Nginx, MariaDB を あえて EC2内部に作成してみる

みなさん、こんにちは。どんぶラッコです。

AWSでサーバ環境を構築しようとすると、VPCを設定して、データベース設定はRDSに切り出して、ELBを設定して…という、王道の設定方法しか出てこないですよね。

「いや、自分でまず実験するだけだから 本当に最小構成でEC2内部で完結しちゃっていいんだけど…」なんてこと、あると思います。

ということで、今回はEC2内部にNginx, MySQL を構築し、 既存のLaravelプロジェクトをgit経由でインストール(ついでに HTTPS化)して動かすところまでの手順をまとめてみました！

- 自分で設定をすることで Nginx + MariaDB(MySQL) 構成の構築手順を復習したい
- 検証用のスモールスケールのプロダクトとして開発する
- EC2単体で動かすことでサーバ代を節約したい

という方、必見です！

(この記事は 2021年7月30日に執筆されました)

EC2インスタンスは作成済み、という前提で進めます！

EC2インスタンスは作成済みで、SSH接続もセットアップ済みという前提で進めます。なので、以下に出てくるコー　ドは全てEC2インスタンス内で実行していただくコマンドです。
また、**EC2インスタンスは Amazon Linux2 で構築されている**ことを前提に進めます。

この方法はベストプラクティスではありません

本文でも触れていますが、これはAWSの良さを全部殺してしまう設定です。スケーラブル設定なども全部自分でやらなければならなくなるので。
それに、無料枠が終わっている場合、別サーバーのVPS (Virtual Private Server)を借りて構築したほうが安い場合もあるかと思います。

もし別サーバを利用する場合、OSをCentOSにした上で、 `amazon-linux-extras` コマンドで実施しているものを `yum` 管理に置き換える必要があります。機会があれば CentOS 上での構築方法でも書き換えてご紹介します。

ハンズオンの流れ

PHP, Nginx のインストール・設定

インストール後、NginxやPHP-FPMの起動設定を記述します。

MariaDBの設定

Amazon Linux2ではデフォルトで MariaDBがインストールされています。Laravelから操作ができるようにLaravel用のデータベースとユーザを作成します。

gitのインストール・Laravelリポジトリのclone

gitを経由して既存のLaravelリポジトリをクローンします。

Laravelプロジェクトの設定

Laravelのプロジェクト設定に必要な composer, node をそれぞれインストールし、Laravelのプロジェクトをセットアップします。

権限設定, nginx設定

storageに対する書き込み権限を nginx に付与します。また、公式を参考に nginxの設定をLaravel仕様に変更します

HTTPS化

表示が確認できたら、 HTTPS化しましょう。HTTPS化に際して、ドメインが必要です。

PHP, Nginx のインストール・設定

まずは、 パッケージ管理ツール `yum` を最新状態にしておきましょう。

```
$ sudo yum update
```

Amazon Linux2 には `yum` とは別に、 `amazon-linux-extras` というパッケージ管理ツールがあります。

> **参考**
> **Amazon Linux 2のExtras Library(amazon-linux-extras)を使ってみた | DevelopersIO**
> **クラスメソッド発「やってみた」系技術メディア | DevelopersIO**

PHP と Nginx はこの `amazon-linux-extras` を使ってセットアップしていきましょう。

amazon-linux-extras を使ってインストール

まずは現存するPHPのバージョンを確認します。

```
$ amazon-linux-extras | grep php
 15  php7.2                   available    \
 17  lamp-mariadb10.2-php7.2  available    \
 31  php7.3                   available    \
 42  php7.4                   available    [ =stable ]
 51  php8.0                   available    [ =stable ]
```

`[ =stable ]` が安定版ということです。

今回は php7.4 をインストールします。

同様にnginxについても確認します。

```
$ amazon-linux-extras | grep nginx
 38  nginx1=latest            enabled      [ =stable ]
```

nginx1 で良さそうですね。

ということでインストールしていきます。ついでに `vim` と `epel` もインストールしちゃいましょう。

```
$ sudo amazon-linux-extras install php7.4 nginx1 vim epel -y
```

vim と epel について

vim は コマンド上から実行できるテキストエディタです。

epel は Extra Packages for Enterprise Linux の略で、インストールできるパッケージを拡張してくれるものです。

-y オプションについて

`-y` オプションをつけることで「このパッケージをインストールしますか？」という確認手順を飛ばすことができます。もし不安な方は `-y` を外して何がインストールされるのかを確認しましょう。

nginx, php-fpm の起動

インストールが完了したら nginx を起動しましょう。

```
sudo service nginx start
# または
sudo systemctl start nginx
```

ついでに `enable` の設定もつけます。

```
sudo systemctl enable nginx
```

この設定をすることで、サーバを再起動した時にも nginx サービスが立ち上がるようにしてくれます。

service と systemctl の違い

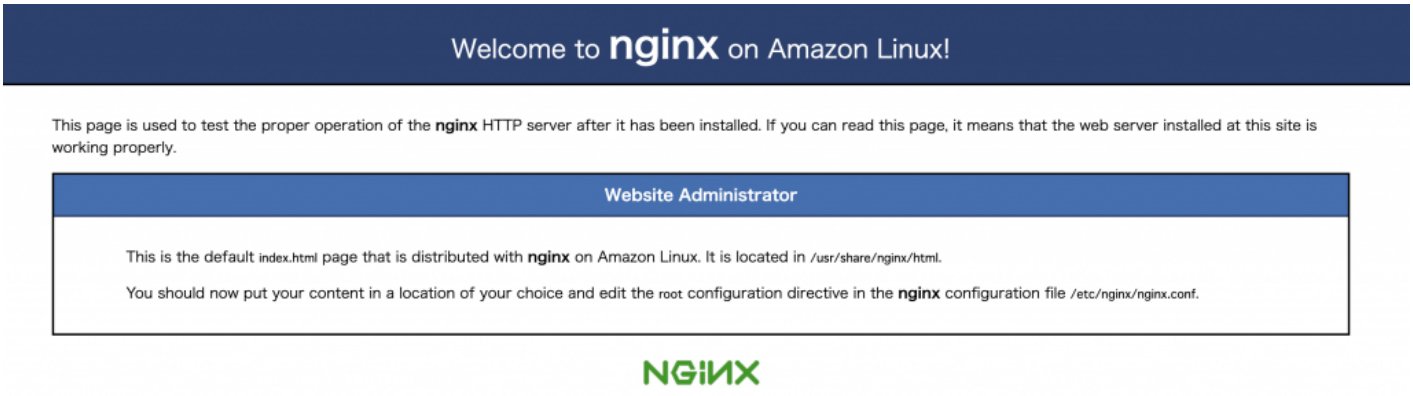
大きな違いはありません。 CentOS7以降であれば、 service は systemctl のエイリアスです。

```
$ service nginx status
Redirecting to /bin/systemctl status nginx.service
# systemctl にリダイレクトしている
```

CentOS6 系で `service` と `checkconfig` が使われていましが、 CentOS7系以降では `systemctl` に統合された名残で、両方とも使えるようになっています。
コマンドの対応表を下記サイトがわかりやすいです。

> **参考** **CentOS6, CentOS7 システムコマンド対応表**
> null

無事にサービスが立ち上がっていれば nginx の welcomeページが web上から確認できます。 `http://XXX.XXX.XXX.XXX` (XXX はご自身のIPアドレス) で確認してみましょう。



同様に php-fpmの起動設定も記述します。

```
$ sudo systemctl start php-fpm
$ sudo systemctl enable php-fpm
```

起動しているか、確認をしましょう。

```
$ service php-fpm status
Redirecting to /bin/systemctl status php-fpm.service
● php-fpm.service - The PHP FastCGI Process Manager
   Loaded: loaded (/usr/lib/systemd/system/php-fpm.service; enabled; vendor
preset: disabled)
   Active: active (running) since 金 2021-07-30 05:21:25 UTC; 14h ago
```

Active になっていればOKです。

起動が確認できたら、設定ファイルを一部書き換えます。

```
$ sudo vim /etc/php-fpm.d/www.conf
```

`user` と `group` を `nginx` に書き換えます。Laravelインストール後のpermission設定で必要になってくる設定なので忘れずに変更しましょう。

/etc/php-fpm.d/www.conf

```
; Unix user/group of processes
; Note: The user is mandatory. If the group is not set, the default user's group
;       will be used.
; RPM: apache user chosen to provide access to the same directories as httpd
user = nginx ;apacheから変更
; RPM: Keep a group allowed to write in log dir.
group = nginx ;apacheから変更
```

ついでに、listen のパスを控えておいてください。こちらも後ほどLaravelの設定で必要となります。

<meta charset="utf-8">/etc/php-fpm.d/www.conf

```
listen = /run/php-fpm/www.sock
```

保存が完了したら、php-fpmを再起動します。

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

`netstat` コマンドwを使って php-fpm の UNIXドメインソケットが開いていることを確認しましょう。先ほど確認した listenのパスで待ち受け状態になっていることが確認できます。

```
$ netstat -l | grep php-fpm
unix  2      [ ACC ]     STREAM     LISTENING     71538    /run/php-fpm/www.sock
```

`-l` は `listening` の略です。

> **参考** **netstat – ネットワークの接続状況を表示 – Linuxコマンド**
> **Linux入門**

> **参考** **AWSのEC2で行うAmazon Linux2（nginx・php-fpm）環境構築 – Qiita**
> **Qiit**

MariaDBの設定

MariaDBのインストール

MariaDBがデフォルトで導入されています。確認をしてみましょう。

```
$ sudo yum list installed | grep mariadb
mariadb-libs.x86_64                 1:5.5.68-1.amzn2             installed
```

もし導入されていない場合、 `amazon-linux-extras` を使ってインストールを実施します。

```
sudo amazon-linux-extras install mariadb10.5 -y
```

インストールが完了したらMariaDBの起動し、有効化します。

```
sudo systemctl start mariadb
sudo systemctl enable mariadb
```

ユーザ・データベースの作成

続いて、MariaDB内部の設定を行っていきます。

> **参考** **【AWS】EC2でMariaDBを使う方法｜インストールから対話モードに入り終了するまでのコマンド**
> **Proぐらし（プロぐらし）**

まずは セキュリティの設定をして、rootユーザのパスワードを設定します。

```
$ sudo mysql_secure_installation
# 対話型シェルが起動します
```

パスワード設定後、MariaDBにログインします。

```
$ mysql -u root -p
```

ログインしたら、データベースとそのデータベースにアクセスできるユーザ情報を作成します。

```
MariaDB [(none)]> create database DB_NAME;
MariaDB [(none)]> show databases;
MariaDB [(none)]> create user USER_NAME@localhost identified by PASSWORD;
MariaDB [(none)]> grant all privileges on DB_NAME.* to USER_NAME@localhost;
```

`DB_NAME` にはデータベース名 (`laravel` など)、 `USER_NAME` には任意のユーザ名 (`laravel_user` など)、 `PASSWORD` にはパスワードをそれぞれご自身の環境に合わせて設定してください。

これで、 `DB_NAME` に対してのみ全権限を持つ `USER_NAME` が作成されました。

gitのインストール・Laravelリポジトリのclone

さて、いよいよLaravelのプロジェクトを導入します。今回はすでに作成ずみのリポジトリを `git clone` で取得する想定の手順を記述します。

git のインストール

例によって `yum` を利用してインストールします。

```
sudo yum install git -y
```

リポジトリのclone

リポジトリを引っ張ってきます。今回は nginxがデフォルトで作成してくれる `/usr/share/nginx` 内に cloneします。

```
cd /usr/share/nginx
sudo git clone REPOSITORY_URL
```

きちんとインストールされているかを確認してみましょう。

```
$ ll
合計 4
drwxr-xr-x  3 root     root      112  7月 29 07:47 html
drwxr-xr-x 14 ec2-user ec2-user 4096  7月 29 07:52 REPOSITORY
```

取得してきたリポジトリの権限ユーザが `ec2-user` ではなかった場合、 `ec2-user` が触れるように権限を書き換えてください。

```
$ sudo chown ec2-user:ec2-user REPOSITORY -R
```

エイリアスを貼っておくと便利です

`cd /usr/share/nginx` へのエイリアスを作成しておくと便利です

```
$ vim ~/.bashrc
```

で編集をします。

~/.bashrc

```
alias app='cd /usr/share/nginx/REPOSITORY'
```

その後、設定を有効化しましょう。

```
$ source ~/.bashrc
```

Laravelプロジェクトの設定

Laravel のパッケージをインストールするために composer, node をインストールした後、Laravelをセットアップします。

composerのインストール

公式の `/download` に手順が載っているので、最新版の手順をを参照するようにしましょう！

> **参考** **Composer**
> **null**

2021年7月31日時点での手順は↓↓です。順番に実行していきます。

```
php -r "copy('https://getcomposer.org/installer', 'composer-setup.php');"
php -r "if (hash_file('sha384', 'composer-setup.php') ===
'756890a4488ce9024fc62c56153228907f1545c228516cbf63f885e036d37e9a59d27d63f46af1d4
d07ee0f76181c7d3') { echo 'Installer verified'; } else { echo 'Installer
corrupt'; unlink('composer-setup.php'); } echo PHP_EOL;"
php composer-setup.php
php -r "unlink('composer-setup.php');"
```

また、このタイミングで `composer install` 時に追加で必要になるPHPパッケージをインストールしてしまいましょう。

```
$ sudo yum install php-mbstring php-xml -y
```

composerのインストールをしましたが、現状はパスが通っていない状態なので、インストールしたディレクトリでしかcomposerコマンドが使えません。なので、 composer コマンドをどこでも使えるようにします。

まずは `$PATH` を叩いて、パスの格納先を確認します。

```
$ echo $PATH
/usr/local/bin # ... 省略
```

今回は `/usr/local/bin` にコマンドを移動することにします。

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

コマンドを叩いてバージョン情報が取得できていればOKです！

```
$ composer --version
Composer version 2.1.5 2021-07-23 10:35:47
```

node のインストール

nodeも `yum` パッケージで提供されているのですが、デフォルトのままではかなり古い nodeがインストールされてしまいます。(6系のバージョンです！！)

ということで、nodeのパッケージを追加で設定します。

> **参考** **GitHub – nodesource/distributions: NodeSource Node.js Binary Distributions**
> **GitHub**

今回はバージョンに制約を設けていないので、LTS版でセットアップします。 2021年7月時点では v14がインストールされます。

```
$ curl -fsSL https://rpm.nodesource.com/setup_lts.x | sudo bash -
$ sudo yum install nodejs -y
```

完了したらきちんとインストールされたか確認しましょう。

```
$ node --version
v14.17.2
```

Laravelのセットアップ

Laravelをセットアップします。この辺は従来の手順と一緒です。composerでパッケージをインストールします。

```
$ composer install
```

`.env` ファイルを作成します。

```
$ cp .env.example .env
$ vim .env
```

そしてデータベースの部分を編集します。 データベース名、ユーザ名、パスワード名はご自身が作成したものを活用してください。

```
# 前略
DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE= # データベース名
DB_USERNAME= # ユーザ名
DB_PASSWORD= # パスワード名
# 後略
```

フロント側で Laravel Mixの機能を使っている場合、 `npm` でパッケージをインストール & ビルドします。

```
$ npm i
$ npm run prod
```

権限設定

続いてディレクトリの権限設定を実施します。

```
$ sudo chown -R nginx:nginx storage/
$ sudo chmod 775 -R storage/
$ sudo chown -R nginx:nginx bootstrap/cache
$ sudo chmod 775 ./bootstrap/cache
```

このままだと、自身のユーザ (`ec2-user`)でコマンドを実行すると弾かれてしまうので、 `ec2-user` を `nginx` グループに追加します。

```
$ sudo usermod -aG nginx ec2-user
```

設定を有効化するには再ログインが必要です。

また、デフォルトでは `storage` に生成される `laravel.log` のパーミッションが 644 のため、ec2-userで書き込みをした際にログが書き込まれません。なので、パーミッションを変更する必要があります。

> **参考** | **Laravelのログファイルのパーミッションを変更**
> **もがき系プログラマの日常**

config/logging.php

```
        'daily' => [
            'permission' => 0664,     //644から変更
        ],
```

アプリケーションキーを作成します。

```
$ php artisan key:generate
```

ここまでできたら `migrate` が通ることを確認しましょう。

```
$ php artisan migrate
```

nginxの設定

公式の設定を参考に、Nginxの設定ファイルを記述していきます。

> **参考** | **デプロイ 8.x Laravel**

```
$ sudo /etc/nginx/nginx.conf
```

nginx.conf

```
http {
    # 中略

    server {
        server_name  example.com; # ご自身で取得したドメイン https化の際に必要です
        root         /usr/share/nginx/REPOSITORY/public; # Laravel リポジトリへのパス

        add_header X-Frame-Options "SAMEORIGIN";
        add_header X-Content-Type-Options "nosniff";

        index index.php;

        charset utf-8;

        location / {
                try_files $uri $uri/ /index.php?$query_string;
        }

        location = /favicon.ico { access_log off; log_not_found off; }
        location = /robots.txt  { access_log off; log_not_found off; }

        error_page 404 /index.php;

        location ~ \.php$ {
                fastcgi_pass unix:/run/php-fpm/www.sock; # PHP-FPMの際に確認したパスを記述します
                fastcgi_param SCRIPT_FILENAME $realpath_root$fastcgi_script_name;
                include fastcgi_params;
        }

        location ~ /\.(?!well-known).* {
                deny all;
        }
}
```

ポイントは3つです。

`server_name` にご自身が取得されたドメイン名を記述します。HTTPS化しないのであれば省略しても動作します

`root` にリポジトリへのリンクを記述します。 `/public` ディレクトリにつなげることを忘れずに！

`location.fastcgi_pass` に PHP-FPM のソケットURLを記述します

ここまでできたらnginxを再起動します

```
$ sudo systemctl restart nginx
```

さあ、 `http://XXX.XXX.XXX.XXX` (XXX はご自身のIPアドレス) でLaravelのページが表示ができているか確認してみましょう！

SELinuxが原因で表示されない場合もあります

その際は SELinuxを無効化すると表示されます

```
$ sudo setenforce 0
```

HTTPS化

最後にHTTPS化の手順をご説明します。

> **参考** **無料SSL証明書 Let's Encryptのススメ その2**
> **株式会社龍野情報システム**

ドメインの設定を！

本章ではサーバへのドメインの設定が済んでいる前提で進めます。
Route53などを使ってEC2に対するドメインの設定を完了させた状態で読み進めてください

snap をインストール

HTTPS化するために Let's Encryptを利用します。 `certbot` コマンドを使うと簡単にセットアップすることができるのですが、 `certbot` をセットアップするために `snap` コマンドを設定します。

epelパッケージが必要です

この記事を上から順に実施している方は既にインストール済みです。
ここから読んだ方はEPELパッケージをインストールしてから読み進めてください。

```
sudo amazon-linux-extras install epel -y
```

snapをインストールするためには EPELが必要なのですが、 `amazon-linux-extras` を使ってインストールする `epel` だけではパッケージが足りません笑

ということで、追加で必要なパッケージを追加でインストールしましょう。

参考

AmazonLinux2 に snapd を入れて certbot による証明書自動更新生活を満喫する
AR ホームベーカリー

```
$ cd /etc/yum.repos.d/
$ sudo wget https://people.canonical.com/~mvo/snapd/amazon-linux2/snapd-amzn2.repo
```

インストール完了後、 `yum.conf` にパッケージを追加します

```
$ sudo vim /etc/yum.conf
```

/etc/yum.conf

```
# 一番下に追記
exclude=snapd-*.el7 snap-*.el7
```

さて、ここまで完了したら `snap` コマンドを設定しましょう。

参考

Installing snap on CentOS | Snapcraft documentation
Snapcraft

```
$ sudo yum install snapd -y
$ sudo systemctl enable --now snapd.socket
$ sudo ln -s /var/lib/snapd/snap /snap
```

これで準備完了です！

certbotのインストール

さて、いよいよ certbotをインストールしていきます。この手順は2021年7月30日時点のものです。最新の手順は公式ページを参照するようにしましょう。

> **参考** | **Certbot – Centosrhel7 Nginx**

```
$ sudo snap install core
$ sudo snap refresh core
```

```
$ sudo snap install --classic certbot
$ sudo ln -s /snap/bin/certbot /usr/bin/certbot
```

certbot-auto は使わない！

certbotの記事を確認すると `certbot-auto` を使った方法をよく見かけますが、サポートが終了しています。

> We used to have a shell script named `certbot-auto` to help people install Certbot on UNIX operating systems, however, this script is no longer supported.
>
> *https://certbot.eff.org/docs/install.html#id8*

なので、使わないようにしましょう！
この記事も1年経てば手順が変わっている可能性があるので、必ず１次情報に当たるようにしましょう！

さて、いよいよ `certbot` にセットアップをしてもらいます。今回は nginxを使っているので `--nginx` オプションを利用します。

```
$ sudo certbot --nginx
```

あとは自動で書き込みをしてくれます！ 改めて `nginx.conf` を見ると、追記されていることが確認できます。

/etc/nginx/nginx.conf

```
http {
    # 中略

    server {

      listen [::]:443 ssl ipv6only=on; # managed by Certbot
      listen 443 ssl; # managed by Certbot
      ssl_certificate PATH; # managed by Certbot
      ssl_certificate_key PATH; # managed by Certbot
      include PATH; # managed by Certbot
      ssl_dhparam PATH; # managed by Certbot
    }

    server {
    if ($host = example.com) {
        return 301 https://$host$request_uri;
    } # managed by Certbot


        listen       80;
        listen       [::]:80;
        server_name  example.com;
    return 404; # managed by Certbot


}}
```

HTTPSへの自動リダイレクトも書いてくれていますね！

最後にnginxを再起動させて設定を有効化しましょう。

```
$sudo systemctl restart nginx
```

これでHTTPSの設定は完了です！即時反映されるはずです。

EC2インスタンスのセキュリティグループで443ポートを許可することを忘れないにしましょう！(いつもELBを使っているので、僕はここで詰まっちゃいました笑)

ポート範囲	プロトコル	ソース	セキュリティグループ
80	TCP	0.0.0.0/0	
80	TCP	::/0	
22	TCP		
22	TCP		
443	TCP	0.0.0.0/0	

Nginxの server_name が設定されているか確認を！

`server_name` が指定されていないとコマンドの実行が失敗します。設定忘れがないか確認しましょう。

```
http {
    # 中略

    server {
        server_name  example.com; # ご自身で取得したドメイン https化の際に必要です
        # 中略
    }
}
```

この記事のまとめ

かなりの長編でしたし、こうやって１から設定する機会も少なくなると思います。ただ、１つずつステップを追っていくことでより理解が深まりますよね！

ぜひ皆さんも挑戦してみてください♪

修正・改善点あったらコメントお待ちしています！