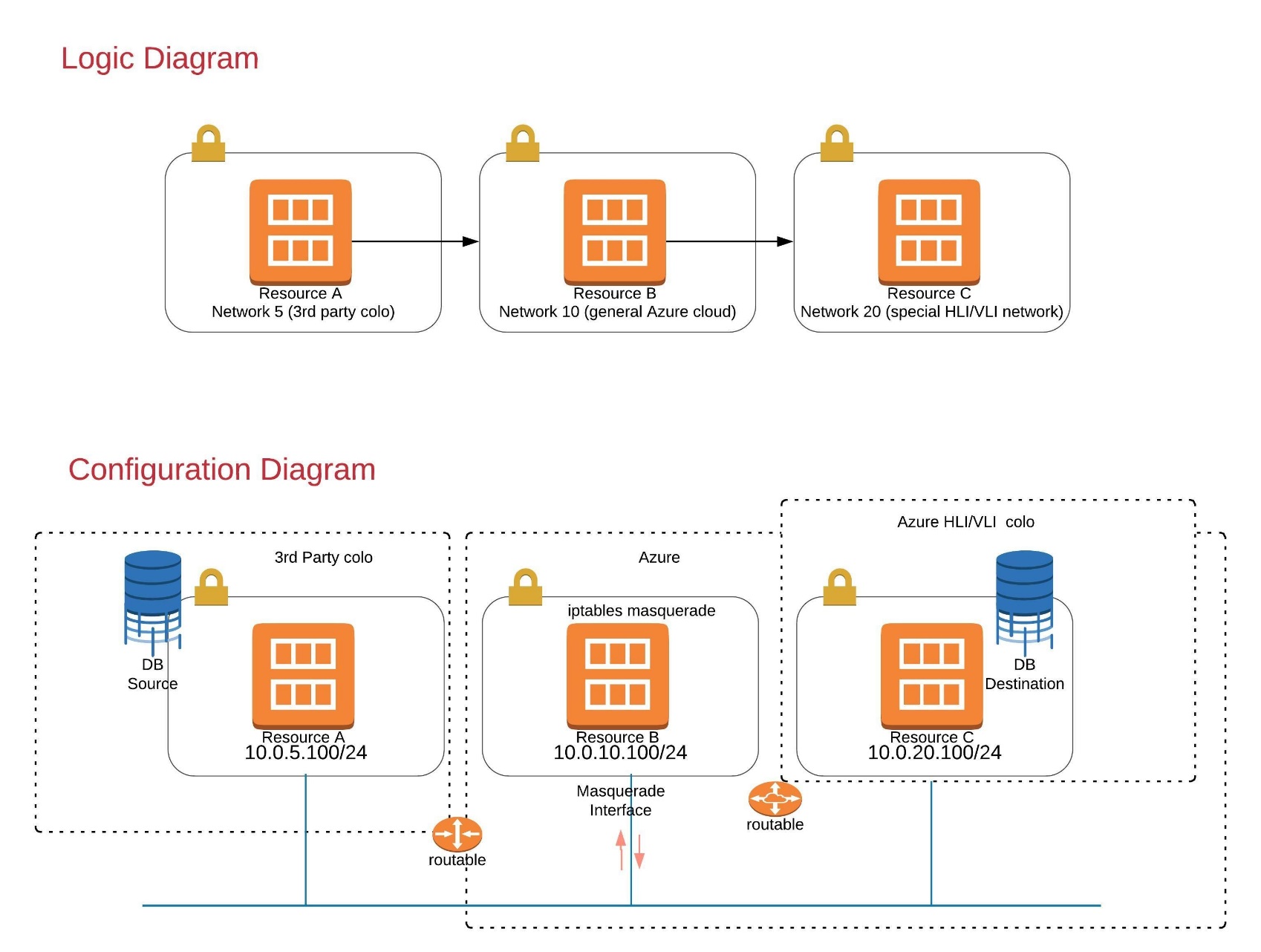
****

**Cloud configuration**

In real life, the hosts would be behind a NAT or firewall and be presented by an outside public interface as the sending source. Let’s simplify it. For lab purposes, let us assume that the following statements are true.

Route Tables, ACL’s, and Security Groups are configured to support the following.

* Node A represents the remote datacenter facility and migration source. An example might be a colocation facility in some public datacenter provider space (a cage.)
  + Node A resides in the 10.0.5.0/24 network
  + Node A can only reach Host B.
* Node B
  + Resides in the 10.0.10.0/24 network.
  + Node B can reach Nodes A and C. This becomes our Masquerade node.
* Node C
  + Resides in 10.0.20.0/24.
  + Can only reach node B.

**Considering performance**

Environmental requirement for this PoC: Node B has only a single NIC. All ingress and egress traffic between nodes will travers this single virtual 1GBps NIC. This was a condition in our set of requirements from the customer. The goal was an eventual 1-way migration of data.

This solution would have a maximum throughput rate of 125 MBps or less due to the single shared virtual NIC, barring any other QoS limitations against the VM instance or within the shared infrastructure.

Gigabit Ethernet runs up to 125 MBps (1,000 Mbps / 8 = 125 MBps.) Due to the nature of traffic flows on a virtual network device, throughput will likely be closer to half this amount, or 62MBps.

Latency due to localization, the software virtualization stack, congestion on shared infrastructure, or other factors will further affect throughput.

Other configurations are certainly possible, such as multiple bonded NIC’s or Enhanced 10GB or better networking.

All 3 SLES 12 sp3 instances were instantiated with default settings and without a desktop environment. Your installation options may vary depending upon the hardware or hosting environment.

**Let’s get it configured!**

**Verify routes on Node A**

**# route -n**

Kernel IP routing table

Destination     Gateway         Genmask         Flags Metric Ref    Use Iface

0.0.0.0         10.0.5.1        0.0.0.0         UG    0      0        0 eth0

10.0.5.0        0.0.0.0         255.255.255.0   U     0      0        0 eth0

**Verify routes, iptables version and firewall status on Node B**

**# route -n**

Kernel IP routing table

Destination     Gateway         Genmask         Flags Metric Ref    Use Iface

0.0.0.0         10.0.10.1       0.0.0.0         UG    0      0        0 eth0

10.0.10.0       0.0.0.0         255.255.255.0   U     0      0        0 eth0

**# iptables -V**

iptables v1.4.21

**#systemctl is-enabled SuSEfirewall2**

disabled

*(a future blog may cover the use of the SUSE firewall service as an alternate solution.)*

**Routes on Node C**

**# route -n**

Kernel IP routing table

Destination     Gateway         Genmask         Flags Metric Ref    Use Iface

0.0.0.0         10.0.20.1       0.0.0.0         UG    0      0        0 eth0

10.0.20.0       0.0.0.0         255.255.255.0   U     0      0        0 eth0

**Testing will be done from Node A and monitored on Nodes B and C. We recommend configuring password-less SSH logins from node A to node C. Alternatively you could enable username and password connections as illustrated below. It is not necessary to configure this credential on Node B.**

**Create a user on node C and enable password based authentication**

**# useradd -m -d /home/TestUser -c “IPtables Test” TestUser**

**# passwd TestUser**

**# sed -i ‘s|[#]\*PasswordAuthentication no|PasswordAuthentication yes|g’ /etc/ssh/sshd\_config**

**# cat /etc/ssh/sshd\_config |grep “PasswordAuthentication yes”**

PasswordAuthentication yes

**# systemctl restart sshd && systemctl status sshd |tail -5**

…

May 11 21:10:52 ip-10-0-20-100 systemd[1]: Started OpenSSH Daemon.

**All additional configuration is done on node B**

**Enable persistent IP forwarding**

**# echo “net.ipv4.ip\_forward=1” >> /etc/sysctl.d/99-sysctl.conf**

**# sysctl -p**

**# sysctl net.ipv4.ip\_forward**

net,ipv4.ip\_forward = 1

**Verify persistentence**

**# reboot**

**# sysctl net.ipv4.ip\_forward**

net,ipv4.ip\_forward = 1

**Configure iptables rules to NAT table and PREROUTING Chain to support SSH traffic from node A to Destination NAT of Node C**

**# iptables -t nat -A PREROUTING -p tcp -s 10.0.5.100 –dport 40022 -j DNAT –to-destination 10.0.20.100:22**

**Enable the Masquerade function for the nay table on the POSTROUTING chain**

**# iptables -t nat -A POSTROUTING -j MASQUERADE**

**Configure Node B to survive reboots.**

**#iptables-save > /var/iptables.save**

File looks similar to this.

**# cat /var/iptables.save**

# Generated by iptables-save v1.4.21 on Fri May 11 19:13:24 2018

\*nat

:PREROUTING ACCEPT [0:0]

:INPUT ACCEPT [0:0]

:OUTPUT ACCEPT [1:76]

:POSTROUTING ACCEPT [0:0]

-A PREROUTING -s 10.0.5.100/32 -p tcp -m tcp –dport 40022 -j DNAT –to-destination 10.0.20.100:22

-A POSTROUTING -j MASQUERADE

COMMIT

**# echo “iptables-restore /var/iptables.save” >> /etc/init.d/after.local**

**# cat /etc/init.d/after.local |grep “**iptables-restore /var/iptables.save”

iptables-restore /var/iptables.save

**SSH test from Node A to Node B**

(forwarded to node C) The “TestUser” only exists on node C

**# ssh TestUser@10.0.10.100 -p 40022**

Have a lot of fun…

TestUser@ip-10-0-20-100:~>

**View rule hits on Node B – Example:**

# iptables -t nat -L -n -v

Chain PREROUTING (policy ACCEPT 3 packets, 180 bytes)

pkts bytes target     prot opt in     out     source               destination

0     0 DNAT       tcp  —  \*      \*       10.0.5.100           0.0.0.0/0            tcp dpt:40022 to:10.0.20.100:22

…

6   672 MASQUERADE  all  —  \*      \*       0.0.0.0/0            0.0.0.0/0

**On Node C, confirm connection source using netstat and tcpdump**

We should see the SSH connection as coming from Node B. You may have to refresh your SSH session to see additional output.

**# tcpdump -i eth0 src 10.0.10.100 -vv**

20:12:18.482889 IP (tos 0x10, ttl 63, id 25893, offset 0, flags [DF], proto TCP (6), length 52)

10.0.10.100.58056 > 10.0.20.100.ssh: Flags [.], cksum 0x2432 (correct), seq 2693, ack 3150, win 272, options [nop,nop,TS val 216308186 ecr 326450094], length 0

**# netstat -tan | grep “:22\|ssh” | grep -v “LISTEN” | awk ‘{print ”     “$4” -> “$5”      “$6}’**

10.0.20.100:22 -> 66.235.19.xxx:51180      ESTABLISHED  (this is the admin workstation)

10.0.20.100:22 -> 10.0.10.100:58058      ESTABLISHED  (this is node B)

**Complete and test**

Reboot all Nodes and repeat tests to verify configuration.

Next Documentation Steps: Create a more elegant solution with multiple high speed NIC’s or pass through tunneling.

Future Example Use Case:  High Speed Bi-directional replication for DR/BC