**PUBLISH AN APP (NODEJS AND ANGULAR) WITH NGINX IN CENTOS 8 (HOST DIGITAL OCEAN)**

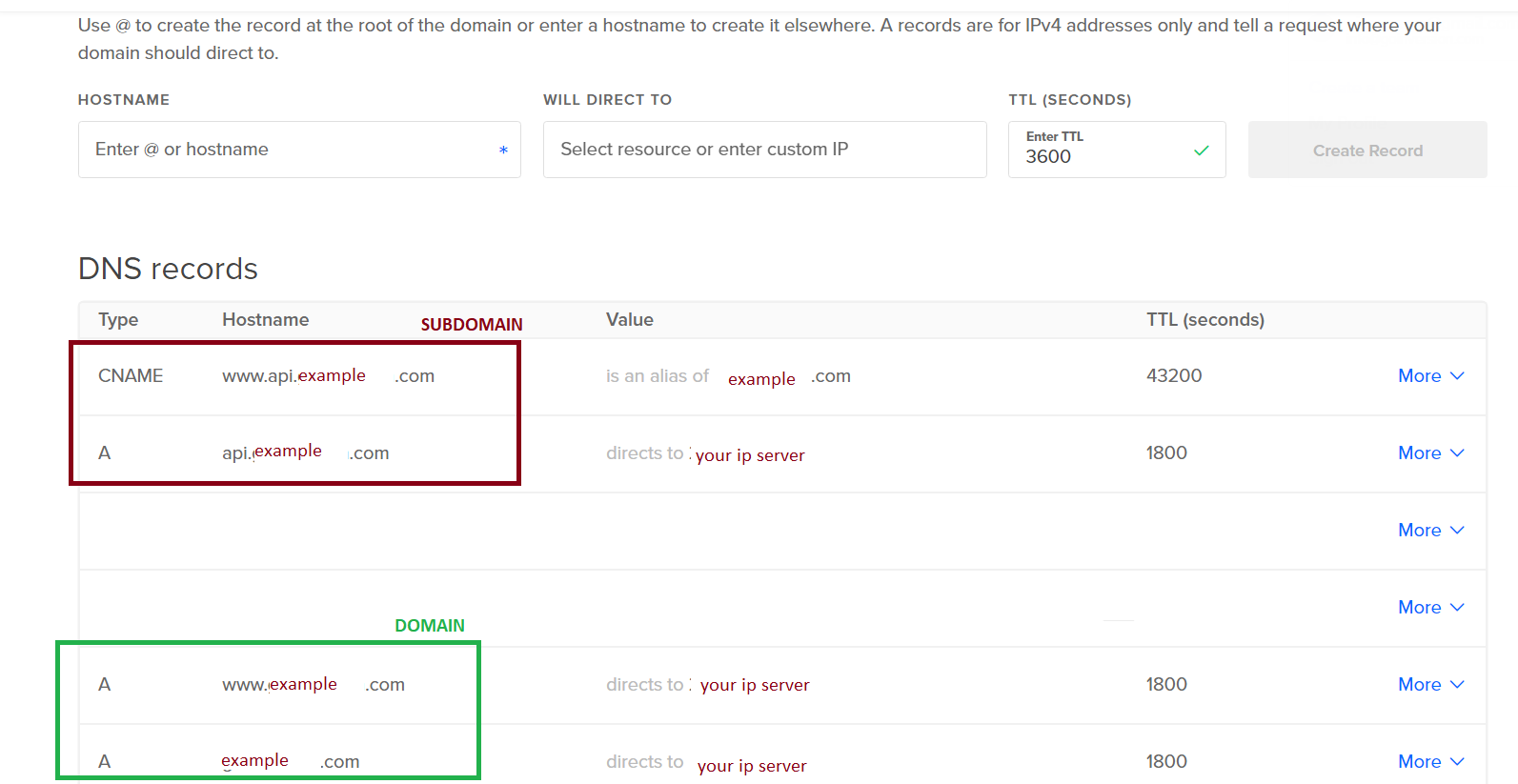
**THE APP HAS PUBLISHED IS DEVELOPED IN:**

**BACKEND:** FRAMEWORK NESTJS (NODEJS)

**FRONTEND:** ANGULAR 9

1. **Requirements:**

* Have the backend running and have done the ng build –prod to get the folder dist
* Have previously created domain and subdomains example:



1. **Create folder for Frontend**

We create the folder where we will have to place the files in the dist folder of the frontend to change example.com for your domain:

$ sudo mkdir -p /var/www/example.com/html

We copy all the files from the dist folder (files only) to our previously created address and rename it as html:

$ cp -R dist     /var/www/var/www/example.com/html

We grant permissions (in $ USER put the name of the current user example root: root):

$ sudo chown -R $USER:$USER /var/www/your\_domain/html

El siguiente comando permitirá presentar el root de su documento personalizado como contenido HTTP:

$ chcon -vR system\_u:object\_r:httpd\_sys\_content\_t:s0 /var/www/your\_domain/

1. **Installation of NGINX in Centos:**  
   Run the following commands necessary for the proper installation and operation of nginx:

$ sudo yum install epel-release

$ sudo yum install nginx

We permanently enable http and https connections:

$ sudo firewall-cmd --permanent --add-service=http

$ sudo firewall-cmd --permanent --add-service=https

We restart the firewall:

$ sudo firewall-cmd –reload

Use the editor of your choice for editing files in the Shell in this case vim will be used:

We go to the folder conf.d in the following path:

$ cd /etc/nginx/conf.d

Create a file with .conf extension example:

$ vim example.com.conf

Inside the file we configure the following to change the example data for the domain and subdomain of your application:

#Frontend

server{

listen 80;

       server\_name example.com  www.example.com;

       location / {

                root /var/www/example.com/html;

                index index.html index.htm index.nginx-debian.html;

                try\_files $uri $uri/ /app/index.html;

       }

}

#Backend

server {

listen: 80;

       server\_name api.example.com  www.api.example.com;

       location / {

                proxy\_pass http://localhost:3001;

                proxy\_http\_version 1.1;

                proxy\_set\_header Upgrade $http\_upgrade;

                proxy\_set\_header Connection 'upgrade';

                proxy\_set\_header Host $host;

                proxy\_cache\_bypass $http\_upgrade;

       }

}

We save the changes and execute the following commands:

$ sudo systemctl start nginx

$ sudo systemctl enable nginx

We check the status of Nginx:

$ sudo systemctl status nginx

We activate the following boolean values:

$ sudo setsebool -P httpd\_can\_network\_relay on

$ sudo setsebool -P httpd\_can\_network\_connect on

We restart Nginx:

$ sudo systemctl restart nginx

1. **SSL Certificates with LetsEncrypt**

To have the ssl certificates in our domains and subdomains we execute the following commands:

$ sudo dnf install certbot

$ sudo dnf install certbot python3-certbot-nginx

We add certificates to domains and subdomains to replace example.com with your domain:

$ sudo certbot --nginx -d example.com -d www.example.com -d api.example.com -d www.api.example.com

To automatically renew the certificate, copy and paste the following command

$ echo "0 0,12 \* \* \* root python3 -c 'import random; import time; time.sleep(random.random() \* 3600)' && certbot renew -q" | sudo tee -a /etc/crontab > /dev/null

It will not request an email we enter it and Enter.

We accept the terms with A and Enter.

We do not add our mail to the Electronic Frontier foundation we put N and Enter.

We wait for the validation and the SSL would already be configured in your domains and subdomain.