

BitNami RubyStack 2.0-3

Quick Start Guide

BitNami RubyStack 2.0-3

Release 2.0-3 2010-03-09

Copyright © 2007 BitNami

<http://bitnami.org>

All rights reserved.

This product and its documentation are protected by copyright. The information in this document is provided on an "as is" basis, without warranty. Although every precaution has been taken in the preparation of this document, the authors will not have any liability to any person or entity with respect to any loss or damage caused or alleged to be caused directly or indirectly by the information contained in this work.

Trademark names may appear in this document. All registered and unregistered trademarks in this document are the sole property of their respective owners.

Acknowledgements

BitNami RubyStack is based on a number of open source components:

Ruby. The interpreted scripting language for quick and easy object-oriented programming.

<http://www.ruby-lang.org/>.

Ruby Enterprise Edition, a server-oriented friendly branch of Ruby.

<http://www.rubyenterpiseedition.com>

Rails. Together with Ruby (RoR) is a open-source web application framework that's optimized for programmers happiness and sustainable productivity.

<http://www.rubyonrails.org/>.

MySQL. The world's leading open source database.

<http://www.mysql.com>

The Apache HTTP server, developed by The Apache Software Foundation.

<http://www.apache.org/>.

PHP. The widely used scripting language.

<http://www.php.net>.

The open source browser-based MySQL management tool phpMyAdmin.

<http://www.phpmyadmin.net>

Git. Open source, distributed version control system.

<http://git-scm.com>

Subversion. The open-source version control system.

<http://www.subversion.tigris.org>

RubyGems. The premier ruby packaging system

<http://rubygems.org/>

The Mongrel web server.

<http://mongrel.rubyforge.org/>.

Phusion Passenger ? a.k.a. mod_rails or mod Rack ? makes deployment of Ruby web applications a breeze.

<http://www.modrails.com/>.

Nginx. High-performance HTTP server and reverse proxy.

<http://nginx.org/>.

Sphinx. Full-text search engine.

<http://www.sphinxsearch.com/>.

Memcached. High-performance, distributed memory object caching system.

<http://www.memcached.org/>.

The OpenSSL library for encrypting communications.

<http://www.openssl.org/>.

The zlib data-compression library.

<http://www.zlib.net>

Rmagick gem for ImageMagick.

<http://rmagick.rubyforge.org/>

ImageMagick image processor.

<http://www.imagemagick.org>

The libiconv library for multiple character encoding.

<http://www.gnu.org/software/libiconv/>

jpegsrc. Software developed by the The Independent JPEG Group.

<http://www.ijg.org>

The gd graphics library.

<http://www.boutell.com/gd/>

You can find the individual licenses for the above projects as part of the installation.

BitNami RubyStack Overview

The BitNami Project was created to help spread the adoption of freely available, high quality, open source web applications. BitNami aims to make it easier than ever to discover, download and install open source software such as document and content management systems, wikis and blogging software. You can learn more about BitNami at <http://bitnami.org>.

The **BitNami RubyStack** is an installer that greatly simplifies the installation of Ruby on Rails and runtime dependencies. It includes ready-to-run versions of Ruby, Rails, MySQL and Subversion. RubyStack is distributed for free under the Apache 2.0 license. Please see the appendix for the specific licenses of all open source components included. You can learn more about BitNami Stacks at <http://bitnami.org/stacks/>.

Components

BitNami RubyStack includes **Apache 2.2.14**, **ImageMagick 6.5.8**, **MySQL 5.1.30**, **Ruby 1.8.7-2010.01**, **Rails 2.3.5**, **RubyGems 1.3.5**, **PHP 5.2.13**, **phpMyAdmin 2.11.9.4**, **Git 1.6.5.2**, **Subversion 1.6.6**, **nginx 0.7.64** and more requirement libraries.

Ruby is a dynamic, open source programming language with a focus on simplicity and productivity. It has an elegant syntax that is natural to read and easy to write. You can learn more about Ruby at <http://www.ruby-lang.org>.

Ruby Enterprise Edition is a server-oriented friendly branch of Ruby which includes various enhancements. You can learn more about Ruby enterprise edition at <http://www.rubyenterpiseedition.com>

Ruby on Rails is a full-stack MVC framework for database-backed web applications that's optimized for programmer happiness and sustainable productivity. It lets you write beautiful code by favoring convention over configuration. You can learn more about Ruby at <http://www.rubyonrails.org>.

Apache is the most popular HTTP server on the Internet. It provides a secure, efficient and extensible web platform. It is maintained by the Apache Software Foundation. You can find more information about Apache at <http://www.apache.org>.

MySQL is the world's most popular open source database. It is a relational database management system that combines speed, reliability and ease of use. It is developed and maintained by MySQL AB. You can find more information about MySQL at <http://www.mysql.com>.

PHP is a web development scripting language that can be embedded into HTML. Powerful and easy to use, it is the most popular Apache module. It is maintained by the PHP Group. You can find more information about PHP at <http://www.php.net>.

phpMyAdmin is a tool written in PHP intended to handle the administration of MySQL through a web interface. It allows you to manage anything from a single database to a complete MySQL server. You can find more information about phpMyAdmin at <http://www.phpmyadmin.net>.

Git is a free & open source, distributed version control system designed to handle everything from small to very large projects with speed and efficiency. You can find more information about Git at <http://git-scm.com/>.

Subversion The goal of the Subversion project is to build a version control system that is a compelling replacement

for CVS in the open source community. The software is released under an Apache/BSD-style open source license. <http://subversion.tigris.org/>.

Phusion Passenger ? a.k.a. mod_rails or mod_rack ? makes deployment of Ruby web applications, such as those built on the revolutionary Ruby on Rails web framework, a breeze. It follows the usual Ruby on Rails conventions, such as ?Don?t-Repeat-Yourself?. You can find more information about Passenger at <http://www.modrails.com/>.

Nginx is a free, open-source, high-performance HTTP server and reverse proxy, as well as an IMAP/POP3 proxy server. It is known for its high performance, stability, rich feature set, simple configuration, and low resource consumption. You can find more information about Nginx at <http://nginx.org/>.

Sphinx is a full-text search engine, meant to provide fast, size-efficient and relevant fulltext search functions to other applications. Sphinx was specially designed to integrate well with SQL databases and scripting languages. You can find more information about Sphinx at <http://www.sphinxsearch.com/>.

Memcached free & open source, high-performance, distributed memory object caching system, generic in nature, but intended for use in speeding up dynamic web applications by alleviating database load. Memcached is an in-memory key-value store for small chunks of arbitrary data (strings, objects) from results of database calls, API calls, or page rendering. You can find more information about Memcached at <http://www.memcached.org/>.

Requirements

To run BitNami RubyStack you will need:

- Intel x86 or compatible processor
- Minimum of 512 MB RAM
- Minimum of 150 MB hard drive space
- A Windows operating system
- TCP/IP protocol support
- An x86 Linux operating system or
A 32-bit Windows operating system such as Windows 2000, XP, Vista or Windows Server 2003 or
an OS X operating System (x86 or PowerPC).

Installation Guide

This section describes where to download BitNami RubyStack and the different installation modes that are available.

Downloading BitNami RubyStack

You can download the BitNami RubyStack binary file from <http://bitnami.org/stacks/>. It will be named `bitnami-rubystack-2.0-3-windows-installer.exe` for Windows or `bitnami-rubystack-2.0-3-linux-installer.bin` for Linux or `bitnami-rubystack-2.0-3-osx-x86-installer.app.zip` for OS X x86 or `bitnami-rubystack-2.0-3-osx-powerpc-installer.app.zip` for OS X PowerPC.

Installing BitNami RubyStack

To begin the installation process, double-click on the file from your Desktop environment or invoke it directly from the command line.

You will be greeted by the 'Welcome' screen. The next step is to select the installation directory.

Then you can choose to install or not phpMyAdmin to handle the administration of MySQL through a web interface. If you want a sample Rails application to be created during installation, you should select "Sample Rails application" component.

RubyStack bundles a number of components that use TCP/IP ports. In particular, the default listening port for MySQL 3306 for Mongrel 3000 and for Apache 80 on Windows and 8080 on Linux. If any of these ports are already taken, the installation will ask you to enter a different port number.

The next screen will prompt you for data necessary to create the initial admin user:

MySQL root password: Here you can set the password that will be used when performing operations such as creating or deleting databases.

If you selected phpMyAdmin component, it will be configured with the user name 'administrator' and the password provided for MySQL root user. Once installation is completed, you can access at `http://127.0.0.1:80/phpmyadmin` on Windows and `http://127.0.0.1:8080/phpmyadmin` on Linux.

If you have chosen to install a sample Rails application, then installation will ask you about the application name and database password: RubyStack will create a sample Rails application with the name selected here. It will also create the corresponding databases using that name as a prefix. Finally, the password written here will be used by your new application when accessing the databases with the application name as user name.

You are now ready to begin the installation, which will start when you press 'Next'. Once the installation process has been completed, you will see the 'Installation Finished' page. At this point, you can launch a mongrel server and a browser. In addition to that, a report summarizing the installation options will be shown. The report is located at the root of the installation directory and will also be accessible from a link in the Start Menu. It contains the passwords and usernames you provided during installation, so you may want to store it in an alternate location if other people are able to access that file.

If you receive an error message during installation, please refer to the Troubleshooting section.

Directory Structure

The installation process will create several subfolders under the main installation directory:

- `mysql/`: MySQL Database.
- `sqlite/`: SQLite Database.
- `ruby/`: Ruby and ruby gems files.
- `img/`: Additional image files and icons.
- `imagemagick/`: ImageMagick image processor.
- `subversion/`: Subversion files.
- `scripts/`: Script with environment vars.
- `licenses/`: Component licenses.
- `apache2/`: Apache2 files.
- `php/`: PHP files.
- `apps/phpMyAdmin/`: phpMyAdmin files.
- `nginx/`: nginx files.
- `memcached/`: Memcached files.
- `sphinx/`: Sphinx files.
- `git/`: git files.

Uninstalling BitNami RubyStack

As part of the installation, an uninstall program will be created at the installation root directory and a link placed in the Start Menu. The uninstaller will stop the currently running servers and delete any files and registry entries created during installation. The uninstaller will not delete any files or databases that were created by the end user.

Ruby on Rails

On this section we are going to assume that Rubystack has been installed at `C:\Program Files\Bitnami Rubystack\` on Windows, `/home/user/rubystack` on Linux or `/Applications/rubystack/projects/` on OS X and your Rails application name is `rubystack`.

This part of the tutorial has been written following a Questions and Answers model.

- **How do I start doing things with RubyStack?**

The first thing you must do is to run a Rails Environment by selecting:

```
Start -> BitNami RubyStack -> Use Ruby on Windows or
/home/user/rubystack/rubyconsole on Linux.
```

That will open a console window with a properly setup environment where you can then issue commands.

- **How do I launch or stop MySQL and Apache services?**

On Windows, open the Rails Environment and then run the `servicerun.bat` script to start or stop the services. In order to start services you must run:

```
servicerun.bat START
```

and to stop them:

```
servicerun.bat STOP
```

Be sure you are in the main directory of the RubyStack installation (`C:\Program Files\BitNami RubyStack\`).

On Linux, you can start/stop Apache, MySQL and Subversion server with `ctlscript.sh`. You can type

```
./ctlscript.sh help to see all options:
```

```
usage: ./ctlscript.sh help
```

```
./ctlscript.sh (start|stop|restart)
./ctlscript.sh (start|stop|restart) mysql
./ctlscript.sh (start|stop|restart) apache
./ctlscript.sh (start|stop|restart) subversion
```

```
help - this screen
start - start the service(s)
stop - stop the service(s)
restart - restart or start the service(s)
```

- **How do I launch my web application?**

Launch "Use Ruby" from the Start Menu, go to your rails application's directory and start the server. For example, if you did not change its name, the default Rails application is named `rubystack` and is located under `C:\Documents and Settings\User\BitNami RubyStack projects` on Windows and `/home/user/rubystack/projects/` on Linux or `/Applications/rubystack/projects/` on OS X, which is also the location that the "Use Ruby" console starts at.

```
cd rubystack
```

```
ruby script/server
```

- **How can I access my web application?**

Ensure that you are running the application server (see "How do I launch my web application?") and then point your web browser to <http://localhost:3000/>.

- **How can I stop my web server?**

Just close the window were the server is running.

- **How can I create another rails web application?**

Run "rails" followed by the name of your desired application inside the Rails Environment:

```
rails my_new_app
```

That will create a directory called `my_new_app` with the corresponding files inside. The next step is to create databases for that application, which is explained below.

- **How can I create the corresponding databases for my new web application?**

Open a Rails Environment window and connect to the mysql database:

```
"C:\Program Files\BitNami RubyStack\mysql\bin\mysql.exe" -u root --port=your-port-  
here on windows or /home/user/rubystack/mysql/bin/mysql -u root --port=your-port-  
here on Linux.
```

From there you can run the following.

```
CREATE DATABASE IF NOT EXISTS my_new_app_production;  
CREATE DATABASE IF NOT EXISTS my_new_app_development;  
CREATE DATABASE IF NOT EXISTS my_new_app_test;  
GRANT ALL PRIVILEGES on my_new_app_test.* to 'rubystack'@'localhost' identified  
by 'rubystack';  
GRANT ALL PRIVILEGES on my_new_app_production.* to 'rubystack'@'localhost';  
GRANT ALL PRIVILEGES on my_new_app_development.* to 'rubystack'@'localhost';  
flush privileges;
```

- **How can I configure my new Rails application to use the previously created databases?**

You need to create a file called "database.yml" inside of your web application config directory. In order to do that, open "notepad" or your preferred text editor and paste the following on Windows:

```

development:
  adapter: mysql
  database: my_new_app_development
  username: rubystack
  password: your-password-here
  host: localhost
  port: ${mysql_port}

test:
  adapter: mysql
  database: my_new_app_test
  username: rubystack
  password: your-password-here
  port: ${mysql_port}

production:
  adapter: mysql
  database: my_new_app_production
  username: rubystack
  password: your-password-here
  port: ${mysql_port}

```

Save it to "`${rails_application_dir}\config\database.yml`". On Linux and OS X you have to replace `port: ${mysql_port}` with `socket: ${installdir}/mysql/tmp/mysql.sock`

- **How can I create a mongrel cluster?**

You may configure Apache or Nginx to act as a proxy balancer between different mongrel processes. Those Mongrel instances should be using specific ports so the web server is able to redirect the requests to them.

For example, you may start two mongrels in ports 3001, 3002 manually as follows from the Ruby console:

```

cd rubystack
ruby script/server -p 3001 -d
ruby script/server -p 3002 -d

```

On Linux and OS X you can use `mongrel_cluster` plugin to start/restart/stop a group of mongrels easily. You can create a mongrel cluster configuration file running:

```

mongrel_rails cluster::configure -e production -p 3001 -N 2 -C
${rails_application_dir}/config/mongrel_cluster.yml

```

This will create a cluster with two mongrels starting at port 3001 storing the settings in `mongrel_cluster.yml`. Then, to start the cluster you can type:

```

cd ${rails_application_dir}
mongrel_rails cluster::start

```

- **How can install mongrels as services on Windows?**

You can install mongrels as Windows services:

```

mongrel_rails service::install -N "service_name" -p "port" -c
"application_directory"

```

To start the service you can use Windows service tool or you can type:

```

net start "service_name"

```

To uninstall the service:

```
mongrel_rails service::remove -N "service_name"
```

or

```
sc delete "service_name"
```

DEPLOY RUBY ON RAILS APPLICATION ON THE STACK.

Using Apache with Mongrel

This approach consists on having multiple Mongrel processes running in different ports. Once they are running, Apache will be set up as a proxy balancer, sending the Rails requests to those ports.

In order to setup the set of Mongrels, you may configure a mongrel cluster or generate a windows service for each as it was described in previous section.

Regarding Apache, you may include the following lines to `apache2/conf/httpd.conf`:

```
ProxyPass /appname balancer://appcluster
ProxyPassReverse /appname balancer://appcluster
```

```
<Proxy balancer://appcluster>
  BalancerMember http://127.0.0.1:3001/appname
  BalancerMember http://127.0.0.1:3002/appname
  ...
</Proxy>
```

That way, each time a Rails request is received it will be sent to the mongrels running in ports 3001, 3002..

After restarting Apache the application should be accessible at `http://domain:port/appname`.

Using Apache with Passenger

If you are using Linux or OS X, you can deploy the application with Phusion Passenger. In this case, Mongrel processes are not necessary.

To apply this method we just need to include a virtual host which `DocumentRoot` points to 'public' directory in Rails application.

```
<VirtualHost *:80>
ServerName myapplication.com
DocumentRoot ${rails_app_dir}/public
<Directory ${rails_app_dir}/public>
  Allow from all
  Options -MultiViews
</Directory>
</VirtualHost>
```

You should make sure that 'public' and 'config' application folders and their parent directories are readable and executable by Apache.

You can find more information about Phusion Passenger settings at <http://www.modrails.com/documentation.html>.

Using Nginx with Mongrel

This approach consists on having multiple Mongrel processes running in different ports. Once they are running, Nginx will be set up as a proxy balancer, sending the Rails requests to those ports.

In order to setup the set of Mongrels, you may configure a mongrel cluster or generate a windows service for each as

it was described in previous section.

Regarding Nginx, you may include a new location inside a new file `nginx/conf/vhosts/railsapp.conf`:

```
upstream backend {
    server 127.0.0.1:3001;
    server 127.0.0.1:3002;
    ... }

server {
    location /appname {
        proxy_redirect off;
        port_in_redirect off;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;

        if (-f $request_filename) {
            break;
        }

        if (!-f $request_filename) {
            proxy_pass http://backend;
            break;
        }
    }
}
```

That way, each time a Rails request is received it will be sent to the mongrels running in ports 3001, 3002...

Finally, we start nginx:

```
<installation directory>/nginx/sbin/nginx (linux, osx)
<installation directory>\nginx\nginx.exe (windows)
```

By default, nginx server uses port 1234 but it can be changed adding a "listen" parameter to server section. The application should be accessible at `http://domain:nginx_port/appname`.

A more detailed example can be seen in `nginx/conf/sample_domain_host.conf`.

Using Nginx with Passenger

Nginx has been compiled with Phusion Passenger module in Linux and OS X systems. You can use this module to easily deploy a Rails application.

In order to do so, you should include the options "passenger_enabled" and specify the root path and rails environment. You can do this with a new file in `nginx/conf/vhosts/apname.conf` containing:

```
server {
    ...
    root ${rails_appdir}/public; # Path to rails application public dir
    passenger_enabled on;
    rails_env production;
    ...
}
```

In this case, it is not necessary to start additional processes apart from Nginx. After restarting Nginx, the application

will be accessible at `http://domain:nginx_port/`.

It is also possible to deploy a set of applications to different sub URIs. If you want to use this approach, you should create symbolic links to each application public directory, inside a root path.

```
ln -s /path/to/rails_app/public /websites/phusion/railsapp
ln -s /path/to/rails_app2/public /websites/phusion/railsapp2
```

Then, include a `passenger_base_uri` option for each of them inside the server element:

```
server {
    ...
    root /websites/phusion;
    passenger_enabled on;
    passenger_base_uri /railsapp;
    passenger_base_uri /railsapp2;
    ...
}
```

After restarting Nginx, the applications will be accessible at `http://localhost:nginx_port/railsapp/`, `http://localhost:nginx_port/railsapp2/ ..`

You can find more information about Phusion Passenger settings at <http://www.modrails.com/documentation.html>.

USING RUBY-1.9

Ruby-1.9 is included into BitNami RubyStack too, in case you wish to use it you should:

- Linux, OS X: replace `scripts/setenv.sh` with `scripts/setenv19.sh` and use `rubyconsole19` instead of `rubyconsole`.
- Windows: replace `scripts\setenv.bat` with `scripts\setenv19.bat`

You can rollback to initial ruby version replacing `setenv` file with `setenv18`.

CREATING THE FIRST REPOSITORY WITH GIT.

Git is an open source version control system designed to handle very large projects with speed and efficiency.

- At first, **we need to run the rubyconsole:**

```
cd ~/rubystack-2.0-3
./rubyconsole
```

- The second step will be to **create a new empty repository**, we need to type the following:
git init

And this will create an empty repository called .git.

Now we can use this repository adding new files and committing them. For example:

```
echo "Git working over BitNami RubyStack" > test.txt
git add test.txt
git commit test.txt -m "Initial upload"
```

- The last step will be to do a **checkout for a external repository:**

```
git clone git://git.kernel.org/pub/scm/git/git.git
```

PHP

This section describes how to test your PHP installation. You can find the PHP technical documentation at <http://www.php.net/manual/en/index.php>

Testing Your Installation

The easiest way to test your PHP installation is to create a test script using the `phpinfo()` function. Open your favorite text editor and type:

```
<?php phpinfo(); ?>
```

Save the file as `phptest.php` in `C:\Program Files\BitNami RubyStack-2.0-3\apache2\htdocs\` on Windows or `/home/user/rubystack-2.0-3/apache2/htdocs` on Linux or `/Applications/rubystack-2.0-3/apache2/htdocs/`. Make sure Apache is up and running, open a browser and type `http://127.0.0.1:8080/phptest.php`. You should then see a screen showing detailed information about the PHP version you are using.

phpMyAdmin

This section describes how to access your phpMyAdmin installation. You can find more information about phpMyAdmin in http://www.phpmyadmin.net/home_page/docs.php

phpMyAdmin is an easy to use management tool for MySQL that allows you to create and drop databases, create, drop and modify tables, delete, edit and add fields, execute SQL statements, manage keys on fields, manage privileges and export data in various formats through a web-based interface.

Access to phpMyAdmin

To test your phpMyAdmin installation first make sure that your Apache and MySQL servers are up and running. You can access your phpMyAdmin installation by opening a browser and typing `http://127.0.0.1:8080/phpmyadmin`. You will then be asked for a username and password. As user name, use "**administrator**" and as password use the value specified during installation.

Initially only requests from 127.0.0.1 will be allowed access to that section. You can change this behavior editing the Apache main configuration file located at `C:\Program Files\BitNami RubyStack-2.0-3\apache2\conf\httpd.conf` on Windows or `/home/user/rubystack-2.0-3/apache2/conf/httpd.conf` on Linux or `/Applications/rubystack-2.0-3/apache2/conf/httpd.conf` on OS X.

Troubleshooting

This section describes some of the most common problems you may find when installing BitNami RubyStack.

Installer

Installer Payload Error

You may get the following error while trying to run the installer from the command line:

```
Installer payload initialization failed. This is likely due to an incomplete or corrupt
downloaded file.
```

The installer binary is not complete, likely because the file was not downloaded correctly. You will need to download the file and repeat the installation process.

Installation does not complete and hangs at the end forever.

You are probably overwriting a previous RubyStack installation and therefore a previous MySQL with a different MySQL root password. Remove or move your previous RubyStack installation and try to install RubyStack again.

MySQL

If you encounter any problems starting MySQL, the first place to look in is the "Problems and Common Errors" section of the MySQL manual, which you will find at <http://dev.mysql.com/doc/>

The following are some common problems:

Access denied when trying to connect to MySQL.

If you get an `Access Denied` message while trying to connect to MySQL, make sure you are using the correct username and password.

"Can't connect to server" message.

Make sure the MySQL daemon is up and running.

Apache

If you find any problem starting Apache, the first place you should look at is the error log file that will be created at `C:\Program Files\BitNami RubyStack-2.0-3\apache2\logs\error_log` on Windows or `/home/user/rubystack-2.0-3/apache2/logs/error_log` on Linux or `/Applications/rubystack-2.0-3/apache2/logs/error_log` on OS X. There you will likely find useful information to determine what the problem may be. For issues not covered in this Quick Start guide, please refer to the Apache documentation included as part of the installation, which is located at `apache2/htdocs/manual`.

Test page is not working

If the server seems to start correctly (i.e. you can see the `./apachectl start: httpd started` message) but you cannot see the test page when you type `http://127.0.0.1:80/` on Windows or `http://127.0.0.1:8080/` on Linux in your browser, it may be that there is already a server running in that port.