



# Cloud Developers Certification Training

Debugging Applications in Bluemix

## Lab Exercise

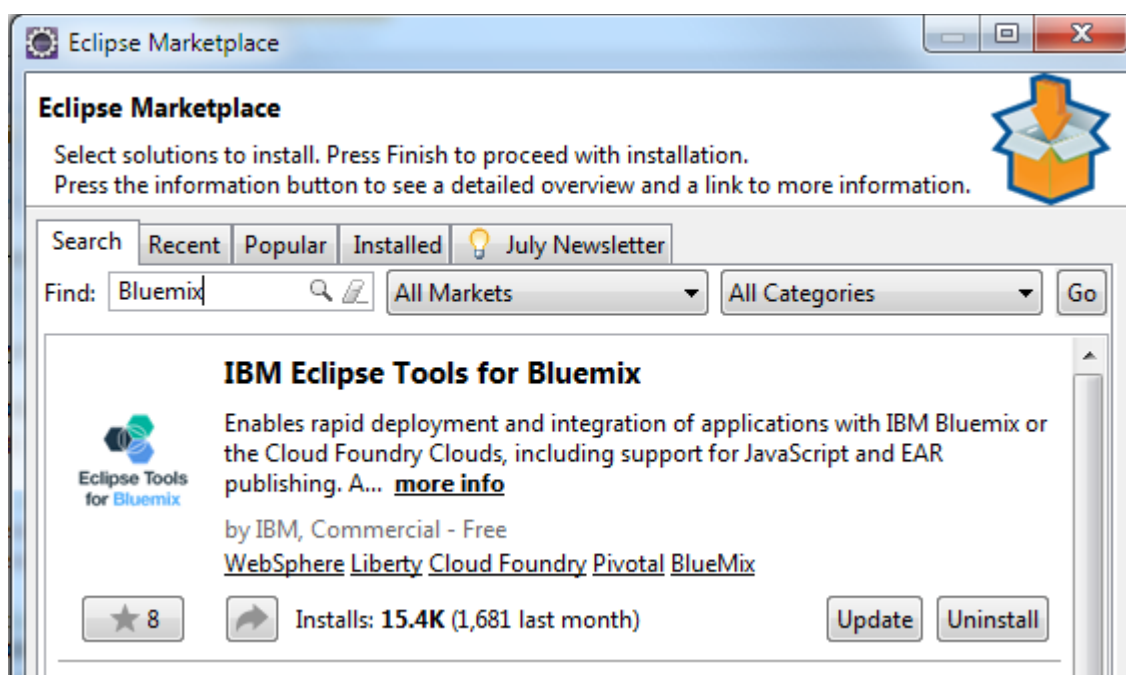
<b>Version:</b>	1
<b>Last modification date:</b>	3-Sep-15
<b>Owner:</b>	IBM Ecosystem Development



**LAB OBJECTIVE:** Debug a cloud ready application in Bluemix. Our objective is to look at the errors encountered while running an application and resolve them. We will debug a Java application and Node.js application.

**PRE-REQUISITES :**

1. Install Eclipse Luna (4.4) or Kepler (4.3). We expect the audience to be familiar with using Eclipse.
2. Install Bluemix Plugin for Eclipse using Eclipse MarketPlace as shown below :



3. A Java project already deployed to Bluemix.

**Debugging Java Application using IBM Java Liberty Runtime**

**How:** Please import SampleWebApp.zip from <https://github.com/sandhya9/SampleWebApp>.

**Benefits:** Switch an application instance in Bluemix to Dev Mode to be able to:

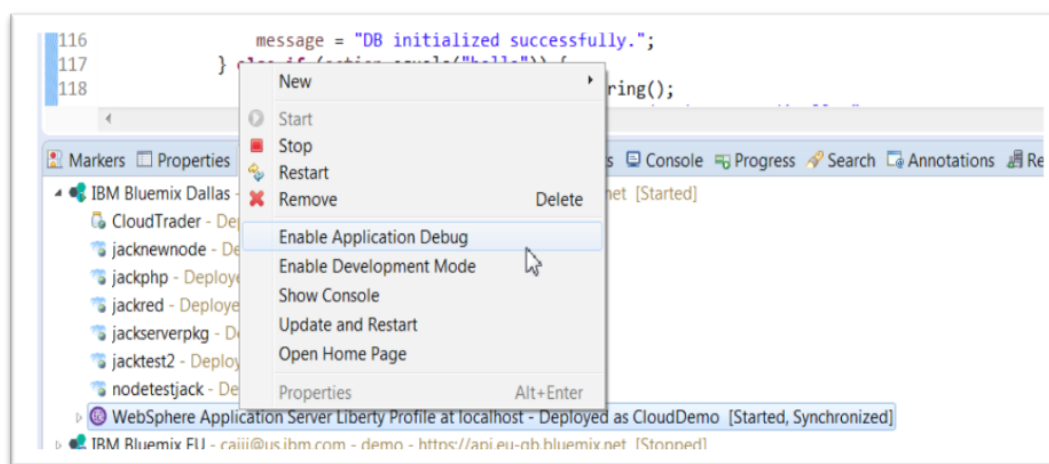
- Initiate remote debug sessions with that application instance
- Push incremental file updates to that application instance and see the changes without repushing the app;
- Run and access additional tools inside the app container such as a SSH web console, as supported by each Buildpack's Dev Mode.

**Steps:**

- IBM Bluemix Components need to be installed on Eclipse as described at

<https://www.ng.bluemix.net/docs/#manageapps/eclipsetools/eclipsetools.html>

- Next, under the Bluemix server, right-click the application that you want to debug.
- Note: This function cannot be enabled if the application deployment name has an underscore. Change the name before enabling remote debug.
- Click Enable Application Debug. The Progress View shows the status of Establishing debug session for <AppName>. The application will show that it is "Developing, Debugging <AppName>". At this point, the debugger is running and ready to use



- Back to Normal Mode: Switch an application instance in Bluemix from Dev Mode to Normal Mode, so that the application comes back to normal as if nothing happened during Dev Mode.
- You can disable the debug process and leave development mode enabled, if you so choose.
  - Right-click the application.
  - Click Disable Application Debug.
  - A dialog box asks if you want to disable development mode, also. Click Yes or No.

Note: If developer repushes/restages/restarts the application, all the application instances will also come back to Normal Mode. And all file changes made during Dev Mode will be lost.

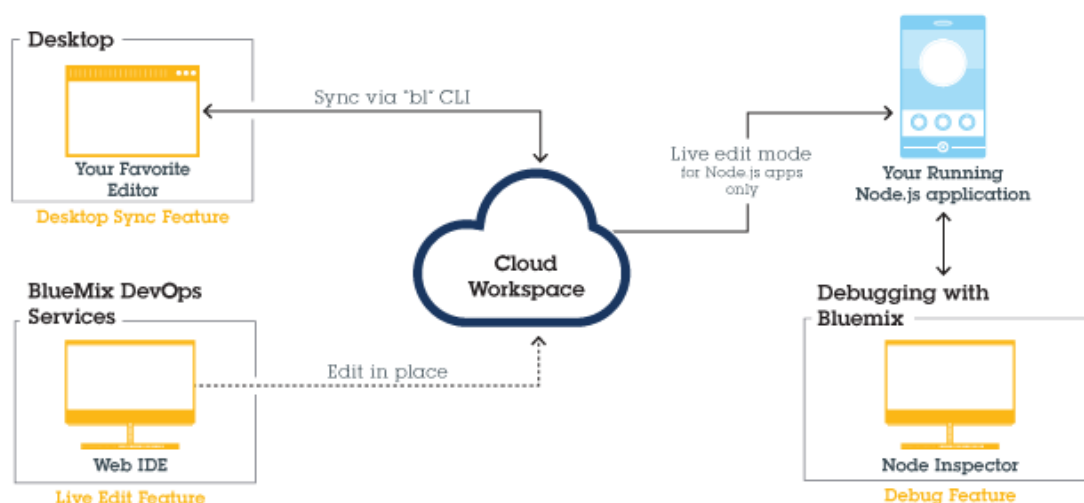
## Debugging Node.js using SDK for Node.js

### PRE-REQUISITES :

- [Sign up for DevOps Services](#). When you sign up, you'll create an IBM id, create an alias, and register with Bluemix.
- If you don't already have a Node.js project to work with, [create one](#).

- To develop locally from your desktop, download and install the Bluemix Live Sync command-line interface for [Windows](#) or for [Mac OS X](#). The Live Sync command-line interface is available only for Windows 7 and 8 and Mac OS X version 10.9 or later.

**Steps:** Use the Bluemix DevOps service's "Live Edit" feature via Favorite IDE like Eclipse or DevOps Service after turning on dev mode

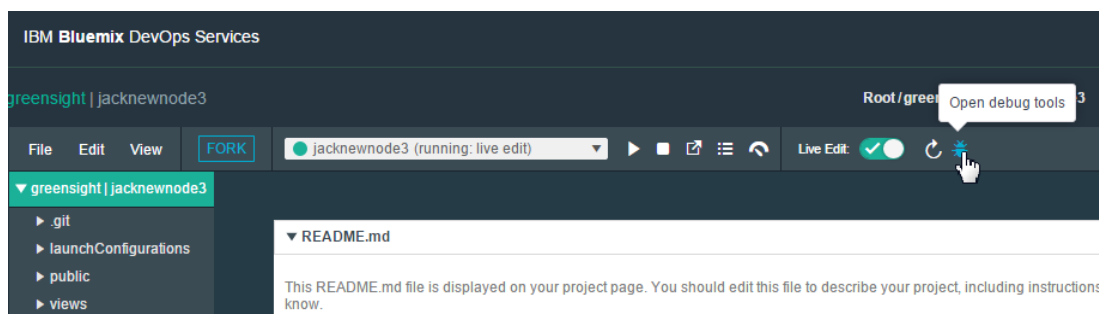


**Benefits:** Switch an application instance in Bluemix to Dev Mode in SDK for Node.js to :

- Set breakpoints in the app code to pause execution at a specific line.
- Edit breakpoint conditions to pause execution only when certain criteria are met.
- Inspect the state of local variables and fields.
- View debug output from console.log() calls immediately. This action is faster than monitoring cf logs.
- Use the built-in source code editor to make immediate, yet temporary, changes to the running app code.

## Enabling Dev Mode for Debugging in IBM Node JS Build pack

- Allow the buildpack to detect the app start command. The start command must be auto-detected by the buildpack, not set in the manifest.yml file.
- Ensure that the package.json file contains a start script that includes a start command for the app. If the App's manifest.yml file contains a command, set it to null.
- Set the environment variable. In the App's manifest.yml file, add this variable:  
Env:  
ENABLE\_BLUEMIX\_DEV\_MODE: "true"
- Increase the memory. In the App's manifest.yml file, add 128M or more to the value that is specified for the memory attribute.
- Install Bluemix Live Debug to use the debug tools.
- Push the app then browse to <https://<app-host>.mybluemix.net/bluemix-debug/manage> to access the Bluemix debug user interface. When you are prompted, enter your IBM ID and password to authenticate.
- One can also navigate to the debug user interface by clicking on the “Open Debug Tools” button on the DevOps page



- This is an example of the debug user interface

You are now managing the runtime (Node.js process) of your app. To manage your project, visit your [app dashboard](#).

## Jacknewnode3

Route: [jacknewnode3.mybluemix.net](#)

✓ Your app is running.

SUSPEND  
RESTART

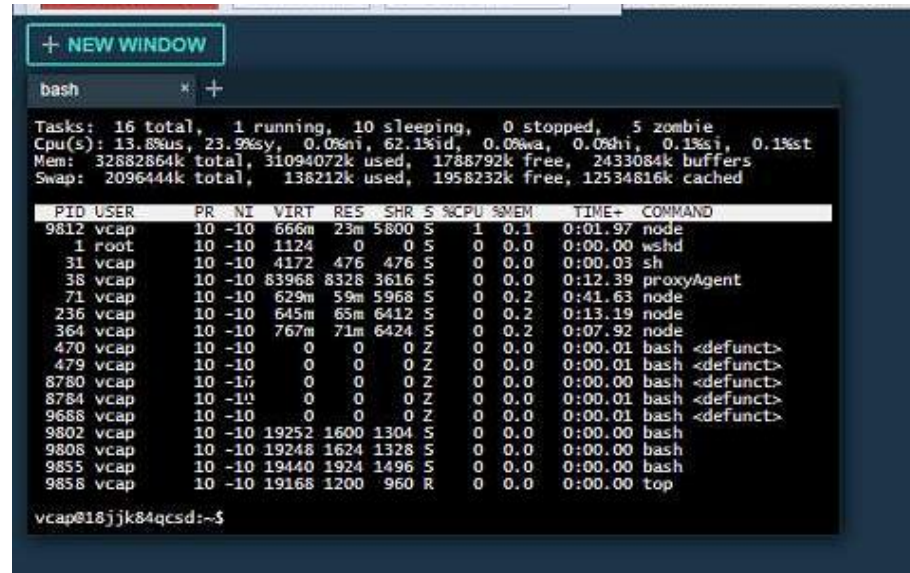
Open Shell →

Open Debugger →

From this DEBUG TOOLS page – we can

1. SUSPEND APP
2. RESTART
3. OPEN SHELL INTERFACE TO THE CONTAINER
4. OPEN DEBUG INSPECTOR (OPEN DEBUGGER)

BELOW IS AN EXAMPLE OF THE CONSOLE WINDOW



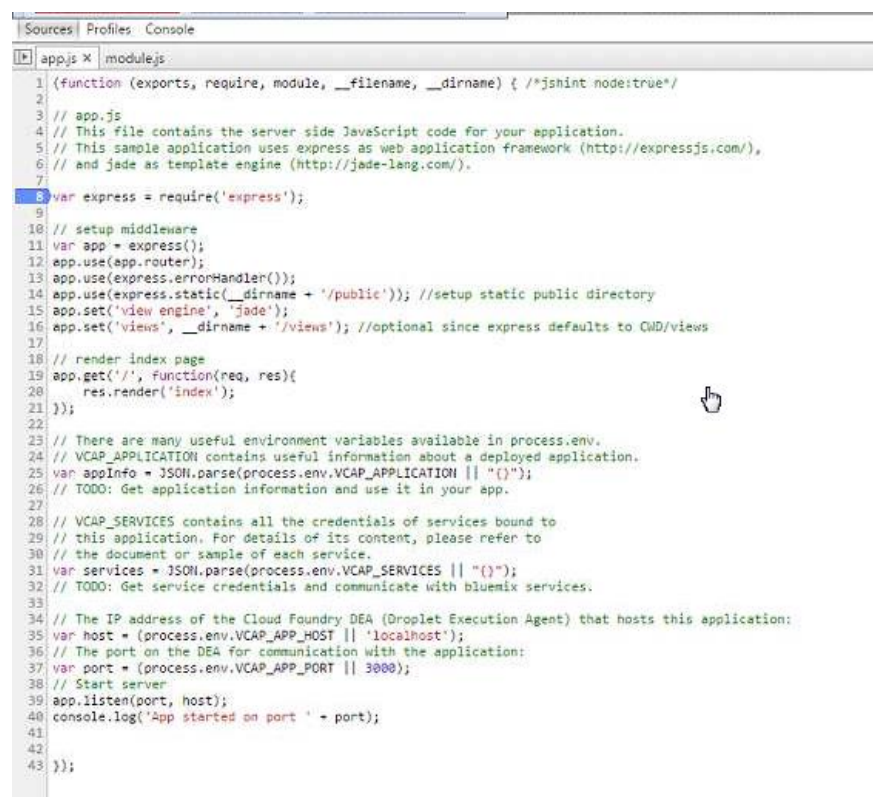
```

+ NEW WINDOW
bash
Tasks: 16 total, 1 running, 10 sleeping, 0 stopped, 5 zombie
Cpu(s): 13.8%us, 23.9%sy, 0.0%ni, 62.1%id, 0.0%wa, 0.0%hi, 0.1%si, 0.1%st
Mem: 32882864k total, 31094072k used, 1788792k free, 2433084k buffers
Swap: 2096444k total, 138212k used, 1958232k free, 12534816k cached

  PID USER      PR  NI  VIRT  RES  SHR  S %CPU  %MEM    TIME+  COMMAND
 9812 vcap     10   -10 666m 23m 5800 S   1  0.1   0:01.97 node
    1 root      10   -10 1124    0    0 S   0  0.0   0:00.00 wshd
   31 vcap     10   -10 4172   476 476 S   0  0.0   0:00.03 sh
   38 vcap     10   -10 83968 8328 3616 S   0  0.0   0:12.39 proxyAgent
   71 vcap     10   -10 629m 59m 5968 S   0  0.2   0:41.63 node
  236 vcap     10   -10 645m 65m 6412 S   0  0.2   0:13.19 node
  364 vcap     10   -10 767m 71m 6424 S   0  0.2   0:07.92 node
  470 vcap     10   -10    0    0    0 Z   0  0.0   0:00.01 bash <defunct>
  479 vcap     10   -10    0    0    0 Z   0  0.0   0:00.01 bash <defunct>
  8780 vcap     10   -10    0    0    0 Z   0  0.0   0:00.00 bash <defunct>
  8784 vcap     10   -10    0    0    0 Z   0  0.0   0:00.01 bash <defunct>
  9688 vcap     10   -10    0    0    0 Z   0  0.0   0:00.01 bash <defunct>
  9802 vcap     10   -10 19252 1600 1304 S   0  0.0   0:00.00 bash
  9808 vcap     10   -10 19248 1624 1328 S   0  0.0   0:00.00 bash
  9855 vcap     10   -10 19440 1924 1496 S   0  0.0   0:00.00 bash
  9858 vcap     10   -10 19168 1200   960 R   0  0.0   0:00.00 top

vcap@18jkk84qcsd:~$
  
```

BELOW IS AN EXAMPLE OF THE DEBUG INSPECTOR



```

Sources Profiles Console
app.js module.js
1 {function (exports, require, module, __filename, __dirname) { /*jshint node:true*/
2
3 // app.js
4 // This file contains the server side JavaScript code for your application.
5 // This sample application uses express as web application framework (http://expressjs.com/),
6 // and jade as template engine (http://jade-lang.com/).
7
8 var express = require('express');
9
10 // setup middleware
11 var app = express();
12 app.use(app.router);
13 app.use(express.errorHandler());
14 app.use(express.static(__dirname + '/public')); //setup static public directory
15 app.set('view engine', 'jade');
16 app.set('views', __dirname + '/views'); //optional since express defaults to CWD/views
17
18 // render index page
19 app.get('/', function(req, res){
20   res.render('index');
21 });
22
23 // There are many useful environment variables available in process.env.
24 // VCAP_APPLICATION contains useful information about a deployed application.
25 var appInfo = JSON.parse(process.env.VCAP_APPLICATION || "{}");
26 // TODO: Get application information and use it in your app.
27
28 // VCAP_SERVICES contains all the credentials of services bound to
29 // this application. For details of its content, please refer to
30 // the document or sample of each service.
31 var services = JSON.parse(process.env.VCAP_SERVICES || "{}");
32 // TODO: Get service credentials and communicate with Bluemix services.
33
34 // The IP address of the Cloud Foundry DEA (Droplet Execution Agent) that hosts this application:
35 var host = (process.env.VCAP_APP_HOST || 'localhost');
36 // The port on the DEA for communication with the application:
37 var port = (process.env.VCAP_APP_PORT || 3000);
38 // Start server
39 app.listen(port, host);
40 console.log('App started on port ' + port);
41
42
43 });
  
```

➤ Note the Restrictions:

- Google Chrome is required.
- The app must use the IBM SDK for Node.js buildpack. Custom buildpacks are not supported.
- For more info on debugging in SDK for Node.js:  
[https://www.ng.bluemix.net/docs/#manageapps/bluemixlive.html#live\\_syn\\_c-edit](https://www.ng.bluemix.net/docs/#manageapps/bluemixlive.html#live_syn_c-edit)

- Back to Normal Mode: Restoring app configurations and disabling Bluemix Live Debug
- Remove the ENABLE\_BLUEMIX\_DEV\_MODE environment variable from the app manifest.yml file.
  - Restore the app's original start command and memory value.
  - Push the app.

**OUTPUTS:** Able to Switch IBM Java Liberty Runtime or IBM Node JS Runtime between Dev Mode and Normal Mode to perform Debugging on Cloud Application.