

Aquí vamos a explicar como habilitar el protocolo IPv6 en Docker, esto permitirá comunicar nuestros contenedores con el mundo exterior por este protocolo aparte del conocido IPv4, funciona solamente en GNU/Linux

Para esto hay que tener configurado en el servidor anfitrión IPv6 en la interfaz WAN (la que llega a internet), en algunos proveedores puede venir activado por defecto pero no es la norma, para probar si tenemos IPv6 en el anfitrión haremos un ping6

```
$ ping6 2600::
```

Si responde correctamente las solicitudes ICMPv6 tendremos IPv6 en la máquina, sino hay que negociar con el proveedor como habilitarlo

En el archivo `/etc/docker/daemon.json` (si no existe lo creamos) hay que añadir lo siguiente como se indica en la documentación de Docker

```
{
  "ipv6": true,
  "fixed-cidr-v6": "2001:db8:1::/80"
}
```

El parámetro `fixed-cidr-v6` indica la subred interna que usarán los contenedores, es decir, la subred de la interfaz bridge que normalmente identificamos con `docker0`, se configurará con la primera dirección de la red disponible

Se elige la máscara de red `/80` que es menor que la que hay en la interfaz externa `/64` para evitar problemas en el enrutado de los paquetes IPv6, lo recomendable es usar una máscara con un rango menor, si ponemos una máscara `/64` o del mismo valor que la externa se tiene que usar algún demonio de enrutado como `ndppd`

Una vez se reinicia Docker puede verse configurada la interfaz interna de red bridge con IPv6 habilitado y la consiguiente red asignada con el comando

```
$ docker network inspect bridge
```

Para usar IPv6 en los contenedores habrá que enrutar el tráfico de vuelta hacia la interfaz externa haciendo masquerading con el comando *ip6tables*

```
$ ip6tables -t nat -A POSTROUTING -s 2001:db8:1::/80 -j MASQUERADE
```

Además hay que configurar el sysctl de la máquina para preparar el kernel con un entorno IPv6, añadimos las siguientes líneas al archivo `/etc/sysctl.d/99-sysctl.conf` modificando la interfaz de red de salida por la correspondiente (hay que reiniciar la máquina)

```
net.ipv6.conf.default.accept_ra=0
net.ipv6.conf.all.accept_ra=0
net.ipv6.conf.default.forwarding=1
net.ipv6.conf.all.forwarding=1
net.ipv6.bindv6only=0
net.ipv6.conf.default.disable_ipv6=0
net.ipv6.conf.all.disable_ipv6=0
net.ipv6.conf.default.proxy_ndp=0
net.ipv6.conf.all.proxy_ndp=0
net.ipv6.conf.ensXXX.proxy_ndp=1
net.ipv6.conf.ensXXX.accept_ra=2
net.ipv6.conf.ensXXX.forwarding=0
```

Solo queda instanciar un contenedor y comprobar que tenemos ping6 desde él

```
$ docker run -ti --rm debian:buster ping6 -c10 2600::
PING 2600::(2600::) 56 data bytes
64 bytes from 2600::: icmp_seq=1 ttl=52 time=140 ms
64 bytes from 2600::: icmp_seq=2 ttl=52 time=140 ms
64 bytes from 2600::: icmp_seq=3 ttl=52 time=140 ms
64 bytes from 2600::: icmp_seq=4 ttl=52 time=140 ms
64 bytes from 2600::: icmp_seq=5 ttl=52 time=140 ms
64 bytes from 2600::: icmp_seq=6 ttl=52 time=140 ms
64 bytes from 2600::: icmp_seq=7 ttl=52 time=141 ms
64 bytes from 2600::: icmp_seq=8 ttl=52 time=140 ms
64 bytes from 2600::: icmp_seq=9 ttl=52 time=140 ms
64 bytes from 2600::: icmp_seq=10 ttl=52 time=140 ms

--- 2600:: ping statistics ---
```

```
10 packets transmitted, 10 received, 0% packet loss, time 20ms  
rtt min/avg/max/mdev = 139.906/140.204/141.457/0.475 ms
```

Y colorín colorado este post se ha terminado, esperamos os haya gustado, nos vemos en la próxima