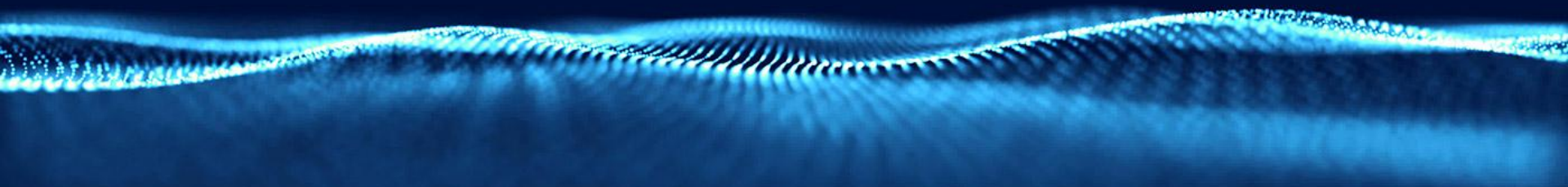
# Scaling to support thousands of BGP peerings in a SaaS environment



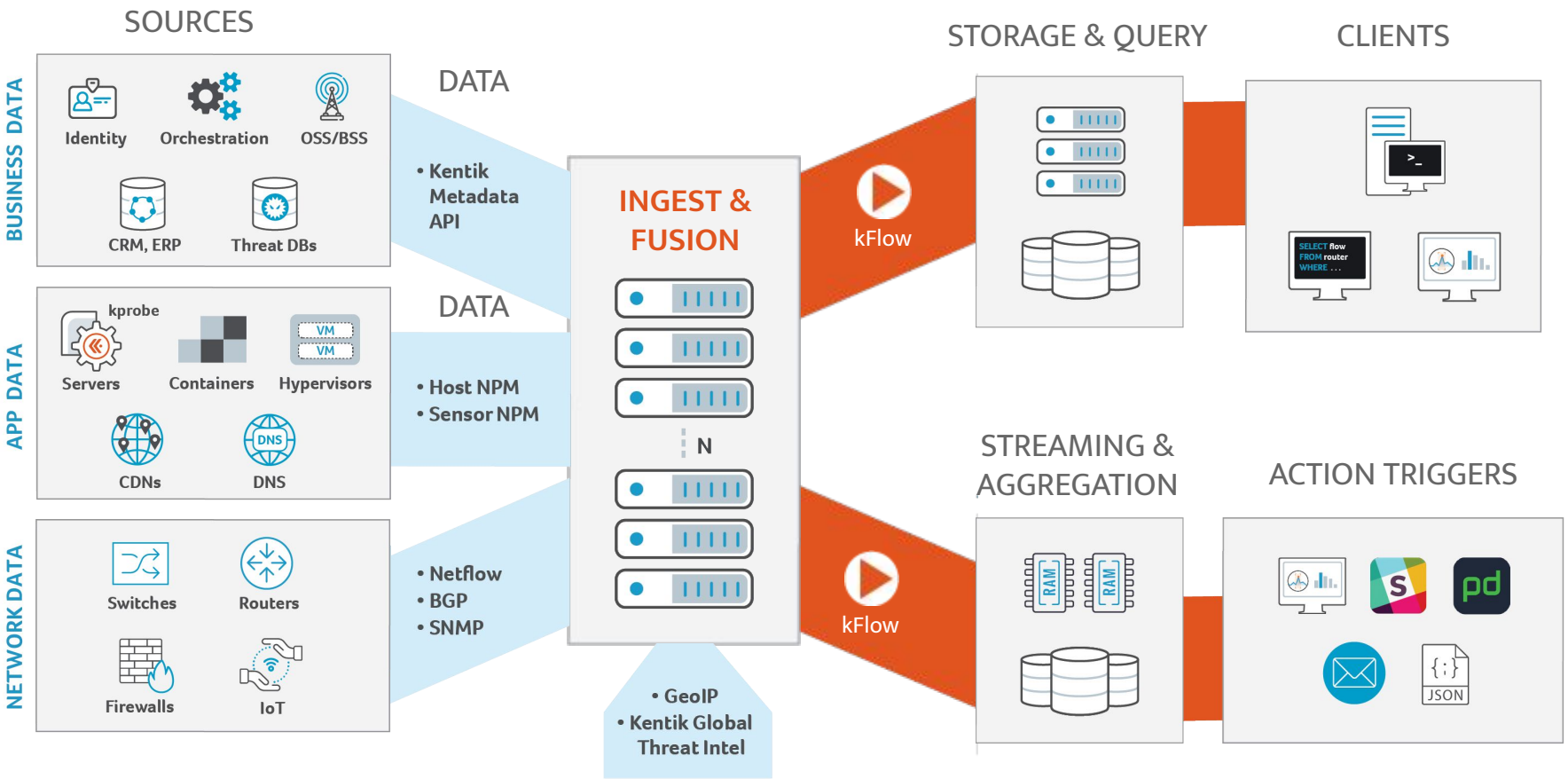
Costas Drogos <costasd@kentik.com>

kentik

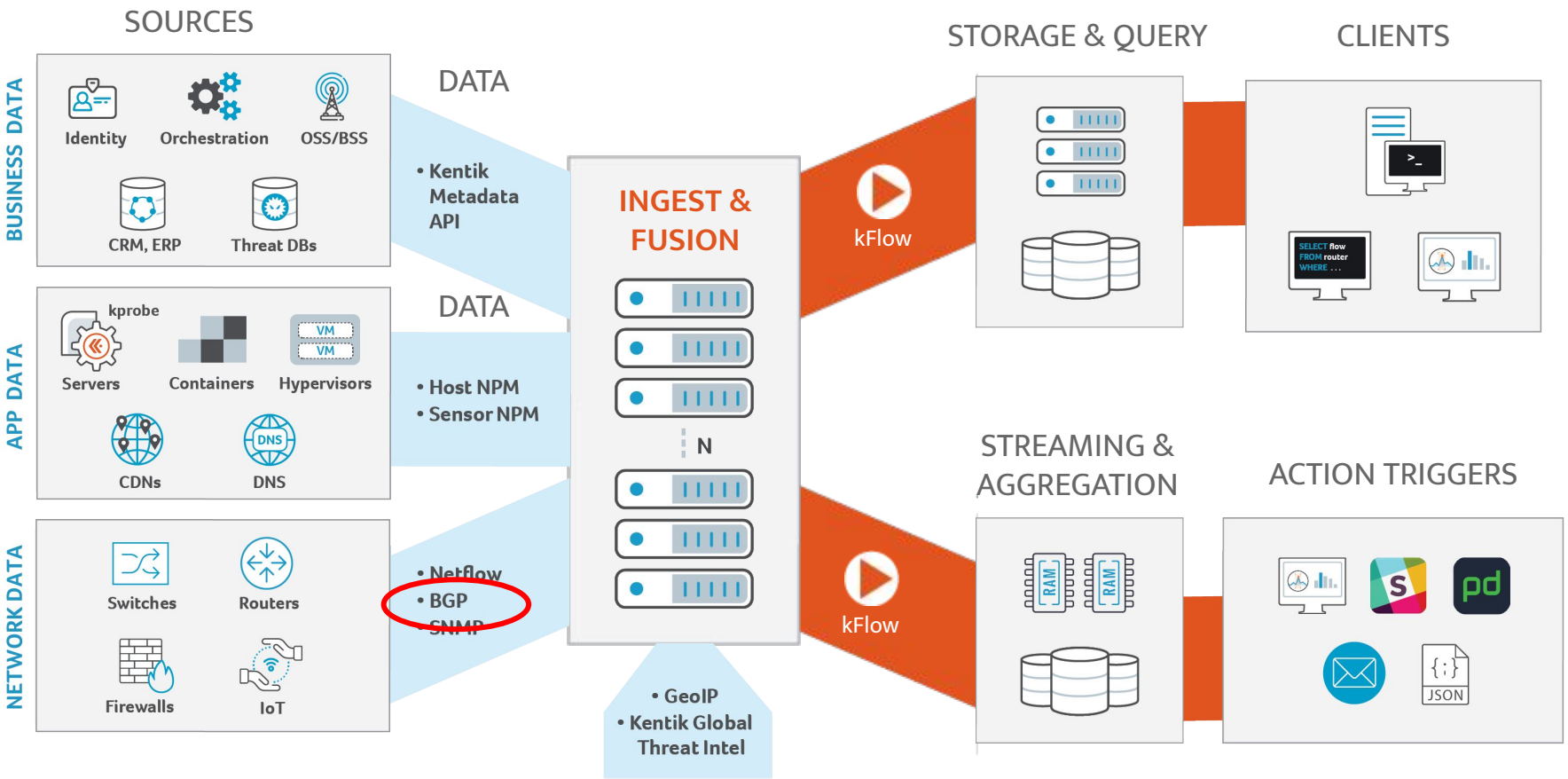
# Contents

- BGP at Kentik
- Scaling Phases
  - 1-2 nodes (active/backup)
  - 4-10 nodes with ‘bgp-vips’ and policy routing
  - 10-16 nodes with ‘vipcontrol’ and policy routing
  - 16+ nodes with LVS-DR

# Kentik Platform



# Kentik Platform



# BGP at Kentik: functionality

- Kentik performs peerings with Customers
  - Preferably with every device sending flow and running BGP for routing
  - Passive iBGP route-reflector speaker
    - On servers running Debian GNU/Linux
  - BGP functionality is part of our contracted SLA - 99.99%

Just for filtering by BGP attributes in queries?

# BGP at Kentik: functionality

- Kentik performs peerings with Customers
  - Preferably with every device sending flow and running BGP for routing
  - Passive iBGP route-reflector speaker
    - On servers running Debian GNU/Linux
  - BGP functionality is part of our contracted SLA - 99.99%

Just for filtering by BGP attributes in queries?

Analytics

- RPKI
- Peering Analytics
- Ultimate Exit calculation
- Network discovery

Alerting/Mitigations

- Prefix alerts
- RTBH
- Flowspec

# The beginning

- Less than 200 peers, IPv4 only
- 2 nodes in Active/Backup mode
  - Kentik BGP software runs in both nodes
  - A floating IP handles the HA/failover part
    - With uCARP
- A script in /root sets everything up on boot via rc.local

# Kentik grows

- Peerings don’t fit in one node anymore
  - more than 300 peers in a server
  - Memory and CPU intensive
- RTBH feature is released to customers
  - BGP peerings now play a more active role
- uCARP doesn’t scale well for 2+ nodes
  - Lots of corner cases, health checks
- Kentik’s internal fabric gets upgraded to 10G
  - But also supports OSPF & BGP now

# Kentik grows

- uCARP gets replaced by ‘bgp-vips’
  - Shell script talking to a spawned exaBGP
  - Floating BGP IP is now mounted on the loopback interface
  - Announced to Kentik’s BGP fabric with a different MED per node

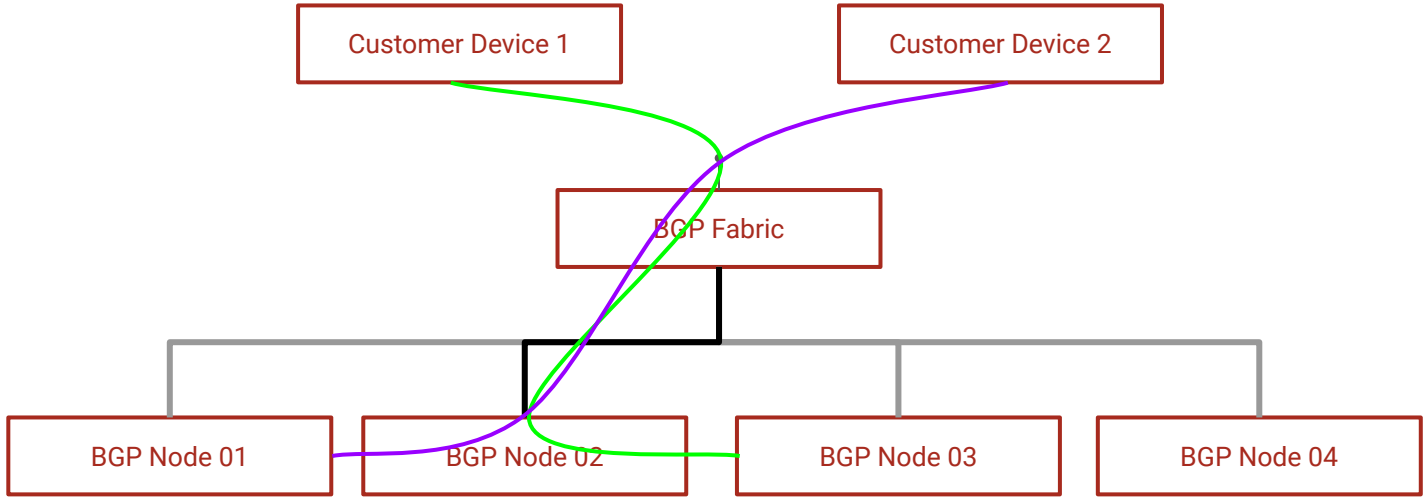
```
R1=$(( ( RANDOM % 10000 )  + 1 ))
STATE="down"
while true; do
  ....
  if [[ "$STATE" == "up" ]]; then
      echo "announce route 208.76.14.223/32 next-hop self med" $R1
  fi
  if [[ "$STATE" == "down" ]]; then
      echo "withdraw route 208.76.14.223/32 next-hop self med" $R1
  fi
  sleep 2
done
```

# Kentik grows

- Connections still land to a single node, but don’t fit there
- We have to offload connections to other nodes
  - So the ‘landing’ node has to act as a router of sorts
  - Enter policy routing, aka MARK and ip rule, ip route in the Linux world
  - How to mark them to achieve a balanced distribution?
    - Oldest trick in the book: wildcard masks

```
iptables -A PREROUTING -t mangle -p tcp -s 0.0.0.1/0.1.1.1 --sport 179 -d 208.76.14.223/32 -j
MARK --set-mark 100
iptables -A PREROUTING -t mangle -p tcp -s 0.0.1.0/0.1.1.1 --sport 179 -d 208.76.14.223/32 -j
MARK --set-mark 101
...
ip route add 208.76.14.223/32 via 1.1.1.2 table bgp-even
ip route add 208.76.14.223/32 via 1.1.1.3 table bgp-odd
ip rule add pri 29000 fwmark 100 table bgp-even
ip rule add pri 29000 fwmark 101 table bgp-odd
```

# BGP setup


Customer Device 1
Customer Device 2
BGP Fabric
BGP Node 01
BGP Node 02
BGP Node 03
BGP Node 04

# Issues

- Can’t find an ipv6 mask good enough for a uniform distribution
  - /32?
  - /48?
  - /64?
  - So IPv6 remains bound to 2 nodes (active/backup) for now :(
- Topology modification means full exaBGP restart

# Kentik grows more

- We've passed 1300 peers
  - IPv6 doesn't fit in one node anymore (yay!)
- Mask-based balancing not very optimal anymore
  - Lots of bigger device customers have devices in the same /24 or /26
- Various shortcomings with 'bgp-vips'
  - Can't modify MEDs without restarting the exabgp process
  - Need to support more complex health checks

But, not much time to design something from scratch, we grow fast!

# Kentik grows more

So, we decide to improve the current setup

- We keep policy routing, routing tables, routing rules
- But we replace mask-based routing with hash-based routing
  - from MARK to HMARK
  - https://lwn.net/Articles/488663/

```
iptables -A PREROUTING -t mangle -d ${BGPVIP1} -j HMARK --hmark-offset 100 --hmark-tuple src
--hmark-mod 10 --hmark-rnd 0xdeadbeef


ip6tables -A PREROUTING -t mangle -d ${BGP6VIP1}/128 -j HMARK --hmark-offset 200
--hmark-tuple src --hmark-mod 10 --hmark-rnd 0xdeadbeef
```

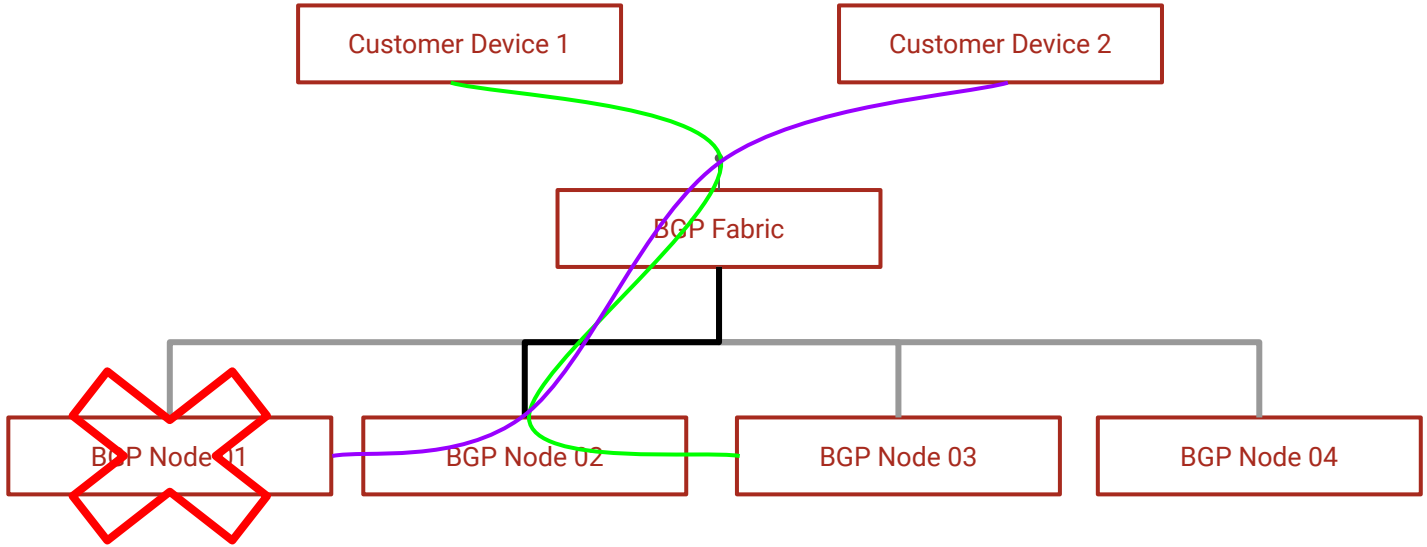
# Kentik grows more

- ...And we decide to replace ‘bgp-vips’ with ‘vipcontrol’
  - A real daemon in python communicating with a side-running exaBGP
  - Assigning random MEDs to configured IPs on startup
  - ...but also offering the ability to dynamically modify exaBGP’s state

Accompanied with a new ‘viphealth’ daemon that is able to dynamically modify MEDs.

```
$ sudo vipcontrolctl list
...
# DEBUG KEY: 208.76.14.223/32 VALUES: {'action': 'announce', 'med': 7854, 'details':
'Added by viphealth: 2019-07-19 18:35:26.414150'}
announce route 208.76.14.223/32 next-hop self med 7854
...
```

# What if a node needs to be removed?

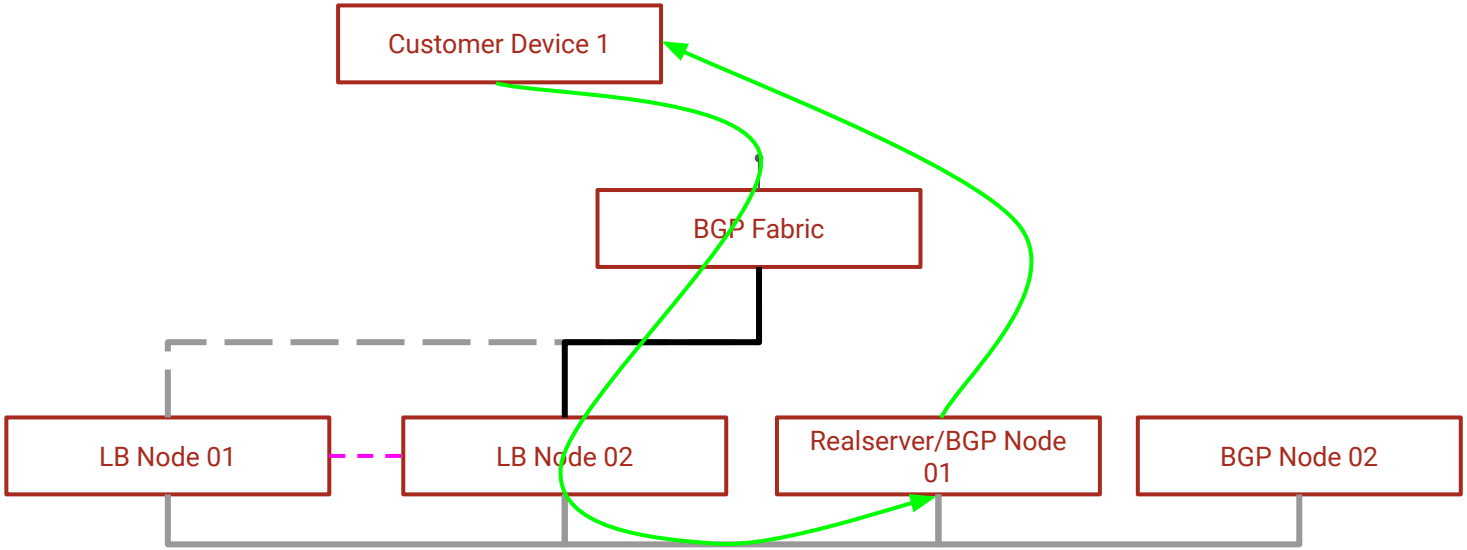


# Kentik grows much more

A new design to replace policy routing

- IPv4 & IPv6 support
- Uniform Session distribution
- Horizontal scaling
- All the state to be persisted in our Puppet code tree
- Easier day-to-day operations: pooling, depooling, code deploys
  - 4000+ sessions, we can’t sustain flapping them all for HW maintenance or A/B testing anymore

# LVS/DSR setup



# Under the hood

- Bird instead of exaBGP
  - +BFD for quicker route failover
- Keepalived in LVS mode
  - With healthchecks for pooling/depooling realservers
- Connection sync with ipvs
  - LB node flapping
  - Realserver flapping
- Configuration set in Puppet+git

# LVS/DR setup

- Pros
  - Online configuration changes are possible
  - Everything is persisted in configuration management
  - Ability to drain/depool/pool servers with minimal impact
  - Total health check flexibility
- Cons
  - No more consistent hashing
  - Can’t scale out of one LAN due to arp
  - All BGP connections pass through a pair of LB nodes

# Testing / Tuning

- Need to emulate thousands of BGP connections
  - Spotify's https://github.com/spotify/super-smash-brogp

- Lots of tuning potential
  - IPVS
    - expire timeouts, sync frequency/threshold,msg_max_size etc
  - Keepalived + Bird
    - notify_up/notify_down, quorum_up/quorum_down, interface tracking etc
  - BFD
    - timers

# Q&A

Thanks!