

Nginx reverse proxy with  
OasisLMF User interface

Table of Contents

1 Introduction ..... 3

1.1 Purpose of this document..... 3

1.2 What you'll need..... 3

1.3 AWS ec2 configuration..... 4

1.4 Freenom setup..... 5

1.5 Domain configuration..... 6

2. CloudFlare..... 7

2.1 Server setup..... 8

3.1 Nginx Reverse proxy setup..... 9

3.2 Use of certbot to get SSL certificate ..... 10

“ Nginx configuration..... 11

“ Nginx configuration..... 12

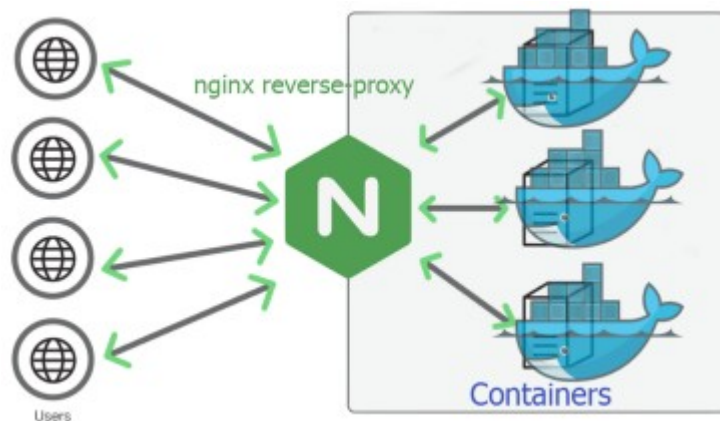
“ Nginx configuration..... 13

4.1 Troubleshooting..... 14

# Configuration Guide

## 1.1 Document

The intended purpose of this document is to give the reader an understanding on how to setup their own Virtual private server running OasisLMF UI docker container using nginx reverse proxy with a free domain name.



## 1.2 What you'll need

- Dedicated server or a VPS. Reachable with a static IP (AWS recommended)
- CloudFlare account (Free)
- Freenom account (Free)
- Familiarity with Linux

# AWS ec2 configuration

Let's get started.

2.

In the following section we are going to cover a brief walkthrough of creating an EC2 instance with the correct security group and network parameters assigned.

First things first, navigate to the AWS EC2 Console

Next we will want to Launch an Instance. Let's select the **Ubuntu 20.04 (HVM), SSD Volume Type**.

Select the instance type t2.micro as it falls under the free tier and has enough processing power to handle the job.

## Configure Instance

- The default settings Amazon provide are suitable unless the user already has created a

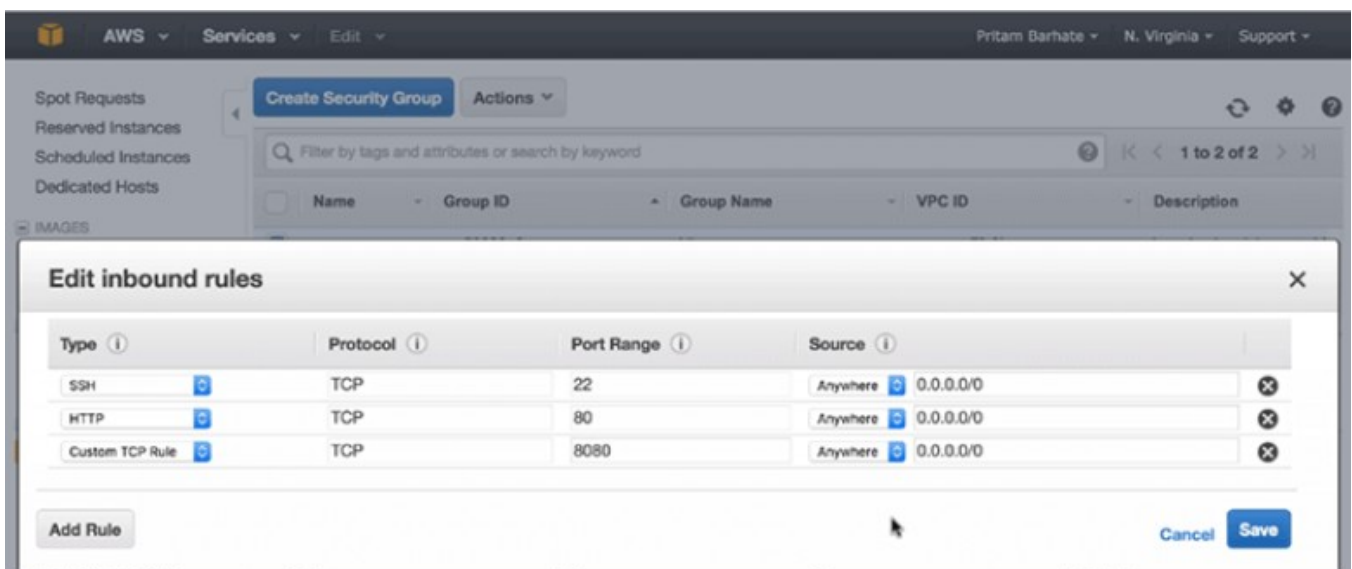
## Add Storage

- 8 GB recommended

## Configure Security group

As we are going to use a web server in addition to regular SSH port 22 we will need ports 80 and 8080 to be open.

- Create new security group called 'Nginx'



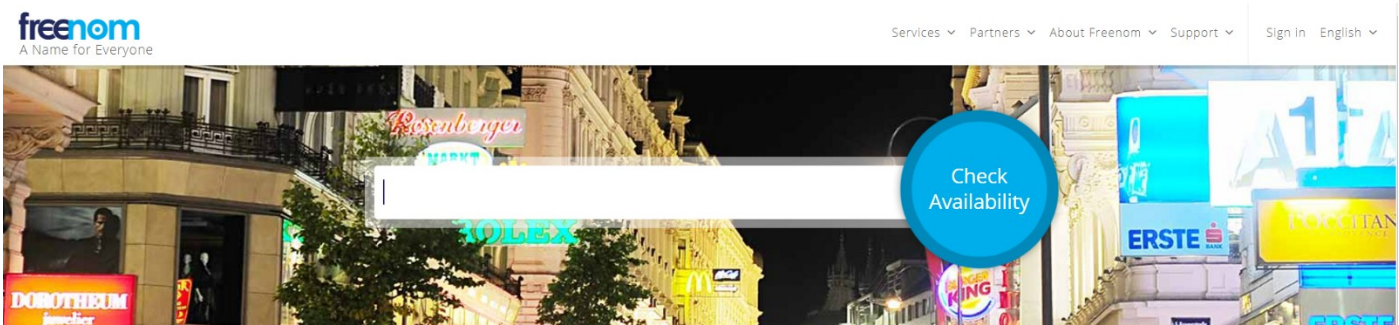
## Review & Launch instance

- Download .pem key to SSH into our machine.
- Navigate to Elastic Ip addresses under Network & Security and assign our newly created instance a public ip address. e.g 32.123.55.2

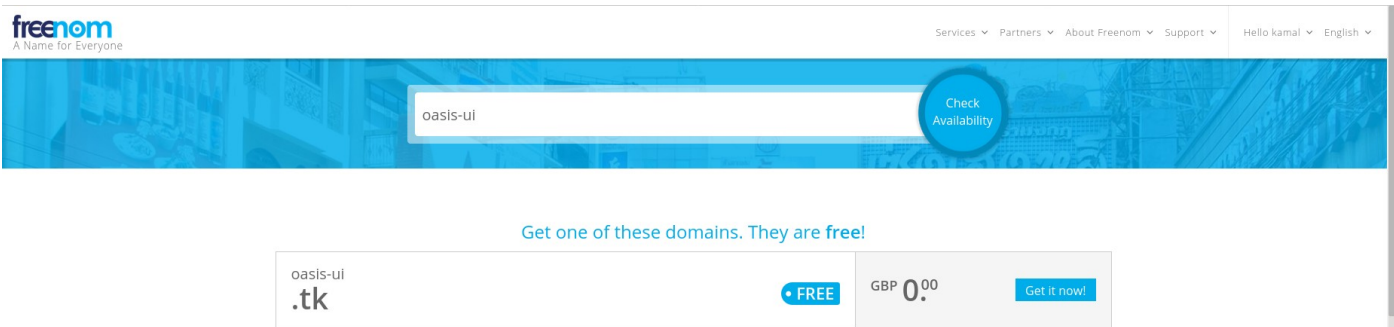
# Freenom Domain setup

Freenom is an online service that lets you get a free domain.

Visit the site <https://freenom.com/>

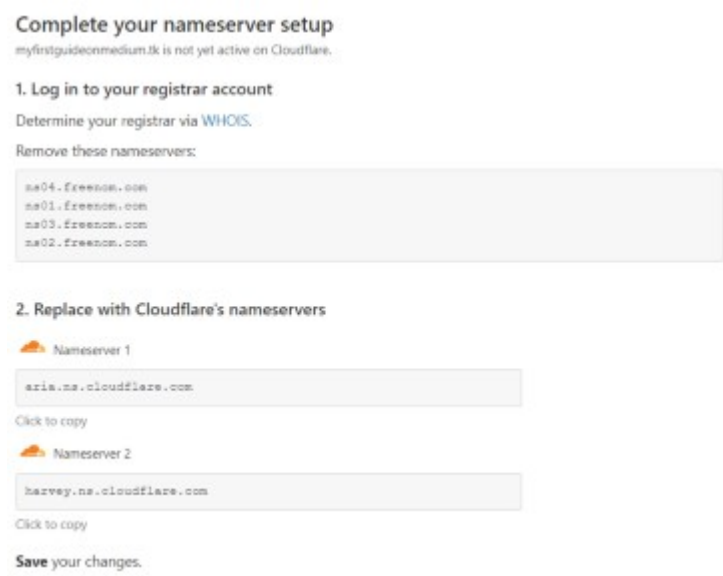


Enter a domain name that you wish to have, add it to cart and select it for 12 months it's free for the first year.



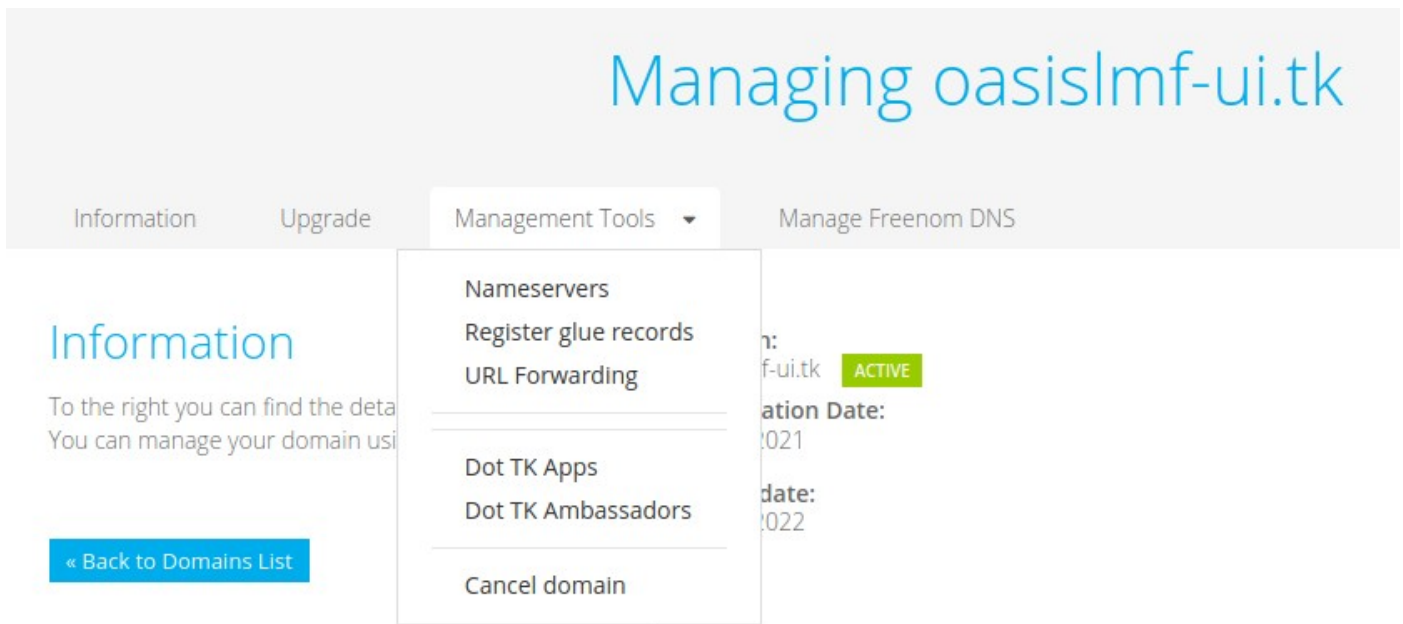
Our next step is to add our site to the Cloudflare service, so that we can have access to multiple features that will help with the cerificate setup

Enter the cloudflare dashboard and click on “add site”, and add your newly registered domain name.



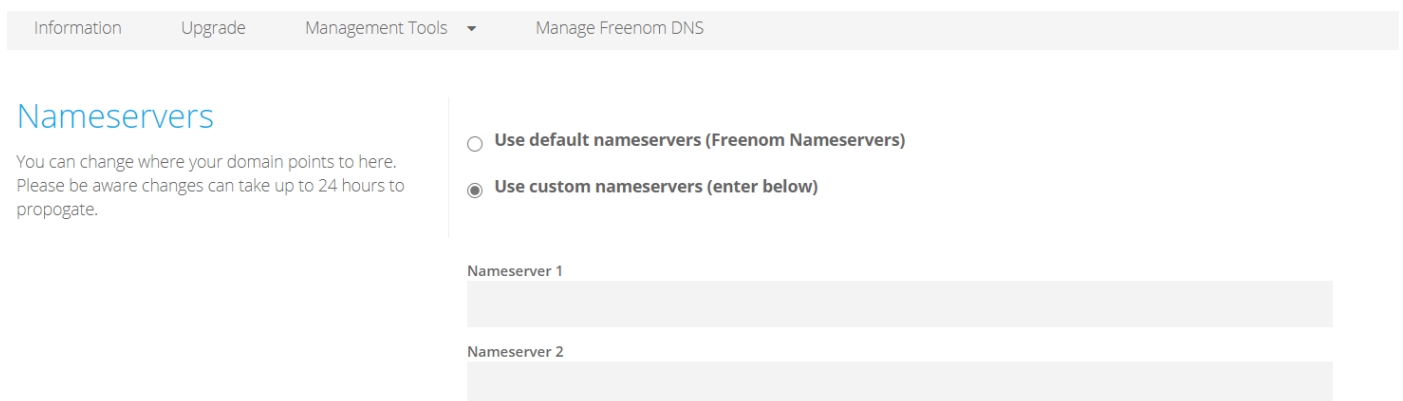
# Freenom Domain setup

To allow cloudflare to have access to our domain, we need to change the default nameservers in our freenom control panel, replacing them with the new ones offered by CloudFlare.



From the freenom client area, enter Services → My Domains and click on Manage Domain. Then navigate to Management Tools → Nameservers

Here you must change the option to use custom nameservers and enter the ones provided by CloudFlare



Add record on Cloudflare to make our domain point to the IP address of our AWS ec2 server.

Now that we have correctly changed our nameservers to work alongside CloudFlare, we can finally add our IP address to let our server be reachable from the domain.

You'll firstly need to add an 'A' record, leaving the name blank and just using the IP address (Change the proxy status to DNS only by clicking it)

**A few more steps are required to complete your setup.**[Hide](#)

✓ Add an A, AAAA, or CNAME record for your **root domain** so that **oasislmf-ui.tk** will resolve.



✓ Add an MX record for your **root domain** so that mail can reach **@oasislmf-ui.tk** addresses.

DNS management for **oasislmf-ui.tk**

+ Add record

Q Search DNS Records

⋮ Advanced

Type	Name	Content	TTL	Proxy status	
A	*	<input type="text"/>	Auto	 DNS only	<a href="#">Edit ▶</a>
A	oasisl	<input type="text"/>	Auto	 DNS only	<a href="#">Edit ▶</a>

Insert public ip address we assigned to the AWS instance

And also a CNAME to point all existing wildcards names to the same IP, like on the following picture.

DNS management for **oasislmf-ui.tk**

+ Add record

Q Search DNS Records

⋮ Advanced

**\*.oasislmf-ui.tk** is an alias of **insert ip address here**.

Type

Name

Target

TTL

Proxy status

CNAME ▼

Auto ▼

DNS only

Cancel

Save

# Setup of the server

## Initial setup of the server

Now we can start doing some work on our server we created earlier.

Given we now have a fresh installation of ubuntu 20.04, let's start by updating our software & dependencies.

```
apt update && sudo apt upgrade -y
```

Create a new user and grant the ability to run commands as root by adding to the sudoers list.

```
sudo adduser yourusername
```

```
sudo usermod -aG sudo yourusername
```

Now let's install docker, and add our user to the docker group in order to ensure we can use launch commands.

```
sudo groupadd docker
```

```
sudo usermod -aG docker yourusername
```

A quick reboot is needed to continue with our reverse proxy setup

```
sudo reboot
```



# Nginx server & reverse proxy setup

We can now move forward to the setup of our nginx server. I prefer to install nginx directly onto the machine instead of using a docker container. We will launch the OasisLMF UI docker container on a seperate AWS ec2 instance later.

```
sudo apt install nginx -y
sudo systemctl enable nginx
sudo unlink /etc/nginx/sites-enabled/default
sudo touch proxy_config.conf ← This is our main configuration file

cd /etc/nginx/sites-enabled
sudo ln -s ../sites-available/proxy_config.conf
```

With the commands listed above we will install nginx, enable automatic start at reboot, and setup our proxy configuration.

The usage of ln -s command is to create a symbolic link so that both sites-available and sites-enabled will see the same file and we can edit only one of them to modify both.

```
cd /etc/nginx/sites-available/
sudo nano proxy_config.conf
sudo unlink /etc/nginx/sites-enabled
```

Edit as follows:

```
server {
    listen 443;
    server_name ui.oasislmf-ui.tk;location / {
    proxy_pass http://localhost.com:8001;
    }
}server {
    listen 443;
    server_name oasislmf-ui.tk;location / {
    proxy_pass http://localhost:80/;
    }
}
```

Now let's check our configuration is correct and restart nginx to apply the changes. Bare in mind we will be changing this later to to represent our local machine running the OasisLMF UI container.

```
Sudo service nginx configtest
sudo service nginx restart
```

# Use of certbot to get SSL certificate

In this section we will use certbot with nginx & CloudFlare plugins to get a SSL certificate valid for both our name + wildcard name \*oasislmf-ui.tk

We'll need to obtain the the renewal manually by inserting a txt challenge in our DNS record from the CloudFlare dashboard

Firstly, let's install certbot and required plugins

```
sudo snap install core; sudo snap refresh core
sudo snap install --beta --classic certbot
sudo ln -s /snap/bin/certbot /usr/bin/certbot
sudo snap set certbot trust-plugin-with-root=ok
sudo snap install --beta certbot-dns-cloudflare
sudo snap connect certbot:plugin certbot-dns-cloudflare
```

Navigate back to the CloudFlare dashboard open and run the following command in terminal.

```
certbot certonly \
--manual \
--preferred-challenges dns \
-d oasislmf-ui.tk \
-d "*.oasislmf-ui.tk" \-i nginx
```

Now certbot will ask you to put a DNS TXT record to provide a challenge to demonstrate that you are the real owner of that domain.

Once on the CloudFlare dashboard you can add the TXT challenge

\*note You'll have to do this 2 times. DO NOT delete the first challenge when adding the second, even if they have the same name, otherwise the challenge won't succeed.

DNS management for **oasislmf-ui.tk**

+ Add record

Q Search DNS Records

⋮ Advanced

**\_acme-challenge.oasislmf-ui.tk** has a record with content **CmNIXeF5PT2355uEh6\_NTqBd544445546mjm1ZjBCUbgV7M2a7lHnU.**

Type	Name	TTL
<div>TXT</div>	<div>_acme-challenge</div>	<div>Auto</div>

Content

CmNIXeF5PT2355uEh6\_NTqBd544445546mjm1ZjBCUbgV7M2a7lHnU

Cancel

Save

# Nginx Configuration

Now we should enable auto redirect of all non https requests to https.

Create this file if it is not already present, otherwise use the second command to edit it

```
sudo touch /etc/nginx/conf.d/domain-name.conf
sudo nano /etc/nginx/conf.d/domain-name.conf
```

Add this configuration replacing with your domain names.

```
server {
    listen localhost;
}server {
    listen 80 default_server;
    listen [::]:80 default_server;
    root /var/www/html;
    server_name myfirstguideonmedium.tk www.myfirstguideonmedium.tk;listen 443 ssl;
# managed by Certbot# RSA certificate
    ssl_certificate /etc/letsencrypt/live/oasislmf.ui.tk/fullchain.pem;
# managed by Certbot
    ssl_certificate_key /etc/letsencrypt/live/ui.oasislmf-ui.tk/privkey.pem;
# managed by Certbotinclude /etc/letsencrypt/options-ssl-nginx.conf;
# managed by Certbot
# Redirect non-https traffic to https
    if ($scheme != "https") {
        return 301 https://$host$request_uri;
    } # managed by Certbot
}
```

Restart the nginx server to apply the changes

```
sudo service nginx restart
```

Check that everything is working accordingly by navigating to the main domain through https: <https://yourdomain.tk> and the wildcard domain like https://test.yourdomain.tk

You should see the nginx default page like this:

## Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to [nginx.org](https://nginx.org).  
Commercial support is available at [nginx.com](https://nginx.com).

*Thank you for using nginx.*

We've successfully connected our SSL certificates

# Nginx Configuration

We will now create another instance that will be running the OasisLMF user interface docker container.

Launch an Instance. Let's select the **Ubuntu 20.04 (HVM), SSD Volume Type**.

Select the instance type t2.large

## Configure Instance

- Ensure the VPC is the same as the nginx VPC
- Private Subnet Mask

## Add Storage

- 20 GB recommended

## Configure Security group

Inbound rules					Edit inbound rules
Type	Protocol	Port range	Source	Description - optional	
All traffic	All	All	0.0.0.0/0	-	
SSH	TCP	22	0.0.0.0/0	-	

Inbound rules

Outbound rules

Tags

Outbound rules

Edit outbound rules

Type	Protocol	Port range	Destination	Description - optional
All traffic	All	All	0.0.0.0/0	-

Review and launch the instance.

This machine will be assigned a local ip address e.g. 10.10.0.99 depending on your subnet mask.

To check if the machines are on the same VPC → Navigate to the NGINX instance terminal and ping the newly created instance **ping 10.10.0.99**

# Nginx Configuration

Now we can go ahead and setup the OasisLMF UI containers on 10.10.0.99

For this example I'll be using the OasisEvaluation repository.

```
apt update && sudo apt upgrade -y
```

```
git clone https://github.com/OasisLMF/OasisEvaluation.git
```

```
cd OasisEvaluation
```

```
./install.sh
```

```
docker ps -a ( list that the containers are up and running)
```

We will need to navigate back the Nginx instance and change the proxy-config.conf to recognise the docker machine. We need a new entry to the nginx configuration, to redirect the requests coming from ui.oasislmf-ui.tk to our docker machine.

```
sudo nano /etc/nginx/sites-available/proxy-config.conf
```

Add the following replacing the ip address and domain name with yours.

```
instream ann server {
    server 10.10.0.99:8080 fail_timeout=0;
}

server {
    listen 80;
    listen [::]:80;
    rewrite ^/shiny/$ $scheme://$http_host/shiny/ permanent;
    server_name ui.oasislmf-ui.tk; location / {
        proxy_pass http://10.10.0.99:8080;
    }
}

server {
    listen 443;
    server_name ui.oasislmf-ui.tk;
    location / {
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection "upgrade";
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header Host $http_host;
        proxy_pass http://localhost:8080;
        proxy_redirect http://10.10.0.99:8080 http:// Enter PUBLIC IP ADDRESS NGINX /;
        if (!-f $request_filename) {
            proxy_pass http://app_server;
            break;
        }
    }
}
```

← Set app\_server with local ip of docker machine

← Rshiny Websocket variable

← Subdomain to attach the user interface URL

← proxy\_pass redirect

```
sudo service nginx restart
```

# Troubleshooting (adding)

Once that has been configured correctly you should now be able to type in your url within your web browser and be presented with the OasisLMF UI splash screen.

