# **Installation Plan**

# Overview

Steps based on this guide and this guide.

# **Steps (BlueHost specific):**

- 1. Get BlueHost credentials
- 2. Create subdomain (here named scheduling)
- 3. Click Subdomain button in cPanel



5. Fill out subdomain information

4.



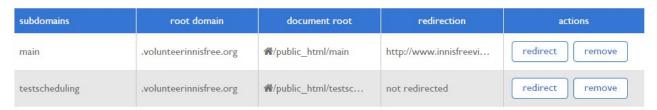


#### Subdomains

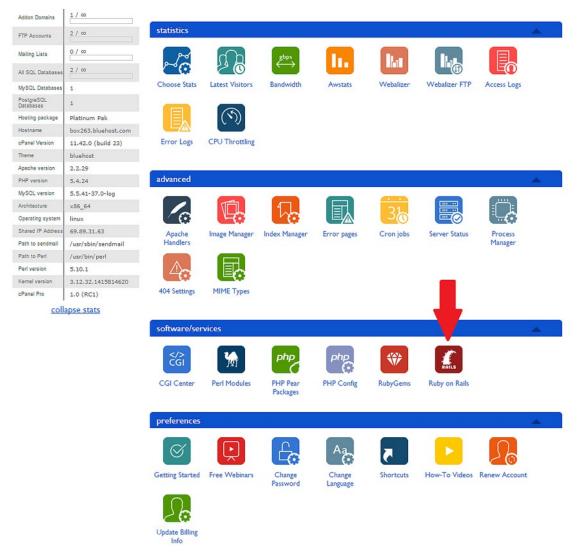
Subdomains are URLs for different sections of your website, which have an added *prefix* to your main domain name. Subdomains are relative to your accounts home directory. The # icon signifies your home directory.



#### **Existing Subdomains**



- 6.
- 7. Create Ruby on Rails application through cPanel
- 8. Click Ruby on Rails button in cPanel



10. Fill out application information

9.

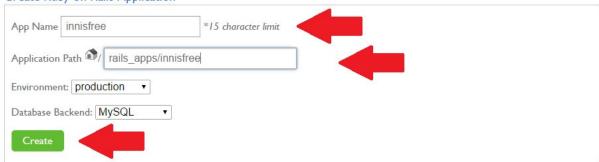




#### Manage Ruby on Rails Applications

Ruby on Rails Applications are based on the Rails framework. Rails applications must be run like any other application. After creating an application, you will need to populate it with your code. Then, you can choose to start or stop the application and even to load the application every time the server reboots.

#### Create Ruby on Rails Application



#### Available Ruby on Rails Applications

APP NAME	Ратн	Production	DELETE

Please Note: Bluehost provides Phusion Passenger. As of Ruby on Rails 3, running applications through fastcgi is no longer supported. We also recommend updating any applications that may be using Mongrel (or similar) so that they can take advantage of Phusion Passenger.

← Go Back

- 11.
- 12. Create a MySQL database for the scheduling application, and add a user to it
- 13. Using BlueHost's File Manager tool, copy zipped code to BlueHost (place it in the home directory for now)
- 14. Alternatively, you can skip this step and pull the code from GitHub later
- 15. SSH to BlueHost server using putty or a similar tool
- 16. Change to the rails\_apps directory: cd ~/rails\_apps
- 17. Install Rails: gem install rails -v 4.1.1 --no-rdoc --no-ri
- 18. Install Rake: gem install rake -v '10.4.2' --no-rdoc --no-ri
- 19. Edit .bash.rc file as described in the first guide
- 20. Unzip application code: "`unzip ~/innisfree.zip""
  - Type A when prompted to overwrite all common files
  - If you chose to instead pull the code from GitHub, do that now: git clone https://github.com/uva-slp/innisfree
- 21. cd innisfree
- 22. Uncomment line containing gem 'therubyracer' in Gemfile
- 23. Install missing gems bundle install --path vendor/bundle
- 24. Create **app** .htaccess file (e.g. ~/rails\_app/innisfree/public/.htaccess) as described in the first guide
  - Change the second to last line to SetEnv GEM\_HOME /home2/voluntf8/ruby/gems
- 25. Delete the subdomain directory: rm -rf ~/public\_html/scheduling

- 26. Link subdomain and application directory as described in the first guide
  - Create symbolic link from domain to app: ln -s
     ~/rails\_apps/innisfree/public ~/public\_html/scheduling
- 27. Create config/database.yml from config/database.yml.template (in the Rails application directory) so production entry points to BlueHost SQL server:

production:

adapter: mysql2

pool: 5
port: 3306

database: \$DATABASE\_NAME
username: \$USER\_NAME
password: \$PASSWORD
host: localhost

28. Run rake db:migrate

29. Delete the application.css file, if it exists: rm

~/rails\_app/innisfree/app/assets/stylesheets/application.css

## **Generic instructions**

- 1. Install Apache2
- 2. Install Ruby (version 1.9.3 is what we deployed on, 2.1.5 has been tested to work)
- 3. Install Rails (version 4.1.1 is what we deployed on, 4.x should work, but has not been tested)
- 4. Install passenger: gem install passenger
- 5. Passenger is an web & application server that integrates nicely with Apache and Nginx to run Rails apps
- 6. Install the passenger module for apache: passenger-install-apache2-module
- 7. You may need to enable this module sudo a2enmod passenger and restart apache sudo service apache2 restart
- 8. Clone the application repository from Github
- 9. Follow steps 6.vi through 6.x from above (sorry about the mix of Arabic and Roman numerals...)
- 10. Link the public directory of the application to the directory Apache is serving out of:

  ln -s ./public /opt/bitnami/apache2/htdocs/scheduling
- 11. This is by default /\*installdir\*/apache2/htdocs/application where *installdir* is the directory Apache was installed to. For bitnami, this looks to be /opt/bitnami/apache2/htdocs
- 12. Your application should now be accessible from the server's IP address (or DNS, if that's your thing)

13.

# **Updating your installation**

Once you've got your code up and running, you may make some changes and want to updated your deployed code. 1. SSH to the server using putty or a similar tool 2. Change to the innisfree application directory: cd ~/rails\_apps/innisfree 3. Pull from the GitHub hosted repository: "'git pull"' - This will prompt you for your GitHub username and password. Once you enter that it'll copy all the updates to your server 4. Run the following command to restart Passenger and update the code: "touch tmp/restart.txt"

### Sample files

### Sample config/database.yml

```
# SQLite version 3.x
    gem install sqlite3
#
    Ensure the SQLite 3 gem is defined in your Gemfile
#
#
    gem 'sqlite3'
default: &default
development:
  adapter: mysql2
  pool: 5
  port: 3306
  database: $DEV_DATABASE_NAME
  username: $USER_NAME
  password: $PASSWORD
  host: localhost
# Warning: The database defined as "test" will be erased and
# re-generated from your development database when you run "rake".
# Do not set this db to the same as development or production.
test:
  adapter: sqlite3
  pool: 5
  timeout: 5000
  database: db/test.sqlite3
production:
  adapter: mysql2
  pool: 5
  port: 3306
  database: $DATABASE NAME
  username: $USER_NAME
  password: $PASSWORD
  host: localhost
```

### Sample public/.htaccess file

```
<IfModule mod_passenger.c>
  Options -MultiViews
  PassengerResolveSymlinksInDocumentRoot on
  #Set this to whatever environment (development, production) you'll be runn
  RailsEnv production
  RackBaseURI /
  SetEnv GEM_HOME /home2/voluntf8/ruby/gems
</IfModule>
```