前提

- Vagrant: 2.2.14
- VirtualBox: 6.1.22
- ruby: 2.7.1
- Rails: 6.0.3
- MySQL: 5.6

上記環境での動作を想定 一応rails3系、ruby1系での環境構築方法も紹介

目次

- (3)ActiveStorageを削除する

Vagrantの環境を作成

1. CentOS7をvagrantにインストールする

```
$ cd {プロジェクトを作成するフォルダに移動}
$ vagrant box add centos/7

(略)

1) hyperv
2) libvirt
3) virtualbox
4) vmware_desktop

# 上記選択肢が表示された際にvirtualboxを選択する
```

vagrant上でCentOS7が使用できるようになる
内部環境は使用する仮想環境毎に異なるため、boxファイル化しても互いに異なるものが出来上がる
プロジェクトフォルダが異なればもう一度`vagrant box add`を行う必要がある

2. Vagrantfileの編集を行う

```
$ vagrant init centos/7

# Vagrantfileが生成されるため，目的に沿って編集を行う
# 下は例

Vagrant.configure("2") do |config|
  config.vm.box = "centos/7"

  config.vm.network "forwarded_port", guest: 3000, host: 3000
  config.vm.network "forwarded_port", guest: 3300, host: 3300

  config.ssh.username = "vagrant"

  config.vm.synced_folder "{ホストOS側の共有フォルダ階層(相対パス)}", "{ゲストOS側の共有フォルダ階層(絶対パス)}", create: true

  if Vagrant.has_plugin?("vagrant-vbguest") then
    config.vbguest.auto_update = true
  end
end
```

3. vagrantを起動する

```
$ vagrant up
```

上記コマンド後に共有フォルダのマウントに失敗した旨のメッセージが表示された場合，
`vagrant vbguest`の実行後に`vagrant vbguest --status`を実行して状態を確かめる

```
[default] No Virtualbox Guest Additions installation found.
```

と表示されていたらvagrant側は正常なため、後述するCentOS側の設定を行う

CentOSの環境作成

`cat /etc/redhat-release`を実行してCentOSのバージョンが正しくなっているか確認しておこう

(SSHをパスワード認証で入れるようにする)

本来は非推奨な方法だが，あくまで開発環境用の設定として行う

```
$ sudo vi /etc/ssh/sshd_config

# Passwordで検索してPasswordAuthenticationの項目をyesに変更

$ sudo systemctl restart sshd
```

1. 共有フォルダの設定を行う

```
$ sudo yum install -y kernel kernel-devel kernel-headers dkms gcc gcc-c++
$ sudo yum update -y kernel kernel-devel
```

`kernel`関連をupdateしてvagrantのGuestAdditionを対応させる

2. 既存のmariaDBを削除する

`MySQL`を利用する場合のみ必要
`mariaDB`や`PostgreSQL`の場合には必要ない

```
$ sudo yum remove mariadb-libs
$ sudo rm -rf /var/lib/mysql/
```

3. MySQLをインストールする

```
# 最新版はこちら
$ sudo yum localinstall https://dev.mysql.com/get/mysql80-community-release-el7-
3.noarch.rpm
# MySQL 5.6はこちら
$ sudo yum localinstall http://dev.mysql.com/get/mysql57-community-release-el6-
7.noarch.rpm

# MySQL 5.6をインストールする場合に必要な操作
$ sudo yum -y install yum-utils
$ sudo yum-config-manager --disable mysql57-community
$ sudo yum-config-manager --enable mysql56-community
# MySQLのバージョンが正しいことを確認
$ sudo yum repolist all | grep mysql

$ sudo yum install -y mysql-community-server
```

インストール完了後，MySQLのサービスを立ち上げる

```
# バージョンの確認
$ mysqld --version

$ sudo systemctl enable mysqld
$ sudo systemctl start mysqld
```

4. anyenvをインストールする

anyenvは現在使われている主要な言語をサポート・インストールできる管理ツール
バージョンを変更するのも容易に行えるので大変便利
余程のことでもない限りこれを使ってバージョン管理を行うのが適切

まず，anyenvのソースコードを取得する

```
$ sudo yum install -y git
$ git clone https://github.com/anyenv/anyenv ~/.anyenv
$ echo 'export PATH="$HOME/.anyenv/bin:$PATH"' >> ~/.bash_profile
$ source ~/.bash_profile
```

続いてanyenvをインストールする

```
$ anyenv init
$ echo 'eval "$(anyenv init -)"' >> ~/.bash_profile
$ anyenv install --init
$ mkdir -p $(anyenv root)/plugins
$ git clone https://github.com/znz/anyenv-update.git $(anyenv
root)/plugins/anyenv-update
$ anyenv update
```

```
# rbenvとnodenvが存在することを確認
$ anyenv install --list
```

5. rbenvをインストールする

```
# anyenv経由でrbenvをインストール
$ anyenv install rbenv
$ exec $SHELL -l
```

続いて，rbenvの動作確認を行う

```
# rbenvのバージョンを確認
$ rbenv --version

# rbenvのリストの中に目的のrubyバージョンがあるかどうか確認
$ rbenv install -L
```

5.1. rubyをインストールする

ruby 2.7.1 をインストールするために必要なパッケージをインストール

```
$ sudo yum install -y openssl-devel readline-devel zlib-devel
```

rbenv経由でrubyをインストールする

```
$ rbenv install 2.7.1
$ rbenv global 2.7.1
$ rbenv rehash

# インストールされているrubyが正しいか確認する
$ ruby -v
```

6. nodenvをインストールする

```
# anyenv経由でnodenvをインストール
$ anyenv install nodenv
$ exec $SHELL -l
```

続いて，nodenvの動作確認を行う

```
# nodenvのバージョンを確認
$ nodenv --version

# nodenvのリストの中に目的のnode.jsバージョンがあるかどうか確認
$ nodenv install -L
```

6.1. node.jsをインストールする

nodenv経由でnode.jsをインストールする

```
$ nodenv install 12.20.0
$ nodenv global 12.20.0
$ nodenv rehash

# インストールされているnode.jsが正しいか確認する
$ node -v
```

7. bundle installを行う

事前に必要なGemの選定等を行い，Gemfileを記載しておく

```
$ bundle install
```

注意点についてはbundle install関連を参照

8. データベースの設定を行う

config/database.ymlの中身の編集を行うが，下記の要点のみに気を付ければよい

```
Adapter: mysql2
username: root
password: (設定していない場合は空)
database: oo_development(test, production)
```

上記完了後，テーブルの作成とマイグレーションの実行を行う｀必要であれば，その後にdumpファイルからデータを取り込ませる

```
$ bundle exec rake db:create
$ bundle exec rake db:migrate

# 必要であれば行う
$ mysql -u root -p oo_development < dump.sql
```

Ruby 1系、Rails 3系での環境構築について

Ruby 1系のインストール

基本的には上述の `ruby 2.7.1` のインストールと同じになる
バージョン的な都合上，`bundler 1系` をインストールする必要があるが，実行パスの設定が行われていないため要注意
なお，`ruby 2.7.1` のインストールに際して追加で必要なパッケージはない

Rails 3系のインストール

基本的には上述の `bundle install` での実行手順と同じ

DBのデータ取り込み(dumpファイルの取り込み)

`bundle exec rake db:create`, `bundle exec rake db:migrate`を実行後に行う操作

```
# MySQLの場合
$ mysqldump -u root -p {DB名} > dump.sql
$ mysql -u root -p {DB名} < dump.sql

# PostgreSQLの場合
$ psql {DB名} > dump.sql
$ psql {DB名} < dump.sql
```

発生したエラーについて

gem 'sass'による`text file busy`問題

`sass 3.2.11 / 3.2.13` 等で発生
`rails server`の起動や`bundle exec rake assets:precompile`の実行時に`text file busy`と表示されて実行されない
解消するには `sass 3.2.10` に変更する必要がある

```
$ vi Gemfile

gem 'sass', '=3.2.10'      <= 追加

$ bundle update sass
```

bundler 1系の実行パス

*`bundler 1系`*を使用する場合に発生
`ruby 1系`ではインストールするだけで`bundler`を使用することはできない
別途で`bundler`をインストールする必要があるが，その時に実行パスが設定されない
そのため，手動で`bundler`の実行パスを環境変数として設定する必要がある

```
$ gem install bundler -v 1.1.3
$ echo 'export PATH="$HOME/.anyenv/envs/rbenv/versions/1.9.3-
p551/lib/ruby/gems/1.9.1/gems/bundler-1.1.3/bin/bundle:$PATH"' >> ~/.bashrc
$ source ~/.bashrc
```

bundle install関連

mysql2について

下記コマンドを実行するだけで解消可能

```
$ sudo yum install -y mysql-devel
```

pgについて

nokogiriについて

`nokogiri`が必要な場合は以下のパッケージを追加でインストールしておく

```
$ sudo yum install -y libxml2-devel libxslt-devel
```

また，bundlerのconfigを追加する

```
$ bundle config build.nokogiri --use-system-libraries
```

これでも上手くいかなかった場合にはvagrantの共有フォルダ的な問題である可能性が高いため， 共有フォルダよりも外側の位置にvendorを配置させる

mimemagicのライセンス変更問題

ライセンスの関係で `mimemagic 0.35` 以前のRailsアプリケーションは `bundle install` が失敗するようになっている
そのため，エラーが発生した場合には以下の2つの対策を取る

(1)Railsのアップグレードを行う

正確には，`Rails 5.2.5 / 6.0.3.6 / 6.1.3.1` は`mimemagic`に依存しなくなった
上記バージョンにすることで解消は可能だが，他のGemのバージョンに要注意

(2)mimemagicをアップグレードして対応する

アプリケーションのライセンスがGPLになるため要相談

```
# shared-mime-infoをインストールする
# CentOS8以降は標準でインストール済なため必要ない
$ sudo yum install shared-mime-info

# Gemfileに gem 'mimemagic', '~> 0.3.10' を追加する
$ vi Gemfile

gem 'mimemagic', '~> 0.3.10'

$ bundle update mimemagic
```

(3)ActiveStorageを削除する

`ActiveStorage`は`mimemagic`に依存しているため，該当Gemを削除すれば理論上使用可能
参考