AUTOMATING APACHE TRAFFIC CONTROL SERVER DEPLOYMENTS WITH IPV6

September 9, 2019

COMCAST

ABOUT THE SPEAKER

JASON TUCKER – SR. PRINCIPAL ENGINEER (SRE), IPCDN OPS

Over two decades of experience working in large-scale environments, mostly in the Cable/MSO industry. Much of that time has been spent in technical operations. I've been with Comcast for over 14 years, in a variety of Ops roles, both as an engineer and team manager.

For the past ~2.5 years, I've been part of the SRE team managing the Comcast CDN, building tools and processes to help a small team manage a relatively large CDN that spans our national footprint. Particular area of focus as of late is automation and configuration management.

I have experience with CFEngine, Puppet, and Ansible; I'm a big fan of ansible-pull in large environments. If anyone wants to discuss these or have a religious debate about them, I'm game!

COMCAST

THE PROBLEM

MOTIVATION FOR CHANGE

THE PROBLEM

TYPICAL HOST PROVISIONING PROCESS

- Generate a unique install ISO for every bare-metal host that gets deployed
  - Time consuming
  - Sometimes difficult to parallelize
  - Repetitive. Repetitive. Repetitive, ad nauseum.

When you need to deploy a large number of servers in a short amount of time, less steps and better automation is a necessity.

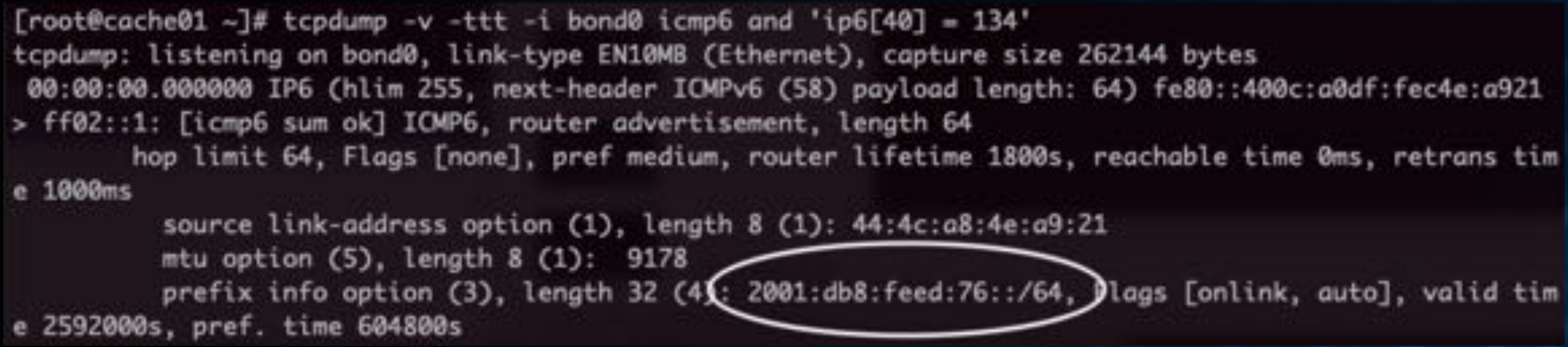
A NEW APPROACH

LEVERAGING THE BENEFITS OF IPv6

IPv6 NEIGHBOR DISCOVERY PROTOCOL

ONE OF THE BENEFITS OF IPv6 IS NDP, WHICH ALLOWS FOR AUTOMATIC (AND STATELESS) NETWORK ADDRESS PROVISIONING FOR CLIENTS

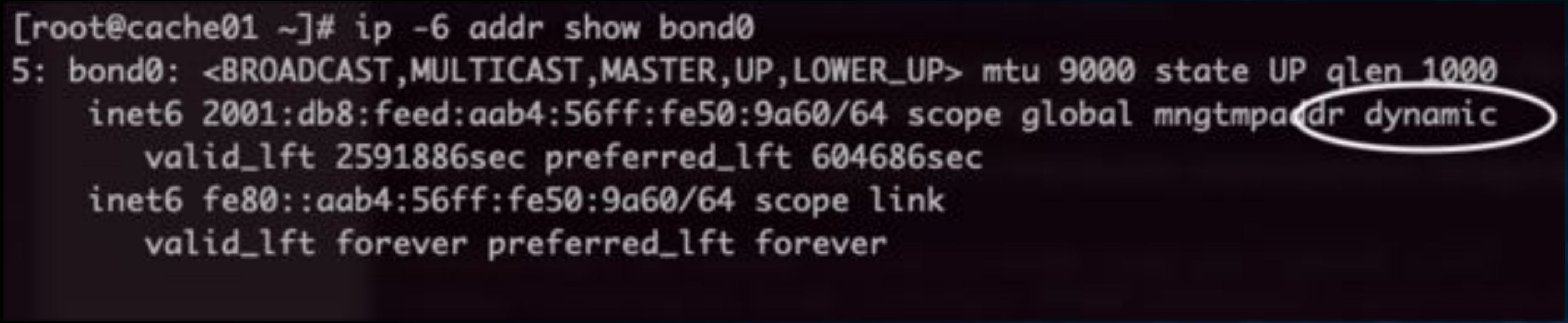
- Router advertises (at some configured interval, or upon request from client) ICMPv6 messages to aid in client interface autoconf
- Below example shows a capture of a IPv6 Router Advertisement datagram:



```
[root@cache01 ~]# tcpdump -v -ttt -i bond0 icmp6 and 'ip6[40] = 134'
tcpdump: listening on bond0, link-type EN10MB (Ethernet), capture size 262144 bytes
 00:00:00.000000 IP6 (hlim 255, next-header ICMPv6 (58) payload length: 64) fe80::400c:a0df:fec4e:a921
> ff02::1: [icmp6 sum ok] ICMP6, router advertisement, length 64
        hop limit 64, Flags [none], pref medium, router lifetime 1800s, reachable time 0ms, retrans tim
e 1000ms
          source link-address option (1), length 8 (1): 44:4c:a8:4e:a9:21
          mtu option (5), length 8 (1):  9178
          prefix info option (3), length 32 (4): 2001:db8:feed:76::/64, Flags [onlink, auto], valid tim
e 2592000s, pref. time 604800s
```

IPv6 NEIGHBOR DISCOVERY PROTOCOL

EXAMPLE ADDRESS OBTAINED VIA AUTOCONF



```
[root@cache01 ~]# ip -6 addr show bond0
5: bond0: <BROADCAST,MULTICAST,MASTER,UP,LOWER_UP> mtu 9000 state UP qlen 1000
    inet6 2001:db8:feed:aab4:56ff:fe50:9a60/64 scope global mngtmpaddr dynamic
       valid_lft 2591886sec preferred_lft 604686sec
    inet6 fe80::aab4:56ff:fe50:9a60/64 scope link
       valid_lft forever preferred_lft forever
```

- NB! With IPv6, it is not necessary for your default gateway to be a member of the same prefix as your host! Our autoconf gateway may simply be the link-local address on the router.

UNIVERSAL ISO CONFIG

IMAGE SETUP

In the geniso directory on Traffic Ops server for whichever OS version you want to set up for autoconf, the following files need to be customized:

GRUB configs (legacy and/or EFI depending on your environment needs):

- `./isolinux/append.cfg`
- `./EFI/BOOT/grub.cfg`
  - ip=eth0:auto6
    (or, whatever interface name is correct for your environment)
  - nameserver=2001:db8:feed:beef::1
    (use a real, reachable IPv6 nameserver here!)

COMCAST

UNIVERSAL ISO CONFIG

IMAGE SETUP (CONTINUED)

Post-install network config – this is the default config that gets placed on the freshly-built server:

- `./ks_scripts/network.cfg`

  -
    ```
    IPV6INIT="yes"
    IPV6_AUTOCONF="yes"
    IPV6_DEFROUTE="yes"
    NAMESERVER="2001:db8:feed:beef::1"
    ```

  This allows the server to come up with IPv6 autoconf address on first boot, which will be be reconfigured automatically later with the "real" network config settings.

COMCAST

OS TUNING CONSIDERATIONS

KERNEL SYSCTL SETTIONS CONTROL AUTOCONF

```
net.ipv6.conf.default.autoconf = 1
net.ipv6.conf.default.accept_ra = 1
```

For out-of-the-box CentOS, these are usually enabled by default; however, it is normal network hardening practice to disable these options, by setting them both to "0". If this is something you do in your environment, you just need to be sure to only disable these *after* final network configuration for the new server is complete. This is something to be considered if you automate this with some sort of config management tool.

NB! I’ve noticed that if CentOS 7 is configured for both static IPv6 *and* autoconf, the host may prefer to use the autoconf address, which may not be desired. Config management ensures that autoconf gets disabled once the static address is configured.

INFRASTRUCTURE CONSIDERATIONS

CRITICAL NETWORK SERVICES FOR INSTALL REQUIRE IPv6 CONNECTIVITY

Remember that during kickstart, your new server will be single-stack IPv6 *only*. Any services required to complete the install will also need to be IPv6-enabled.

- Yum
  - The network-based kickstart triggered by the install image requires yum to load OS packages
- Config Managment
  - Does your post-install run config mgmt tools?
  - Puppet, ansible-pull, etc.
- Traffic Ops API
  - This may be a new dependency for you
  - A new tool, 'tc-netconfig' depends on access to TOAPI to retrieve the servers static network configs

COMCAST

NEW TOOL: tc-netconfig

THE GLUE PROGRAM THAT ALLOWS A DYNAMICALLY CONFIGURE SERVER TO FETCH ITS STATIC NETWORK CONFIG

https://github.com/comcast/tc-netconfig

- A utility to complete host-specific network configuration on an autoconf'd host
- Written in go
- Can be packaged into an RPM (example install scripts and systemd unit files provided in repo)
- Uses Traffic Ops essentially as a "CMDB" to obtain the server's static network configs (which are assumed to be pre-loaded into TODB prior to hardware provisioning)

Config management note: up to this point, the server has no real identity beyond what was retreived from TODB. If you do type-specific configurations (based on TO server profile, etc, you need to be able to identify this info for your config management tool to complete its system setup. This may be an environmental file baked into the ISO, or even a customization of tc-netconfig itself to fetch the appropriate info from TOAPI and store it locally.

tc-netconfig DETAILS

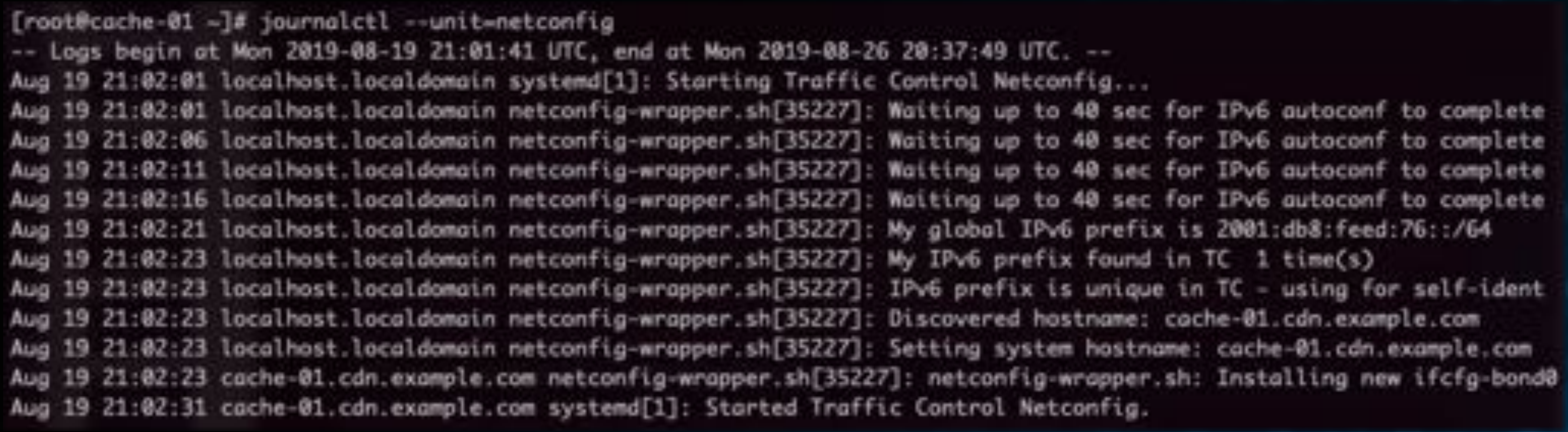
- Primarily uses assigned IPv6 prefix to self-identify the server in Traffic Ops
  - In our environment, each individual cache server gets it's own unique /64 IPv6 prefix
- If 'ipmitool' is present on the freshly-built server, BMC (ILO, DRAC, etc) LAN IP can be used as a secondary identifier
  - Guaranteed to be unique per server
  - Relies on additional dependencies (i.e. 'ipmitool' functionality), hence is only used as a secondary method

NB! If self-identification fails (broken connectivity, missing server object in TODB, etc) the server will remain in a dynamic autoconf config state indefinitely! Once a problem is corrected, a reboot will reconfigure the server. This may actually be handy if you are want to complete the generic imaging of servers in a warehouse before shipping, and once the new servers are powered on in their final destination, they will then "discover" their real network configs.

tc-netconfig DETAILS

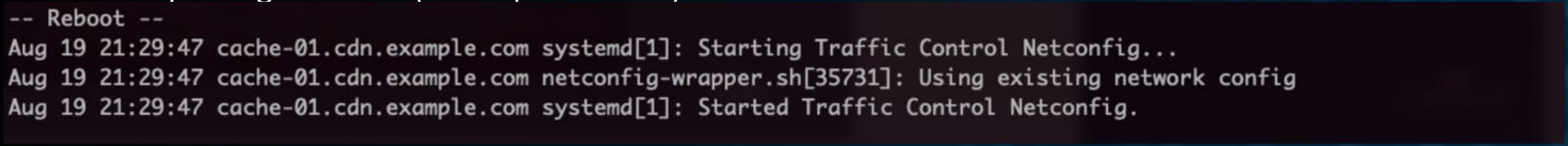
EXAMPLE LOG OUTPUT

Unconfigured host (first boot):



```
[root@cache-01 ~]# journalctl --unit=netconfig
-- Logs begin at Mon 2019-08-19 21:01:41 UTC, end at Mon 2019-08-26 20:37:49 UTC. --
Aug 19 21:02:01 localhost.localdomain systemd[1]: Starting Traffic Control Netconfig...
Aug 19 21:02:01 localhost.localdomain netconfig-wrapper.sh[35227]: Waiting up to 40 sec for IPv6 autoconf to complete
Aug 19 21:02:06 localhost.localdomain netconfig-wrapper.sh[35227]: Waiting up to 40 sec for IPv6 autoconf to complete
Aug 19 21:02:11 localhost.localdomain netconfig-wrapper.sh[35227]: Waiting up to 40 sec for IPv6 autoconf to complete
Aug 19 21:02:16 localhost.localdomain netconfig-wrapper.sh[35227]: Waiting up to 40 sec for IPv6 autoconf to complete
Aug 19 21:02:21 localhost.localdomain netconfig-wrapper.sh[35227]: My global IPv6 prefix is 2001:db8:feed:76::/64
Aug 19 21:02:23 localhost.localdomain netconfig-wrapper.sh[35227]: My IPv6 prefix found in TC  1 time(s)
Aug 19 21:02:23 localhost.localdomain netconfig-wrapper.sh[35227]: IPv6 prefix is unique in TC - using for self-ident
Aug 19 21:02:23 localhost.localdomain netconfig-wrapper.sh[35227]: Discovered hostname: cache-01.cdn.example.com
Aug 19 21:02:23 localhost.localdomain netconfig-wrapper.sh[35227]: Setting system hostname: cache-01.cdn.example.com
Aug 19 21:02:23 cache-01.cdn.example.com netconfig-wrapper.sh[35227]: netconfig-wrapper.sh: Installing new ifcfg-bond0
Aug 19 21:02:31 cache-01.cdn.example.com systemd[1]: Started Traffic Control Netconfig.
```

Previously configured host (subsequent boots):



```
-- Reboot --
Aug 19 21:29:47 cache-01.cdn.example.com systemd[1]: Starting Traffic Control Netconfig...
Aug 19 21:29:47 cache-01.cdn.example.com netconfig-wrapper.sh[35731]: Using existing network config
Aug 19 21:29:47 cache-01.cdn.example.com systemd[1]: Started Traffic Control Netconfig.
```

QUESTIONS?

https://traffic-control-cdn.slack.com

https://trafficcontrol.apache.org/mailing_lists/

COMCAST