

* notas:

- los volúmenes no se iniciaban porque faltaba por instalar el paquete [thin-provisioning-tools](#)
- con `lxc.start.auto=1` en el archivo de configuración del contenedor, este arrancará automáticamente
- * al volumen lógico de gateway lo llamé por error Debian, lo comento para dejar constancia de ello
- * tenía un gran error, haber puesto una mac fija en los bridges de lxc igual al del bridge de openvswitch, he eliminado la mac que había fijado en cada interfaz y virt-manager ha asignado direcciones MAC aleatorias a cada una

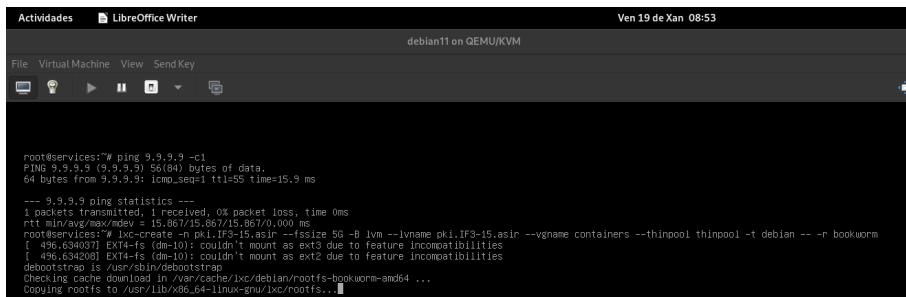
voy a intentar crear el contenedor nuevo:

```
root@services:~# lxc-create -n pki.IF3-15.asir --fssize 2G -B lvm --lvname pki.IF3-15.asir --vgname containers --thinpool thinpool -t debian -- -r bookworm
[ 683.596952] EXT4-fs (dm-10): couldn't mount as ext3 due to feature incompatibilities
[ 683.597217] EXT4-fs (dm-10): couldn't mount as ext2 due to feature incompatibilities
debootstrap is /usr/sbin/debootstrap
Invalid release bookworm (not found in mirror)
lxc-create: pki.IF3-15.asir: lxccontainer.c: create_run_template: 1616 Failed to create container from template
lxc-create: pki.IF3-15.asir: tools/lxc_create.c: main: 319 Failed to create container pki.IF3-15.asir
root@services:~# lxc-stop $gateway
root@services:~#
```

Falla porque no tengo red en services, la había desactivado.

Activo la red en la máquina virtual Services de forma temporal y ejecuto el siguiente comando para crear el contenedor pki, con un tamaño de 5GB y estando en el grupo de volúmenes que en una anterior práctica había creado “containers”, en la thinpool llamada también “thinpool”:

`lxc-create -n pki.IF3-15.asir --fssize 5G -B lvm --lvname pki.IF3-15.asir --vgname containers --thinpool thinpool -t debian -- -r bookworm`



```
root@services:~# ping 9.9.9.9 -c1
PING 9.9.9.9 (9.9.9.9) 56(84) bytes of data:
64 bytes from 9.9.9.9: icmp_seq=1 ttl=55 time=15.9 ms

--- 9.9.9.9 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 15.867/15.867/15.867/0.000 ms
root@services:~# lxc-create -n pki.IF3-15.asir --fssize 5G -B lvm --lvname pki.IF3-15.asir --vgname containers --thinpool thinpool -t debian -- -r bookworm
[ 496.634037] EXT4-fs (dm-10): couldn't mount as ext3 due to feature incompatibilities
[ 496.634308] EXT4-fs (dm-10): couldn't mount as ext2 due to feature incompatibilities
debootstrap is /usr/sbin/debootstrap
Checking cache download in /var/cache/lxc/debian/rootfs-bookworm-and64 ...
Copying rootfs to /usr/lib/x86_64-linux-gnu/lxc/rootfs...
```

Actualización: Aprovecho este momento para instalar [cryptsetup](#) en la máquina virtual.

Ahora que ya está creado el contenedor, vuelvo a dejar a la máquina virtual services sin IP. Modifico el archivo de configuración para que use el bridge apropiado con su correspondiente ip y su gateway para que tenga conexión con internet (usando el contenedor “gateway”).

```
# Template used to create this container: /usr/share/lxc/templates/lxc-debian
# Parameters passed to the template: -r bookworm
# For additional config options, please look at lxc.container.conf(5)

# Uncomment the following line to support nesting containers:
#lxc.include = /usr/share/lxc/config/nesting.conf
# (Be aware this has security implications)
lxc.start.auto = 1

lxc.net.0.type = veth
lxc.net.0.link = VM0
lxc.net.0.flags = up
lxc.net.0.ipv4.address = 172.30.15.5/24
lxc.net.0.ipv4.gateway = 172.30.15.1

lxc.apparmor.profile = generated
lxc.apparmor.allow_nesting = 1
lxc.rootfs.path = lvm:/dev/containers/pki.IF3-15.asir

# Common configuration
lxc.include = /usr/share/lxc/config/debian.common.conf

# Container specific configuration
lxc.tty.max = 4
lxc.uts.name = pki.IF3-15.asir
lxc.arch = amd64
lxc.ptty.max = 1024
```

El contenedor “pki” ya está en la red correcta y tiene conexión a internet a través del gateway.

```
root@pki:~# apt update
Des:1 http://security.debian.org/debian-security bookworm-security InRelease [48,0 kB]
Des:2 http://deb.debian.org/debian bookworm InRelease [151 kB]
Des:3 http://security.debian.org/debian-security bookworm-security/main amd64 Packages [134 kB]
Des:4 http://security.debian.org/debian-security bookworm-security/main Translation-en [80,0 kB]
Des:5 http://deb.debian.org/debian bookworm/main amd64 Packages [8.787 kB]
Des:6 http://deb.debian.org/debian bookworm/main Translation-en [6.109 kB]
Des:7 http://deb.debian.org/debian bookworm/main Translation-es [303 kB]
Descargados 15,6 MB en 2s (10,2 MB/s)
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias... Hecho
Se pueden actualizar 20 paquetes. Ejecute «apt list --upgradable» para verlos.
N: Repository 'http://deb.debian.org/debian bookworm InRelease' changed its 'Version' value from '12.2' to '12.4'
root@pki:~#
```

Instalo los paquetes **neovim**, **openssl** y **cryptsetup** en el contenedor “pki” recién creado.

```
~
/var/lib/xc/pki.IF9-15.asir/config
root@services:~# lxc-attach $pki
root@pki:~# apt install neovim openssl cryptsetup
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias... Hecho
Se instalarán los siguientes paquetes adicionales:
ca-certificates cryptsetup-bin libexpat1 libice6 liblua5.1-2 liblua5.1-common libmsgpackc2 libpopt0 libpython3-stdlib libpython3.11-minimal
libpython3.11-stdlib libreadline8 libsm6 libsqlite3-0 libssl3 libtermkey1 libtree-sitter0 libunibilium4 libuv1 libvterm0 libx11-6 libx11-data libxau6
libxcb1 libxdmcp6 libxext6 libxmu6 libxmuu1 libxt6 lua-luv media-types neovim-runtime python3 python3-greenlet python3-minimal python3-msgpack
python3-pynvim python3.11 python3.11-minimal readline-common x11-common xauth xclip xxd
Paquetes sugeridos:
cryptsetup-initramfs dosfstools keyutils liblocale-gettext-perl ctags vim-scripts python3-doc python3-tk python3-venv python-greenlet-dev
python-greenlet-doc python3.11-venv python3.11-doc binutils binfmt-support readline-doc
Se instalarán los siguientes paquetes NUEVOS:
ca-certificates cryptsetup cryptsetup-bin libexpat1 libice6 liblua5.1-2 liblua5.1-common libmsgpackc2 libpopt0 libpython3-stdlib
libpython3.11-minimal libpython3.11-stdlib libreadline8 libsm6 libsqlite3-0 libtermkey1 libtree-sitter0 libunibilium4 libuv1 libvterm0 libx11-6 libx11-data
libxau6 libxcb1 libxdmcp6 libxext6 libxmu6 libxmuu1 libxt6 lua-luv media-types neovim neovim-runtime openssl python3 python3-greenlet python3-minimal
python3-msgpack python3-pynvim python3.11 python3.11-minimal readline-common x11-common xauth xclip xxd
Se actualizarán los siguientes paquetes:
libssl3
1 actualizados, 46 nuevos se instalarán, 0 para eliminar y 19 no actualizados.
Se necesita descargar 19,6 MB de archivos.
Se utilizarán 64,3 MB de espacio de disco adicional después de esta operación.
¿Desea continuar? [S/n]
```

Instalo el paquete **bind9** en el contenedor “ns” para crear el servidor DNS.

```
Desempaquetando bind9 (1:9.18.19-1~deb12u1) ...
Configurando liblmbd0:amd64 (0.9.24-1) ...
Configurando libicu72:amd64 (72.1-3) ...
Configurando libmaxminddb0:amd64 (1.7.1-1) ...
Configurando libfstrm0:amd64 (0.6.1-1) ...
Configurando libnghttp2-14:amd64 (1.52.0-1+deb12u1) ...
Configurando libjemalloc2:amd64 (5.3.0-1) ...
Configurando libprotobuf-c1:amd64 (1.4.1-1+b1) ...
Configurando dns-root-data (2023010101) ...
Configurando libuv1:amd64 (1.44.2-1) ...
Configurando libxml2:amd64 (2.9.14+dfsg-1.3~deb12u1) ...
Configurando bind9-lbns:amd64 (1:9.18.19-1~deb12u1) ...
Configurando bind9-utils (1:9.18.19-1~deb12u1) ...
Configurando bind9 (1:9.18.19-1~deb12u1) ...
Adding group `bind' (GID 103) ...
Done.
Adding system user `bind' (UID 101) ...
Adding new user `bind' (UID 101) with group `bind' ...
Not creating home directory `/var/cache/bind'.
wrote key file "/etc/bind/rndc.key"
named-resolvconf.service is a disabled or a static unit, not starting it.
Created symlink /etc/systemd/system/bind9.service → /lib/systemd/system/named.service.
Created symlink /etc/systemd/system/multi-user.target.wants/named.service → /lib/systemd/system/named.service.
Procesando disparadores para libc-bin (2.36-9+deb12u2) ...
root@ns:~#
```

Instalo y configuro zona directa bind9 para el dominio if3-15.asir

Fuentes:

https://www.fpgenred.es/DNS/estamento_allowrecursion.html

https://www.fpgenred.es/DNS/estamento_recursion.html

<https://www.hostinger.es/tutoriales/comando-dig-linux>

Primero, configuro el archivo de zonas e indico la ruta de su archivo de configuración

```
root@ns:~#  
root@ns:~# batcat /etc/bind/named.conf.local  
  
File: /etc/bind/named.conf.local  
  
1 //  
2 // Do any local configuration here  
3 //  
4  
5 // Consider adding the 1918 zones here, if they are not used in your  
6 // organization  
7 //include "/etc/bind/zones.rfc1918";  
8  
9 // 2 spaces, 2 in zone  
10  
11 zone "if3-15.asir" {  
12     type master;  
13     file "/etc/bind/zones/db.if3-15.asir";  
14 };  
  
root@ns:~#
```

Ahora, configuro la zona if3-15.asir

Agregué el subdominio pki, que apunta a www pero lo he llamado así ya que es el que uso para alojar los certificados, las listas de revocación y una página default para probar el correcto funcionamiento de la PKI.

```
root@ns:~#  
root@ns:~# batcat /etc/bind/zones/db.if3-15.asir  
  
File: /etc/bind/zones/db.if3-15.asir  
  
1 $TTL 604800  
2 @ IN SOA NS.IF3-15.ASIR. ROOT.NS.IF3-15.ASIR. (  
3     876 ; Serial  
4     604800 ; Refresh  
5     86400 ; Retry  
6     2419200 ; Expire  
7     604800 ) ; Negative Cache TTL  
8 @ IN NS NS.IF3-15.ASIR.  
9 ns IN A 172.30.0.215  
10 pki IN A 172.30.15.5  
  
root@ns:~#
```

Preparo los discos virtuales encriptados... (la ruta . es [/home/pki](#))

```
dd if=/dev/zero of=rootca.disk bs=1M count=400
dd if=/dev/zero of=signca.disk bs=1M count=400
```

Creo los dispositivos loop a partir de los archivos raw.

```
losetup /dev/loop0 rootca.disk
losetup /dev/loop1 signca.disk
```

Ahora, uso cryptsetup para encriptar los dispositivos [loop0](#) y [loop1](#), de esta forma, quedarán inaccesibles sin un paso intermedio de desencriptar.

```
root@services:/home/pki# cryptsetup luksFormat /dev/loop0
WARNING!
=====
Esto sobrescribirá los datos en /dev/loop0 de forma irrevocable.
Are you sure? (Type 'yes' in capital letters): YES
Introduzca la frase contraseña de /home/pki/rootca.disk:
Verifique la frase contraseña:
root@services:/home/pki# cryptsetup luksFormat /dev/loop1
WARNING!
=====
Esto sobrescribirá los datos en /dev/loop1 de forma irrevocable.
Are you sure? (Type 'yes' in capital letters): YES
Introduzca la frase contraseña de /home/pki/signca.disk:
Verifique la frase contraseña:
root@services:/home/pki#
```

```
root@services:/home/pki# cryptsetup luksOpen /dev/loop0 rootca
Introduzca la frase contraseña de /home/pki/rootca.disk:
root@services:/home/pki#
root@services:/home/pki# cryptsetup luksOpen /dev/loop1 signca
Introduzca la frase contraseña de /home/pki/signca.disk:
root@services:/home/pki# _
```

Ahora los discos ya existen en [/dev/mapper](#) de forma accesible como dos archivo de bloques normales, que se pueden formatear y montar como cualquier otro.

Les doy formato ext4 a los discos desencriptados en [/dev/mapper](#).

```
mkfs.ext4 /dev/mapper/rootca
mkfs.ext4 /dev/mapper/signca
```

Y ahora que los archivos raw encriptados han sido asignados a un dispositivo loop y desencriptados y ahora son accesibles de forma transparente como archivos de bloques en [/dev/mapper](#), ya formateados con el sistema de archivos ext4, puedo montarlos como cualquier otro disco.

```
root@services:~# mkdir /mnt/rootca
root@services:~# mount /dev/mapper/rootca /mnt/rootca
root@services:~#
root@services:~# mkdir /mnt/signca
root@services:~# mount /dev/mapper/signca /mnt/signca
root@services:~# █
```

Ahora, estando en el contenedor de la pki, creo el usuario **pki**

```
root@pki:~# adduser pki
Adding user `pki' ...
Adding new group `pki' (1000) ...
Adding new user `pki' (1000) with group `pki (1000)' ...
Creating home directory `/home/pki' ...
Copying files from `/etc/skel' ...
Nueva contraseña:
Vuelva a escribir la nueva contraseña:
passwd: contraseña actualizada correctamente
Cambiando la información de usuario para pki
Introduzca el nuevo valor, o pulse INTRO para usar el valor predeterminado
Nombre completo []:
Número de habitación []:
Teléfono del trabajo []:
Teléfono de casa []:
Otro []:
Is the information correct? [Y/n]
Adding new user `pki' to supplemental / extra groups `users' ...
Adding user `pki' to group `users' ...
root@pki:~#
```

Creo los directorios en el usuario **pki**, que serán asociados con los directorios de services en donde están montados los discos desencriptados **rootca** y **signca**

```
root@services:~# lxc-attach $pki
root@pki:~# su pki
pki@pki:/root$ cd
pki@pki:~$ mkdir rootca
pki@pki:~$ mkdir signca
pki@pki:~$
pki@pki:~$ ls
rootca signca
pki@pki:~$
```

Agrego a la configuración de lxc las entradas para que asocie los directorios

```
root@services:~# cat /var/lib/lxc/pki.IF3-15.asir/config
# Template used to create this container: /usr/share/lxc/templates/lxc-debian
# Parameters passed to the template: -r bookworm
# For additional config options, please look at lxc.container.conf(5)

# Uncomment the following line to support nesting containers:
#lxc.include = /usr/share/lxc/config/nesting.conf
# (Be aware this has security implications)

#lxc.start.auto = 1

lxc.net.0.type = veth
lxc.net.0.link = VM0
lxc.net.0.flags = up
lxc.net.0.ipv4.address = 172.30.15.5/24
lxc.net.0.ipv4.gateway = 172.30.15.1

lxc.apparmor.profile = generated
lxc.apparmor.allow_nesting = 1
lxc.rootfs.path = lvm:/dev/containers/pki.IF3-15.asir

# Common configuration
lxc.include = /usr/share/lxc/config/debian.common.conf

# Container specific configuration
lxc.tty.max = 4
lxc.uts.name = pki.IF3-15.asir
lxc.arch = amd64
lxc.ptty.max = 1024

# Attach decrypted and mounted disks
lxc.mount.entry = /mnt/rootca home/pki/rootca none bind 0 0
lxc.mount.entry = /mnt/signca home/pki/signca none bind 0 0
root@services:~#
```

Desactivo el arranque del contenedor automático ya que en el arranque, los discos no estarán descriptados y por tanto tampoco montados.

```
# Uncomment the following line to support nesting containers:
#lxc.include = /usr/share/lxc/config/nesting.conf
# (Be aware this has security implications)

#lxc.start.auto = 1

lxc.net.0.type = veth
lxc.net.0.link = VM0
```

Arranco el contenedor, como se ve, ha iniciado correctamente.

```
/var/lib/lxc/pki.IF3-15.asir/config [+]
root@services:~# lxc-stop $pki
root@services:~# lxc-start $pki
[ 3925.823655] EXT4-fs (dm-10): couldn't mount as ext3 due to feature incompatibilities
[ 3925.825701] EXT4-fs (dm-10): couldn't mount as ext2 due to feature incompatibilities
root@services:~#
```

Desde el contenedor pki, hago que el usuario pki sea propietario de la carpeta montada que está usando encriptación y de cualquier archivo contenido, porque todo en esta carpeta será manejado por el usuario pki.

Para asegurarme de que pki tiene permiso por toda la estructura de su home, me pongo como root y hago un `chown -R pki:pki /home/pki`

```
root@pki:~# su pki -
pki@pki:/root$ cd
pki@pki:~$ ls
rootca  signca
pki@pki:~$ cd rootca/
```

Me descargo un archivo de configuración para una autoridad certificadora raíz usando openssl.

Fuente: <https://pki-tutorial.readthedocs.io/en/latest/advanced/index.html>

Debido a la forma en la que el montaje se ha hecho, las carpetas descriptadas son visibles no solo desde el contenedor sino también desde la máquina services, no es un punto inseguro realmente ya que services no tiene conexión a ninguna red, se requiere acceso físico al ordenador.

```
pki@pki:~$ ls
rootca  signca
pki@pki:~$ touch rootca/a
pki@pki:~$ exit
exit
root@pki:~# exit
exit
root@services:~# ls /mnt/
rootca/ signca/
root@services:~# ls /mnt/rootca/
a  lost+found  rootca.conf
root@services:~#
```

Para automatizar el proceso, he creado un script [startPKI.sh](#) que se encarga de; generar los discos virtuales loop a partir de los archivos de disco raw con `losetup`, crear los discos descriptados de forma transparente en `/dev/mapper` con `cryptsetup`, montarlos en `/mnt` e iniciar el contenedor pki.

```
root@services:~# cat startPKI.sh
echo Introduce passphrase
read passwd

losetup /dev/loop0 /home/pki/rootca.disk
losetup /dev/loop1 /home/pki/signca.disk

echo $passwd | cryptsetup luksOpen /dev/loop0 rootca
echo $passwd | cryptsetup luksOpen /dev/loop1 signca

mount /dev/mapper/rootca /mnt/rootca
mount /dev/mapper/signca /mnt/signca

lxc-start $pki
root@services:~# █
```

También he creado un script que, en orden inverso; detiene el contenedor, desmonta los discos descriptados, cierra los discos descriptados y elimina los discos virtuales loop.

```
root@services:~# cat stopPKI.sh
lxc-stop $pki

umount /mnt/rootca
umount /mnt/signca

cryptsetup luksClose rootca
cryptsetup luksClose signca

losetup -d /dev/loop0
losetup -d /dev/loop1

echo PKI detenida correctamente, discos encriptados cerrados
root@services:~# █
```

Tras hacer esto, ya se puede ver que de primeras no es accesible, el primer script lo prepara y monta como lo había hecho antes y el segundo script detiene y desmonta todo correctamente, todo esto automatizado para que se haga con dos instrucciones.

```
root@services:~# ls /mnt/rootca/
root@services:~#
root@services:~# ./startPKI.sh
Introduce passphrase
abc123.
[ 2329.408748] EXT4-fs (dm-10): couldn't mount as ext3 due to feature incompatibilities
[ 2329.409256] EXT4-fs (dm-10): couldn't mount as ext2 due to feature incompatibilities
root@services:~# ls /mnt/rootca/
a  lost+found  rootca.conf
root@services:~#
root@services:~# ./stopPKI.sh
PKI detenida correctamente, discos encriptados cerrados
root@services:~# ls /mnt/rootca/
root@services:~# █
```

La estructura que usaré para la autoridad certificadora raíz será la siguiente; teniendo una carpeta **private** para almacenar la clave privada, una carpeta **db** para almacenar los archivos que servirán como base de datos y la carpeta **crl** para almacenar la lista de revocación de certificados.

```
pki@pki:~/rootca$ ls
lost+found  rootca.conf
pki@pki:~/rootca$
pki@pki:~/rootca$ mkdir private db crl certs
pki@pki:~/rootca$ touch db/rootca.db
pki@pki:~/rootca$ touch db/rootca.db.attr
pki@pki:~/rootca$ echo 01 > db/rootca.crt.srl
pki@pki:~/rootca$ echo 01 > db/rootca.crl.srl
pki@pki:~/rootca$
```

Creo el directorio **etc** dentro de **rootca**, y muevo **rootca.conf** a **etc**.

Ahora, creo el csr y la clave privada de la root CA

Estoy usando esta guía: <https://pki-tutorial.readthedocs.io/en/latest/advanced/index.html>

```
openssl req -new -config etc/rootca.conf -out rootca.csr -keyout private/rootca.key
```

De momento, así ha quedado la estructura

```
-----
pki@pki:~/rootca$ ls -lR
.:
total 19
drwxr-xr-x 2 pki pki 1024 feb 27 08:32 certs
drwxr-xr-x 2 pki pki 1024 feb 23 09:04 crl
drwxr-xr-x 2 pki pki 1024 feb 23 09:05 db
drwxr-xr-x 2 pki pki 1024 mar  1 08:19 etc
drwx----- 2 pki pki 12288 feb 22 07:44 lost+found
drwxr-xr-x 2 pki pki 1024 mar  1 08:25 private
-rw-r--r-- 1 pki pki 1115 mar  1 08:25 rootca.csr

./certs:
total 0

./crl:
total 0

./db:
total 2
-rw-r--r-- 1 pki pki 3 feb 27 08:30 rootca.crl.srl
-rw-r--r-- 1 pki pki 3 feb 27 08:30 rootca.crt.srl
-rw-r--r-- 1 pki pki 0 feb 23 09:04 rootca.db
-rw-r--r-- 1 pki pki 0 feb 23 09:04 rootca.db.attr

./etc:
total 4
-rw-r--r-- 1 pki pki 4096 mar  1 08:16 rootca.conf

./lost+found:
total 0

./private:
total 2
-rw----- 1 pki pki 1854 mar  1 08:25 rootca.key
pki@pki:~/rootca$
```

Se puede ver el **rootca.conf** que he personalizado precisamente para la autoridad certificadora raíz que estoy creando (el que he usado para generar el crt y key) [aquí](#).

Ahora, creo el certificado a partir de la solicitud de firma de certificado.

```
openssl ca -selfsign -config etc/rootca.conf -in rootca.csr -extensions root_ca_ext -enddate 20301231235959Z
```

La fecha en el certificado está codificada usando el estándar ISO 8601, caduca exactamente al acabar 2030 (31 de diciembre de 2030 a las 23:59:59 UTC+0).

Creo la lista de revocación de certificados (CRL)

```
openssl ca -gencrl -config etc/rootca.conf -out crl/rootca.crl
```

Ahora ya tengo la autoridad certificadora raíz configurada.

Le pongo permisos **700** (u:rwX) a la carpeta **private**.

Creo las carpetas **serverca** y **userca** dentro de **/home/pki/signca**

Voy a crear la autoridad certificadora **serverca**...

He modificado el TLS CA de la guía Advanced PKI, para crear **serverca.conf**

Se puede acceder al archivo modificado desde [aquí](#).

Me muevo al directorio **serverca** dentro de **/home/pki/signca**, creo los directorios **certs** (para almacenar certificados), **db** (para almacenar la base de datos que openssl usará), **etc** (para almacenar la configuración) y **private** (para almacenar la clave privada de **serverca**) y **crl** (para almacenar la lista de revocación de certificados).

Muevo **serverca.conf** al directorio **etc**.

Le pongo permisos **700** (u:rwX) a la carpeta **private**.

Creo archivos vacíos **serverca.db** y **serverca.db.attr** en **db** con **touch**.

Creo también en **db**, archivos **serverca.crt.srl** y **serverca.crl.srl** que contienen **01**

Ahora que ya tengo la estructura de directorios y archivos, creo una clave privada **serverca.key** usando **openssl req -new -config etc/serverca.conf -out serverca.csr -keyout private/serverca.key**

Al usar ese comando también he creado **serverca.csr**, la solicitud de firma de certificado (**csr**).

Ahora, copio la solicitud de **serverca** a **rootca** y desde **rootca** y expido el certificado firmado para **serverca**.

```
openssl ca -config etc/rootca.conf -in serverca.csr -out certs/serverca.crt -extensions signing_ca_ext
```

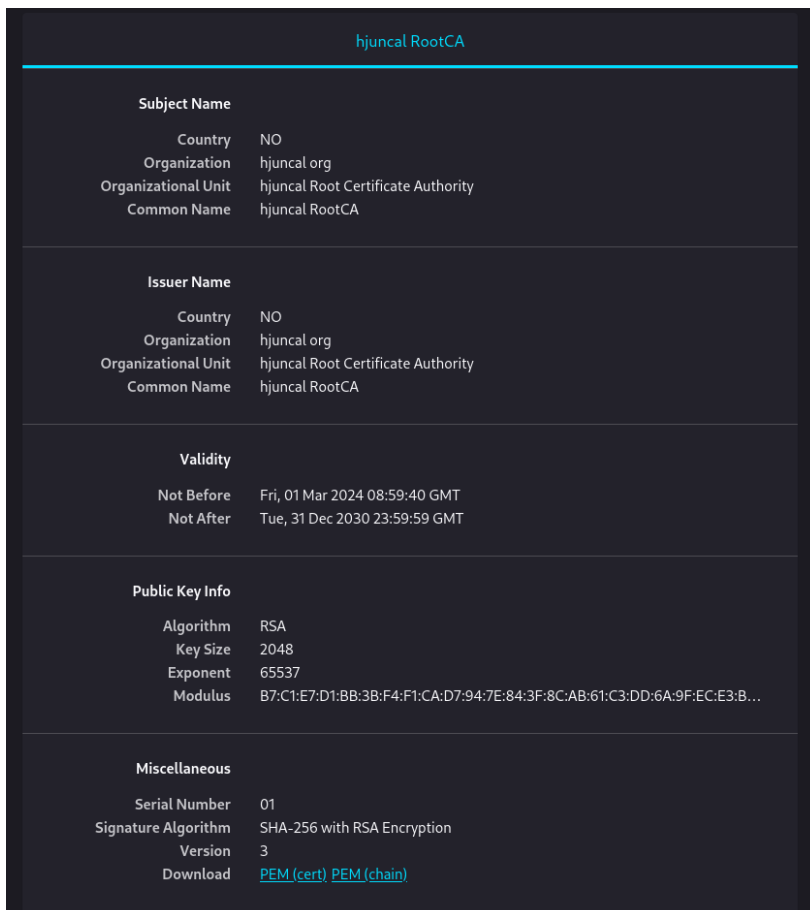
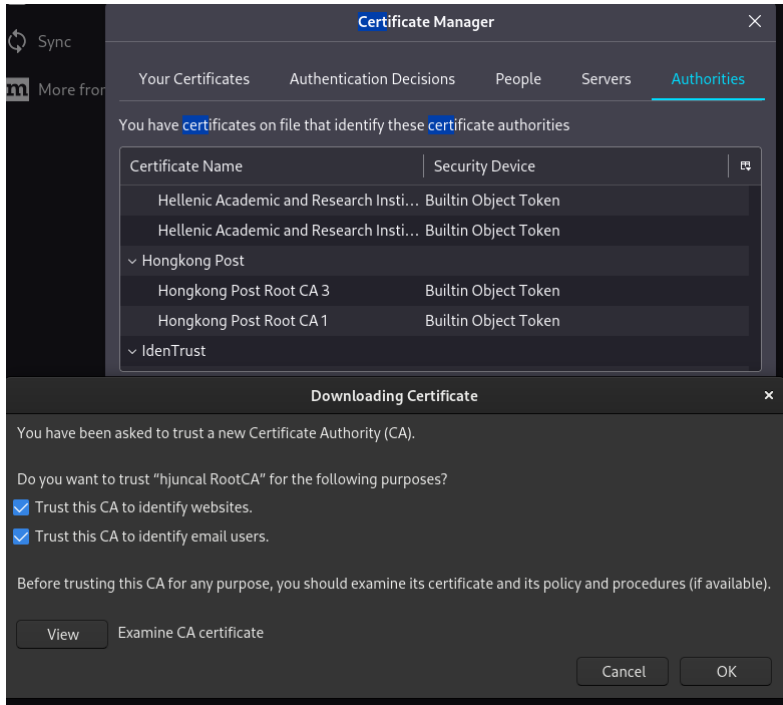
```
pk1@pki:~/rootca$ openssl ca -config etc/rootca.conf -in serverca.csr -out certs/serverca.crt -extensions signing_ca_ext
Using configuration from etc/rootca.conf
Enter pass phrase for /home/pki/rootca/private/rootca.key:
Check that the request matches the signature
Signature ok
Certificate Details:
  Serial Number: 2 (0x2)
  Validity
    Not Before: Mar  7 08:38:04 2024 GMT
    Not After : Mar  7 08:38:04 2034 GMT
  Subject:
    countryName           = NO
    organizationName      = hjuncal.org
    organizationalUnitName = hjuncal Sign Server Certificate Authority
    commonName            = hjuncal ServerCA
  X509v3 extensions:
    X509v3 Key Usage: critical
      Certificate Sign, CRL Sign
    X509v3 Basic Constraints: critical
      CA:TRUE, pathlen:0
    X509v3 Subject Key Identifier:
      BB:C8:D9:E2:BA:28:4C:7B:AF:88:85:24:85:31:C3:24:3C:1B:9F:EF
    X509v3 Authority Key Identifier:
      41:78:86:50:7B:C6:C4:ED:87:B6:6D:47:4F:DE:90:5A:E4:63:93:44
    Authority Information Access:
      CA Issuers - URI:https://pki.if3-15.asir/ca/rootca.crt
    X509v3 CRL Distribution Points:
      Full Name:
        URI:https://pki.if3-15.asir/ca/rootca.crl
Certificate is to be certified until Mar  7 08:38:04 2034 GMT (3652 days)
Sign the certificate? [y/n]:y

1 out of 1 certificate requests certified, commit? [y/n]:y
Write out database with 1 new entries
Database updated
pk1@pki:~/rootca$ ls certs/
01.pem  02.pem  rootca.crt  serverca.crt
pk1@pki:~/rootca$
```

Devuelvo el certificado solicitado con el csr y firmado por RootCA a ServerCA.

Me transfiero el certificado de rootca al ordenador y pruebo a instalarlo y abrirlo en firefox, para eso, uso scp.

Tengo que seleccionar import en Authorities con el Certificate Manager del navegador.



He modificado el archivo de configuración disponible en <https://pki-tutorial.readthedocs.io/en/latest/advanced/server.conf.html> con el que crearé el certificado firmado por serverca para usar en la web de apache.

```
pki@pki:~/signca/serverca$ cat etc/serverca-req.conf
[ default ]
SAN                = DNS:pki.if3-15.asir    # Default value
COMMONNAME         = pki.if3-15.asir

[ req ]
default_bits       = 2048                  # RSA key size
encrypt_key        = no                    # Protect private key
default_md          = sha256                # MD to use
utf8                = yes                  # Input is UTF-8
string_mask        = utf8only              # Emit UTF-8 strings
prompt             = no                    # Prompt for DN
distinguished_name = server_dn             # DN template
req_extensions      = server_reqext        # Desired extensions

[ server_dn ]
countryName        = "NO"
stateOrProvinceName = "Pontevedra"
localityName       = "Cangas"
organizationName    = "hjuncal org"
organizationalUnitName = "hjuncal Sign Server Certificate Authority"
commonName          = $ENV::COMMONNAME

[ server_reqext ]
keyUsage            = critical,digitalSignature,keyEncipherment
extendedKeyUsage    = serverAuth,clientAuth
subjectKeyIdentifier = hash
subjectAltName       = $ENV::SAN
pki@pki:~/signca/serverca$
```

Creo la solicitud de firma del servidor apache para el dominio pki.if3-15.asir

```
openssl req -new -config etc/serverca-req.conf -out pki.csr -keyout private/pki.key
```

Creo el certificado firmado con la ServerCA

```
openssl ca -config etc/serverca.conf -in pki.csr -out certs/pki.crt -extensions server_ext
```

```
-----
pki@pki:~/signca/serverca$ openssl ca -config etc/serverca.conf -in pki.csr -out certs/pki.crt -extensions server_ext
Using configuration from etc/serverca.conf
Enter pass phrase for /home/pki/signca/serverca/private/serverca.key:
Check that the request matches the signature
Signature ok
Certificate Details:
  Serial Number: 4 (0x4)
  Validity
    Not Before: Mar 12 09:44:20 2024 GMT
    Not After : Mar 12 09:44:20 2026 GMT
  Subject:
    countryName           = NO
    stateOrProvinceName   = Pontevedra
    localityName          = Cangas
    organizationName       = hjuncal org
    organizationalUnitName = hjuncal Sign Server Certificate Authority
    commonName             = pki.if3-15.asir
  X509v3 extensions:
    X509v3 Key Usage: critical
      Digital Signature, Key Encipherment
    X509v3 Basic Constraints:
      CA:FALSE
    X509v3 Extended Key Usage:
      TLS Web Server Authentication, TLS Web Client Authentication
    X509v3 Subject Key Identifier:
      88:ED:14:A7:8F:FB:0A:92:87:74:40:14:CA:BD:EF:BB:EE:D5:9D:88
    X509v3 Authority Key Identifier:
      7A:CC:2E:18:57:46:48:82:59:84:E2:92:5E:8F:EE:57:1C:32:51:9D
    Authority Information Access:
      CA Issuers - URI:https://pki.if3-15.asir/serverca/serverca.cer
    X509v3 CRL Distribution Points:
      Full Name:
        URI:https://pki.if3-15.asir/serverca/serverca.crl
    X509v3 Subject Alternative Name:
      DNS:pki.if3-15.asir
Certificate is to be certified until Mar 12 09:44:20 2026 GMT (730 days)
Sign the certificate? [y/n]:y

1 out of 1 certificate requests certified, commit? [y/n]y
Write out database with 1 new entries
Database updated
pki@pki:~/signca/serverca$
```

Ahora que ya tengo las CA root y server, además del certificado para pki.if3-15.asir creado y he agregado tanto al servidor dns bind9 como al /etc/hosts de mi ordenador ese dominio, apuntando a la ip del contenedor www, voy a crear un bundle entre pki.cert y serverca.cert

```
cat signca/serverca/certs/pki.crt signca/serverca/certs/serverca.crt > pki-chain.pem
```

En el contenedor `www` instalo apache y me transfiero con `scp` la cadena. Modifico `default-ssl.conf` de la siguiente forma:

```
VirtualHost *:443>
ServerAdmin webmaster@localhost
ServerName pki.if3-15.asir
DocumentRoot /var/www/html

# Available loglevels: trace8, ..., trace1, debug, info, notice, warn,
# error, crit, alert, emerg.
# It is also possible to configure the loglevel for particular
# modules, e.g.
#LogLevel info ssl:warn

ErrorLog ${APACHE_LOG_DIR}/error.log
CustomLog ${APACHE_LOG_DIR}/access.log combined

# For most configuration files from conf-available/, which are
# enabled or disabled at a global level, it is possible to
# include a line for only one particular virtual host. For example the
# following line enables the CGI configuration for this host only
# after it has been globally disabled with "a2disconf".
#Include conf-available/serve-cgi-bin.conf

# SSL Engine Switch:
# Enable/Disable SSL for this virtual host.
SSLEngine on

# A self-signed (snakeoil) certificate can be created by installing
# the ssl-cert package. See
# /usr/share/doc/apache2/README.Debian.gz for more info.
# If both key and certificate are stored in the same file, only the
# SSLCertificateFile directive is needed.
#SSLCertificateFile /etc/ssl/certs/ssl-cert-snakeoil.pem
#SSLCertificateKeyFile /etc/ssl/private/ssl-cert-snakeoil.key

SSLCertificateFile /etc/ssl/certs/pki-chain.pem
SSLCertificateKeyFile /etc/ssl/private/pki.key
SSLCertificateChainFile /etc/ssl/certs/pki-chain.pem

# Server Certificate Chain:
# Point SSLCertificateChainFile at a file containing the
# concatenation of PEM encoded CA certificates which form the
# certificate chain for the server certificate. Alternatively
# the referenced file can be the same as SSLCertificateFile
# when the CA certificates are directly appended to the server
# certificate for convinience.
#SSLCertificateChainFile /etc/apache2/ssl.crt/server-ca.crt

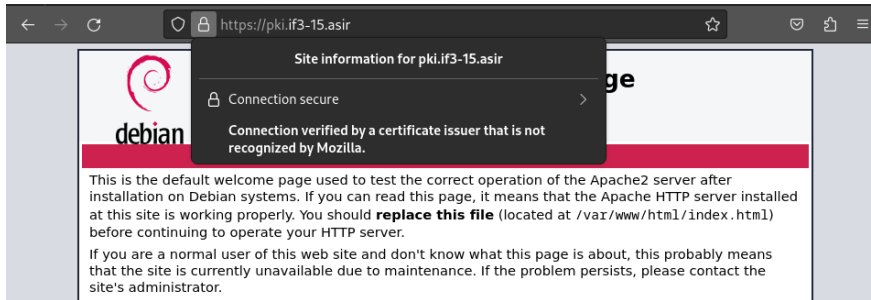
# Certificate Authority (CA):
# Set the CA certificate verification path where to find CA
/etc/apache2/sites-available/default-ssl.conf
```

Habilito el módulo ssl con `a2enmod ssl` y habilito el sitio con `a2ensite default-ssl`
Ahora, reinicio el servicio apache2.

Agrego una ruta a mi ordenador para llegar al contenedor www saltando a traves de gateway

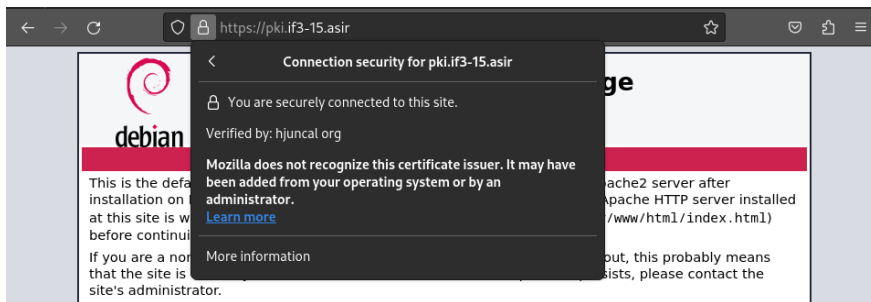
`ip route add 172.30.0.0/16 via 172.20.30.15`

Ahora, en el ordenador, agrego pki.if3-15.asir al archivo /etc/hosts para no cambiar el servidor dns al que mi ordenador apunta, pues podría dar problemas dado que lo necesita para resoluciones relacionadas con el dominio del instituto.



La página carga sin warning y la conexión es segura.

Se puede ver que está certificado por mi CA



Aquí se ve que firefox muestra los dos certificados concatenados (ServerCA firmó el certificado pki.if3-15.asir) y el certificado RootCA, que firmó el certificado de ServerCA

