Python (https://cloud.google.com/python/) Guides

Reporting errors

This tutorial shows how to report and track uncaught exceptions in the Python Bookshelf app (https://cloud.google.com/python/getting-started/tutorial-app) by using Google Cloud Console (https://cloud.google.com/error-reporting/docs/).

Stackdriver Error Reporting provides a centralized dashboard that shows counts of each unique error, stack traces, and a history of occurrences. You can also set up an alert for when errors occur.

This page is part of a multipage tutorial. To start from the beginning and read the setup instructions, go to Python Bookshelf app (https://cloud.google.com/python/monitor-and-debug/tutorial-app).

Configuring settings

This section uses code in the `5-logging` directory. Edit the files and run commands in this directory.

1. Open the `config.py` file for editing and replace the following values:
    - Set the value of `[PROJECT_ID]` to your project ID, which is visible in the Cloud Console.
    - Set the value of `[DATA_BACKEND]` to the same value you used during the Using structured data (https://cloud.google.com/python/getting-started/using-structured-data) tutorial.
    - If you are using Cloud SQL or MongoDB, set the values under the `Cloud SQL` or `Mongo` section to the same values you used during the *Using structured data* step.

- Set the value of `[CLOUD_STORAGE_BUCKET]` to your Cloud Storage bucket name.
- Under the `OAuth2 configuration` section, set the values of `[GOOGLE_OAUTH2_CLIENT_ID]` and `[GOOGLE_OAUTH2_CLIENT_SECRET]` to the application client ID and secret that you created previously.

2. Save and close the `config.py` file.

If you are using Cloud SQL:

1. Open the `app.yaml` file for editing.
2. Set the value of `cloudsql-instance` to the same value used for `[CLOUDSQL_CONNECTION_NAME]` in the `config.py` file. Use the format `project:region:cloudsql-instance`. Uncomment this entire line.
3. Save and close the `app.yaml` file.

Installing dependencies

To create a virtual environment and install dependencies, use the following commands:

LINUX/MACOS WINDOWS

```
virtualenv -p python3 env
source env/bin/activate
pip install -r requirements.txt
```

Running the app on your local machine

1. Start a local web server:

```
python main.py
```

2. In your browser, enter the following address:

```
http://localhost:8080
```

Press `Control+C` to exit the worker and then the local web server.

Deploying the app to the App Engine flexible environment

1. Deploy the sample app:

```
gcloud app deploy
```

2. In your browser, enter the following address. Replace `[YOUR_PROJECT_ID]` with your Google Cloud project ID:

```
https://[YOUR_PROJECT_ID].appspot.com
```

If you update your app, you deploy the updated version by entering the same command that you used to deploy the app. The deployment creates a new version of your app and promotes it to the default version. The earlier versions of your app remain, as do their associated virtual machine (VM) instances. All of these app versions and VM instances are billable resources. To reduce costs, delete the non-default versions of your app.

To delete an app version:

1. In the Cloud Console, go to the **Versions** page for App Engine.

   GO TO THE VERSIONS PAGE (HTTPS://CONSOLE.CLOUD.GOOGLE.COM/APPENGINE/VERSIONS)

2. Select the checkbox for the non-default app version you want to delete.

   Note: The only way you can delete the default version of your App Engine app is by deleting your project. However, you can stop the default version in the Cloud Console (https://console.cloud.google.com/appengine/versions). This action shuts down all instances associated with the version. You can restart these instances later if needed.

   In the App Engine standard environment, you can stop the default version only if your app has manual or basic scaling.

3. Click **Delete** to delete the app version.

For more information about cleaning up billable resources, see the Cleaning up (https://cloud.google.com/python/getting-started/using-pub-sub#clean-up) section in the final step of this tutorial.

Simulate an error

To see Stackdriver Error Reporting in action, intentionally introduce a mistake in your code, and then look for the exception in the Cloud Console's Stackdriver Error Reporting page.

1. In `bookshelf/crud.py`, add an operation that accesses an undefined variable and generates a `ReferenceError` in the index view.

```
@crud.route("/")
def list():
   x[3]
```

2. Deploy the app.

```
gcloud app deploy
```

3. Access the index page.

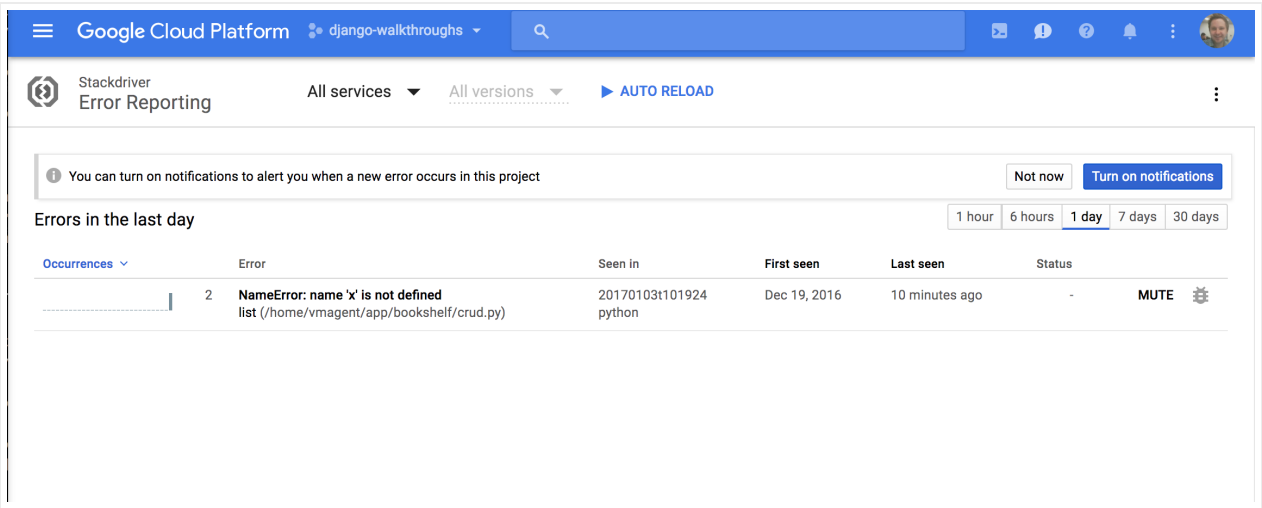
```
gcloud app browse
```

   You can view the message `An internal error occurred`.

4. In the Cloud Console, go to the Stackdriver **Error Reporting** page.

   GO TO ERROR REPORTING (HTTPS://CONSOLE.CLOUD.GOOGLE.COM/ERRORS)

   You can see the error listed.



5. Click the error to see information about the error, such as when the error was last seen,the number of times the error occurred, a histogram of occurrence times, and the stack trace.

Understanding the code

To report uncaught exceptions, the code first uses the Flask `errorhandler` decorator, and then reports the exception to Stackdriver Error Reporting by using the Cloud Client Libraries for Python (https://github.com/GoogleCloudPlatform/google-cloud-python).

5-logging/bookshelf/__init__.py
(https://github.com/GoogleCloudPlatform/getting-started-python/blob/504b3d550b551502cfe96f32542c31b232135eff/5-logging/bookshelf/__init__.py)

YTHON/BLOB/504B3D550B551502CFE96F32542C31B232135EFF/5-LOGGING/BOOKSHELF/__INIT__.PY)

```
@app.errorhandler(500)
def server_error(e):
    client = error_reporting.Client(app.config['PROJECT_ID'])
    client.report_exception(
        http_context=error_reporting.build_flask_context(request))
    return """
    An internal error occurred.
    """, 500
```

The client automatically adds the traceback info and uses a helper function to extract the relevant request details from the Flask request, which populates Error Reporting with the relevant stack traces and HTTP contexts for any uncaught `InternalServerError` `HTTP 500` exception in your app.

Note: The `500` error handler only triggers when the app isn't in `DEBUG` mode. `DEBUG` is on locally by default, so stack trace information is rendered in the browser. You can change the local debug setting in `main.py`.

Cleaning up

To avoid incurring charges to your Google Cloud Platform account for the resources used in this tutorial:

The easiest way to eliminate billing is to delete the project that you created for the tutorial.

To delete the project:

> **Caution**: Deleting a project has the following effects:
>
> - **Everything in the project is deleted.** If you used an existing project for this tutorial, when you delete it, you also delete any other work you've done in the project.
> - **Custom project IDs are lost.** When you created this project, you might have created a custom project ID that you want to use in the future. To preserve the URLs that use the project ID, such as an `appspot.com` URL, delete selected resources inside the project instead of deleting the whole project.
>
> If you plan to explore multiple tutorials and quickstarts, reusing projects can help you avoid exceeding project quota limits.

1. In the Cloud Console, go to the **Manage resources** page.

   GO TO THE MANAGE RESOURCES PAGE (HTTPS://CONSOLE.CLOUD.GOOGLE.COM/IAM-ADMIN/PRC

2. In the project list, select the project you want to delete and click **Delete** .
3. In the dialog, type the project ID, and then click **Shut down** to delete the project.

< PREV (HTTPS://CLOUD.GOOGLE.COM/PYTHON/MONITOR-AND-DEBUG/UPTIME-ALERT) NEXT >

Last updated November 20, 2019.