Step by step process for setting up a development environment for Rise

1. Install Virtual Box from here: <https://www.virtualbox.org/>
2. Install Vagrant: <https://www.vagrantup.com/>.
3. Go to <https://puphpet.com/> and drag the included riseconfig.yaml file into the main window. (If you wish to adjust memory/cpu usage, go to the “locally” tab under the “deploy target” accordion menu.
4. Click create archive, and download the .zip file. Extract it to wherever you want the virtual box information stored.
5. (Caution: this step will take about 10 minutes. Do not interrupt vagrant up!) In your command line of choice, navigate to this folder (which should have been given a random string of letters and numbers as a name, and should have a file called Vagrantfile) and run the command:

**$ vagrant up**

1. Add this line (a) to your hosts file (on the host machine – your actual machine):
   1. “192.168.56.101 rise.dev [www.rise.dev](http://www.rise.dev)”
   2. Hosts file is located at **/etc/hosts** on OSX.

1. Log in to your vagrant box with: **$ vagrant ssh**

1. Clone the Rise repo in any directory (the document root doesn’t matter, because we are using rails server)
2. Create a database and edit the config/database.yml accordingly. The username and password on this dev environment for mysql is “root” and “123”. For anyone not familiar/rusty with databases, you can do these steps:
   1. Access mysql by typing: **$** **mysql -u root -p**
   2. When it asks for the password, type **123**
   3. Create the new db: > **CREATE DATABASE rise;**
   4. Get out of mysql with: > **EXIT;**
3. Cd to the /var/www/rise directory and execute the following:

**$ bundle install**

**$ rake db:migrate**

**$ rake db:seed**

1. To start the server, run **$ rails server**
2. Connect to it via rise.dev:3000
3. When you are done working on it, you can exit the vagrant box with

**$ exit**

and close down vagrant cleanly with

**$ vagrant halt**

On your machine, the folder “rise” within the VM folder will be synchronized with the VM, so you shouldn’t have to be SSH’d to edit files.

Troubleshooting:

<https://www.vagrantup.com/docs/getting-started/up.html>

**bundle install does not work.**

Any number of things can go wrong here, and it depends on what gem it asks you to install manually. If running the suggested command does not work, these libraries might be missing:

libmysqlclient-dev

libpq-dev

**$ rake db:migrate**

**rake aborted! cannot load such file -- unicorn/version**

I fixed this problem by uninstalling the unicorn gem and rerunning bundle install:

**$ gem uninstall unicorn**

**$ bundle install**

**$ gem update**

Another fix that worked was using:

**$ bundle exec gem uninstall unicorn**

**$ bundle install**

**$ rake db:migrate**

**rake aborted!**

**No such file or directory - /var/www/rise/config/database.yml**

Everyone’s database.yml file can be different, so it is not included in the repository.

Duplicate the file **/var/www/rise/config/database\_example.yml** file to a new file in the same directory: **/var/www/rise/config/database.yml.**

Next, in **database.yml**, fill in the database credentials you set up in step 9:

database: rise

username: root

password: 123

**rails server gets stuck up on an error message (I forgot what it says, if someone runs into it please put it here.)**

If the error message is about axlsx it’s probably because you forgot to run bundle install, refer to step 9.

You have not updated your database information yet. refer to step 8.

**DB seed does not work (Could not find user with id=4).**

The current fix is to comment out lines 126-128 from db/seeds\_DAL\_2015.rb so that it no longer checks for User.find(4)

**rise.dev:3000 shows an error about dashboard record missing.**

The database has not been seeded yet, refer to step 10.

**rise.dev:3000 does not respond.**

This is possibly due to several issues:

1. The VM is not powered on.
2. rails server is not running.
3. rails server is not working: (see rails server gets stuck on error message)
4. port 3000 is closed

Solutions:

1. Turn on the VM using “vagrant up” in the according directory
2. ssh into the VM and run “rails server” in the project directory.
3. See *rails server gets stuck on error message*
4. nmap rise.dev:3000 to see if it is open, closed, or filtered.
   1. If it’s filtered or closed check <https://help.ubuntu.com/community/UFW>
   2. If it’s open, try to use curl (from the VM) on 127.0.0.1:3000 to see if there is a response, if not, see if there is an issue with rails server. If so, check <https://help.ubuntu.com/community/UFW>

**Text File is Busy Error Mysql2(on windows):**

Gem::Ext::BuildError: ERROR: Failed to build gem native extension.  
  
/home/vagrant/.rbenv/versions/2.1.1/bin/ruby extconf.rb  
checking for rb\_thread\_blocking\_region()... yes  
checking for sys/select.h... yes  
checking for poll.h... yes  
checking for sys/epoll.h... yes  
checking for sys/event.h... no  
checking for port.h... no  
checking for sys/resource.h... yes  
creating Makefile  
  
Text file busy @ unlink\_internal - ./siteconf20140504-1398-lxfrfm.rb

## **What's causing the problem?**

By default, bundler installs gems into a directory called vendor/bundle local to your project. Consequently, if your current project directory is within a shared folder between your VM and your host, bundler will attempt to install gems and native extensions on this shared folder.

## **How do I fix it?**

Thankfully there's a simple solution that enables us to keep our ruby files on the VM shared folder and keep the isolation provided by bundler/rbenv. We simply need to tell bundler to install gems in a directory that's local to the VM (not on a shared folder). We can achieve this by altering the bundle config file for the project.

### **Step 1**

Since I primarily work with Python, I'm used to the virtualenv ecosystem where all virtual environments for a project have directories within a~/.env directory. To mirror that, let's create a directory called .bundles

$ mkdir ~/.bundles

Go ahead and cd to that directory and get the absolute path via pwd. We'll need that next.

$ cd ~/.bundles  
$ pwd  
/home/vagrant/.bundles

### **Step 2**

Now let's move back to your ruby project directory. If you have a .bundle/config file local to your project, open it in your favorite text editor. Otherwise, create both the .bundle directory and a file called config within that directory. (Note: The .bundle directory should be at the same level as your project Gemfile).

Append the following to the file:

---  
BUNDLE\_PATH: "/home/vagrant/.bundles/YOUR\_PROJECT\_NAME\_HERE"  
BUNDLE\_DISABLE\_SHARED\_GEMS: '1'

### **Step 3**

Now we can finally execute

$ bundle install

Upon completion, bundler will indicate where the gems were installed:

(vagrant)/vagrant/ruby/projects/eztemp.me$ bundle  
Using timers 1.1.0  
Using celluloid 0.15.2  
Using nio4r 1.0.0  
Using celluloid-io 0.15.0  
Using ffi 1.9.3  
Using rb-fsevent 0.9.4  
Using rb-inotify 0.9.4  
Using listen 2.7.4  
Using rack 1.5.2  
Using rack-protection 1.5.3  
Using rerun 0.9.0  
Using tilt 1.4.1  
Using sinatra 1.4.5  
Using bundler 1.6.2  
Your bundle is complete!  
It was installed into /home/vagrant/.bundles/eztemp

Ensure this directory is the one you specified.