Introduction

Whether you are preparing your VPS to try out a new application, or find yourself in need of a solid and isolated Ruby installation, getting your system ready-for-work (inline with CentOS design ideologies of stability, along with its incentives of minimalism) can get you feeling a little bit lost.

In this DigitalOcean article, we are focusing on the simplest and quickest rock-solid way to get the latest Ruby interpreter (version 2.1.0) installed on a VPS running CentOS 6.5 using the Ruby Version Manager - RVM.

Glossary

1. Ruby Version Manager (RVM)

2. Understanding CentOS

3. Getting Started With Installation

1. Preparing The System
2. Downloading And Installing RVM
3. Installing Ruby 2.1.0 On CentOS 6.5 Using RVM
4. Setting Up Any Ruby Version As The Default Interpreter
5. Working With Different Ruby Installations
6. Working With RVM gemsets

Ruby Version Manager (RVM)

Ruby Version Manager, or RVM (and rvm as a command) for short, lets developers and system administrators quickly get started using Ruby and/or developing applications with a Ruby interpreter.

Not only does RVM support multiple versions of Ruby simultaneously, but also it comes with built-in tools to create and work with virtual environments called *gemsets*. With the help of RVM, it is possible to create any number of perfectly isolated - and self-contained - gemsets where dependencies, packages, and the default Ruby installation are crafted to match your needs and kept accordingly between different stages of deployment -- guaranteed to work the same way regardless of where.

RVM gemsets

The power of RVM is its ability to create fully isolated Ruby containers which act like a completely different (and a new) environment. Any application running inside the environment can access (and function) only within its reach.

Understanding CentOS

CentOS operating system is derived from RHEL - Red Hat Enterprise Linux. The target users of these distributions are usually businesses, which require their systems to be running the most stable way for a long time.

The main incentives of CentOS, therefore, is the desire for stability, which is achieved by supplying tested, stable versions of applications.

All the default applications that are shipped with CentOS remain to be used by the system (and its supportive applications such as the package manager YUM) alone. It is neither recommended nor easy to try to work with them.

That is why we are going to prepare our CentOS 6.5 running droplet with necessary tools and

continue with installing a Ruby interpreter targeted to run your applications.

Getting Started With Installation

Preparing The System

CentOS distributions are very lean. They do not come with many of the popular applications and tools that you are likely to need - and this is an intentional design choice as we have seen.

For our installations, however, we are going to need some libraries and tools (i.e. **development [related] tools**
) that are not shipped by default. Therefore, we need to get them downloaded and installed before we continue.

For this purpose we will download various development tools using YUM software groups which consist of bunch of commonly used tools (applications) bundled together, ready to download.

As the first step, in order to get necessary development tools, run the following:

```
yum groupinstall -y development
```

or;

```
yum groupinstall -y 'development tools'
```

Note: The former (shorter) version might not work on **older** distributions of CentOS.

Downloading And Installing RVM

After arming our system with tools needed for development (and deployment) of applications, such as a generic compiler, we are ready to get RVM downloaded installed.

RVM is designed from the ground up to make the whole process of getting Ruby and managing environments easy. It is no surprise that getting RVM itself is simplified as well.

In order to download and install RVM, run the following:

```
curl -L get.rvm.io | bash -s stable
```

And to create a system environment using RVM shell script:

```
source /etc/profile.d/rvm.sh
```

Installing Ruby 2.1.0 On CentOS 6.5 Using RVM

All that is needed from now on to work with Ruby 2.1.0 (or any other version), after downloading RVM and configuring a system environment is the actual installation of Ruby from source - which is to be handled by RVM.

In order to install Ruby 2.1.0 from source using RVM, run the following:

```
rvm reload
rvm install 2.1.0
```

Setting Up Any Ruby Version As The Default Interpreter

If you are working with multiple applications which are already in production, it is a highly likely scenario that at some point you will need to use a different version of Ruby for a certain application.

However, for most situations, you will probably be using the latest version as the interpreter to run all others.

One of RVM's excellent features is its ability to help you set a default Ruby version to be used generally and switch between them when necessary.

To check your current default interpreter, run the following:

```
ruby --version  # ruby command is linked to the selected version of Ruby Interpreter (i.e. 2.1.0)
```

To see all the installed Ruby versions, use the following command:

```
rvm list rubies
```

To set a Ruby version as the default, run the following:

```
# Usage: rvm use [version] --default  rvm use 2.1.0 --default
```

Working With Different Ruby Installations

To use another version for the current session, omit the --default flag:

```
# Usage: rvm use [version]  rvm use 2.1.0
```

Working With RVM gemsets

RVM gemsets consist of virtual environment at physical locations whereby all application related packages (e.g. dependencies, libraries etc.) are kept and used by a single application (i.e. your website).

Although for developers new to the concept, using gemsets (or environments) might appear like an unnecessary, cumbersome process at first. As you continue to develop and produce your application, however, the benefits soon start to become visible. Once you start using environments, both for the production and the development stages of application, it should become a little bit simpler to maintain.

In order to create a new gemset to contain a Ruby application, run the following commands:

```
# Usage: rvm gemset [create/use] [name]  # Create a new gemset using the default Ruby interpreter (2.1.0)  # Run: rvm use [version] if you wish to work with another  # Example: rvm use 2.0.0  rvm gemset create myapp    # Switch to using the new gemset called *myapp*  rvm
```

gemset use myapp

To simplify the above process, you can alternatively go with:

```
# Usage: rvm use [version]@[name] --create  rvm use 2.1.0@myapp --create
```

From this point on, all the actions you take (i.e. installing a Ruby gem) will concern your newly created environment alone. For example, installing a gem via:

```
gem install [package]
```

translates to having [package] installed inside the gemset, restricting other applications' (i.e. from other gemsets) access.

If you ever need to wipe all the gems installed, you can empty the gemset with the following command:

```
# Usage: rvm gemset empty [name]  rvm gemset empty myapp
```

Likewise, deleting a gemset can be done by using the delete argument passed to rvm:

```
# Usage: rvm gemset delete [name]  rvm gemset delete myapp
```

Note: To learn more about using RVM, you can check out our detailed tutorial on the subject by clicking here .