Universidade do Minho

Mestrado Integrado em Engenharia Informática

Guia de Instalação

Grupo 5

Nuno Leite (A70132)

Bruno Carvalho (A76987)

João Alves (A77070)

Hugo Oliveira (A78565)

Raphael Oliveira (A78848)

Joel Rodrigues (A79068)

Elisa Valente (A79093)

Francisco Araújo (A79821)

24 de Janeiro de 2020

Conteúdo

Lista de Figuras

1 Arquitetura geral

A *Stratus* é dividida em duas aplicações distintas: a aplicação *web* e a *API REST*, responsável por fornecer uma interface de acesso ao modelo de negócio da aplicação.

Na figura 1 é representada uma estrutura básica da *Stratus*, alguma da *stack* tecnológica utilizada e os pontos de acesso à aplicação: a verde o acesso à aplicação *web* e a vermelho o acesso à *API REST*.

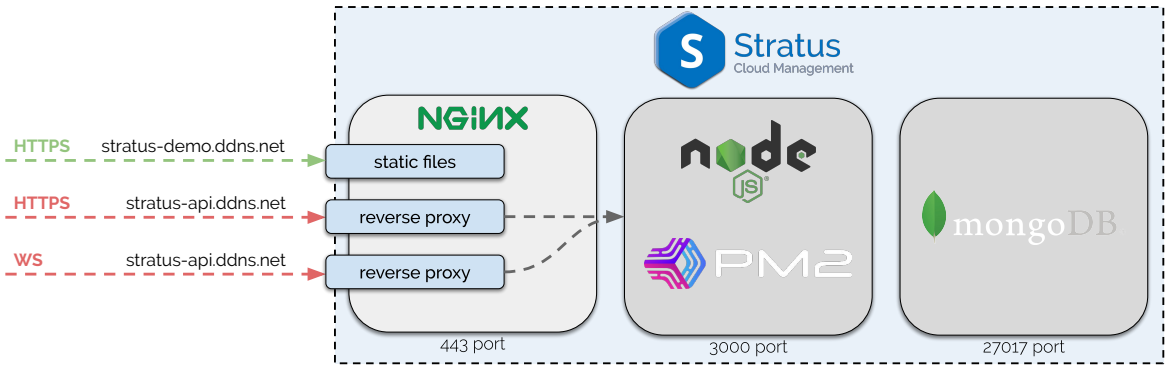

Figura 1: Exemplo de uma arquitetura de *deployment*.

Como podemos observar na figura 1, é necessário:

- que exista uma ou mais instâncias com ou sem endereço *IP* externo: uma instância a servir as duas aplicações ou uma instância por aplicação;
- a instalação de serviços externos à *Stratus*: *nginx*, *node.js*, *mongo*, *redis*, *ansible*, etc

Este documento foca-se na instalação da aplicação *Stratus* no sistema operativo **Ubuntu 18.04 LTS**. Contudo, os passos mencionados são semelhantes para outros sistemas operativos.

Poderá ainda considerar a instalação de serviços externos em instâncias distintas para uma melhor gestão de recursos e um melhor desempenho da aplicação.

Durante as próximas seccções são apresentados alguns dos passos necessário para a instalação da aplicação *Stratus* num única instância.

2 *Deployment* do *backend*

2.1 Instalação do *Node*

A lógica de negócio da *Stratus* foi desenvolvida recorrendo à *framework Node.js*. Nos passos em baixo utilizaremos o *npm* para gerir a instalação das dependências da *Stratus*, no entanto poderá ser utilizado outro gestor, como por exemplo, o *yarn*.

```
sudo apt-get update

# instalar a versão 12 do nodejs
curl -sL https://deb.nodesource.com/setup_12.x -o nodesource_setup.sh
sudo bash nodesource_setup.sh
sudo apt-get install nodejs -y
```

2.2 Instalação do *MongoDB*

A *Stratus* utilizou como sistema de gestão de base de dados o *MongoDB*. Configurações e criação da base de dados são temas abordados mais à frente.

```
sudo apt-get install mongodb -y
# iniciar o serviço
sudo systemctl start mongodb
# ativar o serviço no arranque da instância
sudo systemctl enable mongodb
```

2.3 Instalação do *Redis*

Foi utilizado ainda uma aplicação de persistência rápida *key-value*, o *Redis* que permite uma fácil acesso aos *refresh tokens* de sessão dos utilizadores.

```
sudo apt-get install redis-server -y
# iniciar o serviço
sudo systemctl start redis-server
# ativar o serviço no arranque da instância
sudo systemctl enable redis-server
```

2.4 Instalação do *Ansible*

A instalação de serviços só é possível uma vez que a *Stratus* recorre ao *Ansible*, uma ferramenta de *deployment* automático.

```
sudo apt-get install software-properties-common -y
sudo apt-get update
sudo apt-get install ansible -y
```

2.5 Instalação de um *reverse proxy*

```
sudo apt-get install nginx -y
# iniciar o serviço
sudo systemctl start nginx
# ativar o serviço no arranque da instância
sudo systemctl enable nginx
```

2.6 Instalação do *PM2*

A instalação do *PM2* surge como necessidade de mantermos o *backend* a correr após erros, *reboots*, etc.

É possível ainda com esta ferramenta colocarmos em execução múltiplos *backend*.

```
# instalar o pm2 globalmente
sudo npm install -g pm2
```

2.7 Preparar os ficheiros para produção

Na diretoria do `backend/stratus-api` da *Stratus*, deve-se proceder à configuração dos ficheiros de produção e à instalação das dependências utilizadas.

É necessário criar um ficheiro onde estão presentes todas as variáveis de ambiente (`.env`). Este ficheiro pode ser uma cópia do ficheiro `.env.sample` que é disponibilizado pela *Stratus*.

```
# aceder à diretoria do backend e da api e copiar o ficheiro .env.sample
cd ~/pei/backend/stratus-api
cp .env.sample .env
```

No ficheiro `.env` deverão ser definidas as seguintes variáveis:

```
# APPLICATION SERVER
HOST=0.0.0.0
PORT=3000

# MONGO SERVER
MONGO_HOST=localhost
MONGO_PORT=27017
MONGO_DATABASE=stratus
MONGO_USER=
MONGO_PASSWORD=

# REDIS SERVER
REDIS_HOST=localhost
REDIS_PORT=6379
REDIS_PASSWORD=

# MAIL SERVER
MAIL_HOST=smtp.mailtrap.io
MAIL_PORT=2525
MAIL_USER=16af7716550b67
MAIL_PASSWORD=82ec4e277d2bf3
MAIL_FROM=no-reply@stratus.com
MAIL_NAME=Stratus

# JWT SETTINGS
TOKEN_TTL=1800s
REFRESH_TOKEN_TTL=86400s
```

A instalação das dependências utilizadas é muito simples e basta utilizar o comando em baixo.

```
# aceder à diretoria do backend e da api
cd ~/pei/backend/stratus-api

# instalar as dependências
npm install
```

2.8 Gerar dados de exemplo

Por forma a se testar o *backend*, a *Stratus* desenvolveu um ficheiros que permitem popular a base de dados com informação de teste. Para isso, deverá utilizar os seguintes comandos.

```
# instalar o node-mongodb-fixtures globalmente
sudo npm install node-mongodb-fixtures -g

# aceder à diretoria do backend e da api
cd ~/pei/backend/stratus-api

# carregar os dados para a base de dados
mongodb-fixtures load -u "mongodb://localhost:27017/stratus" -p src/fixtures
```

2.9 Definir chaves para os *tokens* de sessão

O *backend* da *Stratus* recorre aos *tokens JWT* e para isso é necessário criar dois pares de chaves públicas e privadas para o *token JWT* e para o *refresh token*, respetivamente.

```
# aceder à diretoria do backend e da api
cd ~/pei/backend/stratus-api

# criar a diretoria para armazenar as chaves
mkdir src/config/jwt

# generate the private and public keys (JWT Token)
ssh-keygen -t rsa -P "" -b 4096 -m PEM -f src/config/jwt/jwtRS256.key
ssh-keygen -e -m PEM -f src/config/jwt/jwtRS256.key > src/config/jwt/jwtRS256.key.pub

# generate the private and public keys (Refresh Token)
ssh-keygen -t rsa -P "" -b 4096 -m PEM -f src/config/jwt/jwtRS256Refresh.key
ssh-keygen -e -m PEM -f src/config/jwt/jwtRS256Refresh.key > src/config/jwt/jwtRS256Refresh.key.p
```

2.10 Preparar e configurar o arranque da aplicação

Tal como mencionado anteriormente, será utilizado o *PM2* para procedermos à execução contínua da aplicação mesmo após *crash*, *reboot*, etc.

Nesse sentido, é necessário indicar ao *PM2* qual a aplicação que deverá correr e preparar o mesmo para iniciar após um *reboot* do servidor.

```
# aceder à diretoria do backend e da api
cd ~/pei/backend/stratus-api

# adicionar o backend ao serviço pm2
pm2 start src/server.js --name stratus-backend

# preparar o pm2 a iniciar após o reboot do servidor
pm2 startup ubuntu -u oliveirahugo.97
```

```
sudo env PATH=$PATH:/usr/bin pm2 startup ubuntu -u oliveirahugo.97 --hp /home/oliveirahugo.97

# gravar os processos que irão executar após o startup do servidor
pm2 list
pm2 save
```

Utilizar o *reverse proxy* para expormos a aplicação para a rede externa. Isto permitirá também a utilização de *SSL*.

```
# remover a configuração default do nginx
cd /etc/nginx/sites-enabled
sudo unlink default
sudo systemctl restart nginx

# criar um ficheiro para a nova configuração
sudo touch /etc/nginx/sites-available/stratus-api.ddns.net

# criar symbolic link da nova configuração
cd /etc/nginx/sites-enabled
sudo ln -s ../sites-available/stratus-api.ddns.net .
```

Deve então proceder à configuração do *nginx*. Este passo pressupõe a existência de um domínio e de um certificado *SSL*. O domínio de exemplo é *stratus-api.ddns.net*.

```
# virtual host configuration for stratus-api.ddns.net
server {
    listen 80;
    listen [::]:80;

    server_name stratus-api.ddns.net;

    return 301 https://stratus-api.ddns.net$request_uri;
}

server {
    listen 443 ssl;
    listen [::]:443 ssl;
    server_name stratus-api.ddns.net;

    ssl_certificate /etc/letsencrypt/live/stratus-api.ddns.net/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/stratus-api.ddns.net/privkey.pem;
    include /etc/letsencrypt/options-ssl-nginx.conf;

    access_log /var/log/nginx/stratus-api.ddns.net.access.log;
    error_log /var/log/nginx/stratus-api.ddns.net.error.log;

    location / {
        proxy_pass http://127.0.0.1:3000;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection 'upgrade';
        proxy_set_header Host $host;
```

```
        proxy_set_header Access-Control-Allow-Origin *;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-Proto $scheme;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_cache_bypass $http_upgrade;
    }
}
```

Após definir a configuração do *backend*, é necessário carregar reiniciar o serviço *nginx*.

```
sudo systemctl restart nginx
```

O *backend* está agora disponível no endereço IP ou domínio associado à instância onde foi instalado.

3 *Deployment* do *frontend*

3.1 Construir os ficheiros para produção

O *frontend* da *Stratus* foi desenvolvido em *React* e é necessário instalar todas as dependências *javascript* que foram utilizadas, para isso instala-se um gestor de dependências, como o *npm*.

```
sudo apt-get update

# instalar a versão 12 do nodejs
curl -sL https://deb.nodesource.com/setup_12.x -o nodesource_setup.sh
sudo bash nodesource_setup.sh
sudo apt-get install nodejs -y
```

Posteriormente, na diretoria do *frontend* da *Stratus*, deve-se proceder à configuração e compilação dos ficheiros de produção, que serão utilizados pelo servidor *web*.

É necessário criar um ficheiro onde estão presentes todas as variáveis de ambiente (`.env`). Este ficheiro pode ser uma cópia do ficheiro `.env.sample` que é disponibilizado pela *Stratus*.

```
# aceder à diretoria do frontend e copiar o ficheiro .env.sample
cd ~/pei/frontend
cp .env.sample .env
```

No ficheiro `.env` deverão ser definidas as seguintes variáveis:

```
# STRATUS backend variables
REACT_APP_WEB_SOCKET=https://stratus-api.ddns.net
REACT_APP_API=https://stratus-api.ddns.net/api
REACT_APP_IMAGES=https://stratus-api.ddns.net/public

# GOOGLE oauth variables
REACT_APP_GOOGLE_OAUTH_CLIENT_ID=
```

De notar que o domínio `stratus-api.ddns.net` representa o IP da máquina onde está a correr o *backend* da *Stratus* e o valor de `REACT_APP_GOOGLE_OAUTH_CLIENT_ID` representa o *client id* público apresentado anteriormente que, por motivos de segurança, não apresentamos neste documento.

Para compilarmos os ficheiros é necessário utilizar os seguintes comandos.

```
npm install
npm run build
```

3.2 Instalação de um servidor *web*

O *frontend* da *Stratus* foi desenvolvido em *React* e, deste modo, todo o seu conteúdo é estático. O servidor *web* selecionado foi o *nginx*.

```
sudo apt-get install nginx -y
# iniciar o serviço
sudo systemctl start nginx
# ativar o serviço no arranque da instância
sudo systemctl enable nginx
```

Após os ficheiros do *frontend* estarem compilados e o servidor *web* instalado é necessário copiar os ficheiros para uma diretoria acessível pelo serviço *nginx* e proceder posteriormente à configuração do *nginx*.

```
# criar a diretoria de destino
sudo mkdir /var/www/stratus-demo.ddns.net

# aceder à diretoria do frontend e copiar os ficheiros compilados
cd ~/pei/frontend
sudo cp -r build/* /var/www/stratus-demo.ddns.net

# fornecer as permissões necessárias ao nginx
sudo chown -R www-data:www-data /var/www/stratus-demo.ddns.net
sudo chmod 755 -R /var/www/stratus-demo.ddns.net
```

Devemos agora configurar o servidor *nginx* para utilizar os ficheiros da *Stratus* no domínio definido, bem como com os certificados *SSL* corretos.

```
# remover a configuração default do nginx
cd /etc/nginx/sites-enabled
sudo unlink default
sudo systemctl restart nginx

# criar um ficheiro para a nova configuração
sudo touch /etc/nginx/sites-available/stratus-demo.ddns.net

# criar symbolic link da nova configuração
cd /etc/nginx/sites-enabled
sudo ln -s ../sites-available/stratus-demo.ddns.net .
```

Deve então proceder à configuração do *nginx*. Certifica-se que pretende utilizar HTTPS e que possuir os certificados SSL correspondentes ao domínio utilizado.

Nesta demonstração, estamos a utilizar o domínio `stratus-demo.ddns.net`.

```
# default server configuration
server {
    listen 80 default_server;
    listen [::]:80 default_server;

    server_name _;

    return 444;
}

# virtual host configuration for stratus-demo.ddns.net
server {
    listen 80;
    listen [::]:80;

    server_name stratus-demo.ddns.net;

    return 301 https://stratus-demo.ddns.net$request_uri;
}

server {
    listen 443 ssl;
    listen [::]:443 ssl;
    server_name stratus-demo.ddns.net;

    root /var/www/stratus-demo.ddns.net;
    index index.html;

    ssl_certificate /etc/letsencrypt/live/stratus-demo.ddns.net/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/stratus-demo.ddns.net/privkey.pem;
    include /etc/letsencrypt/options-ssl-nginx.conf;

    access_log /var/log/nginx/stratus-demo.ddns.net.access.log;
    error_log /var/log/nginx/stratus-demo.ddns.net.error.log;

    location / {
        try_files $uri /index.html;
    }
}
```

Após definir a configuração do *frontend*, é necessário carregar a mesma para o serviço *nginx*. Para isso basta fazer *restart* ao serviço.

```
sudo systemctl restart nginx
```

O *frontend* está agora disponível no endereço IP ou domínio associado à instância onde foi instalado.