DO NOT READ THIS FILE ON GITHUB, GUIDES ARE PUBLISHED ON https://guides.rubyonrails.org.

Development Dependencies Install

This guide covers how to set up an environment for Ruby on Rails core development.

After reading this guide, you will know:

- How to set up your machine for Rails development

Other Ways to Set Up Your Environment

If you don't want to set up Rails for development on your local machine you can use Codespaces, the VS Code Remote Plugin, or rails-dev-box. Learn more about these options here.

Local Development

If you want to develop Ruby on Rails locally on your machine see the steps below.

Install Git

Ruby on Rails uses Git for source code control. The Git homepage has installation instructions. There are a variety of resources online that will help you get familiar with Git.

Clone the Ruby on Rails Repository

Navigate to the folder where you want to download the Ruby on Rails source code (it will create its own `rails` subdirectory) and run:

```
$ git clone https://github.com/rails/rails.git
$ cd rails
```

Install Additional Tools and Services

Some Rails tests depend on additional tools that you need to install before running those specific tests.

Here's the list of each gems' additional dependencies:

- Action Cable depends on Redis
- Active Record depends on SQLite3, MySQL and PostgreSQL
- Active Storage depends on Yarn (additionally Yarn depends on Node.js), ImageMagick, FFmpeg, muPDF, and on macOS also XQuartz and Poppler.
- Active Support depends on memcached and Redis
- Railties depend on a JavaScript runtime environment, such as having Node.js installed.

Install all the services you need to properly test the full gem you'll be making changes to. How to install these services for macOS, Ubuntu, Fedora/CentOS, and FreeBSD are detailed below.

NOTE: Redis' documentation discourages installations with package managers as those are usually outdated. Installing from source and bringing the server up is straight forward and well documented on Redis' documentation.

NOTE: Active Record tests *must* pass for at least MySQL, PostgreSQL, and SQLite3. Your patch will be rejected if tested against a single adapter, unless the change and tests are adapter specific.

Below you can find instructions on how to install all of the additional tools for different operating systems.

macOS

On macOS you can use Homebrew to install all of the additional tools.

To install all run:

```
$ brew bundle
```

You'll also need to start each of the installed services. To list all available services run:

```
$ brew services list
```

You can then start each of the services one by one like this:

```
$ brew services start mysql
```

Replace `mysql` with the name of the service you want to start.

Potential Issues

This section details some of the potential issues you may run into with native extensions on macOS, particularly when bundling the mysql2 gem in local development. This documentation is subject to change and may be incorrect as Apple makes changes to the developer environment on Rails.

In order to compile the `mysql2` gem on macOS you will need the following:

1. `openssl@1.1` installed (not `openssl@3`)
2. Ruby compiled with `openssl@1.1`
3. Set compiler flags in the bundle config for `mysql2`.

If both `openssl@1.1` and `openssl@3` are installed you will need to tell Ruby to use `openssl@1.1` in order for Rails to bundle `mysql2`.

In your `.bash_profile` set the `PATH` and `RUBY_CONFIGURE_OPTS` to point to `openssl@1.1`:

```
export PATH="/usr/local/opt/openssl@1.1/bin:$PATH"
export RUBY_CONFIGURE_OPTS="--with-openssl-dir=$(brew --prefix openssl@1.1)"
```

In your `~/.bundle/config` set the following for `mysql2`. Be sure to delete any other entries for `BUNDLE_BUILD__MYSQL2`:

```
BUNDLE_BUILD__MYSQL2: "--with-ldflags=-L/usr/local/opt/openssl@1.1/lib --with-
cppflags=-L/usr/local/opt/openssl@1.1/include"
```

By setting these flags before installing Ruby and bundling Rails you should be able to get your local macOS development environment working.

Ubuntu

To install all run:

```
$ sudo apt-get update
$ sudo apt-get install sqlite3 libsqlite3-dev mysql-server libmysqlclient-dev
postgresql postgresql-client postgresql-contrib libpq-dev redis-server memcached
imagemagick ffmpeg mupdf mupdf-tools libxml2-dev

# Install Yarn
$ curl -sS https://dl.yarnpkg.com/debian/pubkey.gpg | sudo apt-key add -
$ echo "deb https://dl.yarnpkg.com/debian/ stable main" | sudo tee
/etc/apt/sources.list.d/yarn.list
$ sudo apt-get install yarn
```

Fedora or CentOS

To install all run:

```
$ sudo dnf install sqlite-devel sqlite-libs mysql-server mysql-devel postgresql-
server postgresql-devel redis memcached imagemagick ffmpeg mupdf libxml2-devel

# Install Yarn
# Use this command if you do not have Node.js installed
$ curl --silent --location https://rpm.nodesource.com/setup_8.x | sudo bash -
# If you have Node.js installed, use this command instead
$ curl --silent --location https://dl.yarnpkg.com/rpm/yarn.repo | sudo tee
/etc/yum.repos.d/yarn.repo
$ sudo dnf install yarn
```

Arch Linux

To install all run:

```
$ sudo pacman -S sqlite mariadb libmariadbclient mariadb-clients postgresql
postgresql-libs redis memcached imagemagick ffmpeg mupdf mupdf-tools poppler yarn
libxml2
$ sudo mariadb-install-db --user=mysql --basedir=/usr --datadir=/var/lib/mysql
$ sudo systemctl start redis mariadb memcached
```

NOTE: If you are running Arch Linux, MySQL isn't supported anymore so you will need to use MariaDB instead (see this announcement).

FreeBSD

To install all run:

```
$ pkg install sqlite3 mysql80-client mysql80-server postgresql11-client
postgresql11-server memcached imagemagick ffmpeg mupdf yarn libxml2
# portmaster databases/redis
```

Or install everything through ports (these packages are located under the `databases` folder).

NOTE: If you run into problems during the installation of MySQL, please see the MySQL documentation.

Database Configuration

There are couple of additional steps required to configure database engines required for running Active Record tests.

PostgreSQL's authentication works differently. To set up the development environment with your development account, on Linux or BSD, you just have to run:

```
$ sudo -u postgres createuser --superuser $USER
```

and for macOS:

```
$ createuser --superuser $USER
```

MySQL will create the users when the databases are created. The task assumes your user is `root` with no password.

Then, you need to create the test databases for both MySQL and PostgreSQL with:

```
$ cd activerecord
$ bundle exec rake db:create
```

You can also create test databases for each database engine separately:

```
$ cd activerecord
$ bundle exec rake db:mysql:build
$ bundle exec rake db:postgresql:build
```

and you can drop the databases using:

```
$ cd activerecord
$ bundle exec rake db:drop
```

NOTE: Using the Rake task to create the test databases ensures they have the correct character set and collation.

If you're using another database, check the file `activerecord/test/config.yml` or `activerecord/test/config.example.yml` for default connection information. You can edit `activerecord/test/config.yml` to provide different credentials on your machine, but you should not push any of those changes back to Rails.

Install JavaScript dependencies

If you installed Yarn, you will need to install the JavaScript dependencies:

```
$ yarn install
```

Installing gem dependencies

Gems are installed with Bundler which ships by default with Ruby.

To install the Gemfile for Rails run:

```
$ bundle install
```

If you don't need to run Active Record tests you can run:

```
$ bundle install --without db
```

Contribute to Rails

After you've set up everything, read how you can start contributing.