

# Servers & Applications

# IPv6 Roadshow

Beirut  
December, 2012

Job Snijders  
Sander Steffann



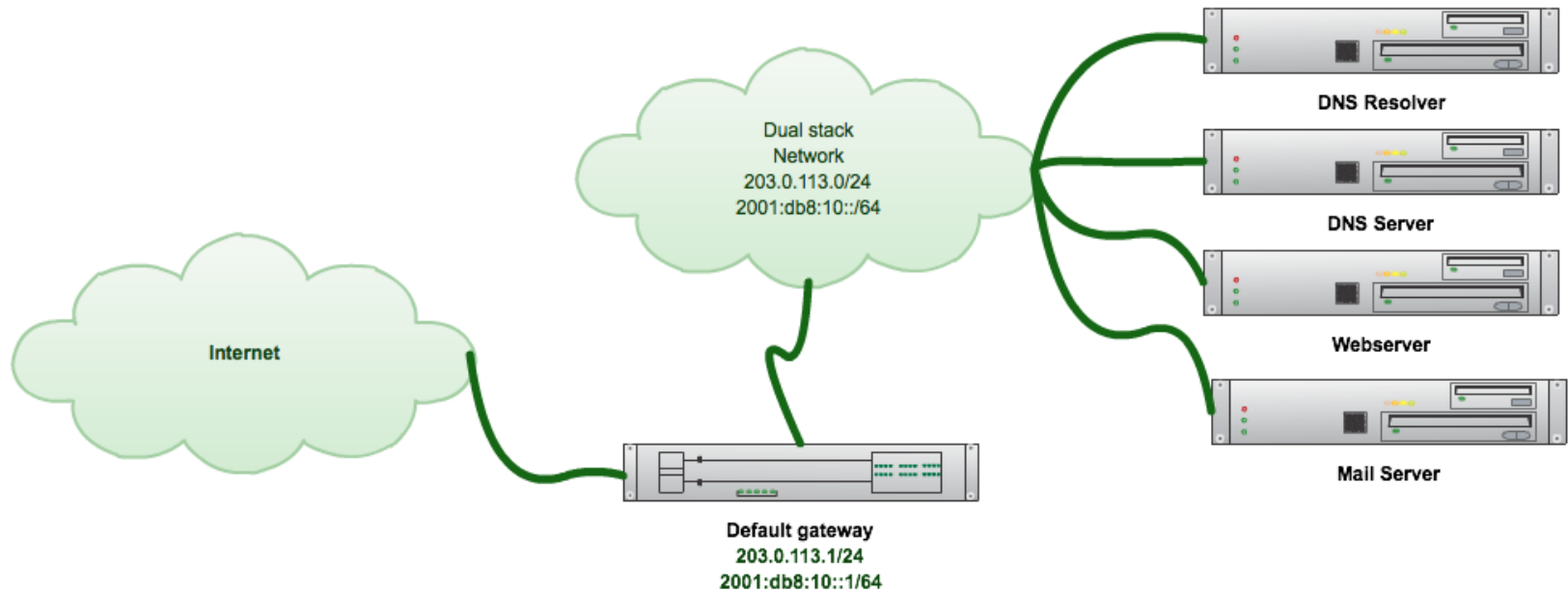
# **Servers and Applications: DNS, Mail, Web, Fw**

- How to enable IPv6 on your servers and configure common applications?
- DNS resolving
- How clients treat A & AAAA record
- Mail (postfix)
- Web (apache)
- Firewall (iptables)



- Basic connectivity
  - How to configure an IPv6 address on the servers
  - Configure DNS Resolvers and use them
  - Setup basic firewall
- **Please note, all configuration examples are the bare minimum! Tuning for production environment is left as an exercise to the reader.**

# Basic Network Overview



# Addressing Plan

Server Role	IPv4 address	IPv6 address
Gateway	203.0.113.1	2001:db8:a2e:10::1
DNS Resolver	203.0.113.10	2001:db8:a2e:10::10
DNS Server	203.0.113.11	2001:db8:a2e:10::11
Webserver	203.0.113.80	2001:db8:a2e:10::80
Mail Server	203.0.113.25	2001:db8:a2e:10::25

# Connect Your Servers to IPv6

---

- Datacenter objectives:
- Dual stack everything
- Disable automatic IPv6 configuration, everything must be static
  - You might want to move addresses between servers
  - SLAAC makes your IPv6 address change when you swap (broken) network adapters
  - You usually don't want your servers to be dependent on things like DHCP servers

- Assuming eth0 is the interface connected to the LAN, modify **/etc/network/interfaces** accordingly

```
# eth0 is the interface connected to the LAN
# regular IPv4 stanza
auto eth0
iface eth0 inet static
    address 203.0.113.10
    netmask 255.255.255.0
    network 203.0.113.0
    gateway 203.0.113.1
    broadcast 203.0.113.255

# IPv6 portion
iface eth0 inet6 static
    pre-up modprobe ipv6
    address 2001:db8:a2e:10::10
    netmask 64
    gateway 2001:db8:a2e:10::1
    # disable autoconfiguration:
    post-up echo 0 > /proc/sys/net/ipv6/conf/default/accept_ra
    post-up echo 0 > /proc/sys/net/ipv6/conf/all/accept_ra
    post-up echo 0 > /proc/sys/net/ipv6/conf/$IFACE/accept_ra
    post-up echo 0 > /proc/sys/net/ipv6/conf/default/autoconf
    post-up echo 0 > /proc/sys/net/ipv6/conf/all/autoconf
    post-up echo 0 > /proc/sys/net/ipv6/conf/$IFACE/autoconf
```



# Network: Ubuntu/Debian Configuration (2) **IPv6** Roadshow

- Restart network configuration with this command:

```
/etc/init.d/networking restart
```

- More information at <https://wiki.ubuntu.com/IPv6>

# Network: Redhat Configuration (1)

- Append the following lines to */etc/sysconfig/network*

```
NETWORKING_IPV6=yes  
IPV6_AUTOCONF=no
```

- Configure the first network interface in */etc/sysconfig/network-scripts/ifcfg-eth0*

```
DEVICE=eth0  
BOOTPROTO=static  
ONBOOT=yes  
HWADDR=00:30:48:33:bc:33  
IPADDR=203.0.113.10  
GATEWAY=203.0.113.1  
NETMASK=255.255.255.0  
IPV6INIT=yes  
IPV6ADDR=2001:0db8:0010:0a2e:0000:0000:0000:0010  
IPV6_DEFAULTGW=2001:0db8:0010:0a2e:0000:0000:0000:0001
```

# Network: Redhat Configuration (2)

- Restart networking configuration with this command:

```
service network restart
```

- More information at <http://ipv6.redhat.com/>

- On Windows 2003 the following commands can be used to configure a static IPv6 address from the command line if the IPv6 extension pack has been installed

```
netsh interface ipv6 set privacy disabled
netsh interface ipv6 add address interface="Local Area Connection" \
    address=2001:db8:a2e:10::10 store=persistent
netsh interface ipv6 add route ::/0 interface="Local Area Connection" \
    2001:db8:a2e:10::1 store=persistent
```



# Network: Windows Configuration

---

- The Windows 2003 IPv6 stack is very rudimentary, and services like IIS don't provide full IPv6 support
- Windows 2008 is strongly recommended
- On Windows 2008 the IPv6 settings can be accessed through the standard network configuration graphical interfaces
- More information: <http://technet.microsoft.com/en-us/network/bb530961.aspx>

# DNS: Resolving (1)

---

- A DNS resolver is responsible for resolving records to resources, e.g. translation of a domain name into an IP address
- A resolver will be recursive through various queries if needed on behalf of the client

- A variety of great resolvers exist which support both IPv4 and IPv6:
  - Unbound: <http://unbound.net/>
  - PowerDNS Recursor (used in this example):  
<http://www.powerdns.com/content/home-powerdns.html>
  - BIND: <http://www.isc.org/software/bind>

# DNS: Install PowerDNS Recursor

- Install PowerDNS Recursor:

- Ubuntu/Debian:

```
sudo apt-get install pdns-recursor
```

- Fedora/Redhat/Centos:

```
yum install pdns-recursor
```



# DNS: Configure PowerDNS Recursor (1)

- Listen on IPv4 and IPv6 address
- Accept queries from the local LAN
  - In **/etc/powerdns/recursor.conf** the following lines are needed:

```
aaaa-additional-processing=on
allow-from=203.0.113.0/24,2001:db8:a2e:10::/64
dont-query=
local-address=203.0.113.10,2001:db8:a2e:10::10
query-local-address6=2001:db8:a2e:10::10
local-port=53
quiet=yes
setgid=pdns
setuid=pdns
logging-facility=0
```

# DNS: Configure PowerDNS Recursor (2)

- Restart the recursor:

```
sudo service pdns-recursor restart
```

- Verify its operation:

```
root@recursor:/etc/powerdns# netstat -lpn | grep :53.*LISTEN
tcp    0  0  203.0.113.10:53          0.0.0.0:*        LISTEN 5316/pdns_recursor
tcp6   0  0  2001:db8:a2e:10::10:53   :::*             LISTEN 5316/pdns_recursor
root@recursor:/etc/powerdns#
```

# DNS: Configure Hosts to Use the Recursor

- Configure all servers to use the previously configured DNS Recursor
- In **/etc/resolv.conf**:

```
search example.net  
nameserver 203.0.113.10  
nameserver 2001:db8:a2e:10::10
```

- Various commands can be used to test if basic connectivity works.
  - Check if interfaces have an IPv6 address and a default route exists:

```
job@resolver:~$ ip -6 addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 16436
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qlen 1000
    inet6 2001:db8:a2e:10::10/64 scope global
        valid_lft forever preferred_lft forever
    inet6 fe80::250:56ff:fe91:19/64 scope link
        valid_lft forever preferred_lft forever
job@resolver:~$ ip -6 route
2001:db8:a2e:10::/64 dev eth0 proto kernel metric 256
fe80::/64 dev eth0 proto kernel metric 256
default via 2001:db8:a2e:10::1 dev eth0 metric 1024
```



## DNS: Testing IPv6 Connectivity and Resolving (2)

- **ping**

```
ping6 2001:db8:a2e:10::1      # ping default gateway
ping6 2001:7fd::1            # ping K Root-Server
ping6 www.ripe.net           # ping RIPE website
```

- **traceroute**

```
traceroute6 2001:7fd::1
traceroute6 www.ripe.net
```

- Executive summary:
- DNS is more then ever important with IPv6
  - Nobody wants to remember the long addresses! 😊
- In a dual-stack environment for every host twice the amount of records: both IPv4 and IPv6
- A few new record-types have been introduced for IPv6 related records

# DNS: Records (1)

- Forward records

- In IPv4 a forward or A record might look like this:

```
$ORIGIN example.net.  
www      IN      A      203.0.113.80
```

- But an IPv6 record looks like this:

```
$ORIGIN example.net.  
www      IN      AAAA    2001:db8:a2e:10::80
```

- Both are needed if a hostname needs to be resolved to both its IP addresses!





# DNS: Records (2)

- Notice how the IPv6 is expanded to full notation and then reversed?
  - sipcalc is a good command-line utility to easily display such information:

```
job@nameserver:/etc/powerdns$ sipcalc -r 2001:db8:10:0a2e::80
-[ipv6 : 2001:db8:10:0a2e::80]- 0

[IPV6 DNS]
Reverse DNS (ip6.arpa) -
0.8.0.0.0.0.0.0.0.0.0.0.0.0.0.0.e.2.a.0.0.1.0.0.8.b.d.0.1.0.0.2.ip6.arpa.

job@nameserver:/etc/powerdns$
```

- Now that basic connectivity and DNS resolving have been established we proceed to configure a Authoritative DNS Server for the **example.net** zone

# DNS: Install PowerDNS (1)

```
sudo apt-get install pdns-server; mkdir /etc/powerdns/zones
```

- An example of `/etc/powerdns/pdns.conf` can be the following:

```
local-address=203.0.113.11  
local-ipv6=2001:db8:a2e:10::11  
launch=bind  
bind-config=/etc/powerdns/named.conf  
setgid=pdns setuid=pdns
```

# DNS: Install PowerDNS (2)

- **/etc/powerdns/named.conf** can contain the following:

```
options {
    directory "/etc/powerdns";
    recursion no;
};

zone "example.net" {
    type master;
    file "zones/example.net";
};

zone "113.0.203.in-addr.arpa" {
    type master;
    file "zones/113.0.203.in-addr.arpa";
};

zone "0.1.0.0.e.2.a.0.8.b.d.0.1.0.0.2.ip6.arpa" {
    type master;
    file "zones/0.1.0.0.e.2.a.0.8.b.d.0.1.0.0.2.ip6.arpa";
};
```

# DNS: The Actual example.net Zone

- Contents of `/etc/powerdns/zones/example.net`

```
; BIND db file for example.net
$TTL 86400
@      IN      SOA      nameserver.example.net. hostmaster.example.net. (
        2012012701 ; serial number YYYYMMDDNN
        24h        ; Refresh
        30m        ; Retry
        2d         ; Expire
        86400      ; Min TTL
)
        IN      NS       nameserver.example.net.
        MX 10      mailserver.example.net.
$ORIGIN example.net.
gateway      IN      A       203.0.113.1
              AAAA      2001:db8:a2e:10::1
recursor     IN      A       203.0.113.10
              AAAA      2001:db8:a2e:10::10
nameserver   IN      A       203.0.113.11
              AAAA      2001:db8:a2e:10::11
www          IN      A       203.0.113.80
              AAAA      2001:db8:a2e:10::80
database     IN      A       203.0.113.36
              AAAA      2001:db8:a2e:10::36
mailserver   IN      A       203.0.113.25
              AAAA      2001:db8:a2e:10::25
```

# IPv6 Roadshow

- Contents of **/etc/powerdns/zones/0.1.0.0.e.2.a.0.8.b.d.0.1.0.0.2.ip6.ar**

[illegible]



# DNS: Example IPv4 reverse zone

- Contents of `/etc/powerdns/zones/113.0.203.in-addr.arpa`

```
$TTL 86400
@      IN      SOA      nameserver.example.net. hostmaster.example.net. (
      201201270      ; Serial number (YYYYMMdd)
      24h            ; Refresh time
      30m            ; Retry time
      2d             ; Expire time
      86400          ; Default TTL (bind 8 ignores this, bind 9 needs it)
)
      IN      NS       nameserver.example.net.
$ORIGIN 113.0.203.in-addr.arpa
1      IN      PTR      gateway.example.net.
10     IN      PTR      recursor.example.net.
11     IN      PTR      nameserver.example.net.
25     IN      PTR      mailserver.example.net.
36     IN      PTR      database.example.net.
80     IN      PTR      www.example.net.
```

# DNS: Verify DNS Operation (1)

---

- Apply new configuration:

```
sudo service pdns restart
```

# DNS: Verify DNS Operation (2)

- Verify the Authoritative DNS Server is working on IPv4 and IPv6:

```
job@recursor:/etc/powerdns$ host www.example.net 203.0.113.11
Using domain server:
Name: 203.0.113.11
Address: 203.0.113.11#53
Aliases:

www.example.net has address 203.0.113.80
www.example.net has IPv6 address 2001:db8:a2e:10::80

job@recursor:/etc/powerdns$ host www.example.net 2001:db8:a2e:10::11

Using domain server:
Name: 2001:db8:a2e:10::11
Address: 2001:db8:a2e:10::11#53
Aliases:

www.example.net has address 203.0.113.80
www.example.net has IPv6 address 2001:db8:a2e:10::80

job@recursor:/etc/powerdns$
```

- Verify reverse DNS:

[illegible]

# Clients: A & AAAA (1)

---

- In 1994 the RFC1671 document already stated: "The dual-stack code may get two addresses back from DNS; which one to use?"
- After all, we did put both A & AAAA records into the DNS zone for example.net.
- RFC3484 "Default Address Selection for Internet Protocol version 6 (IPv6)" describes a selection algorithm, but performs poorly in practice.

- RFC 6555 (Happy Eyeballs) algorithm that aims to solve that problem:
  - Provide fast experience for end-users:
    1. Quickly attempt to connect using IPv6, and if that connection attempt is not fast enough successfully
    2. Connect using IPv4
- Avoids thrashing the network, by not always making simultaneous connection attempts on both IPv6 and IPv4



# Clients: A & AAAA (3)

- How does Happy Eyeballs work?
- Two DNS requests were sent, IPv4 was used because IPv6 didn't get a response fast enough:

	DNS Server	Client	Server
1.			
	<--www.example.com A?-----		
2.	<--www.example.com AAAA?--		
3.	---192.0.2.1----->		
4.	---2001:db8::1----->		
5.			
6.		==TCP SYN, IPv6===>X	
7.		--TCP SYN, IPv4----->	
8.		<-TCP SYN+ACK, IPv4----	
9.		--TCP ACK, IPv4----->	
10.		==TCP SYN, IPv6===>X	

- **Current status within various software projects:**
  - Chrome Browser: first to implement Happy Eyeballs
  - MacOSX Snow Leopard: good implementation
  - Mozilla Firefox 10: supports Happy Eyeballs
  - Windows 7: no support for Happy Eyeballs
  - MacOSX Lion: complicated implementation, doesn't select IPv6 in half the cases when both IPv4 and IPv6 are present

# **Servers and Applications: Web, Mail, DB**

# Applications: Web, Mail, DB

---

- Now that we have connectivity and DNS out of the way, time for the easier applications 😊

- **Goals:**
- Setup SMTP daemon
  - Accept mail for example.net domain
  - Accept mail from 203.0.113.0/24, 2001:db8:a2e:10::/64 and relay to the internet
- Setup IMAP daemon
- Setting up a mail server that is capable of receiving and sending email over both IPv4 and IPv6 is easy!

# Mail: Install Postfix and Courier

```
sudo apt-get -y install postfix mailutils courier-imap

sudo postconf -e "mydestination = example.net, mailserver.example.net, localhost.localdomain, localhost"

sudo postconf -e "mynetworks = 203.0.113.0/24, [2001:db8:a2e:10::]/64"

sudo postconf -e "home_mailbox = Maildir/"

sudo postconf -e "inet_protocols = all"

sudo postconf -e "inet_interfaces = all"

sudo service postfix restart
```



# Mail: How to Use This Setup

---

- This is a very straight-forward non-virtual setup (e.g. great for testing)
- Users can be added with the 'adduser' system command
- Email-usernames are the same as the system-usernames (system-username@domain)
- The IMAP daemon authenticates against PAM

- **Goals:**
- Install apache and serve a simple website both on IPv4 and IPv6

```
sudo apt-get install apache2 curl
```

- **Done! 😊**
- Apache listens by default on both IPv4 and IPv6 and will show the same default page on both IPv4 and IPv6

# Web: Create Virtual Host for example.net

- Create a simple website:

```
mkdir /var/www/www.example.net  
echo Testing 1 2 3 > /var/www/www.example.net/index.html
```

- Place the following in **/etc/apache2/sites-enabled/www.example.net**

```
<VirtualHost *:80>  
    ServerAdmin webmaster@example.net  
    ServerName www.example.net  
    DocumentRoot /var/www/www.example.net  
</Virtualhost>
```

# Web: Create Virtual Host for example.net

---

- Reload apache configuration

```
apache2ctl configtest && apache2ctl graceful
```

# Web: Verify Web Server Operation

- Test if web server is listening both on IPv4 and IPv6:

```
root@www:~# netstat -ltn | grep apache
tcp6    0      0      :::80      :::*      LISTEN    20879/apache2
unix    2      [ ACC ]     STREAM  LISTENING  2824765   20985/apache2   /var/run/apache2/cgisock.20879
root@www:~#
```

- Test our small website is displayed:

```
root@www:~# curl -6 www.example.net Testing 1 2 3
root@www:~# curl -4 www.example.net Testing 1 2 3
root@www:~#
```



- **Other:**
  - NGINX
  - Apache2
  - HAProxy

## Firewall: DROP and ALLOW Some With ufw (1)

- *iptables* & *ip6tables* are the de-facto firewalls on Linux, the BSD distributions usually have *pf*.
- Both *iptables* and *pf* are excellent and powerful tools!
- In this example we will use *ufw* as a simple front-end to *iptables* and *ip6tables*

```
apt-get install ufw
```

- Ensure IPv6 support is enabled:

```
root@www:~# grep IPV6 /etc/default/ufw
IPV6=yes
root@www:~#
```

# Firewall: DROP and ALLOW Some With ufw (2)

```
root@www:/etc/default# ufw status
Status: inactive
```

```
root@www:/etc/default# ufw enable
Command may disrupt existing ssh connections. Proceed with operation (y|n)? Y
Firewall is active and enabled on system startup
```

```
root@www:/etc/default# ufw app list
Available applications:
```

```
    Apache
    Apache Full
    Apache Secure
    OpenSSH
```

```
root@www:/etc/default# ufw allow OpenSSH
Rule added
Rule added (v6)
```

```
root@www:/etc/default# ufw allow "Apache Full"
Rule added
Rule added (v6)
```

# Firewall: DROP and ALLOW Some With ufw (3)

```
root@www:~# ufw status numbered
Status: active
```

To	Action	From
--	-----	----
[ 1] Apache Full	ALLOW IN	Anywhere
[ 2] OpenSSH	ALLOW IN	Anywhere
[ 3] Apache Full (v6)	ALLOW IN	Anywhere (v6)
[ 4] OpenSSH (v6)	ALLOW IN	Anywhere (v6)

```
root@www:~#
```

## Firewall: DROP and ALLOW Some With ufw (4)

- Restrict access to SSH only to the management subnet:

```
root@www:~# ufw delete 4
Deleting:
  allow from any to any app OpenSSH Proceed with operation (y|n)? Y
Rule deleted (v6)

root@www:~# ufw delete 2
Deleting:
  allow from any to any app OpenSSH Proceed with operation (y|n)? Y
Rule deleted

root@www:~# ufw allow from 192.168.15.0/24 to any app OpenSSH
Rule added

root@www:~# ufw allow from 2001:db8:10:20::/64 to any app OpenSSH
Rule added (v6)

root@www:~#
```

- Restrict access to SSH only to the management subnet:

```
root@www:~# ufw status numbered
Status: active
```

To	Action	From
--	-----	----
[ 1] Apache Full	ALLOW IN	Anywhere
[ 2] OpenSSH	ALLOW IN	192.168.15.0/24
[ 3] Apache Full (v6)	ALLOW IN	Anywhere (v6)
[ 4] OpenSSH (v6)	ALLOW IN	2001:db8:10:20::/64

```
root@www:~#
```



- In this module we learned how to setup the following:
  - DNS resolving
  - Authoritative DNS
  - Mail server
  - Web server
  - Simple firewall
- As demonstrated, lots of popular open-source projects support dual-stack setups out of the box.

# Further Reading

---

- Peter Bieringer has written an excellent Linux IPv6 Howto:
  - <http://www.tldp.org/HOWTO/Linux+IPv6-HOWTO/>