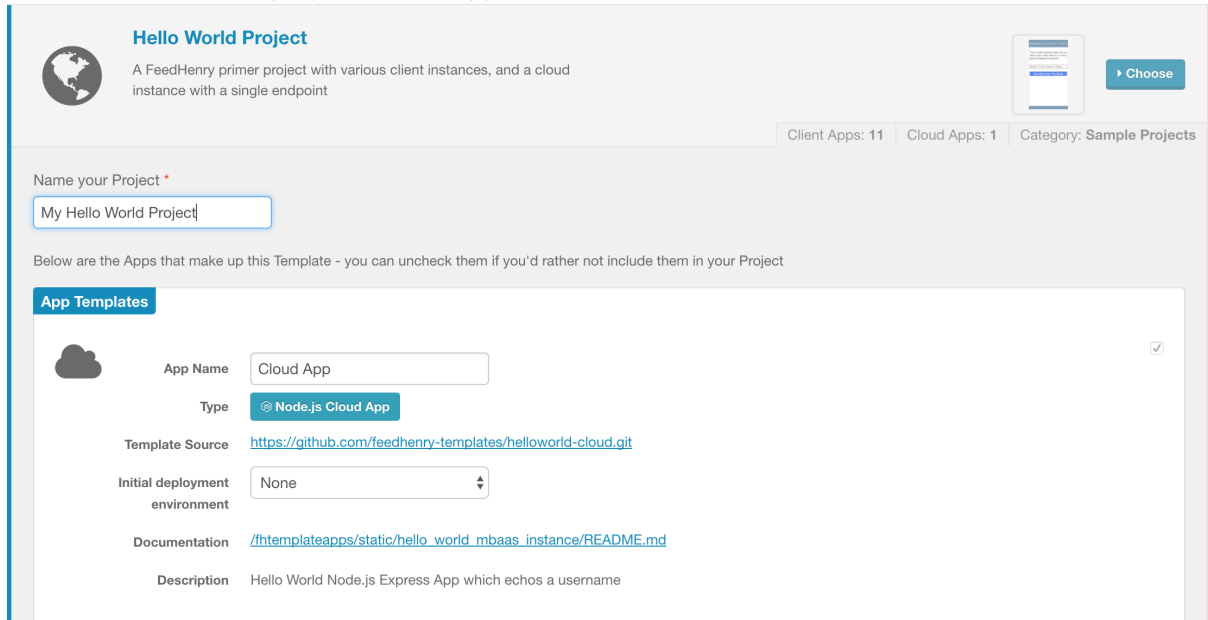
Projects help you group all code bases related to a single mobile application in one place. Projects contain Client Apps, Cloud Apps, MBaaS services, and any data and configurations associated with them.

Create a new project from an existing template.

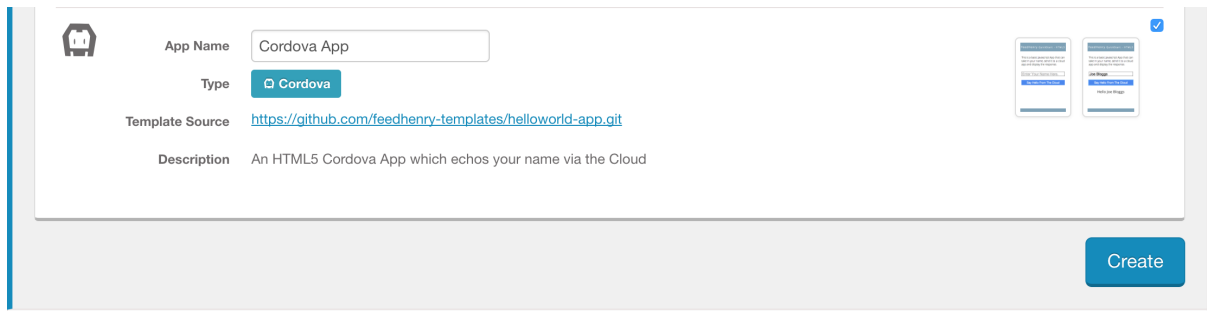
1. Log in to the Studio and navigate to the **Projects** area.
2. Click **New Project**.



3. Select the **Hello World Project** template by clicking **Choose** on the right side next to it.
4. Enter a name for the project in the "App Name" field.



5. Scroll down to find the Cordova App option. Enter a name for the project in the "App Name" field and also ensure the checkbox is selected.



The screenshot shows a project creation form with the following fields and values:

- App Name:** Cordova App
- Type:** Cordova
- Template Source:** <https://github.com/feedhenry-templates/helloworld-app.git>
- Description:** An HTML5 Cordova App which echos your name via the Cloud

On the right side of the form, there are two preview images of a mobile app interface. A blue 'Create' button is located at the bottom right of the form.

6. Click **Create**.

7. The progress bar turns green when the project is successfully created. Click **Finish**.

1.1. EXPLORE THE PROJECT

After creating a project, you can see the **Apps, Cloud Apps & Services** section. This displays the Client and Cloud Apps, and also the MBaaS services associated with the project.

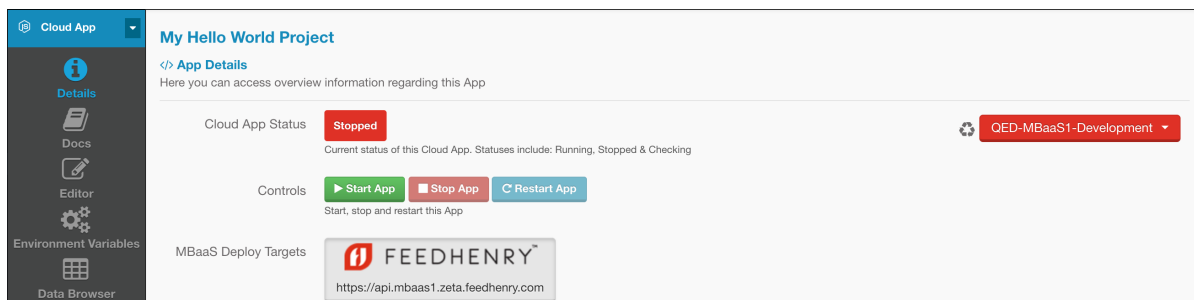
- **Client Apps:** applications deployed on mobile devices used by the end users.
- **Cloud Apps:** applications deployed in the MBaaS that handle requests from Client Apps and communicate with other internal or external systems.
- **MBaaS Services:** reusable services used by Cloud Apps and shared across multiple projects.

The newly created *My Hello World Project* contains one Client App (Cordova technology) and one Cloud App (Node.js technology) with a single HTTP endpoint. You can add more Client and Cloud Apps, and also MBaaS services to the project by clicking the + symbol in each corresponding box.

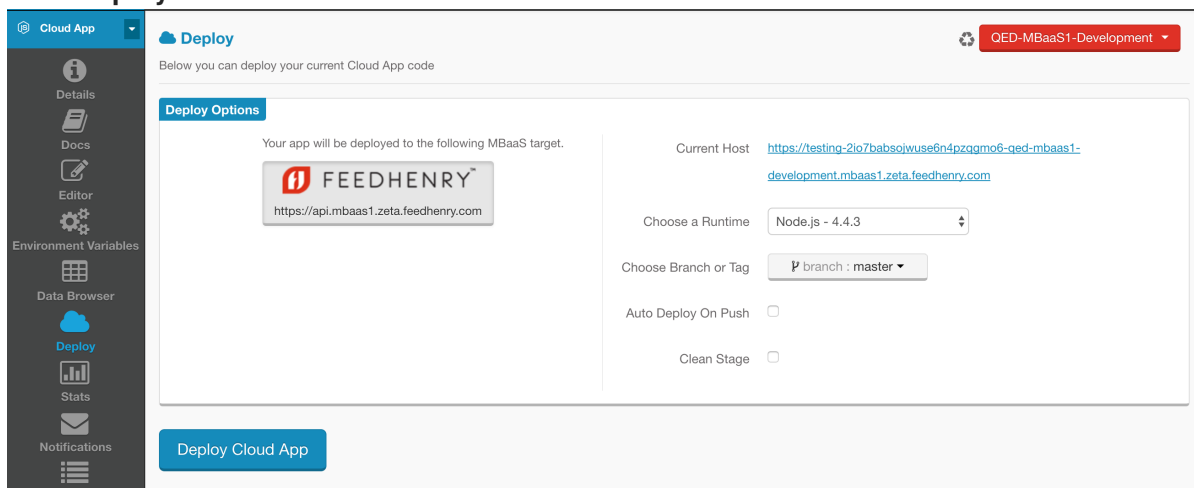
CHAPTER 2. DEPLOY THE CLOUD APP

Depending on the setup of your cluster, the Cloud App may need to be deployed manually after creating the project.

1. In the **Apps, Cloud Apps & Services** section of the screen, click on the *Cloud App* - this will display the *App Details* screen.
2. In the *App Details* section of the screen, see the value of the **Cloud App Status** field. If the status is *Stopped*, you must deploy the Cloud App. If it is *Running*, skip to section 3. [Preview the Client App](#).



3. Click **Deploy** on the sidebar on the left.



4. Click **Deploy Cloud App**



NOTE

By default, the Cloud App deployment reuses existing container images built from the same commit hash and for the same Node.JS runtime. To force RHMAP to create a new container image, select the **Clean Stage** option.

Once the deployment is finished, the progress bar turns green and the Cloud App is deployed.

5. Click **Details** on the sidebar on the left.
6. Verify that the **Cloud App Status** is now *Running*.

2.1. CLOUD APP DEPLOYMENTS

Deploying a Cloud App into an Environment can require a Cloud App source-to-image (S2I) build that creates a new container image. RHMAP reuses the container image created in the previous

Environment when reuse is appropriate, that is, when both the commit hashes and Node.JS runtime versions are identical. This provides more consistency and better performance for Cloud App deployments.

2.2. BUILD QUEUE POLICY

When you deploy a Cloud App, RHMAP creates an asynchronous request to the OpenShift API and the Deploy Cloud App button is disabled until the build completes and the application is deployed. Only one running deployment and one queued deployment can exist at the same time, and if you deploy an additional Cloud App, it will replace the Cloud App currently in the queue. To deploy a Cloud App under these conditions, either wait for the existing deployment to complete, or cancel the existing deployment. This policy is described further in [SerialLatestOnly Run Policy](#).



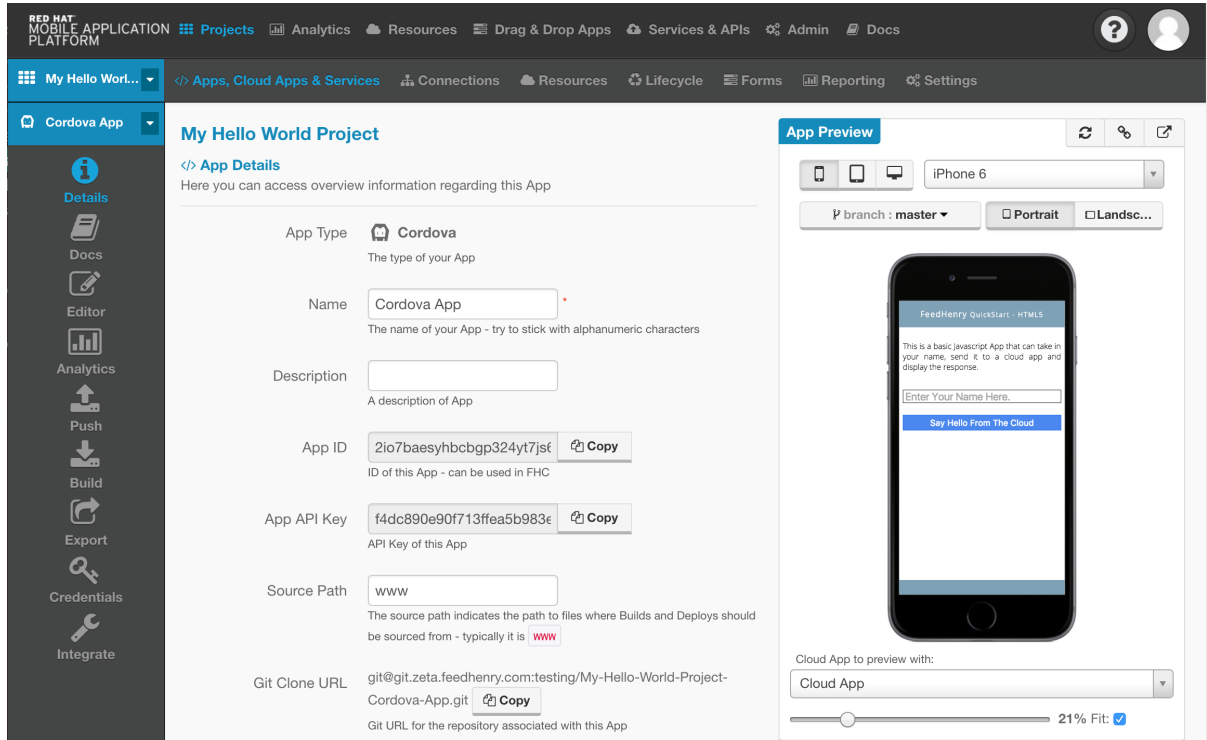
NOTE

Various build policies are described in the [Build Run Policy](#) including the SerialLatestOnly run policy. However, you cannot change this policy for RHMAP, that is, the SerialLatestOnly run policy applies for all RHMAP deployments.

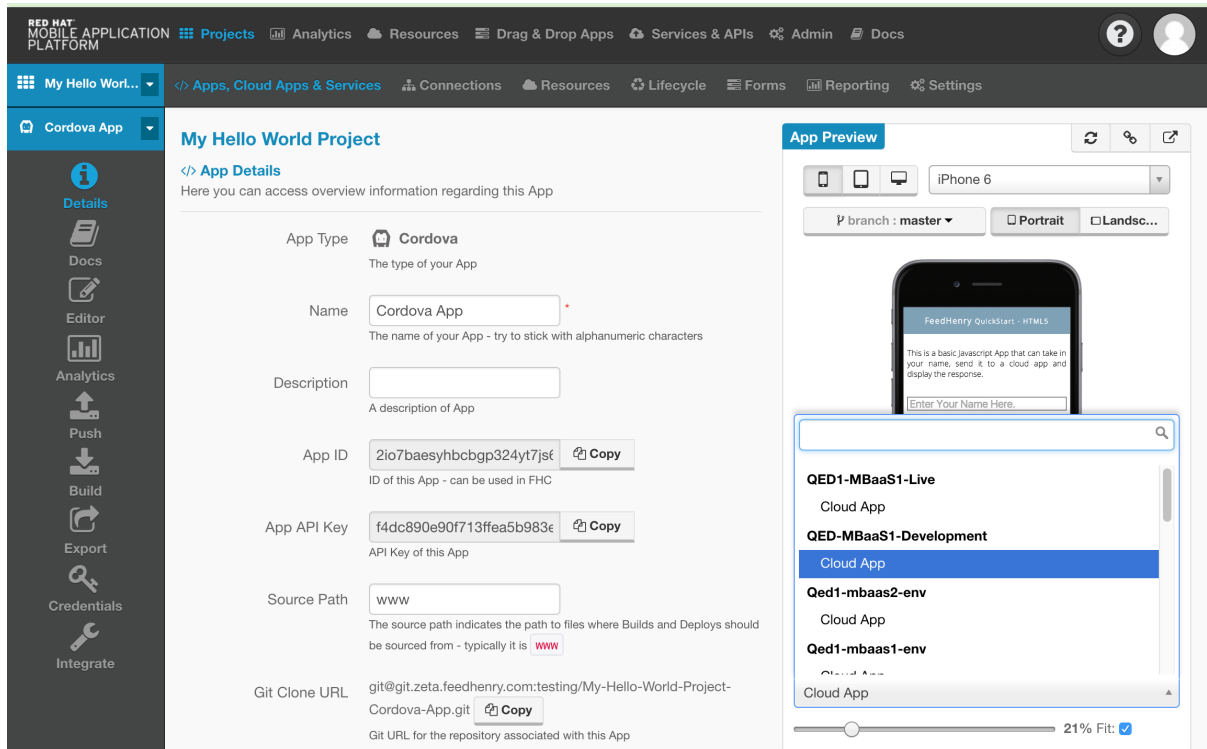
CHAPTER 3. PREVIEW THE CLIENT APP

With the project created and the Cloud App deployed, you can now test the Client App.

1. In the **Apps, Cloud Apps & Services** screen, click on the *Cordova App* in the **Apps** box. This opens the **App Details** page. On the right, you can interact with a running preview of your app. On the left is metadata, such as your app's name, ID, and git repository URL.



2. Ensure that the environment where the Cloud App is deployed to is selected in the "App Preview" section.



3. In the preview, enter your name in the provided box.

4. Click **Say Hello From The Cloud**.

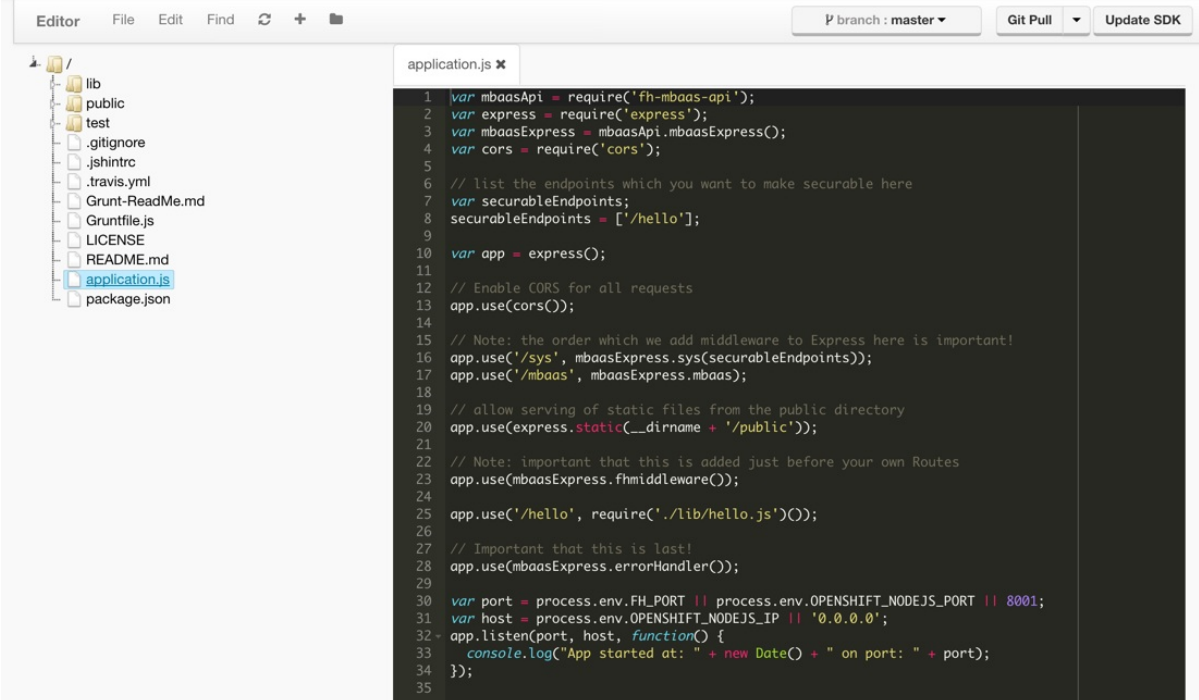
The Client App makes a call to the Cloud App in this project and shows a response in the area beneath the button.

CHAPTER 4. CUSTOMIZE THE CLOUD APP

To better understand how the Cloud App works, make a minor modification to the code. Add a **timestamp** field with the value of the current UNIX time stamp to the server response. In the next section, you will modify the Client App to display the time stamp.

1. Navigate to the **Projects** area using the navigation bar at the top.
2. Open the *My Hello World Project* project.
3. Open the *Cloud App*.
4. Click **Editor** on the sidebar on the left.

This area lets you edit the source code of any file in the Git repository of the Cloud App. The Cloud App in this project is a Node.js web application framework called *Express*.



```

1 var mbaasApi = require('fh-mbaas-api');
2 var express = require('express');
3 var mbaasExpress = mbaasApi.mbaasExpress();
4 var cors = require('cors');
5
6 // list the endpoints which you want to make securable here
7 var securableEndpoints;
8 securableEndpoints = ['/hello'];
9
10 var app = express();
11
12 // Enable CORS for all requests
13 app.use(cors());
14
15 // Note: the order which we add middleware to Express here is important!
16 app.use('/sys', mbaasExpress.sys(securableEndpoints));
17 app.use('/mbaas', mbaasExpress.mbaas);
18
19 // allow serving of static files from the public directory
20 app.use(express.static(__dirname + '/public'));
21
22 // Note: important that this is added just before your own Routes
23 app.use(mbaasExpress.fhmiddleware());
24
25 app.use('/hello', require('./lib/hello.js')());
26
27 // Important that this is last!
28 app.use(mbaasExpress.errorHandler());
29
30 var port = process.env.FH_PORT || process.env.OPENSIFT_NODEJS_PORT || 8001;
31 var host = process.env.OPENSIFT_NODEJS_IP || '0.0.0.0';
32 app.listen(port, host, function() {
33   console.log("App started at: " + new Date() + " on port: " + port);
34 });
35

```



NOTE

The Studio editor provides a quick and easy way to edit files, however, Red Hat recommends cloning the code and using a local editor or IDE to develop applications.

5. Open the **application.js** file.

application.js handles all requests to the Cloud App. The Client App sends requests to the **/hello** endpoint and the **application.js** file routes those requests to another file called **hello.js**.

```
app.use('/hello', require('./lib/hello.js')());
```

To learn more about routing in Express, take a look at the [Express Router documentation](#).

Change the **lib/hello.js** file to return a timestamp in the response.

1. Open **lib/hello.js**.

2. Add a **timestamp** property to the POST response object, with the value of the current UNIX time stamp.

Find this line in the POST handler:

```
res.json({msg: 'Hello ' + world});
```

Change that line to the following:

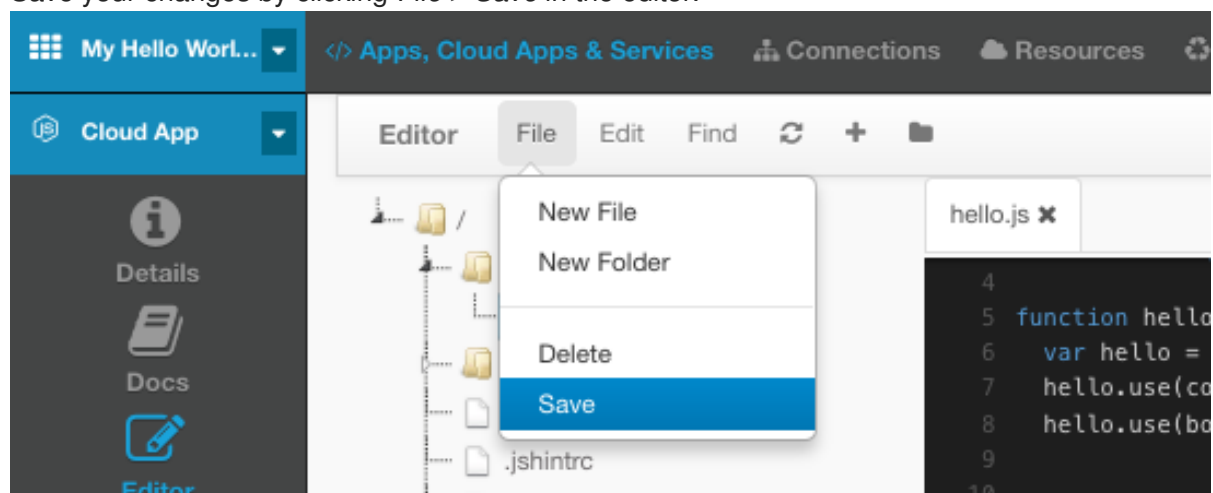
```
res.json({msg: 'Hello ' + world, timestamp: new Date().getTime() });
```

The POST handler now looks like this:

```
hello.post('/', function(req, res) {
  console.log(new Date(), 'In hello route POST / req.body=',
req.body);
  var world = req.body && req.body.hello ? req.body.hello : 'World';

  // see http://expressjs.com/4x/api.html#res.json
  res.json({msg: 'Hello ' + world, timestamp: new Date().getTime()
});
});
```

3. Save your changes by clicking *File > Save* in the editor.



The changes are saved to the Git repository of the Cloud App. To propagate the changes to the running instance, you must re-deploy the Cloud App.

4. Click **Deploy** on the sidebar on the left.
5. Click **Deploy Cloud App**.

CHAPTER 5. MODIFY THE CLIENT APP

Change the Client App to also show the **timestamp** property from the received server response.

First, create a placeholder for the response.

1. Navigate to the **Apps, Cloud Apps & Services** page.
2. Open the **Cordova App** Client App.
3. Open the **Editor**.



NOTE

The Studio editor provides a quick and easy way to edit files, however, Red Hat recommends cloning the code and using a local editor or IDE to develop applications.

4. Open the **www/index.html** file.
5. Add a new **<div>** that will show the received **timestamp**.
This element acts as a placeholder for the received value.

Find this line:

```
<div id="cloudResponse" class="cloudResponse"></div>
```

Replace it with the following:

```
<div id="cloudResponse" class="cloudResponse"></div>
<div id="timestamp" class="cloudResponse"></div>
```

6. Save the changes using *File > Save*, or using the *Ctrl + S* keyboard shortcut (Windows) or *cmd + S* keyboard shortcut (Mac).

Modify the handler of the **Say Hello From The Cloud** button to use the received **timestamp** value to populate the placeholder.

1. Open the **www/js/hello.js** file in the editor.
This file contains a click handler for the **Say Hello From The Cloud** button, which uses the **\$fh.cloud** API to call the **/hello** endpoint of the Cloud App and populates the placeholder **<div id="timestamp">** element.

2. Set the placeholder to the received **timestamp** value.

Find the following code:

```
document.getElementById( 'cloudResponse' ).innerHTML = "<p>" + res.msg
+ "</p>";
```

Replace it with the following:

```
document.getElementById( 'cloudResponse' ).innerHTML = "<p>" + res.msg
+ "</p>";
```



```
document.getElementById('timestamp').innerHTML = "<p>" +  
res.timestamp + "</p>";
```

3. Save your changes.
The preview will update automatically.
4. Click **Say Hello From The Cloud** in the preview.
The area below the button now also contains a long string of numbers, which represent the time stamp. If it does not work, try refreshing the page.



CHAPTER 6. CONCLUSION

You should now have an understanding of basic concepts, such as:

- Projects, Client Apps, and Cloud Apps and how these are related.
- Building an app binary for Android and trying it on a mobile device.
- Making changes to Client Apps and Cloud Apps.

You can find detailed guides and explanations in [the RHMAP documentation](#).

For a more detailed walkthrough, which includes some insight into local development, you can watch the [Red Hat Mobile Application Platform Overview Demo](#).