# FTRV Backend Application Deployment

**Services used for backend development and permission to access those services:**

AWS EC2 Instance
AWS Route53

## Backend Server (AWS EC2 ) Information and specifications:

The cloud service being used for this project is AWS. The specification for backend server are given below:

- **Category:** General
- **RAM:** 4GB
- **CPU:** 2 vCPUs
- **Storage:** 50GB
- **OS:** Ubuntu 20.04 LTS (AWS AMI)

Create EC2 Instance with the above specification. Please select the "us-east-1" as a region since most of our clients are from near that region.

## Getting started with the deployment

Below is the summary of the steps for the deployment process. The steps requiring the detailed description are described in the separate headings later in the document.

**Step 1: CREATE HOSTED ZONE and 'A' RECORDS**
Get the domain from the client and add a new AWS Route53 Hosted Zone.
While creating, choose the public hosted zone in the ***Type*** field. Once the Hosted Zone is created, this will create 4 values with type ***NS***. Add ***NS*** to the configurations of the host to the domain.
The verification of the ***NS*** can be done from [http://leafdns.com/](http://leafdns.com/)

**Step 2:**
Once the ***NS*** are added in the domain configurations, it will take 1-2 hours to set up.

**Step 3:**
Get the domain and create a subdomain by adding it to Route 53 (the details are discussed later in the document under heading *AWS Route 53)*.

**Create Record to point subdomain to the server IP:**
Add a new record in the created Hosted Zone for “A” level to point the subdomain to the ***Server IP***. Add two records in the Hosted Zone of AWS for **api.mydomain.com** and **www.api.mydomain.com**.

## Step 4 : Setup FTRV backend on AWS EC2 .

1) Take a clone of the repository “**ftrv-intranet-backend** ”.
   https://github.com/datics/ftrv-intranet-backend

2) Create a ***.env*** file with the needed variables (according to your setup). The needed variables for the ***.env*** can be copied from the ***.envtemplate*** file in the repo directory.

3) Install PostgreSQL 12 on EC2

i) Run following command to update repository

```
sudo apt update
```

ii) We need to import the GPG key and Add the PostgreSQL 12 repository into our Ubuntu machine.

Run following command to import GPG key

```
curl -fsSL https://www.postgresql.org/media/keys/ACCC4CF8.asc|sudo gpg --dearmor -o
/etc/apt/trusted.gpg.d/postgresql.gpg
```

Run the following command to add a repository.

```
echo "deb http://apt.postgresql.org/pub/repos/apt/ `lsb_release -cs`-pgdg main" |sudo tee
/etc/apt/sources.list.d/pgdg.list
```

Run following command to update repository

```
sudo apt update
```

iii) Run following command to install postgres 12 and client

```
sudo apt -y install postgresql-12 postgresql-client-12
```

iv) Check the status of PostgreSQL Service and enabled it

```
sudo systemctl status postgresql.service
sudo systemctl enable postgresql.service
```

4) Configure the postgreSQL for user “**postgres**” to change authentication from peer to md5

i ) login with user 'postgres' (created at the time of installation)

```
sudo -u postgres psql
```

After login you will get the following prompt

```
psql (12.11 (Ubuntu 12.11-1.pgdg20.04+1))
Type "help" for help.

postgres=#
```

ii ) Set password for user “postgres” with following command.

```
ALTER USER postgres WITH PASSWORD 'yourpassword';
```

iii ) Modify **pg_hba.conf** to change the authentication method from peer to md5 for postgres

Open the following file in the text editor.

```
sudo nano /etc/postgresql/12/main/pg_hba.conf
```

Search and change the following line from peer to md5

```
# Database administrative login by Unix domain socket
local   all             postgres                        peer
```

To

```
# Database administrative login by Unix domain socket
local   all             postgres                        md5
```

iv) Restart the PostgreSQL Service to take effect

```
sudo systemctl restart postgresql.service
```

v ) Verify the login by the following command, it should prompt for the password

```
sudo psql -U postgres
Password: for user postgres:
```

5) After login, create the schema **'funtown'** or any required schema.

```
CREATE SCHEMA funtown;
```

For verification list the schemas by command \dn

```
postgres=# \dn

  List of schemas
  Name   |  Owner
---------+----------
 funtown | postgres
 public  | postgres
(2 rows)
```

6) By default database “**postgres**” already exists. if you need any other please create it.

   For verification list the databases by command \l

```
postgres=# \l

                              List of databases
   Name    |  Owner   | Encoding | Collate |  Ctype  |   Access privileges
-----------+----------+----------+---------+---------+-----------------------
 postgres  | postgres | UTF8     | C.UTF-8 | C.UTF-8 |
 template0 | postgres | UTF8     | C.UTF-8 | C.UTF-8 | =c/postgres          +
           |          |          |         |         | postgres=CTc/postgres
 template1 | postgres | UTF8     | C.UTF-8 | C.UTF-8 | =c/postgres          +
           |          |          |         |         | postgres=CTc/postgres
(3 rows)
```

7) Modify your .env file (created at step 2) with db user, db password , db name etc. created at the above step.

8) Change directory to the repository folder you cloned at step 1. like: **ftrv-intranet-backend** and install the dependencies and run migrations.

   Firstly install the Node package manager toll **npm** if not installed on the system.

   ```
   sudo apt install npm
   ```

   i) Install the node js app dependencies by (it uses package.json)

   ```
   npm install
   ```

   ii) Run the migration script by

   ```
   npx sequelize-cli db:migrate
   ```

   iii) Run seed data

```
npx sequelize-cli db:seed:all
```

9) Build and run the Node js Backend application.

```
npm run build
```

FOR DEVELOPMENT

```
npm start
```

FOR PRODUCTION run the following command (it runs app with pm2)

```
npm run start:prod
```

Output of start command look likes this or differ in case of production (public ip or url)

```
> ftrv-intranet-backend@1.0.0 start /home/ubuntu/ftrv/ftrv-intranet-backend
> babel-node ./src/server
Server running on http://0.0.0.0:3000
```

10) Verify the setup by accessing the /api-docs

i ) In case of EC2 you have to provision the 3000 port or any other port (http or https) you are binding for the app as in-bound rule in the attached Security Group.

ii ) In case of EC2 here is the following example of how to access backend api via browser

[http://ec2-52-90-93-184.compute-1.amazonaws.com:3000/api-docs/](http://ec2-52-90-93-184.compute-1.amazonaws.com:3000/api-docs/)
webpage shows FTRV Intranet Portal with its APIs

11) To enable pm2 logs file rotation , you can use the pm2 module "pm2-logrotate". It rotates logs automatically of processes managed by pm2.

To install pm2-logrotate, run the following command , default file size is 10MB and retains 30 files. different parameters of "pm2-logrotate" can be configured.

```
pm2 install pm2-logrotate
```

NOTE: the command is "pm2 install" NOT "npm install"

To view current config , run following command

```
pm2 config pm2-logrotate
```

### Step 5: Setup Nginx Web Server

Use this guide to install and configure the Nginx web server.

How-to-install-nginx-on-ubuntu-20-04.pdf

OR

https://www.digitalocean.com/community/tutorials/how-to-install-nginx-on-ubuntu-20-04

### Step 6: Setup SSL Certificates

Use this guide to setup SSL and secure the Nginx using Let’s Encrypt

How_To_Secure_Nginx_with_Let's_Encrypt_on_Ubuntu_20.04.pdf

OR

https://www.digitalocean.com/community/tutorials/how-to-secure-nginx-with-let-s-encrypt-on-ubuntu-20-04

***After doing all the above steps Backend deployment is completed.***

# FTRV Frontend Application Deployment

**Services used for backend development and permission to access those services:**

AWS Route53
AWS S3
AWS CLOUDFRONT
AWS ACM

Follow the instructions from one of the links below to create an AWS S3 bucket, CloudFront distribution, Route53 records and AWS.

https://www.codebyamir.com/blog/setup-a-static-site-with-custom-ssl-certificate-on-aws-using-cloudfront-s3

OR

Setup a Static Site with Custom SSL Certificate on AWS using CloudFront and S3.pdf

## Getting started with the deployment

**Step 1: Clone repo**
Clone the branch ***develop*** from the repo https://github.com/datics/ftrv-intranet-frontend into your local machine.

Go to the repo dir and create an environment variable file ***.env*** with the variable name and there value given by frontend developer. See template file **.envtemplate** for needed variables.

**Step 2: Build the application**
Install the prerequisites

*npm install*

*npm run build*

**Step 3: Deploy the application**
i. Upload the content in the ***repo-dir/build/*** directory into the AWS S3 bucket created for this deployment.
ii. Invalidate the AWS CloudFront Distribution created for this deployment.