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

Running Rails 5 on App Engine flexible environment

Get started developing Ruby on Rails apps that run on App Engine flexible environment (https://cloud.google.com/appengine/docs/flexible/). The apps you create run on the same infrastructure that powers all of Google's products, so you can be confident that they can scale to serve all of your users, whether there are a few or millions of them.

This tutorial assumes you are familiar with Rails web development. It walks you through deploying a new Rails app.

This tutorial requires Ruby 2.3.3 (https://www.ruby-lang.org/) or newer.

Before you begin

1. Sign in (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).

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/PROJECTSELECT

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).

4. Install and initialize the Cloud SDK (https://cloud.google.com/sdk/docs/).

Set up your local environment for Rails

Before you begin, take the following steps:

1. Install Ruby (https://www.ruby-lang.org/) version 2.3.3 or greater.
2. Install the Rails 5 (http://guides.rubyonrails.org/) gem.
3. Install the Bundler (https://bundler.io/) gem.

Alternatively, you can use Cloud Shell (https://console.cloud.google.com/?cloudshell=true), which comes with Ruby, Rails, and the Cloud SDK already installed.

For additional information on installing Rails and its dependencies, see the official Getting started with Rails (http://guides.rubyonrails.org/getting_started.html) guide.

After you complete the prerequisites, you can create and deploy a Rails app. The following sections guide you through configuring, running, and deploying an app.

Create a new app

1. Scaffold a new Rails sample app.

```
rails new appengine_example
```

2. Go to the directory that contains the sample code.

```
cd appengine_example
```

Run the app locally

To run the new Rails app on your local computer:

1. Install dependencies by using Bundler.

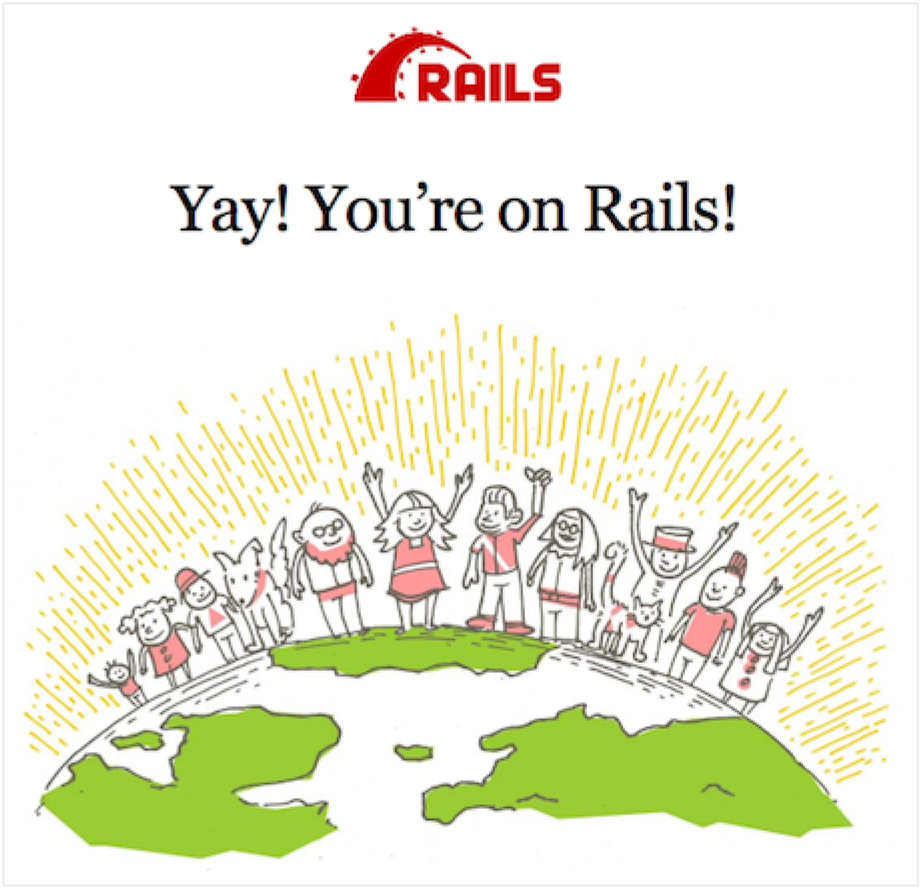
```
bundle install
```

2. Start a local web server.

```
bundle exec rails server
```

3. In your browser, go to http://localhost:3000/ (http://localhost:3000).

You see the "**Yay! You're on Rails!**" message from the sample app displayed on the page.



Note: In your terminal window, press `Ctrl+C` to exit the web server.

Add a simple page

Now add a Welcome page to the generated Rails app.

1. To generate the scaffolding for a new page, create a new Rails controller named `WelcomeController` with an `index` action.

```
bundle exec rails generate controller Welcome index
```

2. Open the file `app/views/welcome/index.html.erb` to see the following boilerplate HTML.

appengine/rails-hello_world/app/views/welcome/index.html.erb (https://github.com/GoogleCloudPlatform/ruby-docs-samples/blob/0a6e898a1acd140ca04c63c7b68da1468acb4082/appengine/rails-hello_world/app/views/welcome/index.html.erb)

53C7B68DA1468ACB4082/APPENGINE/RAILS-HELLO_WORLD/APP/VIEWS/WELCOME/INDEX.HTML.ERB)

```
<h1>Welcome#index</h1>
<p>Find me in app/views/welcome/index.html.erb</p>
```

3. Modify the file however you like. For example, you can use the following content:

appengine/rails-hello_world/app/views/welcome/index.html.erb (https://github.com/GoogleCloudPlatform/ruby-docs-samples/blob/0a6e898a1acd140ca04c63c7b68da1468acb4082/appengine/rails-hello_world/app/views/welcome/index.html.erb)

53C7B68DA1468ACB4082/APPENGINE/RAILS-HELLO_WORLD/APP/VIEWS/WELCOME/INDEX.HTML.ERB)

```
<h1>Welcome</h1>
<p>This is a home page for a new Rails App on Google Cloud Platform!</p>
```

4. Set the `index` controller action as the root action for Rails. Then, whenever a user visits the Rails app, they see your welcome page.
5. Open the file `config/routes.rb` to see the following generated content.

appengine/rails-hello_world/config/routes.rb (https://github.com/GoogleCloudPlatform/ruby-docs-samples/blob/0a6e898a1acd140ca04c63c7b68da1468acb4082/appengine/rails-hello_world/config/routes.rb)

398A1ACD140CA04C63C7B68DA1468ACB4082/APPENGINE/RAILS-HELLO_WORLD/CONFIG/ROUTES.RB)

```
Rails.application.routes.draw do
  get 'welcome/index'

  # For details on the DSL available within this file, see http://guides.rubyon
end
```

6. To modify this file, add `root 'welcome#index'`.

appengine/rails-hello_world/config/routes.rb (https://github.com/GoogleCloudPlatform/ruby-docs-samples/blob/0a6e898a1acd140ca04c63c7b68da1468acb4082/appengine/rails-hello_world/config/routes.rb)

398A1ACD140CA04C63C7B68DA1468ACB4082/APPENGINE/RAILS-HELLO_WORLD/CONFIG/ROUTES.RB)

```
Rails.application.routes.draw do
  get 'welcome/index'

  # For details on the DSL available within this file, see http://guides.rubyon
  root 'welcome#index'
end
```

7. Save the file and close it. Test the Rails app as you did before.

Deploy the app to App Engine flexible environment

App Engine flexible environment uses a file called **`app.yaml`** (https://cloud.google.com/appengine/docs/flexible/ruby/configuring-your-app-with-app-yaml). to describe an app's deployment configuration. If this file isn't present, the `gcloud` command line tool tries to guess the deployment configuration. However, it is a good idea to provide this file because Rails requires a secret key in a production environment.

To configure the sample app for deployment to App Engine, create a new file named `app.yaml` at the root of your sample app directory and add the following:

appengine/rails-hello_world/app.yaml (https://github.com/GoogleCloudPlatform/ruby-docs-samples/blob/0a6e898a1acd140ca04c63c7b68da1468acb4082/appengine/rails-hello_world/app.yaml)

.OB/0A6E898A1ACD140CA04C63C7B68DA1468ACB4082/APPENGINE/RAILS-HELLO_WORLD/APP.YAML)

```
entrypoint: bundle exec rackup --port $PORT
env: flex
runtime: ruby
```

Configure the Rails secret key

When you deploy a Rails app in the production environment, set the environment variable `SECRET_KEY_BASE` to a secret key that is used to protect user session data. This environment variable is read in the `config/secrets.yml` file.

1. Generate a new secret key.

```
bundle exec rails secret
```

2. Copy the generated secret key. You use the secret key in the next step.
3. Open the file `app.yaml` that you created earlier, and add an `env_variables` section. The `env_variables` sets environment variables in the `production` environment in App Engine flexible environment. Your `app.yaml` should look like the example below, with `[SECRET_KEY]` replaced with your copied secret key.

appengine/rails-hello_world/app.yaml (https://github.com/GoogleCloudPlatform/ruby-docs-samples/blob/0a6e898a1acd140ca04c63c7b68da1468acb4082/appengine/rails-hello_world/app.yaml)

.OB/0A6E898A1ACD140CA04C63C7B68DA1468ACB4082/APPENGINE/RAILS-HELLO_WORLD/APP.YAML)

```
entrypoint: bundle exec rackup --port $PORT
env: flex
runtime: ruby

env_variables:
  SECRET_KEY_BASE: [SECRET_KEY]
```

Set up an App Engine flexible environment app

If this is the first time you are deploying an app, you need to create an App Engine flexible environment app to help you select the region in which to run the Rails app. You can read more about regions and zones (https://cloud.google.com/docs/geography-and-regions).

1. Create an App Engine app.

```
gcloud app create
```

2. Select a region that supports App Engine flexible environment for Ruby apps.

Deploy to App Engine flexible environment

1. Before you deploy, precompile your Rails assets.

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

2. After the assets finish compiling, deploy the sample.

```
gcloud app deploy
```

Wait for the message that notifies you that the update has completed. This can take several minutes.

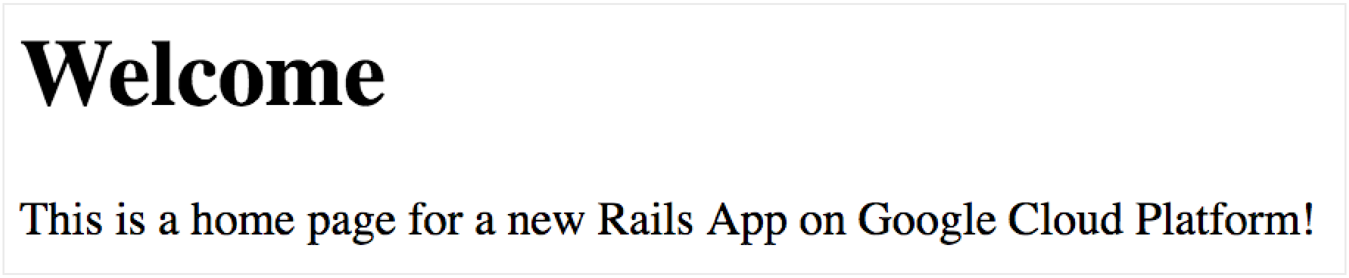
Access the deployed Rails app

1. To retrieve your project ID, run `gcloud info`.
2. In your browser, go to `https://[PROJECT_ID].appspot.com`.

   where:

   - `[PROJECT_ID]` is the project ID you retrieved in the first step.

The following content is displayed.



Welcome

This is a home page for a new Rails App on Google Cloud Platform!

This time, your request is served by the Rails app running in App Engine flexible environment.

This command deploys the app as described in `app.yaml` and sets the newly deployed version as the default version, causing it to serve all new traffic. As the app deploys, you might see several repeated messages while the platform checks whether the app is serving. This is normal. Wait for the message that notifies you that the update of the app is complete.

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.

Read App Engine logs

Now that you have deployed your Rails app, you may want to read the logs. You can read the app logs by using the Logs Viewer (https://console.cloud.google.com/logs/viewer) located in the Cloud Console, or by using `gcloud app logs read`. You can learn more about reading logs by using the Cloud SDK (https://cloud.google.com/sdk/gcloud/reference/app/logs/read).

Clean up resources

After you've finished the Running Rails 5 in the App Engine tutorial, you can clean up the resources that you created on GCP so they won't take up quota and you won't be billed for them in the future. The following sections describe how to delete or turn off these resources.

Delete project

The easiest way to eliminate billing is to delete the project that you created for the tutorial.

To 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/PRC

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 an App Engine version

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.

What's next

- Learn how to use Cloud SQL for MySQL with Rails (https://cloud.google.com/ruby/rails/using-cloudsql-mysql).
- Learn how to use Cloud SQL for PostgreSQL with Rails (https://cloud.google.com/ruby/rails/using-cloudsql-postgres).
- Learn how to run the Ruby Bookshelf sample in App Engine flexible environment (https://cloud.google.com/ruby/getting-started/tutorial-app).

- Learn how to run the Ruby Bookshelf sample on Compute Engine (https://cloud.google.com/ruby/tutorials/bookshelf-on-compute-engine).
- Learn how to run the Ruby Bookshelf sample on GKE (https://cloud.google.com/ruby/tutorials/bookshelf-on-kubernetes-engine).