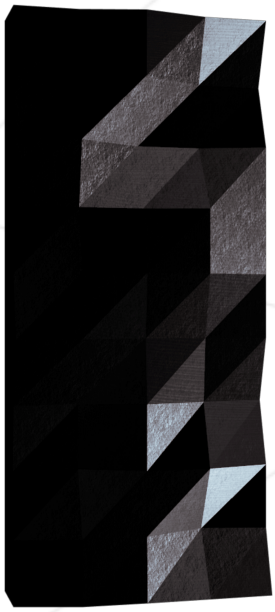

# D08 - Ruby on Rails Training

## servers

Sta ff 42 bocal@staff.42.fr
Stéphane Ballet balletstephane.pro@gmail.com

*Summary: A few notions of system administration and shell-scripting will be necessary to the aparition your application on the web. You will learn how to con fi gure nginx, cougar, and capistrano.// a rails application and a server con fi gurent proudly diff year depending on the environment .You will learn all about*

*-Setting prod- famous.*

# Contents

# Chapter I

# Preamble

Why Are So Many Websites Hosted On Linux?

tl; dr: Because of Windows windowsy's things, and Linux's unixy things

1. Stability

Linux systems are Well Known for Their Ability to run for years without failure; In fact, Many Linux users-have never seen a crash. That's great for users of every kind, PARTICULARLY goal it's valuable for small and medium-sized businesses, for qui downtime can-have Disastrous consequences.

Aussi Linux handles a wide number of processes running at once much better than Windows does-that's something, in fact, That tend to degrade Windows' stability who- CKLY.

Then there's the need for rebooting. Whereas in Windows configuration exchange ty- Pically require a reboot-causing inevitable downtime-there's no need to restart Generally Linux. Almost all Linux configuration exchange can be done while the system is running and without a ff ecting unrelated services.

Similarly, whereas Windows servers must be Often defragmented frequently, that's all aim eliminated on Linux. Let your concurrents endure the plentiful downtime inevitable Tably That goes hand-in-hand with Windows; trusty Linux will keep you up and running and serving your customers around the clock.

2. Security

Linux est innately more secure than Windows is, whether on the server, the desk- top or in an embedded environment. That's due to the fact That Largely Linux, qui est person based on Unix, Was designed from the start to be a multiuser operating system. Only the administrator, or root user, administrative privileges HAS, and Fewer users and applica- tions-have permission to access the kernel gold Each Other. That keeps everything modular and protected.

Of course, Linux gets attacked This aussi less frequently by viruses and malware, and vul- nerabilities tend be found and fi xed by more Quickly icts legions of developers and users. Even the six-year-old kernel bug That Was recently fi xed, for instance-an extremely unusual

instance in the Linux world-had never been exploited.

Internally, Meanwhile, users of a Windows system can hide Sometimes the order from the system administrator. On Linux, HOWEVER, the sys admin always: has a clear view of the fi the system and is always in control.

3. Hardware

Whereas Windows Typically requires frequent hardware upgrades to Accommodate icts ever-Increasing resource demands, Linux is slim, t rim, flexible and scalable, and it is just about Performs admirably Any computer, Regardless of processor machines gold ar- chitecture.

Linux can be aussi Easily recon fi gured to include only the services needed for your business's Purposes, THUS further Top Reducing memory requirements, Improving performance and keeping things Even simpler.

4. TCO

There's no beating Linux's total cost of ownership, since the software is free Generally. Even an enterprise version of you purchased with corporate backing Will Be Cheaper overall than Windows or other proprietary software, qui Generally Involve user-based licensing and a host of expensive add-ons, Especially For security.

Same goes for MOST of the tools and applications might be used That was Linux server. The overall TCO simply can not be beat.

5. Freedom

With Linux, there is no commercial vendor try trying to lock you into some products or protocols. Instead, you're free to mix and match and choose what works best for your business.

In short, with all the advantages Many Provides Linux in the server realm, it's no wonder gouvernements, organisms and major companies around the world, Including Amazon and Google Rely on the open source operating system In Their Own Production systems.

If you're looking for a Linux distribution to run your business it's servers, you'd do well to Consider CentOS (or RHEL, the paid version from Red Hat CentOS That is based on), Slackware, Debian and Gentoo.

# Chapter II

# Instructions

- Only this page serve as a reference: Do not fi rm the hallway noise.
- The subject can change up to an hour before rendering.
- If no contrary information is explicitly present, you must assume the following language versions:
  - Ruby 2.3.0
  - for d09 Rails> 5
  - but for all the other days rails 4.2.6
  - HTML5
  - CSS 3
- We prohibit EXPRESSLY using keywords while, for, redo, break, retry and until in Ruby source code that you will make. Any use of these keywords is considered cheating (and / or unfit), giving you the score of -42.
- The exercises are precisely ordered from simple to more complex. In any case, we do not bear attention or take into account a complex exercise if a simpler exercise is not very successful.
- Attention to the rights of your fi les and your directories.
- You must follow the rendering process for all your exercise: only the present work on your GIT repository will be evaluated in defense.
- Your exercises will be evaluated by your pool mates.
- You should leave in your repertoire no other fi le than those explicitly speci fi ed by the forward drills.
- You have a question ? Ask your neighbor to the right. Otherwise, try your left neighbor.
- Your reference manual called Google / man / Internet / ....
- Think discuss on the forum of your pool Intra!
- Read the examples carefully. They might require things that

not otherwise specified in the subject ...

- Please, by Thor and Odin! Re fl échissez dammit!

# chapter III

# Specific Rules of the day

You do not have the right to install the "gem" nginx ".

No script will be uploaded and / or fetché in rendering script, only the app "foubarre" is uploaded to bitbucket and is synchronized during the deployment by
capistrano in ex01

# Chapter IV

# Exercise 00

| | |
|---|---|
|  | Exercise: 00 Exercise 00: |
| Shell_ fl avored | |
| Render Folder: *ex* 00 / Files to make: create_server.sh | |
| | |
| Authorized Functions: Notes: n | |
| / A | |

In this day early on soft and servers into production must go to the ex00, a script executed in a fresh whole vm'll install and commission a server (puma) in applying " foubarre ".

To do this it is imperative that it operates:

- the git installation
- installing curl
- installation of vim or emacs
- installing rvm
- install ruby
- rail installation
- installing all the necessary dependencies
- the creation of the control application "foubarre"
- compiling assets
- The configuration of the 'SECRET_KEY_BASE'
- and the server start-up in production environments The script make á must integrate the following code to the creation of the moin te- application.

```
$> Mkdir / home / vagrant / Site $> cd / home / vagrant / Site
$> New foubarre rails -d postgresql $> cd foubarre


$> G scaffold rails component great_data
$> Echo "Component.create (great_data 'foo_bar_name')" >> db / seeds.rb $> bundle install

$> Sed -i -e "s / username: foubarre / username: vagrant / g" config / database.yml $> = RAILS_ENV output rake db: create $> =
RAILS_ENV output rake db: migrate $> = RAILS_ENV output rake db : seed


$> Sed -i "2iroot to 'components # index" config / routes.rb $> echo "<h1> <% =% Rails.env> </ h1>"> app / views /
components / index.html .erb
```

First, please verify that you have virtualbox and of vagrant. Then create a folder named "ex00". Start at the root command:

```
hashicorp vagrant init / jessie64
```

Replace y fi le "Vagrant end the" original by:

```
#   - * -    fashion: ruby - * -
#   vi: set ft = ruby:
VAGRANTFILE_API_VERSION = "2"

Vagrant.configure (VAGRANTFILE_API_VERSION) do | config |
    #   Use Ubuntu 14.04 Trusty Tahr 64-bit operating system as our config.vm.box = "debian / jessie64"


    #   Configurate the virtual machine to use 2GB of RAM config.vm.provider: virtualbox
    do | vb |
        vb.customize [ "modifyvm" id, "--memory", "1024"] end


    ##   Un-comment this line in order to copy your script into the VM
    ##   with the "vagrant provision" command:
    # config.vm.provision "file" source "create_server.sh" destination "~ / create_server.sh"


    #   Forward the Rails server default harbor to the host config.vm.network: forwarded_port, guest:
    3000, host: 3031 config.vm.network: forwarded_port, guest: 80, host: end 8090
```

This will give you a vm representing your server. Do not update the distribution!

You can now mount the vm using the command:

```
vagrant up
```

And log in using:

```
vagrant ssh
```

You must assign a password for the root user vagrant, it is strongly advised to use 'vagrant'.

You must change the line of the file "/ etc / hosts"

```
127.0.0.1 localhost
```

in :

```
0.0.0.1 localhost
```

You should make the script that will install the necessary elements and con fi gure so that an application named "foubarre" located at the root of "/ home / vagrant / site" (in the guest system) or accessible via your browser (in the host system) has the address "http: //0.0.0.0: 3031 /"

Uncomment the "con fi g.vm.provision" fi the "..." in the fi Vagrant to copy your script in the vm with the command:

```
vagrant provision
```

If all is well, you should see your application served by puma á "http: //0.0.0.0: 3031 /" in your browser.

# Chapter V

# Exercise 01

| | |
|---|---|
|  | Activity: 01 year 01: Ruby_ fl |
| avored | |
| Render Folder: *ex* 01 / files to make: create_server_2.sh, create_app.sh | |
| | |
| Authorized Functions: Notes: n | |
| / A | |

In this exercise we will use Capistrano, an automation library deployment on remote server.

**This means that since our environment development in local you sign in ssh on your server to synchronize your** data from an SVN of your choice, keeping 5 and previous versions of symlinks fi le configuration com- Muns the sake of disk usage optimization.

**So deploy on a machine remote ( here a vm) an application in an environment of production, and if everything** works, with a single command.

**You must also use nginx in "Reverse Proxy" which will allow us to have a ready base á receive configurations of** "load balancing", in bene fi ting from increased security

**In this exercise, you must return two scripts.**

The FIRST "create_app.sh":

This script serves á 'recreate the application and so insŕer the configuration of Nginx when correcting. This script should begin as follows:

```
$> Cat create_app.sh
new rails foubarre2 -d postgresql cd foubarre

rails g scaffold component great_data
echo "Component.create (great_data 'foo_bar_name')" >> db / seeds.rb sed -i "2iroot to 'components # index"
config / routes.rb echo "<h1> <% =% Rails.env > </ h1> "> app / views / components / index.html.erb echo"
group: development do ">> Gemfile echo" gem 'capistrano'

                                                    require: false "Gemfile >>
echo "gem 'Capistrano-rvm'                          require: false "Gemfile >>
echo "gem 'rails capistrano'                        require: false "Gemfile >>
echo "gem 'Capistrano-bundler' require: false" >> Gemfile echo "gem 'capistrano3-puma'
                                                    require: false "Gemfile >>
echo "end" >> Gemfile bundle install
install cap

echo "_votre_configuration_ICI_"> config / deploy.rb
#   Your nginx configuration file MUST be in your
#   config file in your app touch config / nginx.conf

echo "_votre_configuration_ICI _"> config / nginx.conf [...]
```

the creation script must also initialize git á the root of your project if you do not have an account bitbucket create one now. Made a new private repo on the web platform and link it to your local project:

```
    git init git add.

    git commit -m "first commit"
    git remote add origin git@bitbucket.org: user_name / repo_name -u git push origin master
```

Now your folder is synchronized with the online repository. For sake of accuracy and fairness, you have to destroy the depot and start every correction.

The SECOND "create_server_2.sh":

Create a new vm to the application root:

```
hashicorp vagrant init / jessie64
```

Replace the fi le "Vagrant end the" original by:

```
#   - * -   fashion: ruby - * -
#   vi: set ft = ruby:
VAGRANTFILE_API_VERSION = "2"

Vagrant.configure (VAGRANTFILE_API_VERSION) do | config |
   #   Use Ubuntu 14.04 Trusty Tahr 64-bit operating system as our config.vm.box = "debian / jessie64"


   #   Configurate the virtual machine to use 2GB of RAM config.vm.provider: virtualbox
   do | vb |
      vb.customize [ "modifyvm" id, "--memory", "1024"] end


config.vm.provision: shell path: "create_server_2.sh"

   #   Forward the Rails server default harbor to the host config.vm.network: forwarded_port,
   guest: 80, host: 8090
   config.vm.network: forwarded_port, guest: 22, host: 2222 id: "ssh" disabled: true config.vm.network: forwarded_port, guest: 22, host: 64673,
  auto_correct: true end
```

Thus, with the command " up vagrant " your vm will provisionéee with the script "create_server_2.sh" pretty simple right? .

You must adapt this script by using the "create_server.sh" so that it operates:

- creating the user 'deploy' with password 'deploy_password'
- the git installation
- installing curl
- installation of vim or emacs
- installing rvm
- install ruby
- rail installation
- install nginx
- installing all the necessary dependencies
- create a script called "post_deploy_symlink.sh"
- the crushing of the file "/ etc / hosts" by:

```
127.0.0.1                   foubarre.com
127.0.1.1                   jessie.raw                  jessie

:: 1             localhost localhost ip6-ip6-loopback ff02 :: 1 ip6-allnodes ff02
:: 2-ip6 allrouters
```

CHOOSE how many user needs to install anything. Knowing that the user deploy should not be included in the sudoers and Capistrano must log in and make its operations as deploy, which must have its added ssh key has bitbucket

**The fi expensive hosts must contain a jump address foubarre.com IP on the vm that is loopback. This will** declare in the configuration nginx:

```
server_name foubarre.com;
```

Nginx needs as "/ etc / nginx / sites-available / foobarre" is a symbolic link to "/ home / deploy / apps / foobarre / **current / con fi g / nginx.conf". This is the fi le that will contain your configuration for nginx works by** reverse proxy **Puma and serve. It is to this e ff ect the script post_deploy_symlink.sh must be created and executed after the first** deployment.

If all goes well you can make your first deployment:

```
bundle exec producing cap deploy: initial
```

For the exercise to be valid on the host system:

- We execute the script "create_app.sh"
- It initializes the vm via Vagrant fi on the subject
- It is checked that the application is accessible via "foubarre.com" on the browser after déploiment and execution in the script vm "post_deploy_symlink.sh"
- can alter the local app and re-deploy:

```
modified_file git add git commit -m "message" git push
origin master course bundle exec producing deploy
```

**And you must verify that within the VM:**

- the app is in "/ home / deploy / apps"
- **the app's file is "own" the user deploy**

- the user deploy neither root nor in the sudoers

Do not go head down, and read your logs: it is the key.