People using Ruby generally have a way to install different Ruby versions, manage their packages (or gems), and manage their gem dependencies.

Ruby Managers

Some platforms have Ruby pre-installed or available as a package. Most rubyists do not use these, or if they do, they only use them to bootstrap another Ruby installer or implementation. Instead rubyists tend to install a Ruby manager to install and switch between many versions of Ruby and their projects' Ruby environments.

The following are the popular Ruby environment managers:

- RVM - Installs and switches between rubies. RVM also has the concept of gemsets to isolate projects' environments completely.
- ruby-build - Only installs rubies. Use this for finer control over your rubies' installations.
- rbenv - Only switches between rubies. Used with ruby-build. Use this for finer control over how rubies load.
- chruby - Only switches between rubies. Similar in spirit to rbenv. Unopinionated about how rubies are installed.

Ruby Versions

Ruby was created by Yukihiro "Matz" Matsumoto, who remains somewhat of a BDFL, although that is changing recently. As a result, the reference implementation of Ruby is called MRI (Matz' Reference Implementation), and when you hear a Ruby version, it is referring to the release version of MRI.

The three major version of Ruby in use are:

- 2.0.0 - Released in February 2013. Most major libraries and frameworks support 2.0.0.
- 1.9.3 - Released in October 2011. This is the version most rubyists use currently. Also retired
- 1.8.7 - Ruby 1.8.7 has been retired.

The change between 1.8.7 to 1.9.x is a much larger change than 1.9.3 to 2.0.0. For instance, the 1.9 series introduced encodings and a bytecode VM. There are projects still on 1.8.7, but they are becoming a small minority, as most of the community has moved to at least 1.9.2 or 1.9.3.

Ruby Implementations

The Ruby ecosystem enjoys many different implementations of Ruby, each with unique strengths and states of compatibility. To be clear, the different implementations are written in different languages, but *they are all Ruby*. Each implementation has special hooks and extra features, but they all run normal Ruby files well. For instance, JRuby is written in Java, but you do not need to know Java to use it.

Very mature/compatible:

- MRI - Written in C, this is the reference implementation of Ruby. By definition it is 100% compatible (with itself). All other rubies maintain compatibility with MRI (see RubySpec below).
- JRuby - Written in Java and Ruby, this robust implementation is quite fast. Most importantly, JRuby's strength is JVM/Java interop, leveraging existing JVM tools, projects, and languages.
- Rubinius - Written primarily in Ruby itself with a C++ bytecode VM. Also mature and fast. Because it is implemented in Ruby itself, it exposes many VM features into rubyland.

Medium mature/compatible:

- Maglev - Built on top of Gemstone, a Smalltalk VM. Smalltalk has some impressive tooling, and this project tries to bring that into Ruby development.
- RubyMotion - Brings Ruby to iOS development.

Less mature/compatible:

- Topaz - Written in RPython (using the PyPy toolchain), Topaz is fairly young and not yet compatible. It shows promise to be a high-performance Ruby implementation.
- IronRuby - Written in C# targeting the .NET platform, work on IronRuby seems to have stopped since Microsoft pulled their support.

Ruby implementations may have their own release version numbers, but they always target a specific version of MRI for compatability. Many implementations have the ability to enter different modes (for example, 1.8 or 1.9 mode) to specify which MRI version to target.

RubySpec

Most Ruby implementations rely heavily on RubySpec. Ruby has no official specification, so the community has written executable specs in Ruby to test their implementations' compatibility with MRI.

RubyGems

RubyGems is a community-run package manager for Ruby. RubyGems ships with Ruby, so there is no need to download it separately.

Ruby packages are called "gems," and they can be hosted by the community at RubyGems.org. Each gem contains its source code and some metadata, including things like version, dependencies, author(s), and license(s).

Bundler

Bundler is a gem dependency resolver. It uses a project's Gemfile to find dependencies, and then fetches those dependencies' dependencies recursively. It does this until all dependencies are resolved and downloaded, or it will stop if a conflict has been found.

Bundler will raise an error if it finds conflicting dependencies. For example, if gem A requires version 3 or greater of gem Z, but gem B requires version 2, Bundler will notify you of the conflict. This becomes extremely helpful as many gems refer to other gems (which refer to other gems), which can form a large dependency graph to resolve.

Testing

Testing is a large part of Ruby culture. Ruby comes with its own Unit-style testing framework called minitest (Or TestUnit for Ruby version 1.8.x). There are many testing libraries with different goals.

- TestUnit - Ruby 1.8's built-in "Unit-style" testing framework
- minitest - Ruby 1.9/2.0's built-in testing framework
- RSpec - A testing framework that focuses on expressivity
- Cucumber - A BDD testing framework that parses Gherkin formatted tests

Be Nice

The Ruby community takes pride in being an open, diverse, welcoming community. Matz himself is extremely friendly, and the generosity of rubyists on the whole is amazing.