
HelpButtons

Comunidad de HelpButtons

April 26, 2021

Contents

1 Home	3
2 What is HelpButtons?	3
3 Help Buttons installations	3
3.1 On GNU/Linux	3
3.1.1 RVM (Ruby Version Manager) Installation	3
3.1.2 Installing the ruby version of the project	4
3.1.3 Installation and Configuration of PostgreSQL	5
3.1.4 Installation of Nodejs and Yarn	6
3.2 Running the project	6
3.2.1 Hb-backend	6
3.2.2 Hb-frontend	8
3.3 Possible errors during installation	9
3.3.1 Command not found	9
3.3.2 Ruby version	9

1 Home

2 What is HelpButtons?

3 Help Buttons installations

3.1 On GNU/Linux

References followed:

- [Official Ruby on Rails installation documentation in English](#)
- [Official RVM documentation](#)

To install Ruby on Rails (RoR) on Debian or derivatives (Ubuntu) there are the following requirements:

- RVM
- PostgreSQL
- Nodejs
- Yarn

3.1.1 RVM (Ruby Version Manager) Installation

We install the operating system packages required by RVM:

```
1 sudo apt install -y git-core subversion gnupg2 curl
```

Add the signature of the RVM repository that can be found on its [official website](#):

```
1 gpg --keyserver hkp://pool.sks-keyservers.net --recv-keys 409
  B6B1796C275462A1703113804BB82D39DC0E3 7
  D2BAF1CF37B13E2069D6956105BD0E739499BDB
```

We install the stable version of RVM:

```
1 curl -sSL https://get.rvm.io | bash -s stable --ruby
```

This command installs the latest stable versions of **rvm**, **ruby** and **rails**. When executed, it returns many messages on the screen.

Once this process is finished, RVM tells us to run the next command or to restart the open shells. We will also run this command in case we get an error saying that the bundle command has not been found `bash: bundle: command not found`

```
1 source /home/$USER/.rvm/scripts/rvm
```

The rvm configuration is checked using the following command:

```
1 type rvm | head -n 1
```

Which will return that `rvm is a function`. If this is not the case, rvm is not properly configured. In such a case you should consult the documentation at <https://rvm.io/rvm/install> and ask for help.

Next, we check the commands: rvm, ruby, irb, gem.

```
1 rvm list
2 ruby -v
3 irb -v
4 gem -v
```

3.1.2 Installing the ruby version of the project

Bundle is a ruby gem that keeps track and installs the different versions of the gems we need. We now have the environment installed and available to work with.

When we try to install the bundle with the command below, we will probably encounter an error `Your Ruby version is Y.Y.Y.Y, but your Gemfile specified X.X.X.X`.

```
1 bundle install
```

To resolve this, you need to install and use the appropriate X.X.X. version.

```
1 rvm install X.X.X
2 rvm use X.X.X
```

If we run the `bundle install` command again we will get another error indicating that the PostgreSQL database is missing.

3.1.3 Installation and Configuration of PostgreSQL

The packages required for PostgreSQL installation are installed:

```
1 sudo apt install postgresql postgresql-contrib libpq-dev -y
```

The service is lifted with:

```
1 systemctl start postgresql
2 systemctl enable postgresql
```

Open a PostgreSQL console to modify the password of the user `postgres`:

```
1 sudo -i -u postgres psql
```

The following command is entered to change the password:

```
1 \password postgres
```

Next, a new PostgreSQL user named `'rails_dev` is created with privileges to create a database with `createdb` and a password `aq1234567890` (it is recommended to change it to a stronger one, ideally generated with KeePassXC or BitWarden)

```
1 create role hb_dev with createdb login password `aq1234567890`;
```

PostgreSQL users are shown with:

```
1 \p
```

Type `exit` to exit the PostgreSQL console.

Run `'bundle install` again to continue with the installation.

```
1 bundle install
```

It finishes the execution without errors. The next step is to install Nodejs and Yarn.

3.1.4 Installation of Nodejs and Yarn

First we install the development tools for compiling C.

```
1 sudo apt install gcc g++ make
```

The Nodejs 14 repositories are installed via the following command:

```
1 curl -sL https://deb.nodesource.com/setup_14.x | sudo bash -
```

Nodejs is installed via the following command:

```
1 sudo apt-get install -y nodejs
```

Check nodejs and npm installation with:

```
1 node -v
2 npm -v
```

The signature and yarn repositories are added using:

```
1 curl -sL https://dl.yarnpkg.com/debian/pubkey.gpg | sudo apt-key add -
2 echo "deb https://dl.yarnpkg.com/debian/ stable main" | sudo tee /etc/apt/sources.list.d/yarn.list
```

Install yarn:

```
1 sudo apt update && sudo apt install yarn
```

3.2 Running the project

3.2.1 Hb-backend

We create the database configuration file `config/database.yml` with vim or nano (nano is recommended in case you don't know how to use vim):

```
1 vim config/database.yml
```

By default, this file will not exist because it is included in the `.gitignore`. Therefore, you can take a template and add the following sections or add

the whole document as discussed below: - In the **development** section, add the PostgreSQL configuration:

```
1 development:
2   <<: *default
3   database: test_project_development
4   username: hb_dev
5   password: aq1234567890
6   host: localhost
7   port: 5432
```

- In the **testing** section, add the PostgreSQL configuration:

```
1 test:
2   <<: *default
3   database: test_project_test
4   host: localhost
5   port: 5432
6   username: hb_dev
7   password: aq1234567890
```

An example of the final `config/database.yml` file will look like this:

```
1 default: &default
2   adapter: postgresql
3   encoding: unicode
4   pool: <%= ENV.fetch("RAILS_MAX_THREADS") { 5 } %>
5
6 development:
7   <<: *default
8   database: test_project_development
9   username: hb_dev
10  password: aq1234567890
11  host: localhost
12  port: 5432
13
14 test:
15   <<: *default
16   database: test_project_test
17   username: hb_dev
18   password: aq1234567890
19   host: localhost
20   port: 5432
21
22 production:
```

```
23 <<: *default
24 database: test_project_production
25 username: test_project
26 password: <%= ENV['TEST_PROJECT_DATABASE_PASSWORD'] %>
```

Save the changes and exit the editor.

Next you will generate the PostgreSQL database schema by running:

```
1 rake db:setup
```

Or alternatively:

```
1 rails db:setup
2 rails db:migrate
```

Lastly, bring up the server locally and access the address <http://localhost:3000/>.

```
1 rails s
```

3.2.2 Hb-frontend

We install ember globally:

```
1 npm install -g ember-cli
```

In case we get an error related to access permissions 'npm ERR! Error: EACCES,, change the owner of the `node_modules` folder from root to the current user to be able to access the directory:

```
1 sudo chown -R $USER /usr/lib/node_modules
```

We install the front-end dependencies:

```
1 npm install
```

The frontend is served and accessed from the browser at the url <http://localhost:4200/>:

```
1 ember s
```


3.3 Possible errors during installation

3.3.1 Command not found

- **Bundle**:** If running `bundle install` returns the error `bash: bundle: command not found` it will be necessary to run it again:

```
1 source /home/$USER/.rvm/scripts/rvm
```

- **Rails**:** If running `rails db:setup` returns the error `bash: rails: command not found`, it will be necessary to run again:

```
1 source /home/$USER/.rvm/scripts/rvm
```

- **ember:** If running `ember -v` returns the error `bash: ember: command not found`, it will be necessary to run again:

```
1 export PATH=node_modules/.bin:$PATH
```

Add it to the PATH to have a permanent solution. For example, if you are using bash as a console, you can include the commands as last lines in the `.bashrc` file and then apply the changes with `source .bashrc`.

3.3.2 Ruby version

When trying to install the bundle with the command below, you will probably encounter an error `Your Ruby version is Y.Y.Y.Y, but your Gemfile specified X.X.X.X`.

```
1 bundle install
```

To resolve this, you need to install and use the appropriate X.X.X. version.

```
1 rvm install X.X.X
2 rvm use X.X.X
```