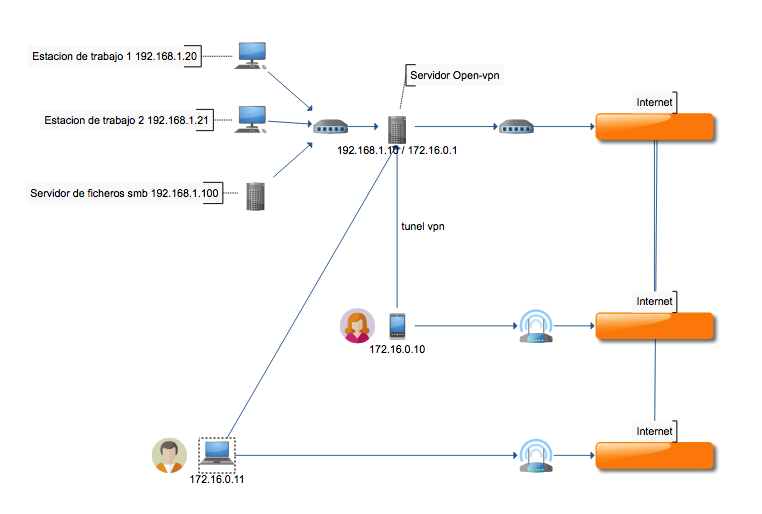
Practica Vpn

Para realizar esta práctica vamos a utilizar OPEN-VPN.

Según la Wikipedia:

**OpenVPN** es una herramienta de conectividad basada en [software libre](https://es.wikipedia.org/wiki/Software_libre): [SSL](https://es.wikipedia.org/wiki/Transport_Layer_Security) (Secure Sockets Layer), [VPN](https://es.wikipedia.org/wiki/Red_privada_virtual) Virtual Private Network (red virtual privada). OpenVPN ofrece conectividad punto-a-punto con validación jerárquica de usuarios y host conectados remotamente.

El esquema logico va a ser el siguiente.



Podemos ver, que tenemos un servidor vpn al cual se van a conectar los clientes desde sus dispositivos, ya sean moviles, ordenadores, en cualquiera de los sistemas operativos habituales (linux, windows, mac, android, ios)

Dentro de la red local de la organización tenemos varias estaciones de trabajo y un servidor de ficheros en la red de rango 192.168.1.0/24

El servidor vpn nos creara una red del rango 172.16.0.0/24, donde se conectaran nuestros usarios vpn.

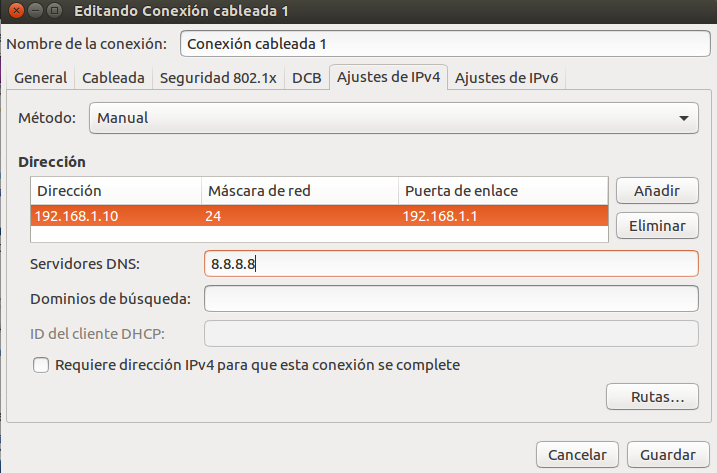
El tipo de vpn que usaremos será del tipo tun (esto hay que ponerlo bien)

Tendremos que crear un enrutamiento entre la red 172.16.0.0/24 y la red 192.168.1.0/24 para que los usuarios de la vpn puedan acceder a los recursos de nuestra red interna (en este caso el servidor de ficheros, aunque podría ser cualquier otro, como impresoras, servidores web de la intranet, impresoras etc.

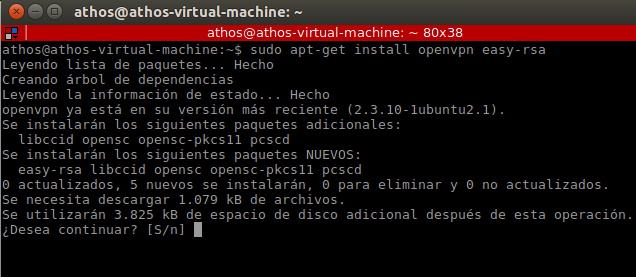
# Configuración servidor OPEN-VPN

Guia: <https://www.digitalocean.com/community/tutorials/how-to-set-up-an-openvpn-server-on-ubuntu-16-04>

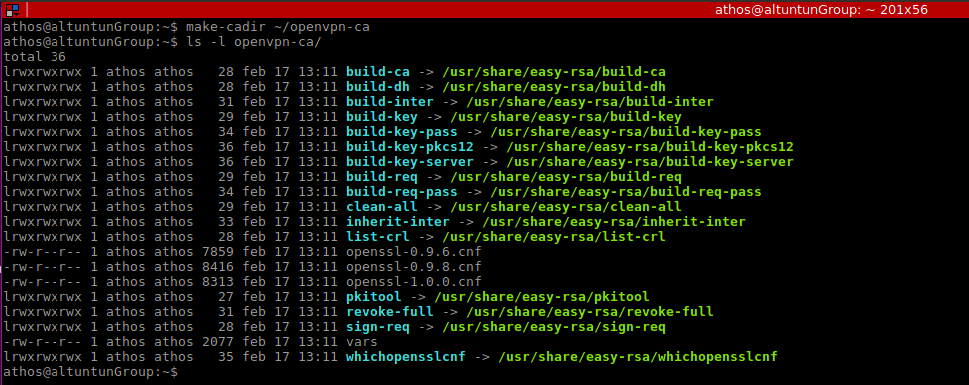
Lo primero que vamos ha hacer, es configurar la tarjeta de red del equipo que va a hacer de servidor.



Instalamos los paquetes openvpn y easy-rsa

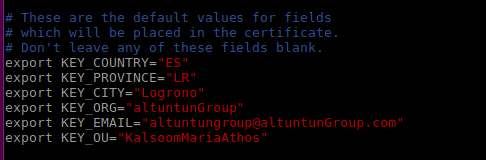


Ahora vamos a configurar el sistema de certificados que nuestro servidor expedirá a los diferentes clientes. Cada uno de los clientes necesitará su propio certificado para establecer el túnel vpn.

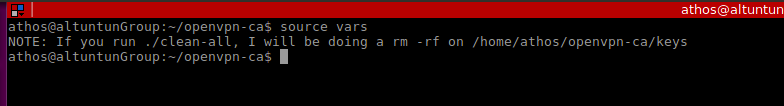


Ahora vamos a configurar las “variables” de la CA. Son los datos del tipo provincia, estado, nombre de organización etc.

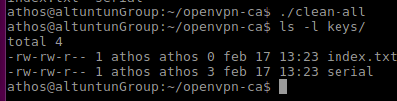
Para esto, configuraremos el archivo de configuración vars dentro del directorio que acabamos de crear (openvpn-ca)



Una vez configurado esto, vamos a proceder a crear el certificado de la CA. Para esto, una vez dentro del directorio, vamos a ejecutar el siguiente comando

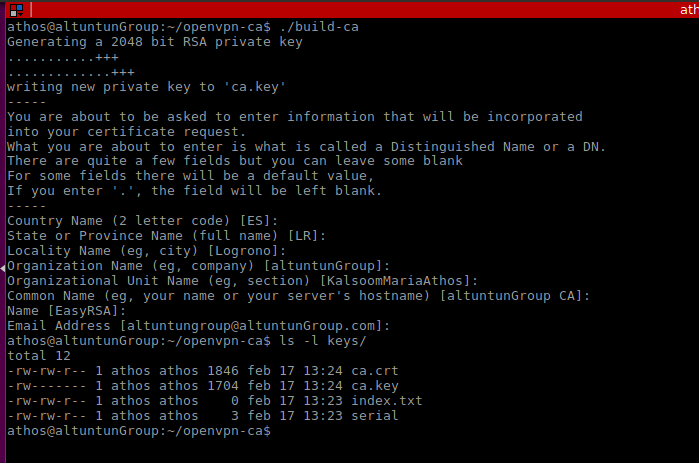


El sistema nos avisa, de que con el comando ./clean-all podemos eliminar todos los certificados, y asi empezar con el sistema limpio. Así, que como esta es una instalación limpia no haría falta, no obstante hacerlo tampoco nos supone ningún problema, asi que ejecutamos el comando.



Aun diciéndonos el aviso que el comando hara un rm –rf sobre el directorio keys, vemos que en realidad quedan dos archivos dentro, index.txt y serial.

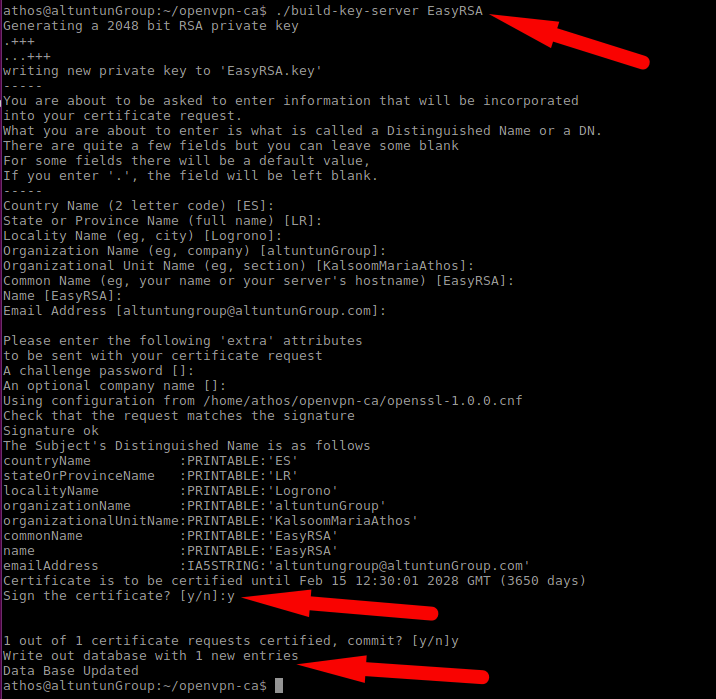
Ahora, vamos a generar la clave privada y la clave publica de la CA, lo haremos ejecutando ./build-ca



Todos los datos que nos ha pedido, han salido bien escritos por defecto gracias a la configuración que hemos hecho anteriormente del archivo vars, aunque nos daba la opción de configurarlos, asi que la edición del archivo vars no es totalmente necesaria.

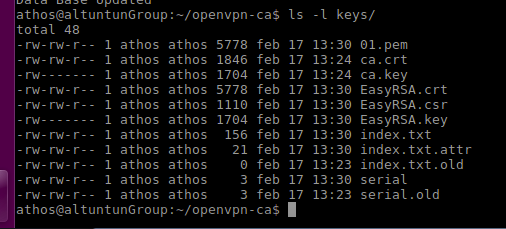
Podemos comprobar que dentro del directorio keys ahora tenemos dos archivos nuevos, ca.crt y ca.key, que son la clave publica y privada respectivamente.

Ahora vamos a crear el certificado del servidor, su clave privada y los archivos de encriptación. Esto lo haremos ejecutando el script build-server-key seguido del nombre que hayamos dado a nuestro servidor, en este caso lo dejamos por defecto asi que introduciremos EasyRSA

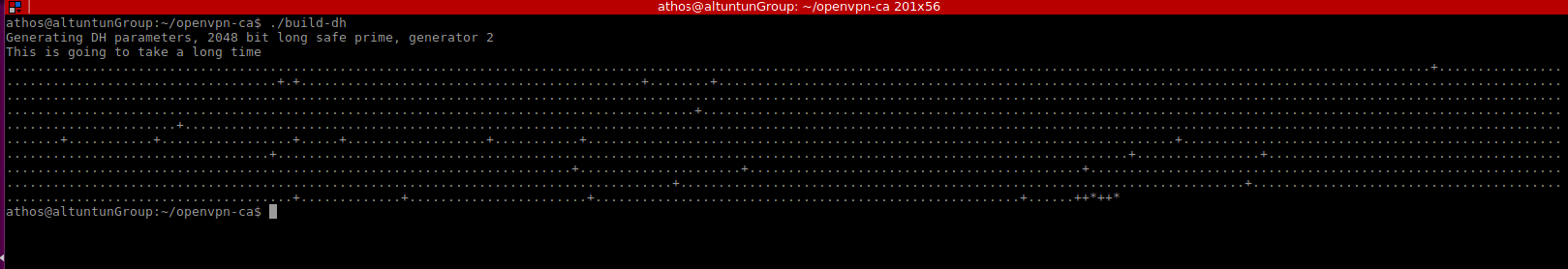


Como vemos en la imagen anterior, en la primera fleche ejecutamos el script, en la segunda, nos pregunta si queremos firmar el certificado que estamos generando para el servidor con el certificado de la CA que hemos generado anteriormente. En la tercera fleche podemos ver que se ha escrito el certificado y que la base de datos ha sido actualizada correctamente.

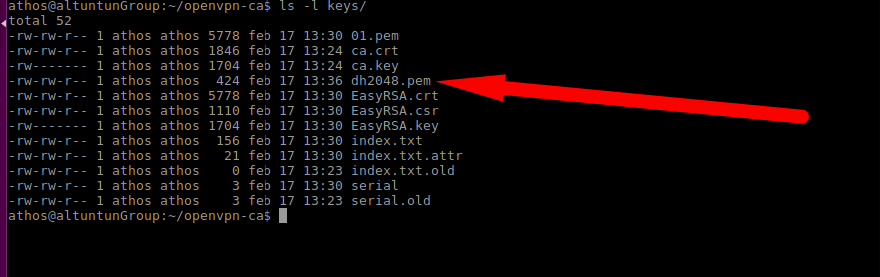
En el directorio keys vemos que ya tenemos mas archivos



Vamos a generar una clave diff-helman que será la que utilizaremso para el itercambio de claves. Esto lo haremos ejecutando el script build-dh



En nuestro archivo de claves vemos que nos ha generado la clave diff-helman



Ahora, vamos a generar una firma HMAC parareforzar la verificación de la integridad TLS del servidor

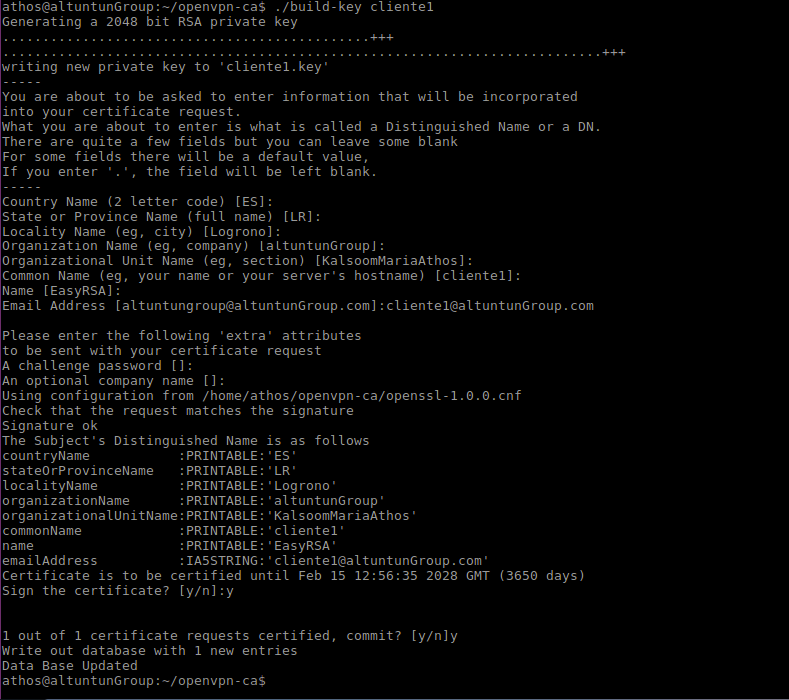


HMAC segun la Wikipedia:

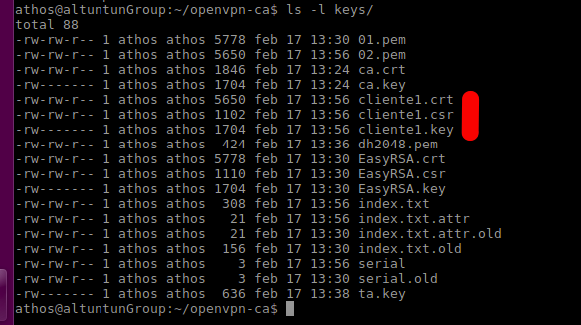
En la [criptografía](https://es.wikipedia.org/wiki/Criptograf%C3%ADa), un **código de autentificación de mensajes en clave-hash** (**HMAC**) es una construcción específica para calcular un [código de autentificación de mensaje](https://es.wikipedia.org/wiki/Message_authentication_code) (MAC) que implica una [función hash criptográfica](https://es.wikipedia.org/wiki/Funci%C3%B3n_hash_criptogr%C3%A1fica) en combinación con una llave criptográfica secreta. Como cualquier MAC, puede ser utilizado para verificar simultáneamente la *integridad de los datos* y la [*autentificación*](https://es.wikipedia.org/wiki/Autentificaci%C3%B3n) de un [mensaje](https://es.wikipedia.org/wiki/Mensaje). Cualquier función hash criptográfica, tales como [MD5](https://es.wikipedia.org/wiki/MD5) o [SHA-1](https://es.wikipedia.org/wiki/SHA-1), puede ser utilizada para el cálculo de un HMAC; el algoritmo MAC resultante se denomina HMAC-MD5 o HMAC-SHA1 en consecuencia. La fuerza criptográfica del HMAC depende de la potencia criptográfica de la función de hash subyacente, el tamaño de su salida de hash y el tamaño y calidad de la llave.

Fuente: https://es.wikipedia.org/wiki/HMAC

Ahora vamos a generar los certificados de los clientes. Sus respectivas claves publicas y privadas. Para esto, ejecutaremos el script build-key pasándole como parámetro el “nombre” del cliente

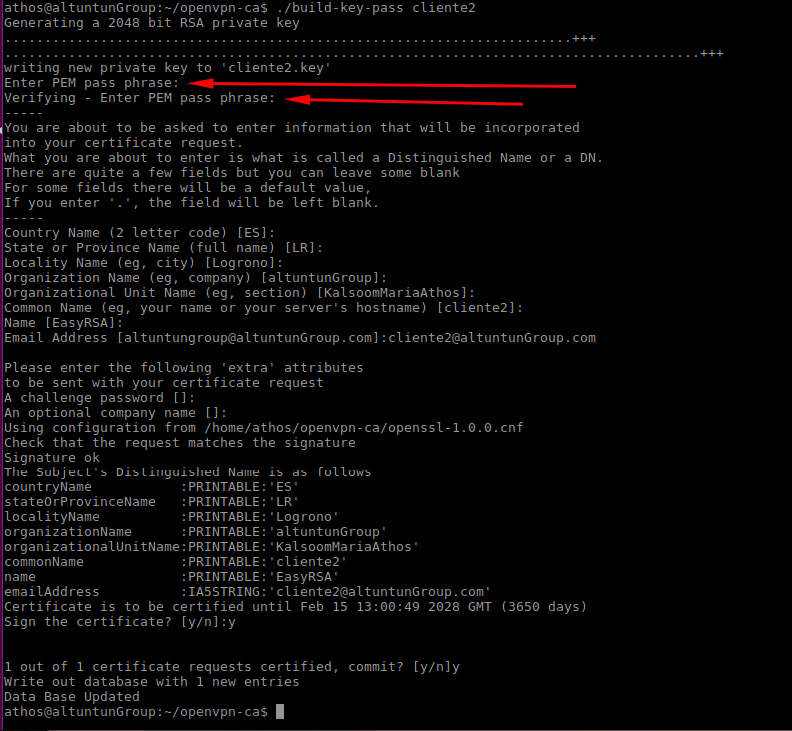


Si vamos al directorio keys, vemos lo siguiente



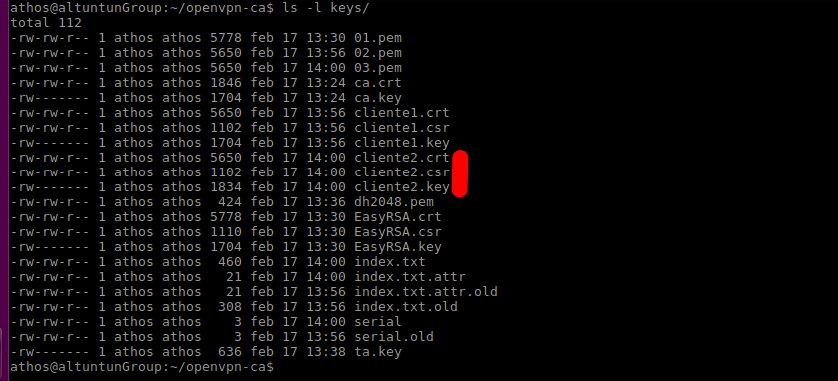
Repetiremos el proceso para el resto de clientes que necesitemos.

Tambien podremos crear los certificados protegidos por contraseña, que es lo que haremos con uno de los clientes. Para esto, ejecutaremos el script build-key-pass

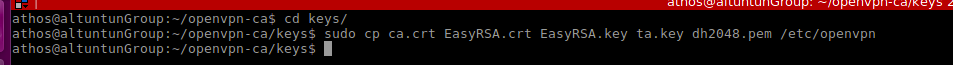


Como podemos observar, en esta ocasión nos solicita que introduzcamos una contraseña

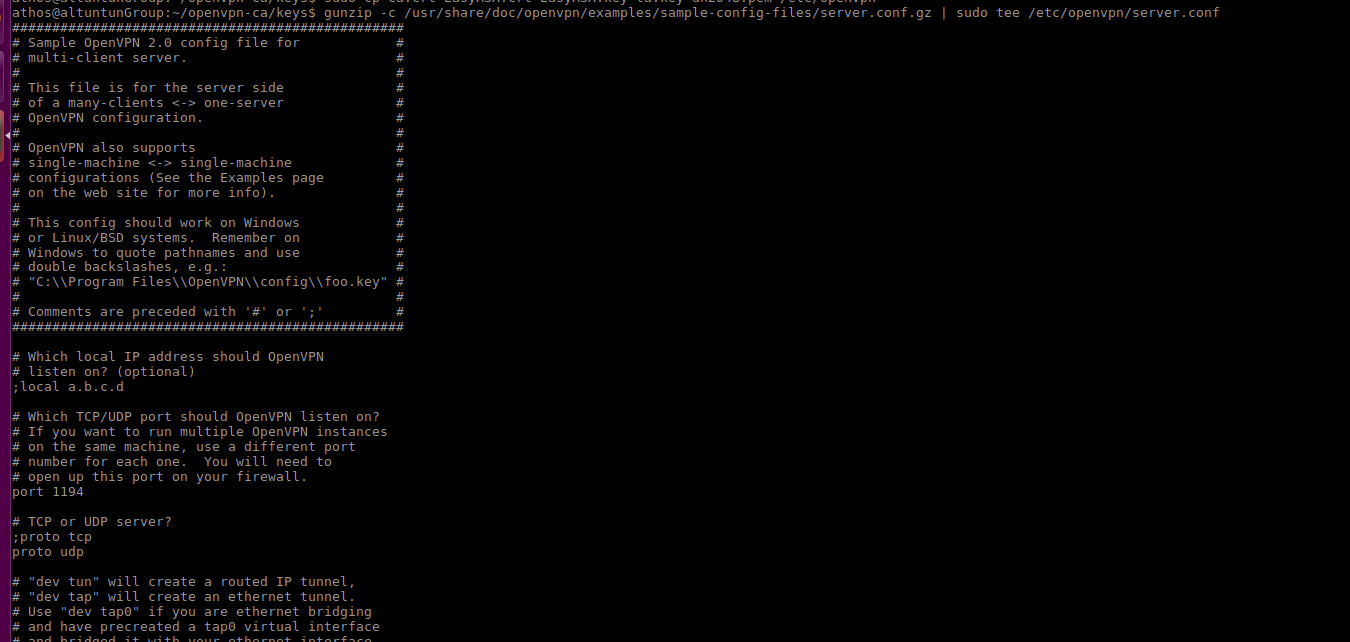
Vemos en el directorio keys que se nos han creado sus archivos



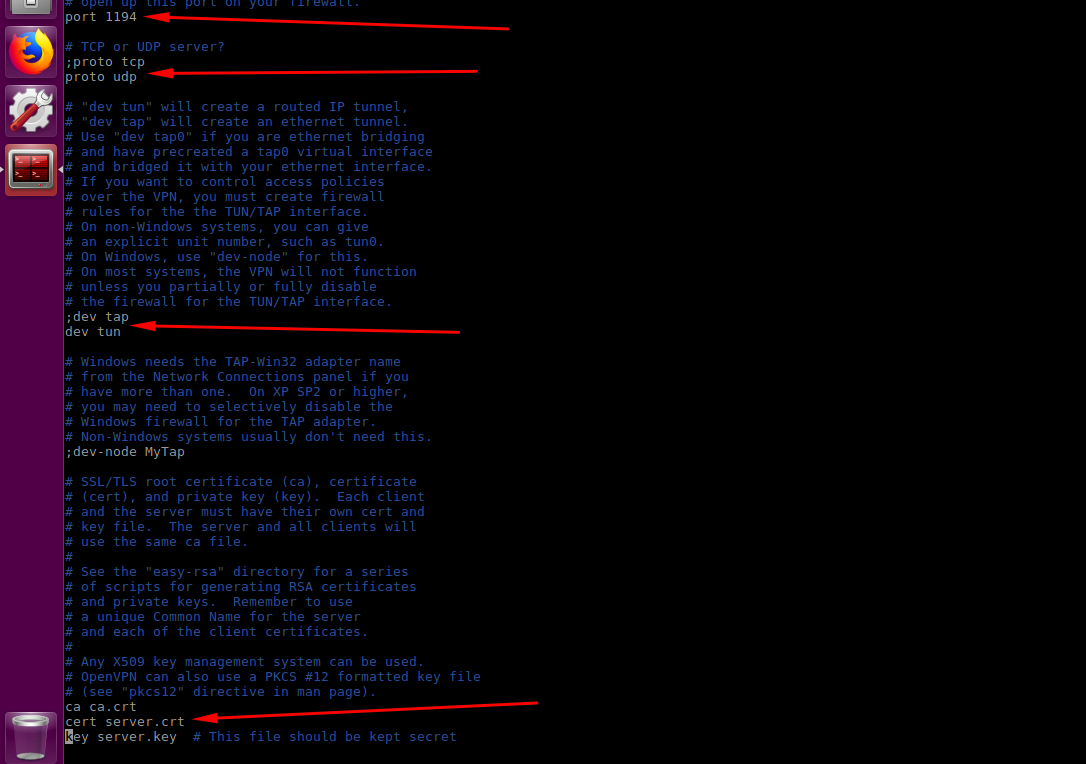
Ahora configuramos OpenVPN como servicio, para eso vamos a copiar algunos de los archivos que hemos generado en pasos anteriores al directorio /etc/openvpn



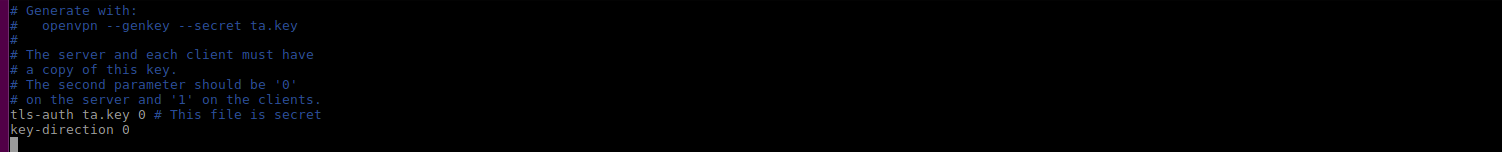
Copiamos un ejemplo de configuración en el directorio de openvpn

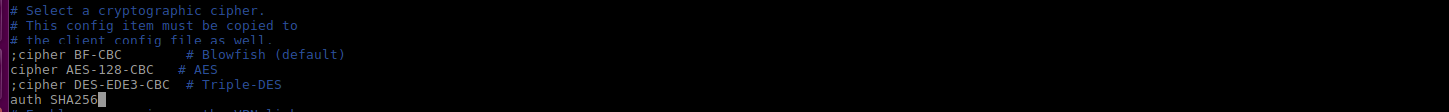


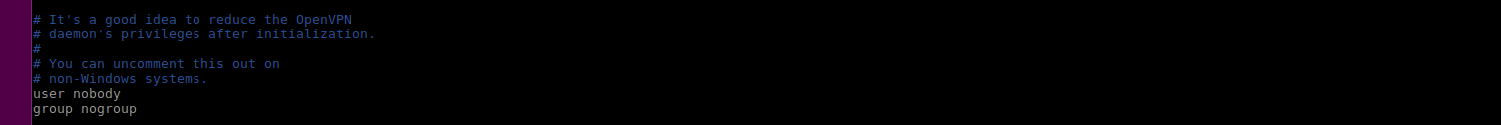
Vamos a configurar este archivo. Mostraremos las opciones mas importantes

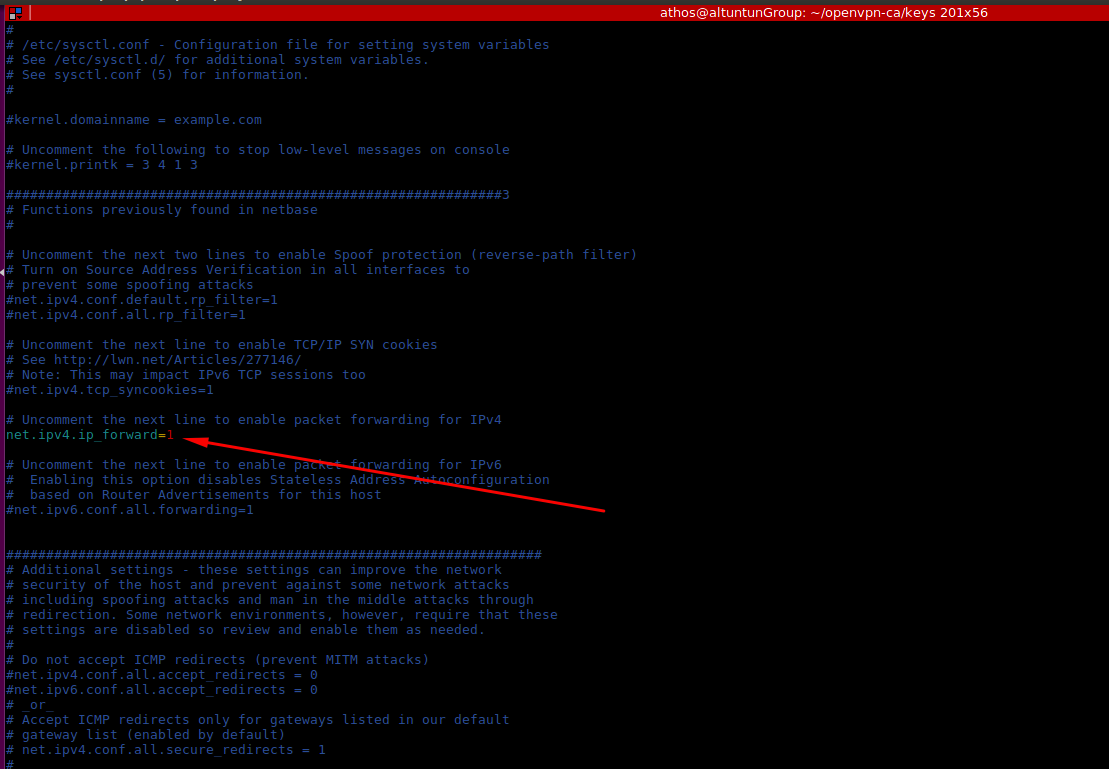


En la imagen anterior vemos el puerto, protocolo, tipo de túnel que crearemos (tap/tun) y cuales son los certificados que vamos a usar.

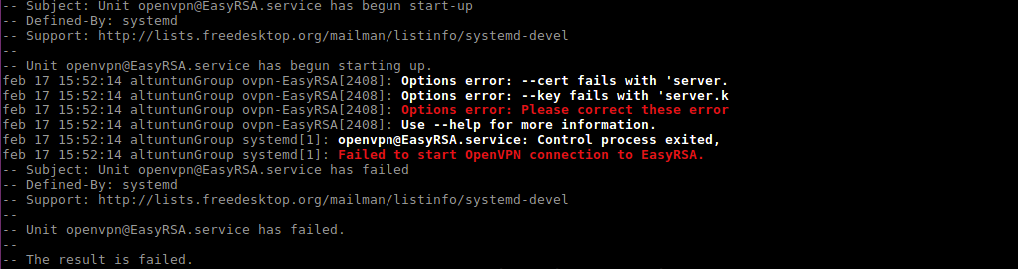




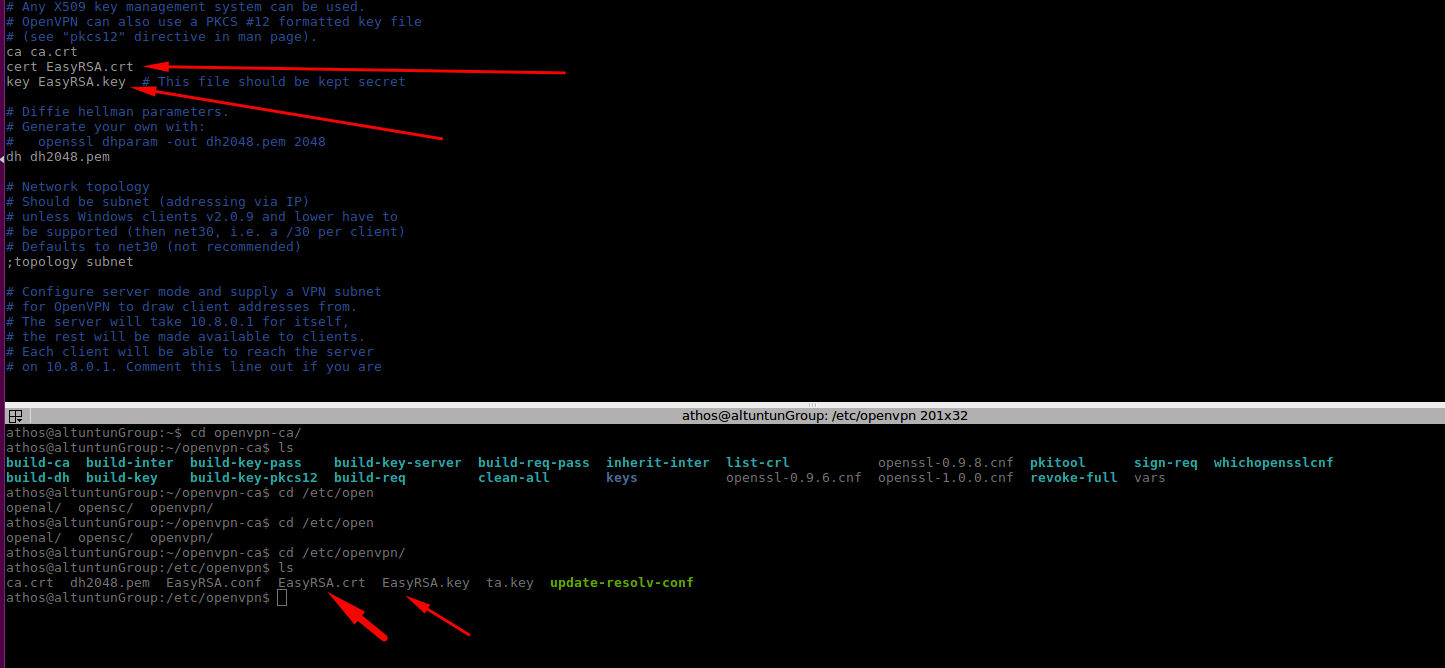




Lanzamos el servicion vpn con el comando sudo systemctl start openvpn@EasyRSA pero nos lanza un error.

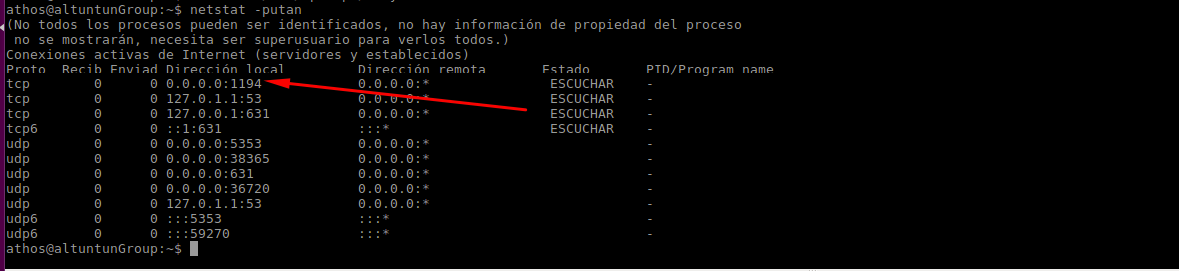


Como se ve en la consola, es un error relacionado con los certificados. Vamos al archivo de configuración y vemos que no habíamos establecido bien los nombres de los certificados, asi que lo corregimos.

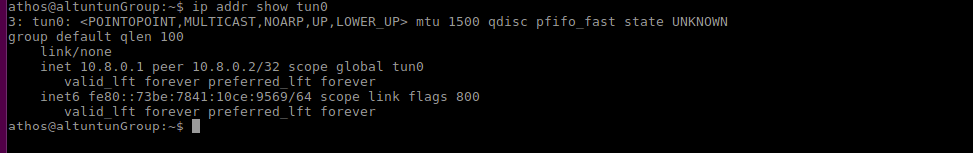


Volvemos a lanzar el comando. En este caso no nos ha dado ningún error.

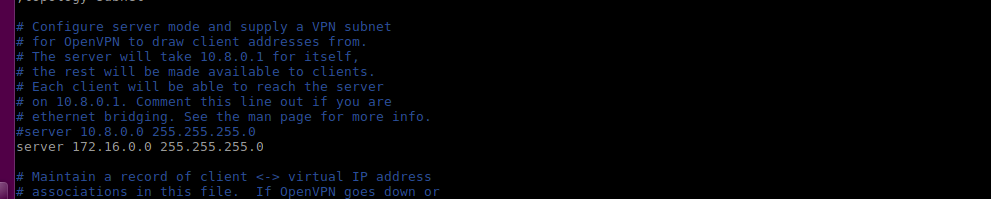
Comprobamos con netstat y vemos que ya tenemos el servicio corriendo en el puerto 1194



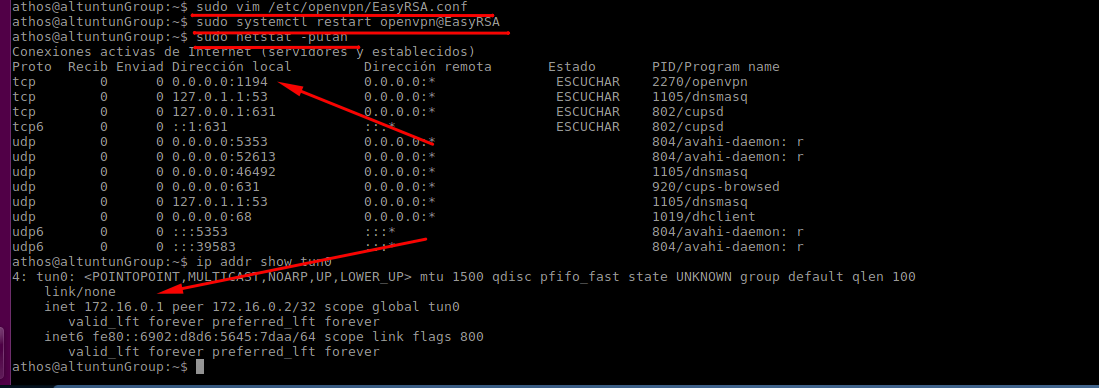
Ahora comprobamos si se ha creado de forma correcta el interface tun



Vemos que se ha creado bien, pero que el rango de ips no es el que queremos, asi que vamos al archivo de configuración y lo modificamos.

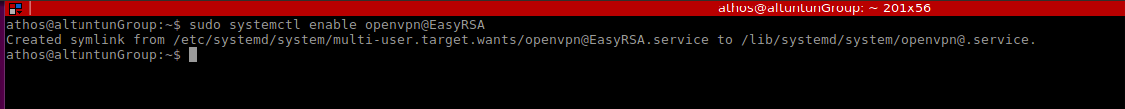


y reiniciamos el servicio



Como podemos ver, ahora si esta en el rango deseado

Ahora que vemos que manualmente el servicio se inicia bien, vamos a hacer que se inicie automáticamente cada vez que el servidor se encienda



Esctructura de configuraciones de los clientes

Vamos a generar lo necesario para crear los scripts de configuración de los clientes.

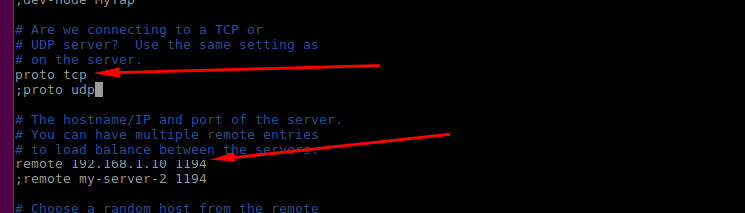
Primero vamos a crear una carpeta para almacenar todo

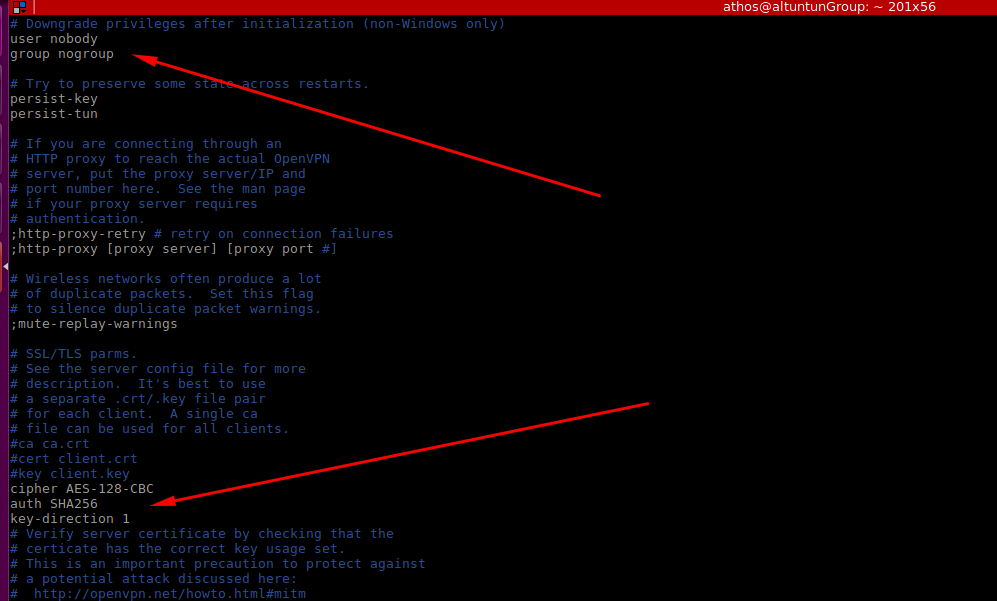
mkdir -p ~/client-configs/files

Hacemos uso de los ejemplos para luego adaptarlos a nuestras necesidades

cp /usr/share/doc/openvpn/examples/sample-config-files/client.conf ~/client-configs/base.conf

Los parámetros que hemos configurado son los siguientes





Ahora vamos a crear un script que nos genere el archivo de configuración

Crearemos el archivo con el siguiente nombre ~/client-configs/make\_config.sh y meteremos dentro lo siguiente

#!/bin/bash

# First argument: Client identifier

KEY\_DIR=~/openvpn-ca/keys

OUTPUT\_DIR=~/client-configs/files

BASE\_CONFIG=~/client-configs/base.conf

cat ${BASE\_CONFIG} \

<(echo -e '<ca>') \

${KEY\_DIR}/ca.crt \

<(echo -e '</ca>\n<cert>') \

${KEY\_DIR}/${1}.crt \

<(echo -e '</cert>\n<key>') \

${KEY\_DIR}/${1}.key \

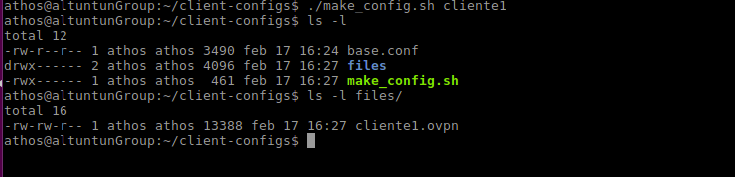
<(echo -e '</key>\n<tls-auth>') \

${KEY\_DIR}/ta.key \

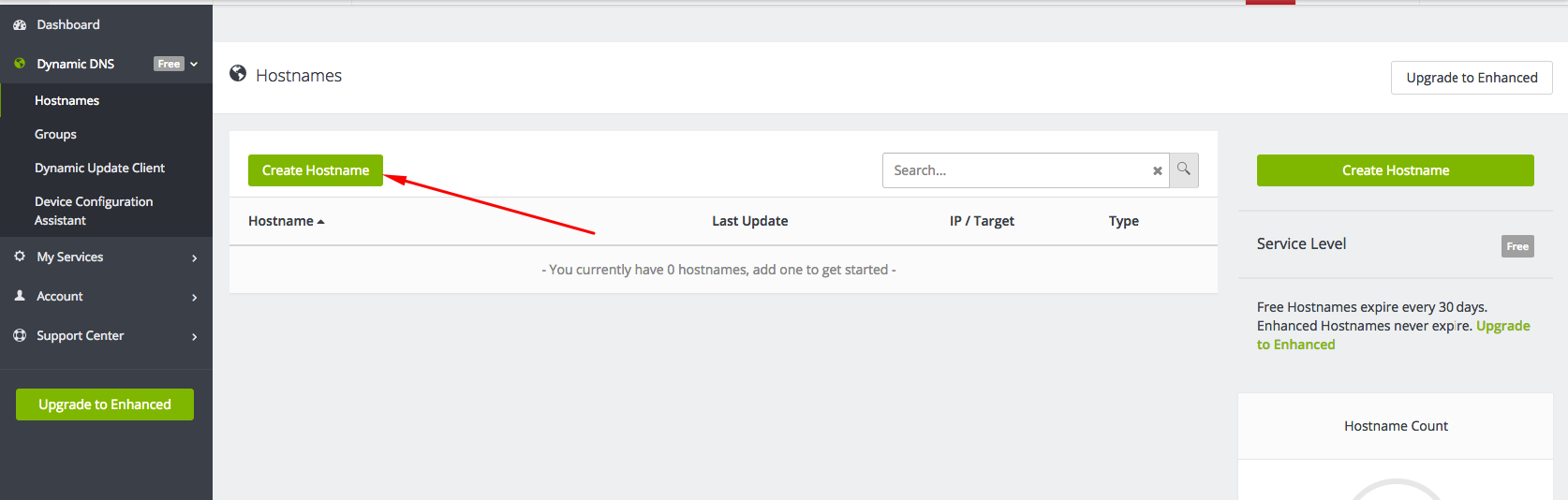
<(echo -e '</tls-auth>') \

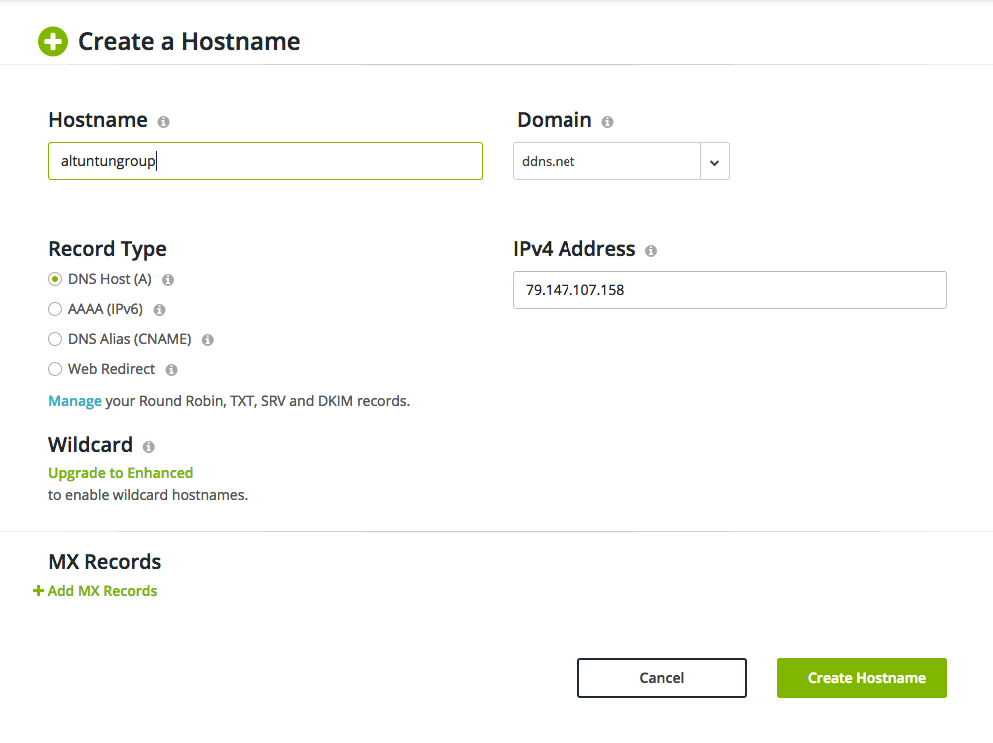
> ${OUTPUT\_DIR}/${1}.ovpn

ahora generaremos el archivo de configuración del cliente1 como se ve en la siguiente imagen



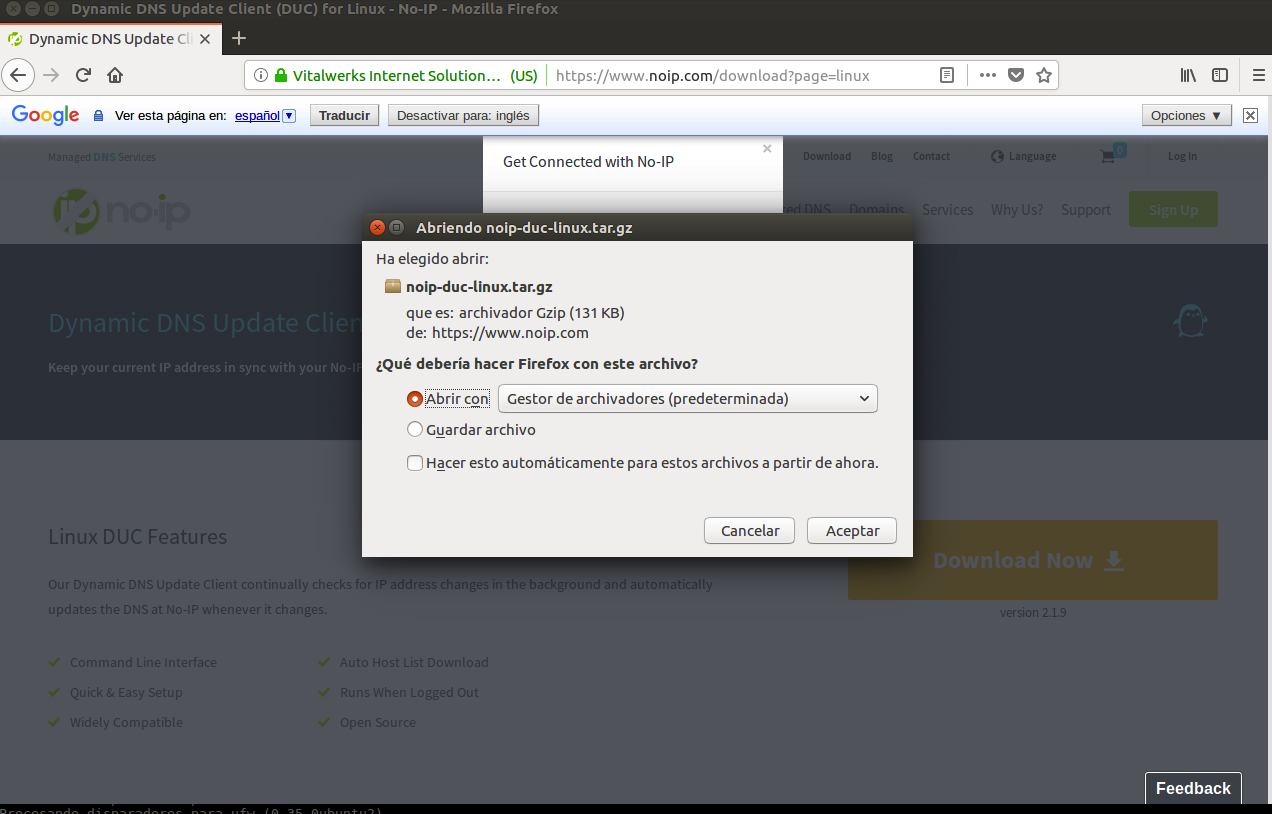
Para esta prueba, vamos a crear un hostname en no-ip para poder poner un nombre de dominio en la configuración de openvpn de nuestros clientes, ya que como tenemos ip dinámica, cada vez que cambiase tendríamos que modificar los archivos de configuración, asi que vamos a noip.com y creamos un nuevo hostname



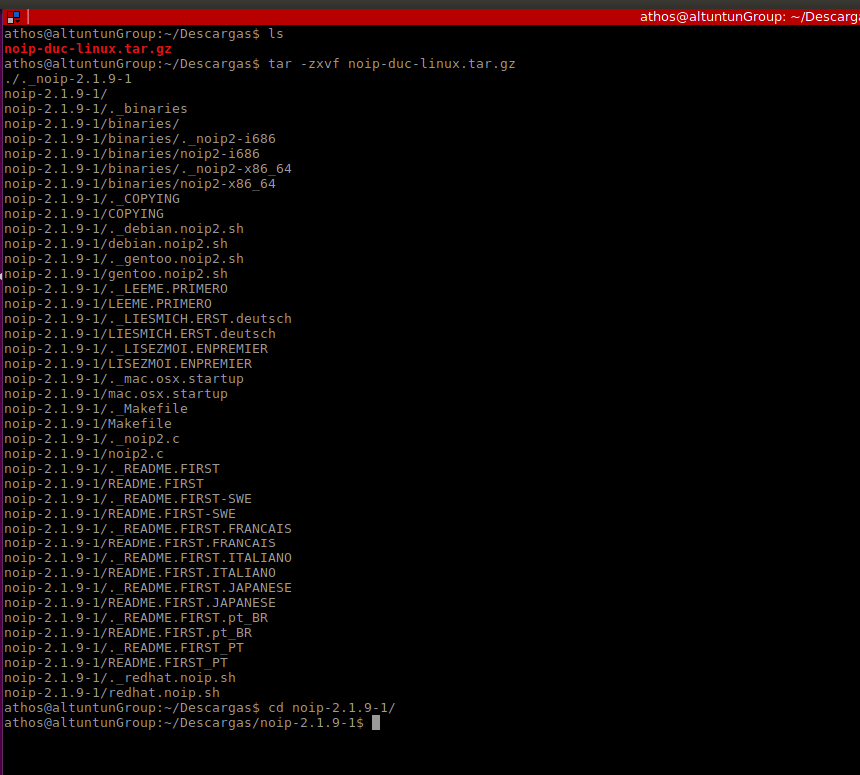


Una vez creado, en nuestro servidor, vamos a instalar el cliente de noip para que cada vez que cambie, actualice los dns de nuestro proveedor ddns (noip)

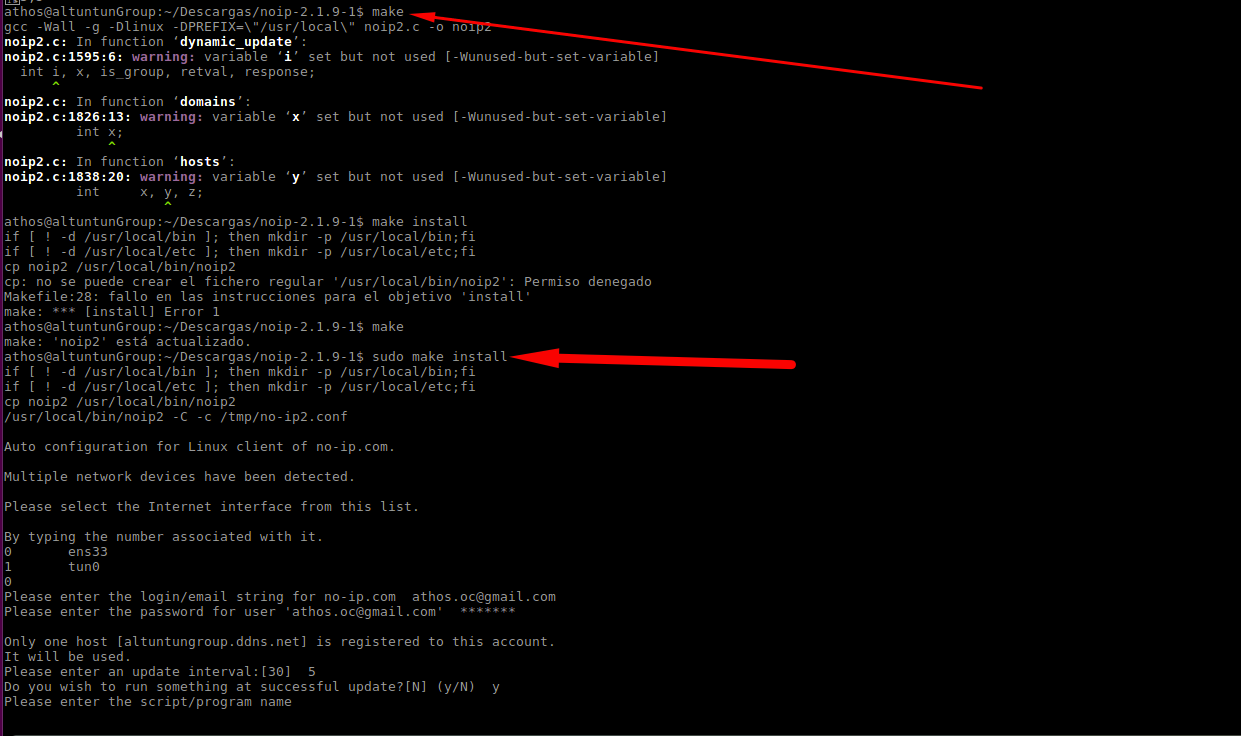
Nos dirigimos a su pagina y descargamos el cliente



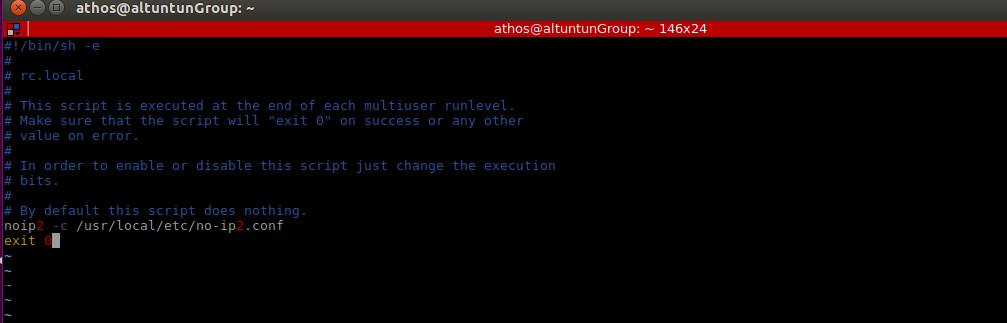
Descomprimimos el archivo



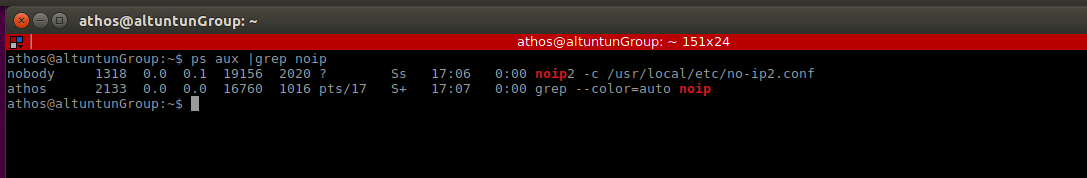
En la siguiente imagen podemos ver como compilamos el programa y lo instalamos



Para que el noip se ejecute cada vez que arranquemos el equipo, lo configuraremos en el archivo /etc/rc.local como se ve en la siguiente imagen



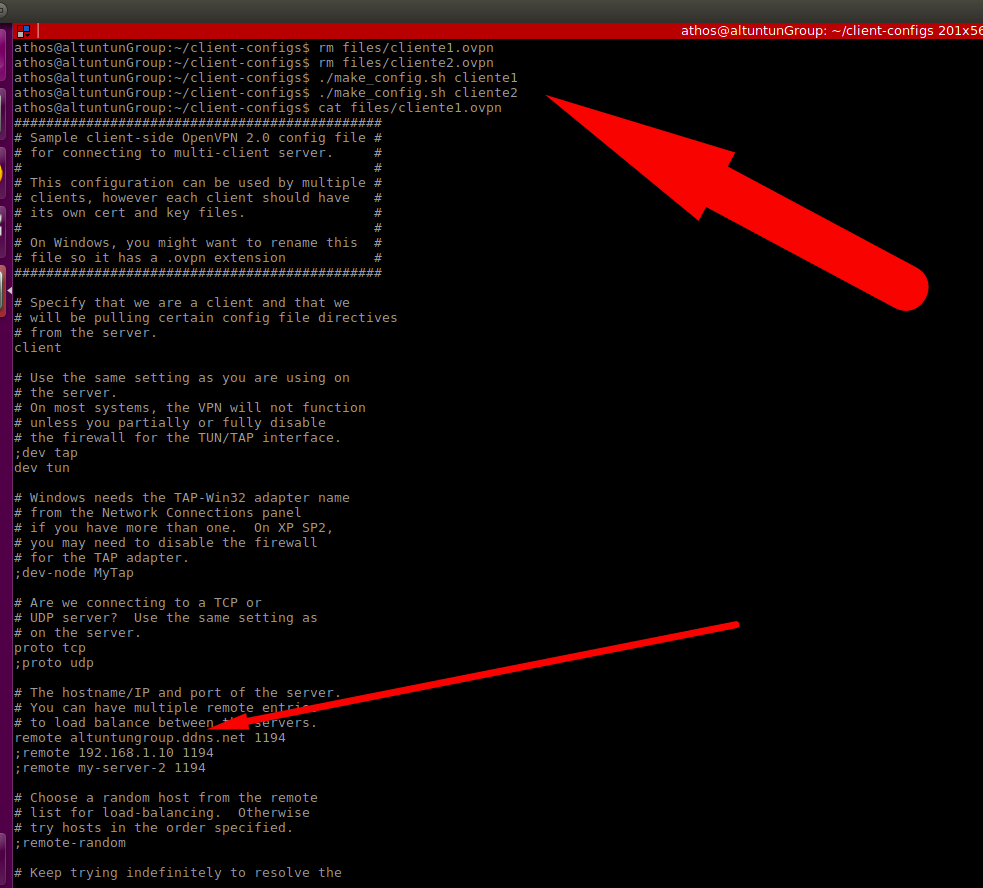
Reiniciamos la maquina y comprobamos que el noip esta corriendo.



Ahora, vamos a cambiar la configuración del script de generación de configuraciones para los clientes, y en vez de poner nuestra ip, vamos a poner el dominio que creamos en noip



Y ahora regeneramos las configuraciones de los clientes.



Hay que tener en cuenta, que deberemos abrir el puerto 1194 bien tcp o udp en función de cómo hayamos configurado el servidor vpn para que acepte las conexiones

