Error Reporting and Debugging (Stackdriver)

In this lab, you learn how to use Stackdriver Error Reporting and integrated Stackdriver Debugger.

Objectives

In this lab, you learn how to perform the following tasks:

- Launch a simple Google App Engine application
- Introduce an error into the application
- Explore Stackdriver Error Reporting
- Use Stackdriver Debugger to identify the error in the code
- Fix the bug and monitor in Stackdriver

Before you click the Start Lab button

Read these instructions. Labs are timed and you cannot pause them. The timer, which starts when you click Start Lab, shows how long Cloud resources will be made available to you.

This Qwiklabs hands-on lab lets you do the lab activities yourself in a real cloud environment, not in a simulation or demo environment. It does so by giving you new, temporary credentials that you use to sign in and access the Google Cloud Platform for the duration of the lab.

What you need

To complete this lab, you need:

- Access to a standard internet browser (Chrome browser recommended).
- Time to complete the lab.
  Note: If you already have your own personal GCP account or project, do not use it for this lab.

Task 1: Create an application

Get and test the application

1. In the GCP Console, launch Cloud Shell by clicking **Activate Cloud Shell** (). If prompted, click **Continue**.
2. To create a local folder and get the App Engine **Hello world** application, run the following commands:

```
mkdir appengine-hello
cd appengine-hello
gsutil cp gs://cloud-training/archinfra/gae-hello/* .
```

3. To run the application using the local development server in Cloud Shell, run the following command:

```
dev_appserver.py $(pwd)
```

4. In Cloud Shell, click **Web Preview** > **Preview on port 8080** to view the application. You may have to collapse the **Navigation menu** pane to access the **Web Preview** icon.

A new browser window opens to the localhost and displays the message **Hello, World!**

5. In Cloud Shell, press **Ctrl+C** to exit the development server.

Deploy the application to App Engine

1. To deploy the application to App Engine, run the following command:

```
gcloud app deploy app.yaml
```

2. If prompted for a region, enter the number corresponding to a region.

3. When prompted, type **Y** to continue.
4. When the process is done, verify that the application is working by running the following command:

```
gcloud app browse
```

If Cloud Shell does not detect your browser, click the link in the Cloud Shell output to view your app. You might have to refresh the page for the application to load.

5. If needed, press **Ctrl+C** to exit the development mode.

Click *Check my progress* to verify the objective.

Create an application

Check my progress

Introduce an error to break the application

1. To examine the `main.py` file, run the following command:

```
cat main.py
```

Notice that the application imports `webapp2`.

You will break the configuration by replacing the import library with one that doesnt exist.

2. To use the sed stream editor to change the import library to the nonexistent `webapp22`, run the following command:

```
sed -i -e 's/webapp2/webapp22/' main.py
```

3. To verify the change you made in the `main.py` file, run the following command:

```
cat main.py
```

Notice that the application now tries to import `webapp22`.

Re-deploy the application to App Engine

1. To re-deploy the application to App Engine, run the following command:

```
gcloud app deploy app.yaml --quiet
```

The `--quiet` flag disables all interactive prompts when running gcloud commands. If input is required, defaults will be used. In this case, it avoids the need for you to type Y when prompted to continue the deployment.

2. When the process is done, verify that the application is broken by running the following command:

```
gcloud app browse
```

If Cloud Shell does not detect your browser, click the link in the Cloud Shell output to view your app.

3. If needed, press **Ctrl+C** to exit development mode.
4. Leave Cloud Shell open.

ch service requires a logging agent installed to collect and send logs to Stackdriver?

Engine Standard

Engine Flexible

ernetes

Compute Engine

Submit

Click *Check my progress* to verify the objective.

Introduce an error to break the application

Check my progress

Task 2: Explore Stackdriver Error Reporting

View Error Reporting and trigger additional errors

1. In the GCP Console, on the **Navigation menu** (), click **Error Reporting**.

You should see an error regarding the failed import of `webapp22`.

2. Click **Auto reload**.

3. In Cloud Shell, run the following command:

```
gcloud app browse
```

If Cloud Shell does not detect your browser, click the link in the Cloud Shell output to view your app.

4. Click the resulting link several times to generate more errors.

The number of errors is displayed in the **Occurrences** column. The graph shows the frequency of errors over time, and the number represents the sum of errors. This is a very handy visual indicator of the state of the error.

The **First seen** and **Last seen** columns show when the error was first seen and when it was last seen, respectively. This can help identify changes that might have triggered the error. In this case, it was the upload of the new version of app engine code.

ch service(s) are currently supported by Stackdriver Error Reporting?

Engine Standard

pute Engine

ernetes

App Engine Flexible

Submit

View details and identify the cause

1. Click the Error name: **ImportError: No module named webapp22**.

Now you can see a detailed graph of the errors. The **Response Code** field shows the explicit error: a **500 Internal Server Error**.

2. For **Stack trace sample**, click **Parsed**. This opens the Stackdriver Debugger, showing the error in the code!

View the logs and fix the error

1. At the bottom of the Debug page, just below the code, find and open **View logs** in a new window or tab. Here you can find more detailed historical information about the error.
2. Introduce more errors by refreshing the page of your application. If you closed your application, use `gcloud app browse` and click the link to view the broken app.

3. On the **Stackdriver Error Reporting** page, ensure that **Auto Reload** is enabled to watch the addition of new errors.
4. In Cloud Shell, fix the error by running the following command:

```
sed -i -e 's/webapp22/webapp2/' main.py
```

5. To re-deploy the application to App Engine, run the following command:

```
gcloud app deploy app.yaml --quiet
```

6. When the process is done, to verify that the application is working again, run the following command:

```
gcloud app browse
```

If Cloud Shell does not detect your browser, click the link in the Cloud Shell output to view your app.

7. On the **Stackdriver Error Reporting** page, ensure that **Auto Reload** is enabled to and see that no new errors are added.

ıt would not be considered a benefit of Stackdriver?

sts all network performance

i-cloud monitoring

uces monitoring overhead

Faster problem resolution

Submit

Task 3: Review

In this lab you deployed an application to App Engine. Then you introduced a code bug and broke the application. You used Stackdriver Error Reporting to identify the issue, and then analyzed the problem, finding the root cause using Stackdriver Debugger. You modified the code to fix the problem, and then saw the results in Stackdriver.