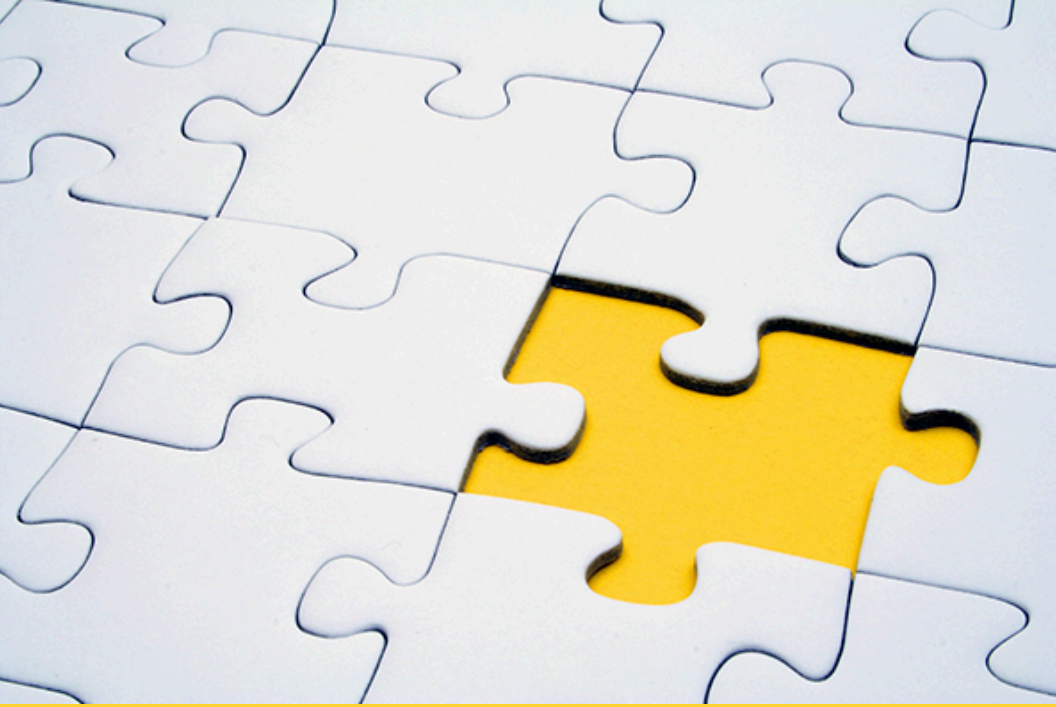


VOIP - KAMAILIO

Initial Installation of Kamailio

Using Kamailio as SBC

By Ihab Khalil

Initial Installation of Kamailio

Using Kamailio as an SPC

By Ihab Khalil

As a first step is configuring any SIP network, and SBC might be good a good start because it will act as a firewall to your internal network, in addition, it will allow you to easily expand your network in the future by adding more PBXs for geo-redundancy and load balancing.

Prologue

In the daily life of voice providers, the first part in our network is install SBC which is our session border controller. Sounds complicated ! not really, it simply means a server which switch the SIP calls between sub systems.

Its definition “A session border controller is like a firewall specifically designed for VoIP. It’s a hardware device or a software application that governs call admission to a network topology at the border. In this instance, the border refers to the space where the private network meets the public Internet. As part of controlling the border, the system filters calls, manages bandwidth, and protects against malware and viruses.” (https://getvoip.com/library/session-border-controller/, Aug 25 2024). The SBC acts like a firewall for the voice (or media) traffic ensuring its secured, protected and appropriately handled.

There are many available commercial SBC software and hardware, however, in this article, we use kamailio (open source software) as an SBC.

Chapter 1

Install your kamailio server is the first step in the process.

As usual, we use Debian 11 as our base server image (OS) to install our packages. I would like to add that currently there are many ways to create your own servers which are faster and easier to manager like using KUBERNETES which i hope to cover in a future article.

For this article, we simply created a new VM in our local openstack server and installing kamailio.

In this installation steps, we will install:

1. Kamailio
2. Mysql server (or we can use mariadb)
3. laravel for UI

Installing Mysql server

we probably have installed mysql servers many times before. BTW: we do not need to install the mysql on the same VM instance as the kamailio. We can use another VM instance or dedicated server on the network as our mysql server. We can connect to mysql from kamailio over the network. If you do have a mysql server already running, you can use it.

However, this article, in our case, since we are building it from scratch to see how we will configure it to be our SBC, we will install mysql on the same VM as the kamailio server.

Please remember if you decided to use a remote mysql server (not installed on the same VM instance), the network connection between your mysql server and kamailio should be with very little latency and secured (preferably over a VPN or over a closed network behind a security firewall). If the connection between the kamailio server and the mysql server has high latency, it will slow down your SBC and all your SIP calls will have high delay which is not what we are looking for.

Installing mysql is very simple step. Run on your Debian 11 these two commands:

apt update
apt install default-mysql-server

Installing Kamailio

To install Kamailio simply run this command on your Debian 11 server:

apt install kamailio kamailio-mysql-modules

Reference: (https://kamailio.org/docs/tutorials/devel/kamailio-install-guide-deb/, Aug 26 2024)

Installing PHP 8.2

sudo dpkg -l | grep php | tee packages.txt

sudo add-apt-repository ppa:ondrej/php

Press enter when prompted

sudo apt update
sudo apt install php8.2 php8.2-cli php8.2-{bz2,curl,mbstring,intl}
sudo apt install php8.2-fpm
apt-get install php8.2-xml
sudo apt-get install php-mysql
sudo apt-get install php8.2-zip

Installing Composer

Composer is a powerful dependency management tool for PHP. It simplifies the process of including and managing external libraries and packages in your PHP projects.

We will use composer to create our Laravel php project. Laravel is a great PHP framework for developing web applications. It also supports mysql database server. The installation requires installing composer

sudo apt install curl git unzip
cd ~
curl -sS https://getcomposer.org/installer -o composer-setup.php
HASH=`curl -sS https://composer.github.io/installer.sig`

Verify the installer has the same hash key as defined in the composer website (https://composer.github.io/pubkeys.html)

echo $HASH

Then compare the outputs to ensure you downloaded is safe

php -r "if (hash_file('SHA384', 'composer-setup.php') === '$HASH') { echo 'Installer verified'; } else { echo 'Installer corrupt'; unlink('composer-setup.php'); } echo PHP_EOL;"

It should output “Installer Verified”

Next step is to install composer globally on the machine
sudo php composer-setup.php --install-dir=/usr/local/bin –filename=composer

Finally, verify that composer is installed by running the command to ensure it will run and show the help outputs.

composer

Installing NGINX

NGINX is a web server to handle the http(s) requests to laravel PHP server to be able to view the configurations pages. NGINX is not used for the voice traffic, its only, in our case, used to view the configurations and edit the configurations for kamailio to secure and switch our voice traffic.

sudo apt-get install nginx

Chapter 2

Creating the user interface is our task for this chapter. As we installed in the last chapter, we will use Laravel to be our user interface to be able to configure kamailio especially:

1. we can configure the outbound peers
2. we can configure the internal PBXs

The project name will be called sbc, and we will install it in the /srv directory.

cd /srv

sudo mkdir sbc

sudo chmod 777 sbc

cd sbc

composer create-project laravel/laravel sbc

cd sbc

In this step we will create a .env file for the project, we will copy the default example version:

cp .env.example .env

Then we should edit the .env file, set the application name to SBC

APP_NAME=SBC

To set use mysql and set the database name to sbc_db, of course if you changed the database user/pass access, set this in this section of the .env file. Also if the database is hosted on a different server, use the server IP or domain name in the DB_HOST field. The configurations below uses password as password, please make sure to change the passwords to hard passwords when deploying to a public environment.

DB_CONNECTION=mysql

```
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=sbc_db
DB_USERNAME=sbc
DB_PASSWORD=password
```

you will need to create the table in the mysql db

```
$ sudo mysql -u root
> create database sbc_db;
> exit
```

Next, we will work on adding the filamentphp package to laravel, the filamentphp package makes creating the UI pages much easier - saves us from making blades and controllers. It is a great package and saves us a lot of development time.

composer require filament/
:"^3.2" -W

php artisan filament:install --panels

php artisan make:filament-user

Next step is to create the database on the mysql server. Again, in our case here, the server is installed on the same VM/machine. So we will login to the mysql server and CREATE a DATABASE using these commands. Kamailio has a script to create its database, if you want to change the database name or its location, edit the file /etc/kamailio/kamctlrc. Please note i did not use the same username for

larvel and kamailio. Kamailio init scripts will fail if the user already exists.

```
DBENGINE=MYSQL
DBHOST=localhost
DBPORT=3306
DBNAME=kamailio
DBRWUSER="kamailio"
DBRWPW="password"
DBROUSER="kamailioro"
DBROPW="kamailioro"
```

Then after editing the configurations to enable mysql and setup the host and user/pass, we should run the script to create the kamailio database

sudo kamdbctl create

It will ask you for the database language charset, choose UTF32 for the database char set.

Configure Nginx

In previous chapter we created the sbc laravel project in the /srv path. Now, we will need to configure nginx to point to /srv/sbc/sbc/public path. To configure nginx, simply enable it first

sudo systemctl enable nginx

sudo systemctl start nginx

You will need to edit file /etc/nginx/sites-enabled/default to include our default server configurations:

```
server {
    listen 80;
    server_name _;
    root /srv/sbc/sbc/public;

    add_header X-Frame-Options
"SAMEORIGIN";
    add_header X-XSS-Protection "1;
mode=block";
    add_header X-Content-Type-Options
"nosniff";

    index index.html index.htm index.php;

    charset utf-8;

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

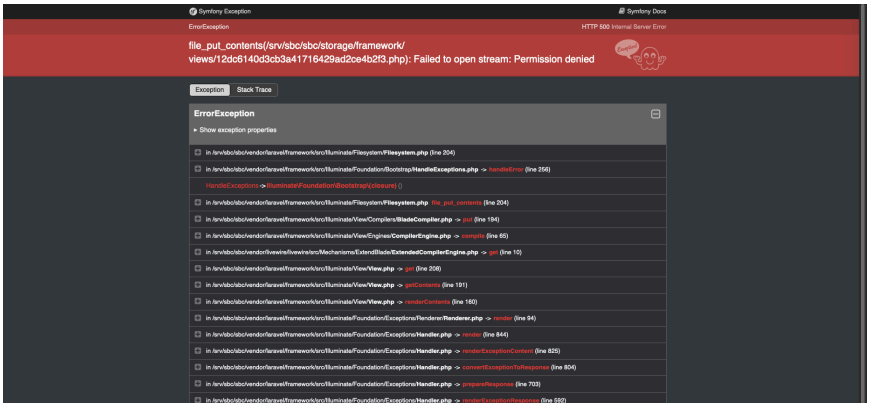
```
    location = /favicon.ico { access_log off;
log_not_found off; }
    location = /robots.txt  { access_log off;
log_not_found off; }

    error_page 404 /index.php;

    location ~ \.php$ {
        fastcgi_pass unix:/var/run/php/php8.1-
fpm.sock;
        fastcgi_index index.php;
        fastcgi_param SCRIPT_FILENAME
$realpath_root$fastcgi_script_name;
        include fastcgi_params;
    }

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

You can delete any other server configurations in the /etc/nginx/sites-enabled/default, the only server config is the one above for now.

In the future, you can add the domain to the nginx configurations (under field server_name, and use https (ssl)

instead of un-encrypted http. But for the purpose of this document, we will just use http.

Now, lets test to see if nginx is running. Simply open your browser and open the IP of the server, you should see a page similar to this one:



The error above appear because we did not setup the laravel permissions yet.

Configure Laravel

first cd to the directory where we created the sec project

cd /src/sbc/sbc

the do a composer update

composer update

composer require laravel/passport

Create a user to access the database

sudo mysql -u root

> CREATE USER 'sbc'@'localhost' IDENTIFIED BY ‘password’;

*> GRANT ALL PRIVILEGES ON *.* TO ‘sbc’@'localhost';*

> FLUSH PRIVILEGES;

Then edit the .env file to put the username and password for the new user:

DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=sbc_db
DB_USERNAME=sbc
DB_PASSWORD=password

Then we give permissions for the directories which will be accessed from the web server nginx:

php artisan storage:link
sudo chown -R www-data:www-data storage

sudo chown -R www-data:www-data bootstrap/cache

sudo chmod -R 775 storage

sudo chmod -R 775 bootstrap/cache

php artisan key:generate

php artisan passport:keys

php artisan cache:clear

php artisan route:clear

php artisan config:clear

php artisan view:clear

php artisan passport:install

`Create the filamentphp panel:`

php artisan filament:install –panels

`then create the first user:`

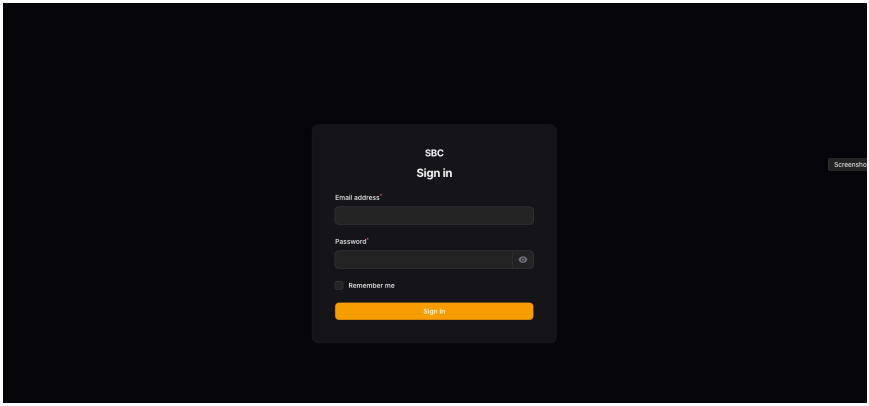
php artisan make:filament-user

It will ask you for the username and password for the user. If it gives you error “Filament has not been installed yet: php artisan filament:install –panels”, then edit app/Providers/Filament/AdminPanelProvider.php and add ->default() to the $panel.

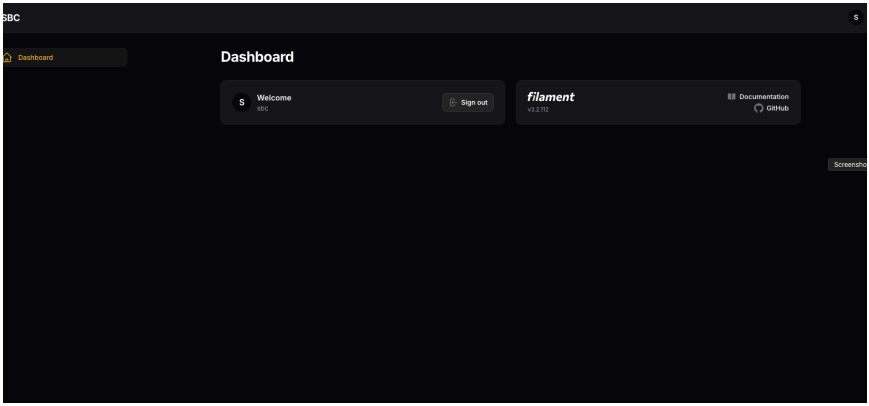
Then add the login() to the panel, edit app/Providers/Filament/AdminPanelProvider.php and add to the $panel

->login()

Now, lets try to login to our web portal http://localhost/admin, you should see the login page



and enter the email and password, you should be able to login

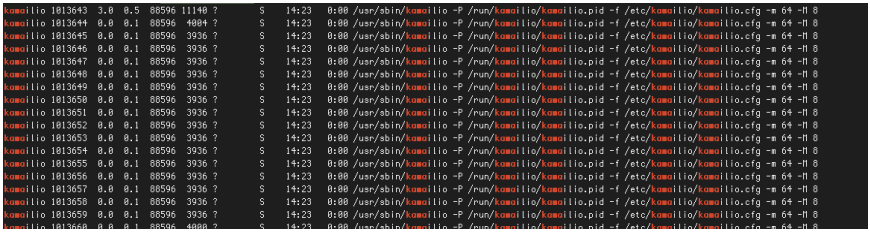


Before starting the next chapter to start adding upstream providers (CLAC or telecom) and PBX peers, then start routing the calls based on the area code and DID (phone number), we need to note:

1. try to run this command to ensure kamailio is running

ps aux | grep kamailio

you should see a lot of processes running



2. If you wish to re-create the kamailio tables:

```
sudo mysql -u root
> drop user kamailio@*;
> drop user kamailio@192.168.0.1;
> drop user kamailioro@*;
```

sudo kamdbctl reinit

```
- Install presence related tables? (y/n): y
- rtpproxy rtpengine secfilter? (y/n): n
- uid_uri_db? (y/n): y
```

3. try this command to get into the kamailio command line interface:

sudo kamcmd

It should print something like:

```
kamcmd 1.5
```

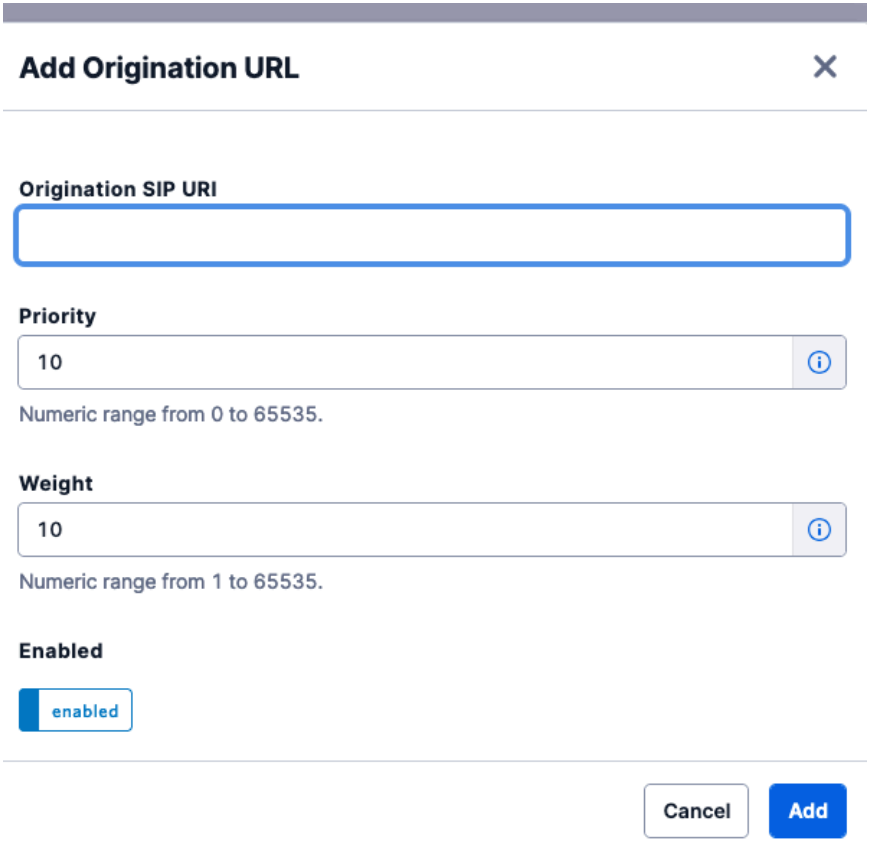
```
Copyright 2006 iptelorg GmbH
This is free software with ABSOLUTELY NO
WARRANTY.
For details type `warranty'.

kamcmd>
```

then enter exit to exit the kamailio command line

Peers

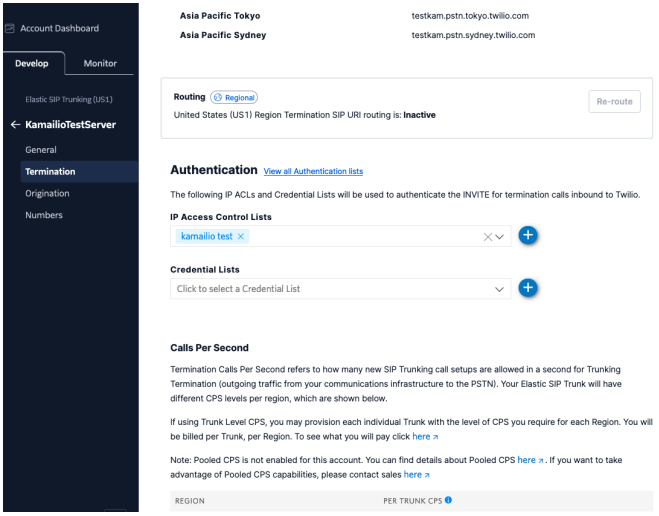
In many of the peers we define them by their IP and port, we usually do not configure authentication user/password. to add these peers to the list. For this part, lets test with twilio, (an account with twilio is required for this step) :

1. goto Elastic SIP Trunking, and create a new trunk.
2. under your new trunk configurations, select origination, then add new origination url:



3. in the Origination SIP URI, add your server IP, don't forget you need to add sip: prefix. For example: sip:88.88.88.88

4. in the Termination configuration, click on add IP Access Control List, and add your server IP



Configure the connection for Laravel to the kamailio DB

Now noticed that laravel is using sbc_db but kamailio is using kamilio database. We need to see the database for kamailio in laravel to be able to edit it, and do reports on it. So in laravel we will add a new connection in config/database.php (edit file /srv/sbc/sbc/config/database.php) and add to the connection array:

```
'kamailio' => array(
        'driver'    => 'mysql',
```

```
            'host'      => 'localhost',
            'database'  => 'kamailio',
            'username'  => 'kamailio',
            'password'  => 'password',
            'prefix'    => '',
        ),
```

Now lets add a peer to kamailio:

kamctl address add 200 54.172.60.0 30 5060 twilio_virgina

Now, lets create a laravel model for the authorization tables (address) to view the list of endpoints (or peers) with IP based authentication and be able to edit this list from web interface.

Creating the Address Page in the Web Portal

First lets create a new model in laravel for the address table (notice the model name starts with capital letter A):

php artisan make:model Address

Next open the created model and set the connection and table it should use:

```
vi app/Models/Address.php
add these two lines to the class Address
```

```
    protected $connection= 'kamailio';
    protected $table = 'address';
```

Next step is to add the view using filamentphp,

1. create the resource

php artisan make:filament-resource address

2. edit the resource file

vi app/Filament/Admin/Resources/AddressResource.php

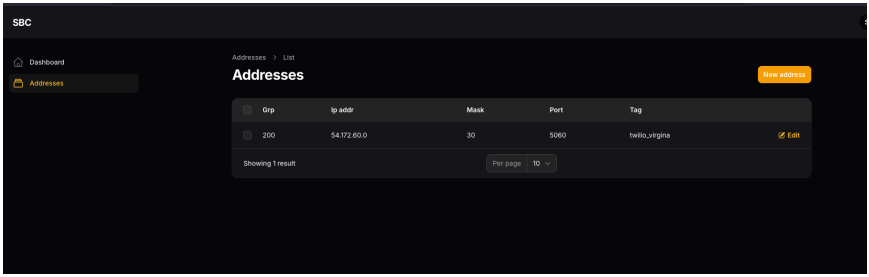
3. add these to the $form->schema([

```
        TextInput::make('grp'),
        TextInput::make('ip_addr'),
        TextInput::make('mask'),
        TextInput::make('port'),
        TextInput::make('tag'),
]);
```

4. add these fields to return $table

```
            ->columns([
                    TextColumn::make('grp'),

TextColumn::make('ip_addr'),

TextColumn::make('mask'),

TextColumn::make('port'),
                TextColumn::make('tag'),
            ])
```

now open the web browser and login, you will find a menu item called address, click on it and you should see the peer information we added above to kamailio



In next section, we will use the kamailio carrier routing module, its documentations is available at this link: https://www.kamailio.org/docs/modules/stable/modules/carrierroute.html#idm59

Configure Routing in Kamailio

1. we need to configure in kamailio configurations:

```
loadmodule "carrierroute.so"

modparam("carrierroute", "config_source",
“db”);
```

```
modparam("carrierroute", "db_url",
"mysql://user:password@localhost/
kamailio")

modparam("carrierroute",
"carrierroute_table", "carrierroute")
```

2. we need to add the route in the kamailio.cfg configurations:

```
# Apply carrier route
if (cr_route("1")) {
    route(RELAY);
    exit;
}
```

Create UI page for the routing

this is the definition of the carrier route table

```
mysql> describe carrierroute;
+---------------+--------------+------+-----+---------+----------------+
| Field         | Type         | Null | Key | Default | Extra          |
+---------------+--------------+------+-----+---------+----------------+
| id            | int unsigned | NO   | PRI | NULL    | auto_increment |
| carrier       | int unsigned | NO   |     | 0       |                |
| domain        | int unsigned | NO   |     | 0       |                |
| scan_prefix   | varchar(64)  | NO   |     |         |                |
| flags         | int unsigned | NO   |     | 0       |                |
| mask          | int unsigned | NO   |     | 0       |                |
| prob          | float        | NO   |     | 0       |                |
| strip         | int unsigned | NO   |     | 0       |                |
| rewrite_host  | varchar(255) | NO   |     |         |                |
| rewrite_prefix| varchar(64)  | NO   |     |         |                |
| rewrite_suffix| varchar(64)  | NO   |     |         |                |
| description   | varchar(255) | YES  |     | NULL    |                |
+---------------+--------------+------+-----+---------+----------------+
12 rows in set (0.00 sec)
```

Add GUI for the Routing

In the laravel project directory, run this command to create a new model for the routing table

php artisan make:model Routing

Then edit the new Routing model

```
vi app/Models/Routing.php
```

```
add these two lines to the class Routing
```

```
    protected $connection= 'kamailio';
    protected $table = 'carrierroute';
```

Next step is to add the view using filamentphp,

1. create the resource

php artisan make:filament-resource routing

2. edit the resource file

vi app/Filament/Admin/Resources/RoutingResource.php

3. add these to the $form->schema([

```
        TextInput::make('carrier'),
        TextInput::make('domain'),
        TextInput::make('prefix'),
        TextInput::make('rewrite_host'),
        TextInput::make('rewrite_prefix'),
]);
```

```
4. add these fields to return $table
```

```
            ->columns([
TextColumn::make('carrier'),
TextColumn::make('domain'),
TextColumn::make('prefix'),
TextColumn::make('rewrite_host'),
TextColumn::make('rewrite_prefix'),
            ])
```

About the Author

Ihab Khalil MEng,