Rocky Enterprise Linux 9.2 Manual Pages on command 'bundle-cache.1'

$ man bundle-cache.1

BUNDLE-CACHE(1) BUNDLE-CACHE(1)

NAME

bundle-cache - Package your needed .gem files into your application

SYNOPSIS

bundle cache

DESCRIPTION

Copy all of the .gem files needed to run the application into the ven?
dor/cache directory. In the future, when running [bundle in?
stall(1)][bundle-install], use the gems in the cache in preference to
the ones on rubygems.org.

GIT AND PATH GEMS

The bundle cache command can also package :git and :path dependencies
besides .gem files. This needs to be explicitly enabled via the --all
option. Once used, the --all option will be remembered.

SUPPORT FOR MULTIPLE PLATFORMS

When using gems that have different packages for different platforms,
Bundler supports caching of gems for other platforms where the Gemfile
has been resolved (i.e. present in the lockfile) in vendor/cache. This

needs to be enabled via the --all-platforms option. This setting will be remembered in your local bundler configuration.

REMOTE FETCHING

By default, if you run bundle install(1)](bundle-install.1.html) after running bundle cache(1) bundle-cache.1.html, bundler will still connect to rubygems.org to check whether a platform-specific gem exists for any of the gems in vendor/cache.

For instance, consider this Gemfile(5):

```
source "https://rubygems.org"
gem "nokogiri"
```

If you run bundle cache under C Ruby, bundler will retrieve the version of nokogiri for the "ruby" platform. If you deploy to JRuby and run bundle install, bundler is forced to check to see whether a "java" platformed nokogiri exists.

Even though the nokogiri gem for the Ruby platform is technically ac? ceptable on JRuby, it has a C extension that does not run on JRuby. As a result, bundler will, by default, still connect to rubygems.org to check whether it has a version of one of your gems more specific to your platform.

This problem is also not limited to the "java" platform. A similar (common) problem can happen when developing on Windows and deploying to Linux, or even when developing on OSX and deploying to Linux.

If you know for sure that the gems packaged in vendor/cache are appro? priate for the platform you are on, you can run bundle install --local to skip checking for more appropriate gems, and use the ones in ven? dor/cache.

One way to be sure that you have the right platformed versions of all your gems is to run bundle cache on an identical machine and check in the gems. For instance, you can run bundle cache on an identical stag? ing box during your staging process, and check in the vendor/cache be? fore deploying to production.

By default, bundle cache(1) bundle-cache.1.html fetches and also in? stalls the gems to the default location. To package the dependencies to

vendor/cache without installing them to the local install location, you can run bundle cache --no-install.