Auto-scaling enhancements for Node.js runtime

Improve the elasticity of your applications

Yang Qi

October 26, 2016

In the previous version of the Auto-Scaling for Bluemix® service, it only supported "memory" metrics for the Node.js application. By incorporating memory usage into the current version, we can evaluate an application's health and decide whether more instances of the evaluated application are needed. However, sometimes we need to control the number of application's instances more accurately.

Introduction

In the previous version of the Auto-Scaling for Bluemix® service, it only supported "memory" metrics for the Node.js application. By incorporating memory usage into the current version, we can evaluate an application's health and decide whether more instances of the evaluated application are needed. However, sometimes we need to control the number of application's instances more accurately.

The Auto-Scaling service recently released an enhancement for Node.js runtime. It provides two more metric types: heap and throughput. In Node.js, heap, the stack and code segment construct the main memory scheme. And heap, which stores objects and is the object of Node.js garbage collection mechanism, is important to evaluate a Node.js application's health. For a web application, its health condition can also be evaluated by how many requests that the application handled in the last period. As the more requests the application handles, the more resources will be used, and the health condition of it might decrease.

This article will demonstrate how to apply the Node.js application to the new enhancements of the Auto-Scaling service.

What you'll need to build your application

Did you know that developerWorks Premium not only gives you a 12-month Bluemix subscription, but also 240 USD credits to develop apps? You can also access over 50 security books with Premium’s Safari Books Online library! Check it out today!
• A Bluemix® account.
• A basic knowledge of Node.js programming, including:
  • Knowing how to write a web application by using Node.js
  • Knowing how to build the "package.json" file for a Node.js application
• Knowledge of HTML and JavaScript.
• A development environment that can upload a Node.js application to Bluemix, such as Eclipse.
• A basic knowledge of Apache JMeter™.

1. Prepare the application

To prepare your application, follow these steps:

1. Update the package.json of the application.
2. Create a dependency entry for "bluemix-autoscaling-agent" with:
   "bluemix-autoscaling-agent": "*"
3. The bluemix-autoscaling-agent is a Node.js module that can be installed from npm. It can collect the application's metrics and send it to Auto-Scaling service server after the application has been bound to an Auto-Scaling service instance.
4. Set the heap limit in the "script" according to the memory that you allocated for your application on Bluemix:
   "start": "node --max-old-space-size=600 app.js"
5. Set a value for "max-old-space-size" if you want to trigger scaling based on heap usage. If the value is not set when you start your application, the default Node.js heap limit of 1.4 GB will be used no matter how much memory you allocated in Bluemix, which might lead to improper auto-scaling decisions.

The following code is the package.json file content after it updates.

```json
{
  "name": "nodejsdemoapp",
  "version": "0.0.1",
  "description": "A sample nodejs app for Bluemix",
  "scripts": {
    "start": "node --max-old-space-size=600 app.js"
  },
  "dependencies": {
    "bluemix-autoscaling-agent": "*"
  },
  "repository": {},
  "engines": {
    "node": "0.12.x"
  }
}
```

6. Start the Auto-Scaling Node.js agent in your application, by adding the following line to your entry JavaScript

```javascript
var agent = require('bluemix_autoscaling-agent');
```
7. Push the application to Bluemix click **Add a Service or API** to bind the application to an Auto-

Scaling service instance.

8. Click **Auto-Scaling** from the services list.

9. Click **Create** and you will see the new Auto-Scaling service that was created. It is now bound
to the application that we just created.

2. **Create an Auto-Scaling policy**

To read more about assigning scaling rules and more details on applying JMeter, refer to the
"Make your application elastic on Bluemix" article for details.

You have now prepared your application and can now create your scaling rules.

Note that three metric types are supported. In the following image, we use "throughput" as an
example.
3. Add workload to the application and monitor its resource usage

In this tutorial, I used Apache JMeter to add the workload to our application. After the workload is run by JMeter, I then opened the Auto-Scaling dashboard and switched to the Metrics Statistics tab.

Notice in the following image that the throughput grows as the workload is running.

52.62 requests/s
When throughput reaches 50 requests per second (the upper threshold) over 120 seconds (the beach duration), a new instance is created and the total instance number is 2, which is illustrated in the following image.

### Conclusion

Now you know how to make your Node.js application work with the new Auto-Scaling enhancements on Bluemix. These new capabilities are easy to use and helps to improve the elasticity of your applications. Utilizing the IBM Auto-Scaling for Bluemix service also saves you on overall cost and on time and effort by automatically increasing or decreasing the CPU threshold of your application based on how you define your policy.
## Downloadable resources

<table>
<thead>
<tr>
<th>Description</th>
<th>Name</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Node.js Agent Demo</td>
<td>nodejsAgentDemo.zip</td>
<td>1MB</td>
</tr>
</tbody>
</table>
Related topics

- Node.js
- JavaScript
- IBM Bluemix: Your applications in the cloud

© Copyright IBM Corporation 2016
Trademarks
(www.ibm.com/developerworks/ibm/trademarks/)