PHP (https://cloud.google.com/php/) Guides

Getting started with PHP

This tutorial is intended for those new to building apps in the cloud, such as engineers and web developers, who want to learn key app development concepts as they apply to Google Cloud.

Objectives

- Learn basic Google Cloud tools, such as the Google Cloud Console (https://cloud.google.com/cloud-console) and gcloud (https://cloud.google.com/sdk/gcloud).

- Deploy your app to the App Engine standard environment (https://cloud.google.com/appengine/docs/standard).

- Persist your data with Firestore (https://cloud.google.com/firestore).

- Store file uploads in Cloud Storage (https://cloud.google.com/storage).

- Monitor your app using Stackdriver (https://cloud.google.com/stackdriver).

For other language-specific tutorials for building your apps, see the following guides:

- Deploying an app to Google Kubernetes Engine (https://cloud.google.com/kubernetes-engine/docs/quickstarts/deploying-a-language-specific-app).

Costs

This tutorial uses the following billable components of Google Cloud:

- App Engine (https://cloud.google.com/appengine/pricing)

- Cloud Storage (https://cloud.google.com/storage/pricing)

- Firestore (https://cloud.google.com/firestore/pricing)

- Stackdriver (https://cloud.google.com/stackdriver/pricing)

The tutorial is designed to keep your resource usage within the limits of Google Cloud's Always Free (https://cloud.google.com/free/docs/frequently-asked-questions#always-free) tier. To generate a cost estimate based on your projected usage, use the pricing calculator (https://cloud.google.com/products/calculator). New Google Cloud users might be eligible for a free trial (https://cloud.google.com/free-trial).

When you finish this tutorial, you can avoid continued billing by deleting the resources you created. For more information, see Cleaning up (#clean-up).

Before you begin

1. Sign in (https://accounts.google.com/Login) to your Google Account.

   If you don't already have one, sign up for a new account (https://accounts.google.com/SignUp).

2. In the Cloud Console, on the project selector page, select or create a Google Cloud project.

   > **Note**: If you don't plan to keep the resources that you create in this procedure, create a project instead of selecting an existing project. After you finish these steps, you can delete the project, removing all resources associated with the project.

   GO TO THE PROJECT SELECTOR PAGE (HTTPS://CONSOLE.CLOUD.GOOGLE.COM/PROJECTSELECT

3. Make sure that billing is enabled for your Google Cloud project. Learn how to confirm billing is enabled for your project (https://cloud.google.com/billing/docs/how-to/modify-project).

4. To create a Firestore database in Native mode, complete the following steps:

   a. In the Cloud Console, go to the **Firestore viewer** page.

      GO TO THE FIRESTORE VIEWER (HTTPS://CONSOLE.CLOUD.GOOGLE.COM/FIRESTORE/DATA)

   b. From the **Select a Firestore mode** screen, click **Select Native Mode**.

   c. Select a location (https://cloud.google.com/firestore/docs/locations#types) for your Firestore database. This location setting is the default Google Cloud resource

location for your Google Cloud project
(https://cloud.google.com/firestore/docs/locations#default-cloud-location). This location is used for Google Cloud services in your Google Cloud project that require a location setting, specifically, your default Cloud Storage (https://cloud.google.com/storage/docs) bucket and your App Engine (https://cloud.google.com/appengine/docs/) app.

> Warning: After you set the default resource location for your Google Cloud project, you cannot change it.

    d. Click **Create Database**.

5. Enable the App Engine Admin, Cloud Storage, Stackdriver Logging, and Stackdriver Error Reporting APIs.

ENABLE THE APIS (HTTPS://CONSOLE.CLOUD.GOOGLE.COM/FLOWS/ENABLEAPI?APIID=APPENGIN

6. In Cloud Shell, open the app's source code.

GO TO CLOUD SHELL (HTTPS://CLOUD.GOOGLE.COM/CONSOLE/CLOUDSHELL/OPEN?GIT_BRANCH

Cloud Shell provides command-line access to your Google Cloud resources directly from the browser.

7. To download the sample code and change into the app directory, click **Proceed**.

8. In Cloud Shell, configure the `gcloud` tool to use your new Google Cloud project:

```
# Configure gcloud for your project
gcloud config set project PROJECT_ID
```

Replace **PROJECT_ID** with the Google Cloud project ID that you created using the Cloud Console.

The gcloud command-line tool (https://cloud.google.com/sdk/gcloud/) is the primary way you interact with your Google Cloud resources from the command line. In this tutorial, you use the `gcloud` tool to deploy and monitor your app.

Running your app

1. Install your app dependencies using `composer`:

```
composer install
```

2. Run the <u>PHP built-in web server</u>
   (https://www.php.net/manual/en/features.commandline.webserver.php):

```
GOOGLE_CLOUD_PROJECT=PROJECT_ID php -S localhost:8080
```

Replace **PROJECT_ID** with the Google Cloud project ID that you created.

3. In Cloud Shell, click **Web preview** ⊡, and select **Preview on port 8080**. This opens a new window with your running app.

Deploying your app to App Engine

Google Cloud offers <u>several options for running your code</u>
(https://cloud.google.com/hosting-options/). For this example, you use <u>App Engine</u>
(https://cloud.google.com/appengine/docs/standard/) to deploy a scalable app to Google Cloud.
With zero-configuration deployments and zero server management, App Engine lets you focus
on writing code. Plus, App Engine automatically scales to support sudden traffic spikes.

The `app.yaml` file is your main configuration file for deploying to App Engine:

<u>bookshelf/app.yaml</u>
(https://github.com/GoogleCloudPlatform/getting-started-php/blob/master/bookshelf/app.yaml)
THUB.COM/GOOGLECLOUDPLATFORM/GETTING-STARTED-PHP/BLOB/MASTER/BOOKSHELF/APP.YAML)

```
runtime: php72

env_variables:
  APP_DEBUG: true
  LOG_CHANNEL: stderr
  APP_STORAGE: /tmp
```

1. In your terminal window, deploy the app to App Engine using the `gcloud` tool:

```
# on the command-line
gcloud app deploy
```

2. Your app is now viewable at the following URL, where **PROJECT_ID** is the Google Cloud project ID that you created at the beginning of this tutorial:

```
https://PROJECT_ID.appspot.com
```

In your web browser, enter the URL to view the app.

> ## Bookshelf   Books
>
> # Books
> **+ Add book**

> **Note**: This SSL-protected domain is created automatically, and is useful for development. You can set up a custom domain
> (https://cloud.google.com/appengine/docs/standard/php7/mapping-custom-domains) with App Engine as well.

For more information on deploying to App Engine, see the PHP 7 runtime environment
(https://cloud.google.com/appengine/docs/standard/php7/runtime).

Persisting your data with Firestore

You cannot store information on your App Engine instances, because it is lost if the instance is restarted, and doesn't exist when new instances are created. Instead, you use a database that all your instances read from and write to.

Google Cloud offers several options for storing your data
(https://cloud.google.com/products/storage/). In this example, you use Firestore to store the data for each book. Firestore is a fully managed, serverless, NoSQL document database that lets you store and query data. Firestore auto scales to meet your app needs, and scales to zero when you're not using it. Add your first book now.

1. In your browser, go to the following URL, where **PROJECT_ID** is the project ID you created at the beginning of the tutorial.

```
https://PROJECT_ID.appspot.com
```

2. To create a book for your deployed app, click **Add book**.

Add book

Title

Author

Date Published

3. In the **Title** field, enter `Moby Dick`.

4. In the **Author** field, enter `Herman Melville`.

5. Click **Save**. There is now an entry to your Bookshelf app.

☑ Edit book   🗑 Delete book

Moby Dick October 18, 1851

By Herman Melville

An exhaustive tome for the nautical enthusiast.

6. In the Cloud Console, to refresh the Firestore page, click **Refresh** ↻ . The data appears in Firestore. The Bookshelf app stores each book as a Firestore document (https://cloud.google.com/firestore/docs/data-model#documents) with a unique ID, and all these documents are stored in a Firestore collection (https://cloud.google.com/firestore/docs/data-model#collections). For the purposes of this

tutorial, the collection is called books.

Root	books	4ff29575ae214b33836b
+ START COLLECTION	+ ADD DOCUMENT	+ START COLLECTION
⋮ books ❯	⋮ 4ff29575ae214b33836b ❯	+ ADD FIELD
		author: "Herman Melville"
		description: "An exhaustive tom...
		image_url: ""
		published_date: "October 18, 18...
		title: "Moby Dick"

Firestore stores the books by using the Firestore Client Library
(http://googleapis.github.io/google-cloud-php/#/docs/google-cloud/latest/firestore/readme). Here is an
example of fetching a Firestore document:

bookshelf/routes/web.php
(https://github.com/GoogleCloudPlatform/getting-started-php/blob/master/bookshelf/routes/web.php)
M/GOOGLECLOUDPLATFORM/GETTING-STARTED-PHP/BLOB/MASTER/BOOKSHELF/ROUTES/WEB.PHP)

```php
// Use the client library to call Firestore
use Google\Cloud\Firestore\FirestoreClient;

$firestore = new FirestoreClient([
    'projectId' => $projectId,
]);
$collection = $firestore->collection($collectionName);
```

```php
$bookRef = $collection->document($bookId);
$snapshot = $bookRef->snapshot();
```

For more information on using Firestore, see Adding data to Firestore
(https://cloud.google.com/firestore/docs/manage-data/add-data).

Storing file uploads in Cloud Storage

Now that you've added a book, it's time to add the book cover image. You cannot store files on
your instances. A database isn't the right choice for image files. Instead, you use Cloud Storage.

Cloud Storage (https://cloud.google.com/storage/docs/creating-buckets) is the primary blob store for Google Cloud. You can use Cloud Storage to host app assets that you want to share across Google Cloud. To use Cloud Storage, you need to create a Cloud Storage bucket (https://cloud.google.com/storage/docs/key-terms#buckets), a basic container to hold your data.

1. In the Cloud Console, go to the **Cloud Storage Browser** page.

   GO TO THE CLOUD STORAGE BROWSER PAGE (HTTPS://CLOUD.GOOGLE.COM/CONSOLE/STORAGE

2. Click **Create bucket**.

3. In the **Create bucket** dialog, enter a name for your bucket by appending your Google Cloud project ID to the string `_bucket` so the name looks like *YOUR_PROJECT_ID*`_bucket`. This name is subject to the bucket name requirements (https://cloud.google.com/storage/docs/bucket-naming#requirements). All other fields can remain at their default values.

4. Click **Create**.

5. After your bucket is created, click **Edit book**, and select an image to upload as your book's cover. For example, you can use this public domain image:

6. Click **Save**. You're redirected to the homepage, where there is an entry to your Bookshelf app.
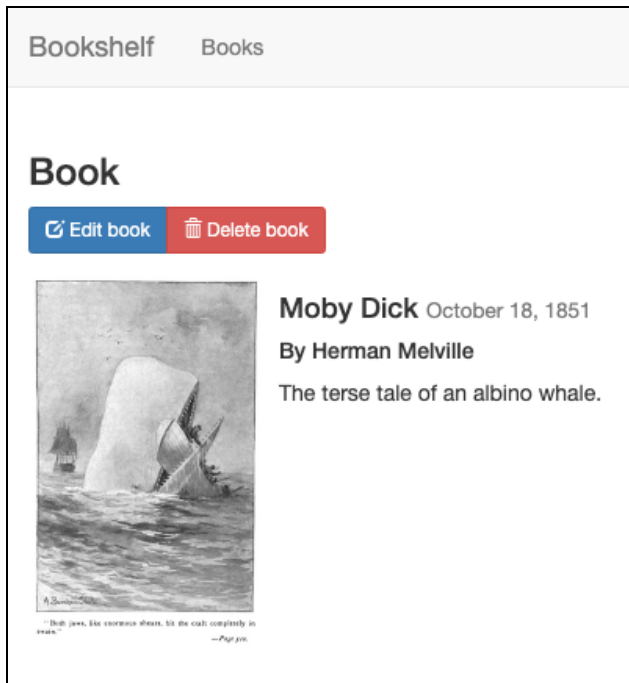
The bookshelf app sends uploaded files to Cloud Storage by using the Cloud Storage Client
Library (http://googleapis.github.io/google-cloud-php/#/docs/google-cloud/latest/storage/readme).

bookshelf/routes/web.php
 (https://github.com/GoogleCloudPlatform/getting-started-php/blob/master/bookshelf/routes/web.php)

M/GOOGLECLOUDPLATFORM/GETTING-STARTED-PHP/BLOB/MASTER/BOOKSHELF/ROUTES/WEB.PHP)

```php
// Use the client library to call Cloud Storage
use Google\Cloud\Storage\StorageClient;

$storage = new StorageClient([
    'projectId' => $projectId,
]);
$bucketId = $projectId . '_bucket';
$gcsBucket = $storage->bucket($bucketId);
```

Note: You can change the bucket name at any time by setting the `$bucketId` variable in the preceding code
to a different name and redeploying your app by using `gcloud app deploy`.

For more information on using Cloud Storage, see the list of how-to guides
 (https://cloud.google.com/storage/docs/how-to).

Monitoring your app using Stackdriver

You've deployed your app and created and modified books. To monitor these events for your users, use Stackdriver APM.

Monitor logs with Stackdriver Logging

CLOUD CONSOLE   GCLOUD

1. In your browser, go to the `/logs` URL in your app:

   ```
   https://PROJECT_ID.appspot.com/logs
   ```

   This sends a custom entry to Stackdriver Logging. The entry has a log severity (https://cloud.google.com/logging/docs/reference/v2/rest/v2/LogEntry#LogSeverity) of `NOTICE` and contains the message `Hey, you triggered a custom log entry. Good job!`.

2. Go to the Logs Viewer (https://cloud.google.com/console/logs), where you can monitor your app in real time. When something goes wrong, this is one of the first places to look.

3. In the resource drop-down list, select `GAE Application`.

4. In the logs drop-down list, select **All logs**.

There is a row for your custom log entry.

```
▶  ⚹   2019-05-01 14:36:00.406 PDT  GET   200      276 B     96 ms  AppEngine…  Hey, you triggered a custo…   ⋮
```

Monitor errors with Stackdriver Error Reporting

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

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

   Stackdriver Error Reporting highlights errors and exceptions in your app and lets you set up alerting around them.

2. In your browser, go to the `/errors` URL in your app.

   ```
   https://PROJECT_ID.appspot.com/errors
   ```

   This generates a new test exception and sends it to Stackdriver.

3. In the Cloud Console, return to the **Error Reporting** page, and in a few moments the new error is visible. Click **Auto Reload** so you don't need to manually refresh the page.

   Errors in the last hour

Resolution Status	Occurrences ⌄		Error
⊘ Open ▾	--------------	1	NEW PHP Notice: Google\Cloud\Core\Exception\FailedPreconditionException: { Google\Cloud\Core\GrpcRequestWrapper->handleStream (/srv/vendor/google/cloud-core/src/GrpcRequestWrapper.php)

Note: Stackdriver APM contains many tools to help debug and monitor your apps. For more information, see the tutorials (https://cloud.google.com/stackdriver/docs/tutorials).

Cleaning up

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

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/PRO) (HTTPS://CONSOLE.CLOUD.GOOGLE.COM/IAM-ADMIN/PRO

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.

What's next

- [Handle sessions with Firestore](https://cloud.google.com/php/getting-started/session-handling-with-firestore)
  (https://cloud.google.com/php/getting-started/session-handling-with-firestore)

- [Add background processing with Pub/Sub and Cloud Run](https://cloud.google.com/php/getting-started/background-processing)
  (https://cloud.google.com/php/getting-started/background-processing)

- [Deploy an app to Compute Engine](https://cloud.google.com/php/tutorials/getting-started-on-compute-engine)
  (https://cloud.google.com/php/tutorials/getting-started-on-compute-engine)

- [Deploy an app to Google Kubernetes Engine](https://cloud.google.com/kubernetes-engine/docs/quickstarts/deploying-a-language-specific-app)
  (https://cloud.google.com/kubernetes-engine/docs/quickstarts/deploying-a-language-specific-app)

- [Authenticate Users with IAP](https://cloud.google.com/php/getting-started/authenticate-users) (https://cloud.google.com/php/getting-started/authenticate-users)