Bundler Version

Last updated 29 September 2016

Table of Contents

When deploying a Ruby application to Heroku, a version of Bundler (https://devcenter.heroku.com/articles/bundler) will be installed and used to pull down the dependencies of your application. The Bundler version that is used is not configurable. To see the current Bundler version deployed, see the Ruby Support (https://devcenter.heroku.com/articles/ruby-support#libraries) page.

Why can’t the Bundler version be configured?

Different versions of Bundler have different known bugs and slightly different behavior. The Bundler version on Heroku is carefully curated. A balance needs to be struck between supporting new Bundler features and stability. The work we put into curating the Bundler version ensures maximum stability, and avoids deprecation and notification cycles on Bundler as it changes, or as bugs are fixed or security issues are patched by Heroku.

The Ruby experience on Heroku is provided by the Heroku Ruby Buildpack (https://github.com/heroku/heroku-buildpack-ruby). This is the tool that installs a version of Bundler and runs all the commands needed to get your application set up. The buildpack relies of publicly exposed internals of Bundler. It is tested and known to work against the currently specified version of Bundler. If you were to take a modern buildpack and use it with an older version of bundler, you would be likely to see unpredictable results.

Being aware of Bundler version changes

Changes to Bundler version will be announced on the Heroku changelog (https://devcenter.heroku.com/changelog), and the Ruby Support (https://devcenter.heroku.com/articles/ruby-support#libraries) page will be updated.

Bundler versions usually take a large amount of verification and validation and are released infrequently.

App not using the currently supported Bundler version?

If you see that a different version of Bundler is being used by your application than is listed in the Ruby Support (https://devcenter.heroku.com/articles/ruby-support#libraries) article, your application might be configured to use the master branch of the Ruby buildpack for deployments.

To check which buildpack is configured, use the `heroku buildpacks` command:

```
$ heroku buildpacks
=== hidden-temple-25627 Buildpack URL
https://github.com/heroku/heroku-buildpack-ruby.git
```

The example output shows that an unreleased version of the buildpack is being used. To switch to the supported version, set the buildpack as follows:

```
$ heroku buildpacks:set heroku/ruby
Buildpack set. Next release on hidden-temple-25627 will use heroku/ruby.
Run git push heroku master to create a new release using this buildpack.
```

You can verify that the new buildpack has been set by typing:

```
$ heroku buildpacks
=== hidden-temple-25627 Buildpack URL
heroku/ruby
```

Another reason your app might not be using the currently supported Bundler version is if it is configured to deploy using a different buildpack URL. This will happen if the `BUILDPACK_URL` config var is set.

```
$ heroku config:get BUILDPACK_URL
BUILDPACK_URL:                      https://github.com/heroku/heroku-buildpack-ruby.git
```

If you see any value, then you are using a custom buildpack. If this value is set to a “multi buildpack” such as `https://github.com/heroku/heroku-buildpack-multi` then you will need to check the `.buildpacks` file to see what buildpacks are used in deployment. If you are using this method of deployment we recommend that you instead follow the guidance in the Using Multiple Buildpacks for an App (https://devcenter.heroku.com/articles/using-multiple-buildpacks-for-an-app) article.

The officially deployed Ruby buildpack will sometimes lag behind master for a few days.

Using an older version of Bundler

There is no way to specify a version of Bundler you want independently of the buildpack. You can, however, lock your app to a deployed version of the Ruby buildpack. When the Ruby buildpack is deployed a Ruby release is created on GitHub (https://github.com/heroku/heroku-buildpack-ruby/releases).

Locking your application to a release should be a temporary measure while you diagnose or troubleshoot the problem you are having with Bundler. Locking to an older release means that when bug fixes and improvements are introduced to the buildpack you will not get them.

One of the most frequent problems for someone who has locked to an older buildpack version is when they encounter an error that prevents them from deploying. They find that the error was fixed in a more recent version of the buildpack, but in order to upgrade they must fix the original incompatibility. If this happens during a critical time then there is no emergency escape valve. They must spend time fixing their app. It is much better to stay up to date and not lock to a specific release version.

Heroku does not guarantee that locking to an older release will work forever. A part of the platform that requires an integration change may occur and introduce bugs or problems. If you encounter a new issue while deploying using a locked release you will be required to upgrade to the latest release before we can troubleshoot. No fixes will be backported to older releases. To mitigate these problems, only lock to a release for short periods of time.

To use a specific release version you can use the `buildpack:set` command and specify the release after a pound `#` in the URL. To lock to `v144` you could run this command:

```
$ heroku buildpacks:set https://github.com/heroku/heroku-buildpack-ruby.git#v144
```

The next time you deploy you'll get version `v144` of the Ruby buildpack. You can see when a version of Bundler was deployed using the Heroku Ruby Buildpack changelog (https://github.com/heroku/heroku-buildpack-ruby/blob/master/CHANGELOG.md).

If your app is failing when using a version of Bundler then you can search for that version number to see when it was deployed. You can temporarily set your app to deploy using the version before it. For example, if your app is failing on version 1.11.2 search for "1.11.2" and see that v144 was the last deploy before this change was introduced so you can lock to `v144` .

⚠ Remember that locking to a release should only be a temporary fix until you can diagnose and fix the root problem.

Known upgrade issues

- Bundler 1.13.1

1) There was a bug fix in the way that required Ruby versions are handled. You may get an error like this:

```
Bundler Output: requires_greater_than_equal_ruby_two_two-0.1.0 requires ruby version >= 2.2.2, which is
```

If that happens, you need to upgrade your version of Ruby to be compatible with the minimum Ruby version specified in your gems.

- Bundler 1.11.2

1) When upgrading from version 1.9.7 to 1.11.2 there was a change in how gemspec validations are used. Your app may be running with a gem that currently does not have a valid gemspec. If that is the case you may get an error like this in some part of your deploy output:

```
 The gemspec at /tmp/build_869ace36768a67943b91e3d7d7d4c576/vendor/gems/<gem-path>.gemspec is not valid
 The gemspec at /tmp/build_869ace36768a67943b91e3d7d7d4c576/vendor/engines/<path>.gemspec is not valid.
 rake aborted!
 LoadError: cannot load such file -- <gem-name>
```

Look for lines that start `"The gemspec at"` , identify the problem that is in the warning and fix it. The above example is of a gemspec that has `"TODO" in the description.

Another example of this class of error is:

```
The gemspec at /tmp/build_a965713cdcd547733e1dc9dd119dd04b/vendor/bundle/ruby/2.1.0/bundler/<gem-path>.
Could not find <gem>-1.1.0 in any of the sources
```

In this case, the gem that was being installed had a duplicate dependency of itself in the gemspec. This was resolved by removing the duplicate.

2) The implicit behavior of the `--without` flag changed between 1.9.7 and 1.11.2. For more information see Bundle without change in implicit behavior in v1.10.0 (https://github.com/bundler/bundler/issues/4351).

Support

Before opening a support ticket, first determine if the problem still occurs using an older version Bundler. If you can reproduce the issue on an older version of Bundler it indicates that the problem is not related to the Bundler change. Please include build output and whether or not the build also failed using an older version of Bundler when communicating to our support staff.

You can get support at https://help.heroku.com (https://help.heroku.com).