

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

Handling sessions with Firestore

Many apps need session handling for authentication and user preferences. [Sinatra](http://sinatrarb.com/) (<http://sinatrarb.com/>) comes with a memory-based implementation to perform this function. However, this implementation is unsuitable for an app that can be served from multiple instances, because the session that is recorded on one instance might differ from other instances. This tutorial shows how to handle sessions on [App Engine](https://cloud.google.com/appengine/docs/standard/) (<https://cloud.google.com/appengine/docs/standard/>).

Objectives

- Write the app.
- Run the app locally.
- Deploy the app on App Engine.

Costs

This tutorial uses the following billable components of Google Cloud:

- [App Engine](https://cloud.google.com/appengine/pricing) (<https://cloud.google.com/appengine/pricing>)
- [Firestore](https://cloud.google.com/firestore/pricing) (<https://cloud.google.com/firestore/pricing>)

To generate a cost estimate based on your projected usage, use the [pricing calculator](https://cloud.google.com/products/calculator) (<https://cloud.google.com/products/calculator>). New Google Cloud users might be eligible for a [free trial](https://cloud.google.com/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) (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) (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/projectselector) (HTTPS://CONSOLE.CLOUD.GOOGLE.COM/PROJECTSELECTOR)

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) (https://cloud.google.com/billing/docs/how-to/modify-project).
4. Enable the Firestore API.

[ENABLE THE API](https://console.cloud.google.com/flows/enableapi?apiid=firestore) (HTTPS://CONSOLE.CLOUD.GOOGLE.COM/FLOWS/ENABLEAPI?APIID=FIREFSTORE)

5. [Prepare your development environment](https://cloud.google.com/ruby/docs/setup) (https://cloud.google.com/ruby/docs/setup).

Setting up the project

1. In your terminal window, start in a directory of your choosing and create a new directory named `sessions`. All of the code for this tutorial is inside the `sessions` directory.
2. Change into the `sessions` directory:

```
cd sessions
```

3. Initialize the `Gemfile`:

```
bundle init
```

4. Append the following to the resulting Gemfile:

[sessions/Gemfile](#)

(<https://github.com/GoogleCloudPlatform/getting-started-ruby/blob/master/sessions/Gemfile>)

GITHUB.COM/GOOGLECLOUDPLATFORM/GETTING-STARTED-RUBY/BLOB/MASTER/SESSIONS/GEMFILE)

```
gem "google-cloud-firestore", "~> 1.2"  
gem "sinatra", "~> 2.0"
```

The Gemfile lists any non-standard Ruby libraries your app needs App Engine to load for it:

- `google-cloud-firestore` is the Ruby client for the Firestore API.
- Sinatra is the Ruby web framework used for the app.

5. Install the dependencies:

```
bundle install
```

Writing the web app

This app displays greetings in different languages for every user. Returning users are always greeted in the same language.

- With a text editor, create a file called `app.rb` in the `sessions` directory with the following content:

[sessions/app.rb](#)

(<https://github.com/GoogleCloudPlatform/getting-started-ruby/blob/master/sessions/app.rb>)

//GITHUB.COM/GOOGLECLOUDPLATFORM/GETTING-STARTED-RUBY/BLOB/MASTER/SESSIONS/APP.RB)

```
require "sinatra"  
  
require_relative "firestore_session"  
  
use Rack::Session::FirestoreSession  
  
set :greetings, ["Hello World", "Hallo Welt", "Ciao Mondo", "Salut le Monde", "  
  
get "/" do
```

```

session[:greeting] ||= settings.greetings.sample
session[:views] ||= 0
session[:views] += 1
"<h1>#{session[:views]} views for \"#{session[:greeting]}\"</h1>"
end

```

Creating the session store

Before your app can store preferences for a user, you need a way to store information about the current user in a session. Sinatra has built-in support for saving session data to a cookie. To save to Firestore instead, you have to define your own `Rack::Session` object.

- In the `sessions` directory, create a file called `firestore_session.rb` with the following content:

```

sessions/firestore_session.rb
(https://github.com/GoogleCloudPlatform/getting-started-
ruby/blob/master/sessions/firestore_session.rb)

```

GOOGLECLOUDPLATFORM/GETTING-STARTED-RUBY/BLOB/MASTER/SESSIONS/FIRESTORE_SESSION.RB)

```

require "google/cloud/firestore"
require "rack/session/abstract/id"

module Rack
  module Session
    class FirestoreSession < Abstract::Persisted
      def initialize app, options = {}
        super

        @firestore = Google::Cloud::Firestore.new
        @col = @firestore.col "sessions"
      end

      def find_session _req, session_id
        return [generate_sid, {}] if session_id.nil?

        doc = @col.doc session_id
        fields = doc.get.fields || {}
        [session_id, stringify_keys(fields)]
      end
    end
  end
end

```

```
def write_session _req, session_id, new_session, _opts
  doc = @col.doc session_id
  doc.set new_session, merge: true
  session_id
end

def delete_session _req, session_id, _opts
  doc = @col.doc session_id
  doc.delete
  generate_sid
end

def stringify_keys hash
  new_hash = {}
  hash.each do |k, v|
    new_hash[k.to_s] =
      if v.is_a? Hash
        stringify_keys v
      else
        v
      end
  end
  new_hash
end
end
end
end
```

Note: Sinatra supports various options for session storage. Firestore is a persistent, distributed, transactional database. Often, it's more appropriate to choose a different storage solution for sessions such as [Memcache](https://redislabs.com/lp/memcached-cloud/) or [Redis](https://redislabs.com/), whose designs might result in faster operation for this use case.

Deleting sessions

As written, this app doesn't delete old or expired sessions. You can [delete session data](https://cloud.google.com/firestore/docs/using-console#delete_data) in the [Google Cloud Console](https://console.cloud.google.com/firestore/), or you could implement an automated deletion strategy.

Running locally

1. Start the HTTP server:

```
bundle exec ruby app.rb
```

2. View the app in your web browser (<http://localhost:4567>).

You see one of five greetings: "Hello World", "Hallo Welt", "Hola mundo", "Salut le Monde", or "Ciao Mondo." The language changes if you open the page in a different browser or in incognito mode. You can see and edit the session data in the Google Cloud Console (<https://console.cloud.google.com/firestore/>).

3. To stop the HTTP server, in your terminal window, press **Control+C**.

Deploying and running on App Engine

You can use the App Engine standard environment

(<https://cloud.google.com/appengine/docs/standard/>) to build and deploy an app that runs reliably under heavy load and with large amounts of data.

This tutorial uses the App Engine standard environment to deploy the server.

1. In your terminal window, create an `app.yaml` file and paste the following into the file:

```
sessions/app.yaml
```

```
(https://github.com/GoogleCloudPlatform/getting-started-ruby/blob/master/sessions/app.yaml)
```

```
ITHUB.COM/GOOGLECLOUDPLATFORM/GETTING-STARTED-RUBY/BLOB/MASTER/SESSIONS/APP.YAML)
```

```
runtime: ruby25
entrypoint: bundle exec ruby app.rb
```

2. Deploy the app on App Engine:

```
gcloud app deploy
```

3. View the live app at the following URL, where **PROJECT_ID** is your Google Cloud project ID:

```
https://PROJECT\_ID.appspot.com
```

The greeting is now delivered by a web server running on an App Engine instance.

Debugging the app

If you cannot connect to your App Engine app, check the following:

1. Check that the `gcloud` deploy commands successfully completed and didn't output any errors. If there were errors (for example, `message=Build failed`), fix them, and try [deploying the App Engine app](#) (`#deploying_the_web_app`) again.
2. In the Cloud Console, go to the **Logs Viewer** page.

[GO TO LOGS VIEWER PAGE \(HTTPS://CONSOLE.CLOUD.GOOGLE.COM/LOGS/VIEWER\)](https://console.cloud.google.com/logs/viewer)

- a. In the **Recently selected resources** drop-down list, click **App Engine Application**, and then click **All module_id**. You see a list of requests from when you visited your app. If you don't see a list of requests, confirm you selected **All module_id** from the drop-down list. If you see error messages printed to the Cloud Console, check that your app's code matches [the code in the section about writing the web app](#) (`#writing_the_web_app`).
- b. Make sure that the Firestore API is enabled.

Cleaning up

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

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.

Delete the App Engine instance


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.

What's next

- Try more [Cloud Functions tutorials](https://cloud.google.com/functions/docs/tutorials/) (https://cloud.google.com/functions/docs/tutorials/).
- [Learn more about App Engine](https://cloud.google.com/appengine/docs/) (https://cloud.google.com/appengine/docs/).
- [Try Cloud Run](https://cloud.google.com/run/docs/quickstarts/prebuilt-deploy) (https://cloud.google.com/run/docs/quickstarts/prebuilt-deploy), which lets you run stateless containers on a fully managed environment or in your own Google Kubernetes Engine cluster.

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 21, 2019.