Codelab: Building and Managing Your Application

Updated: April 2, 2015

This exercise leads you through deploying an existing App Engine application named ConferenceCentral and viewing it in the application dashboard for Quotas, Log files, etc. It will also guide you to configure your application for auto-scaling.

Special Note: Google is in the process of migrating from the AppEngine Console to the Developer Console. Because of this on-going migration this code lab may not provide you with the results described. This lab still provides a valuable view of the information that has been available in the old console and that will be available in the new console.

Special Note: Because of the migration the scaling parameters have been removed from the old console and not yet added to the new console. Instead of using the console, you will make changes to scaling parameters in a file named `app.yaml` for both Python and Java. This is the only Java code lab in the course that uses app.yaml instead of web.xml and appengine.xml.

Important: The instructions in this tutorial assume that you are using a terminal on your local machine and running a browser on that same machine to test against the local development server. You should also have Apache Bench installed. Please refer to the note on installing Apache Bench.

Contents

What You Will Do

- Deploy and view your hosted application on App Engine. We will be hosting a pre-built application named Conference Central.
- Test the auto scaling feature of App Engine using your application instances. By changing parameters in the `app.yaml` configuration file, you will see how to modify the App Engine scaling behavior and test those changes with a stress testing utility.

Get Set Up

1. You need to deploy your application to App Engine which requires an application ID (project ID). If you already have a project to test, you can skip this section. Otherwise, create a new project as follows:

   a. Visit Google Developers Console in your web browser, and click **Create Project**.
   b. Supply an appropriate project name and accept the project ID that is auto-generated. Please write down this application id; you will need this identifier to deploy the application.
   c. Click **Create**.
   d. Enable billing.

Get Ready

You need to download and then deploy an existing App Engine project.

We have prepared these projects for you and depending on your choice of Java or Python, the Project ZIP files for Conference Central application are given below:

- Java zip: ConferenceApp_1_Java_completed.zip
- Python zip: ConferenceApp_1_Python_completed.zip

Alternately, you can also clone the same projects (Java or Python) directly from GitHub:

```
git clone

https://github.com/GoogleCloudPlatformTraining/cp300-gae-conference-app-complete-1-java.git
```

OR

```
git clone

https://github.com/GoogleCloudPlatformTraining/cp300-gae-conference-app-complete-1-python.git
```

Deploying the Application

Follow the Java or Python instructions for deploying your application as given below:

Java Instructions

The project you downloaded is a Maven project. Follow the next steps to deploy the application.

1. Go to the command line or terminal and navigate to the root directory of the project, i.e. **cp300-gae-conference-app-complete-1-java**.

2. Build the project by giving the following command:

   ```
   mvn package
   ```

3. Go to the src/main/webapp/WEB-INF folder. Open the app.yaml file in an editor and provide the value for your application id in the first line. Save the file. You must also add this information to the appengine-web.xml file. Open this file in an editor and add the information in the XML tab <application></application>. Save the file.

4. Go back to the root folder, i.e. **cp300-gae-conference-app-complete-1-java,** and give the following command:

   ```
   mvn appengine:update
   ```

   You will be prompted for an authorization code in the terminal window and your web browser will launch with a consent screen which you must accept in order to be authorized. Follow the prompts to copy any codes from the browser to the command line. For more information refer to the App Engine Maven Guide.

Python Instructions

1. Find the file `app.yaml` and add your application identifier. Save the file.
2. Deploy your application by giving the following command

   ```
   appcfg.py update .
   ```

In the instructions below the directions are tailored for the Java version of the code lab. In each step, make similar changes to `app.yaml` and use the above command `appcfg.py update .` instead of `mvn appengine:update` to re-deploy the application.

Use the Admin Console to Monitor Your Application

Once the application is deployed successfully, make sure it is loaded and running. App Engine requires an initial request for the website before it starts an instance. To request the application, use your browser with the URL http://<project-id>.appspot.com. Once it is running, open the Admin Console for the application. Click your application from the list of applications. This will lead you to the Dashboard page.

In the Admin Console, view the versions page. You should see one version.

This is a good opportunity to start learning your way around the Admin Console.

View Charts

1. In the Dashboard page, view the “Requests by type/second” graph.
   a. Set the time period to 30 minutes.
   b. You might need to visit a few pages in your application to generate some requests.
2. Click through each of the different types of charts.
3. Refresh the page, then scroll down to the section “Current Load.” You can see the statistics for each of the pages.

View Logs

1. In the Admin Console, In the Main section, go to the Logs pages, and review the logs.
2. Every request to the application is recorded in the logs.
3. In the deployed Conference Central application, go to any of the pages.
4. Back in the logs, confirm your request was recorded.

View Quotas and Billing Settings

1. In the Admin Console, in the Main section in the left hand navigation panel, select Quota Details. Scroll down the page to see all the things that have a daily quota.
2. In the Billing (or Resources) section in the left hand navigation panel, select Billing Settings (or Billing Status).
3. Select the link to set up billing, then click Enable Billing, if you haven’t already set it up. Review the data you can set when enabling billing.
4. Change the daily budget, and notice that the weekly charges update.
5. Cancel out of this page -- There is no need to enable billing.

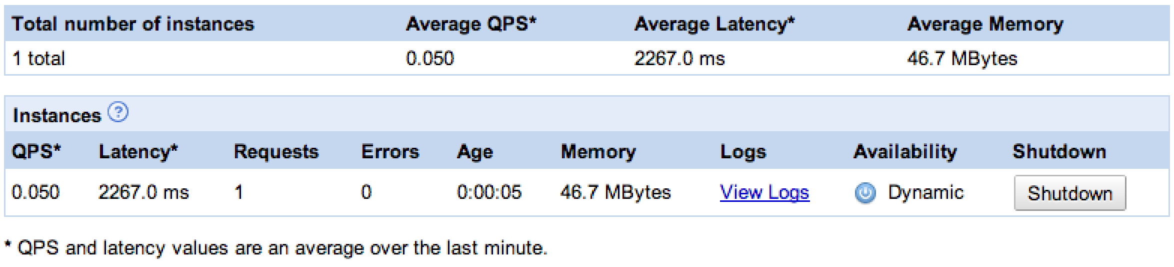
View Application Settings

1. In the Admin Console, in the Administration section in the left hand navigation panel, select Application Settings.
2. Scroll down the page to see the settings that you can change. Notice that you can delete or duplicate your application on this page.

3. Change your application's title and save the changes.
4. Optionally, if you preferred your previous title, change the application title back to what it was.
5. In the Administration section in the left hand navigation panel, select Permissions.
6. Optionally, invite another developer to collaborate on the application with you. Notice the different roles that a developer can have. (In the next exercise, you will limit access to the developers page of your application to developers who have a role in developing your application).
7. In the Administration section in the left hand navigation panel, select Admin Logs to see a history of things that administrators have done with the application.
8. Select the Events drop down menu to see the different kinds of events that can be listed.

Testing The Auto Scaler

1. In the left navigation menu, in the Main section, click the Instances link. This displays the instances page.
2. Confirm you do not that have any application instance running (*note: warmup requests should be disabled by default*). If you do have an instance running you can click the Shutdown button to terminate it.
3. Now, request the home page for the application at https://<APP_ID>.appspot.com to spawn a warm up request.
4. In the navigation menu, click the Instances link. This refreshes the instances page.
5. Confirm you have **one** application instance running as shown in the figure below



Total number of instances	Average QPS*	Average Latency*	Average Memory
1 total	0.050	2267.0 ms	46.7 MBytes

Instances

QPS*	Latency*	Requests	Errors	Age	Memory	Logs	Availability	Shutdown
0.050	2267.0 ms	1	0	0:00:05	46.7 MBytes	View Logs	Dynamic	Shutdown

* QPS and latency values are an average over the last minute.

Configuring the Application for Manual Scaling

You will be making changes to the application configuration information contained in the file "`app.yaml`". In the Python project this file is located at the root of the project. In the Java project it is located at `src/main/webapp/WEB-INF/app.yaml`. During deployment of the Java project, the files `web.xml` and `appengine.xml` will be automatically generated based on the information in `app.yaml`.

Use a text editor to open this file and notice that the tuning parameters are set to manual scaling with one instance:

```
manual_scaling:
  instances: 1
```

This setting is why you see only one instance when you first run the application.

cdChange this value to have two instances, save the file, and then re-deploy your application using the command:

```
mvn appengine:update
```

Each time you deploy a new configuration App Engine will terminate your current instances and wait for an initial request. After deploying the configuration with two manual instances:

```
manual_scaling:
  instances: 2
```

Re-request the application from your browser and then examine the console to see that two instances are now running.

Configuring the Application for Basic Scaling

With basic scaling you can control the maximum number of instances and the idle instance timeout. Edit the app.yaml file and replace the manual scaling parameters with the following basic scaling configuration:

```
basic_scaling:
  max_instances: 4
  idle_timeout: 10s
```

Re-deploy your application using the command:

```
mvn appengine:update
```

Now issue a single request for the application in your browser. This will trigger the creation of a single instance. Verify that only one instance is running in the console. Now wait about a minute and refresh the console screen. You should see that no instances are running. This illustrates a very lightly loaded system working within the parameters you have set - no more than four instance were run (actually, only one!) and after at least 10 seconds of idle time, the instance was shut down, leaving no running instances.

Run a benchmark to add stress to the application to force App Engine to create more than one instance.

1. Open a Terminal / Command Prompt window.

2. Execute the following command:

```
$ ab -n 50 -c 100 -t 60 <URL of your hello world application>
```

3. The `ab` (Apache Bench) command will issue 50 HTTP requests to the application from 100 concurrent users. If the command returns quickly with error messages, try a few times. You should get a message similar to this:

```
This is ApacheBench, Version 2.3 <$Revision: 655654 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd,
http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/
Benchmarking <your app id>.appspot.com (be patient)
```

4. While the command is benchmarking, go back to the Admin Console and click the Instances link. Confirm that up to four instances have been created.
5. After waiting about a minute, refresh the instances console page and notice that all instances have now stopped.

Configuring the Application for Automatic Scaling

With automatic scaling you have finer grained control over the scaling. You can specify a minimum number of instances to have idle - on reserve for a spike in traffic. You can control the maximum number of idle instance to keep when ramping down from a traffic spike. The minimum pending latency is the longest you are willing to have a request wait before a new instance is created.
Edit the app.yaml file and replace the manual scaling parameters with the following automatic scaling configuration:

```
automatic_scaling:
  min_idle_instances: 1
  max_idle_instances: 3
  min_pending_latency: 15s
  max_pending_latency: automatic
  max_concurrent_requests: 8
```

Re-deploy your application using the command:

```
mvn appengine:update
```

Now issue a single request for the application in your browser. This will trigger the creation of a single instance. Verify that only one instance is running in the console and also note that this is marked "resident". This is the always available, idle instance. No matter how long you wait, this instance will not be terminated.

Run a benchmark to add stress to the application to force App Engine to create more than one instance.

1. Open a Terminal / Command Prompt window.
2. Execute the following command:

```
$ ab -n 50 -c 100 -t 60 <URL of your hello world application>
```

Again, monitor the number of instances. You will probably notice two - one resident and one dynamic. The latency will probably be about 20ms to 30ms, which is well below the maximum specified (15s). Since the latency is below this value, no additional instances are created.

Continue to experiment with the automatic settings. Change the min_pending_latency to 10ms (which is below the average you saw previously. Re-deploy the application and re-run the stress test.

Monitor the instances page in the console and you should see between six and eight (the maximum number of concurrent requests) running. After the test finishes, you should see the number of instances drop back down; eventually only the resident instance will remain. (The time it takes to return to only one instance may be long because of the heuristics used).

Additional Resources

- Google App Engine Documentation
- Admin Console
- Developer Console
- How Applications Scale
- Apache Bench reference
- Conference Central Project in Go
- Adjusting Application Performance

Installing Apache Bench

Apache Bench is part of the Apache Web Server. On Mac, this application (ab) is likely to be installed already.

On Windows, if you have the Apache Web Server already installed on your machine, the Apache Bench application (ab) will be present in the /bin directory. Alternately, if you do not have Apache Web Server installed, please go to the Windows binaries for Apache httpd project. Select an appropriate distribution and install the same. The Apache Bench application (ab) will be present in the /bin folder.

On Ubuntu, you can install Apache tools (which includes Apache Bench) as follows:

```
sudo apt-get install apache2utils
```

Summary

In this exercise, you learned how to build and deploy apps with App Engine. You also learned how to tune Google App Engine’s auto scaler.