# Puma::Daemon

## Table of Contents



| Badge | Type |
|---|---|
| [badge] | Test Suite |
| [git%2Bgithub.com%2Fkigster%2Fpuma daemon] | Licensing |
| [Gem Version] | Gem Version |
| [badge] | Test Coverage |
| [sunbirst] | Coverage Areas |
| [coverage-graph] | Coverage Area |

# 1. Usage for the Impatient

1. Add `gem 'puma-daemon'` to your Gemfile. If you prefer, you can add `require: false` (your Rails instances such as Sidekiq will have no use for this gem).
2. You have two options to daemonize Puma using this gem:
   1. In your `config/puma.rb` file, add the following line: `require 'puma/daemon'` at the top, and at the bottom call `daemonize`.
   2. Wherever you use `puma` to start it (except using puma-ctl) replace `puma` with `pumad` and daemonization will be enabled automatically.

# 2. Introduction

Let's start with the facts: Puma is the most popular server to run Ruby and Rails applications. It's both multi-threaded and multi-process, making it one of the only servers that can truly saturate your web hardware. While 100% saturation is probably not what you want, the alternative is paying a fortune for under-utilized hardware. A good rule of thumb is to keep your web servers busy around 70-80% most of the time. Puma let's you do that by configuring the number of workers and threads within each worker.

## 2.1. What is Daemonization?

Unix processes have the ability to daemonize and go into the background.

This is not a trivial task: to properly daemonize a process must detach the from controlling terminal and run in the background as a system daemon.

In Ruby this is accomplished with the Process.daemon method, which receives two boolean arguments:

- Unless the first argument `nochdir` is `true`, it will change the current working directory to the root ("/").
- Unless the second argument `noclose` is true, `daemon()` will also redirect standard input, standard output and standard error to `/dev/null`.
- Finally, it will return zero on success, or raise one of ` Errno::*` and pass the control to the subsequent Ruby code, which will now continue executing within a daemon.

## 2.2. Why was it removed from Puma v5?

For production deployments, tools like `systemd` offer much better alternative, including ability to cap overall memory and CPU consumed by the Puma and all of its workers using Linux cgroups.

The proliferation of Docker deployments meant that Puma is run on the foreground within a Docker container.

Finally, the code which previously daemonized Puma in version 4 was not really maintained, and for this reason was removed from Puma version 5.

## 2.3. What does `puma-daemon` do?

We thought that while the core Puma removing daemonization was the right move, it felt useful in some occastions and so we created this gem to restore the daemonization functionality to Puma v5+.

# 3. Compatibility

## 3.1. Ruby Versions

We did not restore the daemon functionality for JRuby; so at the moment this will work with the MRI distribution, and possibly others that support `Process.daemon(true, true)`.

For supported MRI Ruby Versions see the Github Workflow file.

## 3.2. Puma Versions

Currently Puma versions 5 and 6 are supported.

## 3.3. Known Issues

Please see the list of open issues on the Issues Page. Any help is always welcomed.

## 3.4. How it works?

This gem's goal was to surgically augment Puma's source code to restore daemonization by merely requiring `puma/daemon`.

We accomplished this goal by adding the daemonization call to the routine `output_header()` which is invoked by both `Puma::Single` runner and the `Puma::Cluster` runner at the very beginning of the launch process. While relatively brittle, particularly if the future versions of Puma change this, this approach seems to work with the currently released version of Puma (5 and 6).

If you run into problems, please submit an issue.

# 4. Examples

Add this line to your application's Gemfile:

```
gem 'puma-daemon', require: false
gem 'puma',  '~> 5' # or 6
```

In your `config/puma.rb`, eg.

```
require 'puma/daemon'
bind 'tcp://0.0.0.0:3000'
workers 2
threads 4
daemonize
```

And then execute:

```
$ bundle install
```

```
$ bundle exec puma -C config/puma.rb
```

Make sure you have `config.ru` Rackup file in the current folder. Checkout the shell script inside the `example` folder for more info.

Please see the `example` directory in the source of the gem. It contains `single.sh` and `cluster.sh` scripts that boot Puma via `pumad` binary.

## 4.1. Using Config File

If you want to specify `daemonize` in your config file, simply include `require 'puma/daemon'` at the top of your config file:

```
# file: config/puma.rb
require 'puma/daemon'

port 3001
workers 3
threads 2,3
# accepts true or false, and if false is passed will NOT daemonize
daemonize
```

With this method you can continue using the standard `puma` executable to get it started, but (and this is important) — **you must remove any `-d` or `--daemonize` from the command line**, or Puma v5 and above will fail with an error.

Here is an example of daemonizing via the config file shown above, and using the regular `puma` binary:

```
 cd example
 bundle exec puma -I ../lib -C $(pwd)/puma.rb -w 4 config.ru
[62235] Puma starting in cluster mode...
[62235] * Puma version: 6.1.1 (ruby 2.7.6-p219) ("The Way Up")
[62235] *  Min threads: 0
[62235] *  Max threads: 16
[62235] *  Environment: development
[62235] *   Master PID: 62235
[62235] *  Puma Daemon: Daemonizing...
[62235] *  Gem: puma-daemon v0.2.2
[62235] *  Gem: puma v6.1.1
[62258] *      Workers: 4
[62258] *     Restarts: (✔) hot (✔) phased
[62258] * Listening on unix:///tmp/puma.sock
[62258] * Listening on http://0.0.0.0:9292
```

Note that using this method you can decide whether to daemonize or not by passing true or false to the `daemonize` method.

## 4.2. Using Command Line

If you prefer to make a decision whether to daemonize or not on the command line, you only have to make one chance: replace `puma` with `pumad`.

We did not want to conflict with the `puma` gem by introducing another executable under the same name. The executable this gem provides is called `pumad` (where 'd' stands for daemon, and follows standard UNIX convention, as in eg `sshd`, `ftpd`, etc).

If you replace `puma` with `pumad`, you no longer need to pass any additional command line flag (`-d` and `--daemonize`) to daemonize. You can continue passing them or you can remove them (these flags are stripped out before ARGV is passed onto Puma's CLI parser.)

```
 cd example
 ../exe/pumad -C $(pwd)/puma.rb -w 0 config.ru

Puma starting in single mode...
* Puma version: 6.1.1 (ruby 2.7.6-p219) ("The Way Up")
*  Min threads: 0
*  Max threads: 16
*  Environment: development
*          PID: 63179
*  Puma Daemon: Daemonizing...
*  Gem: puma-daemon v0.2.2
*  Gem: puma v6.1.1
* Listening on unix:///tmp/puma.sock
* Listening on http://0.0.0.0:9292
```

As you can see, at the end it says "Daemonizing".

If you start puma this way, you can still specify `daemonize(false)` in the configuration file to turn it off, but the default is to daemonize. Also, if you start with `pumad` you do not need to include `require 'puma/daemon'` in your configuration file, as the `pumad` binary loads all dependencies prior to parsing the config.

# 5. Contributing

You do need a working `make` utility to use the below commands.

- After checking out the repo, run `make puma-v5` or `make puma-v6` to configure your dependent vesion of Puma.
- After that, run `bin/setup` to install dependencies.
- Then, run `rake spec` to run the tests.
- You can also run `bin/console` for an interactive prompt that will allow you to experiment.
- To install this gem onto your local machine, run `bundle exec rake install`.

- To release a new version, update the version number in `version.rb`, and then run `bundle exec rake release`, which will create a git tag for the version, push git commits and the created tag, and push the `.gem` file to rubygems.org.

Bug reports and pull requests are welcome on GitHub at https://github.com/kigster/puma-daemon.

# 6. License

The gem is available as open source under the terms of the MIT License.