

[Ruby_](https://cloud.google.com/ruby/) (<https://cloud.google.com/ruby/>) [Guides](#)

Using Cloud SQL with Ruby

This part of the Bookshelf tutorial shows how the sample app stores its persistent data in Cloud SQL.

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) (<https://cloud.google.com/ruby/getting-started/tutorial-app>).

Creating a Cloud SQL instance

When deployed, your app uses the Cloud SQL Proxy that is built in to the App Engine environment to communicate with your Cloud SQL instance. However, to test your app locally, you must install and use a local copy of the proxy in your development environment.

Learn more about the [Cloud SQL Proxy](https://cloud.google.com/sql/docs/mysql/sql-proxy) (<https://cloud.google.com/sql/docs/mysql/sql-proxy>).

To perform basic admin tasks on your Cloud SQL instance, you can use the MySQL client.

Note: You must [authenticate gcloud](https://cloud.google.com/sql/docs/mysql/sql-proxy#gcloud) (<https://cloud.google.com/sql/docs/mysql/sql-proxy#gcloud>) before you can use the Cloud SQL Proxy to connect from your local machine.

Enable the Cloud SQL Admin API

Before using Cloud SQL, you must enable the Cloud SQL Admin API:

```
gcloud services enable sqladmin
```



Installing the Cloud SQL Proxy

Download and install the Cloud SQL Proxy. The Cloud SQL Proxy connects to your Cloud SQL instance when running locally.

LINUX 64-BIT LINUX 32-BIT MORE ▾

1. Download the proxy:

```
wget https://dl.google.com/cloudsql/cloud_sql_proxy.linux.amd64 -O cloud_sql_proxy
```
2. Make the proxy executable:

```
chmod +x cloud_sql_proxy
```

If your operating system isn't included here, you can also [compile the proxy from source](http://github.com/GoogleCloudPlatform/cloudsql-proxy) (<http://github.com/GoogleCloudPlatform/cloudsql-proxy>).

Creating a Cloud SQL instance

1. Create a Cloud SQL for MySQL Second Generation instance.

(<https://cloud.google.com/sql/docs/mysql/create-instance>)

Name the instance `[YOUR_INSTANCE_NAME]` or similar. It can take a few minutes for the instance to be ready. When the instance is ready, it's visible in the instances list.

Make sure that you create a **Second Generation** instance.

2. Use the Cloud SDK to run the following command where `[YOUR_INSTANCE_NAME]` represents the name of your Cloud SQL instance:

```
gcloud sql instances describe [YOUR_INSTANCE_NAME]
```

In the output, note the value shown for `[CONNECTION_NAME]`.

The `[CONNECTION_NAME]` value is in the format `[PROJECT_NAME] : [REGION_NAME] : [INSTANCE_NAME]`.

Initializing your Cloud SQL instance

1. Before using `./cloud_sql_proxy` for the first time, create a directory for the proxy sockets:

```
sudo mkdir /cloudsql
sudo chmod 0777 /cloudsql
```

2. Start the Cloud SQL Proxy by using the `[CONNECTION_NAME]` from the previous step.

```
./cloud_sql_proxy -instances="[YOUR_INSTANCE_CONNECTION_NAME]" -dir=/cloudsql
```

Replace `[YOUR_INSTANCE_CONNECTION_NAME]` with the instance connection name of your Cloud SQL instance. This step establishes a connection from your local computer to your Cloud SQL instance for local testing purposes. Keep the Cloud SQL Proxy running the entire time you test your app locally.

3. Create a new Cloud SQL user with an associated database.

CLOUD CONSOLE

MYSQL CLIENT

- a. Create a [new database using the Cloud Console](https://cloud.google.com/sql/docs/mysql/create-manage-databases#create) (<https://cloud.google.com/sql/docs/mysql/create-manage-databases#create>) for your Cloud SQL instance `[YOUR_INSTANCE_NAME]`. For example, you can use the name `[MYSQL_DATABASE]`.
- b. Create a [new user using the Cloud Console](https://cloud.google.com/sql/docs/mysql/create-manage-users#creating) (<https://cloud.google.com/sql/docs/mysql/create-manage-users#creating>) for your Cloud SQL instance `[YOUR_INSTANCE_NAME]`.

Configuring settings

1. Go to the `getting-started-ruby/2-cloud-sql` directory, and copy the `database.example.yml` file.

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

2. To configure your database, edit the `config/database.yml` file.

[2-cloud-sql/config/database.example.yml](https://github.com/GoogleCloudPlatform/getting-started-ruby/blob/2-cloud-sql/config/database.example.yml)

(<https://github.com/GoogleCloudPlatform/getting-started-ruby/blob/2-cloud-sql/config/database.example.yml>)

JDPLATFORM/GETTING-STARTED-RUBY/BLOB/STEPS/2-CLOUD-SQL/CONFIG/DATABASE.EXAMPLE.YML)

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

3. To prepare App Engine with Cloud SQL, edit `app.yaml`.

[2-cloud-sql/app.yaml](https://github.com/GoogleCloudPlatform/getting-started-ruby/blob/2-cloud-sql/app.yaml)

(<https://github.com/GoogleCloudPlatform/getting-started-ruby/blob/2-cloud-sql/app.yaml>)

^HUB.COM/GOOGLECLOUDPLATFORM/GETTING-STARTED-RUBY/BLOB/STEPS/2-CLOUD-SQL/APP.YAML)

```
beta_settings:
  # The connection name of your instance on its Overview page in the Google
  # Cloud Platform Console, or use `[YOUR_PROJECT_ID]:[YOUR_REGION]:[YOUR_INSTA
  cloud_sql_instances: [YOUR_INSTANCE_CONNECTION_NAME]
```

Installing dependencies

In the `2-cloud-sql` directory, enter the following command:

```
bundle install
```



Creating a database and tables

1. Create the database and run migrations to create the required table.

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



A message indicating that the `CreateBooks` migration succeeded is displayed:

```
== 20150706182833 CreateBooks: migrating =====  
-- create_table(:books)  
-> 0.4526s  
== 20150706182833 CreateBooks: migrated (0.4528s) =====
```



The `CreateBooks` migration creates a new `books` table with columns to store the book title, author, publication date, and description:

```
class CreateBooks < ActiveRecord::Migration  
  def change  
    create_table :books do |t|  
      t.string :title, required: true  
      t.string :author  
      t.date :published_on  
      t.text :description  
      t.timestamps null: false  
    end  
  end  
end
```



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 to add, edit, 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 \(HTTPS://CONSOLE.CLOUD.GOOGLE.COM/APPENGINE/VERSIONS\)](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 \(https://console.cloud.google.com/appengine/versions\)](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.

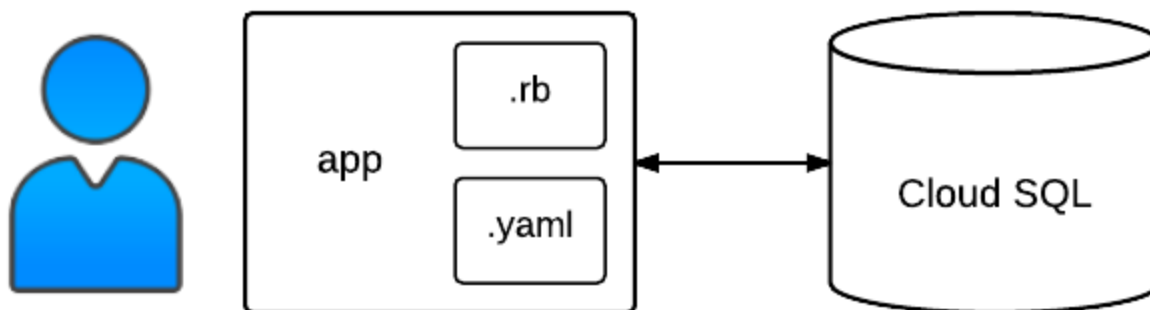
In the App Engine standard environment, you can stop the default version only if your app has manual or basic scaling.

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) (<https://cloud.google.com/ruby/getting-started/using-pub-sub#clean-up>) section in the final step of this tutorial.

App structure

This diagram shows the app's components and how they fit together.



Understanding the code

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

List books

When you visit the app's home page, you are routed to the `index` action of the `BooksController` class. This is configured in the `config/routes.rb` file.

[2-cloud-sql/config/routes.rb](https://github.com/GoogleCloudPlatform/getting-started-ruby/blob/2-cloud-sql/config/routes.rb)

(<https://github.com/GoogleCloudPlatform/getting-started-ruby/blob/2-cloud-sql/config/routes.rb>)

/GOOGLECLOUDPLATFORM/GETTING-STARTED-RUBY/BLOB/STEPS/2-CLOUD-SQL/CONFIG/ROUTES.RB)

```
Rails.application.routes.draw do
```



```
# Route root of application to BooksController#index action
root "books#index"

# Restful routes for BooksController
resources :books

end
```

The `BooksController#index` action retrieves a list of books from the Cloud SQL database. The app lists at most 10 books on each web page, so the list depends on which page the user is viewing. For example, suppose there are 26 books in the database, and the user is on the third page (`?page=3`). In that case, `params[:page]` is equal to 3, which is assigned to the `page_number` variable. Then a list of 6 books, starting at offset 20, is retrieved and assigned to `@books`.

[2-cloud-sql/app/controllers/books_controller.rb](https://github.com/GoogleCloudPlatform/getting-started-ruby/blob/2-cloud-sql/app/controllers/books_controller.rb)

(https://github.com/GoogleCloudPlatform/getting-started-ruby/blob/2-cloud-sql/app/controllers/books_controller.rb)

M/GETTING-STARTED-RUBY/BLOB/STEPS/2-CLOUD-SQL/APP/CONTROLLERS/BOOKS_CONTROLLER.RB)

```
class BooksController < ApplicationController

  PER_PAGE = 10

  def index
    page_number = params[:page] ? params[:page].to_i : 1
    book_offset = PER_PAGE * (page_number - 1)
    @books      = Book.limit(PER_PAGE).offset(book_offset)
    @next_page  = page_number + 1 if @books.count == PER_PAGE
  end
end
```

The `Book` class is a simple Active Record (http://guides.rubyonrails.org/active_record_basics.html) model that represents an individual book in the books table.

[2-cloud-sql/app/models/book.rb](https://github.com/GoogleCloudPlatform/getting-started-ruby/blob/2-cloud-sql/app/models/book.rb)

(<https://github.com/GoogleCloudPlatform/getting-started-ruby/blob/2-cloud-sql/app/models/book.rb>)

OGGLECLOUDPLATFORM/GETTING-STARTED-RUBY/BLOB/STEPS/2-CLOUD-SQL/APP/MODELS/BOOK.RB)

```
class Book < ActiveRecord::Base
  validates :title, presence: true
end
```


end

In the `routes.rb` file, the `resources :books` call configures RESTful routes for creating, reading, updating, and deleting books that are routed to the corresponding actions in the `BooksController` class.

After `BooksController.index` retrieves a list of books, the embedded Ruby code in the `books/index.html.erb` file renders the list.

[2-cloud-sql/app/views/books/index.html.erb](https://github.com/GoogleCloudPlatform/getting-started-ruby/blob/2-cloud-sql/app/views/books/index.html.erb)

(<https://github.com/GoogleCloudPlatform/getting-started-ruby/blob/2-cloud-sql/app/views/books/index.html.erb>)

PLATFORM/GETTING-STARTED-RUBY/BLOB/STEPS/2-CLOUD-SQL/APP/VIEWS/BOOKS/INDEX.HTML.ERB)

```
<% @books.each do |book| %>
  <div class="media">
    <%= link_to book_path(book) do %>
      <div class="media-body">
        <h4><%= book.title %></h4>
        <p><%= book.author %></p>
      </div>
    <% end %>
  </div>
<% end %>

<% if @next_page %>
  <nav>
    <ul class="pager">
      <li><%= link_to "More", books_path(page: @next_page) %></li>
    </ul>
  </nav>
<% end %>
```

Display book details

When you click an individual book on the web page, the `BookController#show` action retrieves the book, specified by its ID, from the table.

[2-cloud-sql/app/controllers/books_controller.rb](https://github.com/GoogleCloudPlatform/getting-started-ruby/blob/2-cloud-sql/app/controllers/books_controller.rb)

(https://github.com/GoogleCloudPlatform/getting-started-ruby/blob/2-cloud-sql/app/controllers/books_controller.rb)

M/GETTING-STARTED-RUBY/BLOB/STEPS/2-CLOUD-SQL/APP/CONTROLLERS/BOOKS_CONTROLLER.RB)

```
def show
  @book = Book.find params[:id]
end
```



Then the embedded Ruby code in the `show.html.erb` file displays the book's details.

[2-cloud-sql/app/views/books/show.html.erb](#)

(<https://github.com/GoogleCloudPlatform/getting-started-ruby/blob/steps/2-cloud-sql/app/views/books/show.html.erb>)

PLATFORM/GETTING-STARTED-RUBY/BLOB/STEPS/2-CLOUD-SQL/APP/VIEWS/BOOKS/SHOW.HTML.ERB)

```
<div class="media">
  <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>
```



Create books

When you click **Add book** on the web page, the `BooksController#new` action creates a new book. The embedded Ruby code in the `new.html.erb` file points to `_form.html.erb`, which displays the form for adding a new book.

[2-cloud-sql/app/views/books/_form.html.erb](#)

(https://github.com/GoogleCloudPlatform/getting-started-ruby/blob/steps/2-cloud-sql/app/views/books/_form.html.erb)

LATFORM/GETTING-STARTED-RUBY/BLOB/STEPS/2-CLOUD-SQL/APP/VIEWS/BOOKS/_FORM.HTML.ERB)

```
<%= form_for @book do |f| %>
  <div class="form-group">
    <%= f.label :title %>
    <%= f.text_field :title %>
  </div>
  <div class="form-group">
    <%= f.label :author %>
    <%= f.text_field :author %>
  </div>
</form_for %>
```



```
</div>
<div class="form-group">
  <%= f.label :published_on, "Date Published" %>
  <%= f.date_field :published_on %>
</div>
<div class="form-group">
  <%= f.label :description %>
  <%= f.text_area :description %>
</div>
<button class="btn btn-success" type="submit">Save</button>
<% end %>
```

When you submit the form, the `BooksController#create` action saves the book in the database. If the new book is saved successfully, the book's page is displayed. Otherwise, the form is displayed again along with error messages. The `book_params` method uses [strong parameters](http://guides.rubyonrails.org/action_controller_overview.html#strong-parameters) (http://guides.rubyonrails.org/action_controller_overview.html#strong-parameters) to specify which form fields are allowed. In this case, only book title, author, publication date, and description are allowed.

[2-cloud-sql/app/controllers/books_controller.rb](https://github.com/GoogleCloudPlatform/getting-started-ruby/blob/2-cloud-sql/app/controllers/books_controller.rb)

(https://github.com/GoogleCloudPlatform/getting-started-ruby/blob/2-cloud-sql/app/controllers/books_controller.rb)

M/GETTING-STARTED-RUBY/BLOB/STEPS/2-CLOUD-SQL/APP/CONTROLLERS/BOOKS_CONTROLLER.RB)

```
def create
  @book = Book.new book_params

  if @book.save
    flash[:success] = "Added Book"
    redirect_to book_path(@book)
  else
    render :new
  end
end

private

def book_params
  params.require(:book).permit(:title, :author, :published_on, :description)
end
```

Edit books

When you click **Edit book** on the web page, the `BooksController#update` action retrieves the book from the database. The embedded Ruby code in the `edit.html.erb` file points to `_form.html.erb`, which displays the form for editing the book.

[2-cloud-sql/app/controllers/books_controller.rb](https://github.com/GoogleCloudPlatform/getting-started-ruby/blob/2-cloud-sql/app/controllers/books_controller.rb)

(https://github.com/GoogleCloudPlatform/getting-started-ruby/blob/2-cloud-sql/app/controllers/books_controller.rb)

M/GETTING-STARTED-RUBY/BLOB/STEPS/2-CLOUD-SQL/APP/CONTROLLERS/BOOKS_CONTROLLER.RB)

```
def update
  @book = Book.find params[:id]

  if @book.update book_params
    flash[:success] = "Updated Book"
    redirect_to book_path(@book)
  else
    render :edit
  end
end
```



When you submit the form, the `BooksController#update` action saves the book in the database. If the new book is saved successfully, the book's page is displayed. Otherwise, the form is displayed along with error messages.

Delete books

When you click **Delete Book** on the web page, the `BooksController#destroy` action deletes the book from the database and then displays the updated list of books.

[2-cloud-sql/app/controllers/books_controller.rb](https://github.com/GoogleCloudPlatform/getting-started-ruby/blob/2-cloud-sql/app/controllers/books_controller.rb)

(https://github.com/GoogleCloudPlatform/getting-started-ruby/blob/2-cloud-sql/app/controllers/books_controller.rb)

M/GETTING-STARTED-RUBY/BLOB/STEPS/2-CLOUD-SQL/APP/CONTROLLERS/BOOKS_CONTROLLER.RB)

```
def destroy
  @book = Book.find params[:id]
  @book.destroy
end
```



```
  redirect_to books_path  
end
```

[< PREVIOUS](https://cloud.google.com/ruby/getting-started/using-structured-data) (HTTPS://CLOUD.GOOGLE.COM/RUBY/GETTING-STARTED/USING-STRUCTURED-DATA)

[NEXT >](https://cloud.google.com/ruby/getting-started/using-cloud-storage) (HTTPS://CLOUD.GOOGLE.COM/RUBY/GETTING-STARTED/USING-CLOUD-STORAGE)

Except as otherwise noted, the content of this page is licensed under the [Creative Commons Attribution 4.0 License](https://creativecommons.org/licenses/by/4.0/) (https://creativecommons.org/licenses/by/4.0/), and code samples are licensed under the [Apache 2.0 License](https://www.apache.org/licenses/LICENSE-2.0) (https://www.apache.org/licenses/LICENSE-2.0). For details, see our [Site Policies](https://developers.google.com/terms/site-policies) (https://developers.google.com/terms/site-policies). Java is a registered trademark of Oracle and/or its affiliates.

Last updated December 4, 2019.