{0.1} Nginx + Puma

This guide describes how to integrate a **Ruby on Rails** with **Puma** and **Nginx** as web server.

Basically, We will create **Puma** socket to connect to our Nginx web server, this generates an additional layer that gives us more control on **Nginx** to handle common tasks from **Rails** side.

```
Nginx <===> Puma <===> Rails
                 ||
            Middleware
```

Requirements

- NodeJS or any other JS runtime environment alternative (Optional)

- Nginx previous installation

▸ In case you use Docker ▸ In case you use global installation:

```
$ sudo apt install nginx
```

- Rails ~> "4"

- Firewall or Security group previously configured

- Rbenv

▸ Why and How I Replaced RVM with RBENV

Installation

> NOTE: This process is completely agnostic, so, I will avoid cloud based services configurations processes such as AWS, GCP or Azure. This is made to work on basically any server.

I will suppose you have your application ready and working locally, so, first step is installing project dependencies and add initial configuration files, let's get started:

Add these dependencies in your current Gemset (Gemfile) under development group:

```
group :development do
  gem 'capistrano'
  gem 'capistrano-bundler'
  gem 'capistrano-rails'
  gem 'capistrano-rbenv'
```

```
    gem 'capistrano-sidekiq' # Optional, just in case you use Sidekiq
framework.
    gem 'capistrano3-puma'
  end
```

And then:

```
  $ bin/bundle install
```

> NOTE: To avoid dependencies issues, I didn't specified gems versions, you can read these gems documentation to see what's the propest version you need or you can just run this command:

```
  $ bin/bundle add capistrano capistrano-bundler capistrano-rails
capistrano-rbenv capistrano-sidekiq capistrano3-puma --group=development
```

And it'll install all dependencies by yourself specifing what versions you need automatically.

> TIP: Don't forget to sort out your Gemfile because of Rubocop style guide. Ordinare: Gemfile Sorter

Then, you can generate Capistrano configuration files by running:

```
  $ bin/bundle exec cap install
```

This will create essential Capistrano files and directories for two environments by default (Production and Staging).

```
├── Capfile
├── config
  │   ├── deploy
│   │   ├── production.rb
│   │   └── staging.rb
│   └── deploy.rb
└── lib
    └── capistrano
            └── tasks
```

To see more information about Capistrano set-up process: (https://github.com/capistrano/capistrano)

Now you need to configure your Capfile that's essentially where you global configuration live in, you can copy and paste from here:

▸ Capfile template

Now let's talk about config/deploy.rb file and think about this in the same way we inherit behavior from top-level classes, example: *ApplicationController*.

All configuration you put here will be inherit from all existing environments, in this file you can place things like:

- Application name
- Project repository
- Linked files
- Ruby version
- Tracked releases

So, you can replace your config/deploy.rb file with our minimalist implementation here config/deploy.rb

And finally, we need to configure our environments, remember that all configuration placed into `config/deploy.rb` will be inherit from our Capistrano environments, so let's modify our production stage to deploy your Rails application:

Go to `config/deploy/production.rb` and replace the current content with production.rb, here you need to replace:

- Server static IP Address
- Application path
- Puma config file
- Project environment

Once you finish your env configuration you could iterate same process for **staging**, **acceptance** or any other environment you have, ready for next stage?

Deployment

> Now we configured our project, it's time to deploy our application, however, there are couple steps you must follow first.

Capistrano will required several steps before we can deploy our app:

- Nginx virtual server

This step is really straightforward, Capistrano provides a task that creates your virtual server for you, it'll also copy to your `/etc/nginx/sites_available` folder.

```
$ bundle exec cap production puma:nginx_config
```

> Permission denied error: In case you need sudoers permissions to copy files to nginx virtual servers path, you can install: sshkit-sudo.

```
  $ bundle add ssh-kit --group=development
```

And then require this gem in your `Capfile`.

```
require 'sshkit/sudo'
```

▸ puma.rb

Capistrano also provides a task to generate and upload your own `Puma` config file.

```
$ cap production puma:config
```

> Permission error again?: Follow above instructions.

Ready to deploy!

Now we have everything ready, you should be able to run:

```
$ bundle exec cap production deploy
```

> Replace **production** with your environment file name, for example: **staging** to explore other envs.

You can also restart Puma by running:

```
$ bundle exec cap production puma:restart
```

See more commands here https://github.com/seuros/capistrano-puma.

Troubleshooting

▸ `.env` missing

> Access to your server and then navigate to your project folder, example:

```
$ cd ~/projects/my_application
```

> Jump onto shared folder to create your `.env` file

```
$ pwd
> ~/projects/my_application

$ cd shared

$ pwd

> ~/projects/my_application/shared

$ touch .env
```

And then populate your secrets inside your `.env` file

‣ Can't clone my application repository from my server instance:

Generate a new SSH key from your server and then add it to your Github account (https://docs.github.com/en/github/authenticating-to-github/generating-a-new-ssh-key-and-adding-it-to-the-ssh-agent)

‣ Can't connect to my server by running **Capistrano** commands

Copy your public key to `~/.ssh/authorized_keys` file inside your server