Configurando um VPS para o Rails
Usando NGINX, Unicorn, MySQL e Capistrano

Atualize o Sistema Operacional
$ sudo apt-get -y update
$ sudo apt-get -y upgrade

Instale os seguintes pacotes
$ sudo apt-get install -y build-essential autoconf bison libssl-dev libyaml-dev libreadline-dev zlib1g-dev libncurses5-dev libffi-dev libgdbm-dev libpq-dev curl nodejs npm ruby-full

Instale o yarn
npm install --global yarn

Configure o Git
$ git config --global user.name '<seu nome>'
$ git config --global user.email <seu email>

Confirme as configurações do git e as versões do Ruby e NodeJS
$ git config -l
$ ruby -v
$ nodejs --version

Configure algumas variáveis de ambiente para o SSH
Edite o arquivo **/etc/evironment** e adicione:
LC_ALL="en_US.UTF-8" // Variável que determina o idioma
RAILS_ENV="production" // Variável de ambiente para o Rails deve rodar em produção

Configure algumas variáveis de ambiente para o perfil
Edite o arquivo **/etc/profile.d/variables.sh** e adicione:
export LC_ALL="en_US.UTF-8" // Variável que determina o idioma
export RAILS_ENV="production" // Variável de ambiente para o Rails rodar em produção

Carregando as variáveis de ambiente
source /etc/profile.d/variables.sh

Testando as vairáveis
echo $LC_ALL
echo $RAILS_ENV

Reinicie o servidor e verifique se as variáveis permanecem configuradas
sudo reboot

Instalando o RVM
$ gpg --keyserver hkp://keys.gnupg.net --recv-keys 409B6B1796C275462A1703113804BB82D39DC0E3
$ curl -sSL https://get.rvm.io | bash

Carregando o RVM a primeira vez
$ source ~/.rvm/scripts/rvm

Listando as versões conhecidas do Ruby
$ rvm list known

Instalando a versão do Ruby desejada
$ rvm install 2.4

Verifique a versão instalada
$ rvm list

Use uma versão específica e a torne padrão
$ rvm use 2.4 --default

Instale o bundler
$ gem install bundler

Reinicie o servidor
sudo reboot

Instale o NGINX
$ sudo apt-get update
$ sudo apt-get install nginx

Ajuste o Firewall (liste as aplicações disponíveis)
$ sudo ufw app list

Libere o Nginx e o SSH
$ sudo ufw allow 'Nginx HTTP'
$ sudo ufw allow ssh

Verifique o status do Firewall
$ sudo ufw status

Checando se o Nginx está ativo
$ systemctl status nginx

Acesse o seu servidor via browser para confirmar se está tudo funcionando
$ http://ip_do_seu_servidor

Comandos básicos do Nginx
sudo systemctl stop nginx
sudo systemctl start nginx
sudo systemctl restart nginx
sudo systemctl disable nginx
sudo systemctl enable nginx

Instalando o MySQL
sudo apt-get install -y mysql-client mysql-server libmysqlclient-dev

Acesse e altere a senha do MySQL

sudo mysql -u root -p

ALTER USER 'root'@'localhost' IDENTIFIED WITH mysql_native_password BY 'senha_da_nasa';
FLUSH PRIVILEGES;

Estando na sua máquina local/vagrant copie a chave pública
cat ~/.ssh/id_rsa.pub

Após logar no servidor crie um usuário e grupo para deploy
sudo adduser deploy --ingroup www-data

Altere para o usuário deploy
su deploy

Crie uma pasta .ssh na home do deploy e altere as permissões
$deploy> cd
$deploy> mkdir .ssh
$deploy> chmod 700 .ssh

Adicione a chave pública da sua máquina local para o usuário deploy
$deploy> echo [cole a chave pública do seu vagrant] >> ~/.ssh/authorized_keys

Altere as permissões do arquivo
$deploy> chmod 600 ~/.ssh/authorized_keys

Faça um teste saindo do servidor e logando com o usuário deploy
ssh deploy@<IP_DO_SERVIDOR>

Crie o diretório onde ficará a aplicação
$root> sudo mkdir /var/www
$root> sudo chown www-data. /var/www
$root> sudo chmod g+w /var/www

Crie o banco de dados no MySQL
$root> mysql -u root -p
CREATE DATABASE <bd_da_app>;
show databases;

No seu projeto Rails

Inicie um repositório Git e envie seu projeto para o Github
git init
git add .
git commit -m "Primeiro Commit"

No seu projeto, mova o arquivo database.yml para o gitignore e crie uma cópia para o mesmo.
cp config/database.yml config/database.yml.example
echo config/database.yml >> .gitignore

Faça o commit removendo o arquivo do projeto
git rm config/database.yml
git add .
git commit -m "Adicionado database.yml ao gitignore"
git push origin master

Crie o arquivo de configuração do Banco de Dados no servidor
deploy$ mkdir -p /var/www/<pasta_da_app>/shared/config
deploy$ vim /var/www/<pasta_da_app>/shared/config/database.yml

```
production:
  adapter: mysql2
  encoding: utf8
  reconnect: true
  database: <bd-da-app>
  pool: 5
  username: root
  password: <senha-do-mysql>
  socket: /var/run/mysqld/mysqld.sock
```

Credentials no Rails...
local$ cat config/master.key

...copie o conteúdo e cole em:
deploy$ vim /var/www/timetoanswer/shared/config/master.key

Se você usou/configurou o WebConsole mova o conteúdo abaixo do arquivo config/application.rb para config/environments/development.rb
# Allow Web Console from Vagrant
config.web_console.whitelisted_ips = '10.0.2.2'

Precompile os assets para testar
rails assets:precompile RAILS_ENV=production

 # Gemfile
```
group :development do
  gem 'capistrano', '~> 3.10'
```

```
  gem 'capistrano-rvm'
  gem 'capistrano-bundler', '~> 1.5'
  gem 'capistrano-rails', '~> 1.4'
end
group :production do
  gem ‘mysql2’ # , '~> 0.3.18'
end
```

Rode o comando
bundle install

Verifique a versão do Capistrano
bundle exec cap -v

Gere os arquivos de configuração
bundle exec cap install

Para conhecer todos os comandos do Capistrano
bundle exec cap -T

 # Capfile
```
require 'capistrano/bundler'
require 'capistrano/rvm'
require 'capistrano/rails'
```

Configuração Global (config/deploy.rb)
```
set :application, <Nome da App>
set :repo_url, 'git@example.com:me/<repo da app>.git'    # repositório git do seu projeto
set :deploy_to, '/var/www/<pasta da app>
set :branch, 'master'
set: keep_releases, 5
set :format, :airbrussh
set :log_level, :debug
append :linked_files, "config/database.yml"
append :linked_dirs, "storage", "log", "tmp/pids", "tmp/cache", "tmp/sockets", "public/system"
```

Faça a configuração do ambiente (config/deploy/production.rb)
```
role :app, %w{deploy@<ip do seu VPS>}
role :web, %w{deploy@<ip do seu VPS>}
role :db,  %w{deploy@<ip do seu VPS>}
```

Mova o conteúdo do WebConsole de config/aplication.rb para config/environments/development.rb
```
# Allow Web Console from Vagrant
config.web_console.whitelisted_ips = '10.0.2.2'
```

Faça um commit para o seu projeto no github
```
git add .
git commit -m "Configurando o Capistrano"
git push origin master
```

Faça o primeiro Deploy
cap production deploy

Faça o check para ver se ocorreu tudo certo
bundle exec cap production deploy:check

Faça o registro do seu domínio no https://registro.br

Registre o DNS da Digital Ocean
>> https://www.digitalocean.com/community/tutorials/how-to-set-up-a-host-name-with-digitalocean

---

Instale as dependências do projeto no servidor Digital Ocean < Se possuir >
sudo apt-get update
sudo apt-get install imagemagick libmagickwand-dev

Povoe o BD (caso necessite)
Entre no servidor como deploy e rode...
deploy$ cd /var/www/<pasta da app>/current
bundle exec rails dev:setup

Configurando o NGINX

Entre no servidor como o usuário root e sobrescreva o conteúdo do arquivo /etc/nginx/nginx.conf **por** esse:

```
user www-data www-data;
worker_processes 4;
pid /var/run/nginx.pid;

events {
  worker_connections 1024;
}

http {
  include       /etc/nginx/mime.types;
  default_type  application/octet-stream;
  log_format    main '$remote_addr - $remote_user [$time_local] '
                     '"$request" $status $body_bytes_sent "$http_referer" '
```

```
                  '"$http_user_agent" "$http_x_forwarded_for"';
  sendfile on;
  keepalive_requests 10;
  keepalive_timeout 60;
  tcp_nodelay off;
  tcp_nopush on;

  add_header X-Content-Type-Options nosniff;
  add_header X-Frame-Options DENY;

  server_tokens off;
  client_header_timeout 60;
  client_body_timeout 60;
  ignore_invalid_headers on;
  send_timeout 60;
  server_name_in_redirect off;

  gzip on;
  gzip_http_version 1.0;
  gzip_comp_level 6;
  gzip_proxied any;
  gzip_types text/plain text/css application/x-javascript text/xml application/xml application/xml+rss
text/javascript;
  gzip_disable "MSIE [1-6] \.";

  include /etc/nginx/sites-enabled/*;
}
```

Altere o conteúdo do arquivo /etc/nginx/sites-enabled/default **por esse:**

```
upstream <seu_app> {
  server unix:/tmp/<seu_app>.sock fail_timeout=0;
}

server {
  listen 80;
  server_name <seu_app>.com.br;
  root /var/www/<seu_app>/current/public;
  index index.html index.htm;
  client_max_body_size 10M;

  location / {
    proxy_set_header X-Real-IP $remote_addr;
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    proxy_set_header Host $http_host;
    proxy_set_header X-FORWARDED_PROTO $scheme;
    proxy_redirect off;

    if ($request_uri ~* "\.(ico|css|js|gif|jpe?g|png)\?[0-9]+$") {
```

```
      access_log off;
      add_header Cache-Control public;
      expires max;
      break;
    }

    if (!-f $request_filename) {
      proxy_pass http://<seu_app>;
      break;
    }
  }

  error_page 500 502 503 504 /500.html;
  location = /500.html {
    root /var/www/<seu_app>/current/public;
  }

  error_page 404 /404.html;
  location = /404.html {
    root /var/www/<seu_app>/current/public;
  }
}
```

Verifique se a configuração foi feita de forma correta
sudo nginx -t

Reinicie o NGINX
sudo systemctl nginx restart

Configurando o Unicorn

Adicione a Gemfile do seu projeto

```
group :production do
   gem "unicorn"
end

group :development do
   gem "capistrano3-unicorn"
end

bundle install
```

Faça o require no Capfile

```
require 'capistrano3/unicorn'
```

Adicione ao final do arquivo config/deploy.rb

```
after 'deploy:finished, 'deploy:restart'

namespace :deploy do
   task :restart do
      invoke 'unicorn:stop
      invoke 'unicorn:start'
   end
end
```

Adicione ao arquivo config/unicorn/production.rb

```
root = "/var/www/escamboapp/current"
working_directory root

pid "#{root}/tmp/pids/unicorn.pid"

stderr_path "#{root}/log/unicorn.log"
stdout_path "#{root}/log/unicorn.log"

worker_processes 4
timeout 30
preload_app true

listen '/tmp/escamboapp.sock', backlog: 64
```

Faça um commit

```
git add .
git commit -m "Adicionando o Unicorn."
git push origin master
```

Faça um novo deploy

```
bundle exec cap production deploy
```

Quando precisar, entre como root no servidor e Reinicie o NGINX

```
sudo systemctl nginx restart
```

Acesse a aplicação! <sua app>.com.br

Configurando o envio de emails transacionais

Use o mailgun.com (cadastre-se adicione um domínio e siga os passos apresentados pelo mailgun)

Adicione a gem

```
group :production do
   gem 'mailgun-ruby', '~>1.1.6'
end

bundle
```

Abra a credentials do rails e adicione a chave
EDITOR=vim rails credentials:edit

MAILGUN_KEY: asdfasdfasdfasdf

Localmente, crie no seu projeto o arquivo
config/initializers/mailgun.rb

Adicione o conteúdo ao arquivo
```
Mailgun.configure do |config|
  config.api_key = Rails.application.credentials.MAILGUN_KEY
end
```

No arquivo config/environments/production.rb adicione o conteúdo
```
# Devise Config
config.action_mailer.default_url_options = { host: www.<seu_site>.com.br' }

# Mailgun Config
config.action_mailer.delivery_method = :mailgun
config.action_mailer.mailgun_settings = {
  api_key: Rails.application.credentials.MAILGUN_KEY,
  domain: 'mg.<seu_site>.com.br'
}
```

Faça um novo commit
```
git add .
git commit -m "Configurando o envio emails"
git push origin master
```

Lembe-se que a gem mailgun geralmente precisa de 2GB no servidor para poder se instalada. Se for o caso, faça um resize em seu droplet

Faça um novo deploy
```
bundle exec cap production deploy
```

Teste a aplicação. Neste momento já deve ser possível recuperar senha através do site.