**8. Deploying a cloud application on Google App Engine using backed database hosted on Google Cloud SQL to perform CRUD operations.**

**HARDWARE REQUIREMENTS**: Core I5 Processor, 4 GB RAM, 40GB HDD

**SOFTWARE REQUIREMENTS**: Google Cloud Platform, Compute Engine, Google cloud shell, GIT, Composer, Google Cloud SDK for windows.

**Description:**

We will clone a sample PhP application from GitHub that allows us to perform CRUD operations on a Bookshelf application, allowing us to insert, retrieve, update and delete the list of books. This application is deployed as a Google App engine application and the CRUD operations are performed over a Google Cloud SQL instance.

**Setup Google Cloud SDK**

1. To deploy your app with the gcloud tool, you must download, install, and initialize the Cloud SDK, for the appropriate OS
   1. Windows: https://dl.google.com/dl/cloudsdk/channels/rapid/GoogleCloudSDKInstaller.exe
   2. Start Cloud Shell and enter gcloud init command to login into GCloud environment using GCloud SDK
   3. Enter the project ID taken from Google cloud
   4. Select the zone where the project is deployed

**Downloading Sample App**

1. Download or clone the app and Navigate to the getting-started directory

https://github.com/GoogleCloudPlatform/php-docs-samples.git

cd appengine/php72/getting-started

**Informing dependencies using composer.json file**

1. Open the composer.json file to review all direct dependencies

{

"require": {

"google/cloud-storage": "^1.6",

"slim/slim": "^4.0",

"slim/twig-view": "^3.0",

"slim/http": "^1.0",

"slim/psr7": "^1.0"

},

"autoload": {

"psr-4": {

"Google\\Cloud\\Samples\\AppEngine\\GettingStarted\\": "src"

}

},

"require-dev": {

}

}

1. Run ***composer install*** to download dependencies and produce a composer.lock file. The composer.lock file is used to ensure your app will retrieve the same versions of the packages you use across multiple builds and environments.

**Initializing the app and defining front controllers**

1. The index.php file initializes the app and forwards all requests to controllers defined in the ./src/controllers.php file.

// Use the composer autoloader to load dependencies.

require\_once \_\_DIR\_\_ . '/vendor/autoload.php';

// Load the application code.

/\*\* @var Slim\App $app \*/

$app = require \_\_DIR\_\_ . '/src/app.php';

require \_\_DIR\_\_ . '/src/controllers.php';

// Bootstrap the slim framework to handle the request.

$app->run();

**Configuring the Cloud SQL instance**

1. To create and configure a Cloud SQL instance:
   1. Create a Cloud SQL Second Generation instance.

gcloud sql instances create INSTANCE\_NAME --tier=MACHINE\_TYPE --region=REGION

gcloud sql instances create bookshelf --tier=db-n1-standard-2 --region=us-central1

Or using the Google Cloud SQL console

* 1. If you haven't already, set the password for the default user on your Cloud SQL instance:

gcloud sql users set-password root --host=% --instance [INSTANCE\_NAME] --password [PASSWORD]

gcloud sql users set-password root --host=% --instance=bookshelf --password=root

* 1. If you don't want to use the default user to connect, create a user.
  2. Record the connection name for the instance:

gcloud sql instances describe [INSTANCE\_NAME]

* 1. For example:

connectionName:angularjs-crud-project:us-central1:instance1

* 1. You can also find this value in the Instance details page of the Google Cloud Platform Console.
  2. Create a database on your Cloud SQL instance.

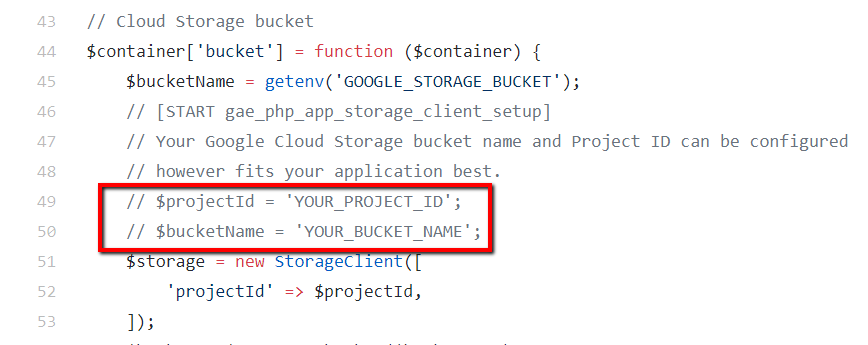
gcloud sql databases create [DATABASE\_NAME] --instance=[INSTANCE\_NAME]

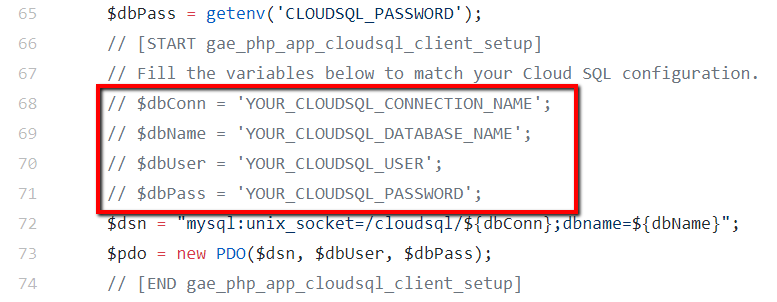
gcloud sql databases create book-data --instance=bookshelf

**Connecting to Cloud SQL database**

1. This sample app uses PHP's PDO to interact with the MySQL database.

appengine/php72/getting-started/src/app.php





**Query a Single Row**

1. When the user clicks on a book, the app queries the database and returns a single row that includes the title, author, publication date, and description of the book.

$statement = $pdo->prepare('SELECT \* FROM books WHERE id = :id');

$statement->bindValue('id', $id, PDO::PARAM\_INT);

$statement->execute();

$result = $statement->fetch(PDO::FETCH\_ASSOC);

**Using Cloud Storage**

1. Cloud Storage uses buckets to organize and control access to your data.
   1. Use Cloud SDK to create a Cloud Storage bucket:

gsutil mb -l BUCKET\_REGION gs://BUCKET\_NAME/

gsutil mb -l us-central1 gs://picture-storage/

If the storage bucket creation is successful, then you get

Creating gs://picture-storage/

Else

ServiceException: 409 Bucket picture-storage already exists

**App Deploy**

1. Go back to the ***getting-started*** folder and run the command

*gcloud app deploy*

**Input/Output:**