13.4.4 Image upload in production

The image uploading developed in Section 13.4.3 is good enough for development, but (as seen below in Listing 13.73) it uses the local disk for storing the images, which isn't a good practice in production. (Among other things, file storage on Heroku is temporary, so uploaded images will be deleted every time you deploy.) Instead, we'll use a cloud storage service to store images separately from our application.

There are many choices for cloud storage, but we'll use one of the most popular and well-supported, Amazon.com's Simple Storage Service (S3), part of Amazon Web Services (AWS).[23]

To configure our application to use cloud storage in production, we'll add the `aws-sdk-s3` gem to the **`:production`** environment, as shown in Listing 13.72.

Listing 13.72: Adding a gem for Amazon Web Services (AWS).
`Gemfile`

```
source 'https://rubygems.org'
git_source(:github) { |repo| "https://github.com/#{repo}.git" }

gem 'rails',                      '6.0.1'
gem 'image_processing',           '1.9.3'
gem 'mini_magick',                '4.9.5'
gem 'active_storage_validations', '0.8.2'
.
.
.
group :production do
  gem 'pg',         '1.1.4'
  gem 'aws-sdk-s3', '1.46.0', require: false
end
.
.
.
```

Then **`bundle`** one more time:

[23]S3 is a paid service, but the storage needed to set up and test the Rails Tutorial sample application costs less than a cent per month.

```
$ bundle install
```

AWS configuration

At this point, you'll need to configure your AWS system to use S3. Here are the basic steps:[24]

1. Sign up for an Amazon Web Services account if you don't have one already (Figure 13.31). (If you signed up for the Cloud9 IDE in Section 1.1.1, you already have an AWS account and can skip this step.)

2. Create a user via AWS Identity and Access Management (IAM). This involves using the IAM users interface (Figure 13.32) to navigate to the "Add user" page (Figure 13.33), where you should create a user while enabling "programmatic access" (Figure 13.34), grant the user administrator access (Figure 13.35), and then skip the optional user tags (Figure 13.36).

3. After clicking "Create user", you should see the name of the user together with the access key ID and the secret access key (Figure 13.37). Copy these keys and store them some place safe.

4. Create an S3 bucket using the AWS Console (Figure 13.38). S3 buckets exist in a global namespace, so the name has to be unique, but otherwise the default bucket settings should be fine.

You may find setting up S3 to be a challenging exercise in technical sophistication (Box 1.2); for further details on the steps above, consult the S3 documentation, the article "Setting up Rails 5 [or higher] with Active Storage with Amazon S3", and, if necessary, Google or Stack Overflow.

[24]The steps are current as of this writing, but services like AWS are constantly evolving, so the user interface may have changed in the interim. Use your technical sophistication to resolve any discrepancies.

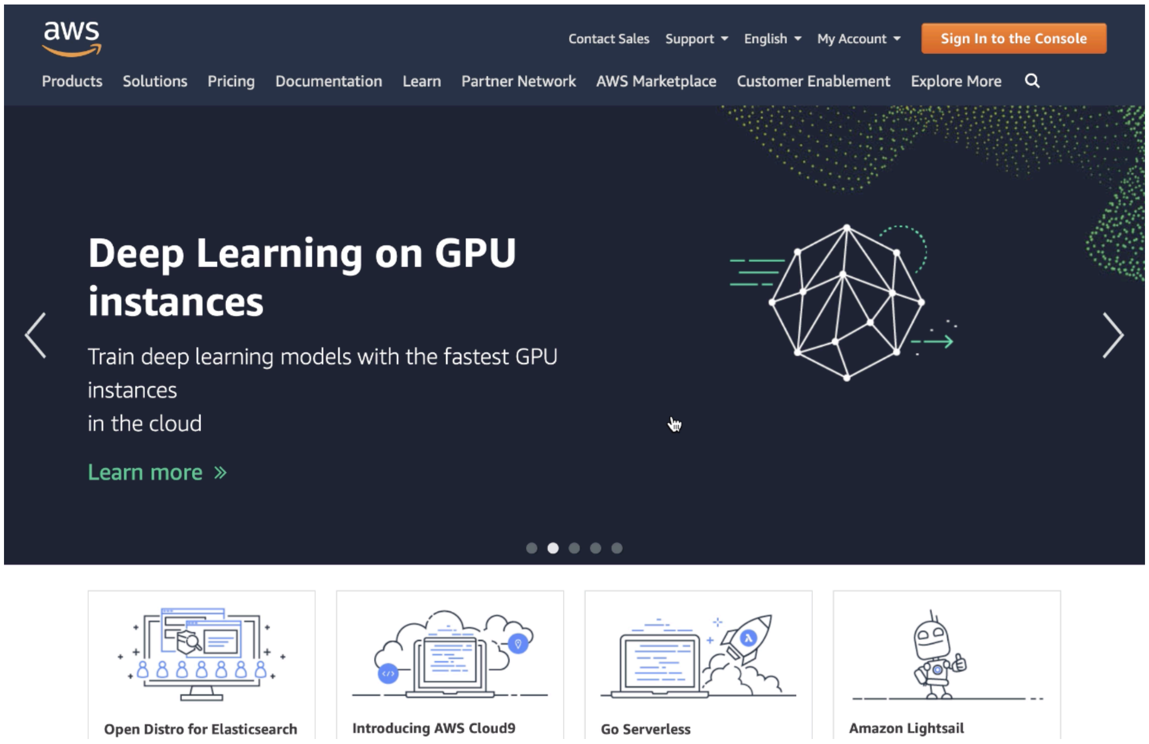


Figure 13.31: Signing up for AWS.

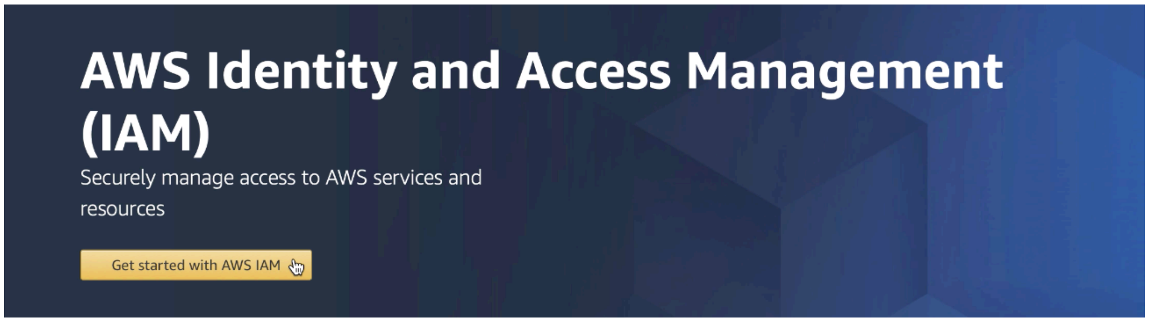

AWS Identity and Access Management (IAM)
Securely manage access to AWS services and resources
Get started with AWS IAM

Figure 13.32: The AWS IAM interface.

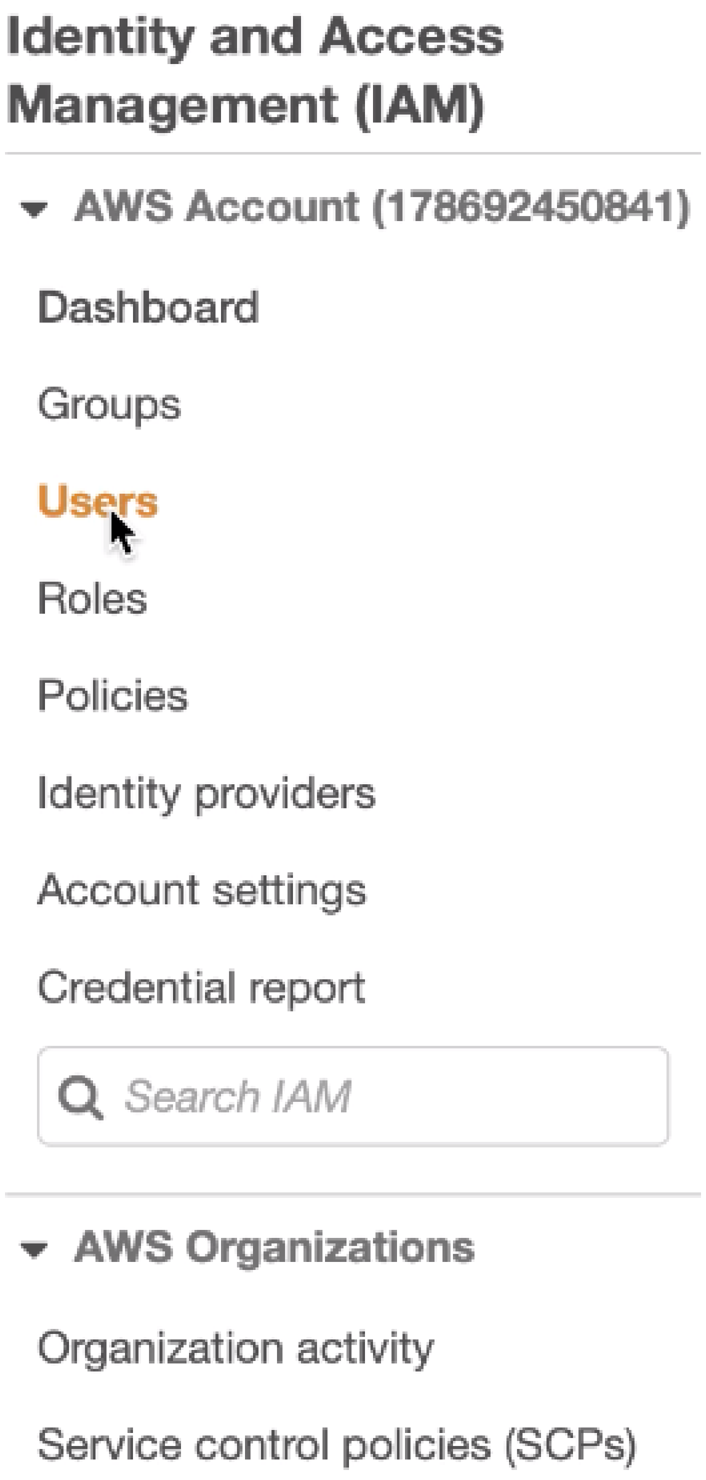
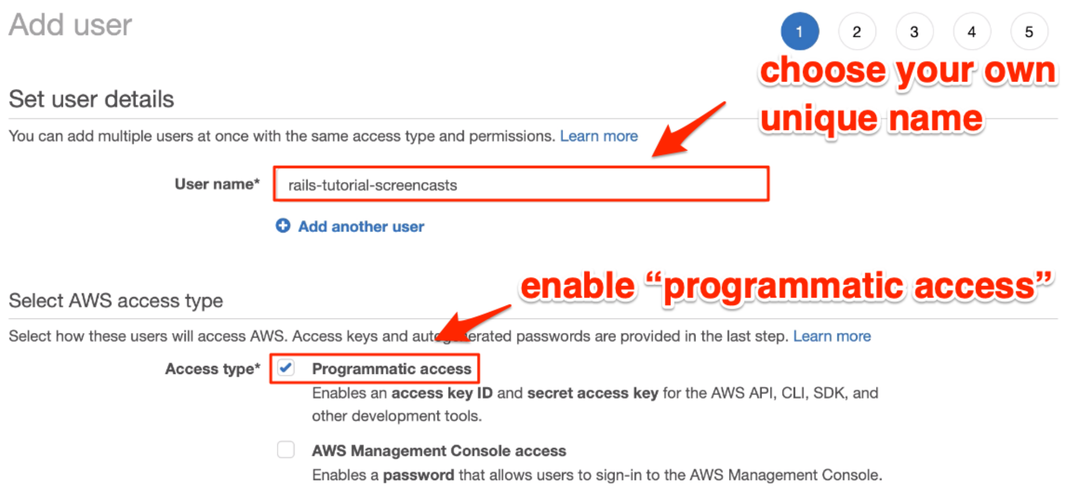


Figure 13.33: Navigating to the “Add user” page.



Add user

1 2 3 4 5

Set user details

You can add multiple users at once with the same access type and permissions. Learn more

User name* rails-tutorial-screencasts

Add another user

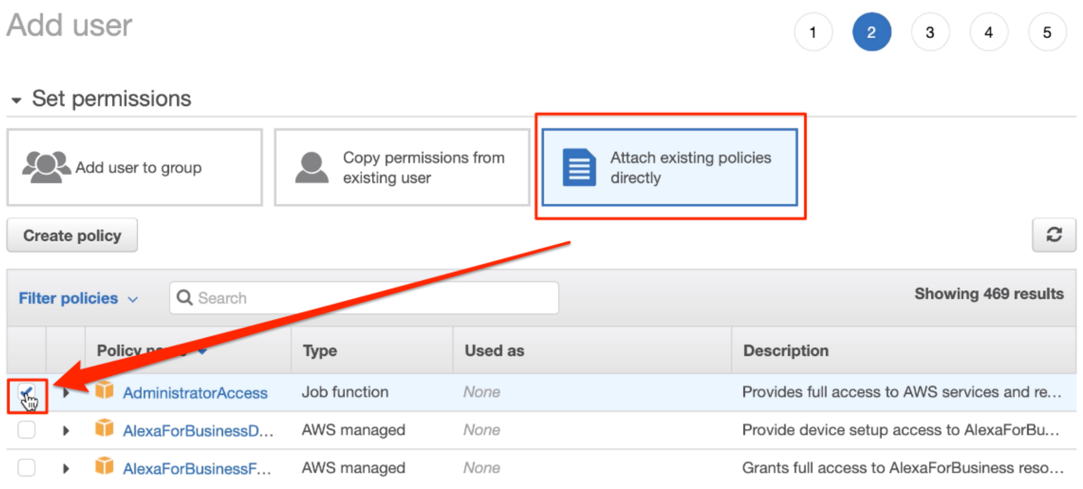
Select AWS access type

Select how these users will access AWS. Access keys and autogenerated passwords are provided in the last step. Learn more

Access type* ✓ Programmatic access
Enables an **access key ID** and **secret access key** for the AWS API, CLI, SDK, and other development tools.

AWS Management Console access
Enables a **password** that allows users to sign-in to the AWS Management Console.

Figure 13.34: Creating a user with "programmatic access".



Add user

1 2 3 4 5

Set permissions

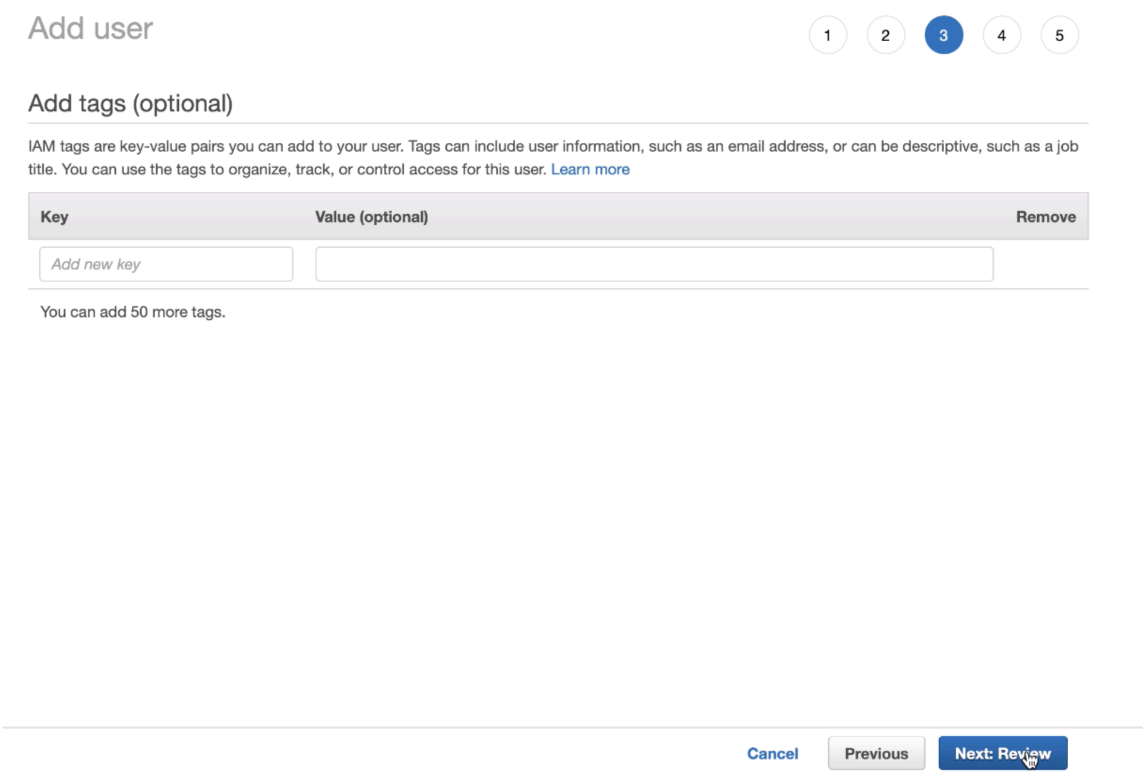
Add user to group | Copy permissions from existing user | Attach existing policies directly

Create policy

Filter policies Search Showing 469 results

		Policy name	Type	Used as	Description
✓	▸	AdministratorAccess	Job function	None	Provides full access to AWS services and re…
	▸	AlexaForBusinessD…	AWS managed	None	Provide device setup access to AlexaForBu…
	▸	AlexaForBusinessF…	AWS managed	None	Grants full access to AlexaForBusiness reso…

Figure 13.35: Granting the user administrator access.

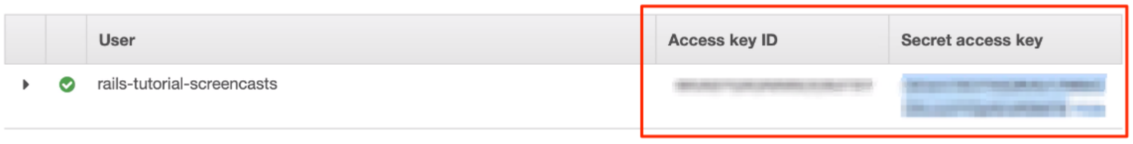


IAM tags are key-value pairs you can add to your user. Tags can include user information, such as an email address, or can be descriptive, such as a job title. You can use the tags to organize, track, or control access for this user. Learn more

Key	Value (optional)	Remove
Add new key		

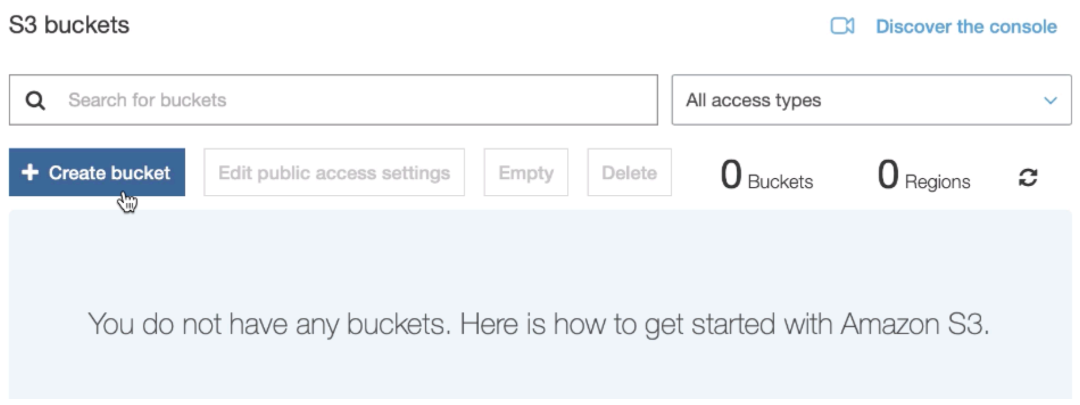
You can add 50 more tags.

Figure 13.36: Skipping optional user tags.



		User	Access key ID	Secret access key
▸	✓	rails-tutorial-screencasts	[illegible]	[illegible]

Figure 13.37: Displaying the access key ID and the secret access key.



You do not have any buckets. Here is how to get started with Amazon S3.

Figure 13.38: Creating an AWS bucket.

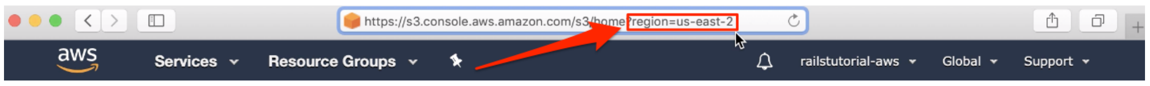


Figure 13.39: Getting the AWS region from the S3 console URL.

Production AWS

As with production email configuration (Listing 11.41), we'll be using Heroku **ENV** variables to avoid hard-coding sensitive information like AWS keys. In Section 11.4 and Section 12.4, these variables were defined automatically via the SendGrid add-on, but in this case we need to define them explicitly. Per the gem configuration documentation, these variables should be named using the prefix **AWS**, which we can accomplish using **heroku config:set** as follows:

```
$ heroku config:set AWS_ACCESS_KEY=<access key>
$ heroku config:set AWS_SECRET_KEY=<secret key>
$ heroku config:set AWS_REGION=<region>
$ heroku config:set AWS_BUCKET=<bucket name>
```

You should paste in the values for your configuration in place of the placeholder values above. To get the region, you can inspect the URL on the S3 console page (Figure 13.39).

Once the Heroku variables are set, the next step is to use them in a special YAML file for configuring storage options called **`storage.yml`**. We can create a storage option for Amazon using the code in Listing 13.73.

Listing 13.73: Adding Amazon AWS as a storage option.
`config/storage.yml`

```
test:
  service: Disk
  root: <%= Rails.root.join("tmp/storage") %>

local:
  service: Disk
  root: <%= Rails.root.join("storage") %>

amazon:
  service: S3
  access_key_id:      <%= ENV['AWS_ACCESS_KEY_ID'] %>
  secret_access_key: <%= ENV['AWS_SECRET_ACCESS_KEY'] %>
  region:             <%= ENV['AWS_REGION'] %>
  bucket:             <%= ENV['AWS_BUCKET'] %>
```

Finally, we can put the option defined in Listing 13.73 to use in a production environment by adding the Active Storage service configuration parameter in **`production.rb`**. The result appears in Listing 13.74.

Listing 13.74: Configuring the production environment to use Amazon AWS (S3).
`config/environments/production.rb`

```
Rails.application.configure do
  .
  .
  .
  # Store uploaded files on Amazon AWS.
  config.active_storage.service = :amazon
  .
  .
  .
end
```

With the configuration above, we are ready to commit our changes and deploy:

```
$ rails test
$ git add -A
$ git commit -m "Add user microposts"
```

Because so many things can go wrong with the configuration, we'll deploy the app directly from our current topic branch, making sure it's working before merging into **`master`**. We can do this by including the branch name in the push to Heroku as follows:

```
$ git push heroku user-microposts:master
```

As usual, we then reset the database and reseed the sample data:

```
$ heroku pg:reset DATABASE
$ heroku run rails db:migrate
$ heroku run rails db:seed
```

Because Heroku comes with an installation of ImageMagick, the result is successful image resizing and upload in production, as seen in Figure 13.40.

Exercises

Solutions to the exercises are available to all Rails Tutorial purchasers here.

To see other people's answers and to record your own, subscribe to the Rails Tutorial course or to the Learn Enough All Access Bundle.

1. Upload a large image and confirm directly that the resizing is working in production. Does the resizing work even if the image isn't square?

13.5 Conclusion

With the addition of the Microposts resource, we are nearly finished with our sample application. All that remains is to add a social layer by letting users