Ruby (https://cloud.google.com/ruby/) Guides

Using Cloud Storage with Ruby on Rails

This part of the Bookshelf tutorial shows how the sample app stores images in Cloud Storage.

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

Creating a Cloud Storage bucket

Cloud Storage lets you store and serve binary data. A bucket is a high-level container for binary objects.

The following instructions detail how to create a Cloud Storage bucket. Buckets are the basic containers that hold your data in Cloud Storage.

Note: You can choose any name (https://cloud.google.com/storage/docs/naming) for your Cloud Storage bucket. It's a good practice to give your bucket the same name as your project ID. Bucket names must be unique across all of Google Cloud, so it's possible that you can't use your project ID as the bucket name.

1. In your terminal window, create a Cloud Storage bucket, where ***YOUR_BUCKET_NAME*** represents the name of your bucket:

   ```
   gsutil mb gs://YOUR_BUCKET_NAME
   ```

2. To view uploaded images in the Bookshelf app, set the bucket's default access control list (ACL) to `public-read`:

   ```
   gsutil defacl set public-read gs://YOUR_BUCKET_NAME
   ```

Installing dependencies

Go to the `getting-started-ruby/3-cloud-storage` directory, and enter the following command:

```
bundle install
```

Configuring settings

1. Copy the example `settings` file:

   ```
   cp config/settings.example.yml config/settings.yml
   ```

2. Open `settings.yml` for editing. Replace the placeholders with your project and Cloud Storage bucket names.

   For example, suppose your project name is `my-project` and your bucket name is `my-bucket`. Then the `default` section of your `settings.yml` file would look like this:

   ```
   default: &default
     project_id: my-project
     gcs_bucket: my-bucket
   ```

3. Copy the example `database` file:

   ```
   cp config/database.example.yml config/database.yml
   ```

4. Configure the sample app to use the same database that you set up during the Using structured data (https://cloud.google.com/ruby/getting-started/using-structured-data) portion of this tutorial:

   CLOUD SQL POSTGRESQL DATASTORE

   - Edit `database.yml`. Uncomment the lines in the Cloud SQL portion of the file.

     ```
     mysql_settings: &mysql_settings
       adapter: mysql2
       encoding: utf8
       pool: 5
     ```

```
   timeout: 5000
   username: [MYSQL_USER]
   password: [MYSQL_PASS]
   database: [MYSQL_DATABASE]
   socket: /cloudsql/[YOUR_INSTANCE_CONNECTION_NAME]
```

- Replace `[MYSQL_USER]` and `[MYSQL_PASS]` with your Cloud SQL instance username and password that you created previously.
- Replace `[MYSQL_DATABASE]` with the name of the database that you created previously.
- Replace `[YOUR_INSTANCE_CONNECTION_NAME]` with the `Instance Connection Name` of your Cloud SQL instance.

Note: You can retrieve the Cloud SQL instance connection name by running `gcloud beta sql instances describe [YOUR_INSTANCE_NAME]`.

- Run migrations.

```
bundle exec rake db:migrate
```

Running the app on your local machine

1. Start a local web server.

```
bundle exec rails server
```

2. In your web browser, enter the following address:

   http://localhost:3000 (http://localhost:3000)

Now you can browse the app's web pages, add books with cover images, edit books, and delete books.

To exit the local web server, press `Control+C`.

Deploying the app to the App Engine flexible environment

1. Compile the JavaScript assets for production.

   ```
   RAILS_ENV=production bundle exec rake assets:precompile
   ```

2. Deploy the sample app.

   ```
   gcloud app deploy
   ```

3. In your web browser, enter the following address.

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

If you update your app, you can deploy the updated version by entering the same command you used to deploy the app the first time. The new deployment creates a new version (https://console.cloud.google.com/appengine/versions) of your app and promotes it to the default version. The older versions of your app remain, as do their associated VM instances. Be aware that all of these app versions and VM instances are billable resources.

You can reduce costs by deleting 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.

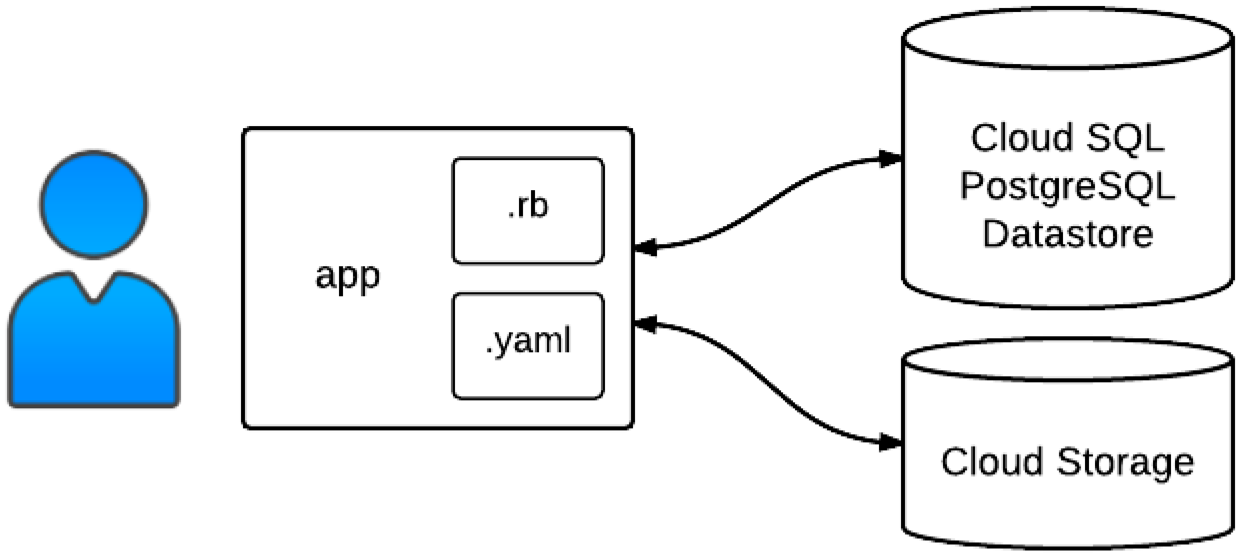
   In the App Engine standard environment, you can stop the default version only if your app has manual or basic scaling.

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

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

App structure

The following diagram shows the app's components and how they connect to each other.



The app uses {storage_name_short}} to store binary data, pictures in this case, while continuing to use a structured database for the book information, either Datastore, Cloud SQL, or PostgreSQL.

Understanding the code

This section walks you through the app's code and explains how it works.

Handle user uploads

To allow users to upload images, the add/edit form was modified to allow file uploads. The form is now multipart.

3-cloud-storage/app/views/books/_form.html.erb
(https://github.com/GoogleCloudPlatform/getting-started-ruby/blob/steps/3-cloud-storage/app/views/books/_form.html.erb)

)RM/GETTING-STARTED-RUBY/BLOB/STEPS/3-CLOUD-STORAGE/APP/VIEWS/BOOKS/_FORM.HTML.ERB)

```
<%= form_for @book, :html => { :multipart => true } do |f| %>
```

Also, the form has a new field for a book cover image.

3-cloud-storage/app/views/books/_form.html.erb (https://github.com/GoogleCloudPlatform/getting-started-ruby/blob/steps/3-cloud-storage/app/views/books/_form.html.erb)

)RM/GETTING-STARTED-RUBY/BLOB/STEPS/3-CLOUD-STORAGE/APP/VIEWS/BOOKS/_FORM.HTML.ERB)

```
<div class="form-group">
  <%= f.label :cover_image %>
  <%= f.file_field :cover_image %>
</div>
```

Connect to Cloud Storage

The Bookshelf app uses the **google-cloud-storage** (https://googleapis.dev/ruby/google-cloud-storage/latest/Google/Cloud/Storage.html) gem to access Google Cloud services. It creates a connection toCloud Storage by using the local credentials you acquired for your workstation, or ambient credentials when running on Google Cloud VMs.

The `storage_bucket` class method of the `Book` class returns a reference to the storage bucket specified by the settings in the `settings.yml` file. This bucket is used to store cover images.

3-cloud-storage/app/models/book.rb (https://github.com/GoogleCloudPlatform/getting-started-ruby/blob/steps/3-cloud-storage/app/models/book.rb)

:CLOUDPLATFORM/GETTING-STARTED-RUBY/BLOB/STEPS/3-CLOUD-STORAGE/APP/MODELS/BOOK.RB)

```
require "google/cloud/storage"

class Book < ActiveRecord::Base

  def self.storage_bucket
    @storage_bucket ||= begin
      config = Rails.application.config.x.settings
      storage = Google::Cloud::Storage.new project_id: config["project_id"],
                                           credentials: config["keyfile"]
      storage.bucket config["gcs_bucket"]
    end
  end
```

Upload to Cloud Storage

The `Book` class has a `upload_image` method that is called each time a book is created. In `upload_image`, the call to `image.save` creates a new publicly readable file in the Cloud Storage bucket by using the content of `cover_image`. After the image is saved, the book's `image_url` is updated to the public URL of the saved image.

3-cloud-storage/app/models/book.rb (https://github.com/GoogleCloudPlatform/getting-started-ruby/blob/steps/3-cloud-storage/app/models/book.rb)

:CLOUDPLATFORM/GETTING-STARTED-RUBY/BLOB/STEPS/3-CLOUD-STORAGE/APP/MODELS/BOOK.RB)

```
after_create :upload_image, if: :cover_image

def upload_image
  file = Book.storage_bucket.create_file \
    cover_image.tempfile,
    "cover_images/#{id}/#{cover_image.original_filename}",
    content_type: cover_image.content_type,
    acl: "public"

  update_columns image_url: file.public_url
end
```

Serve images from Cloud Storage

Serving directly from Cloud Storage is helpful because the requests leverage Google's global serving infrastructure and the app doesn't have to respond to requests for images, freeing up CPU cycles for other requests.

To make a cover image visible in the app, update the view for the book and the view for the index.

3-cloud-storage/app/views/books/show.html.erb (https://github.com/GoogleCloudPlatform/getting-started-ruby/blob/steps/3-cloud-storage/app/views/books/show.html.erb)

ORM/GETTING-STARTED-RUBY/BLOB/STEPS/3-CLOUD-STORAGE/APP/VIEWS/BOOKS/SHOW.HTML.ERB)

```
<div class="media">
  <div class="media-left">
    <img src="<%= @book.image_url %>">
  </div>
  <div class="media-body">
    <h4><%= @book.title %> | &nbsp; <small><%= @book.published_on %></small></h4>
    <h5>By <%= @book.author || "unknown" %></h5>
    <p><%= @book.description %></p>
  </div>
</div>
```

Delete images

When you delete a book, the book model class checks to see whether the book has a cover image saved in the Cloud Storage bucket. If there is a cover image, it is deleted.

3-cloud-storage/app/models/book.rb
(https://github.com/GoogleCloudPlatform/getting-started-ruby/blob/steps/3-cloud-storage/app/models/book.rb)

ECLOUDPLATFORM/GETTING-STARTED-RUBY/BLOB/STEPS/3-CLOUD-STORAGE/APP/MODELS/BOOK.RB)

```
before_destroy :delete_image, if: :image_url

def delete_image
  image_uri = URI.parse image_url

  if image_uri.host == "#{Book.storage_bucket.name}.storage.googleapis.com"
    # Remove leading forward slash from image path
    # The result will be the image key, eg. "cover_images/:id/:filename"
    image_path = image_uri.path.sub("/", "")

    file = Book.storage_bucket.file image_path
    file.delete
  end
end
```

Update images

When you update a book, you can provide a new cover image. If there is an existing cover image, it is deleted. The new image is saved in the Cloud Storage bucket.

3-cloud-storage/app/models/book.rb (https://github.com/GoogleCloudPlatform/getting-started-ruby/blob/steps/3-cloud-storage/app/models/book.rb)

:CLOUDPLATFORM/GETTING-STARTED-RUBY/BLOB/STEPS/3-CLOUD-STORAGE/APP/MODELS/BOOK.RB)

```
before_update :update_image, if: :cover_image

def update_image
  delete_image if image_url?
  upload_image
end
```

< PREV (HTTPS://CLOUD.GOOGLE.COM/RUBY/GETTING-STARTED/USING-STRUCTURED-DATA)

NEXT > (HTTPS://CLOUD.GOOGLE.COM/RUBY/GETTING-STARTED/AUTHENTICATE-USERS)

Last updated November 19, 2019.