comment installer-ruby-sur-rails-sur-debian-12

Ruby on Rails or RoR is a free and open-source web application framework written in Ruby and released under the MIT license. Rails is a full-stack web framework for easily building enterprise-grade applications. Rails shipped with different tools that allow developers easily to create both frontend and backend applications. Ruby on Rails also has built-in security features such as protection for common attacks like SQL injection, XSS, and CSRF.

Ruby on Rails provides a default structure for the database, rendering HTML templates, a web service, and a web page. It follows the model-view-controller (MVC) architecture and also uses well-known design philosophies such as Don't Repeat Yourself (DRY), Convention over Configuration (CoC), and active records pattern. Ruby on Rails was designed to be fast and easy to use and learn, Some notable sites developed with Rails such as Twitch, Airbnb, Github, Soundcloud, etc.

In this guide, we'll walk you through the installation of Ruby on Rails on the Debian 12 server. You will install Ruby on Rails with a PostgreSQL database server and Rbenv Ruby version manager. You will also create a scaffold, the basic structure of the Rails project.

Prerequisites

Before commencing, confirm you've got:

- A Debian 12 server.
- A non-root user with sudo administrator privileges.

Installing Dependencies

In the first step, you will install some basic dependencies on your Debian server. This includes the PostgreSQL database server that will be used as the database for your Rails project, the node.js, and the Yarn package manager that will be used by Rails to compile static assets.

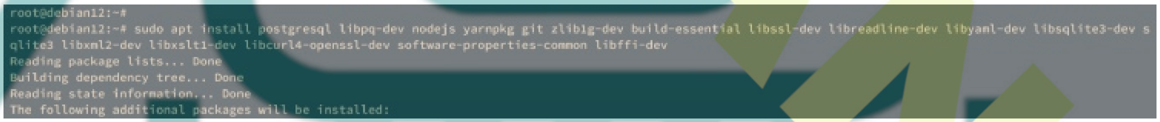
To start, update and refresh your package index by executing the apt update command below.

```
sudo apt update
```

Next, install dependencies using the following apt install command. This includes the PostgreSQL database server, libpq driver, Node.js, Yarn, Git, and some system libraries and tools.

```
sudo apt install postgresql libpq-dev nodejs yarnpkg git zlib1g-dev build-essential libssl-dev libreadline-dev libyaml-dev
libsqlite3-dev sqlite3 libxml2-dev libxslt1-dev libcurl4-openssl-dev software-properties-common libffi-dev
```

Type y to confirm and proceed with the installation.

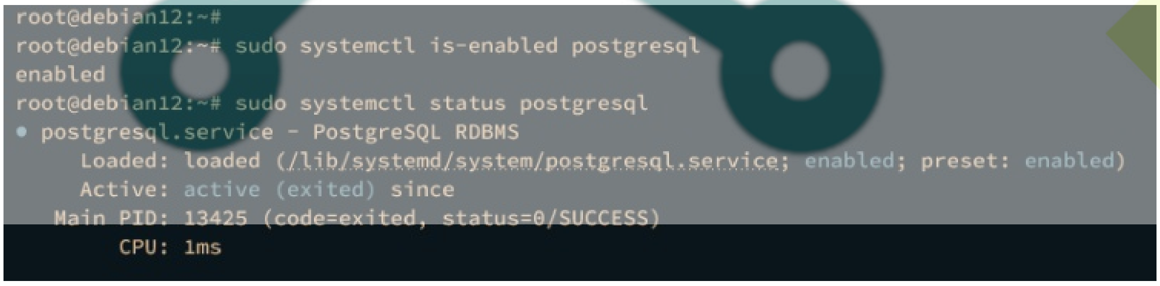


```
root@debian12:~#
root@debian12:~# sudo apt install postgresql libpq-dev nodejs yarnpkg git zlib1g-dev build-essential libssl-dev libreadline-dev libyaml-dev libsqlite3-dev s
qlite3 libxml2-dev libxslt1-dev libcurl4-openssl-dev software-properties-common libffi-dev
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
```

Once dependencies are installed, check the PostgreSQL server status using the following command. This will ensure that PostgreSQL is running and enabled on your Debian machine.

```
sudo systemctl is-enabled postgresql
sudo systemctl status postgresql
```

If PostgreSQL is enabled, you should get the output enabled. When PostgreSQL running, you should get the output active(running) or active(exited).

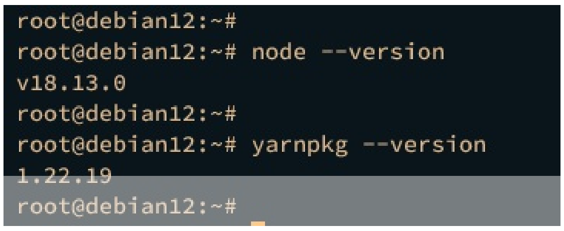


```
root@debian12:~#
root@debian12:~# sudo systemctl is-enabled postgresql
enabled
root@debian12:~# sudo systemctl status postgresql
● postgresql.service - PostgreSQL RDBMS
     Loaded: loaded (/lib/systemd/system/postgresql.service; enabled; preset: enabled)
     Active: active (exited) since
   Main PID: 13425 (code=exited, status=0/SUCCESS)
        CPU: 1ms
```

Lastly, check the Node.js and Yarn package manager by executing the command below.

```
node --version
yarnpkg --version
```

In this example, Node.js 18 and Yarn 1.22 is installed.



```
root@debian12:~#
root@debian12:~# node --version
v18.13.0
root@debian12:~#
root@debian12:~# yarnpkg --version
1.22.19
root@debian12:~#
```

Installing Rbenv

After installing package dependencies, the next step is to install Rbenv, the Ruby version manager for Unix-like operating systems. With Rbenv, you can easily manage your Ruby apps environment, also you can install multiple Ruby versions on your system.

Log in to your User using the following command.

```
su - user
```

Download the **rbenv** source code and the **ruby-build plugin** via the git command below.

```
git clone https://github.com/rbenv/rbenv.git ~/.rbenv
git clone https://github.com/rbenv/ruby-build.git ~/.rbenv/plugins/ruby-build
```

Now execute the following command to add a custom **PATH** to your shell.

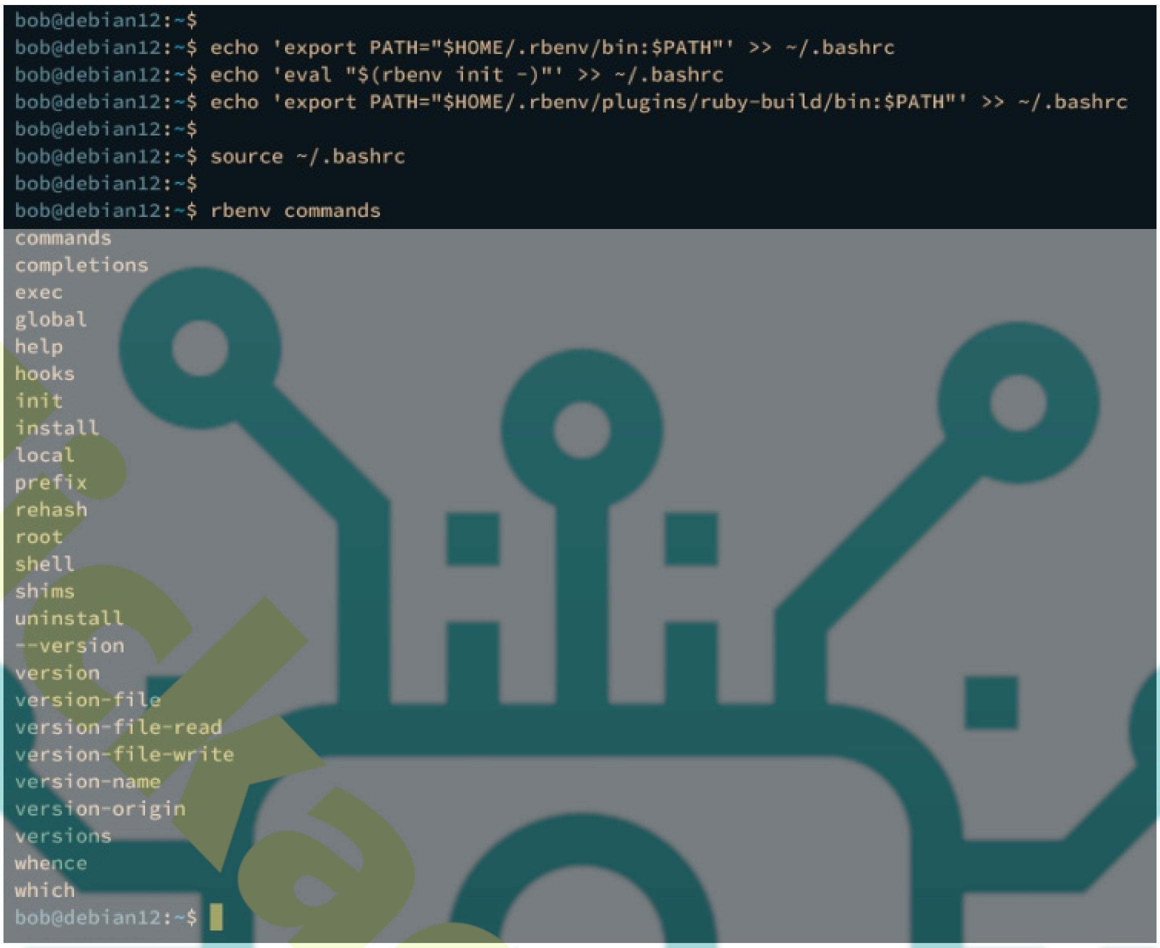
```
echo 'export PATH="$HOME/.rbenv/bin:$PATH"' >> ~/.bashrc
echo 'eval "$(rbenv init -)"' >> ~/.bashrc
echo 'export PATH="$HOME/.rbenv/plugins/ruby-build/bin:$PATH"' >> ~/.bashrc
```

Reload your *~/.bashrc* configuration to apply the changes. After executing the command, your rbenv installation should be activated.

```
source ~/.bashrc
```

Verify rbenv by executing the rbenv command below. If rbenv installation is successful, you should see available rbenv commands/options.

```
rbenv commands
```



```
bob@debian12:~$
bob@debian12:~$ echo 'export PATH="$HOME/.rbenv/bin:$PATH"' >> ~/.bashrc
bob@debian12:~$ echo 'eval "$(rbenv init -)"' >> ~/.bashrc
bob@debian12:~$ echo 'export PATH="$HOME/.rbenv/plugins/ruby-build/bin:$PATH"' >> ~/.bashrc
bob@debian12:~$
bob@debian12:~$ source ~/.bashrc
bob@debian12:~$
bob@debian12:~$ rbenv commands
commands
completions
exec
global
help
hooks
init
install
local
prefix
rehash
root
shell
shims
uninstall
--version
version
version-file
version-file-read
version-file-write
version-name
version-origin
versions
whence
which
bob@debian12:~$
```

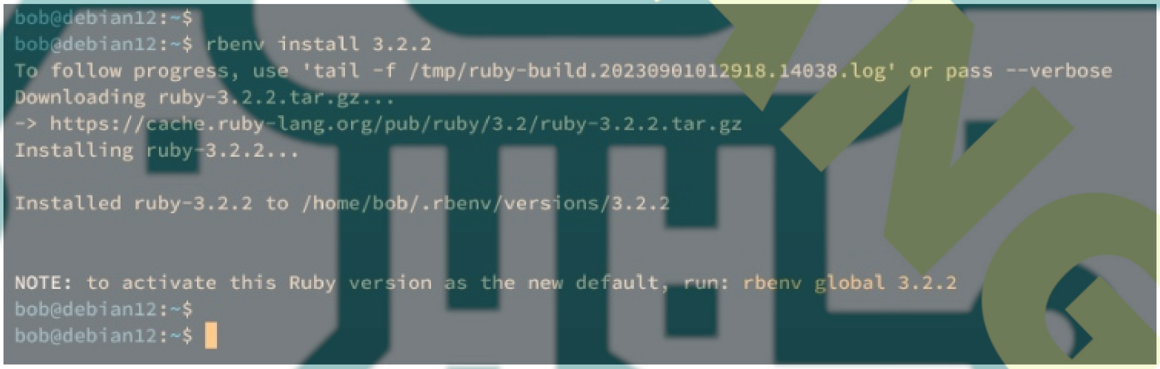
Installing Ruby via Rbenv

With Rbenv installed, you can now install Ruby on your system. With Rbenv, you will install Ruby on your current environment only, which does not affect the whole system. You will install Ruby 3.2.2 to your current user environment.

Execute the rbenv install command below to install Ruby 3.2.2 to your system.

```
rbenv install 3.2.2
```

During the installation, you should see this:



```
bob@debian12:~$
bob@debian12:~$ rbenv install 3.2.2
To follow progress, use 'tail -f /tmp/ruby-build.20230901012918.14038.log' or pass --verbose
Downloading ruby-3.2.2.tar.gz...
-> https://cache.ruby-lang.org/pub/ruby/3.2/ruby-3.2.2.tar.gz
Installing ruby-3.2.2...

Installed ruby-3.2.2 to /home/bob/.rbenv/versions/3.2.2


NOTE: to activate this Ruby version as the new default, run: rbenv global 3.2.2
bob@debian12:~$
bob@debian12:~$
```

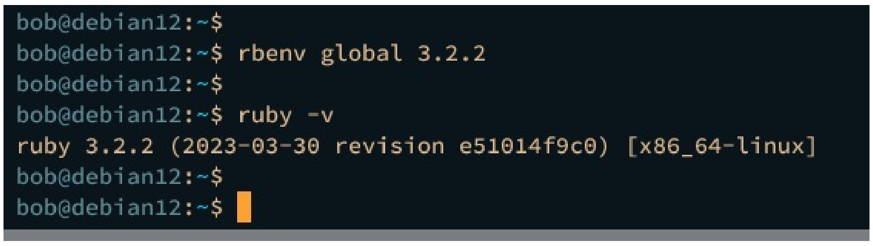
Once Ruby is installed, execute the following command to set up the default Ruby version to 3.2.2.

```
rbenv global 3.2.2
```

Lastly, verify the Ruby version on your system using the command below.

```
ruby -v
```

If everything goes well, you should see Ruby 3.2.2 is installed.


```
bob@debian12:~$
bob@debian12:~$ rbenv global 3.2.2
bob@debian12:~$
bob@debian12:~$ ruby -v
ruby 3.2.2 (2023-03-30 revision e51014f9c0) [x86_64-linux]
bob@debian12:~$
bob@debian12:~$
```

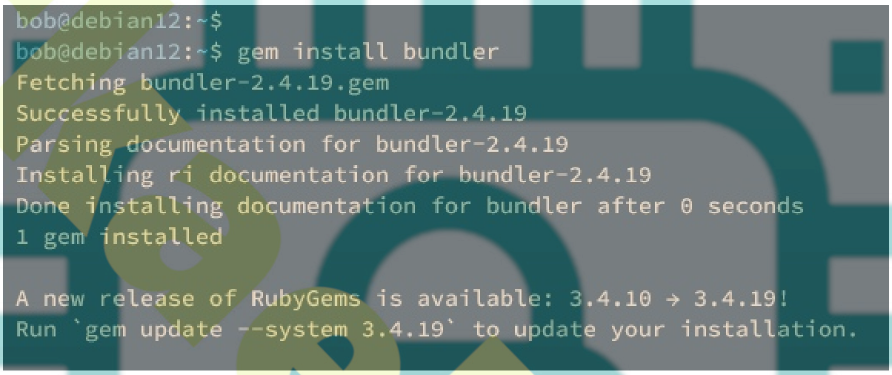
Installing Ruby on Rails

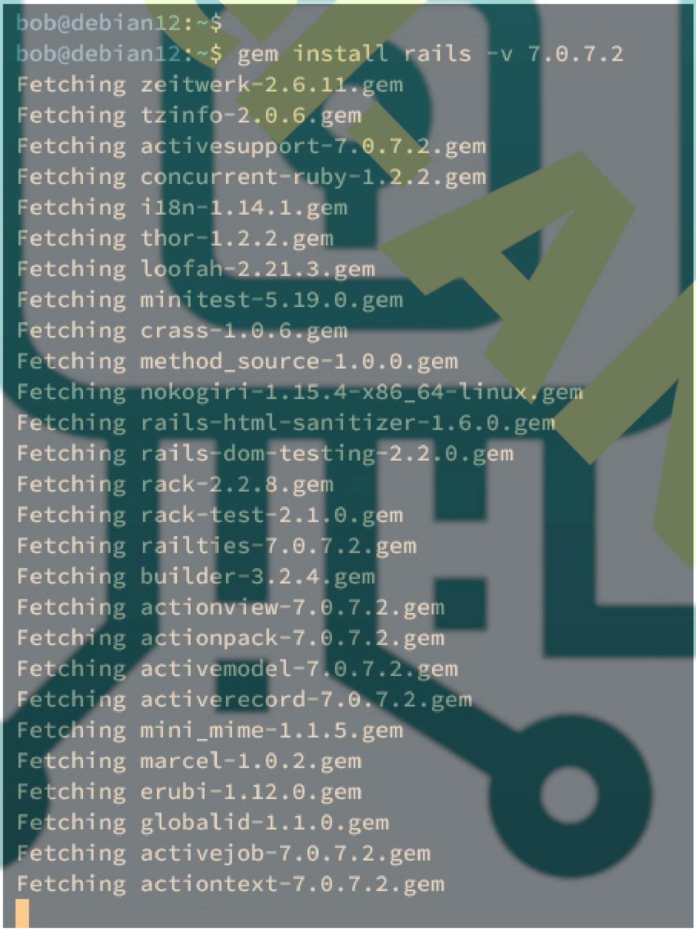
At this point, your system is configured and ready to install Ruby on Rails to your Debian machine. In this example, you will install Ruby on Rails 7.0, and check the list of available versions of Rails on the official site.

Execute the gem commands below to install the bundler, then install Ruby on Rails 7.0.7.2.

```
gem install bundler
gem install rails -v 7.0.7.2
```

During the installation, you should see an output like the following:


```
bob@debian12:~$
bob@debian12:~$ gem install bundler
Fetching bundler-2.4.19.gem
Successfully installed bundler-2.4.19
Parsing documentation for bundler-2.4.19
Installing ri documentation for bundler-2.4.19
Done installing documentation for bundler after 0 seconds
1 gem installed

A new release of RubyGems is available: 3.4.10 → 3.4.19!
Run `gem update --system 3.4.19` to update your installation.
```


```
bob@debian12:~$
bob@debian12:~$ gem install rails -v 7.0.7.2
Fetching zeitwerk-2.6.11.gem
Fetching tzinfo-2.0.6.gem
Fetching activesupport-7.0.7.2.gem
Fetching concurrent-ruby-1.2.2.gem
Fetching i18n-1.14.1.gem
Fetching thor-1.2.2.gem
Fetching loofah-2.21.3.gem
Fetching minitest-5.19.0.gem
Fetching crass-1.0.6.gem
Fetching method_source-1.0.0.gem
Fetching nokogiri-1.15.4-x86_64-linux.gem
Fetching rails-html-sanitizer-1.6.0.gem
Fetching rails-dom-testing-2.2.0.gem
Fetching rack-2.2.8.gem
Fetching rack-test-2.1.0.gem
Fetching railties-7.0.7.2.gem
Fetching builder-3.2.4.gem
Fetching actionview-7.0.7.2.gem
Fetching actionpack-7.0.7.2.gem
Fetching activemodel-7.0.7.2.gem
Fetching activerecord-7.0.7.2.gem
Fetching mini_mime-1.1.5.gem
Fetching marcel-1.0.2.gem
Fetching erubi-1.12.0.gem
Fetching globalid-1.1.0.gem
Fetching activejob-7.0.7.2.gem
Fetching actiontext-7.0.7.2.gem
```

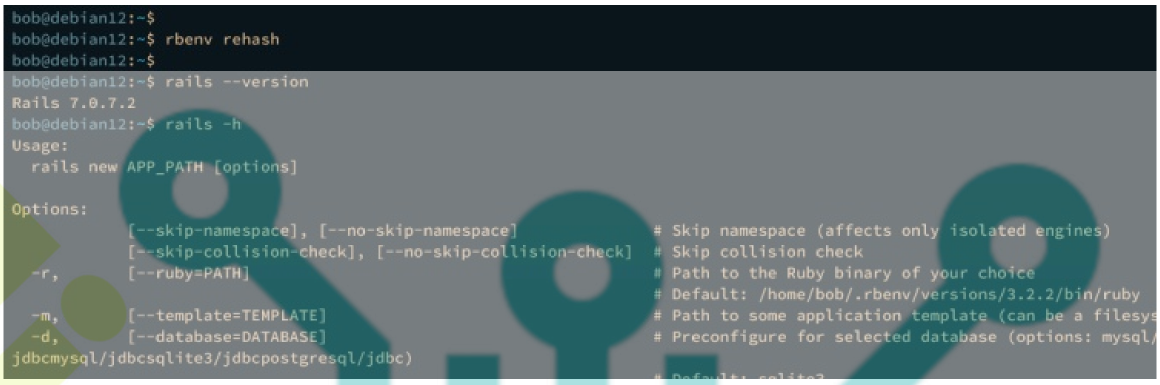
Now run the rebenv command below to rehash and reload your current environment.

```
rbenv rehash
```

Lastly, execute the rails command below to ensure that Ruby on Rails is installed.

```
rails version
rails -h
```

If the installation is successful, you should see your current Rails version and the help page of the rails command.



```
bob@debian12:~$
bob@debian12:~$ rbenv rehash
bob@debian12:~$
bob@debian12:~$ rails --version
Rails 7.0.7.2
bob@debian12:~$ rails -h
Usage:
  rails new APP_PATH [options]

Options:
            [--skip-namespace], [--no-skip-namespace]              # Skip namespace (affects only isolated engines)
            [--skip-collision-check], [--no-skip-collision-check]  # Skip collision check
  -r,       [--ruby=PATH]                                          # Path to the Ruby binary of your choice
                                                                   # Default: /home/bob/.rbenv/versions/3.2.2/bin/ruby
  -m,       [--template=TEMPLATE]                                  # Path to some application template (can be a filesys
  -d,       [--database=DATABASE]                                  # Preconfigure for selected database (options: mysql/
jdbcmysql/jdbcsqlite3/jdbcpostgresql/jdbc)
```

Creating First Rails Project

In this section, you will learn how to create your first project with Ruby on Rails. You will be using PostgreSQL as the default database for your Rails project. To achieve that you must complete the following:

- Preparing the PostgreSQL user for application.
- Creating the first Rails project.

Prepare Database User

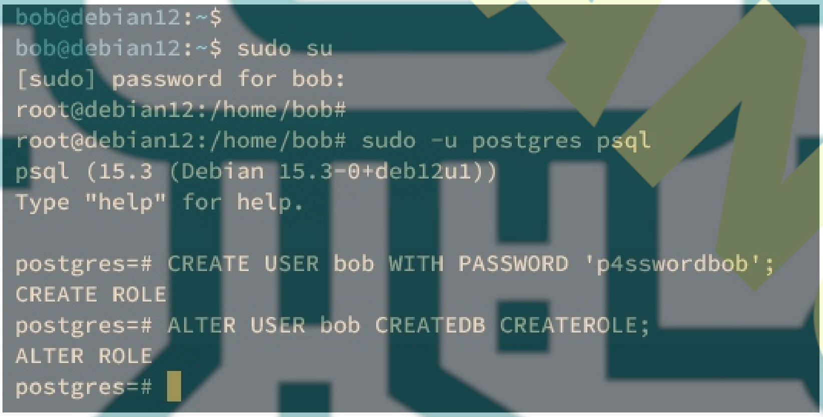
First, you must create a new PostgreSQL user that will be used for your Rails application. This user must have privileges for creating databases and users.

Back to your user account and log in to the PostgreSQL server using the command below.

```
sudo su
sudo -u postgres psql
```

Now create a new user **bob** with the password **p4sswordbob**. Then, assign new privileges for creating a database and roles to the user **bob**.

```
CREATE USER bob WITH PASSWORD 'p4sswordbob';
ALTER USER bob CREATEDB CREATEROLE;
```

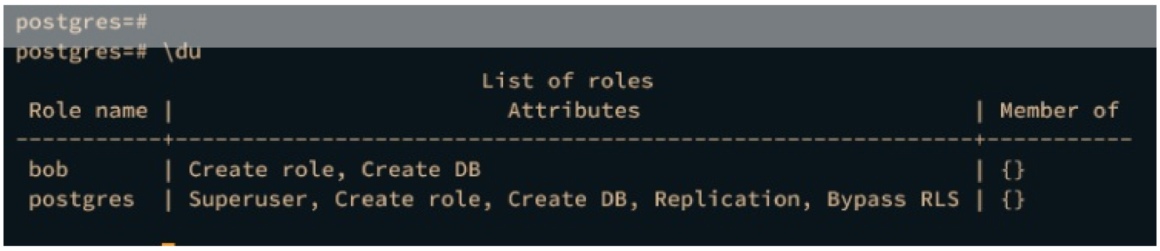


```
bob@debian12:~$
bob@debian12:~$ sudo su
[sudo] password for bob:
root@debian12:/home/bob#
root@debian12:/home/bob# sudo -u postgres psql
psql (15.3 (Debian 15.3-0+deb12u1))
Type "help" for help.

postgres=# CREATE USER bob WITH PASSWORD 'p4sswordbob';
CREATE ROLE
postgres=# ALTER USER bob CREATEDB CREATEROLE;
ALTER ROLE
postgres=#
```

Verify the list of users and privileges on your PostgreSQL server using the command below.

```
\du
```

You should see the user bob with privileges CREATEDB and CREATEROLE.



```
postgres=#
postgres=# \du
                                   List of roles
 Role name |                         Attributes                         | Member of
-----------+------------------------------------------------------------+-----------
 bob       | Create role, Create DB                                     | {}
 postgres  | Superuser, Create role, Create DB, Replication, Bypass RLS | {}
```

Type *\q* to log out from the PostgreSQL server.

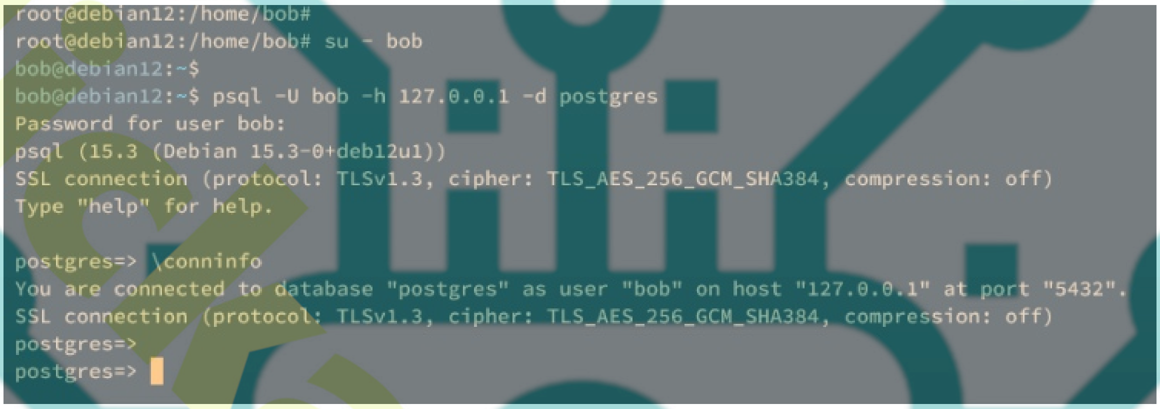
Lastly, log in to your user and execute the following psql command to log in to the PostgreSQL server as the new user bob.

```
su - user
psql -U bob -h 127.0.0.1 -d postgres
```

Once connected to the PostgreSQL server, execute the following query to verify your connection information.

```
\conninfo
```

You should see that you've connected to the PostgreSQL server as a user bob.



```
root@debian12:/home/bob#
root@debian12:/home/bob# su - bob
bob@debian12:~$
bob@debian12:~$ psql -U bob -h 127.0.0.1 -d postgres
Password for user bob:
psql (15.3 (Debian 15.3-0+deb12u1))
SSL connection (protocol: TLSv1.3, cipher: TLS_AES_256_GCM_SHA384, compression: off)
Type "help" for help.

postgres=> \conninfo
You are connected to database "postgres" as user "bob" on host "127.0.0.1" at port "5432".
SSL connection (protocol: TLSv1.3, cipher: TLS_AES_256_GCM_SHA384, compression: off)
postgres=>
postgres=>
```

Type *\q* to exit from the PostgreSQL server.

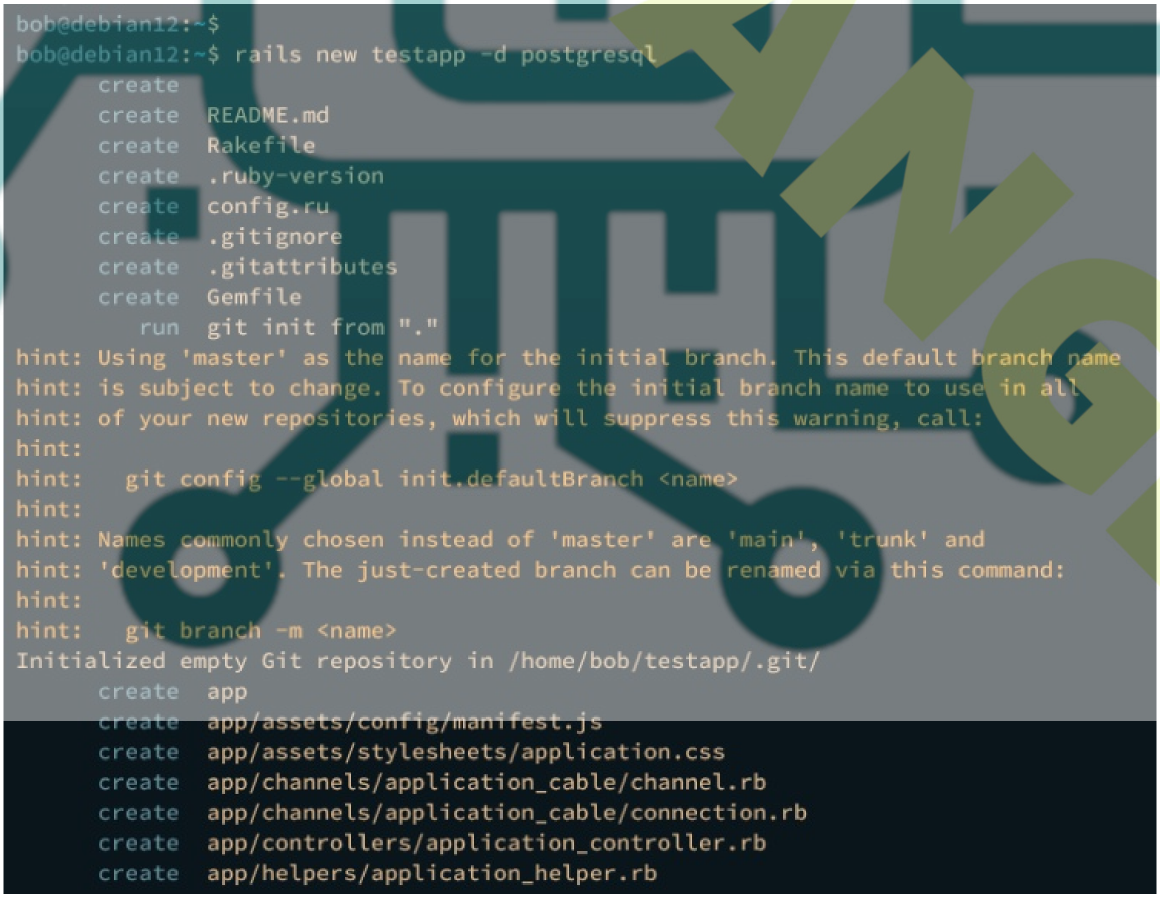
Creating Rails Project

After creating a PostgreSQL user, you can now start creating a new Rails project via the rails command-line utility.

To create a new rails project, run the rails command below. In this example, you will create a new project testapp with the default database PostgreSQL.

```
rails new testapp -d postgresql
```

The output of the command should look like this:



```
bob@debian12:~$
bob@debian12:~$ rails new testapp -d postgresql
      create
      create  README.md
      create  Rakefile
      create  .ruby-version
      create  config.ru
      create  .gitignore
      create  .gitattributes
      create  Gemfile
         run  git init from "."
hint: Using 'master' as the name for the initial branch. This default branch name
hint: is subject to change. To configure the initial branch name to use in all
hint: of your new repositories, which will suppress this warning, call:
hint:
hint:   git config --global init.defaultBranch <name>
hint:
hint: Names commonly chosen instead of 'master' are 'main', 'trunk' and
hint: 'development'. The just-created branch can be renamed via this command:
hint:
hint:   git branch -m <name>
Initialized empty Git repository in /home/bob/testapp/.git/
      create  app
      create  app/assets/config/manifest.js
      create  app/assets/stylesheets/application.css
      create  app/channels/application_cable/channel.rb
      create  app/channels/application_cable/connection.rb
      create  app/controllers/application_controller.rb
      create  app/helpers/application_helper.rb
```

After the project is created, the new directory *~/testapp* will also be created. Move into the *~/testapp* directory and open the database configuration *config/database.yml* using your preferred text editor.

```
cd testapp/
nano config/database.yml
```

Change the default database settings for development, test, and production. Be sure to input your PostgreSQL username and password.

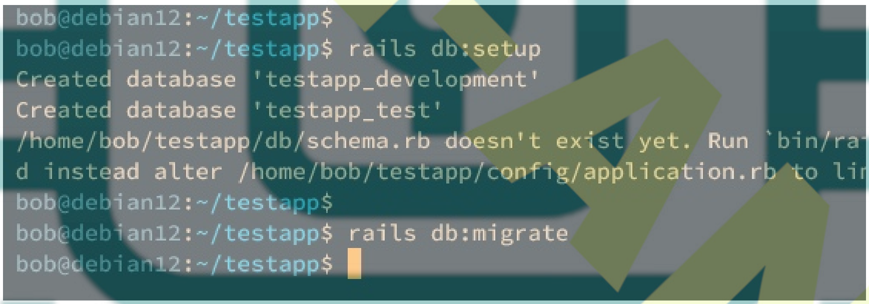
```
development:
  <<: *default
  database: testapp_development

  # The specified database role is being used to connect to postgres.
  # To create additional roles in postgres see `$ createuser --help`.
  # When left blank, postgres will use the default role. This is
  # the same name as the operating system user running Rails.
  username: bob

  # The password associated with the postgres role (username).
  password: p4sswordbob

  # Connect on a TCP socket. Omitted by default since the client uses a
  # domain socket that doesn't need configuration. Windows does not have
  # domain sockets, so uncomment these lines.
  host: localhost

  # The TCP port the server listens on. Defaults to 5432.
  # If your server runs on a different port number, change accordingly.
  port: 5432
```

Save and exit the file when you're done.

Now run the rails command below to migrate the database. This will automatically create a new database for your testapp project.

```
rails db:setup
rails db:migrate
```

Below you should see the output during the database migration of the testapp project.

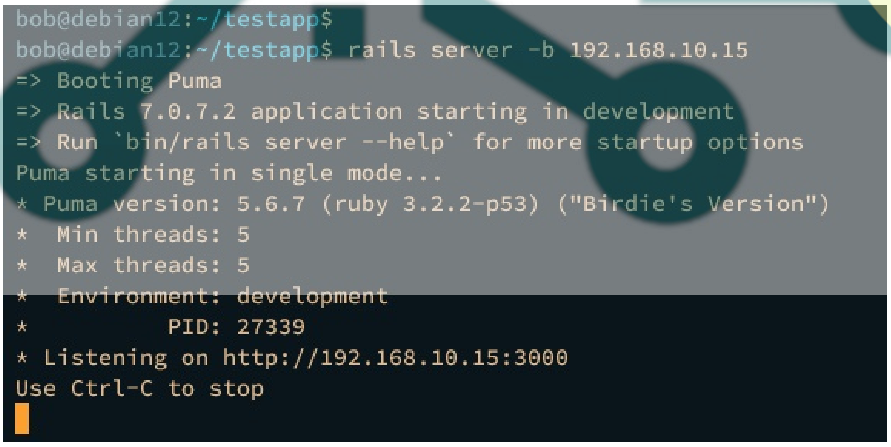


```
bob@debian12:~/testapp$
bob@debian12:~/testapp$ rails db:setup
Created database 'testapp_development'
Created database 'testapp_test'
/home/bob/testapp/db/schema.rb doesn't exist yet. Run `bin/ra
d instead alter /home/bob/testapp/config/application.rb to li
bob@debian12:~/testapp$
bob@debian12:~/testapp$ rails db:migrate
bob@debian12:~/testapp$
```

After the database is migrated, execute the rails command below to run the testapp project. This will run testapp within your IP address on port 3000.

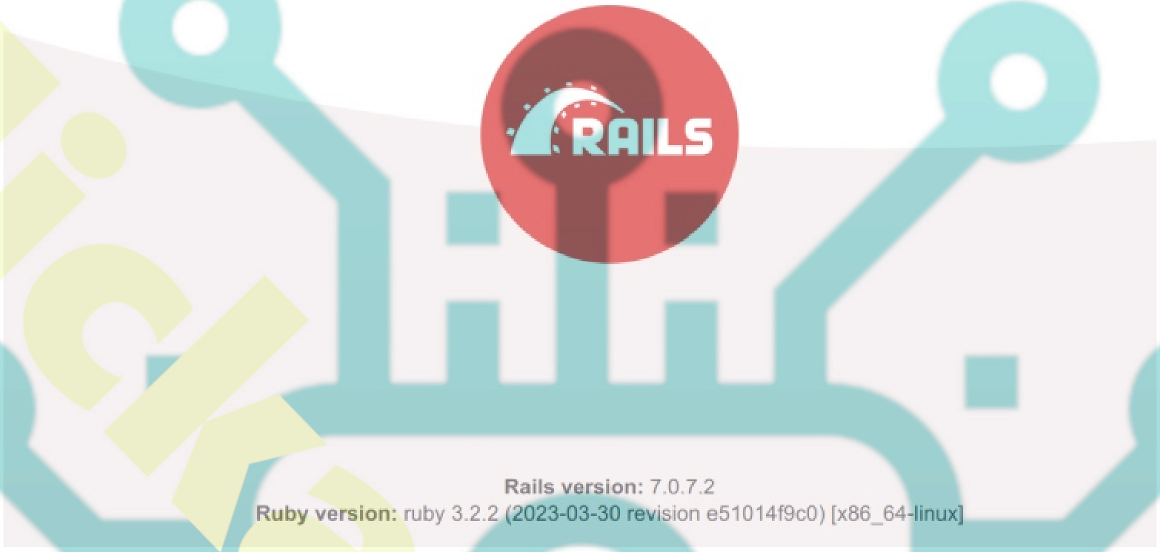
```
rails server -b 192.168.10.15
```

In the following output, you should see that testapp is running.



```
bob@debian12:~/testapp$
bob@debian12:~/testapp$ rails server -b 192.168.10.15
=> Booting Puma
=> Rails 7.0.7.2 application starting in development
=> Run `bin/rails server --help` for more startup options
Puma starting in single mode...
* Puma version: 5.6.7 (ruby 3.2.2-p53) ("Birdie's Version")
*  Min threads: 5
*  Max threads: 5
*  Environment: development
*          PID: 27339
* Listening on http://192.168.10.15:3000
Use Ctrl-C to stop
```

Now launch your favorite web browser and visit your server IP address followed by port 3000, such as http://192.168.10.15:3000/. If your installation is successful, you should see the default index.html page of Ruby on Rails.



Rails version: 7.0.7.2
Ruby version: ruby 3.2.2 (2023-03-30 revision e51014f9c0) [x86_64-linux]

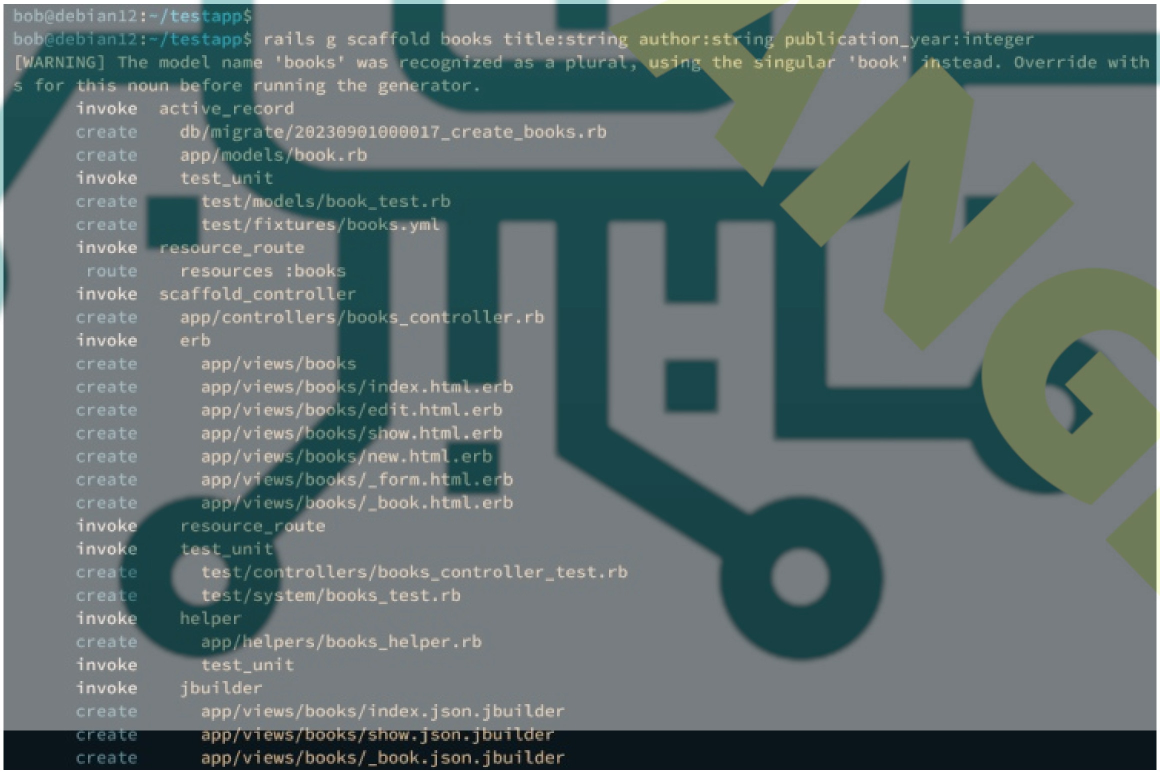
Press Ctrl+c to terminate your Rails application.

Rails Scaffolding for Starter Kit

A scaffold is an automatic way to generate the basic structure of a Rails project, which includes a controller, a model, and a view.

Execute the rails command below to create scaffold *books* with three fields *title*, *author*, and *publication_year*.

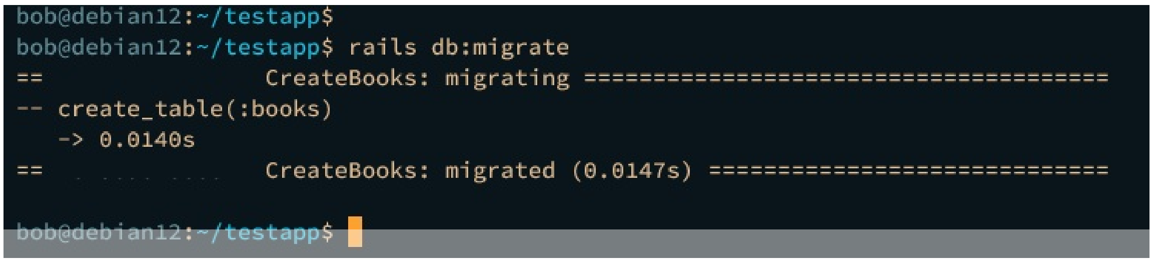
```
rails g scaffold books title:string author:string publication_year:integer
```



```
bob@debian12:~/testapp$
bob@debian12:~/testapp$ rails g scaffold books title:string author:string publication_year:integer
[WARNING] The model name 'books' was recognized as a plural, using the singular 'book' instead. Override with
s for this noun before running the generator.
      invoke  active_record
      create    db/migrate/20230901000017_create_books.rb
      create    app/models/book.rb
      invoke    test_unit
      create      test/models/book_test.rb
      create      test/fixtures/books.yml
      invoke  resource_route
       route    resources :books
      invoke  scaffold_controller
      create    app/controllers/books_controller.rb
      invoke    erb
      create      app/views/books
      create      app/views/books/index.html.erb
      create      app/views/books/edit.html.erb
      create      app/views/books/show.html.erb
      create      app/views/books/new.html.erb
      create      app/views/books/_form.html.erb
      create      app/views/books/_book.html.erb
      invoke    resource_route
      invoke    test_unit
      create      test/controllers/books_controller_test.rb
      create      test/system/books_test.rb
      invoke    helper
      create      app/helpers/books_helper.rb
      invoke      test_unit
      invoke    jbuilder
      create      app/views/books/index.json.jbuilder
      create      app/views/books/show.json.jbuilder
      create      app/views/books/_book.json.jbuilder
```

Now migrate the database to apply the changes using the rails command below.

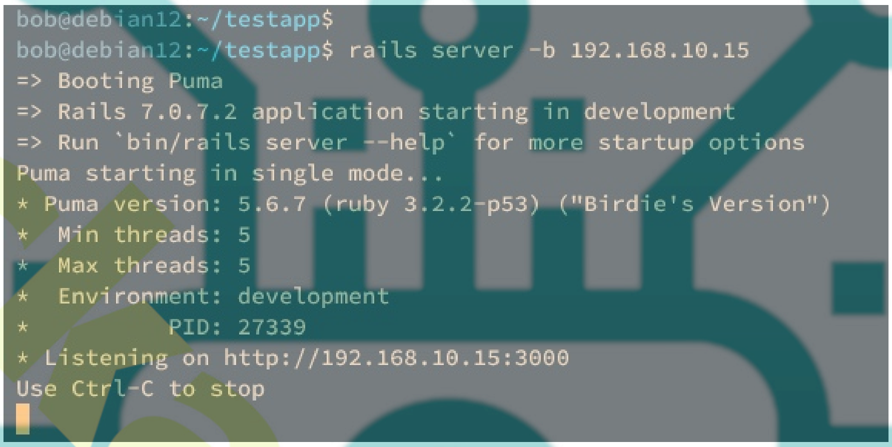
```
rails db:migrate
```



```
bob@debian12:~/testapp$
bob@debian12:~/testapp$ rails db:migrate
==                CreateBooks: migrating ======================================
-- create_table(:books)
   -> 0.0140s
==                CreateBooks: migrated (0.0147s) =============================

bob@debian12:~/testapp$
```

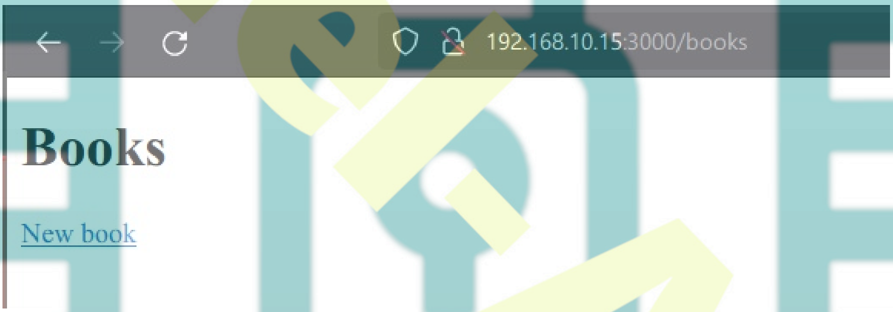
Next, run your Rails project by executing the rails server command below.

```
rails server -b 192.168.10.15
```

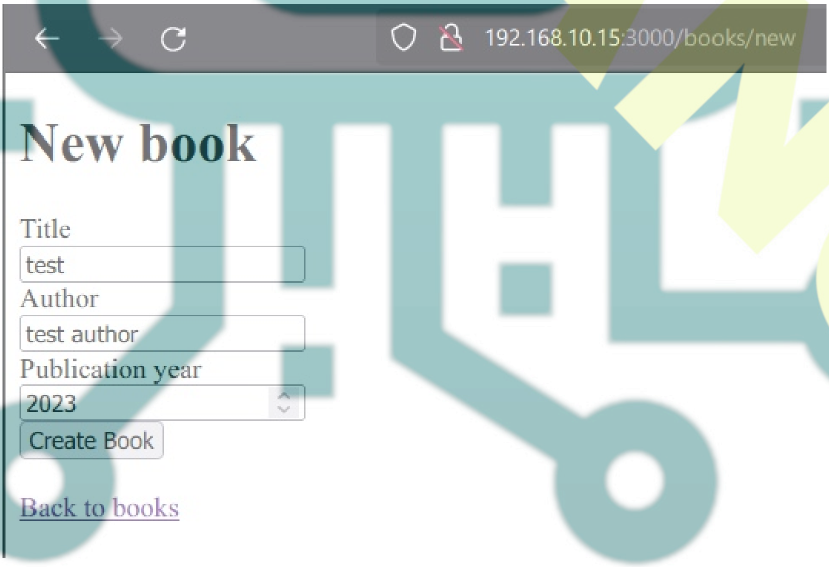


```
bob@debian12:~/testapp$
bob@debian12:~/testapp$ rails server -b 192.168.10.15
=> Booting Puma
=> Rails 7.0.7.2 application starting in development
=> Run `bin/rails server --help` for more startup options
Puma starting in single mode...
* Puma version: 5.6.7 (ruby 3.2.2-p53) ("Birdie's Version")
*  Min threads: 5
*  Max threads: 5
*  Environment: development
*          PID: 27339
* Listening on http://192.168.10.15:3000
Use Ctrl-C to stop
```

Once testapp is running, check the books scaffold via URL path */books*, such as http://192.168.10.15:3000/books. If everything goes well, you should see the generated scaffold like the following:



You can now insert new data to the books scaffold like the following:



Conclusion

In conclusion, you've completed the installation of Ruby on Rails with the PostgreSQL database server and Rbenv on the Debian 12 server. You've also learned how to generate scaffolds for basic structures for the Rails project.