

Dominios y EC2+Route53(DNS)+NginX(WebS)

Autor: Alberto Herrera Poza
Github: <https://github.com/AlbertoHP>

Introducción

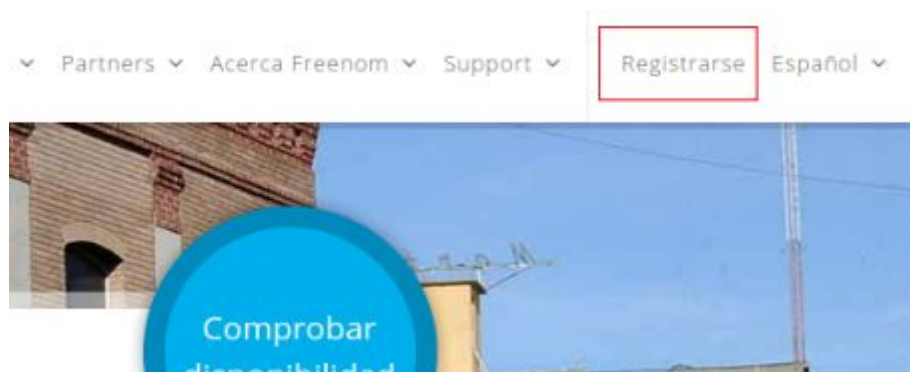
El presente documento tiene la finalidad de desarrollar el proceso necesario para obtener un dominio y configurar un DNS para apuntarlo hacia un VPS en AWS EC2.

1. Obtención de un dominio

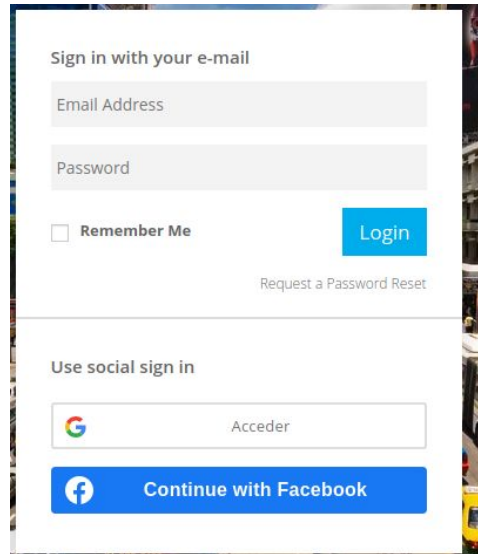
Un dominio, es un nombre alfanumérico que puede ser asignado a una dirección IP, esto con la intención de volver más amigable la relación humano-computador, para poder recordar las distintas direcciones del inmenso número de aplicaciones web que hay hoy en el mercado.

Para la obtención de uno, es necesario recurrir a un proveedor de dominios. Estos proveen por lo general de una variedad de servicios, entre ellos un servicio de nombre de dominio (DNS). Un DNS, es básicamente una lista de hash (clave-valor) en la cual se pueden configurar las distintas variaciones del dominio en cuestión, y asignarlas a una dirección IP específica.

Freenom, es un proveedor de dominios gratuitos, y es el que será utilizado para desarrollar el objetivo de este documento. Para esto es necesario ir a la dirección <https://www.freenom.com> y registrarse dando click en la esquina superior derecha en '**Registrarse**' como se muestra en la imagen a continuación.

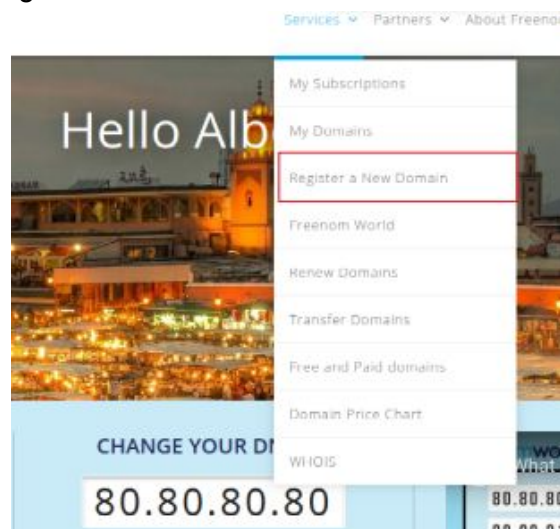


Este proveedor de dominios facilita el registro a través de distintas OAuth API's, como por ejemplo Facebook y Google como se muestra a continuación.

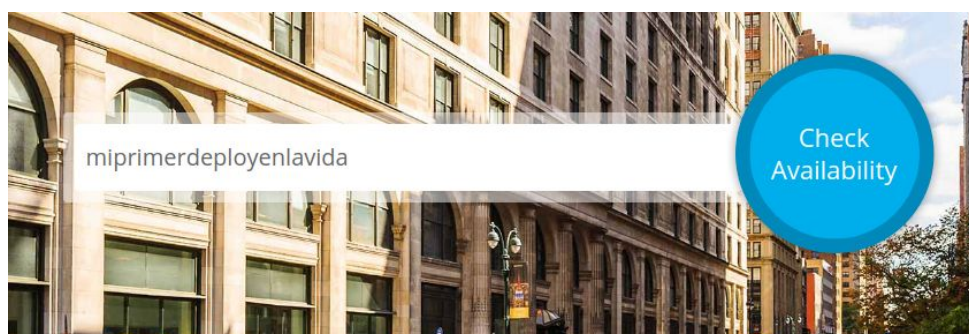


The image shows a login and social sign-in interface. At the top, it says "Sign in with your e-mail". Below this are two input fields: "Email Address" and "Password". There is a checkbox labeled "Remember Me" and a blue "Login" button. A link "Request a Password Reset" is located below the "Login" button. Below the login section, it says "Use social sign in". There are two buttons: one with the Google logo and the text "Acceder", and another with the Facebook logo and the text "Continue with Facebook".

Una vez realizado el registro, es necesario ir a la sección de registro de dominio como se indica en la siguiente imagen.



El dominio a utilizar en este documento es 'miprimerdeployenlavid', para esto es necesario llenar el input y dar en '**Check availability**' como se muestra en la siguiente imagen.



Freenom, permite además comprar dominios con Top Level Domain (TLD) comunes como .com, .cl, .co, etc. TK, es el TLD que utilizaremos en esta guía como se muestra en la siguiente imagen.

Dando en **'Checkout'** podemos pasar a la revisión de nuestra compra (que por cierto no tiene costo). Aquí es necesario hacer un par de configuraciones, Lo primero es activar el DNS dando a **'Use DNS'** como se muestra en la siguiente imagen.

Dando en **'Use your own DNS'** podremos activar la configuración para utilizar AWS Route53 como nuestro DNS personalizado.

Aquí, agregaremos por ahora miprimerdeployenlaveda.tk y www.miprimerdeployenlaveda.tk, a la dirección 0.0.0.0 como se muestra en la imagen.

Use your new domain

Period: 3 Months @ FREE

Forward this domain or Use DNS

Use Freenom DNS Service Use your own DNS

Enter your A record here

Nameserver	miprimerdeployenlaveda	IP address	0.0.0.0
Nameserver	www.miprimerdeployen	IP address	0.0.0.0

Continue

Por último, es necesario configurar el periodo que estará activo, Freenom a la fecha de creación de este documento ofrece hasta 12 meses sin costo para un dominio de TLD no común o comercial. En este caso, se utilizará por defecto 3 meses. Para continuar, es necesario dar en el botón **'Continue'** como se muestra en la imagen, se mostrará un resumen de la compra en general, donde es necesario dar en aceptar terminos y condiciones, y luego para seguir, dar en **Complete Order**.

REVIEW & CHECKOUT

Description	Price
Domain Registration - miprimerdeployenlaveda.tk	\$0.00USD
Subtotal:	\$0.00USD
Total Due Today:	\$0.00USD

☒ I have read and agree to the Terms & Conditions Complete Order

Finalmente, para finalizar la obtención del dominio, se debería desplegar una pantalla como la de la imagen a continuación.

Order Confirmation

Thank you for your order. You will receive a confirmation email shortly.

Your Order Number is: 7193453817

If you have any questions about your order, please open a support ticket from your client area and quote your order number.

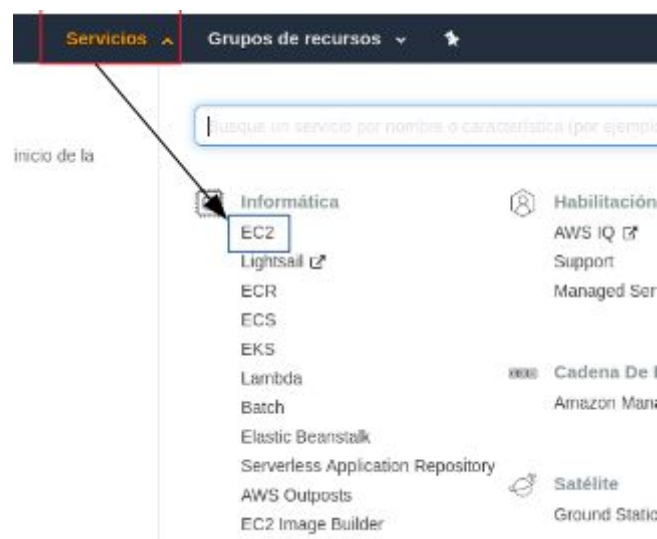
Click here to go to your Client Area

2. Obtención de VPS (AWS EC2)

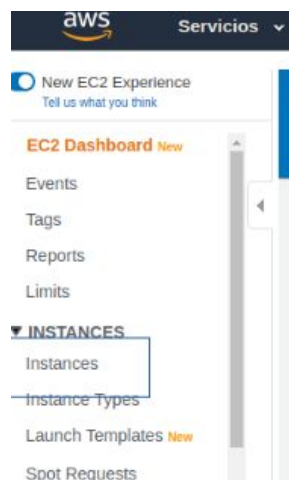
Un servidor virtual privado (VPS) es una partición de máquina, a nivel de software, que habilita la virtualización de hardware, pudiendo así, comportarse como una máquina totalmente independiente, que incluso puede tener acciones a nivel de una BIOS virtual, que le permite además, poder reiniciarse, apagarse o encenderse de manera independiente a su máquina padre o host.

Existen varios proveedores de VPS (hostings), entre ellos están Google Cloud Services (GCS), Digitalocean, Amazon Web Services (AWS), etc. Para el desarrollo de este documento se utilizará AWS, con su módulo específico para hosting EC2.

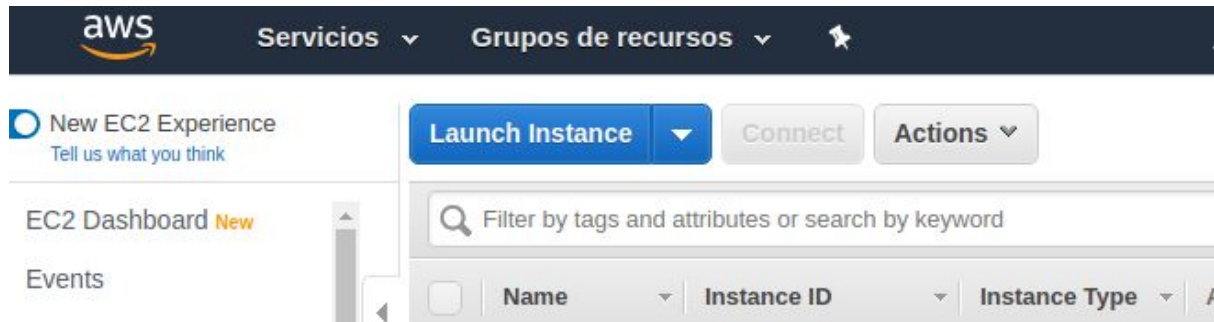
Para esto es necesario ir a la dirección <https://aws.amazon.com/es/console/> e iniciar sesión con las credenciales respectivas. Una vez dentro, ir a EC2 como se muestra en la siguiente imagen.



Luego es necesario ir a 'Instances' como se muestra en la siguiente imagen.



Para crear un nuevo VPS, es necesario dar en **'Launch Instance'** como se muestra en la siguiente imagen.

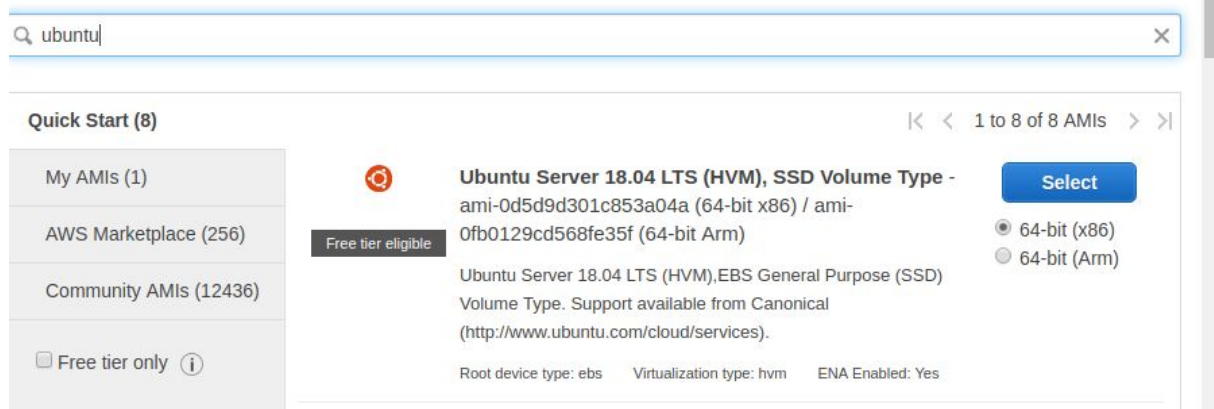


Para este caso, utilizaremos un Ubuntu server pre configurado por AWS.

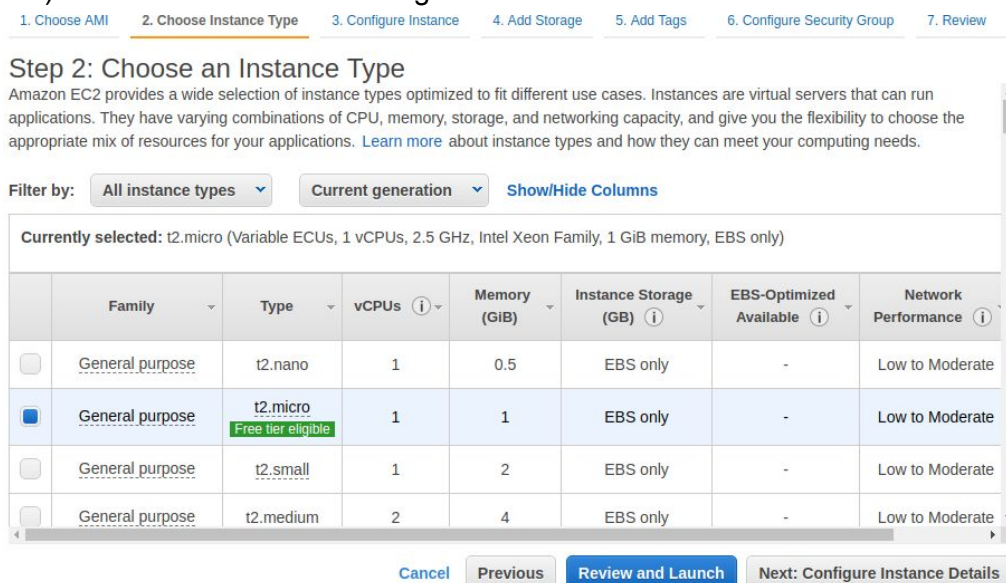
Step 1: Choose an Amazon Machine Image (AMI)

[Cancel and Exit](#)

An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. You can select an AMI provided by AWS, our user community, or the AWS Marketplace; or you can select one of your own AMIs.



El paso que sigue luego de dar en **'Select'** es definir el tipo de instancia que queremos, en la cual es posible elegir número de procesadores, memoria RAM, espacio de disco duro entre otros. En este caso se dejará por defecto debido a que se utilizara el nivel gratuito (Free Tier) como se muestra en la imagen.



Luego es necesario dar en **'Next: Configure Instance Details'** donde es posible configurar detalles un poco más específicos de nuestro VPS, como su red o redes locales, el número de instancias que se quieren desplegar, entre otras configuraciones que por ahora se pasaran por alto. Luego, es necesario dar en **'Next: Add Storage'** para configurar el espacio que estará disponible como disco duro en el VPS. En este caso se dejará por defecto debido al Free Tier, para seguir dar en **'Next: Add Tags'**, esto último, es una funcionalidad bastante útil de AWS, que permite agregar elementos clave-valor (Hash) a nuestros VPS, para poder, en el caso de tener múltiples instancias en nuestra arquitectura, identificarlas mediante por ejemplo, una etiqueta que tenga la clave **'SERVER_TYPE'** y el valor **'DEVELOPMENT'**, así, podríamos facilmente saber que esta instancia específica de EC2, corresponde a un servidor de tipo desarrollo. Es este caso, se agregara esta misma clave, y además otra con la clave **'SERVER_OWNER'**, y el valor **'Alberto Herrera'** como se muestra en la siguiente imagen.

1. Choose AMI

2. Choose Instance Type

3. Configure Instance

4. Add Storage

5. Add Tags

6. Configure Security Group

7. Review

Step 5: Add Tags

A tag consists of a case-sensitive key-value pair. For example, you could define a tag with key = Name and value = Webserver.

A copy of a tag can be applied to volumes, instances or both.

Tags will be applied to all instances and volumes. [Learn more](#) about tagging your Amazon EC2 resources.

Key (128 characters maximum)	Value (256 characters maximum)	Instances <small>i</small>	Volumes <small>i</small>	
<input type="text" value="SERVER_OWNER"/>	<input type="text" value="ALBERTO HERRERA"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="button" value="X"/>
<input type="text" value="SERVER_TYPE"/>	<input type="text" value="DEVELOPMENT"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="button" value="X"/>

Add another tag

(Up to 50 tags maximum)

Cancel

Previous

Review and Launch

Next: Configure Security Group

Por último, es necesario configurar el grupo de seguridad bajo el cual se determinarán los puertos que este VPS tendrá abiertos, para esto es necesario dar en **'Next: Configure Security Group'** en la pantalla de arriba. Luego, dando en **'Add Rule'** se agregara una nueva fila, donde se abrirán los puertos del **'0-4000'** y surtirá efecto de cualquier parte que venga la petición como se muestra en la siguiente imagen.

1. Choose AMI 2. Choose Instance Type 3. Configure Instance 4. Add Storage 5. Add Tags 6. Configure Security Group

Step 6: Configure Security Group

Security group name: launch-wizard-4

Description: launch-wizard-4 created 2019-12-07T19:25:35.878-03:00

Type	Protocol	Port Range	Source	Description
SSH	TCP	22	Custom 0.0.0.0/0	e.g. SSH for A
Custom TCP F	TCP	0-4000	Anywhere 0.0.0.0, ::/0	e.g. SSH for A

Add Rule

Warning

Rules with source of 0.0.0.0/0 allow all IP addresses to access your instance. We recommend setting security group rules to allow access from known IP addresses only.

Cancel Previous Review and Launch

Para finalizar y ver un resumen de la configuración de la instancia dar en **'Review and Launch'**, luego de revisar que toda la configuración esta correcta, dar en **'Launch'**. Aquí, aparecera una pantalla que nos preguntará sobre qué llave usar para la conexión de bash segura (SSH), donde en el selector pondremos **'Create a new key pair'** para crear una nueva llave, la cual llevará el nombre de **'aws_ssh_key'**, luego dar en **'Download Key Pair'** para descargar la llave. Por último dar en **'Launch Instances'** para levantar el VPS.

Select an existing key pair or create a new key pair

A key pair consists of a **public key** that AWS stores, and a **private key file** that you store. Together, they allow you to connect to your instance securely. For Windows AMIs, the private key file is required to obtain the password used to log into your instance. For Linux AMIs, the private key file allows you to securely SSH into your instance.

Note: The selected key pair will be added to the set of keys authorized for this instance. Learn more about [removing existing key pairs from a public AMI](#).

Create a new key pair

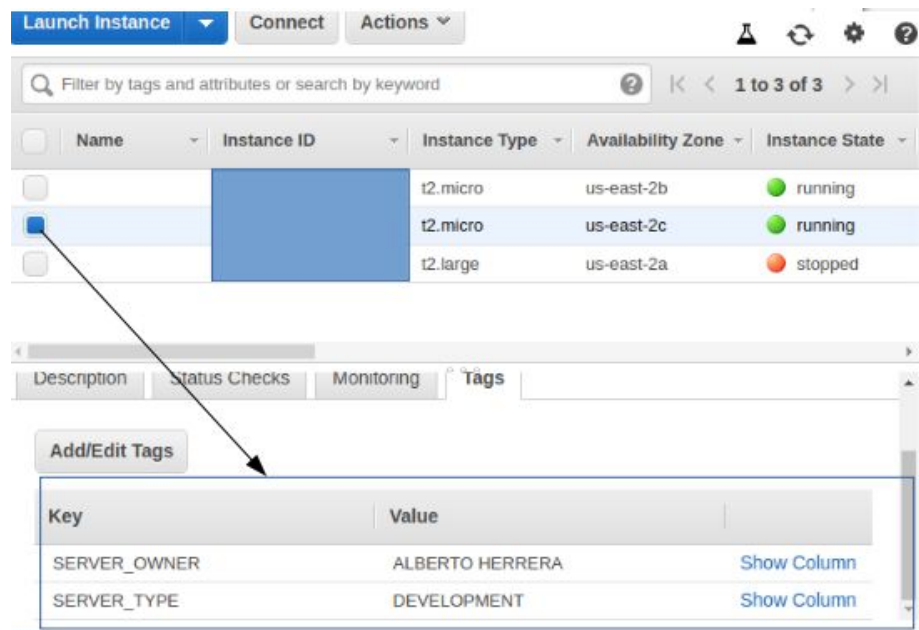
Key pair name
aws_ssh_key

Download Key Pair

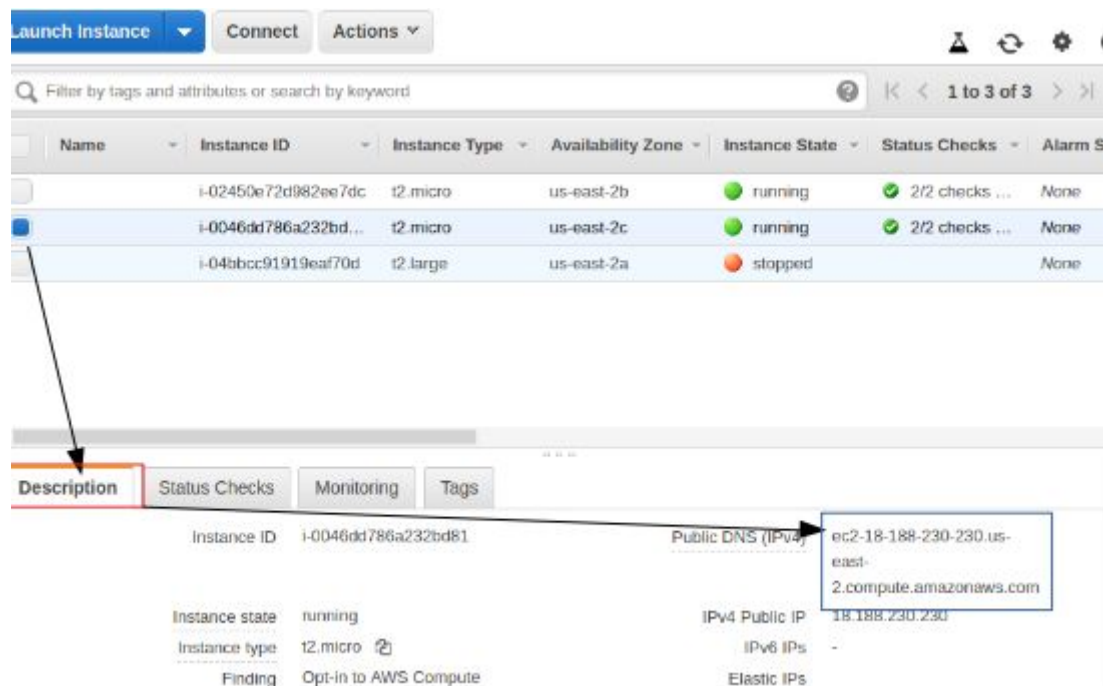
You have to download the **private key file** (*.pem file) before you can continue. **Store it in a secure and accessible location.** You will not be able to download the file again after it's created.

Cancel Launch Instances

Se desplegará información relativa al estado del levantamiento de la instancia de EC2, para ver las instancias dar en **'View Instances'**. Aquí se vuelve útil nuevamente las etiquetas de EC2, como se ve en la imagen, hay varias instancias con ID's distintas, y aquí, las etiquetas nos ayudan a saber cual es el VPS que recién se creo.



Ahora para poder ingresar al VPS a través de nuestra llave que creamos y descargamos es necesario modificar el fichero con extensión **.pem**, que en este caso, corresponde a **aws_ssh_key.pem**, con el comando **chmod 400 aws_ssh_key.pem**, posteriormente basándose en la última imagen, ir a la pestaña de **'Description'** y ahí buscar el nombre de DNS con el que se está cubriendo el VPS.



Mediante una consola bash, y con el siguiente comando podemos conectarnos al VPS creado, `ssh -i "aws_ssh_key.pem" ubuntu@ec2-18-188-230-230.us-east-2.compute.amazonaws.com`, este comando ssh, utiliza el flag `-i` para asignar un fichero `.pem` con configuración `chmod 400`, luego, es necesario asignar el usuario con el cual se intenta establecer conexión en el VPS, en este caso `ubuntu` (es necesario recordar que este VPS está con un ubuntu server pre configurado por AWS, en el caso de hacer un VPS desde cero, o bien querer asignar usuarios adicionales a un VPS con una imagen pre configurada, entonces, este usuario variará), junto al usuario con un **arroba (@)** va la dirección DNS o bien IP del VPS, en este caso, utilizaremos `ec2-18-188-230-230.us-east-2.compute.amazonaws.com` que se obtuvo anteriormente.

```
→ Descargas ssh -i "aws_ssh_key.pem" ubuntu@ec2-18-188-230-230.us-east-2.compute
e.amazonaws.com
The authenticity of host 'ec2-18-188-230-230.us-east-2.compute.amazonaws.com (18
.188.230.230)' can't be established.
ECDSA key fingerprint is ████████████████████████████████████████████████████
Are you sure you want to continue connecting (yes/no)? █
```

Cuando un fichero `.pem` es utilizado por primera vez, el sistema nos consultará si es que estamos seguro de que la huella es correcta, y si deseamos continuar con la conexión, es necesario escribir **yes**, para que la conexión se establezca.

```
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

ubuntu@ip-172-31-41-86:~$ █
```

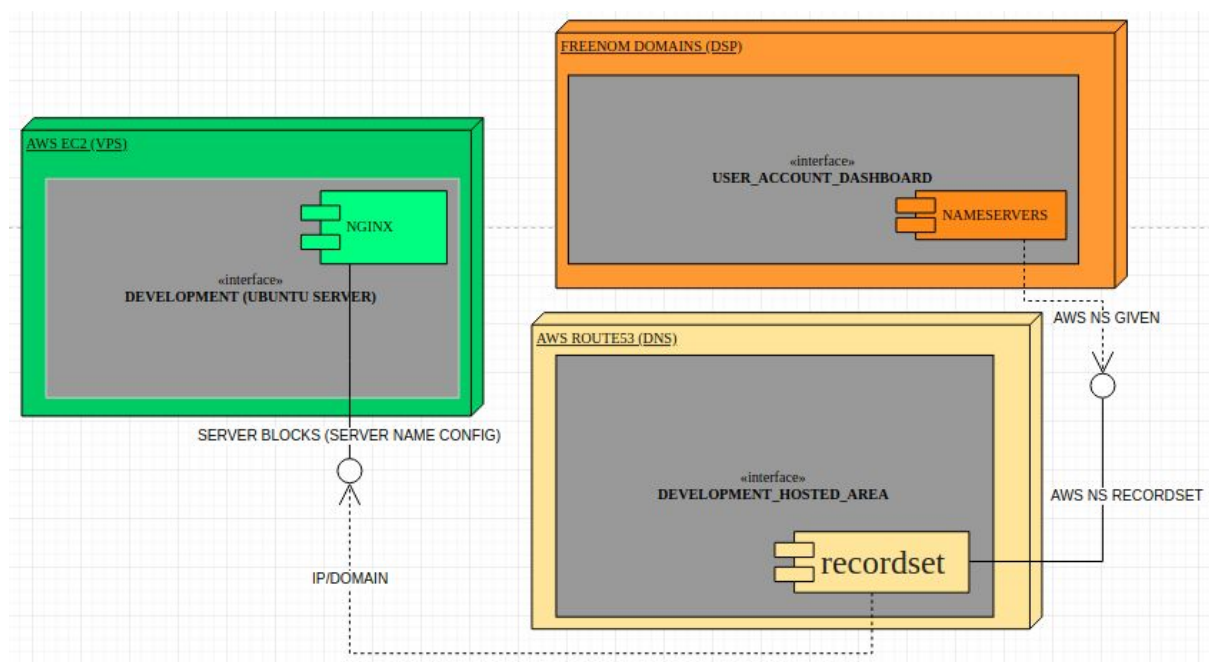
3. Configuración DNS (AWS Route53)

El sistema de nombres de dominio (DNS) es un sistema de nomenclatura jerárquica, su uso más conocido es el redireccionamiento de dominio a IP, y es precisamente esta utilidad la que se hace necesaria en la configuración de un servidor. Existen varios proveedores de DNS, entre ellos el mismo Freenom, Cloudflare, AWS Route53, etc. Para efectos de este documento, se utilizará en la configuración AWS Route53 como proveedor de DNS.

Hasta este punto, a modo de resumen, en nuestra arquitectura, existen los siguientes recursos:

- Dominio: Existe un dominio habilitado mediante **Freenom** como **proveedor de dominio**.
- VPS: Existe una máquina virtual provista por **AWS EC2**, la cual es accesible mediante su llave fichero **.pem**.

Con esto, entonces ya es posible agregar a la ecuación el **DNS**, el cual guiará las peticiones hacia nuestro **VPS**, actualmente solo hay un dominio disponible, **pero finalmente serán tres los dominios servidos por el mismo VPS**. A grandes rasgos, la arquitectura que la aplicación tendrá es la siguiente.



En la imagen anterior, existe un mainframe habilitado por la plataforma **AWS EC2**, la cual presenta una interfaz (VPS) que dentro contiene un componente de servicio web **NGINX**. Será este último quien mediante los bloques de servidor (**Server Blocks**) redireccione a los distintos espacios de trabajo (código fuente / contenido).

Ademas, como middleware, esta el mainframe de **AWS Route53**, la cual presenta una interfaz (**Hosted Zone**) que a través de un componente de DNS (**recordset** o conjunto de nombres de servidores AWS NS) permite que el proveedor de dominio, en este caso bajo el mainframe de **Freenom Domains**, el cual mediante la interfaz del dashboard habilitado para sus usuarios, es permite utilizar un componente para los **Nameservers** y asociarlos a dichos recordset de AWS NS.

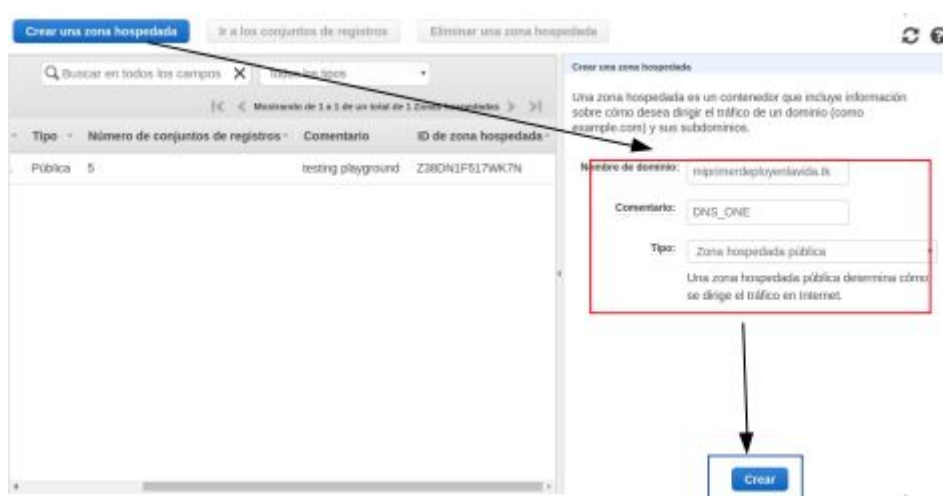
Agregaremos entonces esta configuración que en este caso se usa como un **middleware**, al DNS AWS Route53. Ir a entonces ahora en la misma consola de administración de AWS, al servicio Route53.



Ir a 'Zonas Hospedadas'.



Crear una zona hospedada, y configurarla con el dominio que hemos adquirido en este documento.



El comentario no es más que un metadato para poder identificar dicha zona hospedada. Como vemos, la configuración por defecto de Route53 cuenta con 4 servicios de nombre de servidor (NS), y además con una configuración de autenticación (SOA).

Volver a Zonas hospedadas

Crear un conjunto de registros

Importar archivo de zona

Q

Nombre del conjunto de registro

X

Cualquier tipo

☐ Solo con alias

☐ Solo ponderados

<<

<

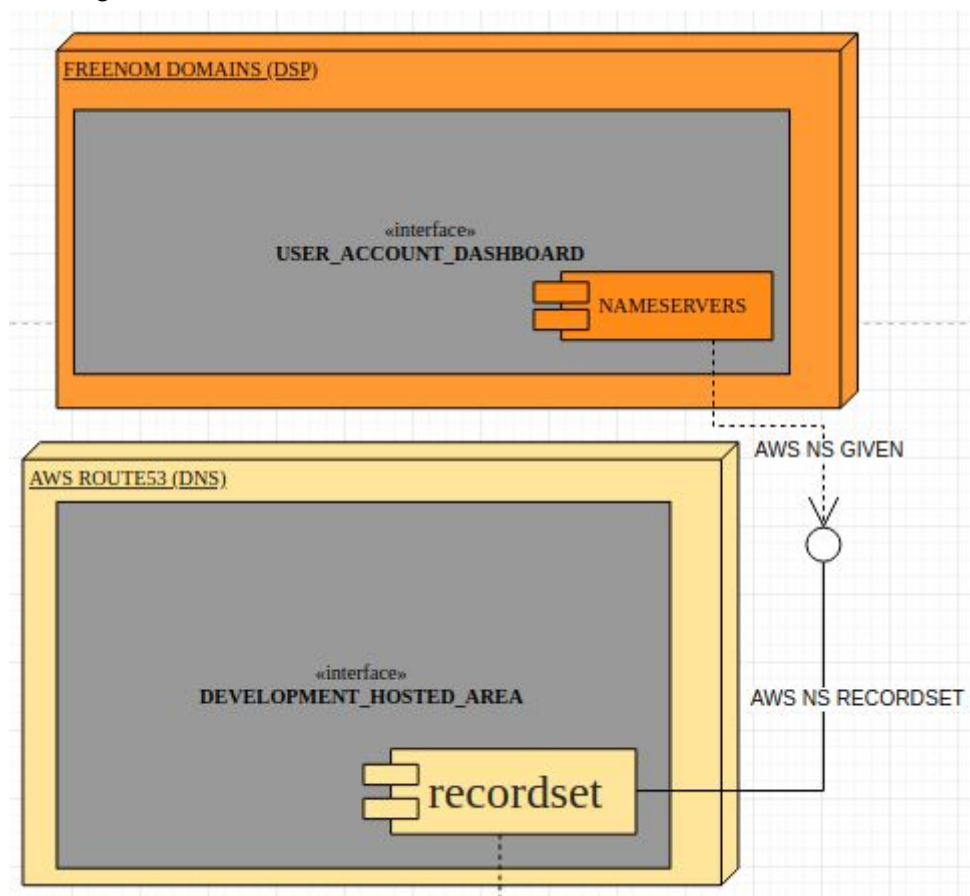
Mostrando de 1 a 2 de un total de 2 Conjuntos de registros

>

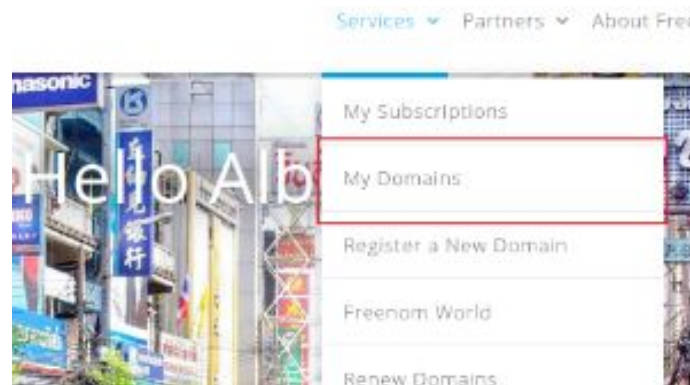
>>

<input type="checkbox"/>	Nombre	Tipo	Valor	Evaluar el estado
<input type="checkbox"/>	miprimerdeployenlavida.tk.	NS	ns-97.awsdns-12.com. ns-522.awsdns-01.net. ns-1860.awsdns-40.co.uk. ns-1286.awsdns-32.org.	-
<input type="checkbox"/>	miprimerdeployenlavida.tk.	SOA	ns-97.awsdns-12.com. awsdns-hostmaster.amazon.	-

Recordando entonces la arquitectura, estos nombre de servidor (**NS**) que **AWS Route53 provee**, serán consumidos por el mainframe de **Freenom Domains**, como se recuerda en la siguiente imagen.



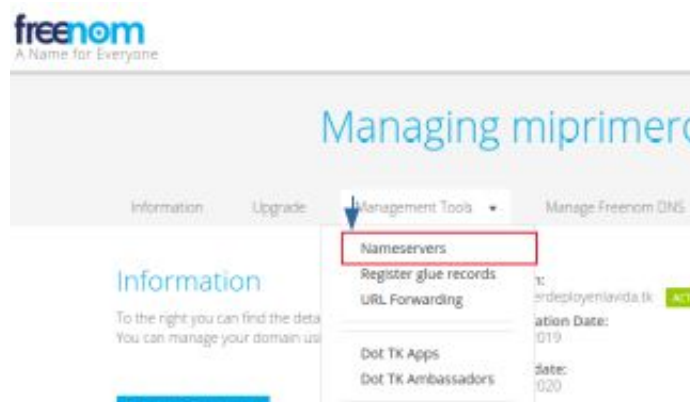
Ahora bien, se hace necesario ir al dashboard de configuración de Freenom Domains para poder agregar estos **NS**, ir a <https://my.freenom.com/clientarea.php> e ingresar con las credenciales respectivas.



Dar en **'Manage Domain'**.



Ir a **'Management Tools'** y dar en **Nameservers**.



Aquí dar en **'Use custom nameservers (enter below)'**, aquí es necesario ingresar los 4 NS provistos por la zona hospedada con Route53.

Volver a Zonas hospedadas Crear un conjunto de registros Importar ar

Nombre del conjunto de regi X Cualquier tipo Solo con alias Sol

Mostrando de 1 a 2 de un total de 2 Conjuntos de

Nombre	Tipo	Valor
miprimerdeployenlavidia.tk	NS	ns-97.awsdns-12.com, ns-522.awsdns-01.net, ns-1860.awsdns-40.co.uk, ns-1286.awsdns-32.org.
miprimerdeployenlavidia.tk	SOA	ns-97.awsdns-12.com, awsdns-hostmaster.amazon.

Nameservers

You can change where your domain points to here. Please be aware changes can take up to 24 hours to propagate.

☐ Use default nameservers (Freeom Nameservers)

☒ Use custom nameservers (enter below)

Nameserver 1

ns-97.awsdns-12.com.

Nameserver 2

ns-522.awsdns-01.net.

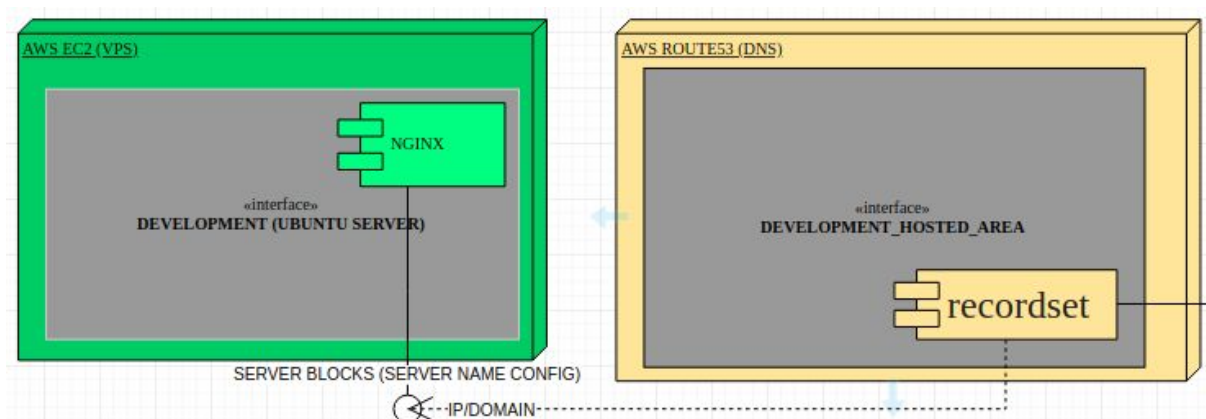
Nameserver 3

ns-1860.awsdns-40.co.uk.

Nameserver 4

ns-1286.awsdns-32.org.

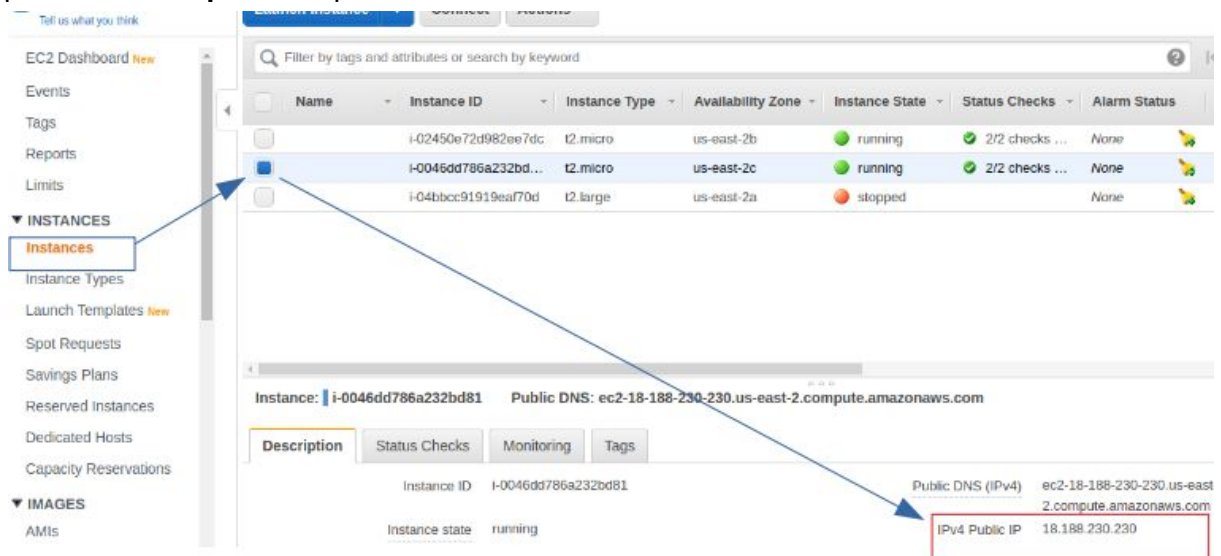
Dar en **'Change Nameservers'**. Con esto, nuestro proveedor de dominio, ya tiene los nombres de servidores provistos por AWS Route53. El siguiente paso es apuntar el DNS a nuestro VPS, recordando nuevamente la arquitectura como se muestra en la siguiente imagen.



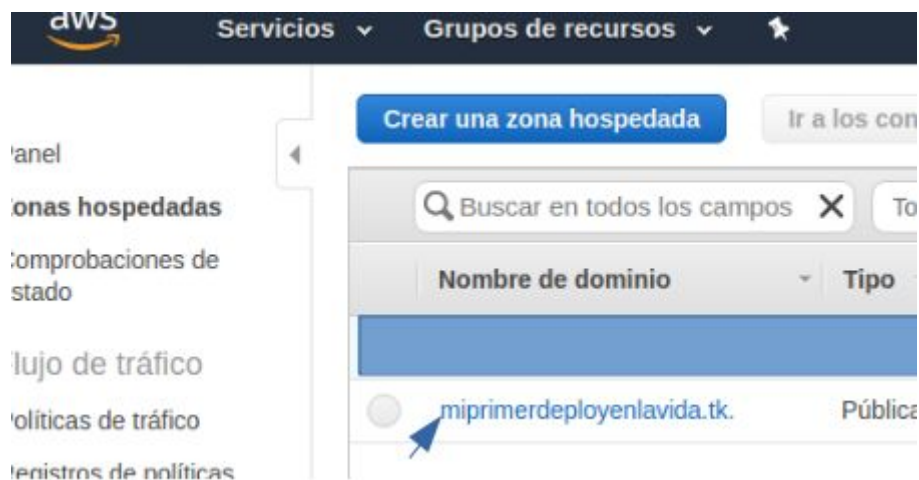
Para esto necesario primero saber la dirección IP del VPS, para esto ir a **servicios** en la misma consola de AWS, y dar en el servicio de **EC2**.



En **instances** y luego seleccionar el VPS que se ha creado en este documento. En la pestaña **Description**, se puede ver la IPv4 de nuestro VPS.



Ahora sabiendo que la dirección IP es **18.188.230.230**, es posible ir a Route53, también a través de **servicios**, como en pasos anteriores, y dar en Route53, aquí, ir a **Zonas Hospedadas**, y elegir el que se ha creado en este documento como se muestra en la imagen.



Dar en **‘Crear un conjunto de registros’**, y ahí agregar la dirección IP rescatada anteriormente, luego dar en **‘Crear’**.

Crear un conjunto de registros

Nombre:

Tipo: A: dirección IPv4

Alias: ☐ Sí ☒ No

TTL (segundos): 300 1 m 5 m

Valor: 18.188.230.230

Dirección IPv4. Escriba varias direcciones en líneas distintas.
Ejemplo:
192.0.2.235
198.51.100.234

Política de direccionamiento: Simple

Route 53 responde a las consultas solo en función del registro. [Más información](#)

Crear

Luego repitiendo el proceso dar en **‘Crear un conjunto de registros’**, esta vez agregando **www** al nombre, y dejándolo como **Alias** al dominio **miprimerdeployenlavida.tk**

Crear un conjunto de registros

Nombre: www.miprimerdeployenlavida.tk

Tipo: A: dirección IPv4

Alias: ☒ Sí ☐ No

Destino de alias: miprimerdeployenlavida.tk

ID de alias de zona hospedada: Z2ECK84G8IPR6

También puede escribir el nombre de dominio del recurso. Ejemplos:

- Nombre de dominio de la distribución de CloudFront: d111111abcdef8.cloudfront.net
- CNAME del entorno de Elastic Beanstalk: ejemplo.elasticbeanstalk.com
- Nombre del DNS del balanceador de carga de ELB: ejemplo-1.us-east-2.elb.amazonaws.com
- Punto de enlace de sitio web de S3: s3-website.us-east-2.amazonaws.com
- Conjunto de registros de recursos en esta zona alojada: www.ejemplo.com
- punto de enlace de la VPC: ejemplo.us-east-2.vpc.amazonaws.com
- API regional personalizada de API Gateway: d-abcde12345.execute-api.us-west-2.amazonaws.com
- Nombre DNS de Global Accelerator: a0123456789abcdef.awsglobalaccelerator.com

[Más información](#)

Crear

Cabe recordar que esta última configuración, será necesaria para cada dominio que se pretenda alojar en este VPS, todos apuntando a la dirección IP del VPS, pero en este, un servidor web **NginX**, se encargará de hacer el match entre los dominios solicitados a la misma IP, y los bloques de servidor configurados en el.

A modo de resumen la configuración de Route53 debería lucir algo así.

Nombre del conjunto de registro

X

Cualquier tipo

Solo con alias

Solo ponderados

Mostrando de 1 a 4 de un total de 4 Conjuntos de registros

<div></div>	Nombre	Tipo	Valor	Evaluar el
<div></div>	www.miprimerdeployenlavidat.tk.	A	ALIAS miprimerdeployenlavidat.tk. (z2ecck84g8ipr6)	No
<div></div>	miprimerdeployenlavidat.tk.	SOA	ns-97.awsdns-12.com. awsdns-hostmaster.amazon.	-
<div></div>	miprimerdeployenlavidat.tk.	NS	ns-522.awsdns-01.net. ns-97.awsdns-12.com. ns-1860.awsdns-40.co.uk. ns-1286.awsdns-32.org.	-
<div></div>	miprimerdeployenlavidat.tk.	A	18.188.230.230	-

4. Configuración el servidor Web (NginX+Firewall)

Hasta este punto, existe una arquitectura que nos permite establecer un número n de dominios en un mismo VPS. Un DNS, que nos permitirá configurar cada uno de estos dominios (y si fuera necesario sus sub dominios), para que puedan apuntar correctamente al VPS. Pero se hace necesario un último paso en este proceso, la configuración de quien manejara las peticiones, el uso de la memoria durante las sesiones de consulta, y claro, el código fuente de todo el contenido que los distintos dominios tienen.

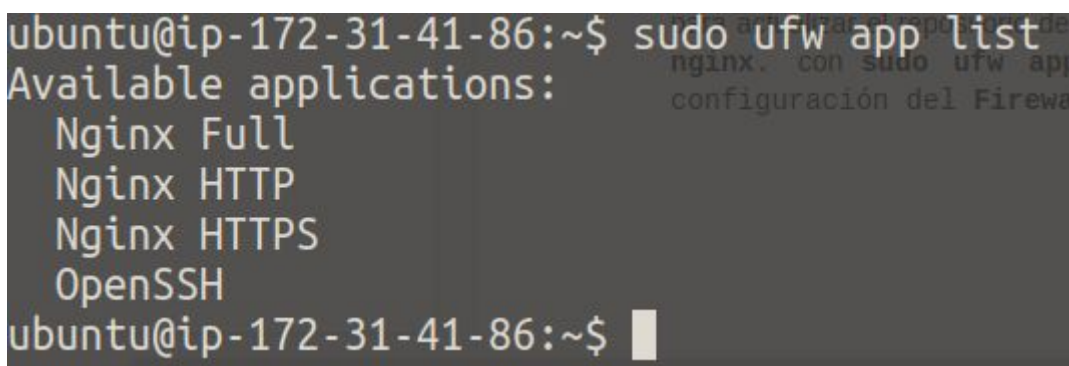
Este último paso, es precisamente la instalación del servidor web. Un servidor web o servidor HTTP es un programa informático que procesa una aplicación del lado del servidor, realizando conexiones bidireccionales o unidireccionales y síncronas o asíncronas con el cliente y generando o cediendo una respuesta en **cualquier lenguaje o aplicación del lado del cliente**. El código recibido por el cliente es renderizado por un navegador web. Para la transmisión de todos estos datos suele utilizarse algún protocolo. Generalmente se usa el protocolo HTTP para estas comunicaciones, perteneciente a la capa de aplicación del modelo **OSI**.

En este documento se trabajará con el servidor web **NginX**, para iniciar con su instalación es necesario acceder al VPS mediante la llave SSH que AWS nos facilitó en el inciso de su configuración presente en este documento (Página 10).

Con el comando

```
ssh -i "aws_ssh_key.pem" ubuntu@ec2-18-188-230-230.us-east-2.compute.amazonaws.com
```

Accederemos al VPS. Una vez dentro utilizar los siguientes comandos, `sudo apt update`, para actualizar el repositorio de aplicaciones de ubuntu, y posteriormente `sudo apt install nginx`. con `sudo ufw app list`, es posible visualizar las opciones de configuración del **Firewall** (filtro de peticiones por puerto).



```
ubuntu@ip-172-31-41-86:~$ sudo ufw app list
Available applications:
  Nginx Full
  Nginx HTTP
  Nginx HTTPS
  OpenSSH
ubuntu@ip-172-31-41-86:~$
```

En la imagen anterior, se muestran las opciones al ejecutar el comando, en este caso utilizaremos **Nginx Full** (Este perfil abre tanto el puerto 80 (HTTP) como el puerto 443 (HTTPS)).

Para activar esta configuración, es necesario usar el comando `sudo ufw allow 'Nginx Full'`. Además es necesario activar SSH para poder conectarnos al VPS con la llave que creamos y descargamos anteriormente. Usar el comando `sudo ufw allow 'OpenSSH'`, además es necesario recordar que al momento de configurar el VPS, se creo una regla de seguridad que habría los puertos desde el 0 hasta el 4000, con el comando `sudo ufw allow 1:4000/tcp` podemos replicar esta configuración para que no hayan variaciones. Finalmente para habilitar el **Firewall**, con el comando `sudo ufw enable`.

```
ubuntu@ip-172-31-41-86:~$ sudo ufw enable
Command may disrupt existing ssh connections. Proceed with operation (y|n)? y
```

Luego de esto, al usar el comando `sudo ufw status verbose`, veremos reflejada la configuración que hemos hecho.

```
ubuntu@ip-172-31-41-86:~$ sudo ufw status verbose
Status: active
Logging: on (low)
Default: deny (incoming), allow (outgoing), disabled (routed)
New profiles: skip

To Action From
--
80,443/tcp (Nginx Full) ALLOW IN Anywhere
22/tcp (OpenSSH) ALLOW IN Anywhere
1:4000/tcp ALLOW IN Anywhere
80,443/tcp (Nginx Full (v6)) ALLOW IN Anywhere (v6)
22/tcp (OpenSSH (v6)) ALLOW IN Anywhere (v6)
1:4000/tcp (v6) ALLOW IN Anywhere (v6)
```

Por fines prácticos, en este ejemplo se utilizó la apertura arbitraria de un rango de puertos bastante amplia. Es importante entender que en un contexto de producción real, **no es una buena práctica la apertura de puertos de esta manera**.

Con **Nginx** instalado, y el Firewall configurado y habilitado. Dirigiendonos a la dirección <http://miprimerdeployenlavidat.tk/>, podremos ver que nuestro sistema de DNS ya está funcionando, pero que el contenido desplegado no es más que un **HTML default** incluido en la instalación de **NginX**, además de estar funcionando por HTTP (sin certificado de seguridad SSL).

No seguro | miprimerdeployenlavidat.tk

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.
Commercial support is available at nginx.com.

Thank you for using nginx.

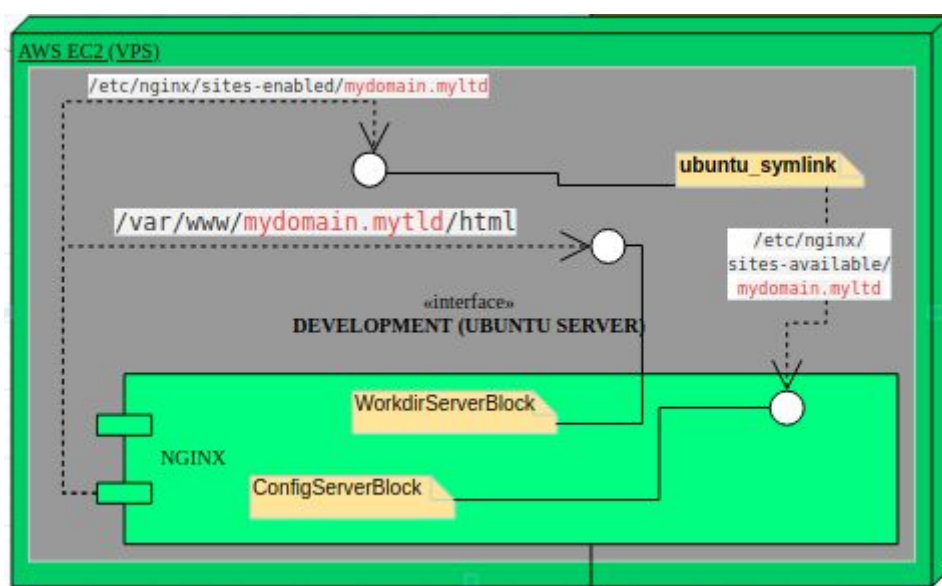
Para dejar **NginX**, funcionando con el arranque del sistema usaremos el comando `sudo systemctl enable nginx`.

```
ubuntu@ip-172-31-41-86:~$ sudo systemctl enable nginx
Synchronizing state of nginx.service with SysV service script with /lib/systemd/systemd-sysv-install.
Executing: /lib/systemd/systemd-sysv-install enable nginx
ubuntu@ip-172-31-41-86:~$ █
```



5. Bloque de Servidor en NginX y certificado (HTTPS)

Un bloque de servidor en un servidor web, no es más que una configuración exclusiva de un sitio con contenido específico. Por lo general los servidores web dividen el contexto de trabajo en recursos, y un fichero de configuración asociado. En este caso, **NginX** toma estas mismas prácticas, y apunta a un directorio de trabajo, mientras que un archivo, guarda configuración relacionada a los puertos, nombres de servidor, recursos lógicos multimedia, etc.



Crearemos entonces un nuevo directorio de trabajo para nuestro bloque de servidor, correspondiente a **miprimerdeployenlavidat.tk**, aquí tendremos todo el contenido de nuestra aplicación. Para concretar esta acción, es necesario utilizar el siguiente directorio,

sudo mkdir -p /var/www/miprimerdeployenlavidat.tk/html (El flag -p, permite crear de manera recursivas sub directorios que pertenecen a un padre que aun no esta creado), dentro de este directorio, en **html**, colocaremos el siguiente repositorio, que tiene una página promocional (Landing Page). El cual se encuentra trabajado en AngularJS embebido en un HTML puro. Con el siguiente comando nos moveremos al directorio

cd /var/www/miprimerdeployenlavidat.tk/html, luego, con el siguiente comando podremos realizar el clonado del repositorio en cuestión

'sudo git clone https://github.com/AlbertoIHP/ipsum_landing_v2 .' Luego, es necesario cambiar el propietario de dicho directorio para poder trabajar en él sin problemas, con el comando **'sudo chown -R \$USER:\$USER /var/www/miprimerdeployenlavidat.tk/html'**. Por último, el servidor web necesita igualmente de permisos para poder trabajar en dicho directorio, con **chmod** podemos cambiar los permisos de quienes intentan ejecutar o leer información contenida en este **workdir**. La siguiente imagen muestra el valor numérico de la modificación que se ejecutará con el comando

'sudo chmod -R 755 /var/www/miprimerdeployenlavidat.tk'.

CHMOD is used to change permissions of a file.

PERMISSION			COMMAND
U	G	W	
rwX	rwX	rwX	chmod 777 filename
rwX	rwX	r-X	chmod 775 filename
rwX	r-X	r-X	chmod 755 filename
rw-	rw-	r--	chmod 664 filename
rw-	r--	r--	chmod 644 filename
User	Group	World	r = Readable w = Writable x = Executable - = None

En la imagen anterior, muestra que con el código **755**, el usuario propietario tendrá permisos para leer (**r**), escribir (**w**) y ejecutar (**x**). Mientras que tanto **el grupo propietario**, como **el resto de los usuarios tendrán solo permiso para leer y ejecutar**.

Por otro lado, este repositorio contiene un proyecto además que embebe al comentado anteriormente, un pequeño servidor en NodeJS, con Express para el manejo de rutas. En este caso ignoraremos dicho servidor, y utilizando solo el contenido dentro del directorio **public**. Con esto entonces, podemos pasar al siguiente paso, el fichero de configuración del bloque de servidor. Con el comando

'sudo nano /etc/nginx/sites-available/miprimerdeployenlavidat.tk', crearemos el archivo de configuración en cuestión. En el, agregaremos la siguiente configuración.

```
server {
    listen 80;
    listen [::]:80;

    root /var/www/miprimerdeployenlavidat.tk/html/public;
    index index.html index.htm index.nginx-debian.html;

    server_name miprimerdeployenlavidat.tk www.miprimerdeployenlavidat.tk;

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

De la configuración anterior, se configura con **listen**, el puerto por el cual este bloque de servidor recibirá y enviará información. Con **root**, se indica el **workdir** del bloque de servidor, con **index**, todos los archivos que son considerados como índice de contenido, osea, que serán mostrados con preferencia ante cualquier otro tipo de archivo. Con **server_name**, se configuran los nombres de dominio a los cuales este bloque de servidor

responderá y desplegará información (Proxy). Finalmente, a través de **location**, podemos definir con **try_files**, la manera en la que el servidor reaccionara al intentar obtener una **uri** que no está disponible ni configurada. En este caso, lo apuntamos arbitrariamente al **index.html**, presente en nuestro **workdir**. Con **Ctrl+Shift+v**, copiaremos a la consola la configuración. Con **Ctrl+X**, podemos indicarle al editor que queremos guardar los cambios, nos preguntara si deseamos guardar, escribiendo una **Y**, nos preguntará ahora el nombre del archivo para guardarlo, donde daremos enter sin realizar ningún cambio.

Con esto, **NginX**, ya tiene un bloque de servidor asociado a nuestra configuración. Resta activarlo creando el **symlink** o enlace simbólico (básicamente un acceso directo) en **/etc/nginx/sites-enabled**, con el mismo nombre. Con el comando **sudo ln -s /etc/nginx/sites-available/miprimerdeployenlavida.tk /etc/nginx/sites-enabled/**, lo crearemos, y mediante el comando **ls -la /etc/nginx/sites-enabled**, visualizamos que efectivamente el symlink ha sido creado correctamente apuntando al fichero de configuración en cuestión.

```
ubuntu@ip-172-31-41-86:/var/www$ ls -la /etc/nginx/sites-enabled
total 8
drwxr-xr-x 2 root root 4096 Dec  9 14:18 .
drwxr-xr-x 8 root root 4096 Dec  8 21:13 ..
lrwxrwxrwx 1 root root   34 Dec  8 21:13 default -> /etc/nginx/sites-available/default
lrwxrwxrwx 1 root root   52 Dec  9 14:18 miprimerdeployenlavida.tk -> /etc/nginx/sites-available/miprimerdeployenlavida.tk
```

Para evitar problemas de memoria, es necesario modificar el archivo de configuración base de **NginX**, con '**sudo nano /etc/nginx/nginx.conf**' accederemos al fichero en cuestion, y para buscar específicamente una fracción de texto usar **Ctrl+W**, y escribir ahí **server_names_hash_bucket_size 64**;, el cual estará comentado con una **#**, eliminar el comentario y guardar con **Ctrl+X** igual que anteriormente.

Resta entonces solo reiniciar **NginX**, con el comando **sudo systemctl restart nginx** surtirán efectos los cambios realizados en el VPS.

Recordar además crear la regla en la zona hospedada de Route53 para el certificado.

The screenshot shows the AWS Route 53 console interface for configuring a CAA (Certification Authority Authorization) record. The record name is 'miprimerdeployenlavida.tk'. The type is set to 'CAA: autorización de la entidad de certif'. The alias is set to 'No'. The TTL (Time To Live) is set to 300 seconds. The value field contains '0 issue "letsencrypt.org"'. Below the value field, there is explanatory text: 'Valores que especifican las entidades de certificación que pueden emitir certificados para este dominio. Escriba varios valores en líneas distintas. Formato: [marca] [etiqueta] [valor] Ejemplos: 0 issue "caa.example.com" 0 issuewild ";"'. The policy is set to 'Simple'. At the bottom, there is a note: 'Route 53 responde a las consultas solo en función de los valores de este registro. Más información'.

Hasta este punto, la arquitectura completa se encuentra desplegada y funcionando correctamente, resta implementar un certificado SSL para tener nuestra página funcionando bajo **HTTPS**. Para esto se utilizará **Certbot**, quien nos provee gratuitamente de dicho certificado. Para agregar el repositorio al paquete de aplicaciones de ubuntu, utilizar **'sudo add-apt-repository ppa:certbot/certbot'**, dar **enter**. Actualizar luego el paquete de aplicaciones con **'sudo apt update'**. Por último, instalar certbot para nginx con **'sudo apt install python-certbot-nginx'**. Para solicitar el certificado utilizar **'sudo certbot --nginx -d miprimerdeployenlavidat.tk -d www.miprimerdeployenlavidat.tk'**, aquí se nos solicitará un correo, el cual por lo general debe corresponder al propietario del dominio, en este caso **'dopthisshit123@gmail.com'**.

```
ubuntu@ip-172-31-41-86:/var/www$ sudo certbot --nginx -d miprimerdeployenlavidat.tk -d www.miprimerdeployenlavidat.tk
Saving debug log to /var/log/letsencrypt/letsencrypt.log
Plugins selected: Authenticator nginx, Installer nginx
Enter email address (used for urgent renewal and security notices) (Enter 'c' to
cancel): dopthisshit123@gmail.com
```

Luego se consulta si estamos de acuerdo con los términos y condiciones, ponemos **A**, para aceptar.

```
----- correo, el cual por lo general debe corresponder al propietario del dominio, en este caso -----
Please read the Terms of Service at
https://letsencrypt.org/documents/LE-SA-v1.2-November-15-2017.pdf. You must
agree in order to register with the ACME server at
https://acme-v02.api.letsencrypt.org/directory
-----
(A)gree/(C)ancel: A
```

Luego nos preguntará si queremos suscribirnos a let'sencrypt, damos en **Y** si es que queremos obtener información recurrente de esta plataforma.

```
-----
Would you be willing to share your email address with the Electronic Frontier
Foundation, a founding partner of the Let's Encrypt project and the non-profit
organization that develops Certbot? We'd like to send you email about our work
encrypting the web, EFF news, campaigns, and ways to support digital freedom.
-----
(Y)es/(N)o: Y
```

Posteriormente, certbot realizará una serie de consultas para habilitar el certificado, donde si todo sale bien, debería preguntarnos por una dirección arbitraria, a la cual responderemos con una **no redirección**.

```

Obtaining a new certificate
Performing the following challenges:
http-01 challenge for miprimerdeployenlavida.tk
http-01 challenge for www.miprimerdeployenlavida.tk
Waiting for verification...
Cleaning up challenges
Deploying Certificate to VirtualHost /etc/nginx/sites-enabled/miprimerdeployenlavida.tk
Deploying Certificate to VirtualHost /etc/nginx/sites-enabled/miprimerdeployenlavida.tk

Please choose whether or not to redirect HTTP traffic to HTTPS, removing HTTP access.
- - - - -
1: No redirect - Make no further changes to the webserver configuration.
2: Redirect - Make all requests redirect to secure HTTPS access. Choose this for
new sites, or if you're confident your site works on HTTPS. You can undo this
change by editing your web server's configuration.
- - - - -
Select the appropriate number [1-2] then [enter] (press 'c' to cancel): 1

```

Finalmente, se desplegarán un par de notas útiles como la dirección de las llaves y las credenciales de certbot asociadas a let'sencrypt.

IMPORTANT NOTES:

- Congratulations! Your certificate and chain have been saved at: `/etc/letsencrypt/live/miprimerdeployenlavida.tk/fullchain.pem`
Your key file has been saved at: `/etc/letsencrypt/live/miprimerdeployenlavida.tk/privkey.pem`
Your cert will expire on **2020-03-08**. To obtain a new or tweaked version of this certificate in the future, simply run certbot again with the "**certonly**" option. To non-interactively renew ***all*** of your certificates, run "**certbot renew**"
- Your account credentials have been saved in your Certbot **configuration** directory at `/etc/letsencrypt`. You should make a secure backup of this folder now. This **configuration** directory will also contain certificates and private keys obtained by Certbot so making regular backups of this folder is ideal.
- If you like Certbot, please consider supporting our work by:

Donating to ISRG / Let's Encrypt: <https://letsencrypt.org/donate>
Donating to EFF: <https://eff.org/donate-le>

Por último, con el comando **'sudo certbot renew --cert-name miprimerdeployenlavidat.tk -a nginx -i nginx --dry-run'**, probaremos la renovación de nuestro certificado.

Nuestra aplicación de software ya se encuentra desplegada correctamente y al ingresar al dominio debería entonces mostrarnos la siguiente información con respecto al certificado.

Visualizador de certificados:
miprimerdeployenlavidat.tk

General Detalles

Este certificado se verificó para los siguientes usos:

Certificado del servidor SSL

Emitido a

Nombre común (CN)	miprimerdeployenlavidat.tk
Organización (O)	<No forma parte de un certificado>
Unidad organizativa (OU)	<No forma parte de un certificado>

Proporcionada por

Nombre común (CN)	Let's Encrypt Authority X3
Organización (O)	Let's Encrypt
Unidad organizativa (OU)	<No forma parte de un certificado>

Período de validez

Emitido el	lunes, 9 de diciembre de 2019, 10:32:54
Vence el	domingo, 8 de marzo de 2020, 10:32:54

Huellas digitales

Huella digital SHA-256	E7 C1 C2 DB 38 00 1A 88 59 6D 1D 3A 52 FE 7D 1A B1 25 E0 D6 8B 88 EE A5 BB 82 8F EF 8A 1E 3F A6 7D 9B FC 59 25 25 09 4A 34 28 52 21 0E CC 8C 4F E7 C6 04 37
Huella digital SHA-1	E7 C6 04 37