Setup Ruby On Rails on Ubuntu 14.04 Trusty Tahr

A guide to setting up a Ruby on Rails development environment

Ubuntu 15.04 (/setup/ubuntu/15.04) Ubuntu 14.10 (/setup/ubuntu/14.10)

Ubuntu 14.04 (/setup/ubuntu/14.04) Ubuntu 13.10 (/setup/ubuntu/13.10)

Ubuntu 13.04 (/setup/ubuntu/13.04) Mac OSX (/setup/osx)

Overview

This will take about 30 minutes.

We will be setting up a Ruby on Rails development environment on Ubuntu 14.04 Trusty Tahr.

The reason we're going to be using Ubuntu is because the majority of code you write will run on a Linux server. Ubuntu is one of the easiest Linux distributions to use with lots of documentation so it's a great one to start with.

You'll want to download the latest Desktop version here: http://releases.ubuntu.com/14.04/ (http://releases.ubuntu.com/14.04/)

Some of you may choose to develop on Ubuntu Server so that your development environment matches your production server. You can find it on the same download link above.

Installing Ruby

Choose the version of Ruby you want to install:

2.2.1 (Recommended) ▾

The first step is to install some dependencies for Ruby.

```
sudo apt-get update
sudo apt-get install git-core curl zlib1g-dev build-essential libssl-dev li
```

Next we're going to be installing Ruby using one of three methods. Each have their own benefits, most people prefer using rbenv these days, but if you're familiar with rvm you can follow those steps as well. I've included instructions for installing from source as well, but in general, you'll want to choose either rbenv or rvm.

Choose one method. Some of these conflict with each other, so choose the one that sounds the most interesting to you, or go with my suggestion, rbenv.

Using rbenv (Recommended) Using rvm From source

Installing with `rbenv` is a simple two step process. First you install `rbenv` , and then `ruby-build` :

```
cd
git clone git://github.com/sstephenson/rbenv.git .rbenv
echo 'export PATH="$HOME/.rbenv/bin:$PATH"' >> ~/.bashrc
echo 'eval "$(rbenv init -)"' >> ~/.bashrc
exec $SHELL

git clone git://github.com/sstephenson/ruby-build.git ~/.rbenv/plugins/ruby
echo 'export PATH="$HOME/.rbenv/plugins/ruby-build/bin:$PATH"' >> ~/.bashrc
exec $SHELL

git clone https://github.com/sstephenson/rbenv-gem-rehash.git ~/.rbenv/plug

rbenv install 2.2.1
rbenv global 2.2.1
ruby -v
```

Now we tell Rubygems not to install the documentation for each package locally and then install Bundler

```
echo "gem: --no-ri --no-rdoc" > ~/.gemrc
gem install bundler
```

Configuring Git

We'll be using Git for our version control system so we're going to set it up to match our Github account. If you don't already have a Github account, make sure to register (https://github.com). It will come in handy for the future.

Replace my name and email address in the following steps with the ones you used for your Github account.

```
git config --global color.ui true
git config --global user.name "YOUR NAME"
git config --global user.email "YOUR@EMAIL.com"
ssh-keygen -t rsa -C "YOUR@EMAIL.com"
```

The next step is to take the newly generated SSH key and add it to your Github account. You want to copy and paste the output of the following command and paste it here (https://github.com/settings/ssh).

```
cat ~/.ssh/id_rsa.pub
```

Once you've done this, you can check and see if it worked:

```
ssh -T git@github.com
```

You should get a message like this:

```
Hi excid3! You've successfully authenticated, but GitHub does not provide s
```

Installing Rails

Choose the version of Rails you want to install:

4.2.0 (Recommended) ▾

Since Rails ships with so many dependencies these days, we're going to need to install a Javascript runtime like NodeJS. This lets you use Coffeescript and the Asset Pipeline (http://guides.rubyonrails.org/asset_pipeline.html) in Rails which combines and minifies your javascript to provide a faster production environment.

To install NodeJS, we're going to add it using a PPA repository:

```
sudo add-apt-repository ppa:chris-lea/node.js
sudo apt-get update
sudo apt-get install nodejs
```

And now, without further adieu:

```
gem install rails -v 4.2.0
```

If you're using rbenv, you'll need to run the following command to make the rails executable available:

```
rbenv rehash
```

Now that you've installed Rails, you can run the `rails -v` command to make sure you have everything installed correctly:

```
rails -v
# Rails 4.2.0
```

If you get a different result for some reason, it means your environment may not be setup properly.

Setting Up MySQL

Rails ships with sqlite3 as the default database. Chances are you won't want to use it because it's stored as a simple file on disk. You'll probably want something more robust like MySQL or PostgreSQL.

There is a lot of documentation on both, so you can just pick one that seems like you'll be more comfortable with. If you're coming from PHP, you may already be familiar with MySQL. If you're new to databases, I'd suggest skipping to setting up PostgreSQL.

You can install MySQL server and client from the packages in the Ubuntu repository. As part of the installation process, you'll set the password for the root user. This information will go into your Rails app's `database.yml` file in the future.

```
sudo apt-get install mysql-server mysql-client libmysqlclient-dev
```

Installing the `libmysqlclient-dev` gives you the necessary files to compile the `mysql2` gem which is what Rails will use to connect to MySQL when you setup your Rails app.

When you're finished, you can skip to the Final Steps.

Setting Up PostgreSQL

For PostgreSQL, we're going to add a new repository to easily install a recent version of Postgres 9.3.

```
sudo sh -c "echo 'deb http://apt.postgresql.org/pub/repos/apt/ precise-pgdg
wget --quiet -O - http://apt.postgresql.org/pub/repos/apt/ACCC4CF8.asc | su
sudo apt-get update
sudo apt-get install postgresql-common
sudo apt-get install postgresql-9.3 libpq-dev
```

The postgres installation doesn't setup a user for you, so you'll need to follow these steps to create a user with permission to create databases. Feel free to replace `chris` with your username.

```
sudo -u postgres createuser chris -s

# If you would like to set a password for the user, you can do the followin
sudo -u postgres psql
postgres=# \password chris
```

Final Steps

And now for the moment of truth. Let's create your first Rails application:

```
#### If you want to use SQLite (not recommended)
rails new myapp

#### If you want to use MySQL
rails new myapp -d mysql

#### If you want to use Postgres
# Note that this will expect a postgres user with the same username
# as your app, you may need to edit config/database.yml to match the
# user you created earlier
rails new myapp -d postgresql

# Move into the application directory
cd myapp

# If you setup MySQL or Postgres with a username/password, modify the
# config/database.yml file to contain the username/password that you specif

# Create the database
rake db:create

rails server
```

You can now visit http://localhost:3000 (http://localhost:3000) to view your new website!

Now that you've got your machine setup, it's time to start building some Rails applications.

If you received an error that said `Access denied for user 'root'@'localhost' (using password: NO)` then you need to update your config/database.yml file to match the database username and password.

Tweet 75 +1 23 Share 4 23

We're sharing everything we know about how to write great quality code

Join the GoRails mailing list to **learn more about what great code looks like, why it is great, and how you can write great code yourself** that you will enjoy working with.

First Name

Last Name

Email Address

Send Me Lessons On Rails

Follow @excid3 2,163 followers Tweet 44