1. Overview

In this lab, you explore Google App Engine using the Bookshelf sample application. Bookshelf uses Google Cloud Datastore to store data persistently. You deploy the application to the App Engine standard runtime environment, and then you test saving data to Cloud Datastore.

What you need

To complete this lab, you need:

Access to a supported Internet browser:

- The latest version of Google Chrome, Firefox, or Microsoft Edge
- Microsoft Internet Explorer 11+
- Safari 8+ (Safari private mode is not supported)

A Google Cloud Platform project

What you learn

In this lab, you:

- Deploy a sample Python application called Bookshelf to the App Engine standard runtime environment
- Test the Bookshelf application and inspect data saved to Cloud Datastore

2. Introduction

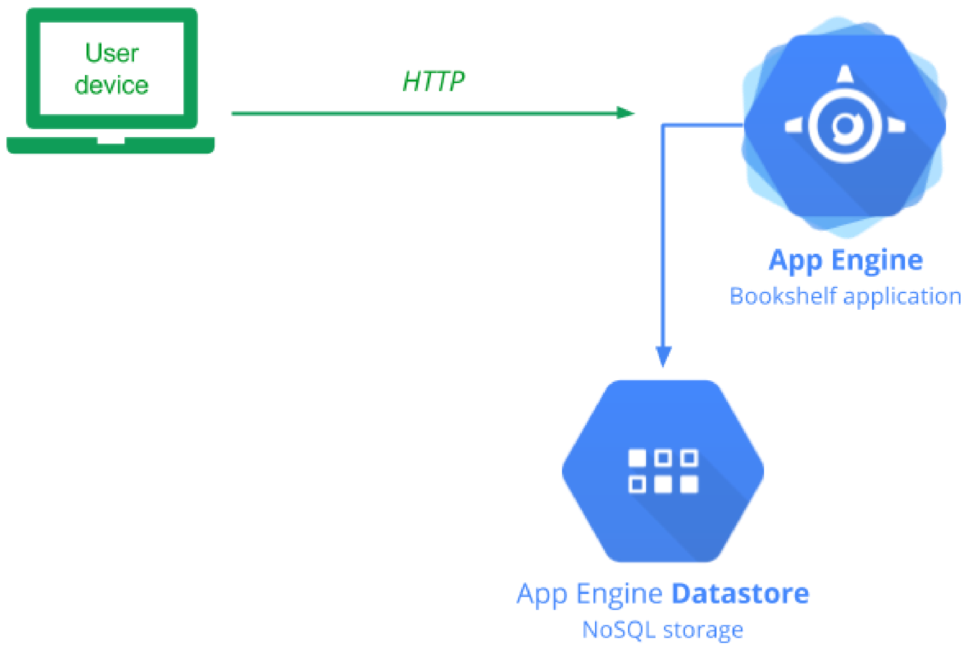
In this lab, you explore both Google App Engine and Google Cloud Datastore using the Bookshelf sample application. App Engine is a managed computing environment that lets you focus on writing and running code. As you see in this lab, you do not need to create, configure, or manage any servers, storage, or networks to get started. App Engine handles all of the web traffic generated by your users and automatically scales to meet changes in traffic levels.

Bookshelf is a simple Python web application that lets users create and manage a list of books. The application uses the Flask web microframework to coordinate reading and writing to storage. You can also use a variety of other frameworks to build Python applications on App Engine, including webapp2, and Django.

This version of Bookshelf uses Cloud Datastore to persistently store book data. Google Cloud Datastore is a NoSQL document database, and compliments App Engine through automatic scaling, high performance, and ease of application development.

As you observe in this lab, Bookshelf uses a programming design pattern called model-view-controller, separating the concerns of storage, business logic, and user interface (or view). This simplifies the process of swapping Cloud Datastore, for an alternative service such as Cloud SQL. You can even build, configure, and substitute your own custom storage solution, such as MongoDB to work with Bookshelf. You can find detailed instructions on customizing Bookshelf in the getting started on Google Cloud Platform guide for Python. Tutorials are also available for other programming languages; choose from the list on the Google Cloud Platform Documentation home page.

The following diagram illustrates the high-level components and resources that make up the version of Bookshelf you deploy in this lab.



You also make use of a number of Google Cloud Platform services and features in this lab to assist with the development process. Each Google Cloud Platform project supports Git-compatible Google Cloud Source Repositories that you can use to manage the code for your application.

You also use Google Cloud Shell, a managed command-line environment, to commit code to your repository and deploy code to App Engine. Cloud Shell provides access to many of the tools and utilities typically required to manage Google Cloud Platform projects, including the Cloud SDK you use in this lab. Finally, you use the Google Cloud Platform Console to access your application, and to review data stored in Cloud Datastore.

3. Clone the source code into your project

In this section of the lab, you use Google Cloud Shell to clone a Github repository and to populate a Cloud Source repository with your application source code.

To clone and populate the repository:

Step 1

Open the Google Cloud Platform Console, and if necessary, select the **cp100** project.

Step 2

In the Cloud Platform Console navigation pane, click **Tools > Source Repositories > Source Code**.

Step 3

Click **Repositories**.

Step 4

If you already have a repository called `default`, skip to step 6. If not, click **CREATE REPOSITORY**.

Step 5

In the 'Create New Repository' dialogue:

- For **Repository Name**, type **default**.
- Click **Create**.

Step 6

In the top right corner of the console window, click the **Activate Google Cloud Shell** button ( ).

You can use **Cloud Shell** to:

- Interact with hosted or remote Git repositories, including Google Cloud Source Repositories
- Build and test Google App Engine applications

Note

Cloud Shell comes with a number of development tools installed, such as Python, Java, Cloud and App Engine SDKs, and Git.

If a box titled "Google Cloud Shell" appears, read its contents to learn more about Cloud Shell, then dismiss it by clicking **Start Cloud Shell**.

Step 7

All of the code you need for this course is available in a single repository, which you copy to the source code repository in your project.

You use a local clone of your repository in Cloud Shell to deploy and manage the Google Cloud Platform services and resources in your project.

Type the following command to create a directory for the repository.

```
mkdir cp100
```

Step 8

Type the following command to change to the cp100 directory.

```
cd cp100
```

Step 9

Each Google Cloud Platform project comes with a single Cloud Source Repository called ‘default', which is currently empty. Type the following command to clone your default repository.

```
gcloud source repos clone default
```

The output describes the progress of the clone operation and notes that your repository is currently empty.

Step 10

Type the following command to change to the default directory.

```
cd default
```

Step 11

The code for this course is hosted on GitHub. Type the following command to pull the code into your empty project repository.

```
git pull  https://github.com/GoogleCloudPlatformTraining/cp100-
bookshelf
```

The output confirms that the code has been downloaded to your local repository.

Step 12

Type the following command to push your copy of the code to your project repository.

```
git push origin master
```

The output displays the progress of the push to the default Cloud Source Repository in your project.

Step 13

You can now inspect your copy of the code using the Cloud Platform Console.

In the console, click **Source Code**, and verify you are using the default repository.

You are presented with a list of directory and file names that make up the source code for this course. All of the code required for each of the individual labs is contained in a separate directory.

- **`app-engine`** - Hosts the source code for the application you deploy in this lab.
- **`cloud-storage`** - Hosts the source code for a variation of the application you deploy in this lab that includes support for saving data to Cloud Storage.
- **`compute-engine`** - Contains the source code for a version of the Bookshelf application you deploy in this lab that runs on Compute Engine.
- **`container-engine`** - Holds the source code for a version of the Bookshelf application you deploy in this lab that runs on Container Engine.
- **`.gitignore`** - Used to configure files that should not be managed in your Git repository. There is no need to modify or review this file.

The remainder of the files listed contain supporting documentation for the original GitHub source code repository:

- **`CONTRIBUTING.md`**
- **`LICENSE`**
- **`README.md`**

In the next section, your investigate the source code for this lab in the `app-engine` directory.

Step 14

Leave both the Cloud Platform Console and Cloud Shell windows open.

4. Review Bookshelf code

Before you deploy the Bookshelf application, it is useful to review some of the code involved in making it work.

To review the code:

Step 1

Return to the Source Code browser in the Cloud Platform Console and click the **app-engine** folder.

You are presented with a list of directory and file names that make up the source code for this lab.

- `bookshelf`
- `app.yaml`
- `appengine_config.py`
- `config.py`

- `main.py`
- `requirements.txt`

Most of the application code can be found in the `bookshelf` directory. You explore most of the code in this repository throughout the remainder of this lab.

Step 2

Click **app.yaml**.

An App Engine `app.yaml` configuration file includes a variety of configuration data as required, and optional elements, used to deploy and manage the behavior of your application.

Note in particular, on **line 19** that the application runtime is set to Python version 2.7.

Also note, that **lines 23 through 25** define the only script handler for this application, that describes how all requests to the application (`/.*` indicates all requests), should be handled by `main.app`. A mapping typically defines a URL pattern to match, and the script to be executed. You examine the source code for `main.py` in the next step.

Step 3

Click the dropdown menu between **app-engine** and **app.yaml** and choose **main.py**.

Notice that this file simply imports the bookshelf code and loads any relevant configuration data into the application.

The file also includes some code used to allow the application to run locally using the development server included with the App Engine SDK.

Step 4

Click **/ > app-engine > config.py**.

This file is used to manage configuration data that is specific to your project and copy of the application. You make changes to this file later in the course but do not need to make any changes to it in this lab.

Step 5

Click **/ > app-engine > bookshelf**.

You are presented with another list of directory and file names that make up the source code for the Bookshelf application.

- **`templates`** - Contains HTML files that define the view, or user interface (UI) layer of the application
- **`__init__.py`** - Initalizes the Flask web microframework, loads the storage library and routes requests to the correct part of the application
- **`crud.py`** - Acts as a controller layer (you examine this file later)

- `model_cloudsql.py` - Includes code used to read and write data using Cloud SQL (not currently used in this version of the course)
- `model_datastore.py` - Includes code used to read and write data using Cloud Datastore, the storage layer of the application

Note

Bookshelf is built using the third-party Python Flask Microframework, that is not covered in this course. For more information about Flask, please refer to the Flask project website.

Step 6

Click **crud.py** to examine the file.

This file acts as a controller layer and includes a number of functions that coordinate create, read, update, delete operations for the Bookshelf application.

Examine the `list()` function on **lines 22 through 32**. Notice that the code doesn't include any direct reference to the type of storage being used. This code abstracts away the specific operations required to read and write from the chosen storage system and passes data between the HTML ‘view' layer, and the storage model. In this lab, you use Cloud Datastore to store data relating to book entries you create in Bookshelf. One advantage of this loosely coupled programming model, is that it is simpler to substitute Cloud Datastore for another storage service such as Cloud SQL during development.

Later in the lab, and after you have deployed and tested the Bookshelf application, you examine the Cloud Datastore code.

Step 7

Leave the Cloud Platform Console and Cloud Shell windows open.

5. Deploy Bookshelf

Install all required third-party libraries and dependencies to the local clone of your repository on Cloud Shell. Deploy the application to App Engine using the Cloud SDK and then use the Cloud Platform Console to test the application.

To deploy Bookshelf:

Step 1

Return to Cloud Shell, and type the following command to change to the directory containing the code for this lab.

```
cd ~/cp100/default/app-engine
```

Step 2

In addition to the code you copied from GitHub, you also need to install a number of third-party libraries that the Bookshelf application uses. If you would like to see a list of the libraries you need for this lab, type the following command to view the contents of the `requirements.txt` text file.

```
cat requirements.txt
```

The file lists package names and version numbers that you install in the next step.

Step 3

You use the `pip` command to install Python packages listed as requirements. The `pip` utility is already installed in Cloud Shell. Type the following command to install the packages and dependencies required for this lab into the `lib`directory.

Note

pip is a command line package manager for software written in Python.

```
pip install -r requirements.txt -t lib
```

The output displays the progress of the installations.

Step 4

Once deployed to App Engine, the libraries you installed in the previous step are loaded from the `lib` directory using the code in `appengine_config.py`. App Engine automatically loads this file when a new instance of your application is started. Type the following command to view the contents of `appengine_config.py`.

```
cat appengine_config.py
```

The vendor subpackage is imported from the google.appengine.ext package on **line 7**. This is then used on **line 11** to add the contents of the `lib` directory and make it available for importing in the Bookshelf application.

Step 5

You are now ready to deploy the Bookshelf application to App Engine. You use the gcloud command-line utility that is already installed in Cloud Shell to manage your App Engine application. Type the following command to deploy the application.

```
gcloud app deploy
```

The system asks you into which GCP region you want to deploy your application.

```
Please choose the region where you want your App Engine application
located:

[1] europe-west2 (supports standard and flexible)
```

```
[2] us-east1 (supports standard and flexible)

[3] us-east4 (supports standard and flexible)

[4] asia-northeast1 (supports standard and flexible)

[5] asia-south1 (supports standard and flexible)

[6] australia-southeast1 (supports standard and flexible)

[7] southamerica-east1 (supports standard and flexible)

[8] us-central (supports standard and flexible)

[9] europe-west3 (supports standard and flexible)

[10] europe-west (supports standard and flexible)

[11] cancel

Please enter your numeric choice:
```

Enter a choice, using its number in the list presented, that is geographically close to you.

You are asked to confirm that you want to proceed with the deployment:

```
Do you want to continue (Y/n)?
```

Type **Y** and press **return**.

The output displays the progress of the deployment.

The deployment has successfully completed once you see the following message.

```
Deployed service [default] to [https://<project-id>.appspot.com]
```

Step 6

Type the following command to quit Cloud Shell.

```
exit
```

Step 7

You can now visit and test the deployed Bookshelf application.

In the Cloud Platform Console, click **Compute > App Engine**.

The App Engine Dashboard loads.

Step 8

In the top right-hand corner of the Dashboard, click the link to your deployed application. The link takes the format:

```
<project-id>.appspot.com
```

The Bookshelf application loads in a new browser tab.

Note

You can also visit your application directly by typing the URL in your browser.

Step 9

Leave all browser tabs open.

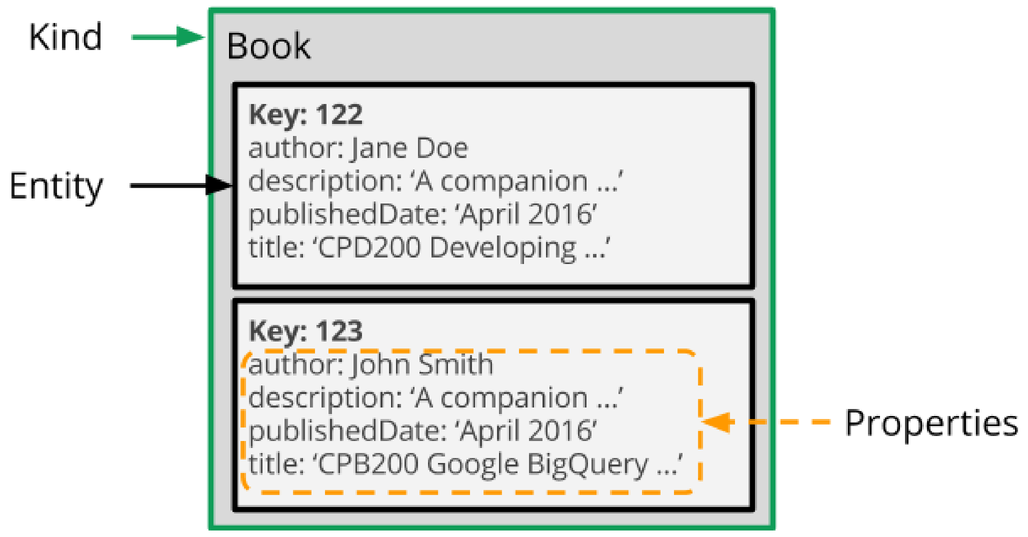
6. Cloud Datastore overview

A kind in Cloud Datastore is roughly equivalent to a table in a relational database. Kinds are used to categorize entities for queries.

Data objects in Google Cloud Datastore are known as entities. An entity has one or more named properties (which are like fields in a relational database). Each property can have one or more values. Entities are identified by a unique key and are akin to rows in a relational database. Entities of the same kind need not have the same properties. These unique characteristics imply a different way of designing and managing data to take advantage of the ability to scale automatically.

Cloud Datastore also organizes data into entity groups. An entity group consists of a root entity and all of its descendants. Applications typically use entity groups to organize highly related data to achieve strong consistency and to enforce transactionality. For example, an application could use an entity group to store data about one product, or one user profile.

The Bookshelf data model for Cloud Datastore is illustrated by the following diagram. Note that the simple data model used in Bookshelf does not make use of entity groups.


Kind
Book
Entity
Key: 122
author: Jane Doe
description: 'A companion ...'
publishedDate: 'April 2016'
title: 'CPD200 Developing ...'
Key: 123
author: John Smith
description: 'A companion ...'
publishedDate: 'April 2016'
title: 'CPB200 Google BigQuery ...'
Properties

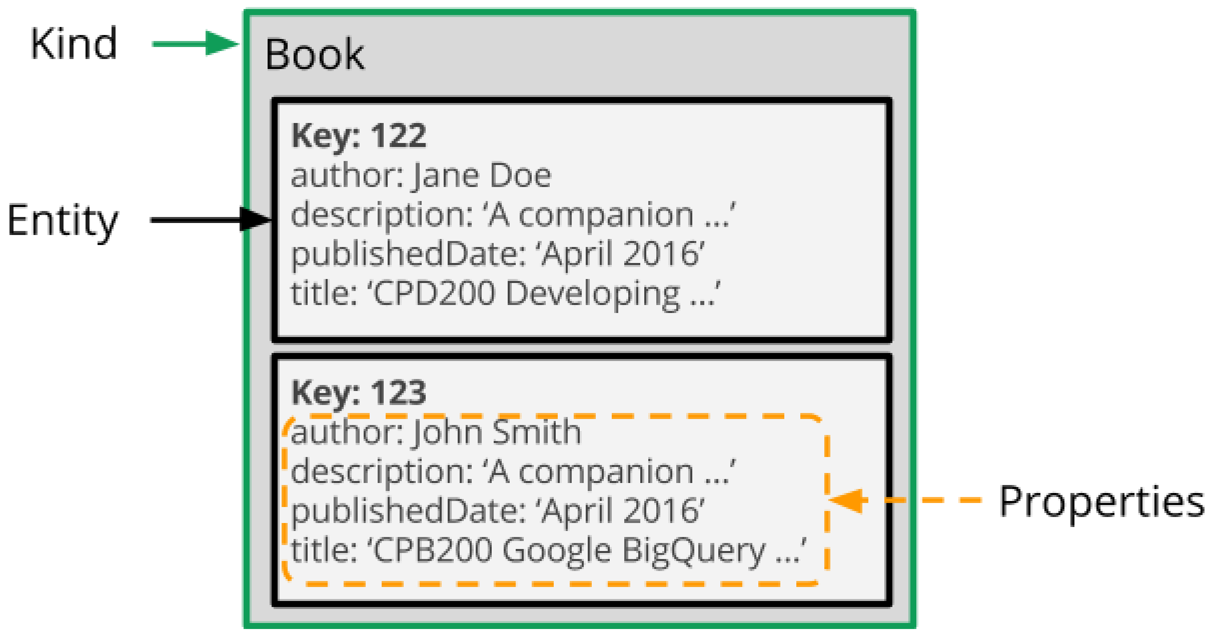
6. Cloud Datastore overview

A kind in Cloud Datastore is roughly equivalent to a table in a relational database. Kinds are used to categorize entities for queries.

Data objects in Google Cloud Datastore are known as entities. An entity has one or more named properties (which are like fields in a relational database). Each property can have one or more values. Entities are identified by a unique key and are akin to rows in a relational database. Entities of the same kind need not have the same properties. These unique characteristics imply a different way of designing and managing data to take advantage of the ability to scale automatically.

Cloud Datastore also organizes data into entity groups. An entity group consists of a root entity and all of its descendants. Applications typically use entity groups to organize highly related data to achieve strong consistency and to enforce transactionality. For example, an application could use an entity group to store data about one product, or one user profile.

The Bookshelf data model for Cloud Datastore is illustrated by the following diagram. Note that the simple data model used in Bookshelf does not make use of entity groups.


Kind
Book
Entity
Key: 122
author: Jane Doe
description: 'A companion ...'
publishedDate: 'April 2016'
title: 'CPD200 Developing ...'
Key: 123
author: John Smith
description: 'A companion ...'
publishedDate: 'April 2016'
title: 'CPB200 Google BigQuery ...'
Properties

8. Clean up

There is no need for you to remove the resources used in this lab.