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 automake bison libssl-dev libyaml-dev libreadline6 libreadline6-dev zlib1g zlib1g-dev libncurses5-dev ncurses-dev libffi-dev libgdbm-dev openssl libc6-dev libsqlite3-dev libtool libxml2-dev libxslt-dev libxslt1-dev sqlite3 curl vim git

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

Confirme as configurações
$ git config -l

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 /home/vagrant/.rvm/scripts/rvm

Listando as versões conhecidas do Ruby
$ rvm list known

Instalando a versão do Ruby desejada
$ rvm install 2.3
Verifique a versão instalada
$ rvm list

Use uma versão específica e a torne padrão
$ rvm use 2.3 --default

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
$ sudo ufw allow 'Nginx HTTP'

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

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>

Instale o NodeJS
$root> sudo apt-get install nodejs
$root> nodejs --version

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 escamboapp;
show databases;

No seu projeto Rails

 # Gemfile
```
group :development do
   gem 'capistrano', '~> 3.6'
   gem 'capistrano-bundler', '~> 1.2'
   gem 'capistrano-rails', '~> 1.2'
end
group :production do
   gem ‘mysql2’
end
```

Rode o comando
bundle install

Verifique a versão do Capistrano
bundle exec cap -v

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

Gere os arquivos de configuração
bundle exec cap install

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

Configuração Global (config/deploy.rb)
set :application, 'EscamboApp'
set :repo_url, 'git@example.com:me/escamboapp.git'   # repositório git do seu projeto
set :deploy_to, '/var/www/escamboapp'
set :scm, :git
set :branch, 'master'
set: keep_releases, 5
set :format, :airbrussh
set :log_level, :debug
append :linked_files, "config/database.yml"
append :linked_dirs, "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>}

Faça o primeiro Deploy
bundle exec cap production deploy:initial

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

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

Crie o arquivo de configuração do Banco de Dados no servidor
deploy$ vim /var/www/escamboapp/shared/config/database.yml
production:
  adapter: mysql2
  encoding: utf8
  reconnect: true
  database: escamboapp
  pool: 5
  username: root
  password: <sua-senha>
  socket: /var/run/mysqld/mysqld.sock

Crie o arquivo de configuração do segredo no servidor
deploy$ vim /var/www/escamboapp/shared/config/secrets.yml

Use o rake secret (localmente) para adicionar um novo conteúdo no arquivo secrets.yml do servidor
bundle exec rake secret

 # Ajuste o Gemfile
group :development do
  gem 'capistrano-rvm', '~> 1.2'
end
group :production do
  gem 'mysql2'**, '~> 0.3.18'**
end

Rode o bundle
bundle install

Faça um commit
git add .
git commit -m "Instalando o Capistrano"
git push origin master

Faça o Deploy
bundle exec cap production deploy

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
sudo apt-get update
sudo apt-get install imagemagick libmagickwand-dev

Popule o BD (caso necessite)
Entre no servidor como deploy e rode...
cd /var/www/escamboapp/current
bundle exec rake 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 escamboapp {
  server unix:/tmp/escamboapp.sock fail_timeout=0;
}

server {
  listen 80;
  server_name escamboapp.com.br;
  root /var/www/escamboapp/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://hackerboard;
      break;
    }
  }

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

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

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

Reinicie o NGINX
sudo service 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:publishing', 'deploy:restart'

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

Acesse a aplicação! www.escamboapp.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', '~>1.1.4'
end

bundle
```

Pegue a key no servidor do mailgun, acesse como deploy o servidor e em /var/www/escambo/shared/secrets.yml adicione a chave

```
production:
        MAILGUN_SECRET_API_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.secrets.MAILGUN_SECRET_API_KEY
end
```

No arquivo config/environments/production.rb adicione o conteúdo

```
# Devise Config
config.action_mailer.default_url_options = { host: 'escamboapp.com.br' }

# Mailgun Config
config.action_mailer.delivery_method = :mailgun
config.action_mailer.mailgun_settings = {
  api_key: Rails.application.secrets.MAILGUN_SECRET_API_KEY,
  domain: 'mg.escamboapp.com.br'
}
```

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

bundle exec cap production deploy

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

Habilitando o HTTPS

Ajuste o UFW
sudo ufw status
sudo ufw allow 'Nginx Full'
sudo ufw delete allow 'Nginx HTTP'
sudo ufw status

Instale o LetsEncrypt
sudo apt-get update
sudo apt-get -y install letsencrypt

Adicione no arquivo /etc/nginx/sites-enabled/default o conteúdo
```
location ~ /.well-known {
    allow all;
    root /var/www/escamboapp;
}
```

sudo nginx -t
sudo service nginx restart

Inicie o LetsEncrypt
sudo letsencrypt certonly --webroot --webroot-path /var/www/escamboapp -d escamboapp.com.br -d www.escamboapp.com.br

Será solicitado email. Digite e confirme.
Aceite também os termos.

Crie o certificado Diffie-Hellman
sudo openssl dhparam -out /etc/ssl/certs/dhparam.pem 2048

Altere o conteúdo do arquivo /etc/nginx/sites-enabled/escamboapp para:

```
upstream escamboapp {
  server unix:/tmp/escamboapp.sock fail_timeout=0;
}

server {
  listen 80;
```

```
  server_name example.com;
  return 301 https://$host$request_uri;
}

server {
  listen 443 ssl http2;
  server_name escamboapp.com.br;
  root /var/www/escamboapp/current/public;
  index index.html index.htm;
  client_max_body_size 10M;

  ssl_protocols TLSv1.2;
  ssl_ciphers
'ECDHE-ECDSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-GCM-SHA384:ECDHE-ECDSA-CHAC
HA20-POLY1305:ECDHE-RSA-CHACHA20-POLY1305:ECDHE-ECDSA-AES128-GCM-SHA256:ECDH
E-RSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES256-SHA384:ECDHE-RSA-AES256-SHA384:ECD
HE-ECDSA-AES128-SHA256:ECDHE-RSA-AES128-SHA256';
  ssl_prefer_server_ciphers On;
  ssl_certificate /etc/letsencrypt/live/escamboapp.com.br/fullchain.pem;
  ssl_certificate_key /etc/letsencrypt/live/escamboapp.com.br/privkey.pem;
  ssl_trusted_certificate /etc/letsencrypt/live/escamboapp.com.br/chain.pem;
  ssl_dhparam /etc/ssl/certs/dhparam.pem;
  ssl_session_cache shared:SSL:128m;
  add_header Strict-Transport-Security "max-age=31557600; includeSubDomains";
  ssl_stapling on;
  ssl_stapling_verify on;

  location ~ /.well-known {
    allow all;
  }

  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://escamboapp;
      break;
    }
  }
```

```
  error_page 500 502 503 504 /500.html;
  location = /500.html {
    root /var/www/escamboapp/current/public;
  }

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

Verifique e Reinicie o servidor

```
sudo nginx -t

sudo service nginx restart
```

Use o crontab para fazer a renovação automática dos certificados

```
crontab -e

30 2 * * * /usr/bin/letsencrypt renew >> /var/log/le-renew.log
35 2 * * * /bin/systemctl reload nginx
```