# AmazonLinux2でLaravelの開発環境構築

PHP AWS Laravel AmazonLinux2

AWSでLaravelの開発環境を構築して行きたいと思います。

前提条件として、
すでにAWSでアカウントは作成済みとしています。
また、簡単なWEBアプリケーションは作成したことがある方を対象としています

## EC2の起動

まずは、AWSのコンソールからEC2を起動します。
「インスタンスの作成」をクリックすると、Amazon マシンイメージ（AMI）の選択画面になるので、**Amazon Linux 2 AMI (HVM),SSD Volume Type** を「選択」します。

次にインスタンスタイプの選択画面になるので、好きなインスタンスタイプを選択します。**t2.micro**が無料利用枠の対象なので、利用できる方はこちらのタイプが良いかと思います。

私は開発環境での利用なので、今回 **t3.nano** を選択しました。

インスタンスの詳細の設定については、基本的には自由に設定できますが、よくわからなければ一旦全てデフォルトのままでも問題ないかと思います。
私は、そこまで頻繁に利用しない開発環境ということもあるので、**スポットインスタンスのリクエスト**にチェックを入れて、費用を安く抑えています。スポットインスタンスは、費用を安く抑えられる反面、連続稼働を保証していないので、本番環境ではお勧めできませんので注意ください。

ストレージの追加も任意ですが、ルートの8GiBのみで進みます。

セキュリティグループの設定は、一旦最小限のセキュリティグループを作成します。

| タイプ | プロトコル | ポート範囲 | ソース |
|---|---|---|---|
| SSH | TCP | 22 | マイIP (ご自身の接続IP) |
| HTTP | TCP | 80 | カスタム 0.0.0.0/0,::/0 |

ターミナルへのログインのために、SSHのポートをご自身のIPで設定。
HTTPは、どこからでも閲覧可能なように設定。自分しか確認できないようにしたい場合は、HTTPのソースについてもマイIPを設定してください。

最後に、ログイン用のキーを生成して終了です。

# EC2の初期設定

EC2が起動したら、ec2-userとしてログインしてみましょう。
作成したキーを .ssh/ 以下に配置しておきます。

```
ssh ec2-user@ec2-**-**-**-***.ap-northeast-1.compute.amazonaws
```

Laravelをセットアップする前に、ざっとEC2の設定を行います。

とりあえず、パッケージを最新に更新

```
$ sudo yum update -y
```

ec2-userのままでも良いですが、実際に利用するユーザーを作成
ユーザー名をいつも利用しているユーザー名を利用してください。説明では munakata として進めます。

```
$ sudo su -
# useradd munakata
# passwd munakata
# usermod -G wheel munakata
```

munakata に sudo権限を付与します。

```
# visudo
```

root以下に追加

visudo

```
## Allow root to run any commands anywhere
root    ALL=(ALL)       ALL
```

```
munakata    ALL=(ALL)       ALL
```

munakataのホームディレクトリ(/home/munakata) 以下の **.ssh/authorized_keys** に公開鍵をセット。

すでにご自身の公開鍵はお持ちかと思いますが、まだない方は作成してください。
SSHキー等で検索すれば、いくつか情報が出てくるかと思います。

一応、権限を記載しておきます。

| 権限 | パス |
| --- | --- |
| 700 | ~/.ssh |
| 600 | ~/.ssh/authorized_keys |

これで、munakataユーザーでログインする準備は完了です。
一度ログアウトして、実際に新しいユーザーでログインしてみましょう。

```
ssh munakata@ec2-**-**-**-***.ap-northeast-1.compute.amazonaws
```

ローカライズ設定を行います。

タイムゾーンを日本時間にセット

/etc/sysconfig/clock

```
$sudo vim /etc/sysconfig/clock

ZONE="Asia/Tokyo"
UTC=false
```

反映には再起動が必要なので、とりあえず日本時間にセット

```
$ sudo cp /usr/share/zoneinfo/Japan /etc/localtime
```

日本語設定

/etc/sysconfig/i18n

```
$ sudo vim /etc/sysconfig/i18n

LANG=ja_JP.UTF-8
```

ここまでで、ざっとEC2の初期設定が完了です。

# 必要なパッケージのインストール

ここからは、AmazonLinux2でLaravelを構築するために必要なパッケージをインストールしていきます。
なるべく特殊なことはせず、ある程度AWSで用意された標準的なもので構築したいと思います。

WEBサーバーとして、**apache2.4** 、PHPのバージョンは、**PHP7.3** を使用します。

いきなりですが、AmazonLinux2で、yumを利用してPHPをインストールするとPHP5になってしまいます。
AmazonLinuxでは、**yum install php72** でPHP7.2をインストールできたのですが、AmazonLinux2には用意されていません。
その代わりに、**Extra Library** が用意されているようです。
**Extra Library** は、**amazon-linux-extras** で利用可能です。

php7.3をインストールしてみます。

```
$ sudo amazon-linux-extras install php7.3
```

一緒に必要なパッケージもインストールされます

```
php-cli.x86_64 7.3.6-1.amzn2.0.1 @amzn2extra-php7.3
php-common.x86_64 7.3.6-1.amzn2.0.1 @amzn2extra-php7.3
php-fpm.x86_64 7.3.6-1.amzn2.0.1 @amzn2extra-php7.3
php-json.x86_64 7.3.6-1.amzn2.0.1 @amzn2extra-php7.3
php-mysqlnd.x86_64 7.3.6-1.amzn2.0.1 @amzn2extra-php7.3
php-pdo.x86_64 7.3.6-1.amzn2.0.1 @amzn2extra-php7.3
```

確認してみましょう。

```
$ php -v
```

amazon-linux-extras でパッケージをインストールすると、拡張モジュールに関しては、yumを使って適切なパッケージをインストールしてくれるようになります。便利ですね！

実際にインストール可能な拡張モジュールを確認してみましょう。

```
$ sudo yum list php* | grep php7.3
```

必要な拡張モジュールをインストールしていきます。必要に応じて各自検討ください。
**php-xml**は、Laravelインストール時に、phpunitのインストールに必要になるようなので、事前にインストールしておきましょう。

```
$sudo yum install php-mbstring php-pecl-memcached php-gd php-a
```

次に、apache2.4をインストールします。

AmazonLinuxの場合は、2.4系を入れる場合は、 **httpd24** でしたが、AmazonLinux2の場合は、**httpd** で、2.4系になるようです。

```
$ sudo yum install httpd
```

起動します。

```
$ sudo systemctl start httpd
```

コンソール上には何も出力されないので、実際に動いているか確認します。
下記のコマンドで、**active (running)** とか表示されているはずです。

```
$ sudo systemctl status httpd
```

実際にWEBブラウザから確認してみます。

起動したEC2に、パブリックDNSが割り振られているかと思うので、そのURLで確認すると、Apacche2.4のTestPageが表示されるかと思います。

IPv4 パブリックIPでも確認可能です。

ElasticIPを紐付けた方は、そちらのIP、もしくはRoute53で設定したドメインで確認してください。

# PHPの設定

この時点で、Laravelのインストール自体は可能なのですが、
PHP、Apacheの細かい設定もしていきましょう。
PHPの設定は、 **/etc/php.ini** で行います。

設定した値を確認できるように、phpinfoの表示ページを作成しておきましょう。

設定内容は、各自調整してください。
私がよく変更する箇所は、この辺です。

php.ini

```
# HTTPヘッダにPHPのバージョンを記載しない(一応セキュリティ的にOffにし
# expose_php = On
expose_php = Off

# メモリ上限を引き上げる(結構デフォルトのメモリは少なめなので増やしてお
# memory_limit = 128M
memory_limit = 256M


# POST送信の許容サイズを引き上げる
# post_max_size = 8M
post_max_size = 16M
```

```
# アップロードファイルの許容サイズを引き上げる(スマホの写真のサイズが
# upload_max_filesize = 2M
upload_max_filesize = 16M

# timezoneの設定
# date.timezone =
date.timezone = Asia/Tokyo
```

設定を反映します。

モジュール版のPHPの場合、通常httpdを再起動すると、**php.ini**の内容が反映されるのですが、AmazonLinux2で、PHP7.3とhttpd2.4を構築した場合、デフォルトでSever APIがFPM/FastCGI となるため、httpdでは、php.iniの設定が反映されず、php-fpmの再起動が必要となります。

php-fpmについて詳しく知りたい方は、「php-fpm」等のキーワードでお調べください。

```
$ sudo systemctl restart php-fpm
```

# Laravelインストール

DocumentRootにLaravelを配置することも可能なのですが、開発環境として構築するので、今回はVirtualHostの機能を利用して、ホームディレクトリに、htmlディレクトリを作成して、その配下にLaravelプロジェクトを配置します。
ホームディレクトリの権限が700のままだと、アクセスした際にforbiddenになってしまうので、755に変更しておきます。

まずは、**Composer**をインストール

```
curl  -sS https://getcomposer.org/installer | php
```

**composer.phar** がダウンロードされるので、**composer** のコマンドで実行できるように、PATHが通っている場所へ移動させます。

```
sudo mv composer.phar /usr/local/bin/composer
```

これで **composer** が利用できるようになったので、Laravelをインストールします。

```
$ cd ~/html/
$ composer create-project --prefer-dist laravel/laravel blog
```

これで、blogというLaravelプロジェクトが構築されます。
ただし、今回利用している **t3.nano** などのインスタンスタイプだと、メモリが足らずにインストールの途中に下記のエラーが出てしまいます。

```
mmap() failed: [12] Cannot allocate memory
```

そこで、ハードディスクにswap領域を作成して、メモリ不足を補います。

最初に、現状確認

```
$ free
              total        used        free      shared  buff/
Mem:         470512      145504      118000         104       2
```

とりあえず、1G程度用意すればLaravelのインストールは可能なのでswapを作成します。

```
$ sudo dd if=/dev/zero of=/swapfile bs=1M count=1024
$ sudo chmod 600 /swapfile
$ sudo mkswap /swapfile
$ sudo swapon /swapfile
```

再度、freeコマンドでSwapが追加されていれば、OKです。

```
$ free
              total        used        free      shared  buff/
Mem:         470512      145504      118000         104       2
Swap:       1048572       27904     1020668
```

これで、やっと準備が整ったので、再度

```
$ composer create-project --prefer-dist laravel/laravel blog
```

あとは、Laravelのドキュメントに記載がある通りに設定をしていきます。

```
$ cd ~/html/blog
$ composer update
$ chmod -R 777 bootstrap/cache
```

```
$ chmod -R 777 storage
$ php artisan key:generate
```

基本的には、設定ファイルは、**.env**になりますが、開発環境専用にする場合は、下記のようにリネームします。

```
$ mv .env .env.development
```

# Apache VirtualHost 設定

最後に、ApacheのVirtualHost設定を行います。

Virtual Hostの記述は、自動で設定が読み込まれる /etc/httpd/conf.d 配下にファイルを作成して記述します。
ファイル名は任意ですが、**vhost.conf**で作成します。

```
$ sudo su -
# cd /etc/httpd/conf.d
# vim vhost.conf
```

一旦必要な記述を記載しますが、Virtual Hostの詳しい記述方法については、他で調べてみてください。
アクセス予定のドメインは、**blog.munakata.net** を仮定しています。適宜変更ください。

vhost.conf

```
<VirtualHost *:80>
    DocumentRoot /home/munakata/html/blog/public
    ServerName blog.munakata.net
    ServerAlias blog.munakata.net
    <Directory "/home/munakata/html/blog/public">

        #.htaccessを利用可能にする
        AllowOverride All

        # Laravelで利用する環境変数を development に設定
        SetEnv APP_ENV development

        #アクセス許可
        Require all granted


    </Directory>
</VirtualHost>
```

## httpd再起動

```
$ sudo systemctl restart httpd
```

Route53で、設定したドメインを紐付けるか、ご自身のマシンのhostsを設定して、ブラウザでアクセスしてみてください。
Laravelのトップページが表示されていれば、一旦完了です。